model using a particular parameter setting on a particular cross-validation split can be done completely independently from the other parameter settings and models. This makes grid search and cross-validation ideal candidates for parallelization over multiple CPU cores or over a cluster. You can make use of multiple cores in `GridSearchCV` and `cross_val_score` by setting the `n_jobs` parameter to the number of CPU cores you want to use. You can set `n_jobs=-1` to use all available cores.

You should be aware that `scikit-learn` *does not allow nesting of parallel operations*. So, if you are using the `n_jobs` option on your model (for example, a random forest), you cannot use it in `GridSearchCV` to search over this model. If your dataset and model are very large, it might be that using many cores uses up too much memory, and you should monitor your memory usage when building large models in parallel.

It is also possible to parallelize grid search and cross-validation over multiple machines in a cluster, although at the time of writing this is not supported within `scikit-learn`. It is, however, possible to use the IPython parallel framework for parallel grid searches, if you don't mind writing the `for` loop over parameters as we did in "Simple Grid Search" on page 261.

For Spark users, there is also the recently developed `spark-sklearn` package, which allows running a grid search over an already established Spark cluster.

Evaluation Metrics and Scoring

So far, we have evaluated classification performance using accuracy (the fraction of correctly classified samples) and regression performance using R^2. However, these are only two of the many possible ways to summarize how well a supervised model performs on a given dataset. In practice, these evaluation metrics might not be appropriate for your application, and it is important to choose the right metric when selecting between models and adjusting parameters.

Keep the End Goal in Mind

When selecting a metric, you should always have the end goal of the machine learning application in mind. In practice, we are usually interested not just in making accurate predictions, but in using these predictions as part of a larger decision-making process. Before picking a machine learning metric, you should think about the high-level goal of the application, often called the *business metric*. The consequences of choosing a particular algorithm for a machine learning application are

called the *business impact*.[2] Maybe the high-level goal is avoiding traffic accidents, or decreasing the number of hospital admissions. It could also be getting more users for your website, or having users spend more money in your shop. When choosing a model or adjusting parameters, you should pick the model or parameter values that have the most positive influence on the business metric. Often this is hard, as assessing the business impact of a particular model might require putting it in production in a real-life system.

In the early stages of development, and for adjusting parameters, it is often infeasible to put models into production just for testing purposes, because of the high business or personal risks that can be involved. Imagine evaluating the pedestrian avoidance capabilities of a self-driving car by just letting it drive around, without verifying it first; if your model is bad, pedestrians will be in trouble! Therefore we often need to find some surrogate evaluation procedure, using an evaluation metric that is easier to compute. For example, we could test classifying images of pedestrians against non-pedestrians and measure accuracy. Keep in mind that this is only a surrogate, and it pays off to find the closest metric to the original business goal that is feasible to evaluate. This closest metric should be used whenever possible for model evaluation and selection. The result of this evaluation might not be a single number—the consequence of your algorithm could be that you have 10% more customers, but each customer will spend 15% less—but it should capture the expected business impact of choosing one model over another.

In this section, we will first discuss metrics for the important special case of binary classification, then turn to multiclass classification and finally regression.

Metrics for Binary Classification

Binary classification is arguably the most common and conceptually simple application of machine learning in practice. However, there are still a number of caveats in evaluating even this simple task. Before we dive into alternative metrics, let's have a look at the ways in which measuring accuracy might be misleading. Remember that for binary classification, we often speak of a *positive* class and a *negative* class, with the understanding that the positive class is the one we are looking for.

Kinds of errors

Often, accuracy is not a good measure of predictive performance, as the number of mistakes we make does not contain all the information we are interested in. Imagine an application to screen for the early detection of cancer using an automated test. If

2 We ask scientifically minded readers to excuse the commercial language in this section. Not losing track of the end goal is equally important in science, though the authors are not aware of a similar phrase to "business impact" being used in that realm.