into two parts: one part for validation of hyperparameters and the other for final model validation. In this case, let's say you reserve 10 datapoints for validation and 10 for final testing. This would be called an 80/10/10 data split.

Why Is the Test Set Necessary?

An important point worth noting is that hyperparameter optimization methods are themselves a form of learning algorithm. In particular, they are a learning algorithm for setting nondifferentiable quantities that aren't easily amenable to calculus-based analysis. The "training set" for the hyperparameter learning algorithm is simply the held-out validation set.

In general, it isn't very meaningful to gauge model performance on their training sets. As always, learned quantities must generalize and it is consequently necessary to test performance on a different set. Since the training set is used for gradient-based learning, and the validation set is used for hyperparameter learning, the test set is necessary to gauge how well learned hyperparameters generalize to new data.

Black-Box Learning Algorithms

Black-box learning algorithms assume no structural information about the systems they are trying to optimize. Most hyperparameter methods are black-box; they work for any type of deep learning or machine learning algorithm.

Black-box methods in general don't scale as well as white-box methods (such as gradient descent) since they tend to get lost in high-dimensional spaces. Due to the lack of directional information from a gradient, black-box methods can get lost in even 50 dimensional spaces (optimizing 50 hyperparameters is quite challenging in practice).

To understand why, suppose there are 50 hyperparameters, each with 3 potential values. Then the black-box algorithm must blindly search a space of size 3^{50}. This can be done, but performing the search will require lots of computational power in general.

Metrics, Metrics, Metrics

When choosing hyperparameters, you want to select those that make the models you design more accurate. In machine learning, a *metric* is a function that gauges the accuracy of predictions from a trained model. Hyperparameter optimization is done to optimize for hyperparameters that maximize (or minimize) this metric on the validation set. While this sounds simple up front, the notion of accuracy can in fact be

quite subtle. Suppose you have a binary classifier. Is it more important to never mislabel false samples as true or to never mislabel true samples as false? How can you choose for model hyperparameters that satisfy the needs of your applications?

The answer turns out to be to choose the correct metric. In this section, we will discuss many different metrics for classification and regression problems. We will comment on the qualities each metric emphasizes. There is no best metric, but there are more suitable and less suitable metrics for different applications.

Metrics Aren't a Replacement for Common Sense!

Metrics are terribly blind. They only optimize for a single quantity. Consequently, blind optimization of metrics can lead to entirely unsuitable outcomes. On the web, media sites often choose to optimize the metric of "user clicks." Some enterprising young journalist or advertiser then realized that titles like "You'll never believe what happened when X" induced users to click at higher fractions. Lo and behold, clickbait was born. While clickbait headlines do indeed induce readers to click, they also turn off readers and lead them to avoid spending time on clickbait-filled sites. Optimizing for user clicks resulted in drops in user engagement and trust.

The lesson here is general. Optimizing for one metric often comes at the cost of a separate quantity. Make sure that the quantity you wish to optimize for is indeed the "right" quantity. Isn't it interesting how machine learning still seems to require human judgment at its core?

Binary Classification Metrics

Before introducing metrics for binary classification models, we think you will find it useful to learn about some auxiliary quantities. When a binary classifier makes predictions on a set of datapoints, you can split all these predictions into one of four categories (Table 5-1).

Table 5-1. Prediction categories

Category	Meaning
True Positive (TP)	Predicted true, Label true
False Positive (FP)	Predicted true, Label false
True Negative (TN)	Predicted false, Label false
False Negative (FN)	Predicted false, Label true

We will also find it useful to introduce the notation shown in Table 5-2.