Methods for loading other datasets:

- Boston house-prices dataset – **datasets.load_boston()**
- Diabetes dataset – **datasets.load_diabetes()**
- Wisconsin breast cancer dataset – **datasets.load_breast_cancer()**
- Wine dataset – **datasets.load_wine()**

Splitting the Dataset into Training and Test Sets

A core practice in machine learning is to split the dataset into different partitions for training and testing. Scikit-learn has a convenient method to assist in that process called **train_test_split(X, y, test_size=0.25)**, where **X** is the design matrix or dataset of predictors and **y** is the target variable. The split size is controlled using the attribute **test_size**. By default, test_size is set to 25% of the dataset size. It is standard practice to shuffle the dataset before splitting by setting the attribute **shuffle=True**.

```
# import module
from sklearn.model_selection import train_test_split
# split in train and test sets
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target,
shuffle=True)
```

```
X_train.shape
'Output': (112, 4)
X_test.shape
'Output': (38, 4)
y_train.shape
'Output': (112,)
y_test.shape
'Output': (38,)
```

Preprocessing the Data for Model Fitting

Before a dataset is trained or fitted with a machine learning model, it necessarily undergoes some vital transformations. These transformations have a huge effect on the performance of the learning model. Transformations in Scikit-learn have a **fit()** and **transform()** method, or a **fit_transform()** method.

Depending on the use case, the **fit()** method can be used to learn the parameters of the dataset, while the **transform()** method applies the data transform based on the learned parameters to the same dataset and also to the test or validation datasets before modeling. Also, the **fit_transform()** method can be used to learn and apply the transformation to the same dataset in a one-off fashion. Data transformation packages are found in the **sklearn.preprocessing** package.

This section will cover some critical transformation for numeric and categorical variables. They include

- Data rescaling
- Standardization
- Normalization
- Binarization
- Encoding categorical variables
- Input missing data
- Generating higher-order polynomial features

Data Rescaling

It is often the case that the features of the dataset contain data with different scales. In other words, the data in column A can be in the range of 1–5, while the data in column B is in the range of 1000–9000. This different scale for units of observations in the same dataset can have an adverse effect for certain machine learning models, especially when minimizing the cost function of the algorithm because it shrinks the function space and makes it difficult for an optimization algorithm like gradient descent to find the global minimum.

When performing data rescaling, usually the attributes are rescaled with the range of 0 and 1. Data rescaling is implemented in Scikit-learn using the **MinMaxScaler** module. Let's see an example.

```
# import packages
from sklearn import datasets
from sklearn.preprocessing import MinMaxScaler
```