UNIT -3
DECISION TREES

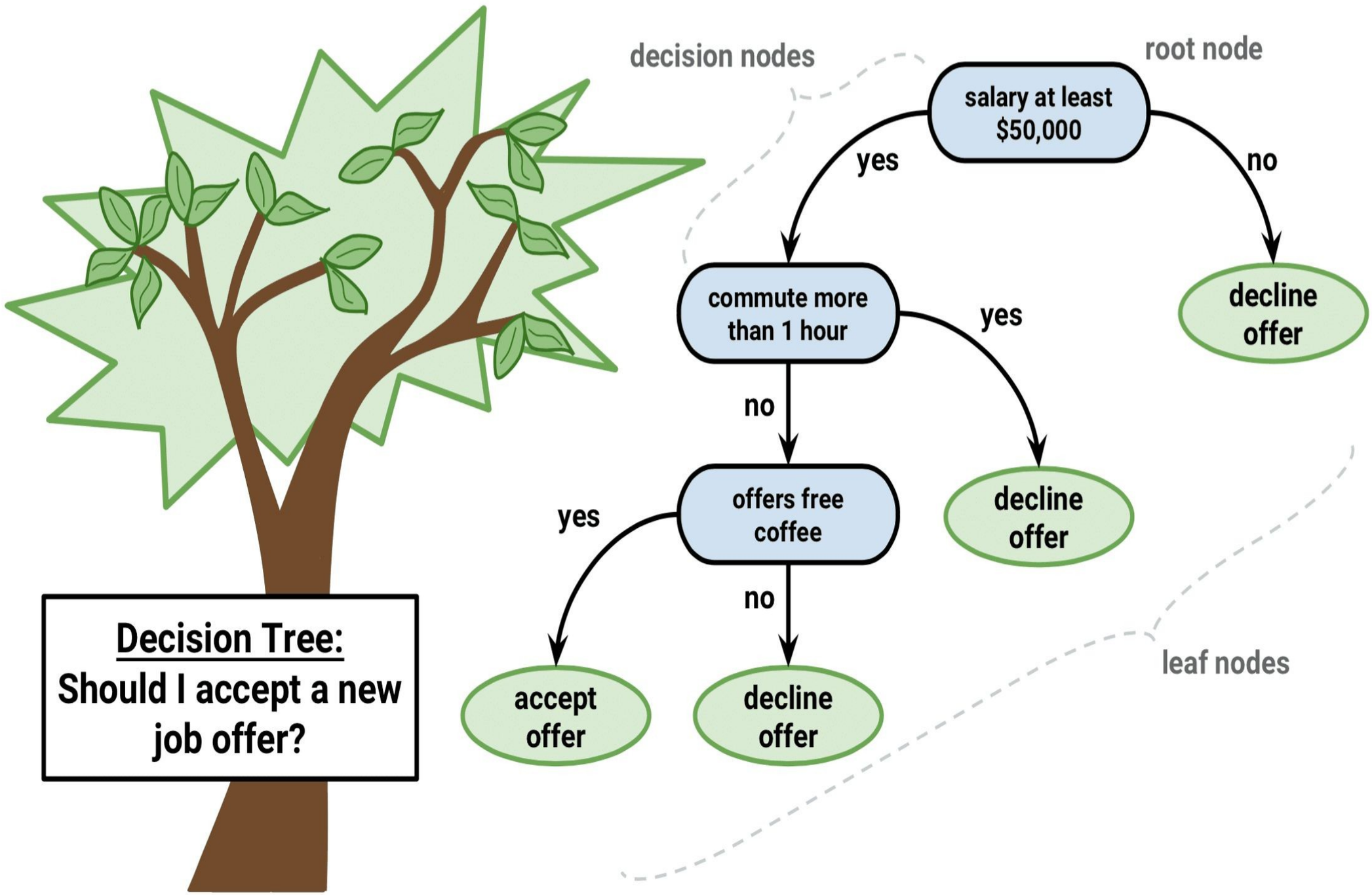
What do we learn?

- Working with Decision Tree
  - Training and Visualizing
  - Make Predictions & Decision Boundary
- Estimating Class Probabilities
- CART algorithm
- Computational Complexity
- Gini Impurity or Entropy
- Regularization Hyperparameters.

What is a Decision Tree

- A decision tree is one of the supervised machine learning algorithms.
- *Decision Trees* are versatile Machine Learning algorithms that can perform both classification and regression tasks.
- A decision tree follows a set of if-else conditions to visualize the data and classify it according to the conditions.

Example of decision tree



1. **How to decide which feature should be located at the root node,**

2. **Most accurate feature to serve as internal nodes or leaf nodes,**

3. **How to divide tree,**

4. **How to measure the accuracy of splitting tree and many more.**

Entropy

it is nothing just the measure of disorder, or measure of purity. Basically, ***it is the measurement of the impurity or randomness in the data points.***

$$\text{Entropy} = -\sum_{i=1}^{n} p_i * \text{Log}_2(p_i)$$

Gini Index, also known as Gini impurity, **calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly.** If all the elements are linked with a single class then it can be called pure.

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} (P_i)^2$$

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

- **Outlook**

Outlook is a nominal feature. It can be sunny, overcast or rain. I will summarize the final decisions for outlook feature.

Outlook	Yes	No	Number of instances
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

$$Gini(Outlook = Sunny) = 1 - (2/5)^2 - (3/5)^2 = 1 - 0.16 - 0.36 = 0.48$$

$$Gini(Outlook = Overcast) = 1 - (4/4)^2 - (0/4)^2 = 0$$

$$Gini(Outlook = Rain) = 1 - (3/5)^2 - (2/5)^2 = 1 - 0.36 - 0.16 = 0.48$$

Then, we will calculate weighted sum of gini indexes for outlook feature.

$$Gini(Outlook) = (5/14) \times 0.48 + (4/14) \times 0 + (5/14) \times 0.48 = 0.171 + 0 + 0.171 = 0.342$$

- **Temperature**

Similarly, temperature is a nominal feature and it could have 3 different values: Cool, Hot and Mild. Let's summarize decisions for temperature feature.

Temperature	Yes	No	Number of instances
Hot	2	2	4
Cool	3	1	4
Mild	4	2	6

$$Gini(Temp = Hot) = 1 - (2/4)^2 - (2/4)^2 = 0.5$$

$$Gini(Temp = Cool) = 1 - (3/4)^2 - (1/4)^2 = 1 - 0.5625 - 0.0625 = 0.375$$

$$Gini(Temp = Mild) = 1 - (4/6)^2 - (2/6)^2 = 1 - 0.444 - 0.111 = 0.445$$

We'll calculate weighted sum of gini index for temperature feature

$$Gini(Temp) = (4/14) \times 0.5 + (4/14) \times 0.375 + (6/14) \times 0.445 = 0.142 + 0.107 + 0.190 = 0.439$$

- **Humidity**

Humidity is a binary class feature. It can be high or normal.

Humidity	Yes	No	Number of instances
High	3	4	7
Normal	6	1	7

$$Gini(Humidity = High) = 1 - (3/7)^2 - (4/7)^2 = 1 - 0.183 - 0.326 = 0.489$$

$$Gini(Humidity = Normal) = 1 - (6/7)^2 - (1/7)^2 = 1 - 0.734 - 0.02 = 0.244$$

Weighted sum for humidity feature will be calculated next

$$Gini(Humidity) = (7/14) \times 0.489 + (7/14) \times 0.244 = 0.367$$

- Wind

Wind is a binary class similar to humidity. It can be weak and strong.

Wind	Yes	No	Number of instances
Weak	6	2	8
Strong	3	3	6

$$Gini(Wind = Weak) = 1 - (6/8)^2 - (2/8)^2 = 1 - 0.5625 - 0.062 = 0.375$$

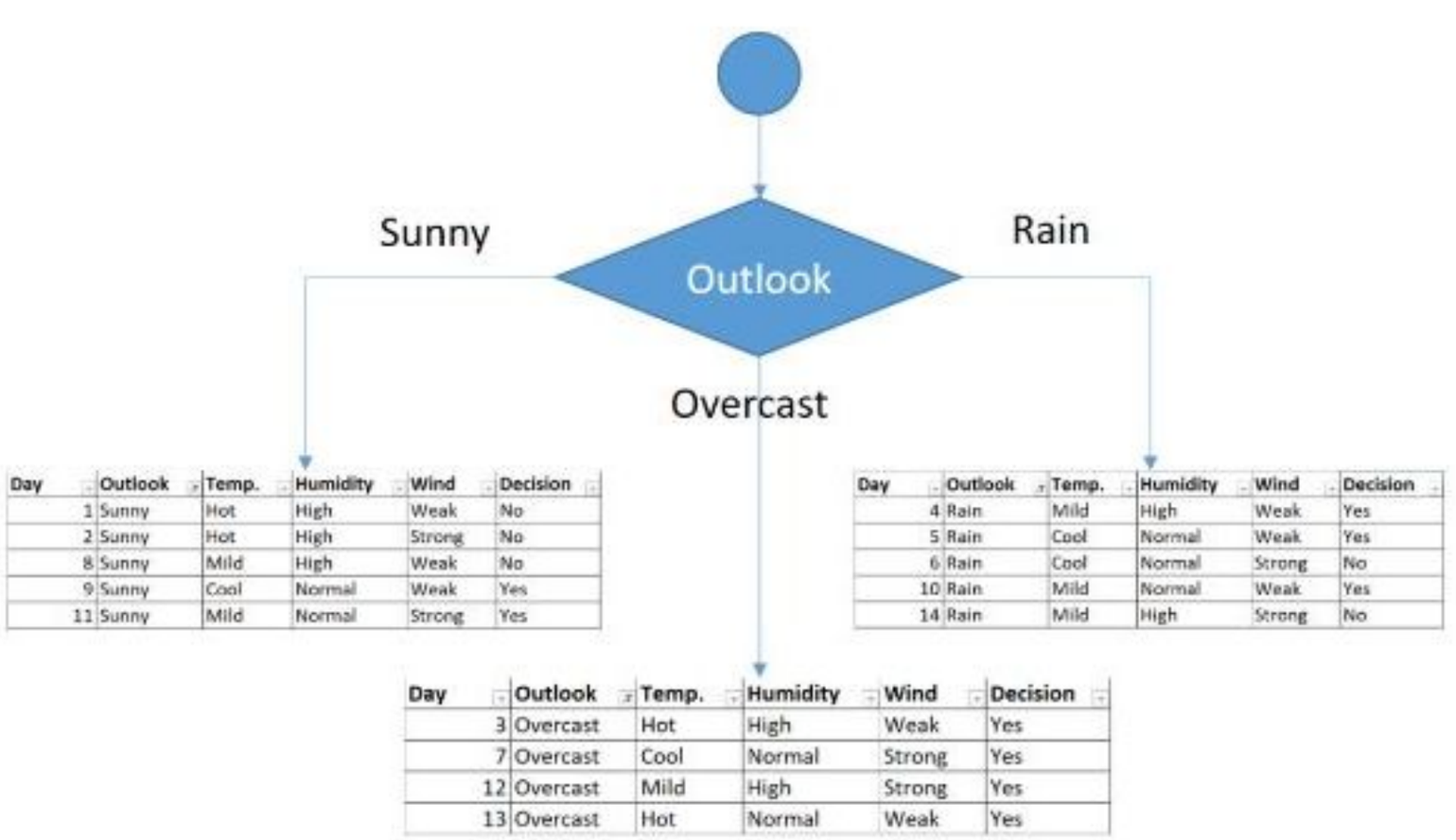
$$Gini(Wind = Strong) = 1 - (3/6)^2 - (3/6)^2 = 1 - 0.25 - 0.25 = 0.5$$

$$Gini(Wind) = (8/14) \times 0.375 + (6/14) \times 0.5 = 0.428$$

We've calculated gini index values for each feature. The winner will be outlook feature because its cost is the lowest.

Feature	Gini Index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428

We'll put outlook decision at the top of the tree.



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

We will apply same principles to those sub datasets in the following steps.

Focus on the sub dataset for sunny outlook. We need to find the gini index scores for temperature, humidity and wind features respectively.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

- Gini of temperature for sunny outlook

Temperature	Yes	No	Number of instances
Hot	0	2	2
Cool	1	0	1
Mild	1	1	2

$$\text{Gini(Outlook=Sunny and Temp.=Hot)} = 1-(0/2)^2-(2/2)^2 = 0$$

$$\text{Gini(Outlook=Sunny and Temp.=Cool)} = 1-(1/1)^2-(0/1)^2 = 0$$

$$\text{Gini(Outlook=Sunny and Temp.=Mild)} = 1-(1/2)^2-(1/2)^2 = 1-0.25-0.25 = 0.5$$

$$\text{Gini(Outlook=Sunny and Temp.)} = (2/5)\times 0 + (1/5)\times 0 + (2/5)\times 0.5 = 0.2$$

- Gini of humidity for sunny outlook

Humidity	Yes	No	Number of instances
High	0	3	3
Normal	2	0	2

$$\text{Gini(Outlook=Sunny and Humidity=High)} = 1-(0/3)^2-(3/3)^2 = 0$$

$$\text{Gini(Outlook=Sunny and Humidity=Normal)} = 1-(2/2)^2-(0/2)^2 = 0$$

$$\text{Gini(Outlook=Sunny and Humidity)} = (3/5) \times 0 + (2/5) \times 0 = 0$$

- Gini of wind for sunny outlook

Wind	Yes	No	Number of instances
Weak	1	2	3
Strong	1	1	2

$$\text{Gini(Outlook=Sunny and Wind=Weak)} = 1-(1/3)^2-(2/3)^2 = 0.266$$

$$\text{Gini(Outlook=Sunny and Wind=Strong)} = 1-(1/2)^2-(1/2)^2 = 0.2$$

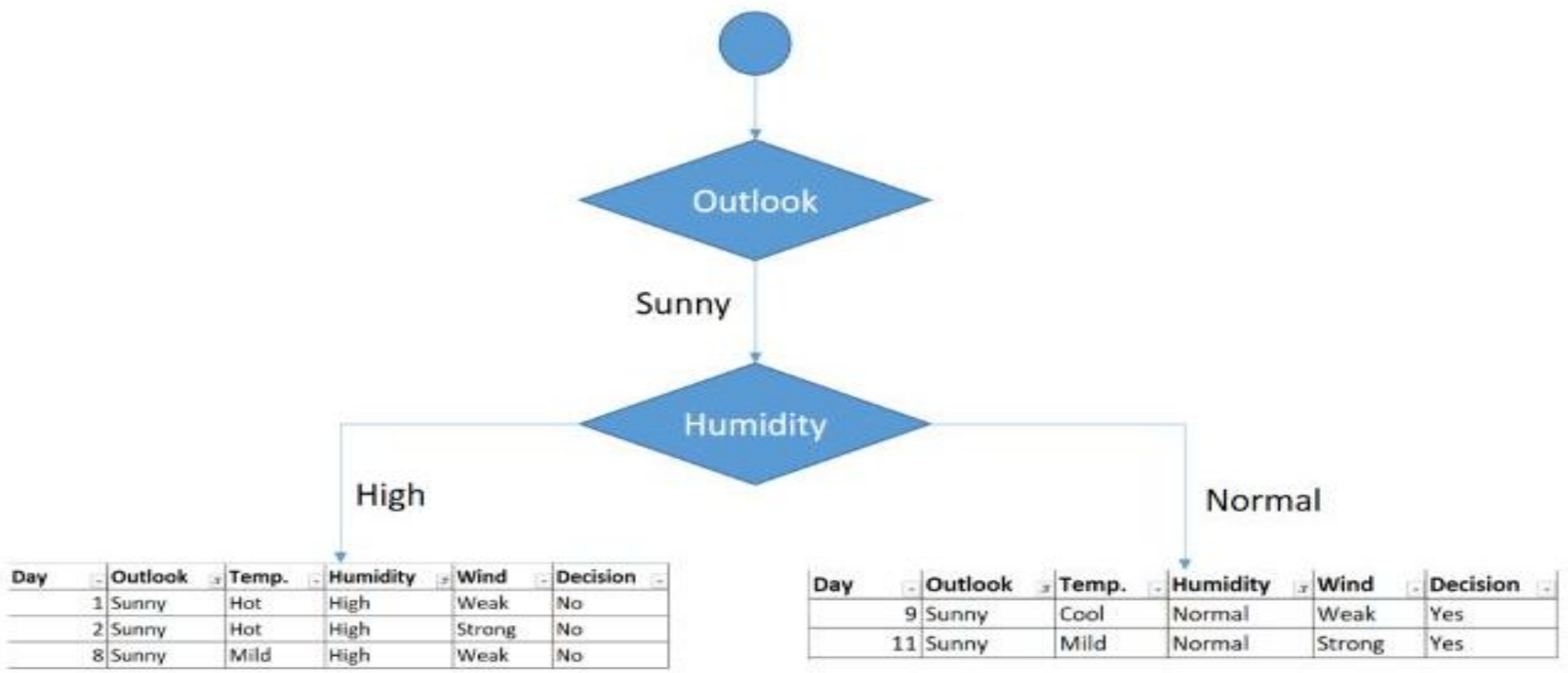
$$\text{Gini(Outlook=Sunny and Wind)} = (3/5) \times 0.266 + (2/5) \times 0.2 = 0.466$$

- **Decision for sunny outlook**

We've calculated gini index scores for feature when outlook is sunny. The winner is humidity because it has the lowest value.

Feature	Gini index
Temperature	0.2
Humidity	0
Wind	0.466

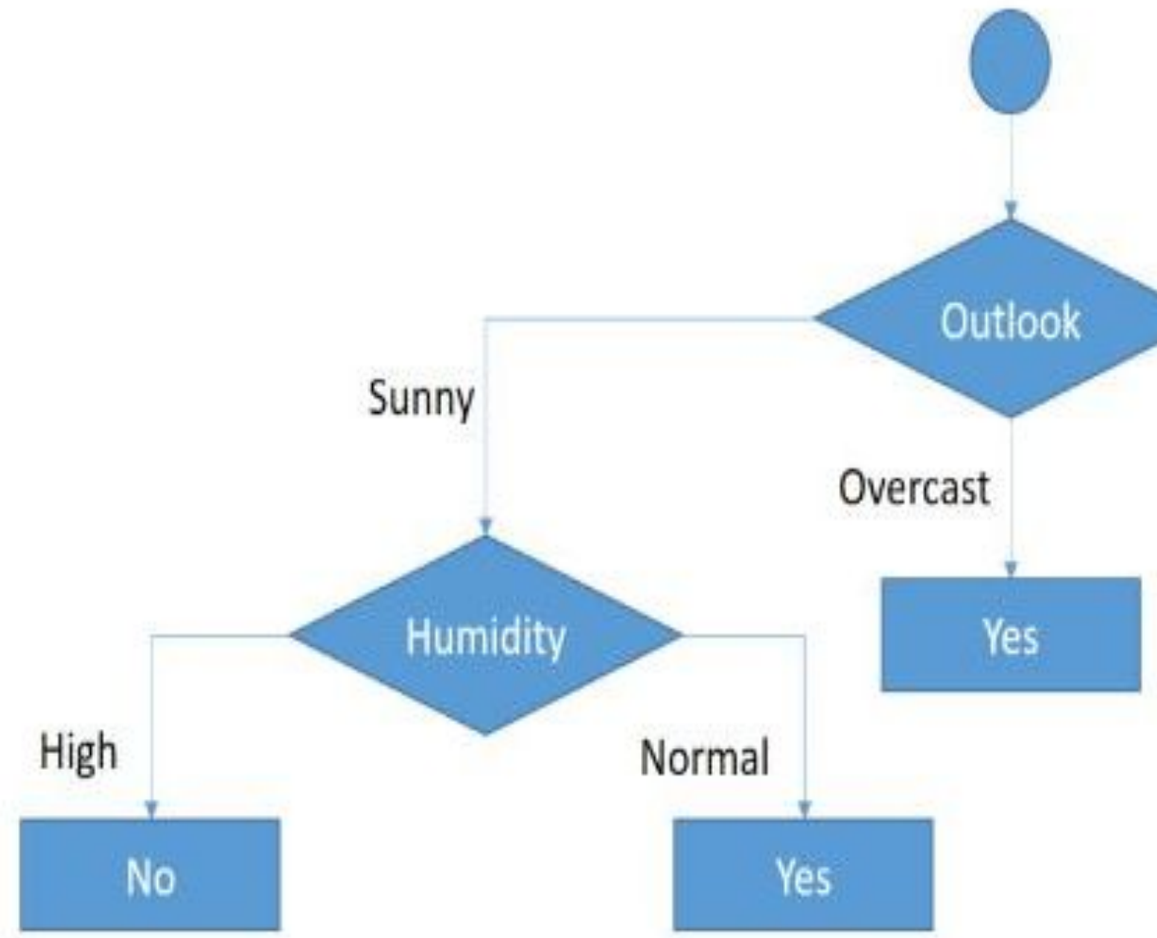
We'll put humidity check at the extension of sunny outlook.



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

As seen, decision is always no for high humidity and sunny outlook. On the other hand, decision will always be yes for normal humidity and sunny outlook. This branch is over.



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Decisions for high and normal humidity

Rain outlook

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

We'll calculate gini index scores for temperature, humidity and wind features when outlook is rain.

- **Gini of temperature for rain outlook**

Temperature	Yes	No	Number of instances
Cool	1	1	2
Mild	2	1	3

$$\text{Gini(Outlook=Rain and Temp.=Cool)} = 1-(1/2)^2-(1/2)^2 = 0.5$$

$$\text{Gini(Outlook=Rain and Temp.=Mild)} = 1-(2/3)^2-(1/3)^2 = 0.444$$

$$\text{Gini(Outlook=Rain and Temp.)} = (2/5) \times 0.5 + (3/5) \times 0.444 = 0.466$$

- **Gini of humidity for rain outlook**

Humidity	Yes	No	Number of instances
High	1	1	2
Normal	2	1	3

$$\text{Gini(Outlook=Rain and Humidity=High)} = 1-(1/2)^2-(1/2)^2 = 0.5$$

$$\text{Gini(Outlook=Rain and Humidity=Normal)} = 1-(2/3)^2-(1/3)^2 = 0.444$$

$$\text{Gini(Outlook=Rain and Humidity)} = (2/5) \times 0.5 + (3/5) \times 0.444 = 0.466$$

- **Gini of wind for rain outlook**

Wind	Yes	No	Number of instances
Weak	3	0	3
Strong	0	2	2

$$\text{Gini}(\text{Outlook=Rain and Wind=Weak}) = 1-(3/3)^2-(0/3)^2 = 0$$

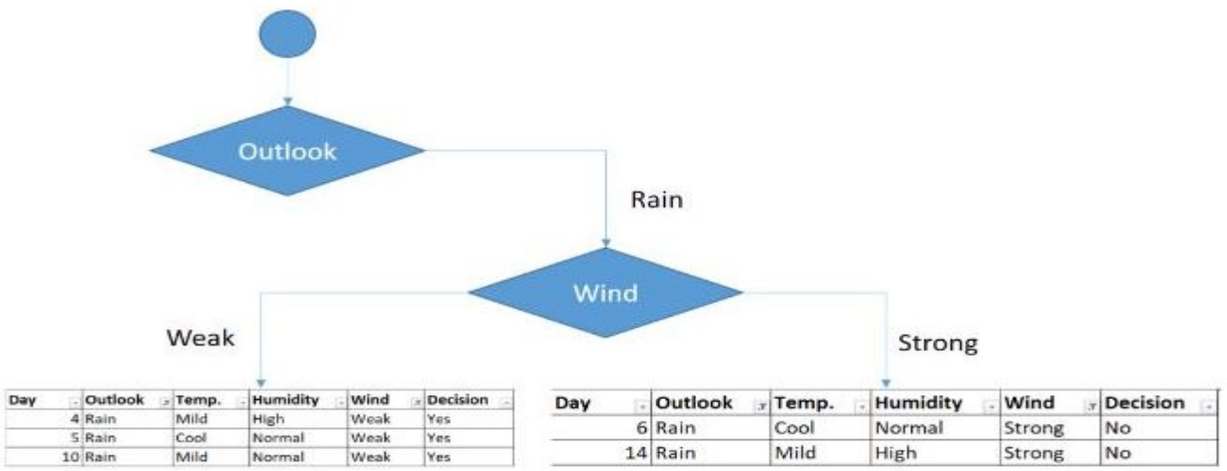
$$\text{Gini}(\text{Outlook=Rain and Wind=Strong}) = 1-(0/2)^2-(2/2)^2 = 0$$

$$\text{Gini}(\text{Outlook=Rain and Wind}) = (3/5)\times 0 + (2/5)\times 0 = 0$$

- **Decision for rain outlook**

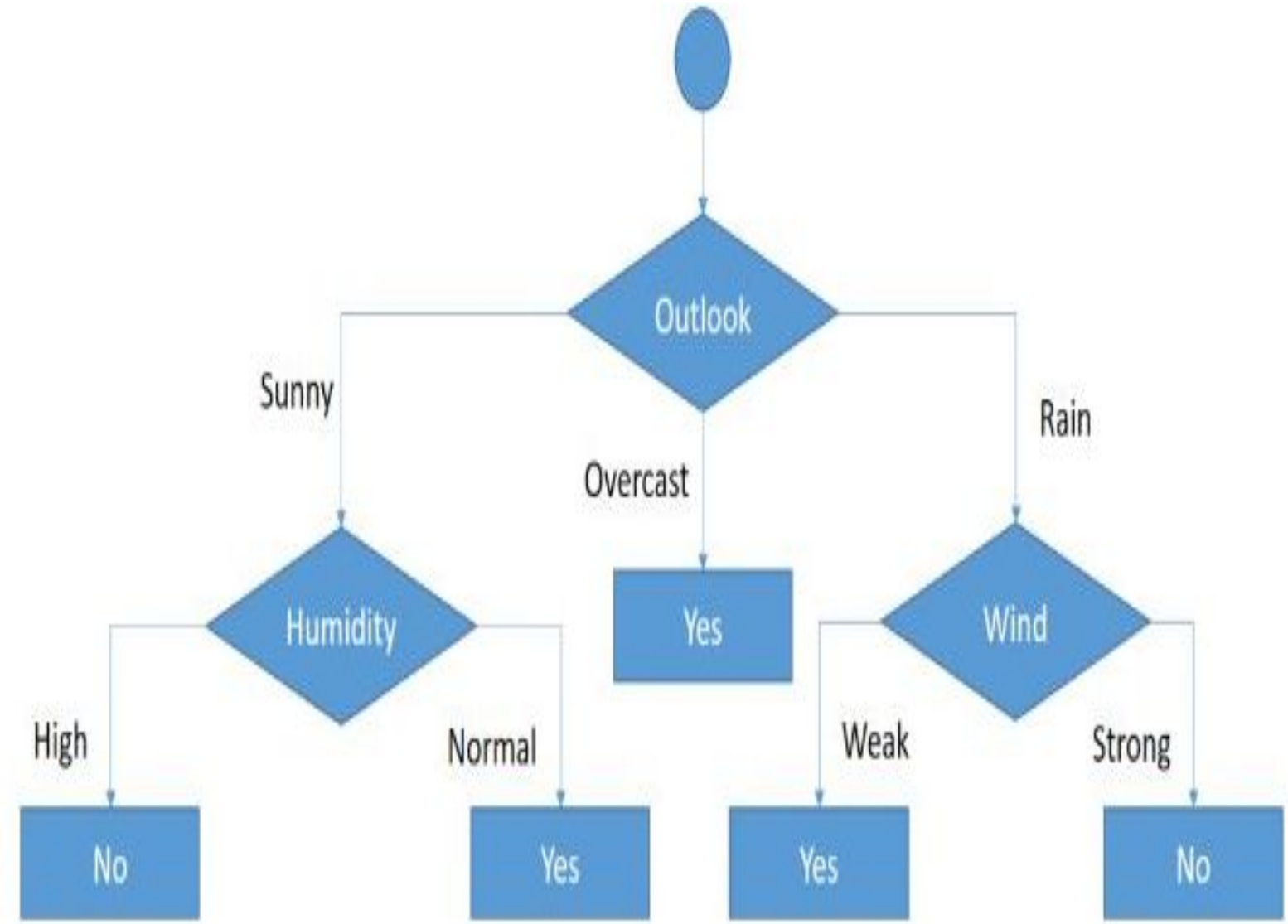
The winner is wind feature for rain outlook because it has the minimum gini index score in features.

Feature	Gini index
Temperature	0.466
Humidity	0.466
Wind	0



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No

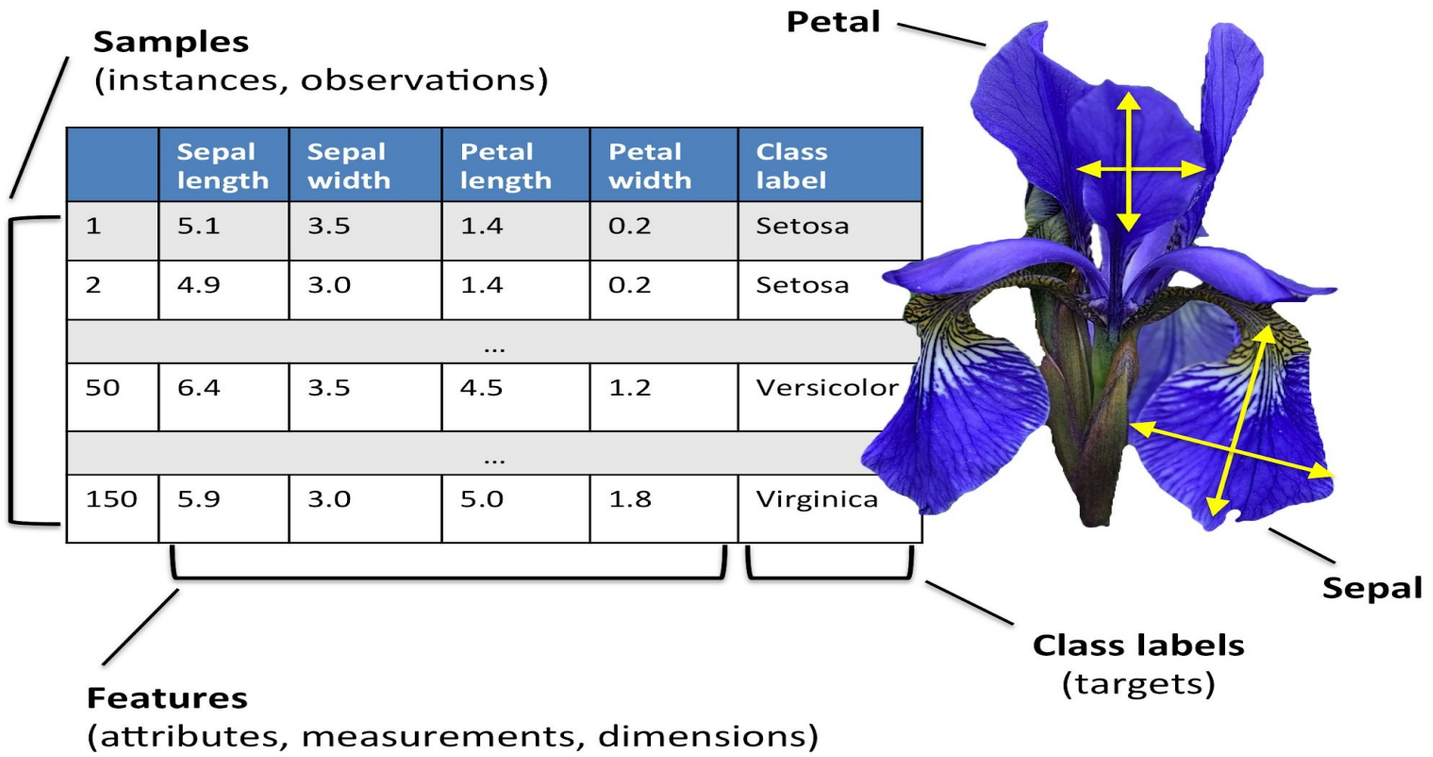


Final form of the decision tree built by CART algorithm

Training and Visualizing a Decision Tree

- Let’s just build a Decision Tree and take a look at how it makes predictions. We'll train a DecisionTreeClassifier using Scikit Learn on the famous Iris dataset.
- The iris dataset consists of 4 features namely :
  - Petal Length
  - Petal Width
  - Sepal Length
  - Sepal Width
- There are three classes namely :
  - Iris Setosa
  - Iris Versicolor
  - Iris Virginica

Iris Dataset - Training and Visualizing a Decision Tree



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

The iris dataset has 4 features petal length, petal width, sepal length and sepal width.

But here we'll only use two features i.e. petal length and petal width

Training and Visualizing a Decision Tree

We will follow the following steps to train and visualize our decision tree

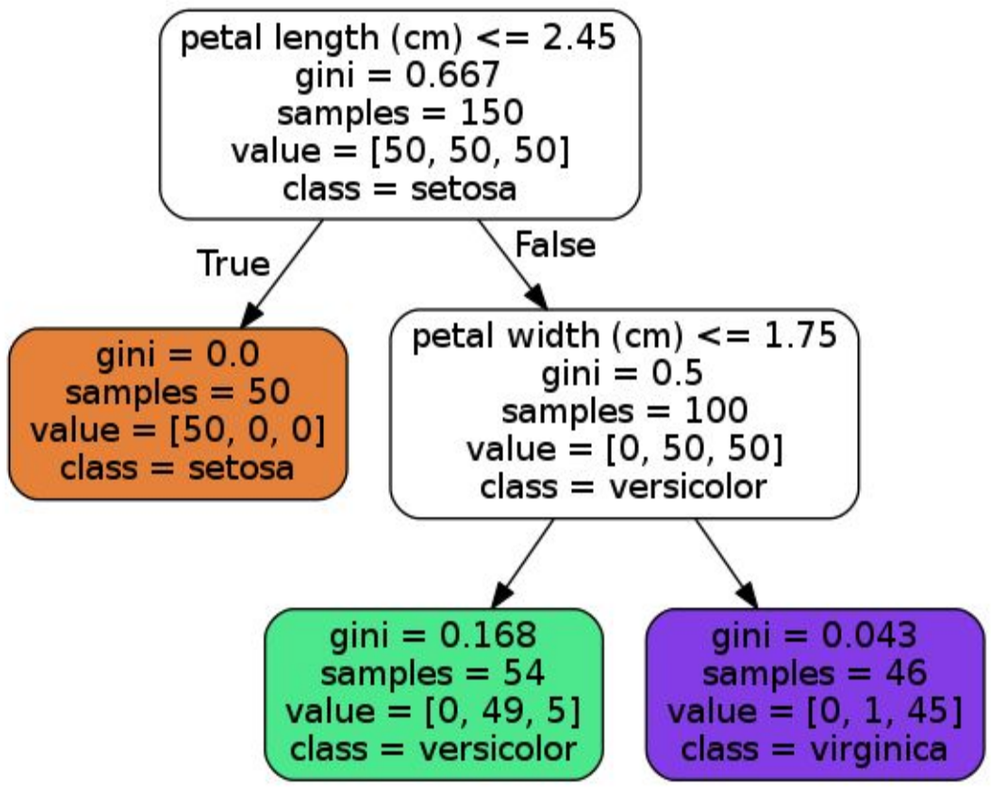
1. Load the Iris dataset using Scikit Learn
2. Select only the Petal length and Petal width features
3. Train our Decision Tree classifier on the Iris Dataset
4. Visualize our Decision Tree using export_graphviz()
5. export_graphviz() gives us a file in .dot format which we will convert to png using the dot command line tool

UNDERSTANDING THE DECISION TREE AND MAKING PREDICTIONS

- A node's value attribute tells you how many training instances of each class this node applies to for example, the bottom-right node applies to **0 Iris-Setosa, 1 Iris-Versicolor, and 45 Iris-Virginica**.

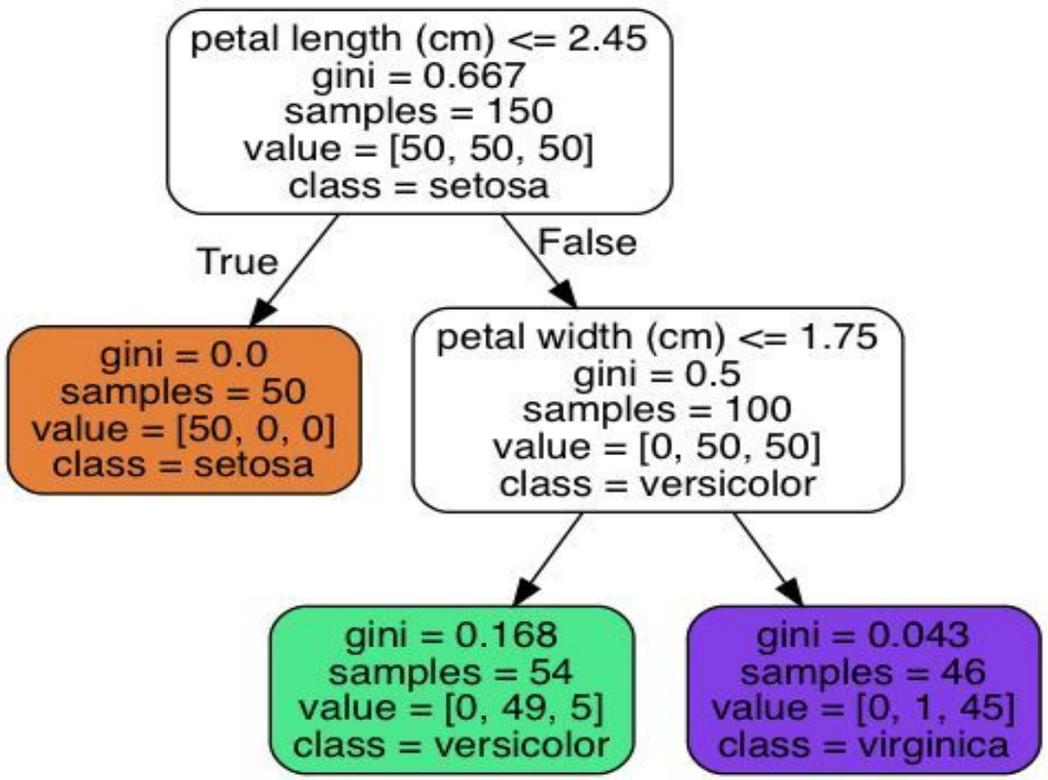
- A node's gini attribute measures its impurity: a node is "pure" (gini=0) if all training instances it applies to belong to the same class.

- For example, since the depth-1 left node applies only to Iris-Setosa training instances, it is pure and its gini score is 0.



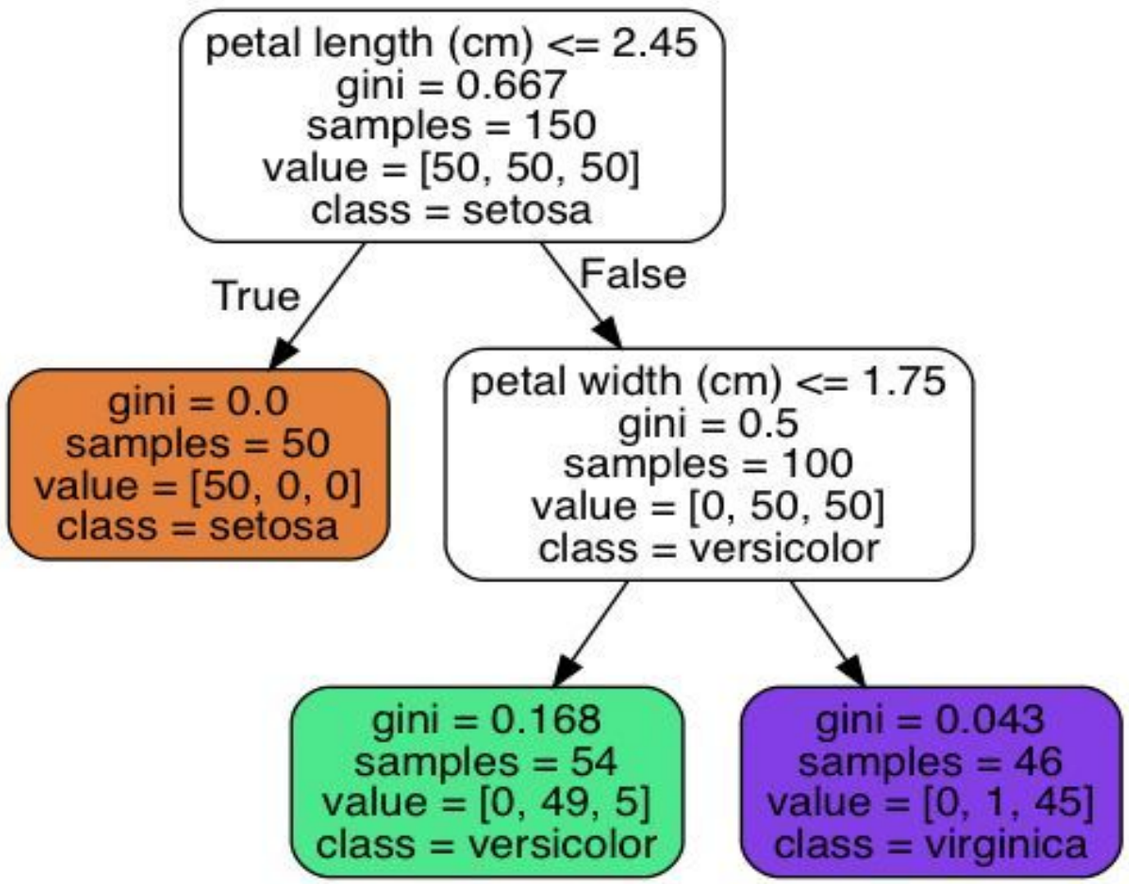
To make a prediction the decision classifier follows these steps :

- Start at the root node (depth 0, at the top), this node asks whether the flower's petal length is smaller than or equal to 2.45 cm:
- If it is, then you move down to the root's left child node (depth 1, left). In this case it is a leaf node hence the flower is predicted as setosa.
- If it is not, then you move down to the root's right child node (depth 1, right), since it is not a leaf node it asks further questions as, is the petal width smaller than or equal to 1.75 cm?
- If it is, then your flower is most likely an Iris- Versicolor (depth 2, left).
- If it is not, If not, it is likely an Iris-Virginica (depth 2, right).



- A **node's samples attribute** counts how many training instances it applies to
- For example, 100 training instances have a petal length greater than 2.45 cm (depth 1, right), and of those 100, 54 have a petal width smaller than 1.75 cm (depth 2, left).

- A **node's value attribute** tells you how many training instances of each class this node applies to: for example, the bottomright node applies to 0 Iris setosa, 1 Iris versicolor, and 45 Iris virginica.



- A node's gini attribute measures its *impurity*: a node is "pure" (gini=0) if all training instances it applies to belong to the same class.
- For example, since the depth-1 left node applies only to *Iris setosa* training instances, it is pure and its gini score is 0.
- A Gini coefficient of 1 expresses maximal inequality among the training samples.

Equation 6-1. Gini impurity

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

In this equation:

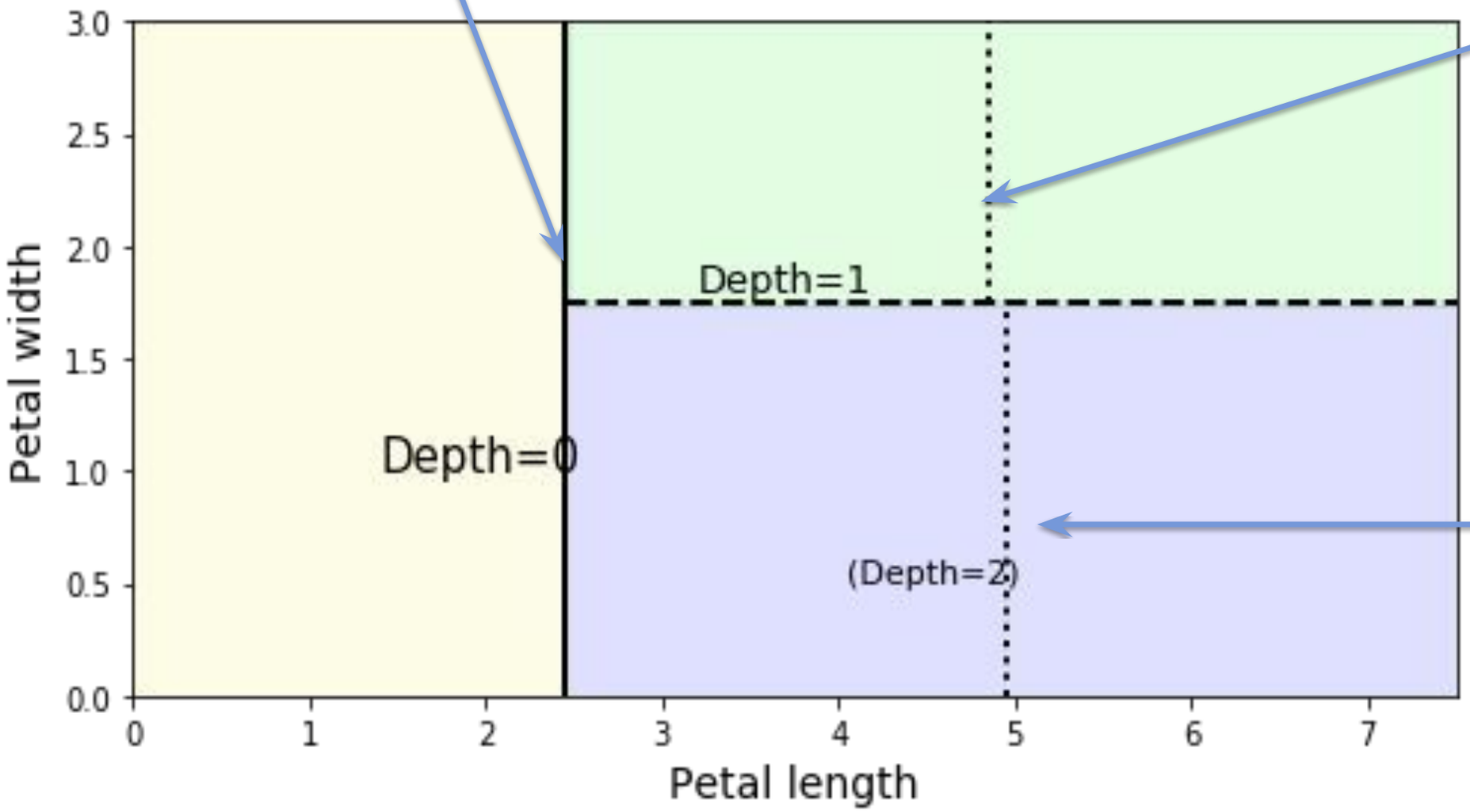
- $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node.

- The depth-2 left node has a gini score equal to

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

Decision Tree's decision boundaries.

Boundary made at depth 0. Petal length <= 2.45



Boundary made at depth 1. Petal width <= 1.75

This would have been the decision boundary to divide the tree further i.e. if max_depth was set to 3

Advantage

- Requires very little data preparation.
- Decision Tree uses **white box models –** Tree splitting is clear and inner working of these models are clearly understood.

Estimating Class Probabilities

A Decision Tree can also estimate the probability that an instance belongs to a particular class k.

To do this it follows the following steps:

- First it traverses the tree to find the leaf node for this instance
- 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 corresponding leaf node is the depth-2 left node, so 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).

It predicts the class as output Iris versicolor (class 1) because it has the highest probability.

Run it in jupyter notebook

The CART Training Algorithm

- Scikit-Learn uses the *Classification and Regression Tree* (CART) algorithm to train Decision Trees (also called "growing" trees).
- The algorithm works by first splitting the training set into two subsets using a single feature k and a threshold t_k (e.g., "petal length ≤ 2.45 cm").

Equation 6-2. CART cost function for classification

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

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

- Once the CART algorithm has successfully split the training set in two, it splits the subsets using the same logic, then the sub-subsets, and so on, recursively.
- It stops once it reaches the maximum depth(hyperparameter), or if it cannot find a split that will reduce impurity.
- A few other hyperparameters stopping conditions (min_samples_split, min_samples_leaf, min_weight_fraction_leaf, and max_leaf_nodes).

Important points on the CART Training Algorithm

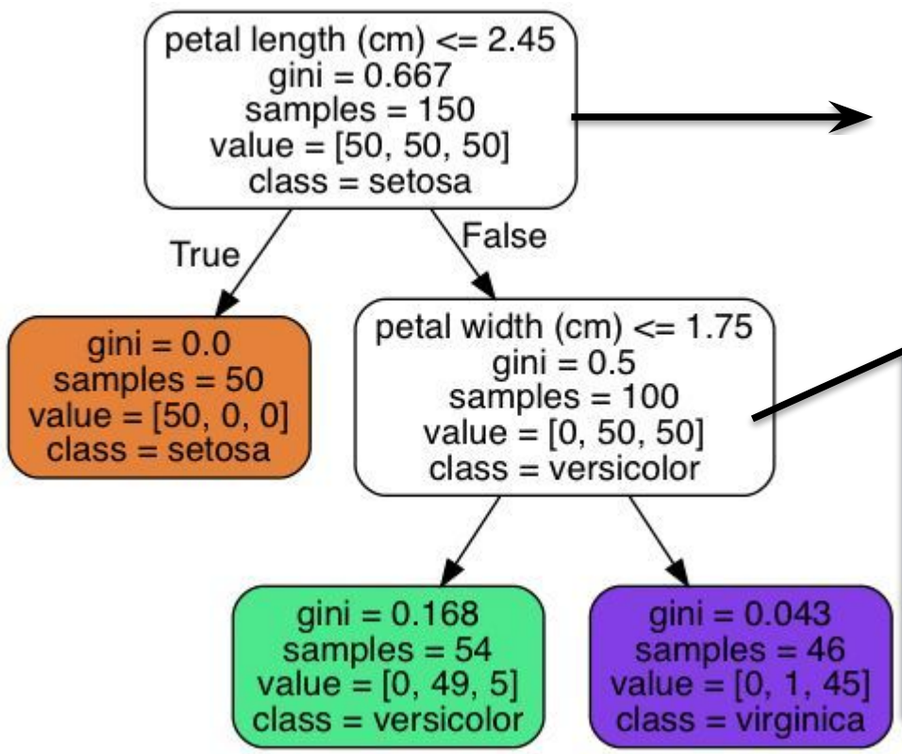
- It is a greedy algorithm as it greedily searches for an optimum split at the top level
- Then repeats the process at each level.
- It does not check whether or not the split will lead to the lowest possible impurity several levels down.
- A greedy algorithm often produces a reasonably good solution, but it is not guaranteed to be the optimal solution

Computational Complexity of Decision Trees

- Making predictions requires traversing the Decision Tree from the root to a leaf.
- Decision Trees are generally approximately balanced, so traversing the Decision Tree requires going through roughly O(log2(m)) nodes, where m is total number of training instances.

Complexity of Prediction :

- Since each node only requires checking the value of one feature, the overall prediction complexity is just O(log2(m))



One feature checked i.e. petal length

One feature checked i.e. petal width

Hence, the complexity of prediction is independent of the number of features. So predictions are very fast, even when dealing with large training sets.

- The training algorithm compares all features on all samples at each node.
- This results in a training complexity of O(n × m log(m)), where n is the number of features, we have to compare all the n features at each of the m nodes.

Which measure to use ? Gini Impurity or Entropy?

- By default, the Gini impurity measure is used.
- But if we want to select the *entropy* impurity measure by setting the hyperparameter to "entropy".
- The concept of entropy originated in thermodynamics as a measure of molecular disorder: entropy approaches zero when molecules are still and well ordered.
- In Machine Learning, entropy is frequently used as an impurity measure: a set’s entropy is zero when it contains instances of only one class

- The definition of the entropy of the *ith* node.

Equation 6-3. Entropy

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

- Entropy equal to

$$-\frac{49}{54}\log_2\left(\frac{49}{54}\right) - \frac{5}{54}\log_2\left(\frac{5}{54}\right) \approx 0.445.$$

- Gini impurity or entropy?
  - Most of the time it does not make a big difference: they lead to similar trees.
  - Gini impurity is slightly faster to compute, so it is a good default.

Regularization Hyperparameters

- Decision Trees make very few assumptions about the training data. If left unconstrained, the tree
- structure will adapt itself to the training data and results into overfitting.
- Such a model is often called a *nonparametric model*, not because it does not have any parameters but because the number of parameters is not determined prior to training, so the model structure is free to stick closely to the data.
- In contrast, a *parametric model*, such as a linear model, has a predetermined number of parameters, so its degree of freedom is limited, reducing the risk of overfitting.

- To avoid overfitting the training data, you need to restrict the Decision Tree's freedom during training with the concept of regularization.
- The regularization hyperparameters depend on the algorithm used, but generally you can at least restrict the maximum depth of the Decision Tree.
- In Scikit-Learn, this is controlled by the max_depth hyperparameter (the default value is None, which means unlimited). Reducing max_depth will regularize the model and thus reduce the risk of overfitting.

- The DecisionTreeClassifier class has a few other parameters that similarly restrict the shape of the Decision Tree:
  - **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.
  - **max_features**- the maximum number of features that are evaluated for splitting at each node.

- Figure 6-3 shows two Decision Trees trained on the moons dataset.
- On the left the Decision Tree is trained with the default hyperparameters (i.e., no restrictions), and on the right it's trained with min_samples_leaf=4. It is quite obvious that the model on the left is overfitting, and the model on the right will probably generalize better.

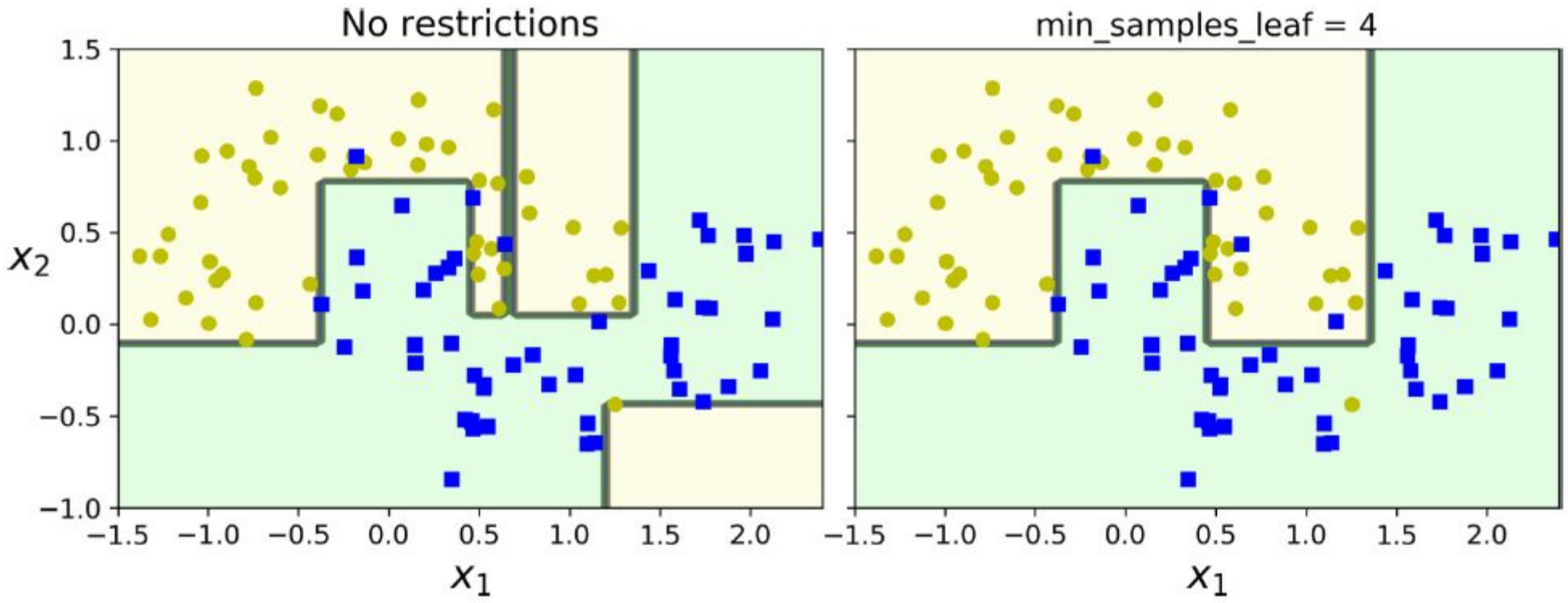


Figure 6-3. Regularization using `min_samples_leaf`