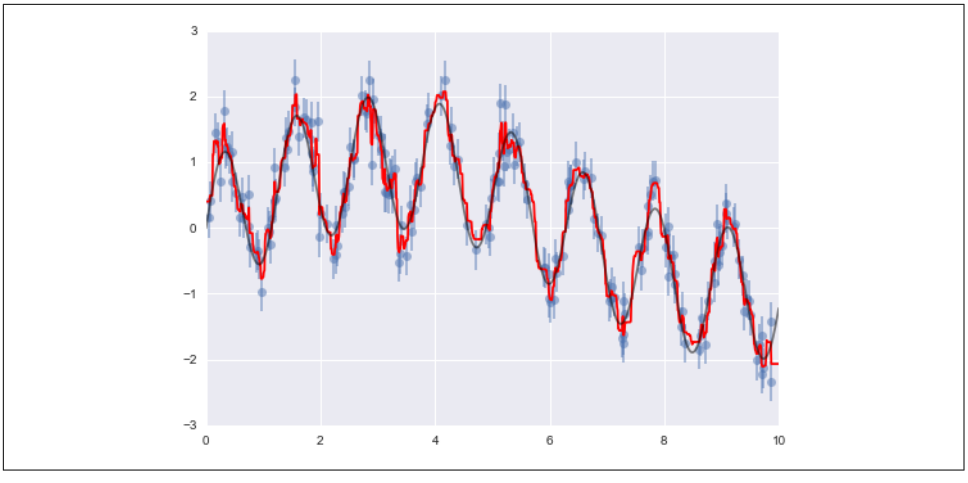

Figure 5-77. Random forest model fit to the data

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]))
```

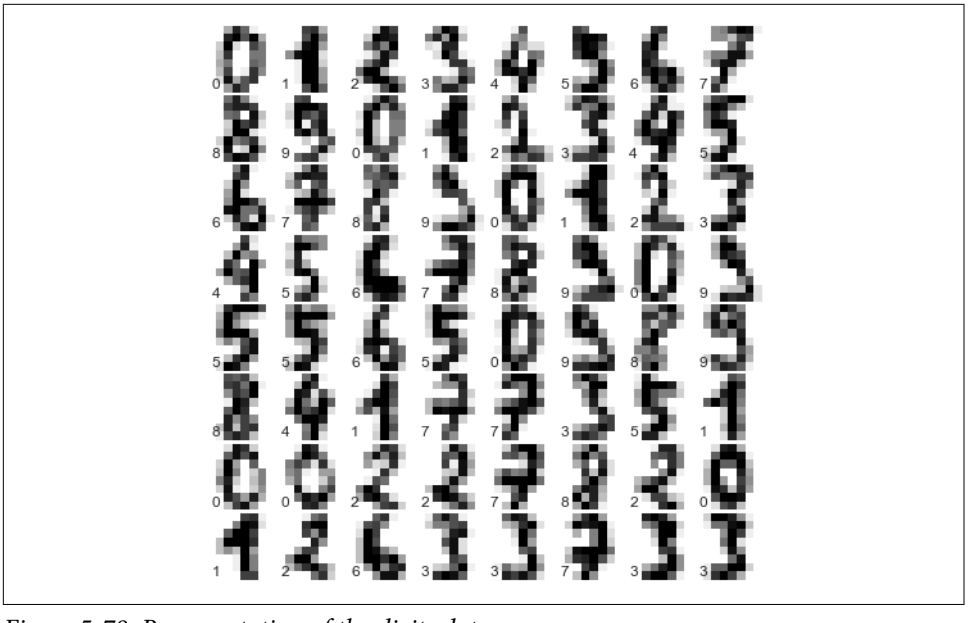

Figure 5-78. Representation of the digits data

We can quickly classify the digits using a random forest as follows (Figure 5-79):

```
In[14]:
from sklearn.cross_validation import train_test_split

Xtrain, Xtest, ytrain, ytest = train_test_split(digits.data, digits.target,
                                                random_state=0)
model = RandomForestClassifier(n_estimators=1000)
model.fit(Xtrain, ytrain)
ypred = model.predict(Xtest)
```

We can take a look at the classification report for this classifier:

```
In[15]: from sklearn import metrics
        print(metrics.classification_report(ypred, ytest))
```

```
             precision    recall  f1-score   support

          0       1.00      0.97      0.99        38
          1       1.00      0.98      0.99        44
          2       0.95      1.00      0.98        42
          3       0.98      0.96      0.97        46
          4       0.97      1.00      0.99        37
          5       0.98      0.96      0.97        49
          6       1.00      1.00      1.00        52
          7       1.00      0.96      0.98        50
          8       0.94      0.98      0.96        46
          9       0.96      0.98      0.97        46

avg / total       0.98      0.98      0.98       450
```

And for good measure, plot the confusion matrix (Figure 5-79):

```
In[16]: from sklearn.metrics import confusion_matrix
        mat = confusion_matrix(ytest, ypred)
        sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
        plt.xlabel('true label')
        plt.ylabel('predicted label');
```

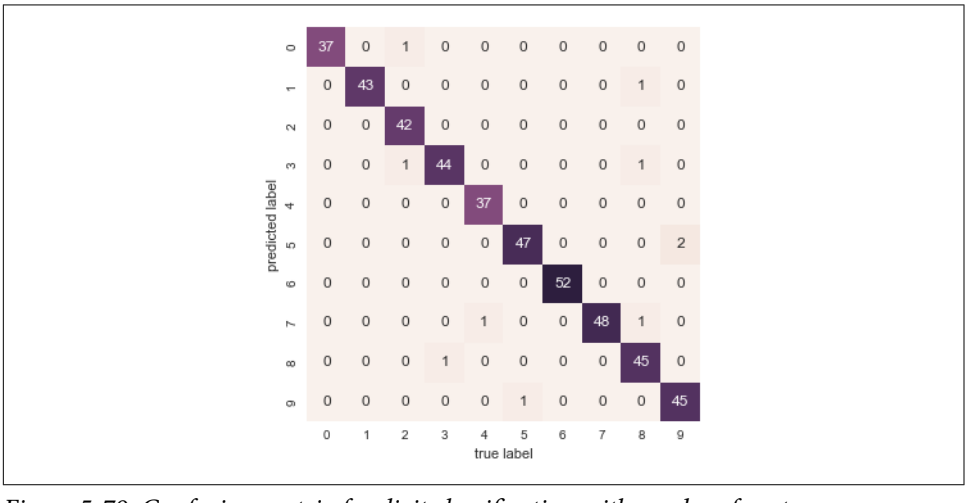

Figure 5-79. Confusion matrix for digit classification with random forests

We find that a simple, untuned random forest results in a very accurate classification of the digits data.

Summary of Random Forests

This section contained a brief introduction to the concept of *ensemble estimators*, and in particular the random forest model—an ensemble of randomized decision trees. Random forests are a powerful method with several advantages:

- Both training and prediction are very fast, because of the simplicity of the underlying decision trees. In addition, both tasks can be straightforwardly parallelized, because the individual trees are entirely independent entities.
- The multiple trees allow for a probabilistic classification: a majority vote among estimators gives an estimate of the probability (accessed in Scikit-Learn with the `predict_proba()` method).
- The nonparametric model is extremely flexible, and can thus perform well on tasks that are underfit by other estimators.