```
import numpy as np

shuffle_index = np.random.permutation(60000)
X_train, y_train = X_train[shuffle_index], y_train[shuffle_index]
```

Training a Binary Classifier

Let's simplify the problem for now and only try to identify one digit—for example, the number 5. This "5-detector" will be an example of a *binary classifier*, capable of distinguishing between just two classes, 5 and not-5. Let's create the target vectors for this classification task:

```
y_train_5 = (y_train == 5)  # True for all 5s, False for all other digits.
y_test_5 = (y_test == 5)
```

Okay, now let's pick a classifier and train it. A good place to start is with a *Stochastic Gradient Descent* (SGD) classifier, using Scikit-Learn's `SGDClassifier` class. This classifier has the advantage of being capable of handling very large datasets efficiently. This is in part because SGD deals with training instances independently, one at a time (which also makes SGD well suited for *online learning*), as we will see later. Let's create an `SGDClassifier` and train it on the whole training set:

```
from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(random_state=42)
sgd_clf.fit(X_train, y_train_5)
```

The `SGDClassifier` relies on randomness during training (hence the name "stochastic"). If you want reproducible results, you should set the `random_state` parameter.

Now you can use it to detect images of the number 5:

```
>>> sgd_clf.predict([some_digit])
array([ True], dtype=bool)
```

The classifier guesses that this image represents a 5 (`True`). Looks like it guessed right in this particular case! Now, let's evaluate this model's performance.

Performance Measures

Evaluating a classifier is often significantly trickier than evaluating a regressor, so we will spend a large part of this chapter on this topic. There are many performance measures available, so grab another coffee and get ready to learn many new concepts and acronyms!