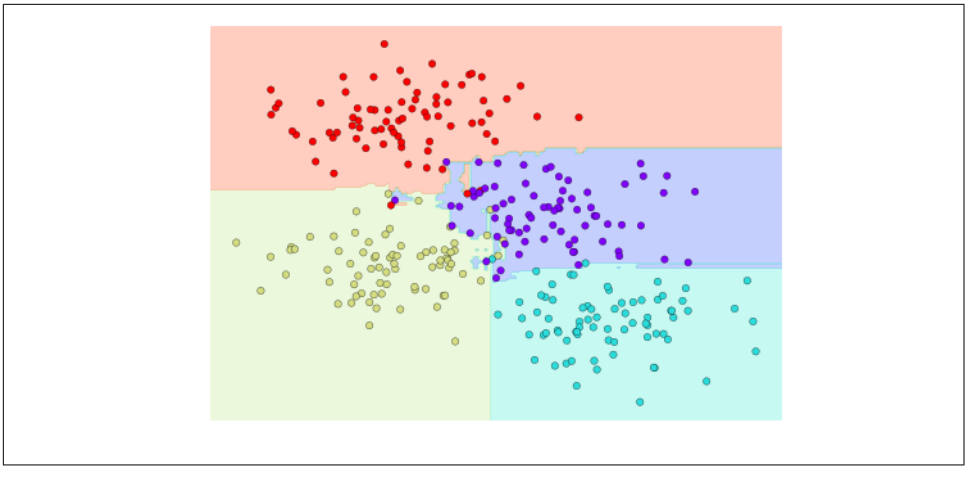

Figure 5-75. Decision boundaries for a random forest, which is an optimized ensemble of decision trees

We see that by averaging over 100 randomly perturbed models, we end up with an overall model that is much closer to our intuition about how the parameter space should be split.

Random Forest Regression

In the previous section we considered random forests within the context of classification. Random forests can also be made to work in the case of regression (that is, continuous rather than categorical variables). The estimator to use for this is the `RandomForestRegressor`, and the syntax is very similar to what we saw earlier.

Consider the following data, drawn from the combination of a fast and slow oscillation (Figure 5-76):

```
In[10]: rng = np.random.RandomState(42)
        x = 10 * rng.rand(200)

        def model(x, sigma=0.3):
            fast_oscillation = np.sin(5 * x)
            slow_oscillation = np.sin(0.5 * x)
            noise = sigma * rng.randn(len(x))

            return slow_oscillation + fast_oscillation + noise

        y = model(x)
        plt.errorbar(x, y, 0.3, fmt='o');
```

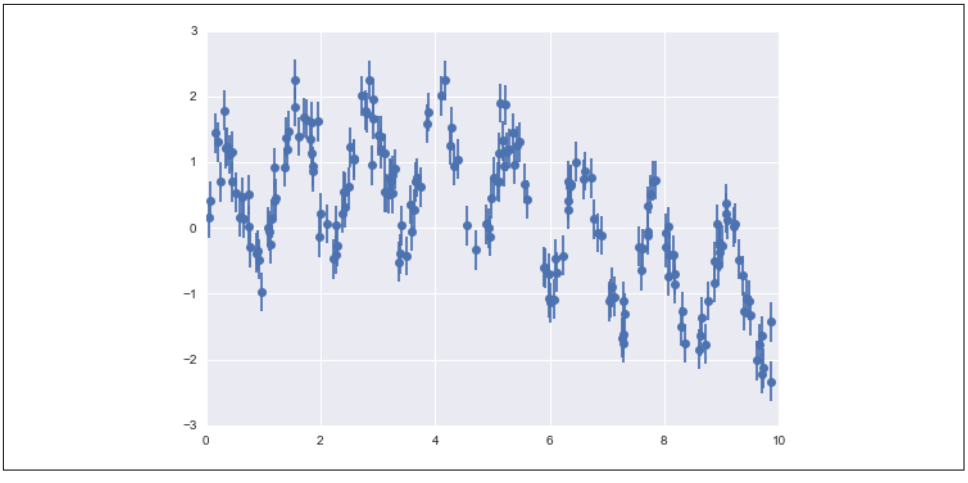

Figure 5-76. Data for random forest regression

Using the random forest regressor, we can find the best-fit curve as follows (Figure 5-77):

```
In[11]: from sklearn.ensemble import RandomForestRegressor
        forest = RandomForestRegressor(200)
        forest.fit(x[:, None], y)

        xfit = np.linspace(0, 10, 1000)
        yfit = forest.predict(xfit[:, None])
        ytrue = model(xfit, sigma=0)

        plt.errorbar(x, y, 0.3, fmt='o', alpha=0.5)
        plt.plot(xfit, yfit, '-r');
        plt.plot(xfit, ytrue, '-k', alpha=0.5);
```

Here the true model is shown by the smooth curve, while the random forest model is shown by the jagged curve. As you can see, the nonparametric random forest model is flexible enough to fit the multiperiod data, without us needing to specify a multiperiod model!

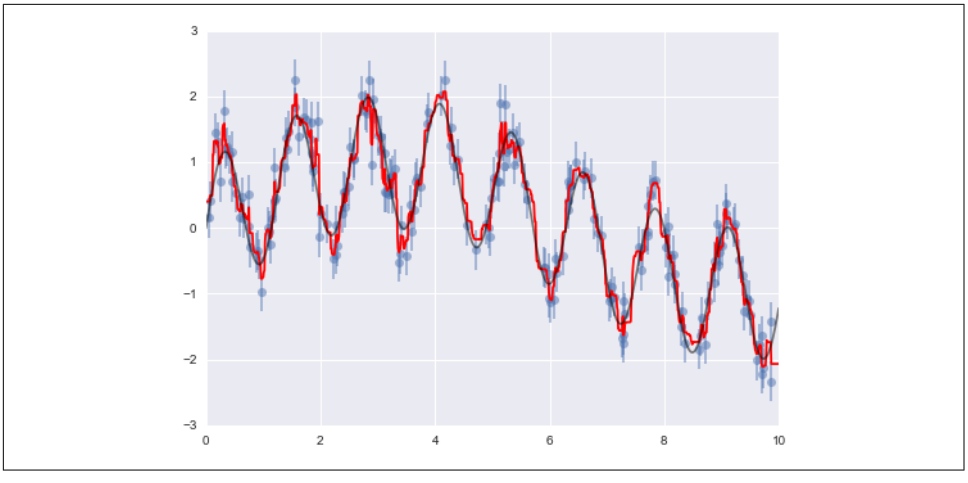

Figure 5-77. Random forest model fit to the data

Example: Random Forest for Classifying Digits

Earlier we took a quick look at the handwritten digits data (see "Introducing Scikit-Learn" on page 343). Let's use that again here to see how the random forest classifier can be used in this context.

```
In[12]: from sklearn.datasets import load_digits
        digits = load_digits()
        digits.keys()
```

```
Out[12]: dict_keys(['target', 'data', 'target_names', 'DESCR', 'images'])
```

To remind us what we're looking at, we'll visualize the first few data points (Figure 5-78):

```
In[13]:
# set up the figure
fig = plt.figure(figsize=(6, 6))  # figure size in inches
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

# plot the digits: each image is 8x8 pixels
for i in range(64):
    ax = fig.add_subplot(8, 8, i + 1, xticks=[], yticks=[])
    ax.imshow(digits.images[i], cmap=plt.cm.binary, interpolation='nearest')

    # label the image with the target value
    ax.text(0, 7, str(digits.target[i]))
```