20 Decision Trees: From Simple Splits to Smart Predictions

20.1 Growing the Tree: How Decision Trees Split

20.1.1 From Questions to Predictions

A decision tree is a widely used machine-learning model that predicts outcomes by repeatedly splitting data into smaller and more homogeneous groups. It can be applied to both classification (predicting a category) and regression (predicting a numerical value). Decision trees are popular because they are intuitive, easy to interpret, and simple to visualize. However, if a tree becomes too deep or too wide, it may memorize the training data and overfit, reducing its ability to generalize to new samples. Later in this chapter, we will introduce pruning and other techniques to prevent this problem.

A decision tree is composed of nodes and branches. As shown in Figure 1, a tree has three types of nodes: a root node at the top, internal (parent) nodes in the middle, and leaf (child) nodes at the bottom. Each internal node represents a decision based on a single feature, and each branch represents the outcome of that decision. The leaf nodes contain the final prediction.

Every split divides the data into two subgroups (left and right), gradually forming smaller regions of the feature space. In Figure 1, the tree has four leaf nodes. The number of leaf nodes and the depth (number of layers from root to leaves) are important hyperparameters. Limiting either of them controls the tree's size and helps prevent overfitting.

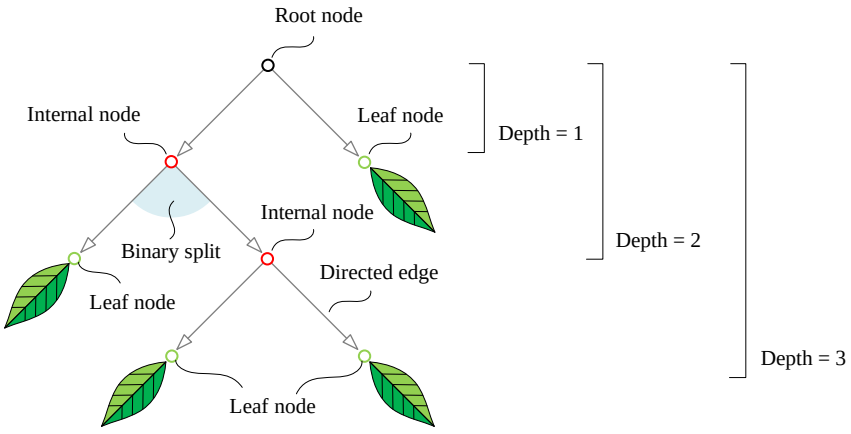

Figure 1. Basic Structure of a Decision Tree

20.1.2 Anatomy of a Tree

To classify a sample, a decision tree repeatedly applies simple yes/no questions based on feature thresholds.

In Figure 2, the root node performs the first split: samples with $x_1 \geqslant a$ go to the right subtree, while samples with $x_1 < a$ go to the left subtree. After this step, the data is separated into two regions, A and B. Region A is dominated by class C_1 (red points), and Region B is dominated by class C_2 (blue points).

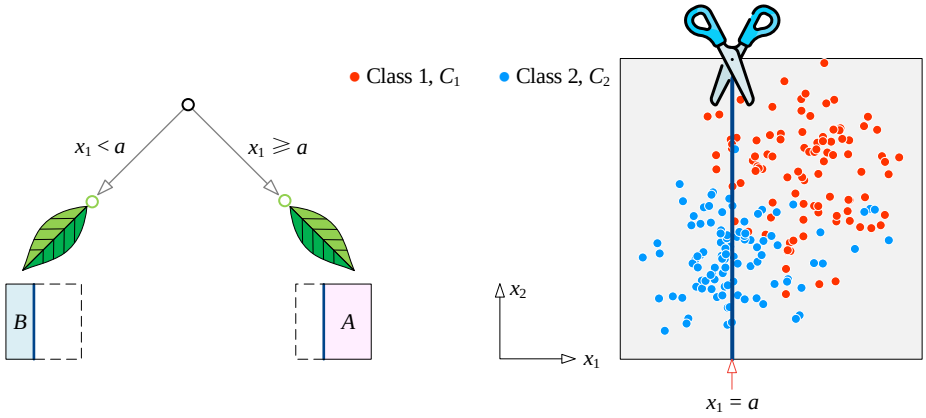

Figure 2. First Split of the Decision Tree

The process continues. In Figure 3, the right subtree grows a new split: samples with $x_2 \geqslant b$ go to the right branch, and those with $x_2 < b$ go to the left. Region A now separates into two smaller regions, C and D, each more homogeneous than before.

This step-by-step splitting continues until the tree reaches its stopping rules. While the tree can grow very large to fit complex data, an overly complex tree easily overfits. Therefore, we later apply pruning or set limits such as maximum depth or maximum leaf nodes to keep the model balanced.

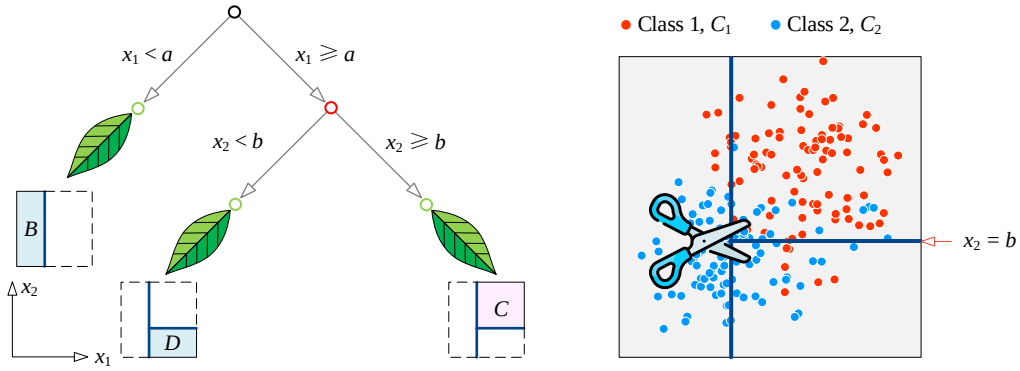

Figure 3. Second Split of the Decision Tree

At this point, you may wonder: *how do we choose the best split at each node?* For example, how are the thresholds (a) in Figure 2 and (b) in Figure 3 selected? That will be the topic of the next section.

20.2 Measuring Purity: The Language of Uncertainty

20.2.1 Shannon's Entropy: Order and Disorder in Data

At each internal node, a decision tree must choose which feature to split on and where to place the threshold. For example, in the earlier Figures, we chose thresholds a and b on features x_1 and x_2. But how are these values determined in practice? To answer this, we need a way to measure how "pure" or "impure" a group of samples is.

The goal of every split is to make the resulting child nodes as pure as possible—meaning each node should contain predominantly one class. To quantify this, decision trees use metrics such as information entropy, information gain, and the Gini index. We begin with the most classical concept: information entropy.

In information theory, entropy measures the level of uncertainty in a random variable. The higher the entropy, the more unpredictable the outcome; the lower the entropy, the more predictable it is. Claude Shannon introduced information entropy in 1948, and it has since become a fundamental tool in machine learning.

For a dataset Ω containing samples from K classes, the entropy is defined as:

$$\mathrm{Ent}(\Omega) = -\sum_{k=1}^{K} p_k \log_2 p_k \tag{1}$$

where p_k is the proportion (probability) of class C_k in the dataset. By convention, when $p_k = 0$, we define $p_k \log_2 p_k = 0$ to avoid undefined expressions.

20.2.2 Entropy in Action: The Two-Class Example

If there are only two classes ($K = 2$), we can write:

$$p_1 = p, \quad p_2 = 1 - p \tag{2}$$

and the entropy becomes:

$$\begin{aligned} \mathrm{Ent}(\Omega) &= -\sum_{k=1}^{2} p_k \log_2 p_k = -\left(p_1 \log_2 p_1 + p_2 \log_2 p_2\right) \\ &= -p \log_2 p - (1-p)\log_2 (1-p) \end{aligned} \tag{3}$$

This function describes how uncertainty changes as the class ratio changes. As illustrated in Figure 4, when one class dominates completely ($p = 0$ or $p = 1$), the dataset is perfectly pure and the entropy is zero—there is no uncertainty.

In contrast, when the classes are evenly split ($p = 0.5$), entropy reaches its maximum, meaning the dataset is highly mixed and most uncertain.

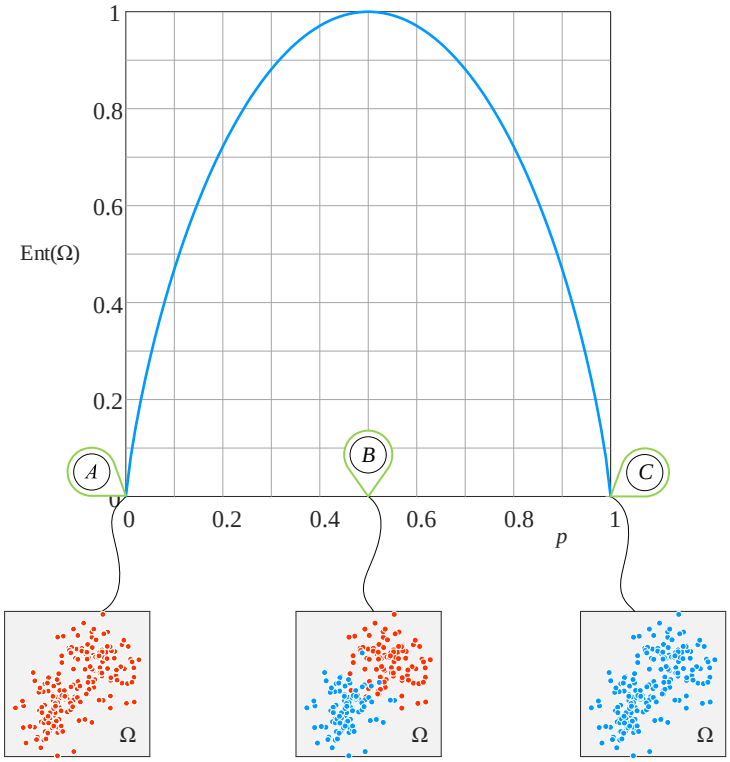

Figure 4. Entropy curve for a two-class dataset

20.2.3 Generalizing to Multiple Classes

If a dataset Ω contains samples from *K* categories:

$$\Omega = \{C_1, C_2, \ldots, C_K\} \tag{4}$$

and the total sample count satisfies:

$$\sum_{k=1}^{K} \text{count}(C_k) = \text{count}(\Omega) \tag{5}$$

then the class probability is:

$$p_k = \frac{\text{count}(C_k)}{\text{count}(\Omega)} \tag{6}$$

Substituting this into Shannon's formula gives the general expression for entropy used in decision trees:

$$\text{Ent}(\Omega) = -\sum_{k=1}^{K} p_k \log_2 p_k = -\sum_{k=1}^{K} \left\{ \frac{\text{count}(C_k)}{\text{count}(\Omega)} \log_2 \left(\frac{\text{count}(C_k)}{\text{count}(\Omega)} \right) \right\} \tag{7}$$

Entropy gives us a numerical way to evaluate how mixed a node is. A good split is one that reduces entropy as much as possible, producing child nodes that are purer than the parent. In the next section, we will use entropy to define information gain, the key quantity that decision trees maximize when choosing the optimal split.

20.3 Making the Best Split: Information Gain

20.3.1 The Idea Behind Information Gain

Entropy gives us a way to measure how mixed a node is. When we split a dataset, we hope that the resulting child nodes become purer—that is, more certain about their class labels. Information gain captures exactly this idea: it measures how much uncertainty is reduced by a particular split.

Suppose a feature a splits the dataset Ω into m subsets:

$$\Omega = \{\Omega_1, \Omega_2, ..., \Omega_m\} \tag{8}$$

For each subset Ω_j, we can calculate its entropy. The entropy of a subset is defined in the same way as before:

$$\text{Ent}(\Omega_j) = -\sum_{k=1}^{K} p_{j,k} \log_2 p_{j,k} \tag{9}$$

where $p_{j,k}$ is the proportion of class C_k inside subset Ω_j.

Once the split is made, the total entropy after division is the weighted sum of the entropies of all subsets:

$$\underbrace{\text{Ent}(\Omega|a)}_{\text{Weighted sum of entropy after split}} = \sum_{j=1}^{m} \left\{ \frac{\text{count}(\Omega_j)}{\text{count}(\Omega)} \text{Ent}(\Omega_j) \right\} \tag{10}$$

This represents the remaining uncertainty after splitting on feature a.

20.3.2 Calculating Information Gain Step by Step

In Figure 5, the dataset Ω contains two classes, C_1 and C_2. Feature a divides the dataset into two subsets, Ω_1 and Ω_2.

If the proportion of class C_1 in Ω_1 is

$$p_{1,1} = \frac{\text{count}(\Omega_{1,1})}{\text{count}(\Omega_1)} \tag{11}$$

then the entropy of Ω_1 is

$$\text{Ent}(\Omega_1) = -\frac{\text{count}(\Omega_{1,1})}{\text{count}(\Omega_1)} \log_2 \left(\frac{\text{count}(\Omega_{1,1})}{\text{count}(\Omega_1)} \right) - \frac{\text{count}(\Omega_{1,2})}{\text{count}(\Omega_1)} \log_2 \left(\frac{\text{count}(\Omega_{1,2})}{\text{count}(\Omega_1)} \right) \tag{12}$$

Similarly, we can compute Ent(Ω_2).

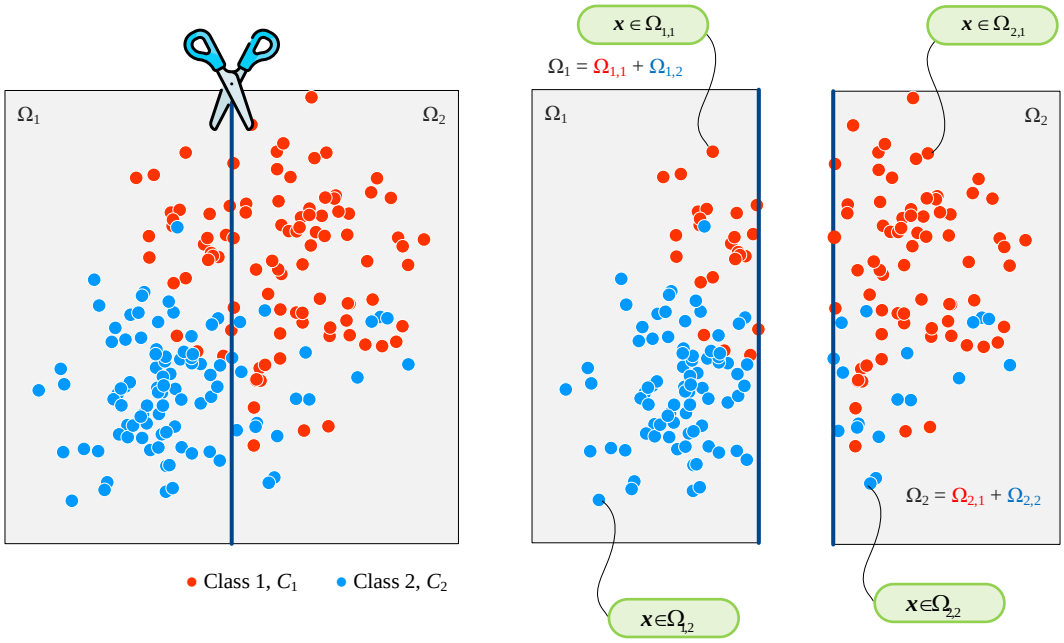

Figure 5. A feature splits dataset Ω into two subsets

20.3.3 Choosing the Optimal Split

Information gain measures how much the entropy decreases after a split. It is defined as:

$$\text{Gain}(\Omega, a) = \underbrace{\text{Ent}(\Omega)}_{\substack{\text{Entropy} \\ \text{before split}}} - \underbrace{\text{Ent}(\Omega|a)}_{\substack{\text{Weighted sum of} \\ \text{entropy after split}}} \tag{13}$$

A good split is one that makes the child nodes purer, which means it produces a larger information gain. Therefore, the decision tree chooses the threshold and feature that maximize:

$$\arg\max_{a} \text{Gain}(\Omega, a) \tag{14}$$

In other words, the best split is the one that reduces uncertainty the most.

20.4 Gini Index: A Simpler View of Impurity

20.4.1 What is the Gini Index?

Just like entropy, the Gini index provides a way to measure how mixed or impure a dataset Ω is. Although it shares the same spirit as entropy, the Gini index is mathematically simpler and is widely used in decision tree algorithms. Note that this Gini index is not the same as the economic Gini coefficient used to measure income inequality.

For a dataset Ω with K classes, the Gini index is defined as:

$$\text{Gini}(\Omega) = \sum_{i=1}^{K} p_i (1 - p_i) = 1 - \sum_{i=1}^{K} p_i^2 \tag{15}$$

where p_k is the proportion of samples that belong to class C_k.

The Gini index becomes smaller when one class dominates (high purity) and becomes larger when the classes are evenly mixed (low purity).

20.4.2 Comparing Gini and Entropy

When there are only two classes, we can write:

$$p_1 = p, \quad p_2 = 1 - p \tag{16}$$

and the Gini index becomes:

$$\mathrm{Ent}(D) = 2p(1-p) \tag{17}$$

As shown in Figure 6 (a), the Gini index reaches its maximum at $p = 0.5$, where the two classes are equally represented and the uncertainty is highest.

When $p = 0$ or $p = 1$, the node is perfectly pure, and the Gini index drops to zero.

Figure 6 (b) compares the shapes of the Gini curve and the entropy curve. Both metrics capture the same intuition—mixed data has higher impurity—but the Gini index changes more smoothly and is slightly easier to compute.

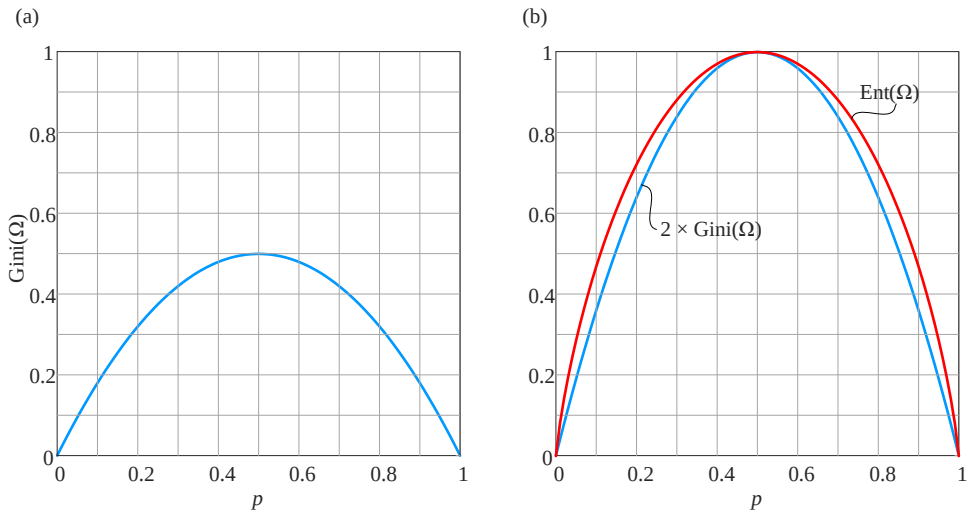

Figure 6. Comparison of Entropy and Gini impurity curves

In practice, decision trees look for splits that reduce the Gini index as much as possible, making child nodes purer than the parent. In fact, the DecisionTreeClassifier in scikit-learn uses the Gini index as its default splitting criterion.

20.5 Controlling the Shape: The Number of Leaf Nodes

20.5.1 Setting a Limit: max_leaf_nodes

In this section, we use the decision tree algorithm to classify the Iris dataset and examine how the maximum number of leaf nodes affects the model's decision boundary.

In scikit-learn, we control this setting using the parameter max_leaf_nodes in sklearn.tree.DecisionTreeClassifier. Throughout the discussion, we will also visualize the resulting trees using sklearn.tree.plot_tree.

20.5.2 Visual Case Studies on the Iris Dataset

Case a): 2 Leaf Nodes — The Simplest Split

Figure 7 shows the classification result when the tree is restricted to only two leaf nodes. In Figure 7(a), the decision boundary is formed using only one feature: the sepal length x_1. The feature space is divided into two large regions, *A* and *B*, corresponding to the two leaf nodes.

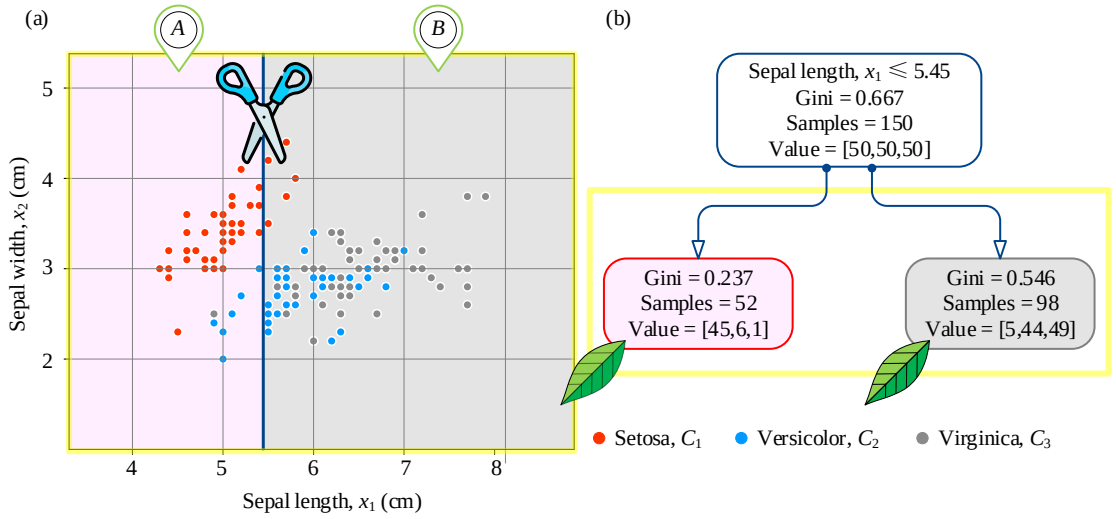

Figure 7. Decision boundary and tree structure when the maximum number of leaf nodes is 2. Figure generated by Ch20_01_Decision_tree.ipynb.

Figure 7 (b) shows the corresponding decision tree.

The root node contains all 150 iris samples and has a Gini impurity of 0.667.

The best split is found at x_1 =5.45. Samples with $x_1 \leqslant 5.45$ fall into Region *A*, and samples with $x_1 > 5.45$ fall into Region *B*.

Region A contains mostly class C_1, resulting in a relatively low Gini value of 0.237. Region B contains a mix of C_2 and C_3, with a higher Gini value of 0.546.

Using equation (10), we can compute the entropy after the split,

$$\underbrace{\mathrm{Ent}\left(\Omega \middle| x_1 = 5.45\right)}_{\substack{\text{Weighted sum of}\\ \text{entropy after split}}} = \frac{52}{150} \times 0.237 + \frac{98}{150} \times 0.546 = 0.4389 \tag{18}$$

and using equation (13), we obtain the information gain

$$\mathrm{Gain}\left(\Omega, a\right) = \underbrace{\mathrm{Ent}\left(\Omega\right)}_{\substack{\text{Entropy}\\ \text{before split}}} - \underbrace{\mathrm{Ent}\left(\Omega \middle| x_1 = 5.45\right)}_{\substack{\text{Weighted sum of}\\ \text{entropy after split}}} = 0.667 - 0.4389 = 0.228 \tag{19}$$

Although this tree is extremely simple, it creates a very coarse decision boundary that cannot cleanly separate all three Iris classes.

Case b): 3 Leaf Nodes — Refining the Boundary

If we increase the limit to three leaf nodes, the decision tree grows one additional split. As shown in Figure 8 (a), Region *B* from the previous case is further divided along the same feature x_1, this time at x_1 =6.15. This produces two new regions, *C* and *D*.

Region *C* still contains a mixture of classes but is dominated by C_2, while Region *D* is dominated by C_3. Compared to Figure 8, the decision boundary becomes more detailed, and both child nodes show reduced impurity.

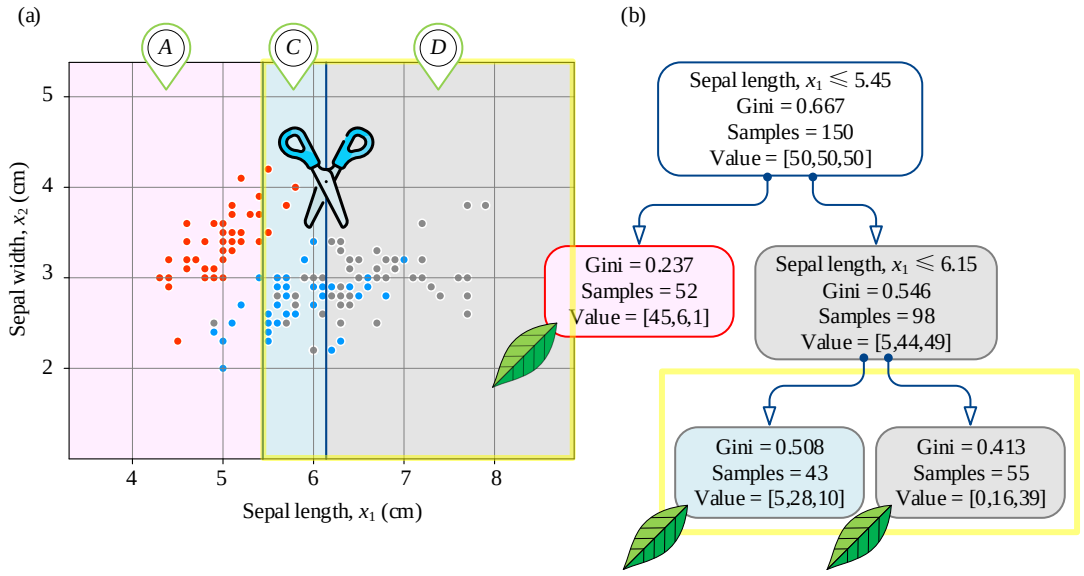

Figure 8. Decision boundary and tree structure when the maximum number of leaf nodes is 3. Figure generated by Ch20_01_Decision_tree.ipynb.

Case c): 4 Leaf Nodes — Adding Another Dimension

Figure 9 shows the result when the number of leaf nodes is increased to four. This time, one of the earlier regions is further split along a different feature, x_2 (petal width). One of the resulting partitions contains 44 samples of a single class, reducing the Gini impurity of that leaf node to 0.043—an improvement in purity over the previous case.

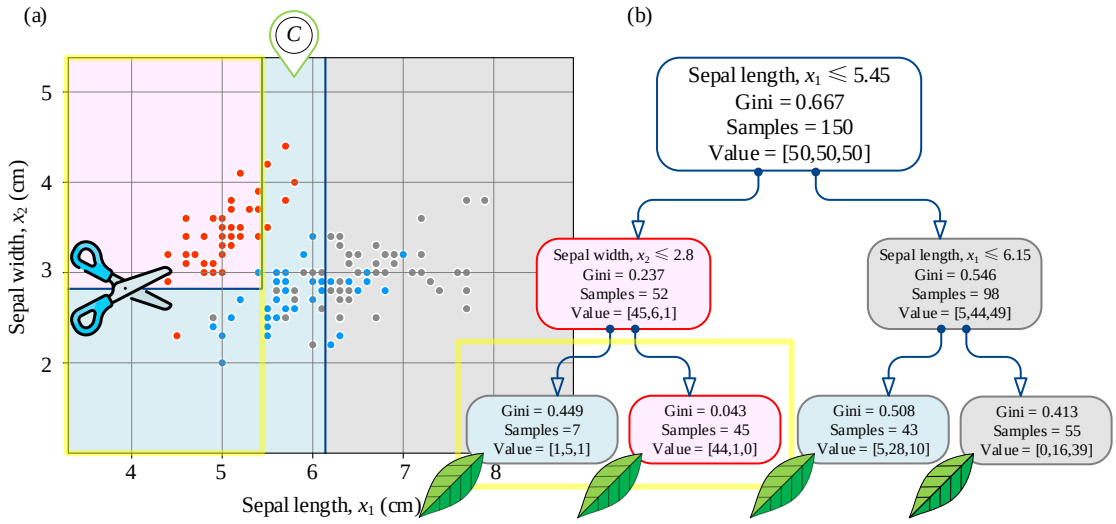

Figure 9. Decision boundary and tree structure when the maximum number of leaf nodes is 4. Figure generated by Ch20_01_Decision_tree.ipynb.

Case d): 5 Leaf Nodes — Approaching Purity

When we increase the limit to five leaf nodes, as shown in Figure 10, another region is split along x_2, and one of the resulting nodes becomes perfectly pure, containing only class C_1. Its Gini impurity is therefore 0. The decision boundary becomes even more refined compared with Figure 9.

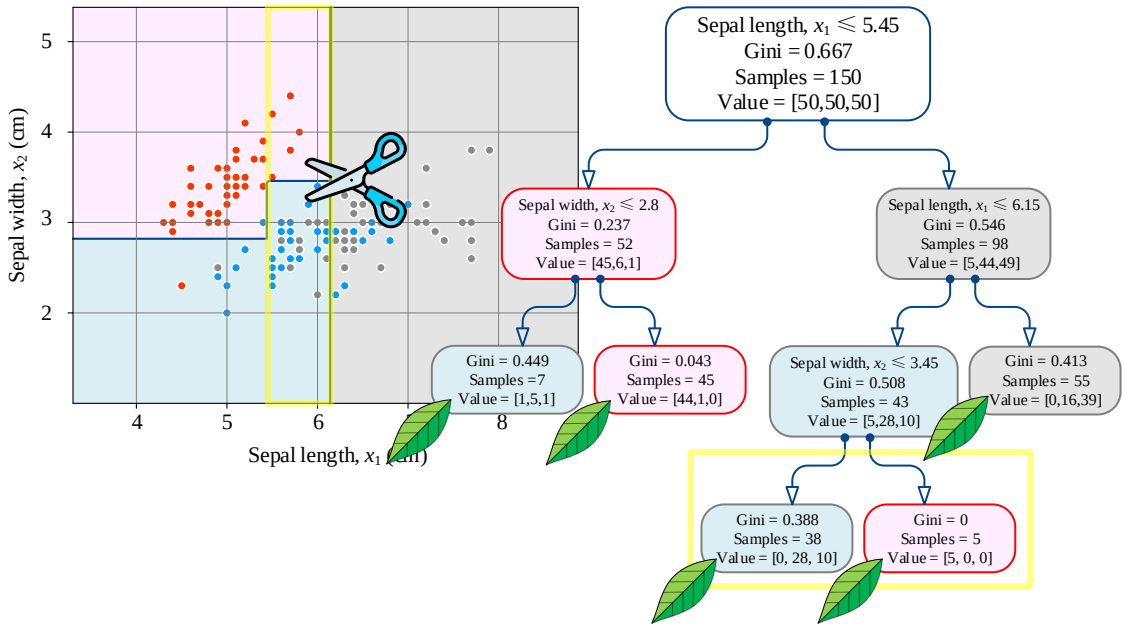

Figure 10. Decision boundary and tree structure when the maximum number of leaf nodes is 5. Figure generated by Ch20_01_Decision_tree.ipynb.

As we increase max_leaf_nodes, the tree becomes more expressive and the decision boundary becomes more detailed. However, a larger number of leaf nodes also increases the risk of overfitting, because the model may begin to memorize the training data rather than learn generalizable patterns.

20.6 Controlling Depth: How Deep Should the Tree Grow?

20.6.1 Understanding max_depth

Another important parameter that influences the structure of a decision tree is its maximum depth, which specifies how many levels the binary tree is allowed to grow. A shallow tree has fewer splits and therefore simpler decision rules, while a deeper tree can capture more complex patterns in the data. In scikit-learn, the DecisionTreeClassifier controls this setting through the max_depth parameter.

20.6.2 Visualizing the Effect of Depth

Case a): Depth = 1: One Big Split

To illustrate how the maximum depth affects model behavior, consider the Iris dataset. When max_depth = 1, the tree is only allowed to make a single split. As shown in Figure 11, the model produces a very coarse decision boundary, and the corresponding tree in Figure 11 (right) contains just one internal node and two leaf nodes. Although overly simple, the predictions in the plot and the structure of the tree match perfectly.

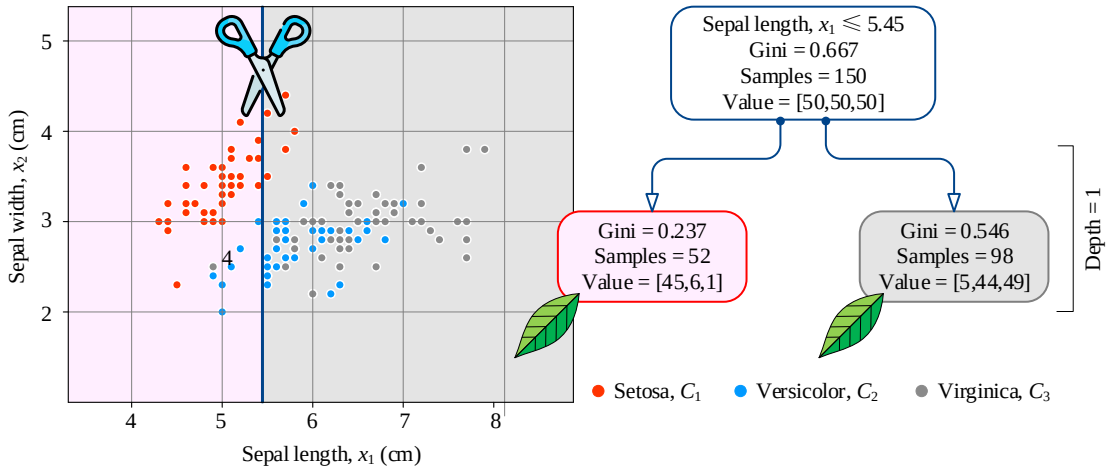

Figure 11. Iris dataset classification result with max_depth = 1 and its corresponding tree. Figure generated by Ch20_01_Decision_tree.ipynb.

Case b) Depth = 2: More Nuanced Decisions

Increasing the depth to max_depth = 2 allows one more layer of splitting. As shown in Figure 12, the resulting decision regions become more refined, and the tree structure reflects this growth by adding an additional layer. Again, the visual classification result is fully consistent with the tree.

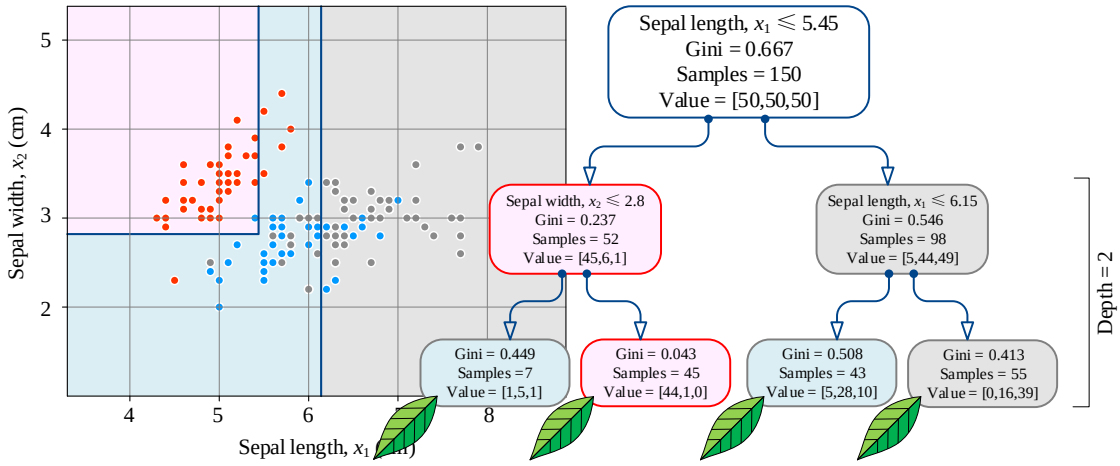

Figure 12. Iris dataset classification result with max_depth = 2 and its corresponding tree. Figure generated by Ch20_01_Decision_tree.ipynb.

Case c) Depth = 3: Capturing Complex Boundaries

When we further increase to max_depth = 3, the tree gains even more flexibility. Figure 13 shows that the model can now capture more detailed boundaries between flower species, and the corresponding tree in Figure 14 has three levels of splits. It is worth noting that even if we allow deeper trees, splitting will stop automatically in any region where all remaining samples belong to the same class. In such cases, the Gini index becomes zero and no further division is possible. This often results in some leaf nodes already achieving perfect purity before the maximum depth is reached.

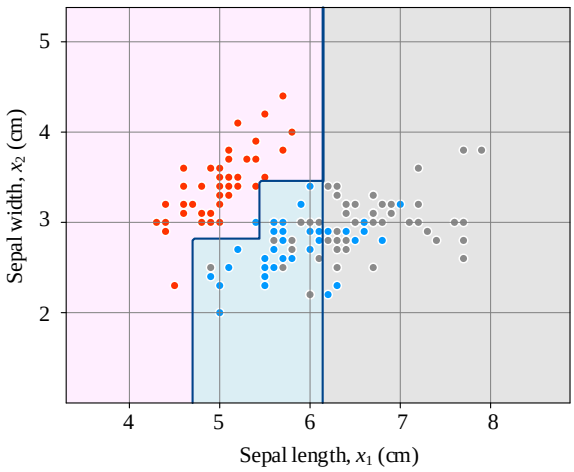

Figure 13. Iris dataset classification result with max_depth = 3. Figure generated by Ch20_01_Decision_tree.ipynb.

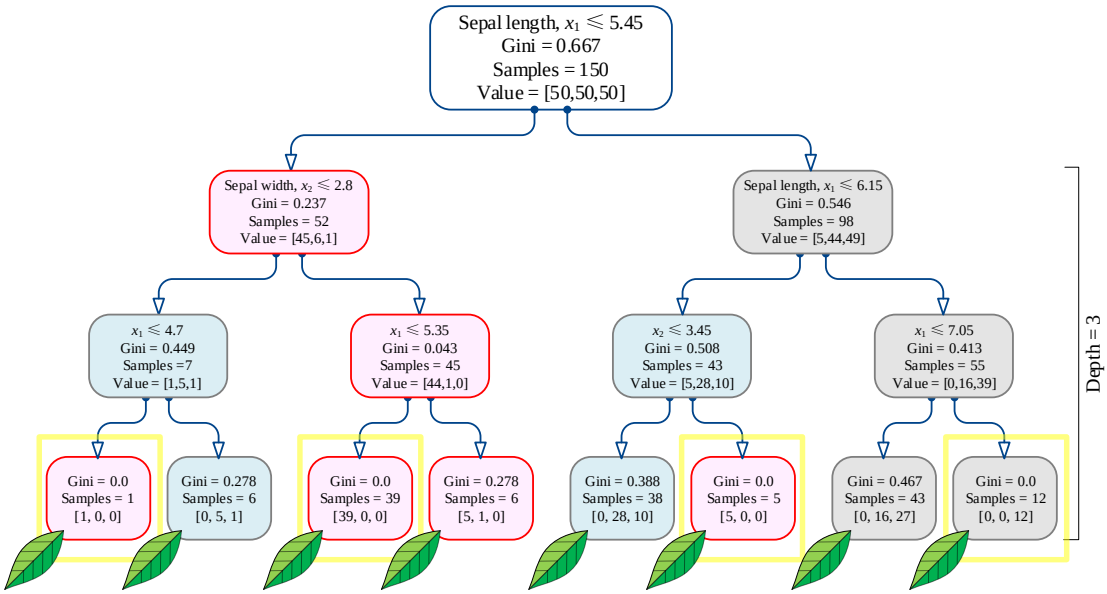

Figure 14. Decision tree structure for max_depth = 3. Figure generated by Ch20_01_Decision_tree.ipynb.

20.7 Conclusion

This chapter introduced decision trees, a widely used machine-learning model for both classification and regression tasks. A decision tree makes predictions by recursively splitting data into increasingly homogeneous subsets using simple decision rules. We first built intuition for how a tree grows from a root node to internal nodes and finally to leaf nodes, where predictions are made.

To choose good splits, the tree measures how mixed a node is using impurity metrics such as entropy and the Gini index. A split is considered effective if it reduces impurity, and information gain provides a way to quantify this improvement. We also compared entropy and the Gini index, noting that both serve the same purpose, although Gini is the default in many implementations due to its simplicity.

Next, we explored how hyperparameters control model complexity. Limiting the number of leaf nodes or the maximum tree depth helps prevent overfitting, which can occur when a tree becomes too large and memorizes the training data. Through visual examples on the Iris dataset, we observed how these parameters affect decision boundaries and tree structures.

Overall, decision trees are intuitive, interpretable, and powerful, especially when combined with proper regularization techniques.