In binary classification we often speak of one class being the *positive* class and the other class being the *negative* class. Here, positive doesn't represent having benefit or value, but rather what the object of the study is. So, when looking for spam, "positive" could mean the spam class. Which of the two classes is called positive is often a subjective matter, and specific to the domain.

The iris example, on the other hand, is an example of a multiclass classification problem. Another example is predicting what language a website is in from the text on the website. The classes here would be a pre-defined list of possible languages.

For regression tasks, the goal is to predict a continuous number, or a *floating-point number* in programming terms (or *real number* in mathematical terms). Predicting a person's annual income from their education, their age, and where they live is an example of a regression task. When predicting income, the predicted value is an *amount*, and can be any number in a given range. Another example of a regression task is predicting the yield of a corn farm given attributes such as previous yields, weather, and number of employees working on the farm. The yield again can be an arbitrary number.

An easy way to distinguish between classification and regression tasks is to ask whether there is some kind of continuity in the output. If there is continuity between possible outcomes, then the problem is a regression problem. Think about predicting annual income. There is a clear continuity in the output. Whether a person makes $40,000 or $40,001 a year does not make a tangible difference, even though these are different amounts of money; if our algorithm predicts $39,999 or $40,001 when it should have predicted $40,000, we don't mind that much.

By contrast, for the task of recognizing the language of a website (which is a classification problem), there is no matter of degree. A website is in one language, or it is in another. There is no continuity between languages, and there is no language that is *between* English and French.[1]

Generalization, Overfitting, and Underfitting

In supervised learning, we want to build a model on the training data and then be able to make accurate predictions on new, unseen data that has the same characteristics as the training set that we used. If a model is able to make accurate predictions on unseen data, we say it is able to *generalize* from the training set to the test set. We want to build a model that is able to generalize as accurately as possible.

1 We ask linguists to excuse the simplified presentation of languages as distinct and fixed entities.

Usually we build a model in such a way that it can make accurate predictions on the training set. If the training and test sets have enough in common, we expect the model to also be accurate on the test set. However, there are some cases where this can go wrong. For example, if we allow ourselves to build very complex models, we can always be as accurate as we like on the training set.

Let's take a look at a made-up example to illustrate this point. Say a novice data scientist wants to predict whether a customer will buy a boat, given records of previous boat buyers and customers who we know are not interested in buying a boat.[2] The goal is to send out promotional emails to people who are likely to actually make a purchase, but not bother those customers who won't be interested.

Suppose we have the customer records shown in Table 2-1.

Table 2-1. Example data about customers

Age	Number of cars owned	Owns house	Number of children	Marital status	Owns a dog	Bought a boat
66	1	yes	2	widowed	no	yes
52	2	yes	3	married	no	yes
22	0	no	0	married	yes	no
25	1	no	1	single	no	no
44	0	no	2	divorced	yes	no
39	1	yes	2	married	yes	no
26	1	no	2	single	no	no
40	3	yes	1	married	yes	no
53	2	yes	2	divorced	no	yes
64	2	yes	3	divorced	no	no
58	2	yes	2	married	yes	yes
33	1	no	1	single	no	no

After looking at the data for a while, our novice data scientist comes up with the following rule: "If the customer is older than 45, and has less than 3 children or is not divorced, then they want to buy a boat." When asked how well this rule of his does, our data scientist answers, "It's 100 percent accurate!" And indeed, on the data that is in the table, the rule is perfectly accurate. There are many possible rules we could come up with that would explain perfectly if someone in this dataset wants to buy a boat. No age appears twice in the data, so we could say people who are 66, 52, 53, or

2 In the real world, this is actually a tricky problem. While we know that the other customers haven't bought a boat from us yet, they might have bought one from someone else, or they may still be saving and plan to buy one in the future.

58 years old want to buy a boat, while all others don't. While we can make up many rules that work well on this data, remember that we are not interested in making predictions for this dataset; we already know the answers for these customers. We want to know if *new customers* are likely to buy a boat. We therefore want to find a rule that will work well for new customers, and achieving 100 percent accuracy on the training set does not help us there. We might not expect that the rule our data scientist came up with will work very well on new customers. It seems too complex, and it is supported by very little data. For example, the "or is not divorced" part of the rule hinges on a single customer.

The only measure of whether an algorithm will perform well on new data is the evaluation on the test set. However, intuitively[3] we expect simple models to generalize better to new data. If the rule was "People older than 50 want to buy a boat," and this would explain the behavior of all the customers, we would trust it more than the rule involving children and marital status in addition to age. Therefore, we always want to find the simplest model. Building a model that is too complex for the amount of information we have, as our novice data scientist did, is called *overfitting*. Overfitting occurs when you fit a model too closely to the particularities of the training set and obtain a model that works well on the training set but is not able to generalize to new data. On the other hand, if your model is too simple—say, "Everybody who owns a house buys a boat"—then you might not be able to capture all the aspects of and variability in the data, and your model will do badly even on the training set. Choosing too simple a model is called *underfitting*.

The more complex we allow our model to be, the better we will be able to predict on the training data. However, if our model becomes too complex, we start focusing too much on each individual data point in our training set, and the model will not generalize well to new data.

There is a sweet spot in between that will yield the best generalization performance. This is the model we want to find.

The trade-off between overfitting and underfitting is illustrated in Figure 2-1.

3 And also provably, with the right math.

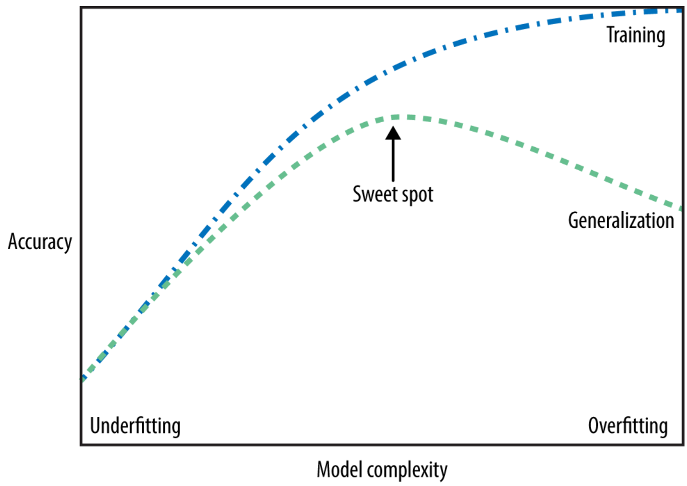

Figure 2-1. Trade-off of model complexity against training and test accuracy

Relation of Model Complexity to Dataset Size

It's important to note that model complexity is intimately tied to the variation of inputs contained in your training dataset: the larger variety of data points your dataset contains, the more complex a model you can use without overfitting. Usually, collecting more data points will yield more variety, so larger datasets allow building more complex models. However, simply duplicating the same data points or collecting very similar data will not help.

Going back to the boat selling example, if we saw 10,000 more rows of customer data, and all of them complied with the rule "If the customer is older than 45, and has less than 3 children or is not divorced, then they want to buy a boat," we would be much more likely to believe this to be a good rule than when it was developed using only the 12 rows in Table 2-1.

Having more data and building appropriately more complex models can often work wonders for supervised learning tasks. In this book, we will focus on working with datasets of fixed sizes. In the real world, you often have the ability to decide how much data to collect, which might be more beneficial than tweaking and tuning your model. Never underestimate the power of more data.

Supervised Machine Learning Algorithms

We will now review the most popular machine learning algorithms and explain how they learn from data and how they make predictions. We will also discuss how the concept of model complexity plays out for each of these models, and provide an over-