

Decision Trees

Samkeun Kim <skim@hknu.ac.kr>

<http://cyber.hankyong.ac.kr>

Decision Trees (DT):

- Classification과 Regression 둘 다를 잘 수행하는 다재 다능한 ML 알고리즘
- 복잡한 데이터셋을 적합시킬 수 있는 강력한 알고리즘
- Random Forests 알고리즘(현존하는 가장 강력한 ML 알고리즘)의 기본 컴포넌트

DT를 학습시키고, 시각화하고, 예측을 수행하는 방법을 살펴본다.

Scikit-Learn의 CART 학습 알고리즘을 실습해본다.

Training and Visualizing a Decision Tree

DT를 이해하기 위해 ... DT 구축해서 예측 수행:

- DecisionTreeClassifier on the Iris dataset:

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier

iris = load_iris()
X = iris.data[:, 2:] # petal length and width
y = iris.target

tree_clf = DecisionTreeClassifier(max_depth=2)
tree_clf.fit(X, y)
```

학습된 Decision Tree를 그래프로 보기 (`export_graphviz()` 메소드를 이용하여)

- `export_graphviz()` 메소드 => `iris_tree.dot`라는 그래프 정의 파일 출력

```
from sklearn.tree import export_graphviz

export_graphviz(
    tree_clf,
    out_file=image_path("iris_tree.dot"),
    feature_names=iris.feature_names[2:],
    class_names=iris.target_names,
    rounded=True,
    filled=True
)
```

- `.dot` 파일을 PNG 파일로 변환하기 위해 Graphviz 패키지의 `dot` 커맨드 사용 (다음 페이지 참조)

Graphviz 설치

Graphviz: <http://www.graphviz.org/>

Graphviz 2.38 Stable Release: https://graphviz.gitlab.io/_pages/Download/Download_windows.html

시작 > Graphviz 2.38 > **gvedit.exe**

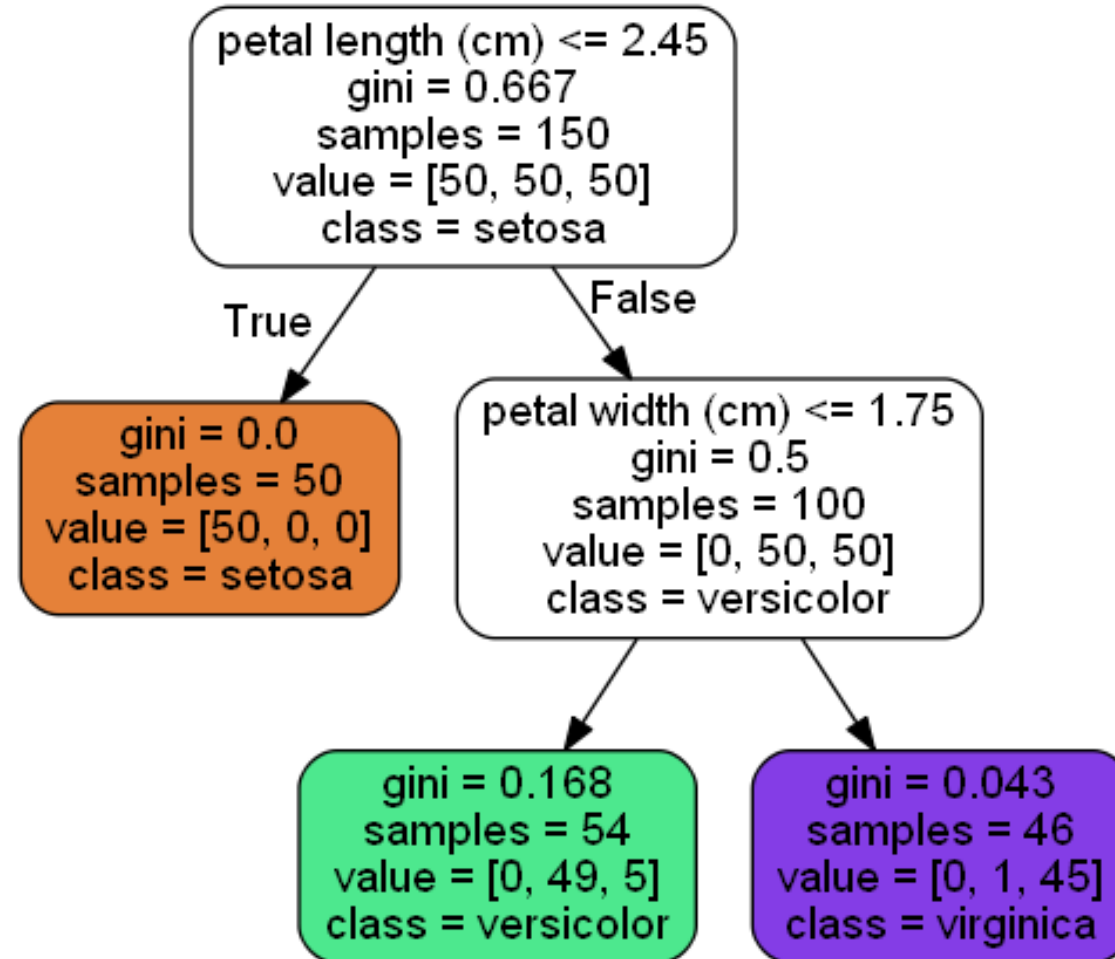
dot 파일 위치: F:\practice\ai\handson(pycharm)\images\decision_trees

또는 DOS Prompt에서:

```
(aisam) F:\practice\ai\handson(pycharm)\images\decision_trees>dot -Tpng iris_tree.dot -o iris_tree.png
```

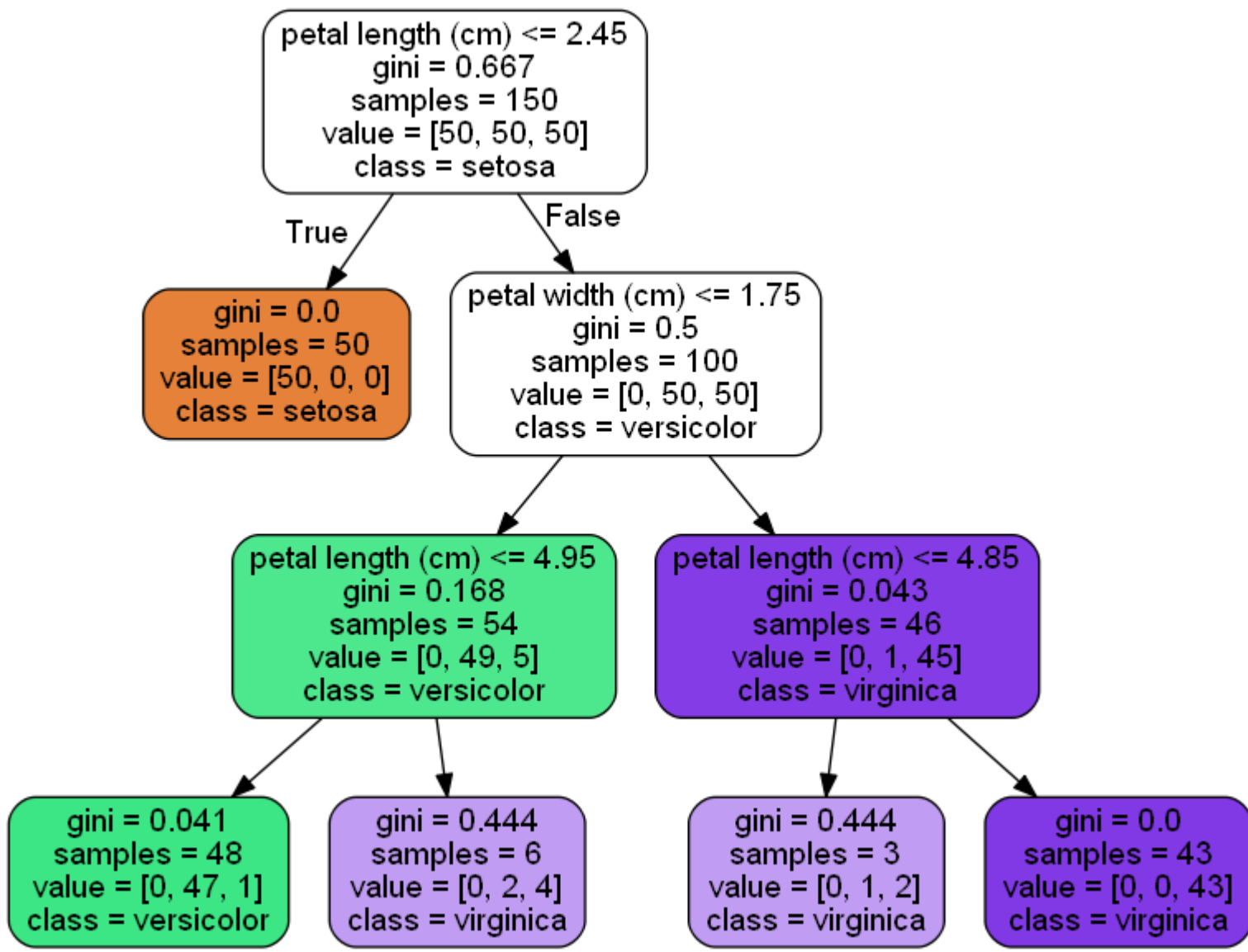
Training and Visualizing a Decision Tree

Iris Decision Tree (max_depth = 2)



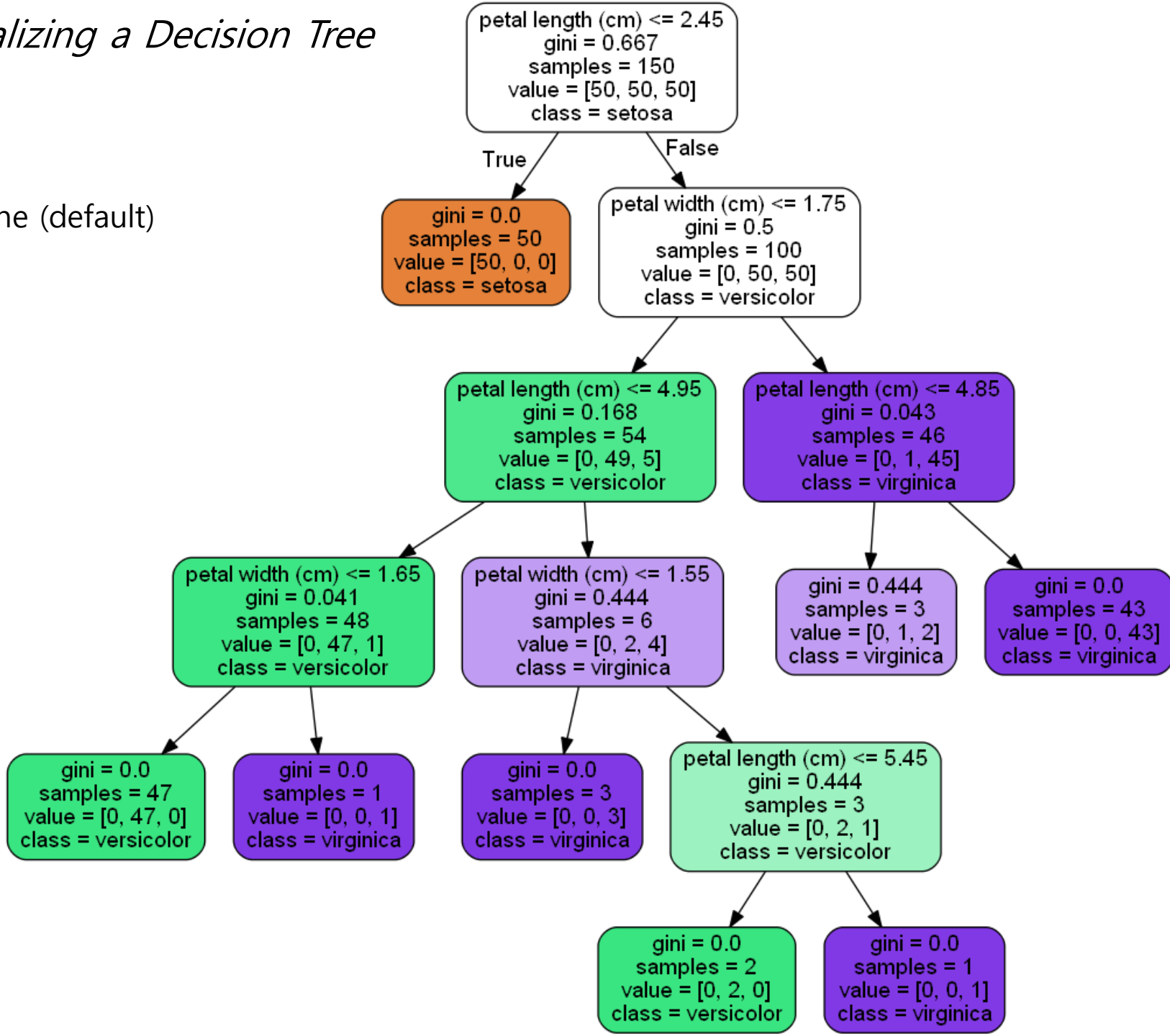
Training and Visualizing a Decision Tree

max_depth = 3



Training and Visualizing a Decision Tree

max_depth = None (default)

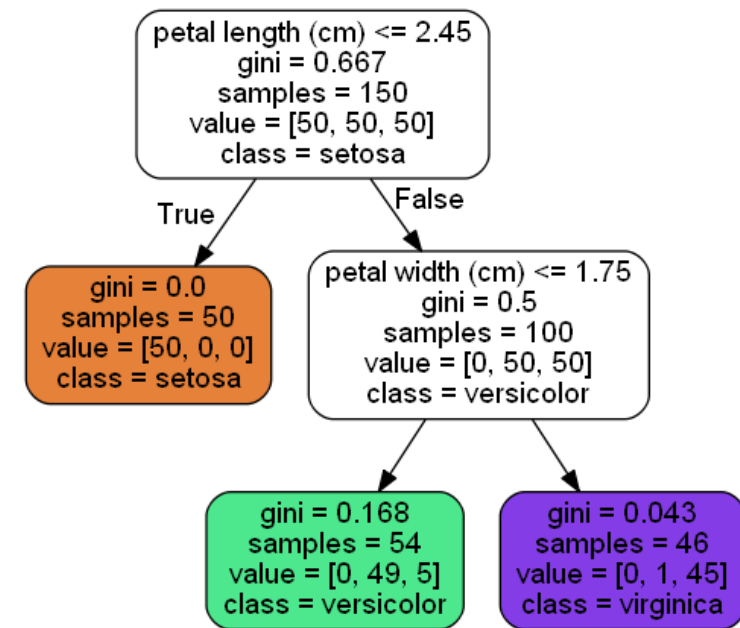


Making Predictions

DT 예측 방법 => Iris 꽃을 발견 ... 분류해보자:

Root 노드(depth 0)에서 출발:

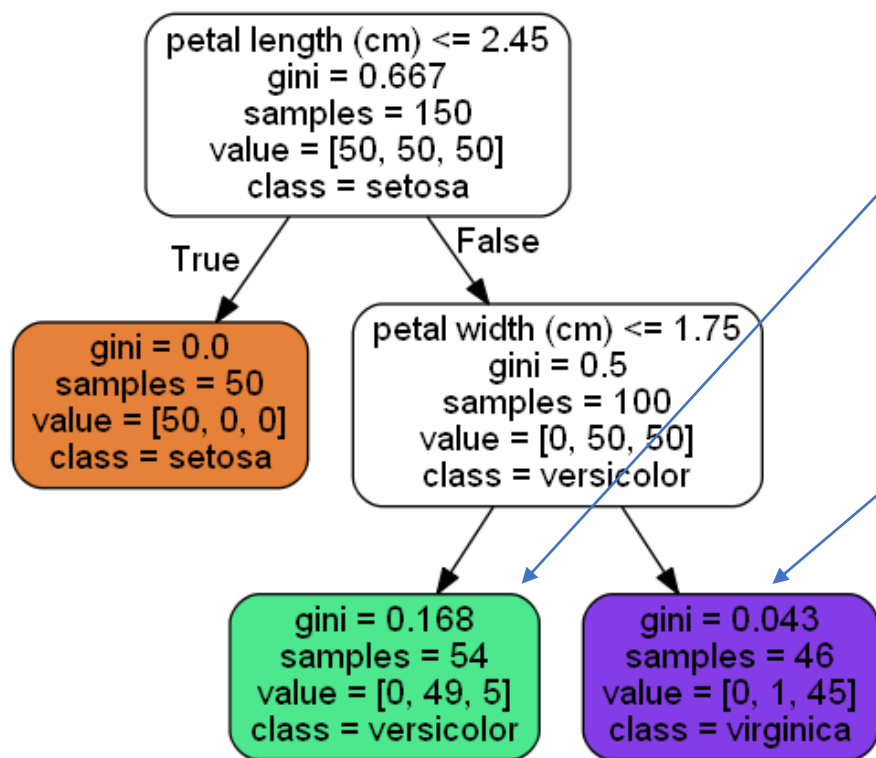
- 꽃잎(petal) 길이가 2.45cm보다 작은가?
 - ✓ 작다면 root의 왼쪽 child(depth 1, left)로 이동
 - ✓ 노드가 leaf 노드(더 이상의 child 노드를 갖지 않음) => 해당 노드에서 예측된 클래스 발견
 - ✓ DT => 꽃이 *Iris setosa* 라고 예측
- 꽃의 petal 길이가 2.45cm보다 큰가?
 - ✓ 크다면 root의 오른쪽 child(depth 1, right)로 이동
 - ✓ Leaf node가 아님. 꽃잎 폭이 1.75cm보다 작은가?
 - ✓ 작다면 *Iris versicolor* (depth 2, *Iris versicolor*), 그렇지 않으면 *Iris virginica* (depth 2, right)



Decision Tree:

특성 scaling이 필요 없음!

Making Predictions



노드의 samples 속성: 얼마나 많은 인스턴스가 포함되어 있는가를 카운트

- 2.45cm보다 더 큰 petal 길이를 갖는 인스턴스가 100개 있다 (depth 1, right)
- 100개 중에서 54개는 1.75cm보다 더 작은 petal 폭을 갖는다 (depth 2, left)

노드의 value 속성: 각 클래스의 인스턴스가 해당 노드에 얼마나 포함되어 있나

- Bottom-right 노드 => 0 Iris setosa, 1 Iris versicolor, 45 Iris virginica

노드의 gini 속성: **impurity**(불순도) 측정

모든 인스턴스가 동일 클래스에 속하면 해당 노드를 "pure" (gini=0)라고 함

Depth-1 left 노드: 오로지 Iris setosa 인스턴스만 있음 (gini=0)

Depth-2 left 노드: ***gini score*** $\Rightarrow 1 - \left(\frac{0}{54}\right)^2 - \left(\frac{49}{54}\right)^2 - \left(\frac{5}{54}\right)^2 \approx 0.168$

Gini impurity:

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

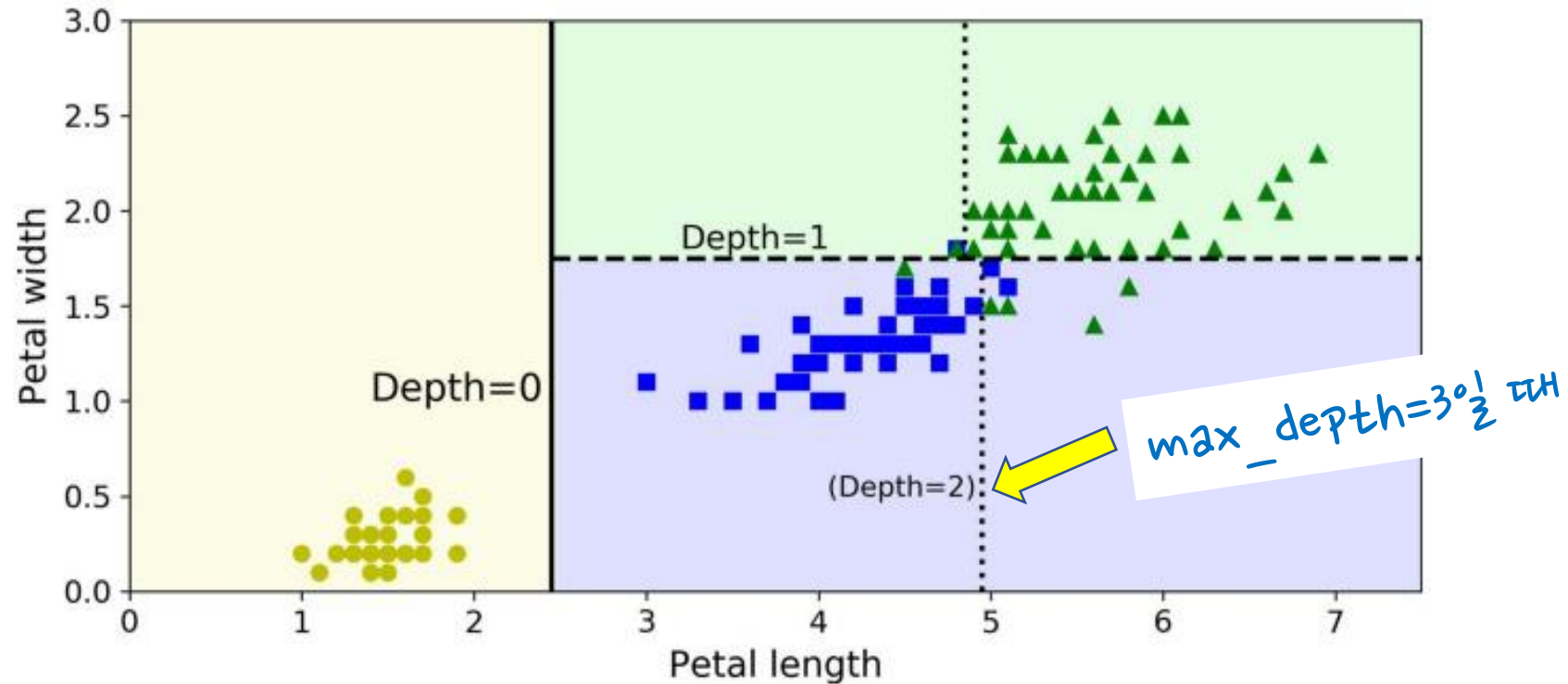
- $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node.

Scikit-Learn CART 알고리즘 이용 => 오로지 binary 트리만 생성

- Nonleaf 노드 => 항상 2개의 children만 가짐
- ID3 알고리즘 => 2개 이상의 children을 갖는 Decision Tree 생성 가능

Making Predictions

Decision Tree decision boundaries:



- Root 노드(depth 0)의 decision boundary (굵은 실선): petal length = 2.45cm.
 - ✓ 왼쪽 영역은 pure(only Iris setosa)이기 때문에 더 이상 분할되지 않음
- Depth-1 오른쪽 노드의 decision boundary (dashed line): petal width = 1.75cm
- max_depth=2로 설정되어 있기 때문에 DT stop!

Estimating Class Probabilities

Decision Tree => 어떤 인스턴스가 특정 클래스 k 에 속하는 확률을 추정할 수 있다.

- 우연히 발견한 Iris 꽃이 꽃잎 길이가 5cm, 폭이 1.5cm 였다고 하자:
- 해당 leaf 노드 => depth-2 left 노드

확률 계산 => 0% for *Iris setosa* (0/54), 90.7% for *Iris versicolor* (49/54), and 9.3% for *Iris virginica* (5/54)

& 반드시 클래스를 예측하라고 하면:

```
>>> tree_clf.predict_proba([[5, 1.5]])  
array([[0.          , 0.90740741, 0.09259259]])  
>>> tree_clf.predict([[5, 1.5]])  
array([1])
```

The CART Training Algorithm


Scikit-Learn은 Decision Tree를 학습시키기 위해 Classification and Regression Tree (CART) 알고리즘 사용

- Training set을 1개의 특성 k 와 임계 값 t_k 를 이용하여 2개의 서브 세트로 분리
- How does it choose k and t_k ?
 - ✓ 가장 순수한(purist) 서브 세트를 생성하는 쌍 (k, t_k) 을 탐색

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

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


Maximum depth까지 또는 더 이상 impurity를 줄일 수 없을 때까지 반복!!

Regularization Hyperparameters

Overfitting을 피하기 위해 Decision Tree의 자유도 제한

- Max_depth를 줄이면 => 모델 정규화 => Overfitting 위험 줄임

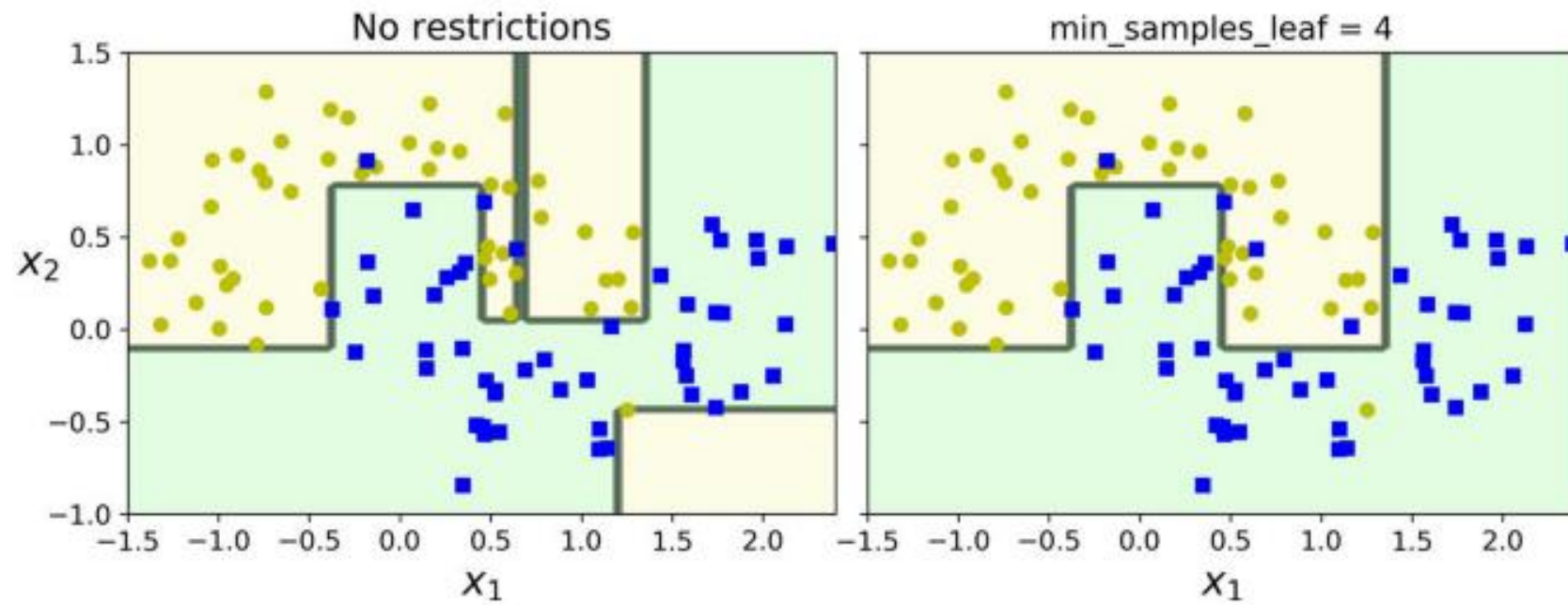
DecisionTreeClassifier 클래스 다른 파라미터들:

`min_samples_split` (the minimum number of samples a node must have before it can be split),
`min_samples_leaf` (the minimum number of samples a leaf node must have),
`min_weight_fraction_leaf` (same as `min_samples_leaf` but expressed as a fraction of the total number of weighted instances), `max_leaf_nodes` (the maximum number of leaf nodes), and
`max_features` (the maximum number of features that are evaluated for splitting at each node).

Making Predictions

Two Decision Trees trained on the moons dataset

- `min_samples_leaf=4`



Overfitting

Better generalization

Regression

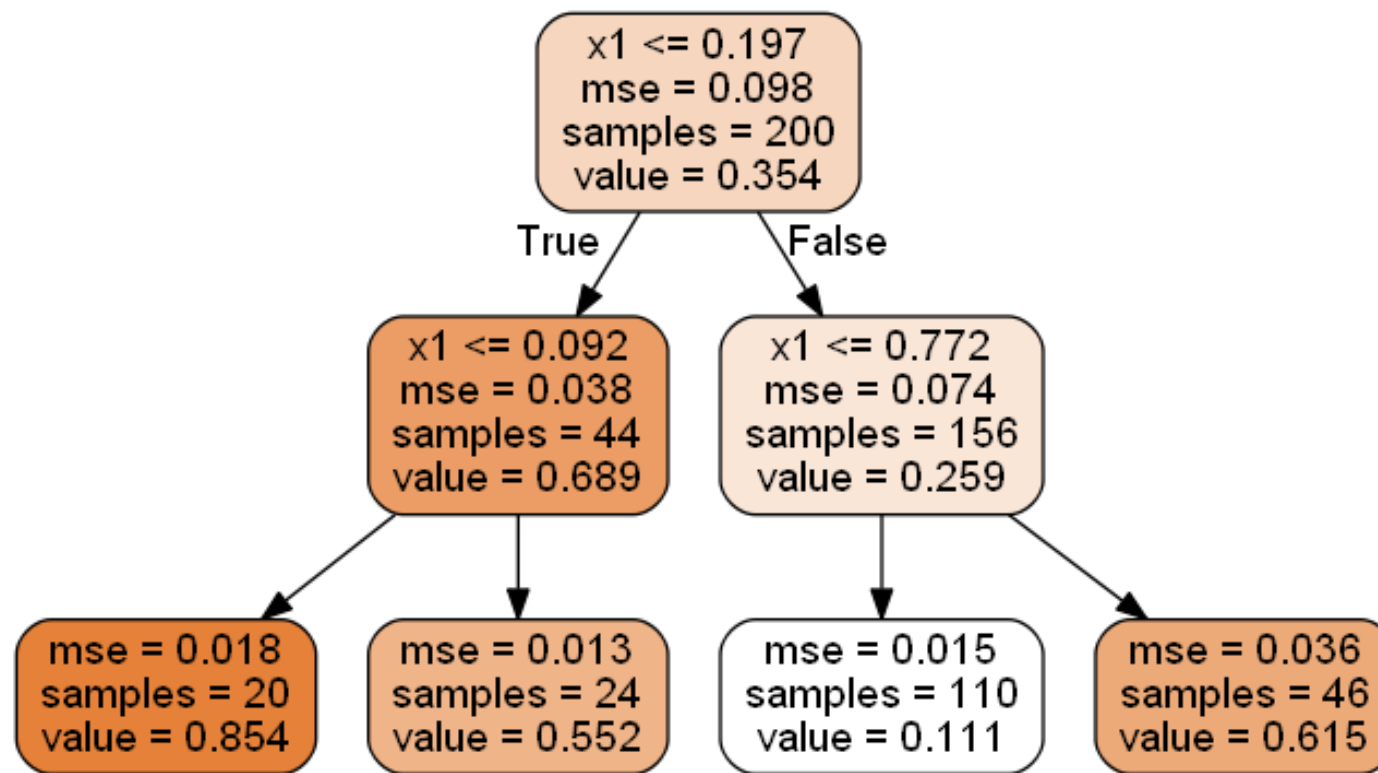
Decision Tree => Regression 문제에도 적용 가능

- Scikit-Learn의 `DecisionTreeRegressor` 클래스를 이용하여 regression 트리 생성

```
from sklearn.tree import DecisionTreeRegressor
```

```
tree_reg = DecisionTreeRegressor(max_depth=2)  
tree_reg.fit(X, y)
```

```
m = 200  
X = np.random.rand(m, 1)  
y = 4 * (X - 0.5) ** 2  
y = y + np.random.randn(m, 1) / 10
```



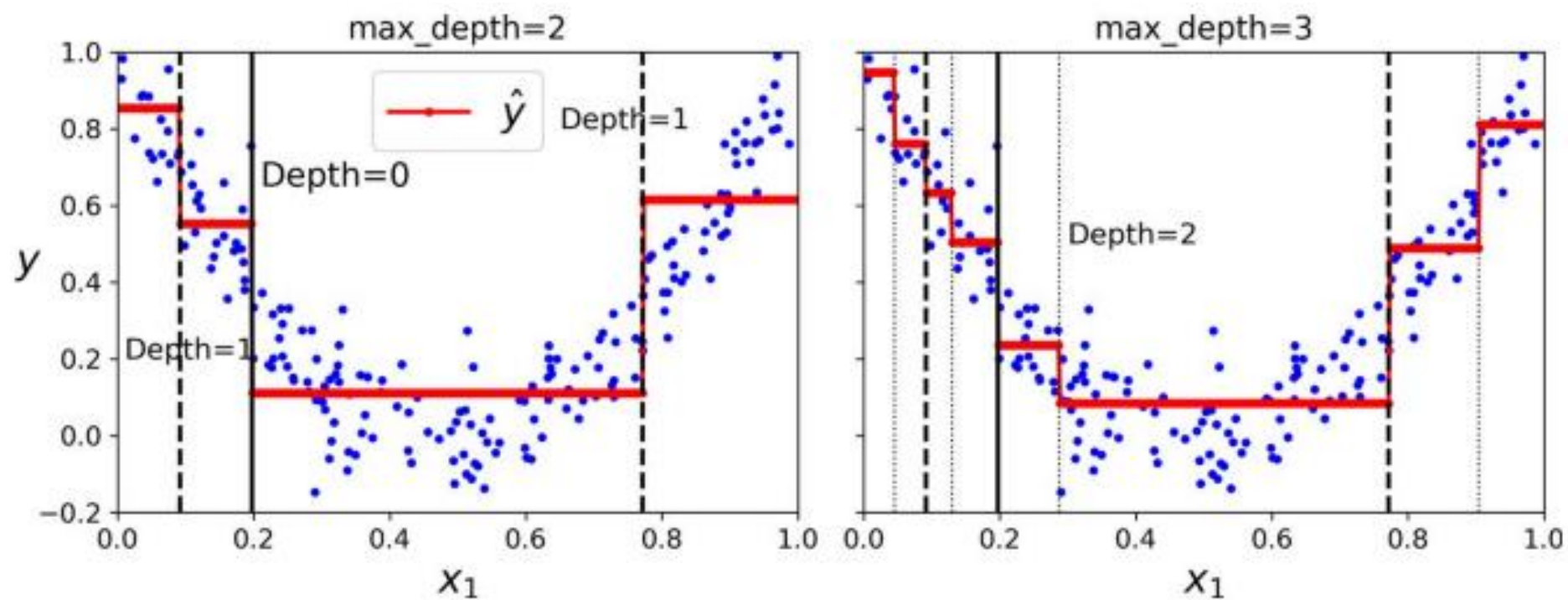
Classification 트리와의 기본 차이점 => 각 노드에서 클래스를 예측하는 것이 아니라 값을 예측!

예) $x_1 = 0.6$ 값을 갖는 새 인스턴스에 대해 예측을 해보자.

- 루트에서 시작하여 결국은 $value = 0.111$ 을 예측하는 leaf 노드에 도달한다.
- 이 예측 값은 해당 leaf 노드와 연관된 110개의 인스턴스들의 평균 타깃 값이다.
- 110개의 인스턴스에 대한 MSE가 0.015가 된다.

앞 모델의 예측:

- (max_depth=2)와 (max_depth=3)일 경우



CART 알고리즘 => MSE를 최소화하는 방향으로 training set을 분리

- CART cost function for regression

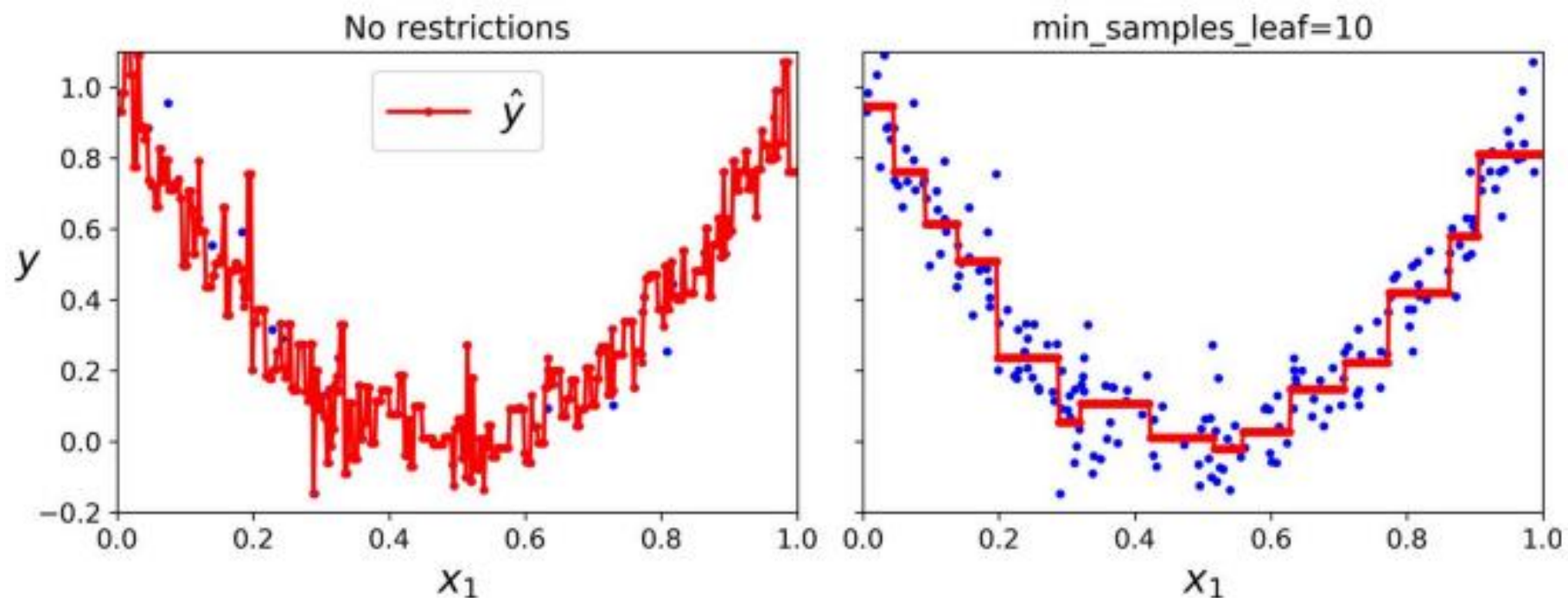
$$J(k, t_k) = \frac{m_{\text{left}}}{m} \text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m} \text{MSE}_{\text{right}} \quad \text{where} \quad \begin{cases} \text{MSE}_{\text{node}} = \sum_{i \in \text{node}} (\hat{y}_{\text{node}} - y^{(i)})^2 \\ \hat{y}_{\text{node}} = \frac{1}{m_{\text{node}}} \sum_{i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

- Classification에서와 마찬가지로 overfitting 문제 있음

어떤 정규화 과정도 수행하지 않으면(디폴트 hyperparameter를 사용한다면) 아래 그림(왼쪽)처럼 예측 결과:

- Overfitting 문제를 피하기 위해 hyperparameter(`min_samples_leaf=10`) 추가(오른쪽 그림)

Regularizing a Decision Tree regressor:



실습과제 09-1

본문에 나오는 전체 내용 PyCharm에서 실행하기

★ 반드시 결과 값 및 결과로 나온 모든 그래프에 대해 자세한 분석을 하시오.

참고: 제 09강 실습과제 #09 Decision Trees - CART Training Algorithm, Regression.pdf

실습과제 09-2

Train and fine-tune a Decision Tree for the moons dataset by following these steps:

- Use `make_moons(n_samples=10000, noise=0.4)` to generate a moons dataset.
- Use `train_test_split()` to split the dataset into a training set and a test set.
- Use grid search with cross-validation (with the help of the `GridSearchCV` class) to find good hyperparameter values for a `DecisionTreeClassifier`. Hint: try various values for `max_leaf_nodes`.
- Train it on the full training set using these hyperparameters, and measure your model's performance on the test set. You should get roughly 85% to 87% accuracy.

★ 반드시 결과 값 및 결과로 나온 모든 그래프에 대해 자세한 분석을 하시오.

참고: [제 09강 실습과제 09-2 Decision Tree for the moons dataset.pdf](#)

실습과제 09-3

Grow a forest:

- Continuing the previous exercise, generate 1,000 subsets of the training set, each containing 100 instances selected randomly. Hint: you can use Scikit-Learn's `ShuffleSplit` class for this.
- Train one Decision Tree on each subset, using the best hyperparameter values found above. Evaluate these 1,000 Decision Trees on the test set. Since they were trained on smaller sets, these Decision Trees will likely perform worse than the first Decision Tree, achieving only about 80% accuracy.
- Use grid search with cross-validation (with the help of the `GridSearchCV` class) to find good hyperparameter values for a `DecisionTreeClassifier`. Hint: try various values for `max_leaf_nodes`.
- Evaluate these predictions on the test set: you should obtain a slightly higher accuracy than your first model (about 0.5 to 1.5% higher). Congratulations, you have trained a Random Forest classifier!

★ 반드시 결과 값 및 결과로 나온 모든 그래프에 대해 자세한 분석을 하시오.

참고: [제 09강 실습과제 09-3 Grow a forest.pdf](#)

