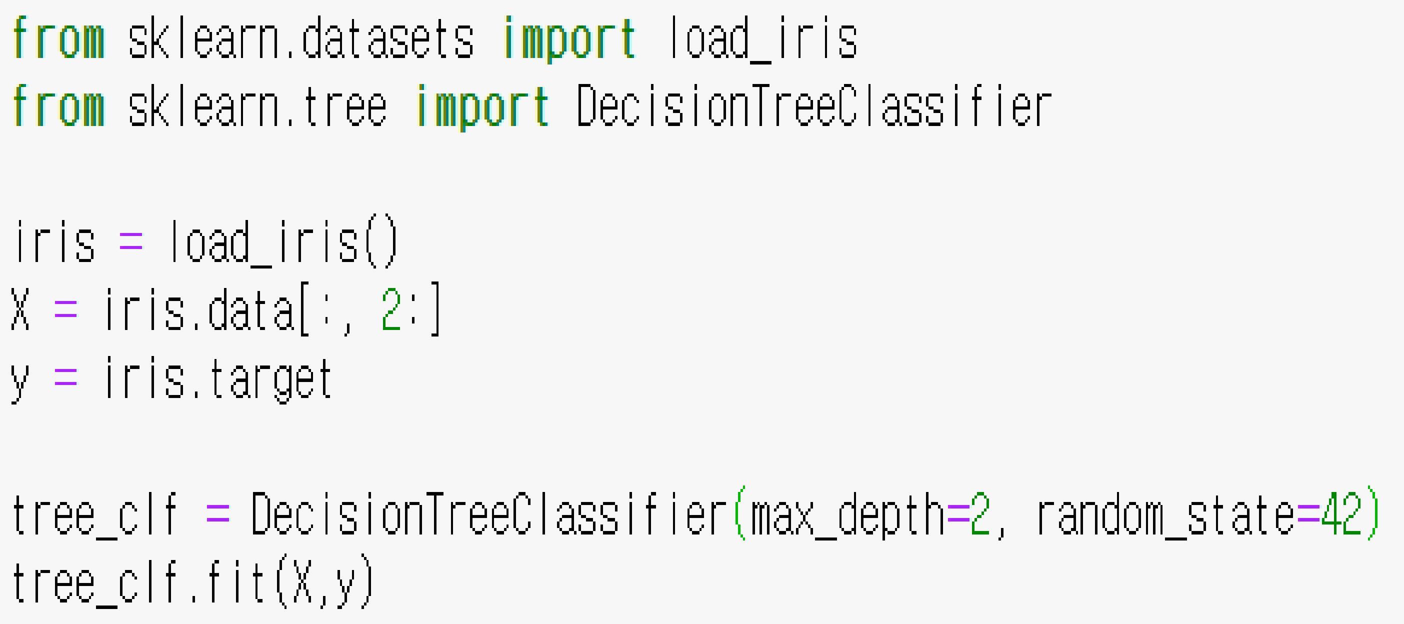
Ch6. Decision Tree

01 Decision Tree

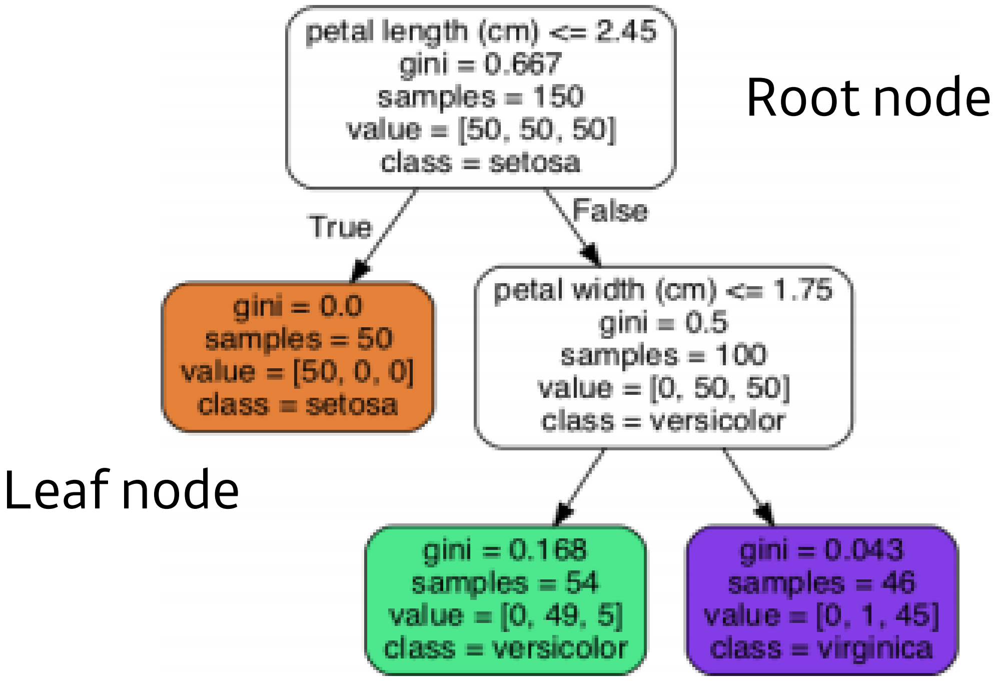
Algorithms that enable classification, regression, and multiple output operations

- **Training**



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

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

tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf.fit(X,y)
```

02 Predict



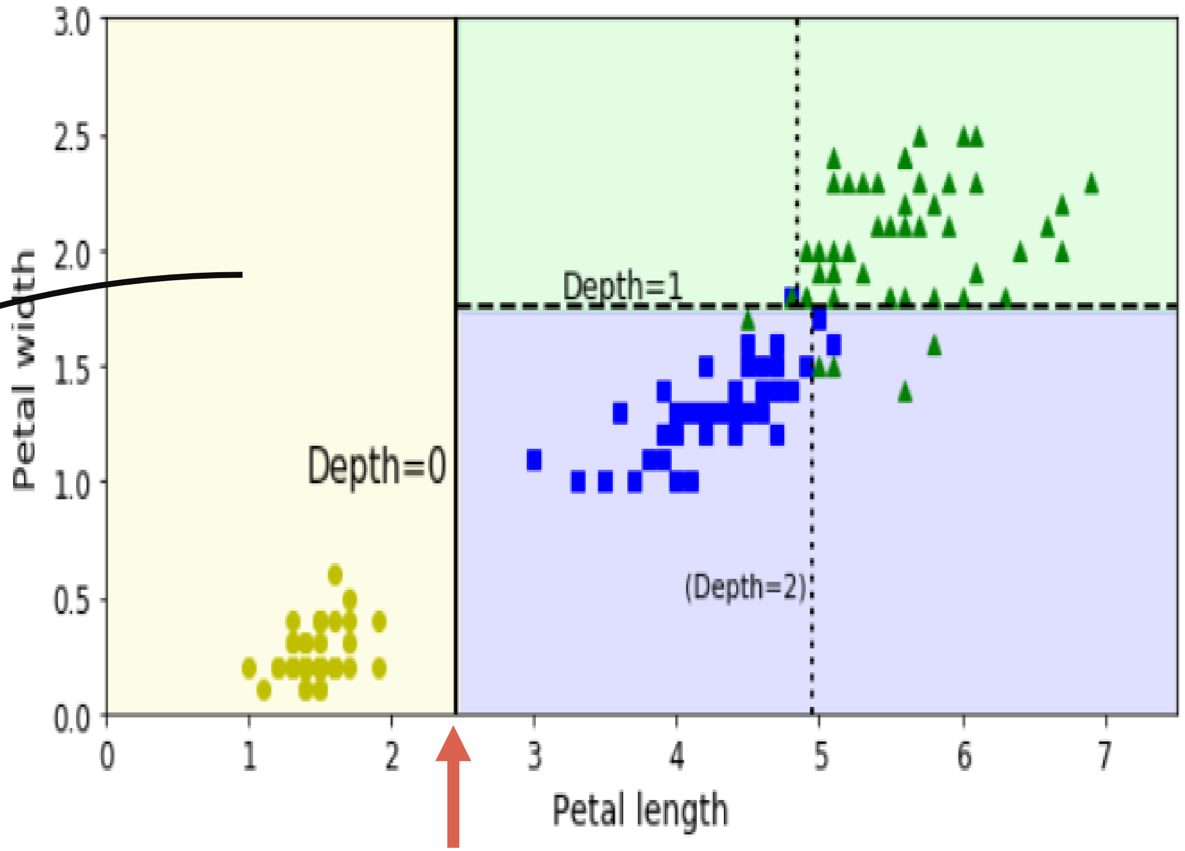
- Sample : Number of data in training sample applied to the tree

- Value : How many training samples are present on the node by class

- Gini : To determine how many data belong to same class on that node by impurity

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

02 Predict



Cannot divied
Only pure nodes

Decision Boundaries

Decision Boundaries

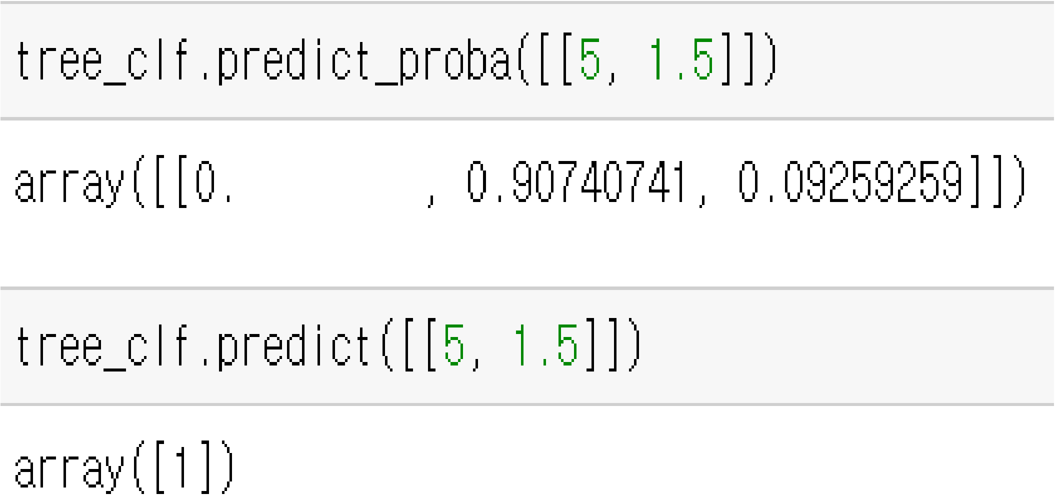
Decision tree is called **white box model** => intuitive and easy to understand how to make decisions

03 Estimate class probability

Calculate the probability that a sample belongs to a particular class

1) Explore the tree to find leaf node for sample

2) Return the percentage of the training samples of class k of that node



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

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

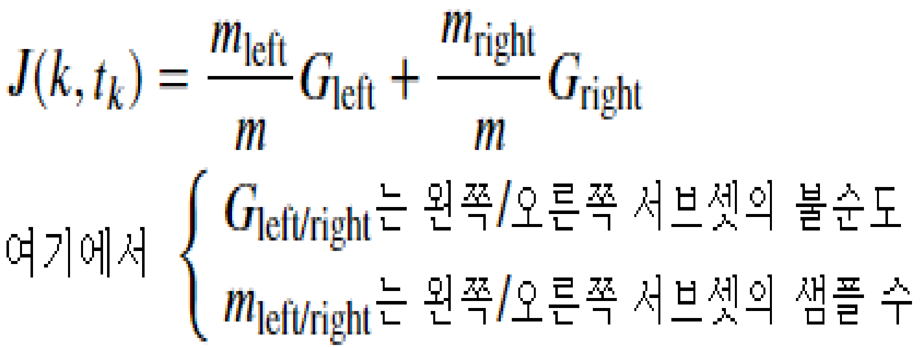
```
tree_clf.predict([[5, 1.5]])
```

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

04 CART algorithm

Divide into 2 subsets with binary algorithm used to train the decision tree

- Cost function to be minimized



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

여기에서 $\begin{cases} G_{\text{left/right}} \text{는 왼쪽/오른쪽 서브셋의 불순도} \\ m_{\text{left/right}} \text{는 왼쪽/오른쪽 서브셋의 샘플 수} \end{cases}$

- k : training set
- t_k : divide subsets into 2 by using this threshold
- Stop dividing
  1) At maximum depth
  2) When no segmentation to reduce impurity is found

=> Finding the optimal tree is NP complete problem

05 Computational Complexity

Decision tree is balanced + Predicted by checking only one characteristic value

=> Total complexity is independent of the number of attributes

: $O(\log_2(m))$

Training Algorithm : Comparing all characteristics of all training samples

=> $O(n * m \log_2(m))$

=> Small training set can be data- aligned to speed up training
Large training set slow down a lot

06 Entropy

To measure the disorder of a molecule => Stable & orderly, entropy = 0

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

- **Gini impurity** : Faster
  but, tendency to isolate the most frequent class to one side

- **Entropy** : Create a more balanced tree

=> No significant difference and create similar tree

07 Regulatory Hyperparameter

If not restricted to parameters, Overfitting is likely to occur

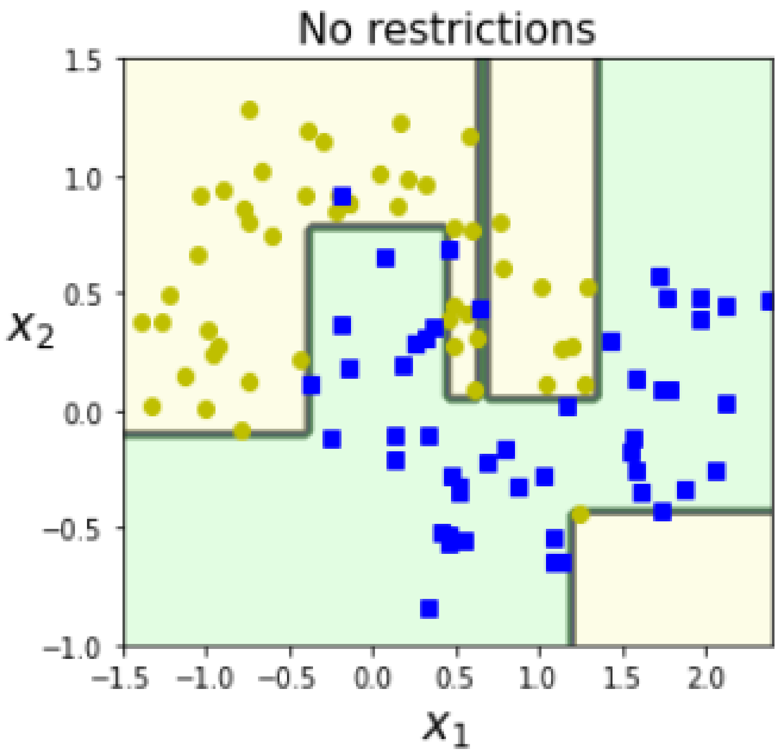
- Hyper parameter type
  - max_depth
  - min_samples_split
  - min_samples_leaf
  - min_weight_fraction_leaf
  - max_leaf_nodes
  - max_features

⇒ Min : increase paratemers

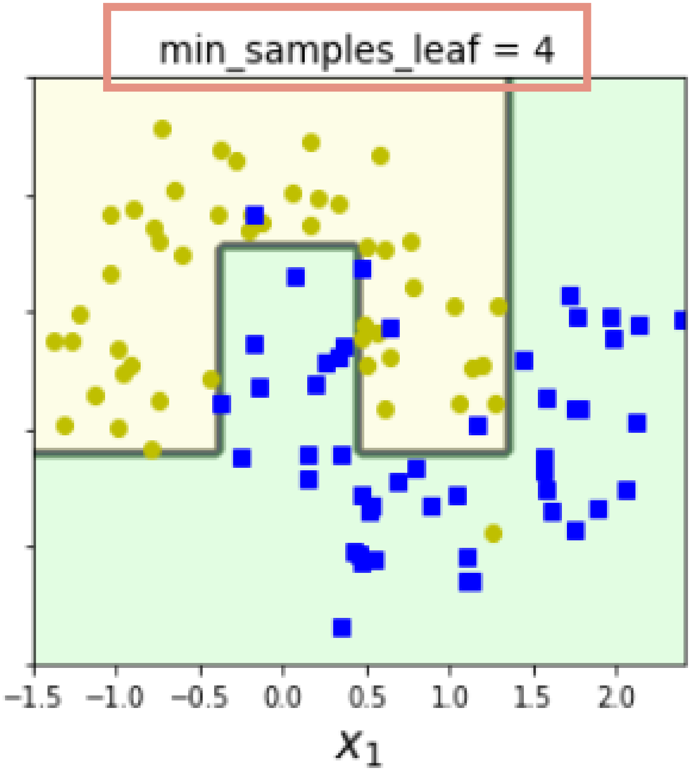
⇒ Max : decrease parameters

To increase regulation of model

07 Regulatory Hyperparameter

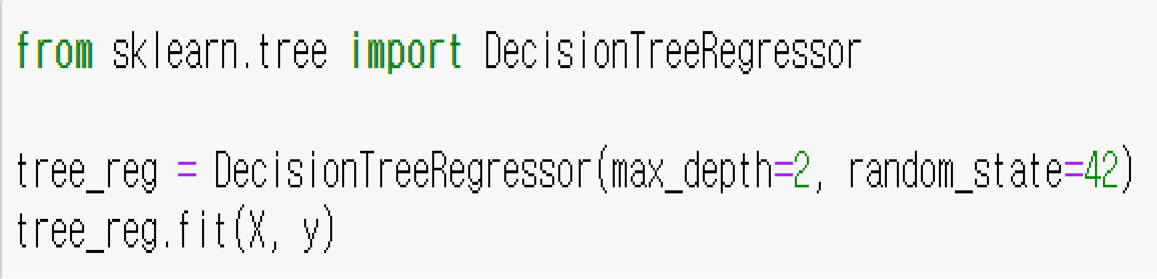


Overfitting
No regulatory variables

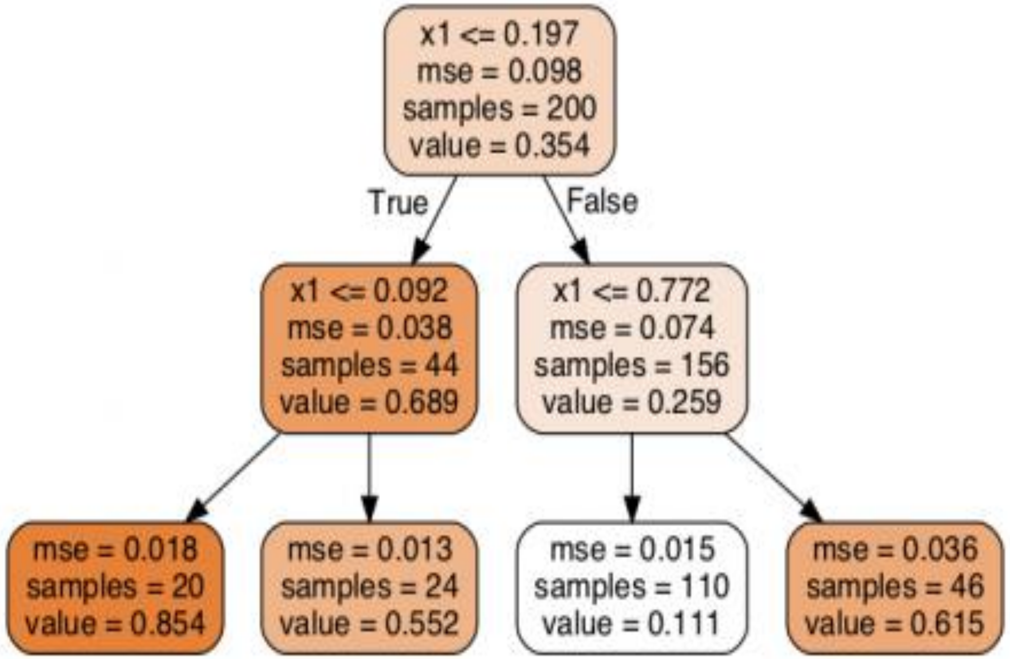


Good generalization
Applying regulatory variables

08 Regression



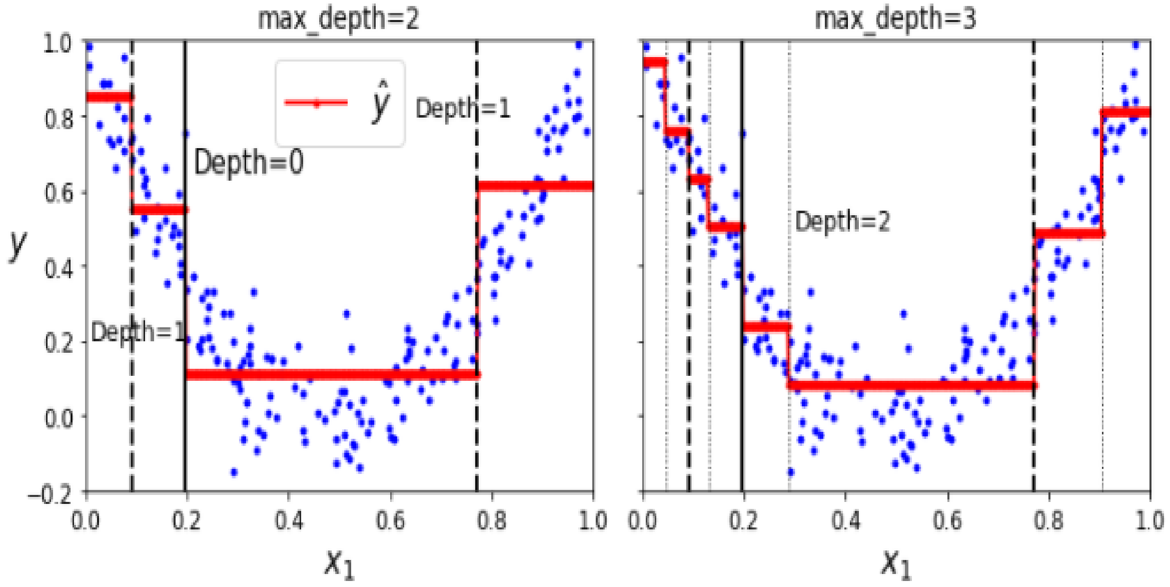
```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor(max_depth=2, random_state=42)
tree_reg.fit(X, y)
```



Difference from classification tree

: Instead of predicting classes on each node,
Predicting a value

08 Regression



- Predicted values : Average of target values

  => Algorithm divides the region so that as many samples as possible are close together

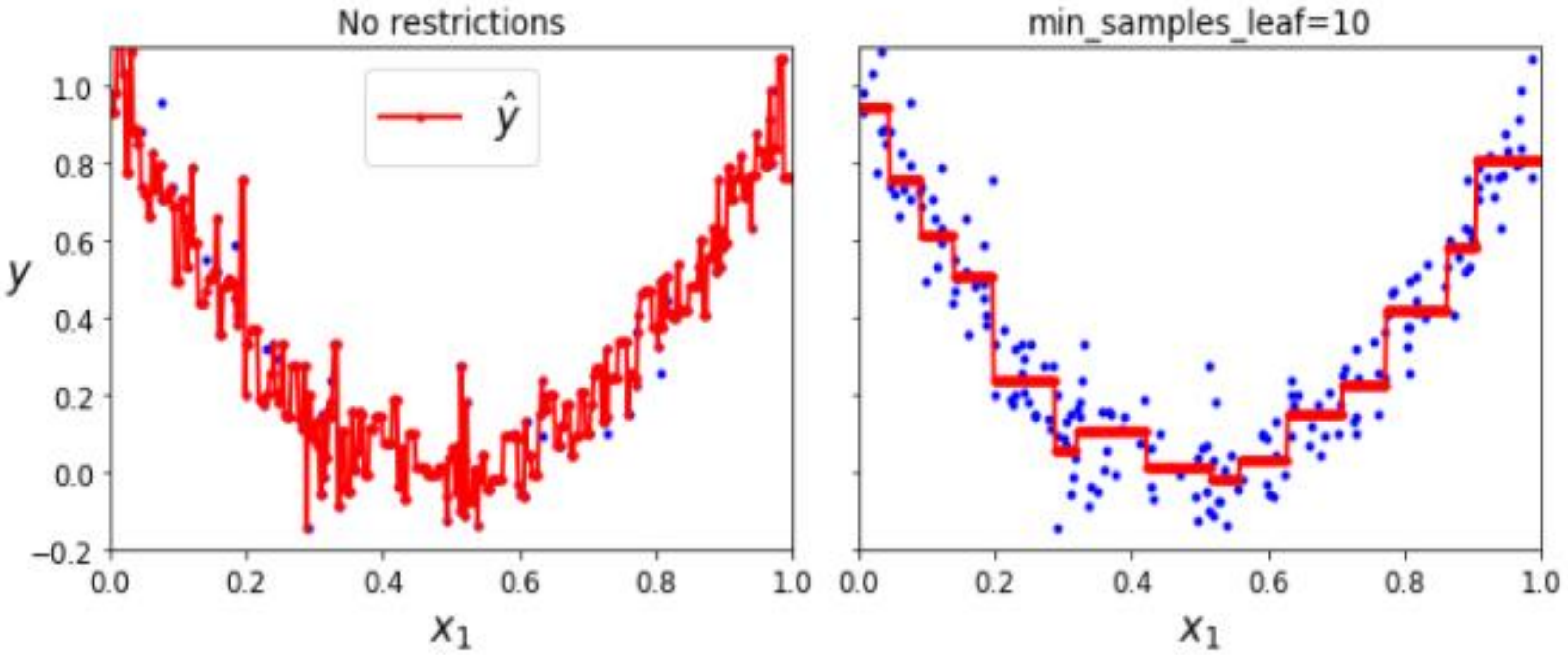
- CART : Split MSE to minimize

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

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

08 Regression

Without regulation in regression, decision trees are likely to be over-fitting

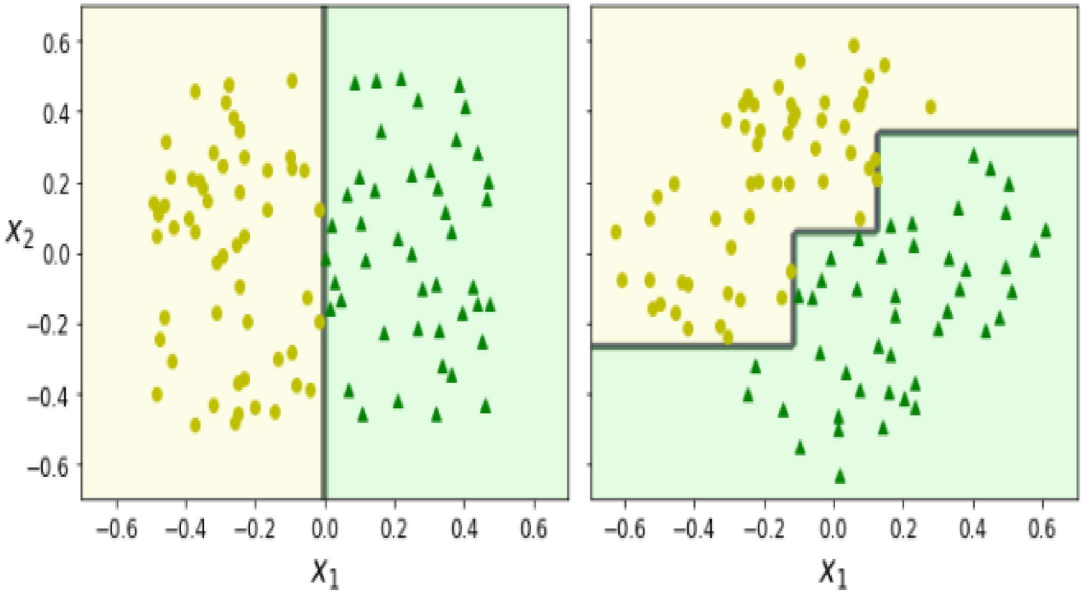


Over-fitting
unregulated

Avoid over-fitting & Derive plausible model
Regulation

09 Instability

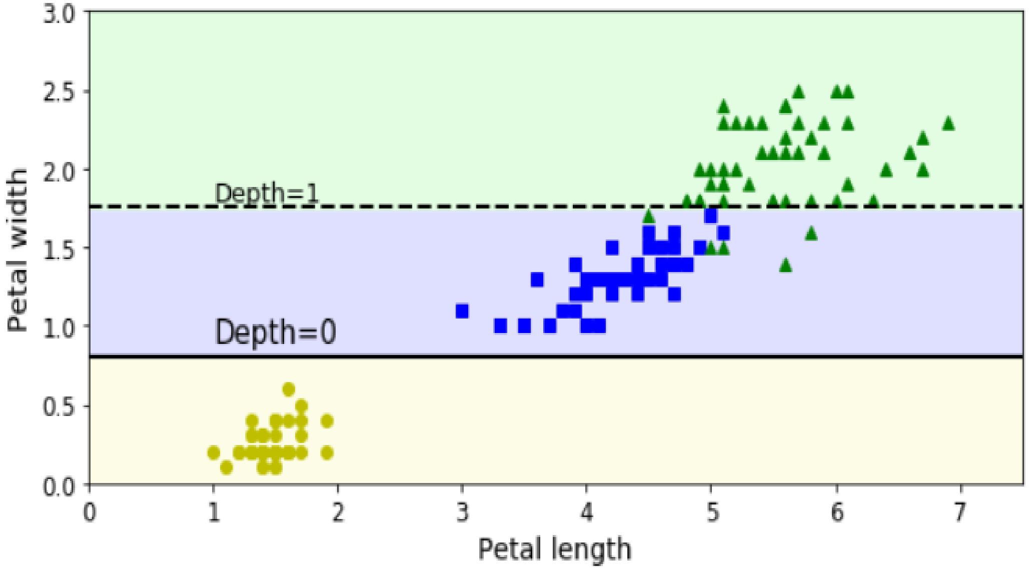
1) Sensitive to rotation of training sets



Right side will not generalize well

⇒ Training data in better direction
Using PCA to rotate

2) Sensitive to small changes in training data



Remove data from training set

⇒ Can confirm that it is different
from before

THANK YOU