In-Depth: Decision Trees and Random Forests

Previously we have looked in depth at a simple generative classifier (naive Bayes; see "In Depth: Naive Bayes Classification" on page 382) and a powerful discriminative classifier (support vector machines; see "In-Depth: Support Vector Machines" on page 405). Here we'll take a look at motivating another powerful algorithm—a non-parametric algorithm called *random forests*. Random forests are an example of an *ensemble* method, a method that relies on aggregating the results of an ensemble of simpler estimators. The somewhat surprising result with such ensemble methods is that the sum can be greater than the parts; that is, a majority vote among a number of estimators can end up being better than any of the individual estimators doing the voting! We will see examples of this in the following sections. We begin with the standard imports:

```
In[1]: %matplotlib inline
       import numpy as np
       import matplotlib.pyplot as plt
       import seaborn as sns; sns.set()
```

Motivating Random Forests: Decision Trees

Random forests are an example of an *ensemble learner* built on decision trees. For this reason we'll start by discussing decision trees themselves.

Decision trees are extremely intuitive ways to classify or label objects: you simply ask a series of questions designed to zero in on the classification. For example, if you wanted to build a decision tree to classify an animal you come across while on a hike, you might construct the one shown in Figure 5-67.

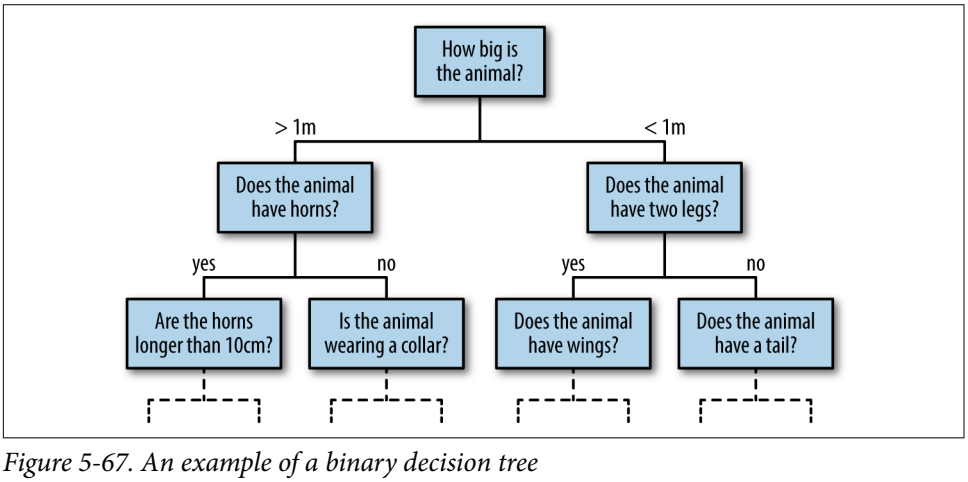

Figure 5-67. An example of a binary decision tree