But the greatest danger in data mining is the opposite problem—overfit—where you fit the noise, as well as the signal, of the data. Overfit models look good while training but fall apart on evaluation when used on new data. Often analysts have brought a model all the way to implementation, feeling quite proud of their accomplishment, only to have reality hit at such a late stage that great damage is done. This is why overfit is so dangerous; the stakes are often high when it is discovered.

### Splitting Data

The essential first step in any modeling task is to split off an evaluation set. This should come before *any* other step (with the exception of examining the data just enough to stratify the training sample, if the minority cases are rare enough). Often, researchers don’t do this early enough, so the lessons they learn—about which variables are useful, what the valid ranges of the data are, what outliers look like, etc.—are all tainted by looking at all the data, and the evaluation results aren’t truly out-of-sample. It is essential that the data a model is presented on evaluation are completely new to the model (and the process that generated the model, including your own examinations). Only then is it a true evaluation.

Training error is obviously not our metric to optimize; if it were most important, a lookup table would be the ideal algorithm (“Let’s see, case 17, ... the answer is ...”). For a model to be used, going forward, it needs to generalize to new data. Recall learning material from a textbook: some problems have answers (output labels) at the back (the training data) and similar but new problems on the test (evaluation). To do well on the test, a good learner induces the lessons (relationship between the questions and the answers) from the specific cases where the label (answer) is known and then can handle questions of the same type in the future.

*But how do you split the data?* As for proportion, it is usual to set aside 20–30% for evaluation. There is a trade-off; you want as much data as possible to drive learning, but also enough to comprehensively test the model. The practical consensus is to put more data on the training than evaluation side. As for case selection, the default method is to randomly split the data. It is too dangerous to split it by the order in the file, as there may be some information embedded in how the data was assembled. You may end up training on the credit applications that first arrived, for instance, and testing on those that last arrived, to find that they have very different behavior. The exception is if the model itself is time-based, e.g., as in stock-market investing. Then the best (toughest) metric is to train on the oldest data and evaluate on the newest. This reflects how the real world will challenge the model. Otherwise, you might not notice how vulnerable the model is to sudden...

---

1 Some like to call it a “test” set, but, as you often have to name files or variables to keep track of data, it’s very useful to label the out-of-sample (evaluation) data with a different first letter than the in-sample (training) data, e.g., Tdata and Edata, etc.