

```
# learning an SVM on the scaled training data
svm.fit(X_train_scaled, y_train)

# scoring on the scaled test set
print("SVM test accuracy: {:.2f}".format(svm.score(X_test_scaled, y_test)))
```

Out[12]:

```
SVM test accuracy: 0.96
```

Now that we've seen how simple data transformations for preprocessing work, let's move on to more interesting transformations using unsupervised learning.

## Dimensionality Reduction, Feature Extraction, and Manifold Learning

As we discussed earlier, transforming data using unsupervised learning can have many motivations. The most common motivations are visualization, compressing the data, and finding a representation that is more informative for further processing.

One of the simplest and most widely used algorithms for all of these is principal component analysis. We'll also look at two other algorithms: non-negative matrix factorization (NMF), which is commonly used for feature extraction, and t-SNE, which is commonly used for visualization using two-dimensional scatter plots.

### Principal Component Analysis (PCA)

Principal component analysis is a method that rotates the dataset in a way such that the rotated features are statistically uncorrelated. This rotation is often followed by selecting only a subset of the new features, according to how important they are for explaining the data. The following example (Figure 3-3) illustrates the effect of PCA on a synthetic two-dimensional dataset:

In[13]:

```
mglearn.plots.plot_pca_illustration()
```

The first plot (top left) shows the original data points, colored to distinguish among them. The algorithm proceeds by first finding the direction of maximum variance, labeled "Component 1." This is the direction (or vector) in the data that contains most of the information, or in other words, the direction along which the features are most correlated with each other. Then, the algorithm finds the direction that contains the most information while being orthogonal (at a right angle) to the first direction. In two dimensions, there is only one possible orientation that is at a right angle, but in higher-dimensional spaces there would be (infinitely) many orthogonal directions. Although the two components are drawn as arrows, it doesn't really matter where the head and the tail are; we could have drawn the first component from the center up to