

Instability

Hopefully by now you are convinced that Decision Trees have a lot going for them: they are simple to understand and interpret, easy to use, versatile, and powerful. However they do have a few limitations. First, as you may have noticed, Decision Trees love orthogonal decision boundaries (all splits are perpendicular to an axis), which makes them sensitive to training set rotation. For example, [Figure 6-7](#) shows a simple linearly separable dataset: on the left, a Decision Tree can split it easily, while on the right, after the dataset is rotated by 45°, the decision boundary looks unnecessarily convoluted. Although both Decision Trees fit the training set perfectly, it is very likely that the model on the right will not generalize well. One way to limit this problem is to use PCA (see [Chapter 8](#)), which often results in a better orientation of the training data.

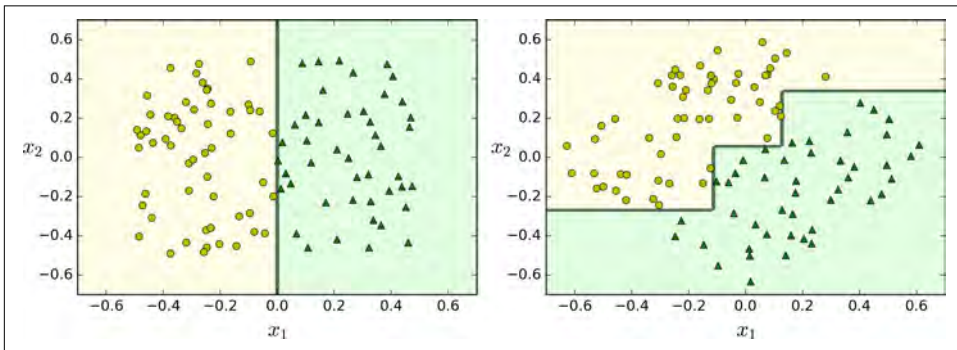


Figure 6-7. Sensitivity to training set rotation

More generally, the main issue with Decision Trees is that they are very sensitive to small variations in the training data. For example, if you just remove the widest Iris-Versicolor from the iris training set (the one with petals 4.8 cm long and 1.8 cm wide) and train a new Decision Tree, you may get the model represented in [Figure 6-8](#). As you can see, it looks very different from the previous Decision Tree ([Figure 6-2](#)). Actually, since the training algorithm used by Scikit-Learn is stochastic⁶ you may get very different models even on the same training data (unless you set the `random_state` hyperparameter).

⁶ It randomly selects the set of features to evaluate at each node.

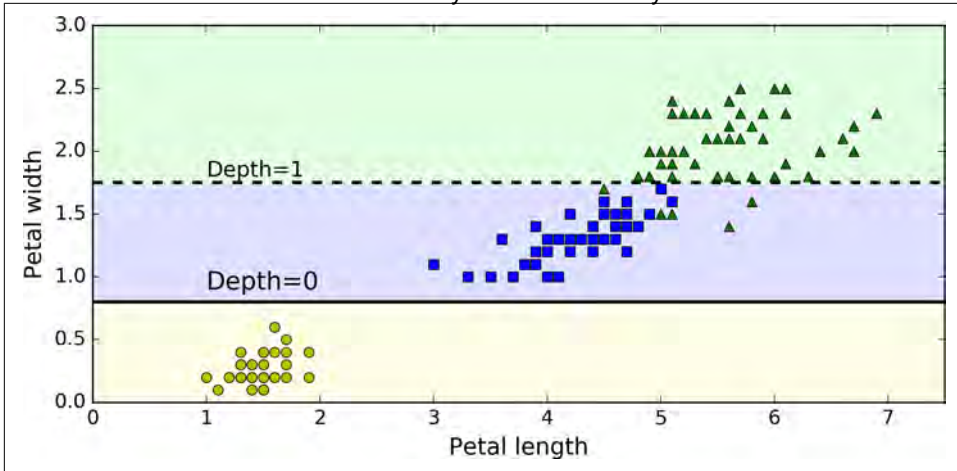


Figure 6-8. Sensitivity to training set details

Random Forests can limit this instability by averaging predictions over many trees, as we will see in the next chapter.

Exercises

1. What is the approximate depth of a Decision Tree trained (without restrictions) on a training set with 1 million instances?
2. Is a node's Gini impurity generally lower or greater than its parent's? Is it *generally* lower/greater, or *always* lower/greater?
3. If a Decision Tree is overfitting the training set, is it a good idea to try decreasing `max_depth`?
4. If a Decision Tree is underfitting the training set, is it a good idea to try scaling the input features?
5. If it takes one hour to train a Decision Tree on a training set containing 1 million instances, roughly how much time will it take to train another Decision Tree on a training set containing 10 million instances?
6. If your training set contains 100,000 instances, will setting `presort=True` speed up training?
7. Train and fine-tune a Decision Tree for the moons dataset.
 - a. Generate a moons dataset using `make_moons(n_samples=10000, noise=0.4)`.
 - b. Split it into a training set and a test set using `train_test_split()`.