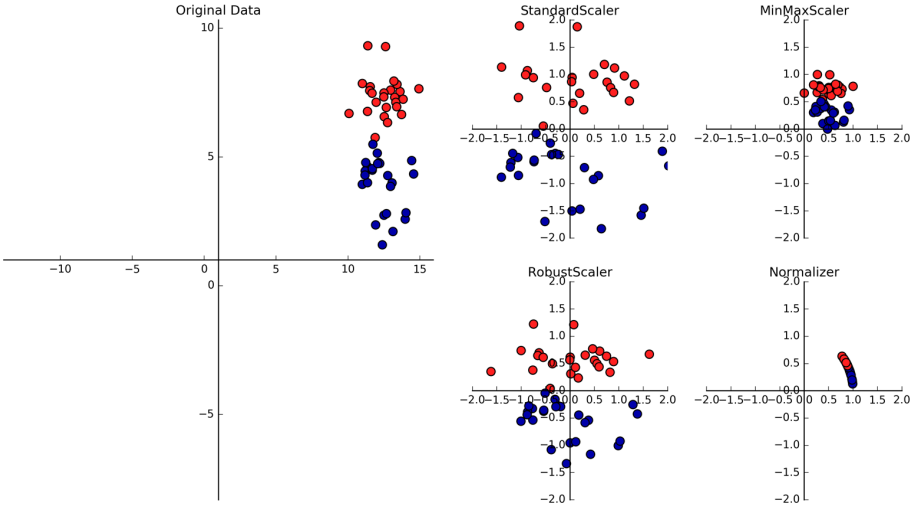

Figure 3-1. Different ways to rescale and preprocess a dataset

Different Kinds of Preprocessing

The first plot in Figure 3-1 shows a synthetic two-class classification dataset with two features. The first feature (the x-axis value) is between 10 and 15. The second feature (the y-axis value) is between around 1 and 9.

The following four plots show four different ways to transform the data that yield more standard ranges. The `StandardScaler` in `scikit-learn` ensures that for each feature the mean is 0 and the variance is 1, bringing all features to the same magnitude. However, this scaling does not ensure any particular minimum and maximum values for the features. The `RobustScaler` works similarly to the `StandardScaler` in that it ensures statistical properties for each feature that guarantee that they are on the same scale. However, the `RobustScaler` uses the median and quartiles,[1] instead of mean and variance. This makes the `RobustScaler` ignore data points that are very different from the rest (like measurement errors). These odd data points are also called *outliers*, and can lead to trouble for other scaling techniques.

The `MinMaxScaler`, on the other hand, shifts the data such that all features are exactly between 0 and 1. For the two-dimensional dataset this means all of the data is con-

1 The median of a set of numbers is the number x such that half of the numbers are smaller than x and half of the numbers are larger than x. The lower quartile is the number x such that one-fourth of the numbers are smaller than x, and the upper quartile is the number x such that one-fourth of the numbers are larger than x.

tained within the rectangle created by the x-axis between 0 and 1 and the y-axis between 0 and 1.

Finally, the `Normalizer` does a very different kind of rescaling. It scales each data point such that the feature vector has a Euclidean length of 1. In other words, it projects a data point on the circle (or sphere, in the case of higher dimensions) with a radius of 1. This means every data point is scaled by a different number (by the inverse of its length). This normalization is often used when only the direction (or angle) of the data matters, not the length of the feature vector.

Applying Data Transformations

Now that we've seen what the different kinds of transformations do, let's apply them using `scikit-learn`. We will use the `cancer` dataset that we saw in Chapter 2. Preprocessing methods like the scalers are usually applied before applying a supervised machine learning algorithm. As an example, say we want to apply the kernel SVM (`SVC`) to the `cancer` dataset, and use `MinMaxScaler` for preprocessing the data. We start by loading our dataset and splitting it into a training set and a test set (we need separate training and test sets to evaluate the supervised model we will build after the preprocessing):

`In[3]:`

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
cancer = load_breast_cancer()

X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target,
                                                    random_state=1)
print(X_train.shape)
print(X_test.shape)
```

`Out[3]:`

```
(426, 30)
(143, 30)
```

As a reminder, the dataset contains 569 data points, each represented by 30 measurements. We split the dataset into 426 samples for the training set and 143 samples for the test set.

As with the supervised models we built earlier, we first import the class that implements the preprocessing, and then instantiate it:

`In[4]:`

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
```