Getting sample data from external sources

- If possible, try working with a familiar dataset while working through this book; in order to level the field, built-in datasets will be used.
- The built-in datasets can be used as stand-ins to test several different modeling techniques such as regression and classification.
- These are, for the most part, famous datasets. This is very useful as papers in various fields will often use these datasets for authors to put forth how their model fits as compared to other models.
- I recommend you use IPython to run these commands as they are presented. Muscle memory is important, and it's best to get to the point where basic commands take no extra mental effort.
- An even better way might be to run IPython Notebook. If you do, make sure to use the `%matplotlib inline` command; this will allow you to see the plots in Notebook.

Preparation

The datasets in scikit-learn are contained within the datasets module. Use the following command to import these datasets:

```
>>> from sklearn import datasets
>>> import numpy as np
```

From within IPython, run `datasets.*?`, which will list everything available within the datasets module.

Working with Data Sets

There are two main types of data within the datasets module.

1. Smaller test datasets are included in the sklearn package and can be viewed by running `datasets.load_*?`.
2. Larger datasets are also available for download as required. The latter are not included in sklearn by default; however, at times, they are better to test models and algorithms due to sufficient complexity to represent realistic situations.

Datasets are included with sklearn by default; to view these datasets, run datasets.

```
load_*?.
```

There are other types of datasets that must be fetched. These datasets are larger, and therefore, they do not come within the package. This said, they are often better to test algorithms that might be used in the wild. First, load the boston dataset and examine it:

```
>>> boston = datasets.load_boston()
>>> print boston.DESCR #output omitted due to length
```

`DESCR` will present a basic overview of the data to give you some context. Next, fetch a dataset:

```
>>> housing = datasets.fetch_california_housing()
downloading Cal. housing from http://lib.stat.cmu.edu [...]
>>> print housing.DESCR #output omitted due to length
```

Bunch Objects

- When these datasets are loaded, they aren't loaded as NumPy arrays. They are of type `Bunch`.
- A `Bunch` is a common data structure in Python.
- It's essentially a dictionary with the keys added to the object as attributes.

To access the data using the data attribute, which is a NumPy array containing the independent variables, the target attribute has the dependent variable:

```
>>> X, y = boston.data, boston.target
```

- There are various implementations available on the Web for the Bunch object; it's not too difficult to write on your own. scikit-learn defines Bunch (as of this writing) in the base module.
- It's available in GitHub at https://github.com/scikit-learn/scikit-learn/blob/ master/sklearn/datasets/base.py.

When you fetch a dataset from an external source it will, by default, place the data in your home directory under `scikit_learn_data/`; this behavior is configurable in two ways:

1. To modify the default behavior, set the `SCIKIT_LEARN_DATA` environment variable to point to the desired folder.
2. The first argument of the fetch methods is `data_home`, which will specify the home folder on a case-by-case basis.

It is easy to check the default location by calling `datasets.get_data_home()`. See also

UCI Machine Learning Repository

The UCI Machine Learning Repository is a great place to find sample datasets. Many of the datasets in scikit-learn are hosted here; however, there are more datasets available. Other notable sources include KDD, your local government agency, and Kaggle competitions.