CSY2082 Introduction to Artificial Intelligence

Decision Trees

Decision Tree

- Like SVMs, Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multioutput tasks.
- Decision Trees are also the fundamental components of Random Forests, which are among the most powerful Machine Learning algorithms available today.

Training / test data

Features				Labels
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	Iris setosa
4.9	3.0	1.4	0.2	Iris setosa
7.0	3.2	4.7	1.4	Iris versicolor
6.4	3.2	4.5	1.5	Iris versicolor
6.3	3.3	6.0	2.5	Iris virginica
5.8	3.3	6.0	2.5	Iris virginica

Training and Visualizing a Decision Tree

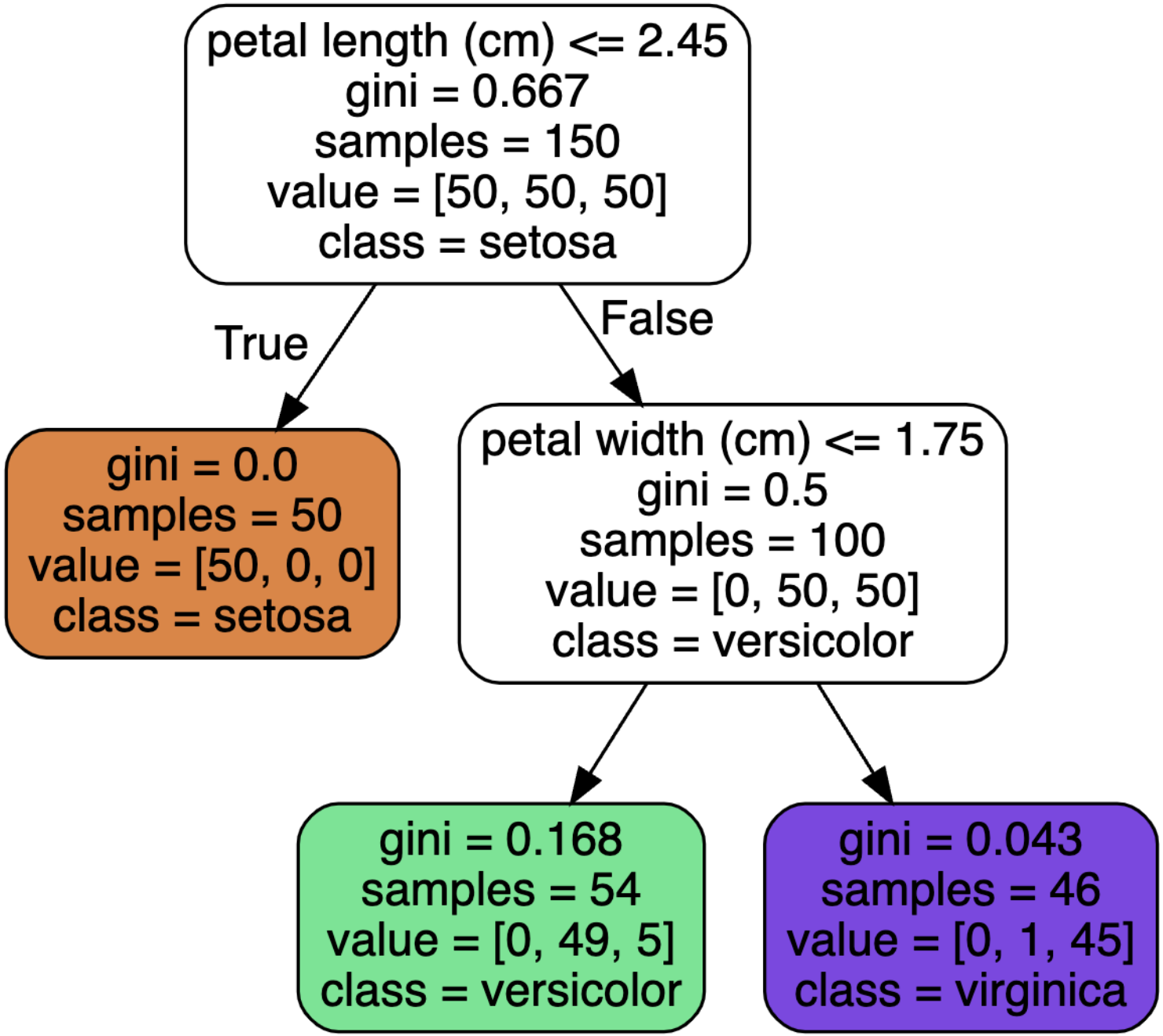
- train a DecisionTreeClassifier on the iris dataset

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

iris = load_iris(as_frame=True)
X_iris = iris.data[["petal length (cm)", "petal width (cm)"]].values
y_iris = iris.target

tree_clf = DecisionTreeClassifier(max_depth=2)
tree_clf.fit(X_iris, y_iris)
```

```
▼ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=2)
```



- A node's *samples* attribute counts how many training instances it applies to.
- A node's *value* attribute tells you how many training instances of each class this node applies to.
- a node's *gini* attribute measures its impurity.

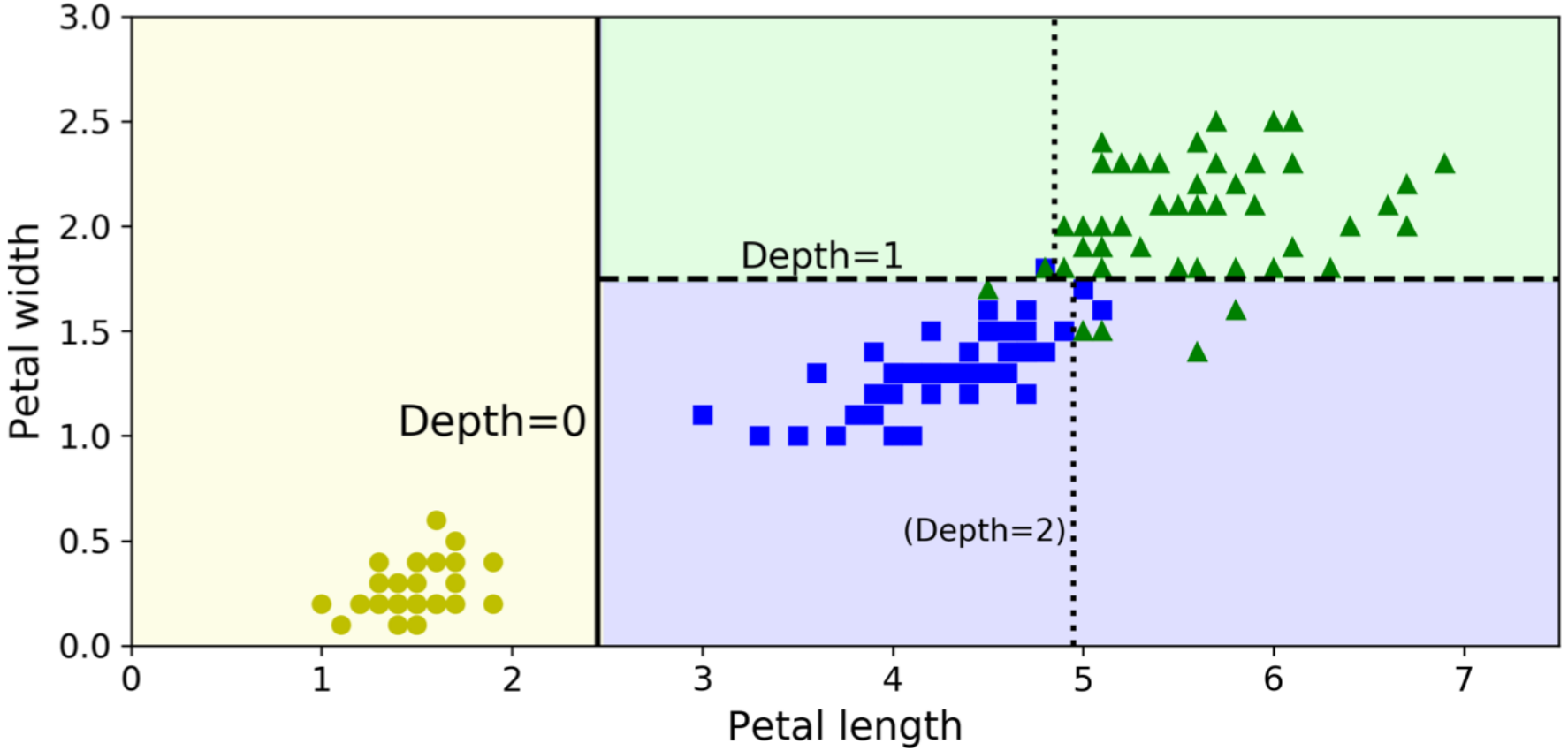
$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

$1 - (0/54)^2 - (49/54)^2 - (5/54)^2 \approx 0.168$.

- Scikit-Learn uses the CART algorithm, which produces only binary trees: non-leaf nodes always have two children.

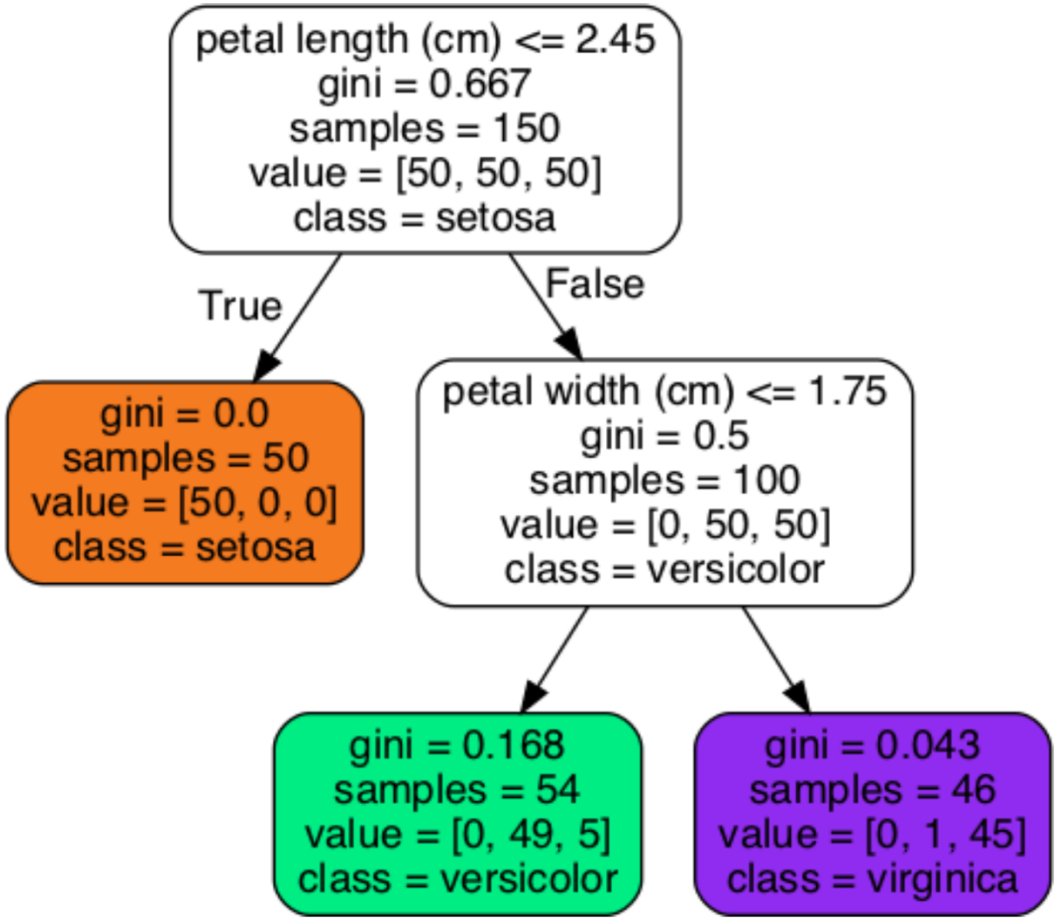
Decision Tree decision boundaries and interpretability

- As you can see Decision Trees are fairly intuitive and their decisions are easy to interpret.
  - Such models are often called **white box models**. In contrast, neural networks are generally considered **black box models**.



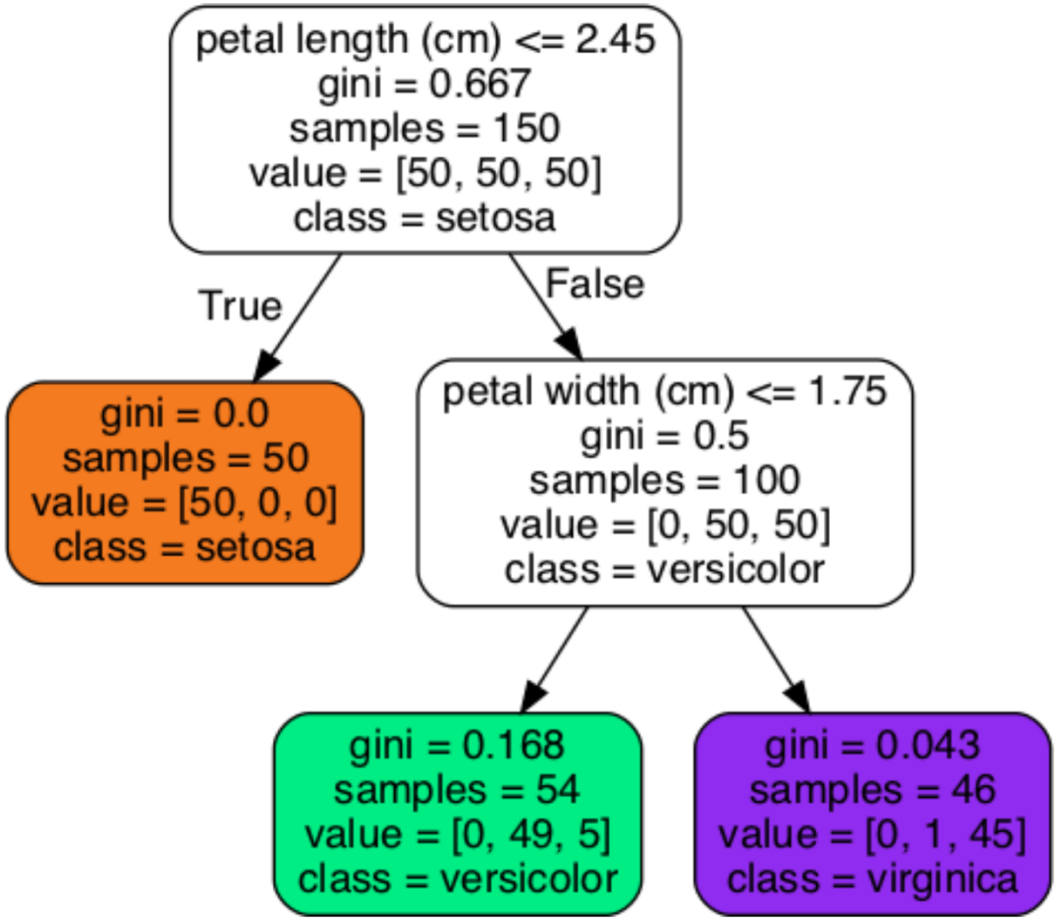
max_depth=2

Estimating Class Probabilities



- A Decision Tree can also estimate the probability that an instance belongs to a particular class k
  - first it traverses the tree to find the leaf node for this instance, and then it returns the ratio of training instances of class k in this node.
- For example, suppose you have found a flower whose petals are 5 cm long and 1.5 cm wide.
  - the Decision Tree should output the following probabilities: 0% for Iris-Setosa (0/54), 90.7% for Iris-Versicolor (49/54), and 9.3% for Iris-Virginica (5/54).

Estimating Class Probabilities




```
tree_clf.predict_proba([[5, 1.5]]).round(3)
```

```
array([[0.   , 0.907, 0.093]])
```

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

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

The CART Training Algorithm

- Scikit-Learn uses the *Classification And Regression Tree (CART)* algorithm to train Decision.
- The idea is really quite simple: the algorithm first splits the training set in two subsets using a single feature k and a threshold tk (e.g., “petal length ≤ 2.45 cm”) recursively.
- It searches for the pair (k, tk) that produces the purest subsets (weighted by their size).

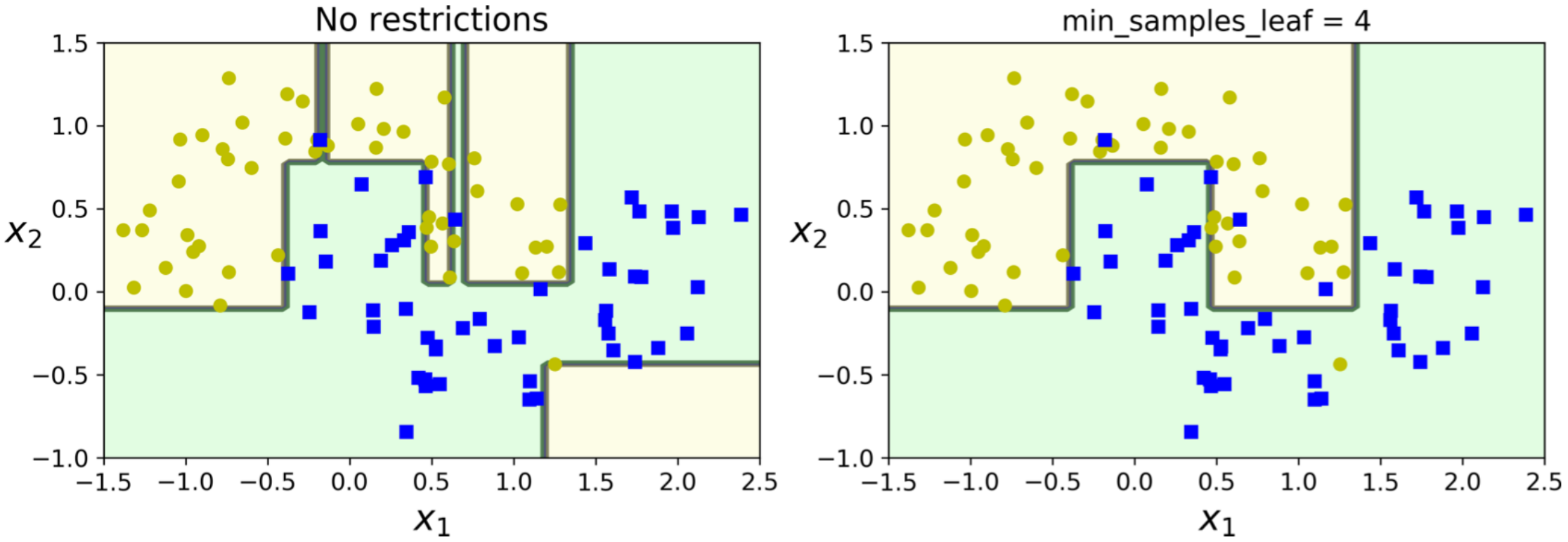
CART cost function for classification

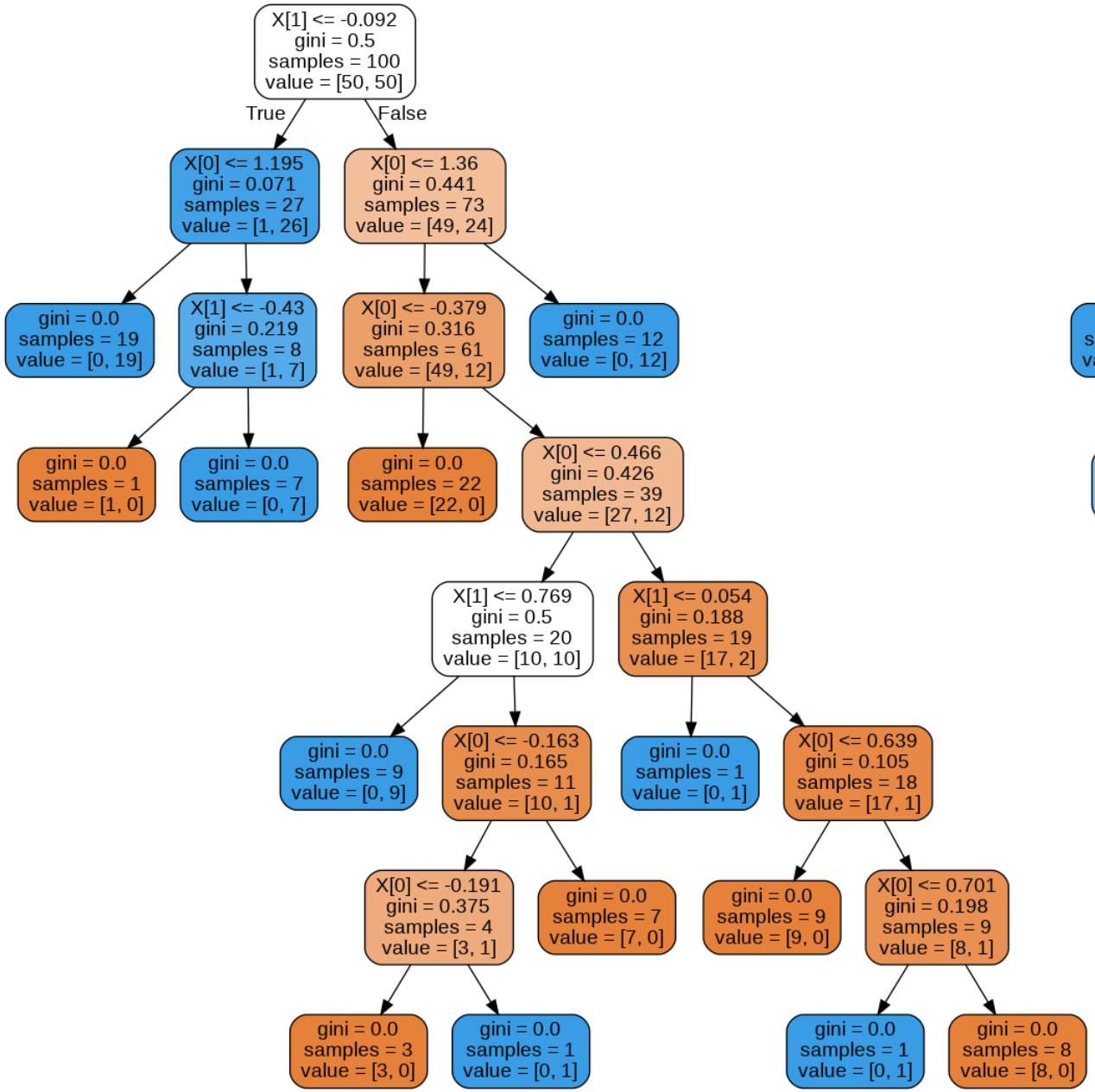
$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

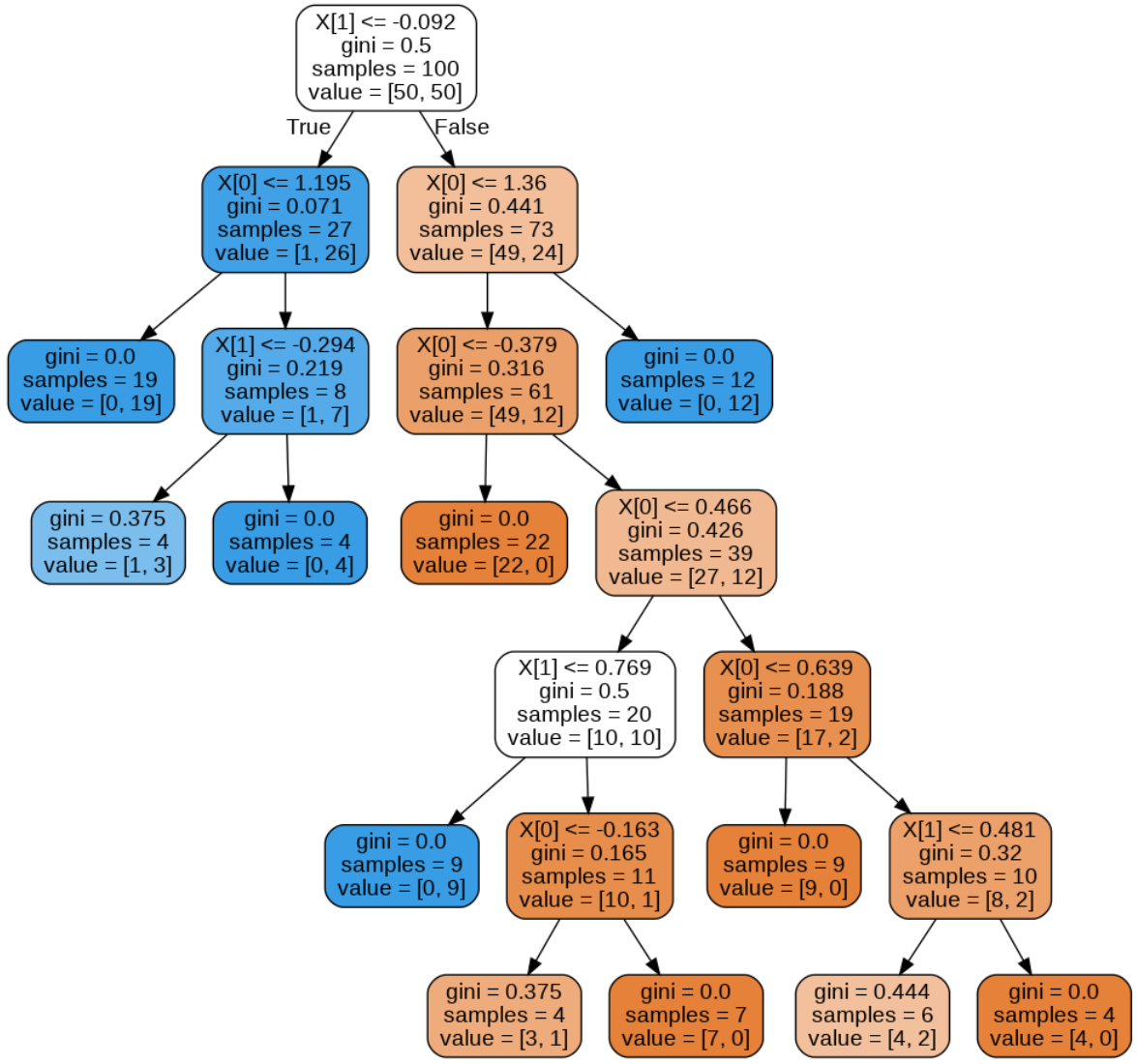
where $\begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$

Regularization

- To avoid overfitting the training data, you need to restrict the Decision Tree's freedom during training such as restricting the maximum depth of the Decision Tree.
  - The DecisionTreeClassifier class has a few other options: *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), *max_leaf_nodes* (maximum number of leaf nodes), and *max_features* (maximum number of features that are evaluated for splitting at each node).

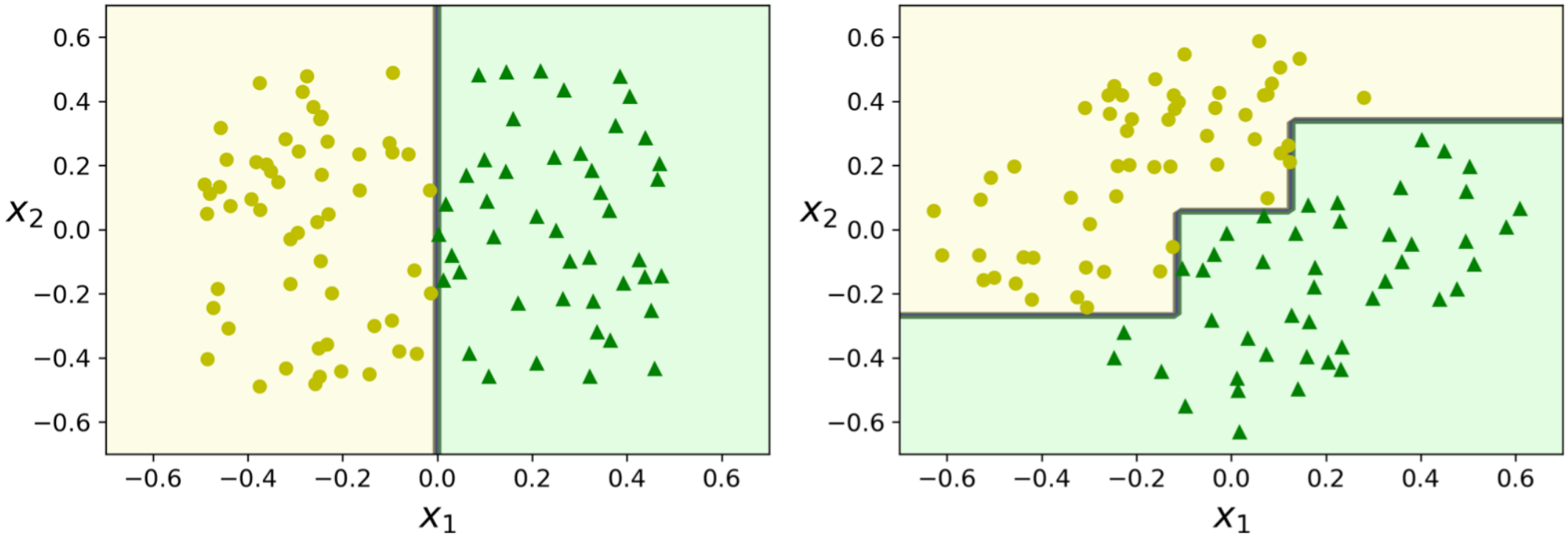



X[1] <= -0.092
gini = 0.5
samples = 100
value = [50, 50]
True
False
X[0] <= 1.195
gini = 0.071
samples = 27
value = [1, 26]
X[0] <= 1.36
gini = 0.441
samples = 73
value = [49, 24]
gini = 0.0
samples = 19
value = [0, 19]
X[1] <= -0.43
gini = 0.219
samples = 8
value = [1, 7]
X[0] <= -0.379
gini = 0.316
samples = 61
value = [49, 12]
gini = 0.0
samples = 12
value = [0, 12]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 7
value = [0, 7]
gini = 0.0
samples = 22
value = [22, 0]
X[0] <= 0.466
gini = 0.426
samples = 39
value = [27, 12]
X[1] <= 0.769
gini = 0.5
samples = 20
value = [10, 10]
X[1] <= 0.054
gini = 0.188
samples = 19
value = [17, 2]
gini = 0.0
samples = 9
value = [0, 9]
X[0] <= -0.163
gini = 0.165
samples = 11
value = [10, 1]
gini = 0.0
samples = 1
value = [0, 1]
X[0] <= 0.639
gini = 0.105
samples = 18
value = [17, 1]
X[0] <= -0.191
gini = 0.375
samples = 4
value = [3, 1]
gini = 0.0
samples = 7
value = [7, 0]
gini = 0.0
samples = 9
value = [9, 0]
X[0] <= 0.701
gini = 0.198
samples = 9
value = [8, 1]
gini = 0.0
samples = 3
value = [3, 0]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 8
value = [8, 0]


X[1] <= -0.092
gini = 0.5
samples = 100
value = [50, 50]
True
False
X[0] <= 1.195
gini = 0.071
samples = 27
value = [1, 26]
X[0] <= 1.36
gini = 0.441
samples = 73
value = [49, 24]
gini = 0.0
samples = 19
value = [0, 19]
X[1] <= -0.294
gini = 0.219
samples = 8
value = [1, 7]
X[0] <= -0.379
gini = 0.316
samples = 61
value = [49, 12]
gini = 0.0
samples = 12
value = [0, 12]
gini = 0.375
samples = 4
value = [1, 3]
gini = 0.0
samples = 4
value = [0, 4]
gini = 0.0
samples = 22
value = [22, 0]
X[0] <= 0.466
gini = 0.426
samples = 39
value = [27, 12]
X[1] <= 0.769
gini = 0.5
samples = 20
value = [10, 10]
X[0] <= 0.639
gini = 0.188
samples = 19
value = [17, 2]
gini = 0.0
samples = 9
value = [0, 9]
X[0] <= -0.163
gini = 0.165
samples = 11
value = [10, 1]
gini = 0.0
samples = 9
value = [9, 0]
X[1] <= 0.481
gini = 0.32
samples = 10
value = [8, 2]
gini = 0.375
samples = 4
value = [3, 1]
gini = 0.0
samples = 7
value = [7, 0]
gini = 0.444
samples = 6
value = [4, 2]
gini = 0.0
samples = 4
value = [4, 0]

Instability

- First, as you may have noticed, Decision Trees love orthogonal decision boundaries (all splits are perpendicular to an axis), which makes them sensitive to training set rotation.
- On the left, a Decision Tree can split it easily, while on the right, after the dataset is rotated by 45°, the decision boundary looks unnecessarily convoluted and won't generalise well..



Exercise

- Run the code to build the decision tree multiple times. Do you always get the same tree?
- If you are getting different trees, how do you know which one to use for your application? Compare their performance.