# Check-in

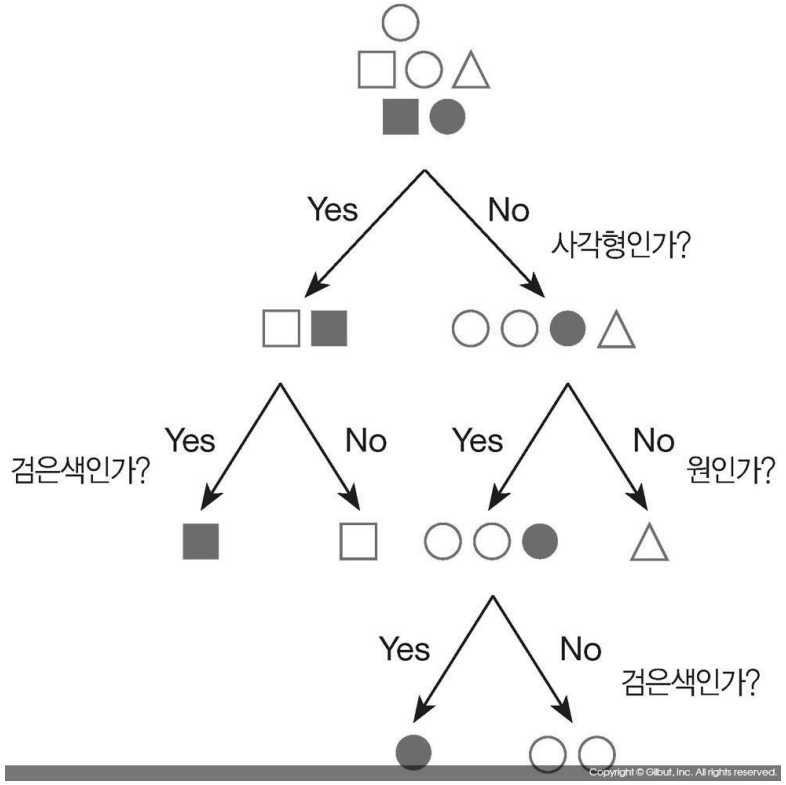
바깥 상황이 어려운데
요즘 어떤 취미를 즐기고 있나요?

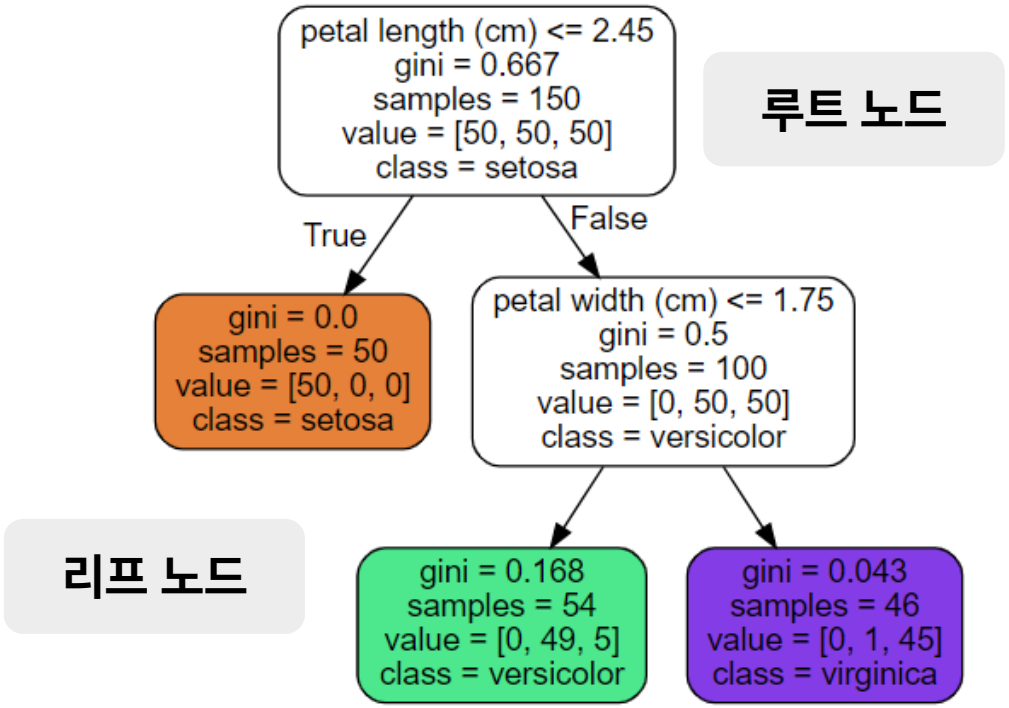
# Decision Tree

## 의사결정나무

# 의사결정 나무란?

의사결정에 필요한 규칙을 나무 형태로 분류해 나가는 분석기법으로 분류모델과 회귀모델이 존재



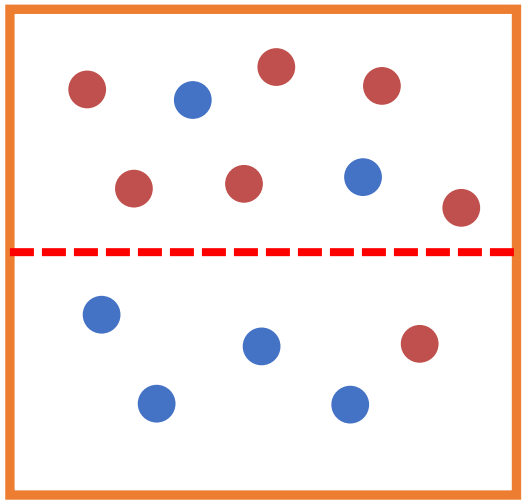




**루트 노드**

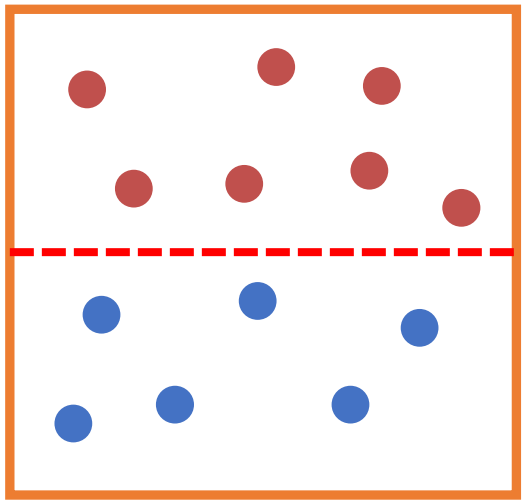
**리프 노드**

# 의사결정 나무의 기준

불순도(Impurity) : 하나의 범주 내에서 서로 다른 데이터가 얼마나 섞여있는지를 나타내는 지표



점선을 기준으로 빨간점과 파란점이
완벽하게 분리되지 않음



점선을 기준으로 빨간점과 파란점이
완벽하게 분리됨

# 의사결정 나무의 기준

## ① 엔트로피(entropy)

- 특정 공간의 무질서도

- 엔트로피의 감소는 불순도의 감소이다.

$$Entropy(A) = -\sum_{k=1}^{m} p_k \log_2 (p_k)$$

## ② 지니 지수(Gini index)

- 자료의 분산정도를 정량화 한 값

- 지니 지수의 감소는 불순도의 감소이다.

$$G.I(A) = \sum_{i=1}^{d} \left( R_i \left( 1 - \sum_{k=1}^{m} p_{ik}^2 \right) \right)$$

# 의사결정 나무_ iris data

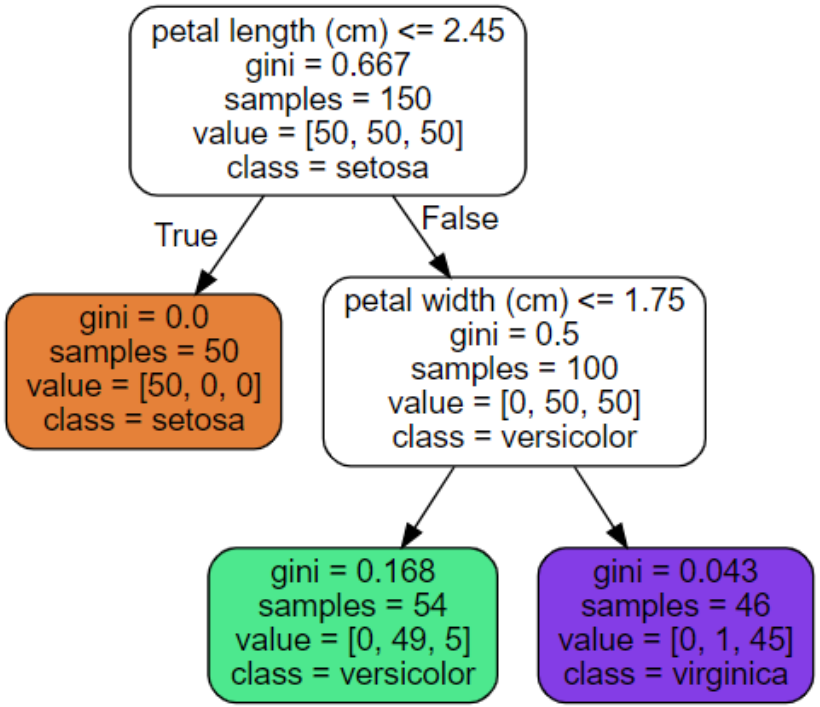
```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier

iris = load_iris()
X = iris.data[:, 2:]
y = iris.target
```

> X : 독립변수_ 꽃잎의 길이와 너비
> y : 종속변수_ 종이름(범주형)
>
> 꽃잎의 길이와 너비를 이용하여 종속변수(종이름)를 분류하는 의사결정나무를 생성할 수 있다.

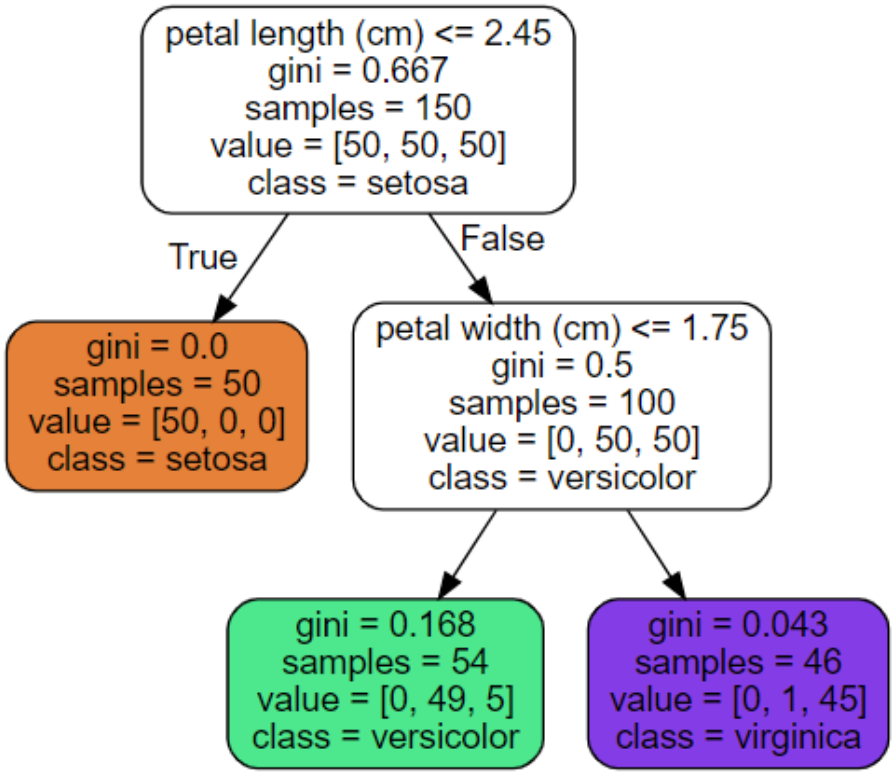
```
tree_clf = DecisionTreeClassifier(max_depth=2)
tree_clf.fit(X,y)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=2,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

```
from sklearn.tree import export_graphviz
from graphviz import Source

export_graphviz(tree_clf,                              # 모델
                out_file= "iris_tree1.dot",  # 저장경로 설정
                feature_names=iris.feature_names[2:],   # 변수명
                class_names=iris.target_names,         # 종속변수
                rounded = True,
                filled = True)

Source.from_file("iris_tree1.dot")
```



# 의사결정 나무_ iris data

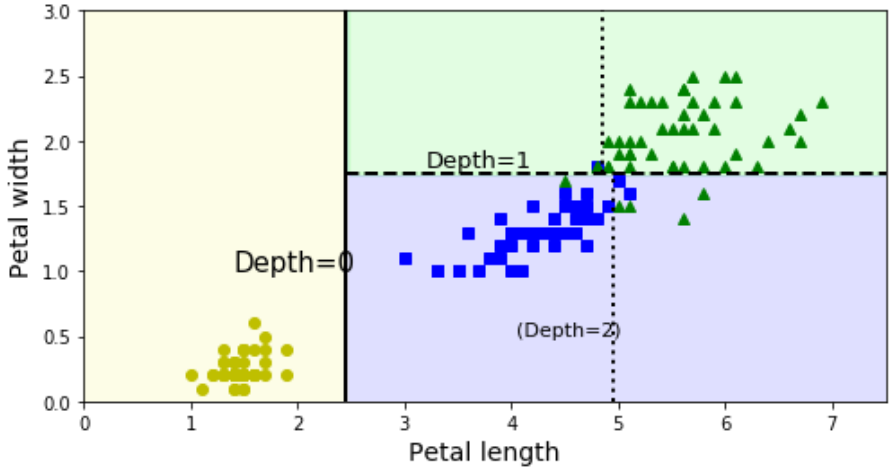



## 결정트리의 변수 설명

- gini : 불순도
- samples : 해당 노드에 적용된 훈련 샘플의 수
- value : 각 클래스에 해당되는 훈련 샘플의 수
- class : 해당 노드의 클래스

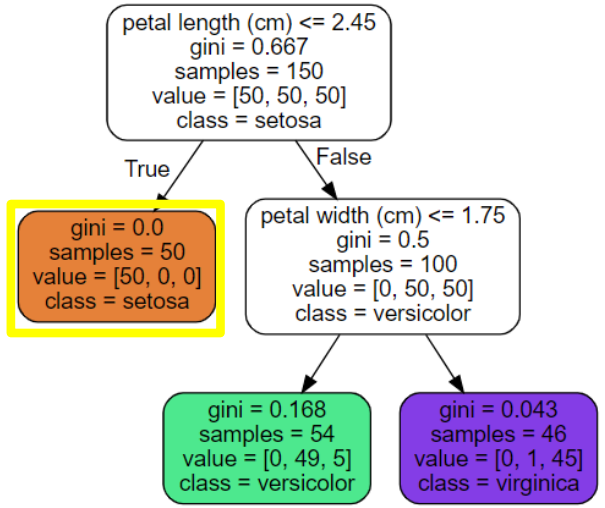
setosa 클래스 노드는 gini=0이기 때문에 순수한 노드

# 의사결정 나무_ iris data

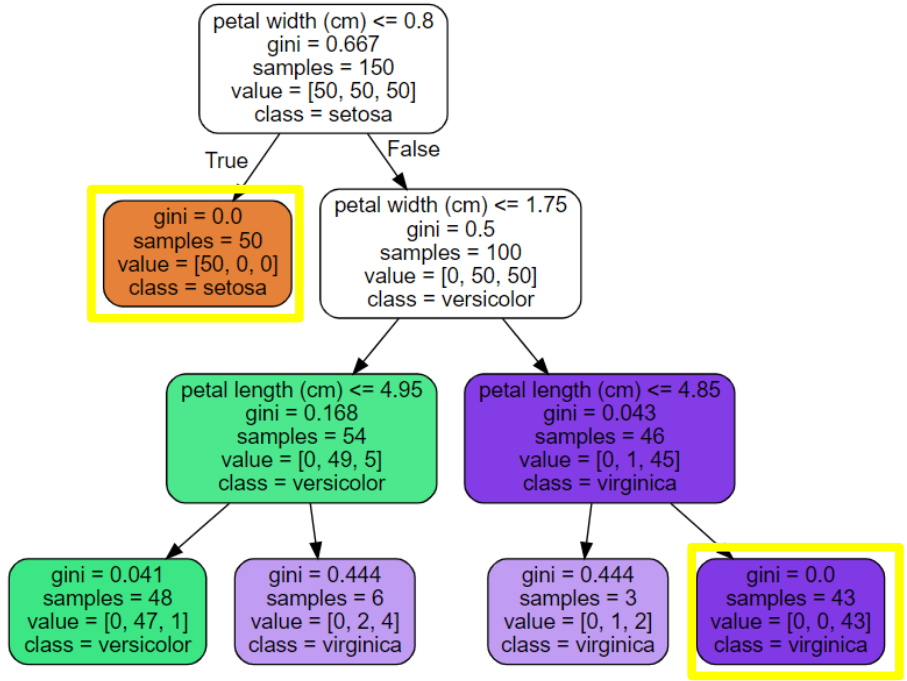



[결정나무의 결정경계]

max_depth = 2




max_depth = 3



# 의사결정 나무_ iris data

## 클래스의 확률 추정

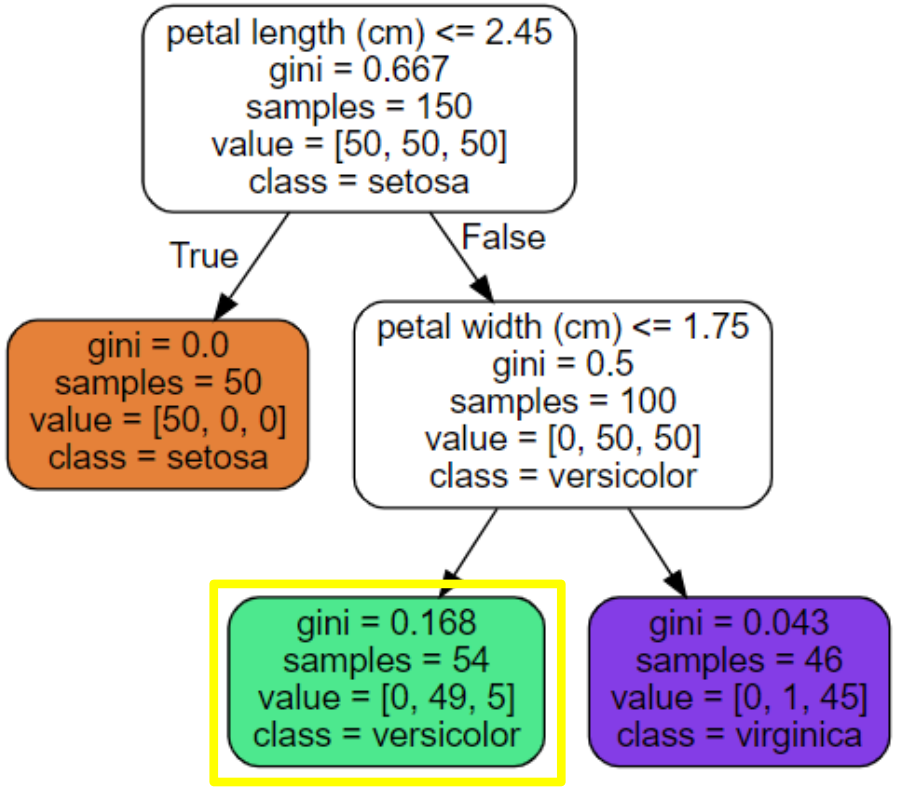
1. 샘플이 속할 리프 노드를 찾기 위하여 트리 탐색
2. 해당 노드의 클래스 k 비율을 반환
3. 가장 높은 확률을 가진 클래스를 반환

```
tree_clf.predict_proba([[5, 1.5]])
```

```
array([[0.        , 0.90740741, 0.09259259]])
```

```
tree_clf.predict([[5, 1.5]])
```

```
array([1])
```



# 의사결정 나무의 기준

## CART 알고리즘

- 하나의 변수k의 임계값 tk를 사용해 가장 순수한(불순도가 낮은) 두 서브셋으로 나눈다.
- 같은 방식으로 서브셋을 또 나누기 위한 변수k와 임계값 tk를 찾는 과정을 반복
- max_depth가  되거나, 불순도를 더이상 줄이는 분할을 찾을 수 없다면 중단

*Equation 6-2. CART cost function for classification*

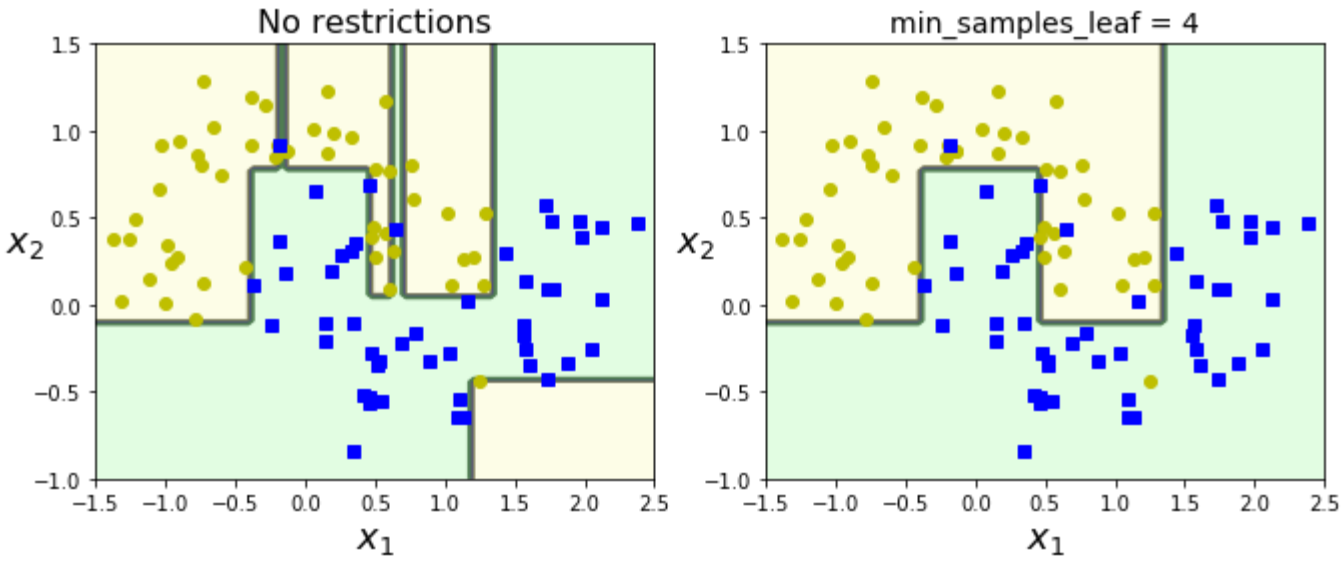
$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

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

# 의사결정 나무_ iris data

결정트리는 훈련 데이터에 대한 제약 사항이 거의 없음

→ 과대적합 될 위험이 있어 규제 매개변수를 설정하는 것이 좋음



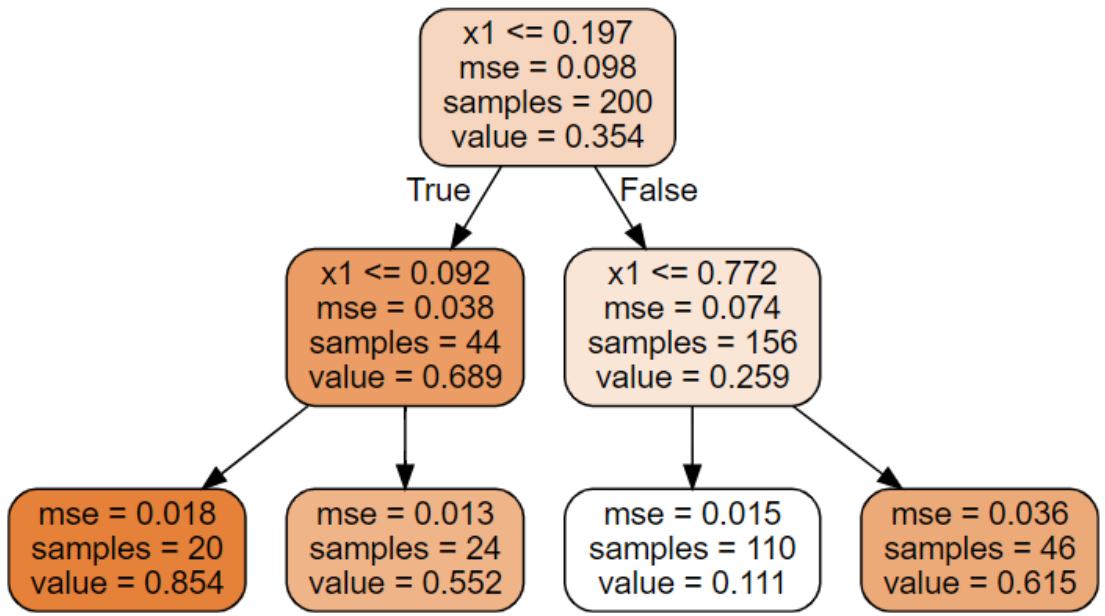
# 의사결정 나무_ regression

```
# 2차함수 + noise 추가
np.random.seed(42)
m = 200
X = np.random.rand(m, 1)             # 0~1사이 난수 생성 (200행1열)
y = 4 * (X - 0.5) ** 2               # 임의의 2차함수 생성
y = y + np.random.randn(m, 1) / 10  # 노이즈 추가
```

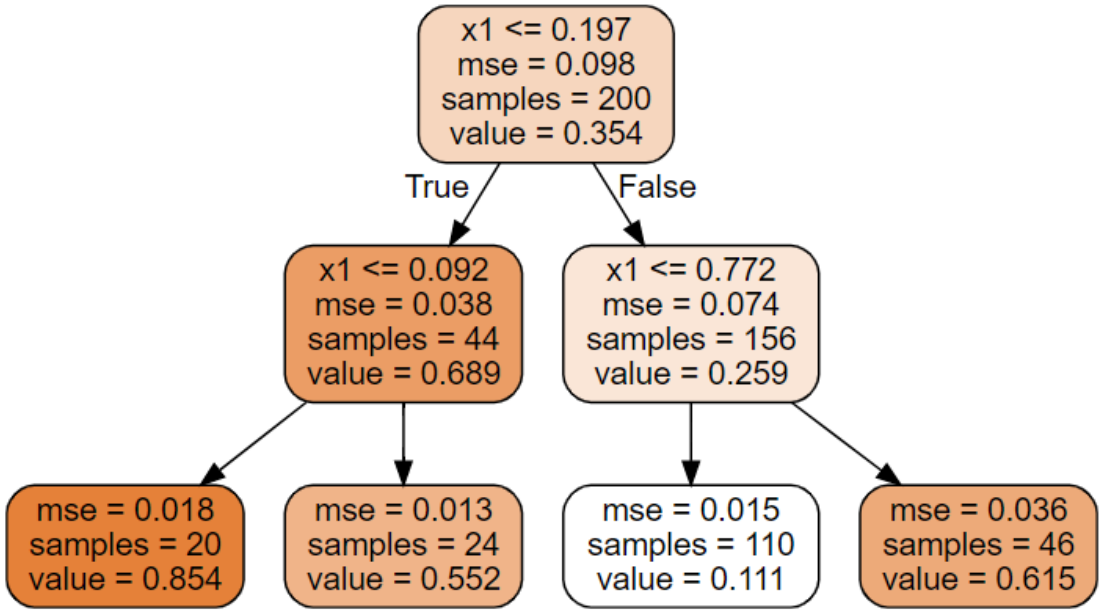
```
from sklearn.tree import DecisionTreeRegressor

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

```
DecisionTreeRegressor(criterion='mse', max_depth=2, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

```
from sklearn.tree import export_graphviz
from graphviz import Source

export_graphviz(tree_reg,                          # 모델
                out_file= "regression.dot",        # 저장경로 설정
                feature_names=["x1"],              # 변수명
                rounded = True,                    # 시각화(둥근네모)
                filled = True)                     # 시각화(도형채우기)

Source.from_file("regression.dot")
```



# 의사결정 나무_ regression




결정트리의 변수 설명

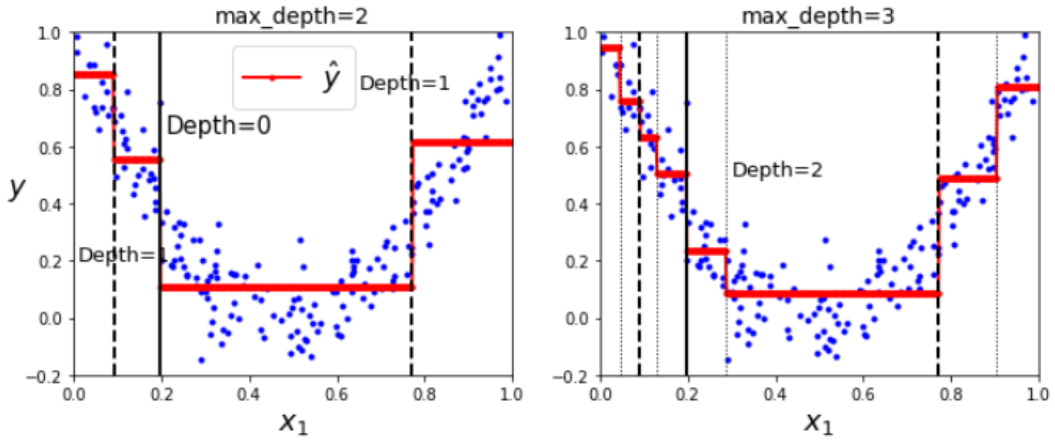
- mse : 평균제곱오차
- samples : 노드 범위에 해당하는 샘플의 수
- value : 예측한 값

x1 = 0.6인 샘플의 값을 예측한다면 예측값은 약 0.11

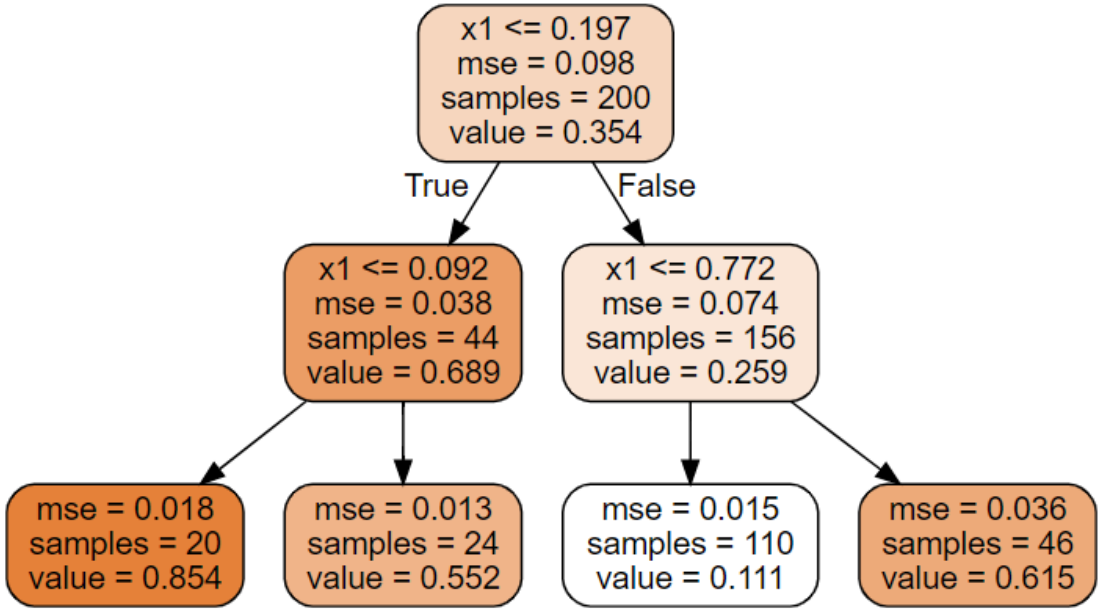
```
tree_reg.predict([[0.6]])
```

```
array([0.11063973])
```

# 의사결정 나무_ regression

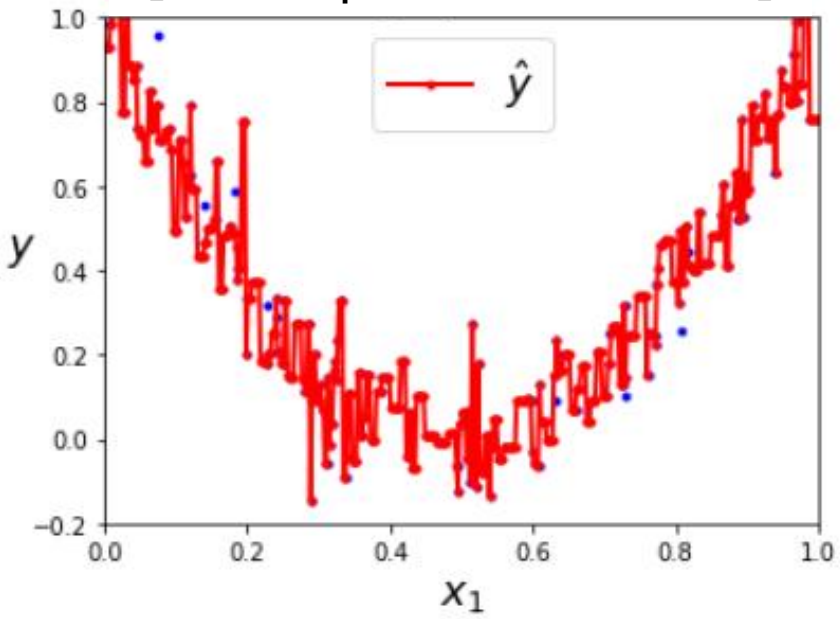



[결정나무의 결정경계]



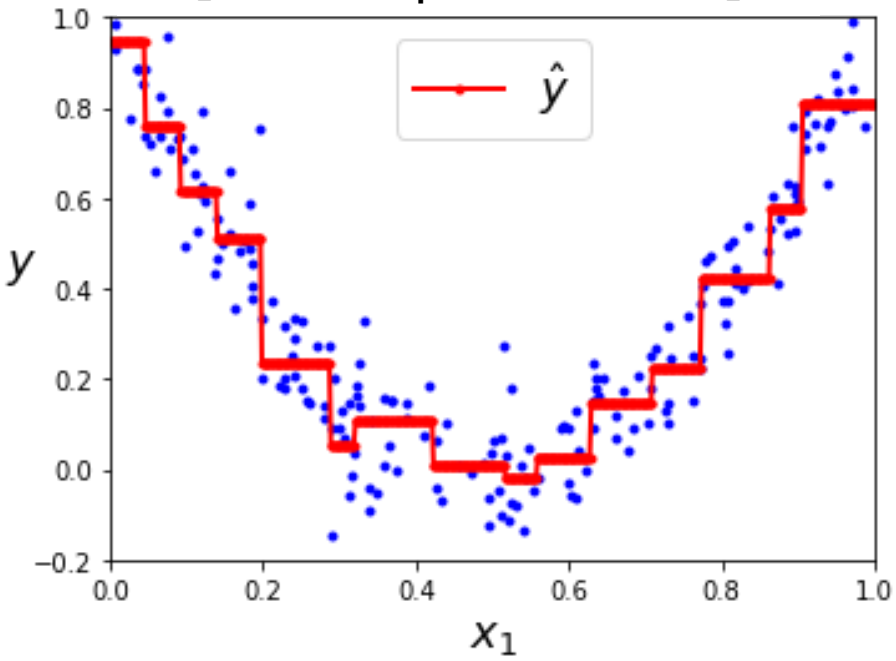
# 의사결정 나무_ regression

[max_depth 조건을 주지 않음]




대부분의 데이터와 적합하나 과적합

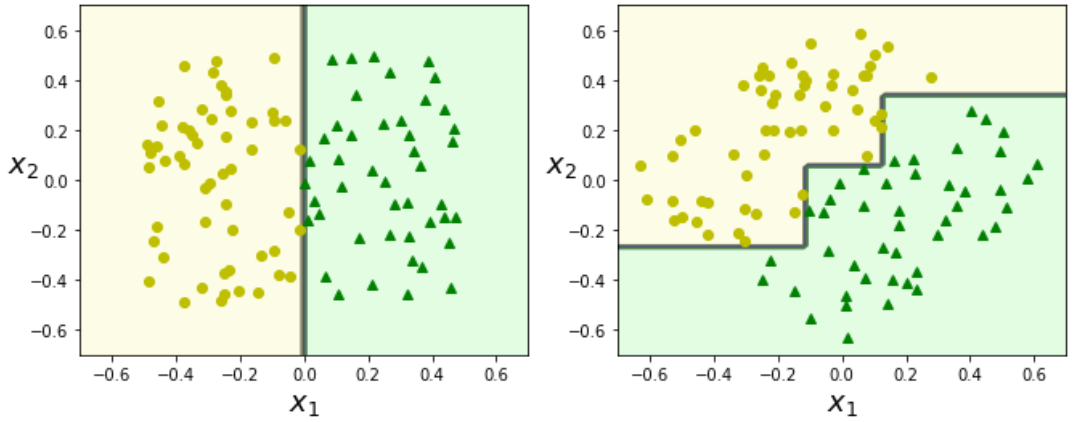
[min_samples_leaf = 10]




매개변수를 추가하여 과적합 방지

# 의사결정 나무의 불안정성

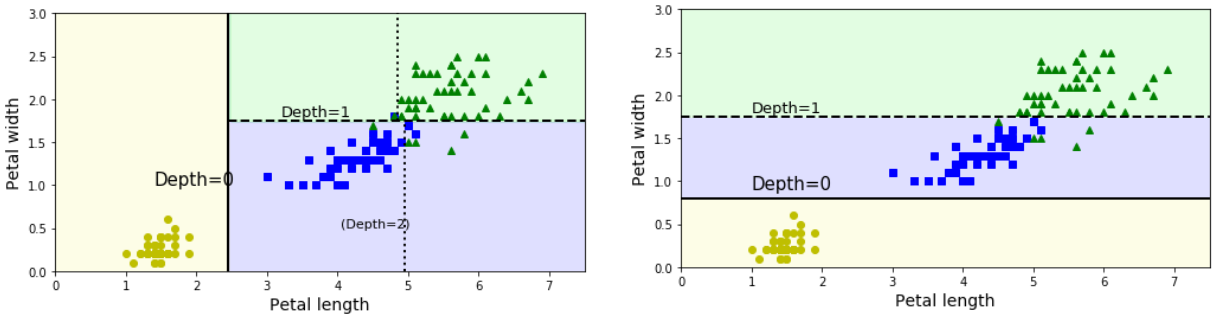
## ① rotation forest




선형 결정경계를 가진 데이터셋을 45도 회전(PCA)하면
계단 결정경계를 가진 데이터 셋이 된다.

주성분분석(PCA): 데이터의 분산을 최대한 보존하면서 서로 직교하는 새 기저(축)를 찾아, 고차원 공간의 표본들을 선형 연관성이 없는 저차원 공간으로 변환하는 기법

## ② train_data 변화에 민감




훈련 데이터셋 중 가장 넓은 데이터를 가진
versicolor를 제외하면 다른 결과를 반환한다.