

---

# Unsupervised Learning and Preprocessing

The second family of machine learning algorithms that we will discuss is unsupervised learning algorithms. Unsupervised learning subsumes all kinds of machine learning where there is no known output, no teacher to instruct the learning algorithm. In unsupervised learning, the learning algorithm is just shown the input data and asked to extract knowledge from this data.

## Types of Unsupervised Learning

We will look into two kinds of unsupervised learning in this chapter: transformations of the dataset and clustering.

*Unsupervised transformations* of a dataset are algorithms that create a new representation of the data which might be easier for humans or other machine learning algorithms to understand compared to the original representation of the data. A common application of unsupervised transformations is dimensionality reduction, which takes a high-dimensional representation of the data, consisting of many features, and finds a new way to represent this data that summarizes the essential characteristics with fewer features. A common application for dimensionality reduction is reduction to two dimensions for visualization purposes.

Another application for unsupervised transformations is finding the parts or components that “make up” the data. An example of this is topic extraction on collections of text documents. Here, the task is to find the unknown topics that are talked about in each document, and to learn what topics appear in each document. This can be useful for tracking the discussion of themes like elections, gun control, or pop stars on social media.

*Clustering algorithms*, on the other hand, partition data into distinct groups of similar items. Consider the example of uploading photos to a social media site. To allow you

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()
```