to organize your pictures, the site might want to group together pictures that show the same person. However, the site doesn't know which pictures show whom, and it doesn't know how many different people appear in your photo collection. A sensible approach would be to extract all the faces and divide them into groups of faces that look similar. Hopefully, these correspond to the same person, and the images can be grouped together for you.

Challenges in Unsupervised Learning

A major challenge in unsupervised learning is evaluating whether the algorithm learned something useful. Unsupervised learning algorithms are usually applied to data that does not contain any label information, so we don't know what the right output should be. Therefore, it is very hard to say whether a model "did well." For example, our hypothetical clustering algorithm could have grouped together all the pictures that show faces in profile and all the full-face pictures. This would certainly be a possible way to divide a collection of pictures of people's faces, but it's not the one we were looking for. However, there is no way for us to "tell" the algorithm what we are looking for, and often the only way to evaluate the result of an unsupervised algorithm is to inspect it manually.

As a consequence, unsupervised algorithms are used often in an exploratory setting, when a data scientist wants to understand the data better, rather than as part of a larger automatic system. Another common application for unsupervised algorithms is as a preprocessing step for supervised algorithms. Learning a new representation of the data can sometimes improve the accuracy of supervised algorithms, or can lead to reduced memory and time consumption.

Before we start with "real" unsupervised algorithms, we will briefly discuss some simple preprocessing methods that often come in handy. Even though preprocessing and scaling are often used in tandem with supervised learning algorithms, scaling methods don't make use of the supervised information, making them unsupervised.

Preprocessing and Scaling

In the previous chapter we saw that some algorithms, like neural networks and SVMs, are very sensitive to the scaling of the data. Therefore, a common practice is to adjust the features so that the data representation is more suitable for these algorithms. Often, this is a simple per-feature rescaling and shift of the data. The following code (Figure 3-1) shows a simple example:

In[2]:

```
mglearn.plots.plot_scaling()
```

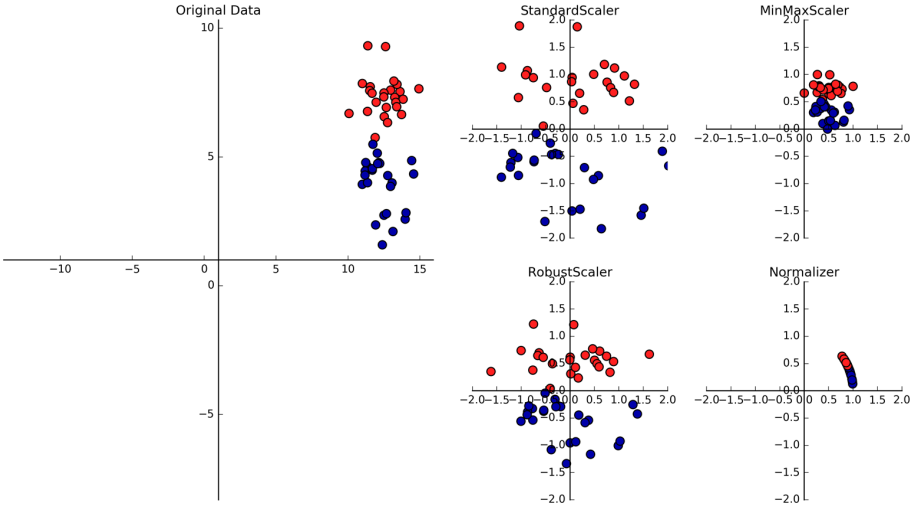

Figure 3-1. Different ways to rescale and preprocess a dataset

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.