moment) control additional stopping conditions (`min_samples_split`, `min_samples_leaf`, `min_weight_fraction_leaf`, and `max_leaf_nodes`).

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:[2] 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.

Computational Complexity

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 $O(log_2(m))$ nodes.[3] 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 $O(n \times m\ log(m))$. 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?

By default, the Gini impurity measure is used, but you can select the *entropy* impurity measure instead by setting the `criterion` 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. It later spread to a wide variety of domains, including Shannon's *information theory*, where it measures

2 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 ≠ NP (which seems likely), then no polynomial algorithm will ever be found for any NP-Complete problem (except perhaps on a quantum computer).

3 log_2 is the binary logarithm. It is equal to $log_2(m) = log(m) / log(2)$.

the average information content of a message:[4] entropy is zero when all messages are identical. In Machine Learning, it is frequently used as an impurity measure: a set's entropy is zero when it contains instances of only one class. Equation 6-3 shows the definition of the entropy of the i^{th} node. For example, the depth-2 left node in Figure 6-1 has an entropy equal to $-\frac{49}{54}\log\left(\frac{49}{54}\right) - \frac{5}{54}\log\left(\frac{5}{54}\right) \approx 0.31$.

Equation 6-3. Entropy

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

So should you use Gini impurity or entropy? The truth is, 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. 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.[5]

Regularization Hyperparameters

Decision Trees make very few assumptions about the training data (as opposed to linear models, which obviously assume that the data is linear, for example). If left unconstrained, the tree structure will adapt itself to the training data, fitting it very closely, and most likely overfitting it. Such a model is often called a *nonparametric model*, not because it does not have any parameters (it often has a lot) 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 (but increasing the risk of underfitting).

To avoid overfitting the training data, you need to restrict the Decision Tree's freedom during training. As you know by now, this is called 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 sam-

4 A reduction of entropy is often called an *information gain*.

5 See Sebastian Raschka's interesting analysis for more details.