

As you saw in the previous examples, binning, polynomials, and interactions can have a huge influence on how models perform on a given dataset. This is particularly true for less complex models like linear models and naive Bayes models. Tree-based models, on the other hand, are often able to discover important interactions themselves, and don't require transforming the data explicitly most of the time. Other models, like SVMs, nearest neighbors, and neural networks, might sometimes benefit from using binning, interactions, or polynomials, but the implications there are usually much less clear than in the case of linear models.

## Automatic Feature Selection

With so many ways to create new features, you might get tempted to increase the dimensionality of the data way beyond the number of original features. However, adding more features makes all models more complex, and so increases the chance of overfitting. When adding new features, or with high-dimensional datasets in general, it can be a good idea to reduce the number of features to only the most useful ones, and discard the rest. This can lead to simpler models that generalize better. But how can you know how good each feature is? There are three basic strategies: *univariate statistics*, *model-based selection*, and *iterative selection*. We will discuss all three of them in detail. All of these methods are supervised methods, meaning they need the target for fitting the model. This means we need to split the data into training and test sets, and fit the feature selection only on the training part of the data.

### Univariate Statistics

In univariate statistics, we compute whether there is a statistically significant relationship between each feature and the target. Then the features that are related with the highest confidence are selected. In the case of classification, this is also known as *analysis of variance* (ANOVA). A key property of these tests is that they are *univariate*, meaning that they only consider each feature individually. Consequently, a feature will be discarded if it is only informative when combined with another feature. Univariate tests are often very fast to compute, and don't require building a model. On the other hand, they are completely independent of the model that you might want to apply after the feature selection.

To use univariate feature selection in `scikit-learn`, you need to choose a test, usually either `f_classif` (the default) for classification or `f_regression` for regression, and a method to discard features based on the  $p$ -values determined in the test. All methods for discarding parameters use a threshold to discard all features with too high a  $p$ -value (which means they are unlikely to be related to the target). The methods differ in how they compute this threshold, with the simplest ones being `SelectKBest`, which selects a fixed number  $k$  of features, and `SelectPercentile`, which selects a fixed percentage of features. Let's apply the feature selection for classification on the