Chapter 6. Decision Trees

Versatile: both classification and regression, even multioutput.

Training and Visualizing a Decision Tree

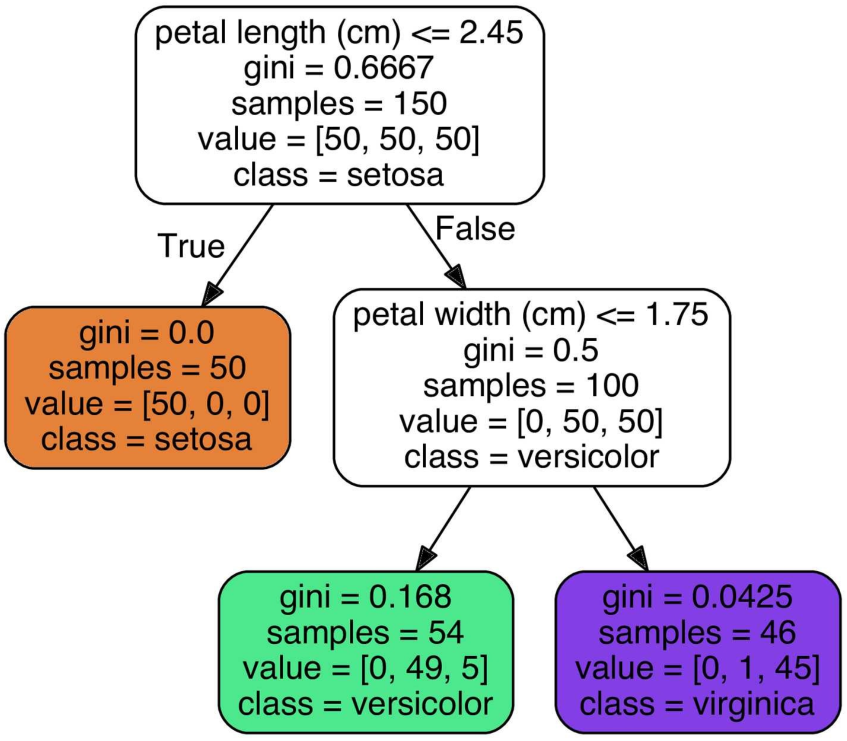

Figure 6-1. Iris Decision Tree

Attributes:

- *sample*: counts how many training instances it applies to.
- *value*: how many training instances of each class this node applies to.
- *gini*: measures its impurity

  Gini impurity

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

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

Making Predictions

Root node: depth 0, at the top.

Leaf node: it doesn't have any children nodes.

NOTE

> One of the many qualities of Decision Trees is that they require very little data preparation. In particular, they don't require feature scaling or centering at all.

White box models Vs *black box models* (Decision Trees Vs Random Forests, neural networks)

div style="width:400 px; font-size:100%; text-align:center;">

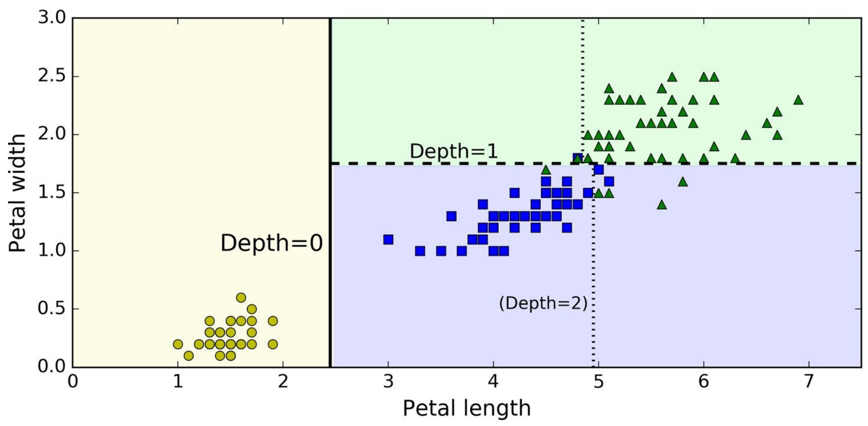

Figure 6-2. Decision Tree decision boundaries

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.

Notice that the estimated probabilities would be identical with different feature values within same decision boundaries.

The CART Training Algorithm

Scikit-Learn uses the *Classification and Regression Tree* (CART) algorithm to train Decision Trees ("grow" trees).

The idea is really quite simple: the algorithm splits the training set in two subsets using a single feature k and a threshold t_k. It searches for the pair (k, t_k) that produces the purest subsets (weighted by their size).

CART cost function for classification

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

where $G_{left/right}$ measures the impurity of the left/right subset; $m_{left/right}$ is the number of instances in the left/right subset.

WARNING

> As you can see, the CART algorithm is a *greedy algorithm*: 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.

Unfortunately, finding the optimal tree is known to be an *NP-Complete* problem: it requires $O(exp(m))$ time, making the problem intractable even for fairly small training sets. This is why we must settle for a "reasonably good" solution.

P is the set of problems that can be solved in polynomial time. NP is the set of problems whose solutions can be verified in polynomial time. An NP-Hard problem is a problem to which any NP problem can be reduced in polynomial time. An NP-Complete problem is both NP and NP-Hard. A major open mathematical question is whether or not $P = NP$. If $P \neq NP$ (which seems likely), then no polynomial algorithm will ever be found for any NP-Complete problem (except perhaps on a quantum computer).

Computational Complexity

Decision Trees are generally approximately balanced, so traversing the Decision Tree requires going through roughly $O(log_2(m))$ nodes. Since each node only requires checking the value of one feature, the overall prediction complexity is just $O(log_2(m))$, independent of the number of features. So predictions are very fast, even when dealing with large training sets.

However, the training algorithm compares all features (or less if max_features is set) on all samples at each node. This results in a training complexity of $O(n \times m \; log_2(m))$. For small training sets (less than a few thousand instances), Scikit-Learn can speed up training by presorting the data (set presort=True), but this slows down training considerably for larger training sets.

Gini Impurity or Entropy?

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.

Entropy

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

Gini impurity is slightly faster to compute, so it is a good default. However, when they differ, Gini impurity tends to isolate the most frequent class in its own branch of the tree, while entropy tends to produce slightly more balanced trees

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, fitting it very closely, and most likely overfitting it.

nonparametric model: the number of parameters is not determined prior to training, so the model structure is free to stick closely to the data.

parametric model: has a predetermined number of parameters, so its degree of freedom is limited, reducing the risk of overfitting (but increasing the risk of underfitting).

Regularization:

- maximum depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, max_features

NOTE

> Other algorithms work by first training the Decision Tree without restrictions, then pruning (deleting) unnecessary nodes. A node whose children are all leaf nodes is considered unnecessary if the purity improvement it provides is not statistically significant.

Regression

Notice how the predicted value for each region is always the average target value of the instances in that region. The algorithm splits each region in a way that makes most training instances as close as possible to that predicted value.

CART cost function for regression

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

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

Instability

Decision Trees are simple to understand and interpret, easy to use, versatile, and powerful.

Limitations:

- Orthogonal decision boundaries (all splits are perpendicular to an axis), which makes them sensitive to training set rotation. (solution: PCA)
- Very sensitive to small variations in the training data. (solution: random Forests can limit this instability by averaging predictions over many trees) (the training algorithm used by Scikit-Learn is stochastic)