Chapter 1

What Is Machine Learning?

Welcome to the world of *machine learning*! You're about to embark upon an exciting adventure discovering how data scientists use algorithms to uncover knowledge hidden within the troves of data that businesses, organizations, and individuals generate every day.

If you're like us, you often find yourself in situations where you are facing a mountain of data that you're certain contains important insights, but you just don't know how to extract that needle of knowledge from the proverbial haystack. That's where machine learning can help. This book is dedicated to providing you with the knowledge and skills you need to harness the power of machine learning algorithms. You'll learn about the different types of problems that are well-suited for machine learning solutions and

the different categories of machine learning techniques that are most appropriate for tackling different types of problems.

Most importantly, we're going to approach this complex, technical field with a practical mind-set. In this book, our purpose is not to dwell on the intricate mathematical details of these algorithms. Instead, we'll focus on how you can put those algorithms to work for you immediately. We'll also introduce you to the R programming language, which we believe is particularly well-suited to approaching machine learning problems from a practical standpoint. But don't worry about programming or R for now. We'll get to that in Chapter 2. For now, let's dive in and get a better understanding of how machine learning works.

By the end of this chapter, you will have learned the following:

- How machine learning allows the discovery of knowledge in data
- How unsupervised learning, supervised learning, and reinforcement learning techniques differ from each other
- How classification and regression problems differ from each other
- How to measure the effectiveness of machine learning algorithms
- How cross-validation improves the accuracy of machine learning models

DISCOVERING KNOWLEDGE IN DATA

Our goal in the world of machine learning is to use algorithms to discover knowledge in our datasets that we can then apply to help us make informed decisions about the future. That's true regardless of the specific subject-matter expertise where we're working, as machine learning has applications across a wide variety of fields. For example, here are some cases where machine learning commonly adds value:

- Segmenting customers and determining the marketing messages that will appeal to different customer groups
- Discovering anomalies in system and application logs that may be indicative of a cybersecurity incident
- Forecasting product sales based on market and environmental conditions
- Recommending the next movie that a customer might want to watch based on their past activity and the preferences of similar customers
- Setting prices for hotel rooms far in advance based on forecasted demand

Of course, those are just a few examples. Machine learning can bring value to almost every field where discovering previously unknown knowledge is useful—and we challenge you to think of a field where knowledge doesn't offer an advantage!

Introducing Algorithms

As we proceed throughout this book, you'll see us continually referring to machine learning techniques as *algorithms*. This is a term from the world of computer science that comes up again and again in the world of data science, so it's important that you understand it. While the term sounds technically complex, the concept of an algorithm is actually straightforward, and we'd venture to guess that you use some form of an algorithm almost every day.

An algorithm is, quite simply, a set of steps that you follow when carrying out a process. Most commonly, we use the term when we're referring to the steps that a computer follows when it is carrying out a computational task, but we can think of many things that we do each day as algorithms. For example, when we are walking the streets of a large city and we reach an intersection, we follow an algorithm for crossing the street. Figure 1.1 shows an example of how this process might work.

Of course, in the world of computer science, our algorithms are more complex and are implemented by writing software, but we can think of them in this same way. An algorithm is simply a series of precise observations, decisions, and instructions that tell the computer how to carry out an action. We design machine learning algorithms to discover

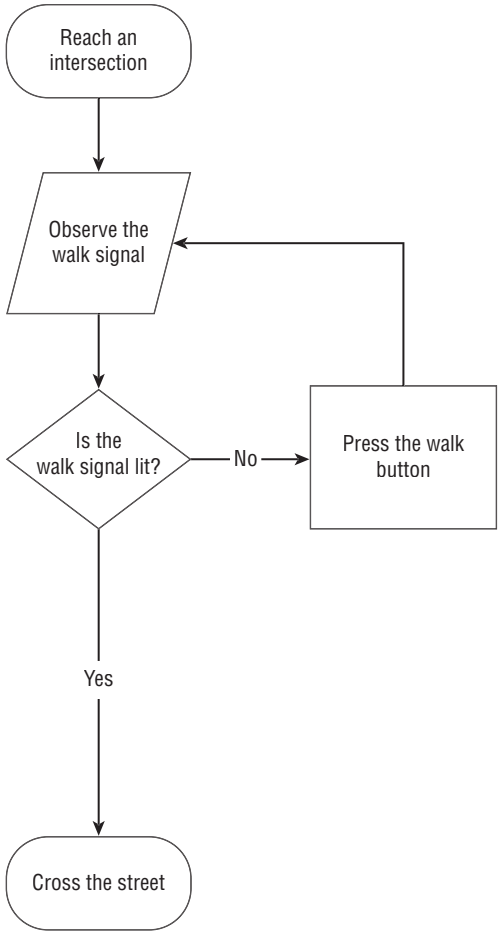


Figure 1.1 Algorithm for crossing the street

knowledge in our data. As we progress through this book, you'll learn about many different types of machine learning algorithms and how they work to achieve this goal in very different ways.

Artificial Intelligence, Machine Learning, and Deep Learning

We hear the terms artificial intelligence, machine learning, and deep learning being used almost interchangeably to describe any sort of technique where computers are working with data. Now that you're entering the world of data science, it's important to have a more precise understanding of these terms.

Artificial intelligence (AI) includes any type of technique where we are attempting to get a computer system to imitate human behavior. As the name implies, we are trying to ask computer systems to artificially behave as if they were intelligent. Now, of course, it's not possible for a modern computer to function at the level of complex reasoning found in the human mind, but we can try to mimic some small portions of human behavior and judgment.

Machine learning (ML) is a subset of artificial intelligence techniques that attempt to apply statistics to data problems in an effort to discover new knowledge by generalizing from examples. Or, in other terms, machine learning techniques are artificial intelligence techniques designed to learn.

Deep learning is a further subdivision of machine learning that uses a set of complex techniques, known as *neural networks*, to discover knowledge in a particular way. It is a highly specialized subfield of machine learning that is most commonly used for image, video, and sound analysis.

Figure 1.2 shows the relationships between these fields. In this book, we focus on machine learning techniques. Specifically, we focus on the categories of machine learning that do *not* fit the definition of deep learning.

MACHINE LEARNING TECHNIQUES

The machine learning techniques that we discuss in this book fit into two major categories. Supervised learning algorithms learn patterns based on labeled examples of past data. Unsupervised learning algorithms seek to uncover patterns without the assistance of labeled data. Let's take a look at each of these techniques in more detail.

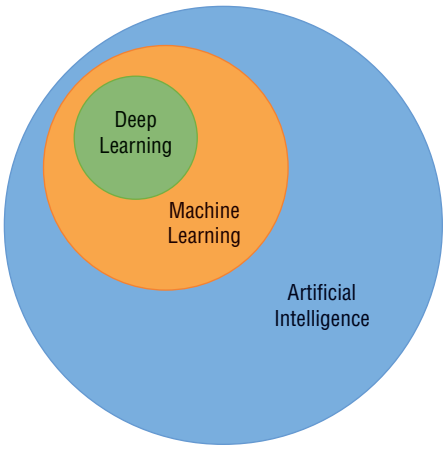


Figure 1.2 The relationship between artificial intelligence, machine learning, and deep learning

Supervised Learning

Supervised learning techniques are perhaps the most commonly used category of machine learning algorithms. The purpose of these techniques is to use an existing dataset to generate a model that then helps us make predictions about future, unlabeled data. More formally, we provide a supervised machine learning algorithm with a *training dataset* as input. The algorithm then uses that training data to develop a *model* as its output, as shown in Figure 1.3.

You can think of the model produced by a supervised machine learning algorithm as sort of a crystal ball—once we have it, we can use it to make predictions about our data. Figure 1.4 shows how this model functions. Once we have it, we can take any new data element that we encounter and use the model to make a prediction about that new element based on the knowledge it obtained from the training dataset.

The reason that we use the term *supervised* to describe these techniques is that we are using a training dataset to supervise the creation of our model. That training dataset contains labels that help us with our prediction task.

Let's reinforce that with a more concrete example. Consider a loan officer working at the car dealership shown in Figure 1.5. The salespeople at the dealership work with individual customers to sell them cars. The customers often don't have the necessary cash on hand to purchase a car outright, so they seek financing options. Our job is to match customers with the right loan product from three choices.

- Subprime loans have the most expensive interest rates and are offered to customers who are likely to miss payment deadlines or default on their loans.
- Top-shelf loans have the lowest interest rate and are offered to customers who are unlikely to miss payments and have an extremely high likelihood of repayment.
- Standard loans are offered to customers who fall in the middle of these two groups and have an interest rate that falls in between those two values.

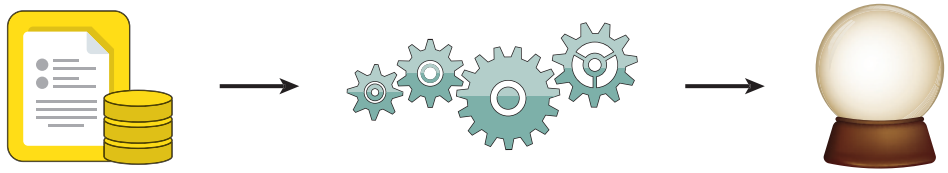


Figure 1.3 Generic supervised learning model

Figure 1.4 Making predictions with a supervised learning model

We receive loan applications from salespeople and must make a decision on the spot. If we don't act quickly, the customer may leave the store, and the business will be lost to another dealership. If we offer a customer a higher risk loan than they would normally qualify for, we might lose their business to another dealership offering a lower interest rate. On the other hand, if we offer a customer a lower interest rate than they deserve, we might not profit on the transaction after they later default.

Our current method of doing business is to review the customer's credit report and make decisions about loan categories based on our years of experience in the role. We've "seen it all" and can rely upon our "gut instinct" to make these important business decisions. However, as budding data scientists, we now realize that there might be a better way to solve this problem using machine learning.

Our car dealership can use supervised machine learning to assist with this task. First, they need a training dataset containing information about their past customers and their loan repayment behavior. The more data they can include in the training dataset, the better. If they have several years of data, that would help develop a high-quality model.

The dataset might contain a variety of information about each customer, such as the customer's approximate age, credit score, home ownership status, and vehicle type. Each of these data points is known as a *feature* about the customer, and they will become the inputs to the machine learning model created by the algorithm. The dataset also needs to contain *labels* for each one of the customers in the training dataset. These labels are the values that we'd like to predict using our model. In this case, we have two labels: default and repaid. We label each customer in our training dataset with the appropriate label for their loan status. If they repaid their loan in full, they are given the "repaid" label, while those who failed to repay their loans are given the "default" label.

Figure 1.5 Using machine learning to classify car dealership customers

A small segment of the resulting dataset appears in Figure 1.6. Notice two things about this dataset. First, each row in the dataset corresponds to a single customer, and those customers are all *past* customers who have completed their loan terms. We know the outcomes of the loans made to each of these customers, providing us with the labels we need to train a supervised learning model. Second, each of the features included in the model are characteristics that are available to the loan officer at the time they are making a loan decision. That's crucial to creating a model that is effective for our given problem. If the model included a feature that specified whether a customer lost his or her job during the loan term, that would likely provide us with accurate results, but the loan officer would not be able to actually *use* that model because they would have no way of determining this feature for a customer at the time of a loan decision. How would they know if the customer is going to lose their job over the term of the loan that hasn't started yet?

Customer Number	Age	Credit Score	Home Status	Vehicle Type	Outcome
1	52	420	Own	Sedan	Default
2	52	460	Own	Sedan	Default
3	64	480	Rent	Sports	Repaid
4	31	580	Rent	Sedan	Default
5	36	620	Own	Sports	Repaid
6	29	690	Rent	Pickup	Repaid
7	23	730	Rent	Sedan	Repaid
8	27	760	Rent	Pickup	Repaid
9	43	790	Own	Pickup	Repaid

Figure 1.6 Dataset of past customer loan repayment behavior

If we use a machine learning algorithm to generate a model based on this data, it might pick up on a few characteristics of the dataset that may also be apparent to you upon casual inspection. First, most people with a credit score under 600 who have financed a car through us in the past defaulted on that loan. If we use that characteristic alone to make decisions, we'd likely be in good shape. However, if we look at the data carefully, we might realize that we could realize an even better fit by saying that anyone who has a credit score under 600 *and* purchased a sedan is likely to default. That type of knowledge, when generated by an algorithm, is a machine learning model!

The loan officer could then deploy this machine learning model by simply following these rules to make a prediction each time someone applies for a loan. If the next customer through the door has a credit score of 780 and is purchasing a sports car, as shown in Figure 1.7, they should be given a top-shelf loan because it is quite unlikely that they will default. If the customer has a credit score of 410 and is purchasing a sedan, we'd definitely want to slot them into a subprime loan. Customers who fall somewhere in between these extremes would be suited for a standard loan.

Now, this was a simplistic example. All of the customers in our example fit neatly into the categories we described. This won't happen in the real world, of course. Our machine learning algorithms will have imperfect data that doesn't have neat, clean divisions between groups. We'll have datasets with many more observations, and our algorithms will inevitably make mistakes. Perhaps the next high credit-scoring young person to walk into the dealership purchasing a sports car later loses their job and defaults on the loan. Our algorithm would make an incorrect prediction. We talk more about the types of errors made by algorithms later in this chapter.

Figure 1.7 Applying the machine learning model

Unsupervised Learning

Unsupervised learning techniques work quite differently. While supervised techniques train on labeled data, unsupervised techniques develop models based on unlabeled training datasets. This changes the nature of the datasets that they are able to tackle and the models that they produce. Instead of providing a method for assigning labels to input based on historical data, unsupervised techniques allow us to discover hidden patterns in our data.

One way to think of the difference between supervised and unsupervised algorithms is that supervised algorithms help us assign known labels to new observations while unsupervised algorithms help us discover new labels, or groupings, of the observations in our dataset.

For example, let's return to our car dealership and imagine that we're now working with our dataset of customers and want to develop a marketing campaign for our service

department. We suspect that the customers in our database are similar to each other in ways that aren't as obvious as the types of cars that they buy and we'd like to discover what some of those groupings might be and use them to develop different marketing messages.

Unsupervised learning algorithms are well-suited to this type of open-ended discovery task. The car dealership problem that we described is more generally known as the *market segmentation* problem, and there is a wealth of unsupervised learning techniques designed to help with this type of analysis. We talk about how organizations use unsupervised *clustering* algorithms to perform market segmentation in Chapter 12.

Let's think of another example. Imagine that we manage a grocery store and are trying to figure out the optimal placement of products on the shelves. We know that customers often run into our store seeking to pick up some common staples, such as milk, bread, meat, and produce. Our goal is to design the store so that impulse purchases are near each other in the store. As seen in Figure 1.8, we want to place the cookies right next to the milk so someone who came into the store to purchase milk will see them and think "Those cookies would be delicious with a glass of this milk!"

Figure 1.8 Strategically placing items in a grocery store based on unsupervised learning

The problem of determining which items customers frequently purchase together is also a well-known problem in machine learning known as the *market basket* problem. We talk about how data scientists use *association rules* approaches to tackle the market basket problem in Chapter 11.

NOTE You may also hear about a third type of machine learning algorithm known as *reinforcement learning*. These algorithms seek to learn based on trial and error, similar to the way that a young child learns the rules of a home by being rewarded and punished. Reinforcement learning is an interesting technique but is beyond the scope of this book.

MODEL SELECTION

In the previous section, we described ways to group algorithms based on the types of data that they use for training. Algorithms that use labeled training datasets are known as *supervised algorithms* because their training is "supervised" by the labels while those that use unlabeled training datasets are known as *unsupervised algorithms* because they are free to learn whatever patterns they happen to discover, without "supervision." Think of this categorization scheme as describing *how* machine learning algorithms learn.

We can also categorize our algorithms based on *what* they learn. In this book, we discuss three major types of knowledge that we can learn from our data. *Classification* techniques train models that allow us to predict membership in a category. *Regression* techniques allow us to predict a numeric result. *Similarity learning* techniques help us discover the ways that observations in our dataset resemble and differ from each other.

Classification Techniques

Classification techniques use supervised machine learning to help us predict a *categorical response*. That means that the output of our model is a non-numeric label or, more formally, a categorical variable. This simply means that the variable takes on discrete, non-numeric values, rather than numeric values. Here are some examples of categorical variables with some possible values they might take on:

- Educational degree obtained (none, bachelor's, master's, doctorate)
- Citizenship (United States, Ireland, Nigeria, China, Australia, South Korea)
- Blood type (A+, A-, B+, B-, AB+, AB-, O+, O-)
- Political party membership (Democrat, Republican, Independent)
- Customer status (current customer, past customer, noncustomer)

For example, earlier in this chapter, we discussed a problem where managers at a car dealership needed the ability to predict loan repayment. This is an example of a classification problem because we are trying to assign each customer to one of two categories: repaid or default.

We encounter all types of classification problems in the real world. We might try to determine which of three promotional offers would be most appealing to a potential customer. This is a classification problem where the categories are the three different offers.

Similarly, we might want to look at people attempting to log on to our computer systems and predict whether they are a legitimate user or a hacker seeking to violate the system's security policies. This is also a classification problem where we are trying to assign each login attempt to the category of "legitimate user" or "hacker."

Regression Techniques

Regression techniques use supervised machine learning techniques to help us predict a *continuous response*. Simply put, this means that the output of our model is a numeric value. Instead of predicting membership in a discrete set of categories, we are predicting the value of a numeric variable.

For example, a financial advisor seeking new clients might want to screen possible clients based on their income. If the advisor has a list of potential customers that does not include income explicitly, they might use a dataset of past contacts with known incomes to train a regression model that predicts the income of future contacts. This model might look something like this:

$$Income = 5000 + 1000 * age + 3000 * yearsPostHighSchoolEducation$$

If the financial advisor encounters a new potential client, they can then use this formula to predict the person's income based on their age and years of education. For each year of age, they would expect the person to have $1,000 in additional annual income. Similarly, their income would increase $3,000 for each year of education beyond high school.

Regression models are quite flexible. We can plug in any possible value of age or income and come up with a prediction for that person's income. Of course, if we didn't have good training data, our prediction might not be accurate. We also might find that the relationship between our variables isn't explained by a simple linear technique. For example, income likely increases with age, but only up until a certain point. More advanced regression techniques allow us to build more complex models that can take these factors into account. We discuss those in Chapter 4.

Similarity Learning Techniques

Similarity learning techniques use machine learning algorithms to help us identify common patterns in our data. We might not know exactly what we're trying to discover, so we allow the algorithm to explore the dataset looking for similarities that we might not have already predicted.

We've already mentioned two similarity learning techniques in this chapter. Association rules techniques, discussed more fully in Chapter 11, allow us to solve problems that are similar to the market basket problem—which items are commonly purchased together. Clustering techniques, discussed more fully in Chapter 12, allow us to group observations into clusters based on the similar characteristics they possess.

Association rules and clustering are both examples of unsupervised uses of similarity learning techniques. It's also possible to use similarity learning in a supervised manner. For example, nearest neighbor algorithms seek to assign labels to observations based on the labels of the most similar observations in the training dataset. We discuss those more in Chapter 6.

MODEL EVALUATION

Before beginning our discussion of specific machine learning algorithms, it's also helpful to have an idea in mind of how we will evaluate the effectiveness of our algorithms. We're going to cover this topic in much more detail throughout the book, so this is just to give you a feel for the concept. As we work through each machine learning technique, we'll discuss evaluating its performance against a dataset. We'll also have a more complete discussion of model performance evaluation in Chapter 9.

Until then, the important thing to realize is that some algorithms will work better than others on different problems. The nature of the dataset and the nature of the algorithm will dictate the appropriate technique.

In the world of supervised learning, we can evaluate the effectiveness of an algorithm based on the number and/or magnitude of errors that it makes. For classification problems, we often look at the percentage of times that the algorithm makes an incorrect categorical prediction, or the *misclassification rate*. Similarly, we can look at the percentage of predictions that were correct, known as the algorithm's *accuracy*. For regression problems, we often look at the difference between the values predicted by the algorithm and the actual values.

NOTE It only makes sense to talk about this type of evaluation when we're referring to supervised learning techniques where there actually is a correct

answer. In unsupervised learning, we are detecting patterns without any objective guide, so there is no set "right" or "wrong" answer to measure our performance against. Instead, the effectiveness of an unsupervised learning algorithm lies in the value of the insight that it provides us.

Classification Errors

Many classification problems seek to predict a binary value identifying whether an observation is a member of a class. We refer to cases where the observation is a member of the class as *positive* cases and cases where the observation is not a member of the class as *negative* cases.

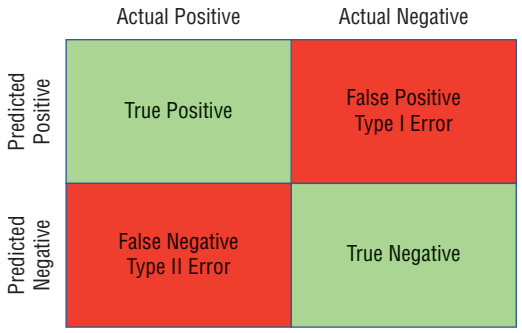
For example, imagine we are developing a model designed to predict whether someone has a lactose intolerance, making it difficult for them to digest dairy products. Our model might include demographic, genetic, and environmental factors that are known or suspected to contribute to lactose intolerance. The model then makes predictions about whether individuals are lactose intolerant or not based on those attributes. Individuals predicted to be lactose intolerant are predicted positives, while those who are predicted to not be lactose intolerant (or, stated more simply, those who are predicted to be lactose tolerant) are predicted negatives. These predicted values come from our machine learning model.

There is also, however, a real-world truth. Regardless of what the model predicts, every individual person is either lactose intolerant or they are not. This real-world data determines whether the person is an actual positive or an actual negative. When the predicted value for an observation differs from the actual value for that same observation, an *error* occurs. There are two different types of error that may occur in a classification problem.

- *False positive errors* occur when the model labels an observation as predicted positive when it is, in reality, an actual negative. For example, if the model identifies someone as likely lactose intolerant while they are, in reality, lactose tolerant, this is a false positive error. False positive errors are also known as *Type I errors*.
- *False negative errors* occur when the model labels an observation as predicted negative when it is, in reality, an actual positive. In our lactose intolerance model, if the model predicts someone as lactose tolerant when they are, in reality, lactose intolerant, this is a false negative error. False negative errors are also known as *Type II errors*.

Similarly, we may label correctly predicted observations as *true positives* or *true negatives*, depending on their label. Figure 1.9 shows the types of errors in chart form.

Figure 1.9 Error types



Of course the absolute numbers for false positive and false negative errors depend on the number of predictions that we make. Instead of using these magnitude-based measures, we measure the percentage of times that those errors occur. For example, the *false positive rate* (FPR) is the percentage of negative instances that were incorrectly identified as positive. We can compute this rate by dividing the number of false positives (FP) by the sum of the number of false positives and the number of true negatives (TN), or, as a formula:

$$FPR = \frac{FP}{FP + TN}$$

Similarly, we can compute the *false negative rate* (FNR) as follows:

$$FNR = \frac{FN}{FN + TP}$$

There is no clear-cut rule about whether one type of error is better or worse than the other. This determination depends greatly on the type of problem being solved.

For example, imagine that we're using a machine learning algorithm to classify a large list of prospective customers as either people who will purchase our product (positive cases) or people who will not purchase our product (negative cases). We only spend the money to send the mailing to prospects labeled by the algorithm as positive.

In the case of a false positive mailing, you send a brochure to a customer who does not buy your product. You've lost the money spent on printing and mailing the brochure. In the case of a false negative result, you do not send a mailing to a customer who would have responded. You've lost the opportunity to sell your product to a customer. Which of

these is worse? It depends on the cost of the mailing, the potential profit per customer, and other factors.

On the other hand, consider the use of a machine learning model to screen patients for the likelihood of cancer and then refer those patients with positive results for additional, more invasive testing. In the case of a false negative result, a patient who potentially has cancer is not sent for additional screening, possibly leaving an active disease untreated. This is clearly a very bad result.

False positive results are not without harm, however. If a patient is falsely flagged as potentially cancerous, they are subjected to unnecessary testing that is potentially costly and painful, consuming resources that could have been used on another patient. They are also subject to emotional harm while they are waiting for the new test results.

The evaluation of machine learning problems is a tricky proposition, and it cannot be done in isolation from the problem domain. Data scientists, subject-matter experts, and, in some cases, ethicists, should work together to evaluate models in light of the benefits and costs of each error type.

Regression Errors

The errors that we might make in regression problems are quite different because the nature of our predictions is different. When we assign classification labels to instances, we can be either right or wrong with our prediction. When we label a noncancerous tumor as cancerous, that is clearly a mistake. However, in regression problems, we are predicting a numeric value.

Consider the income prediction problem that we discussed earlier in this chapter. If we have an individual with an actual income of $45,000 annually and our algorithm's prediction is on the nose at exactly $45,000, that's clearly a correct prediction. If the algorithm predicts an income of $0 or $10,000,000, almost everyone would consider those predictions objectively wrong. But what about predictions of $45,001, $45,500, $46,000, or $50,000? Are those all incorrect? Are some or all of them close enough?

It makes more sense for us to evaluate regression algorithms based on the magnitude of the error in their predictions. We determine this by measuring the distance between the predicted value and the actual value. For example, consider the dataset shown in Figure 1.10.

In this dataset, we're trying to predict the number of bicycle rentals that occur each day based on the average temperature that day. Bicycle rentals appear on the y-axis while temperature appears on the x-axis. The black line is a regression line that says that we expect bicycle rentals to increase as temperature increases. That black line is our model, and the black dots are predictions at specific temperature values along that line.

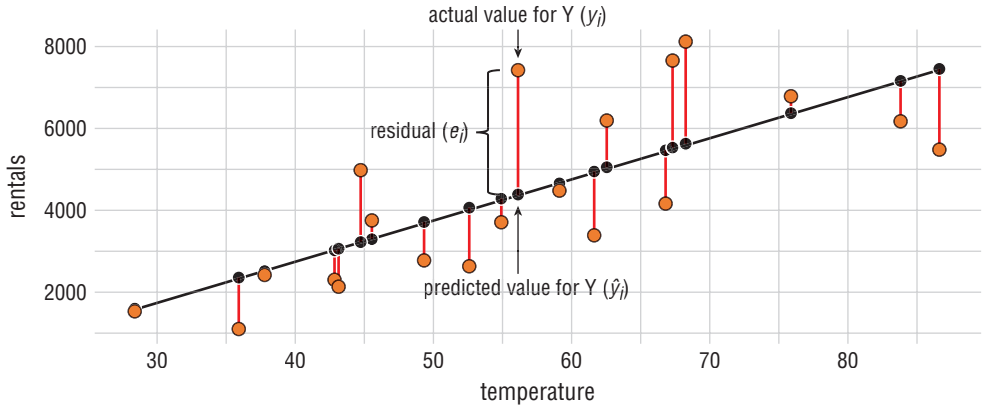


Figure 1.10 Residual error

The orange dots represent real data gathered during the bicycle rental company's operations. That's the "correct" data. The red lines between the predicted and actual values are the magnitude of the error, which we call the *residual value*. The longer the line, the worse the algorithm performed on that dataset.

We can't simply add the residuals together because some of them are negative values that would cancel out the positive values. Instead, we square each residual value and then add those squared residuals together to get a performance measure called the *residual sum of squares*.

We revisit the concept of residual error, as well as this specific bicycle rental dataset, in Chapter 4.

Types of Error

When we build a machine learning model for anything other than the most simplistic problems, the model will include some type of prediction error. This error comes in three different forms.

- *Bias* (in the world of machine learning) is the type of error that occurs due to our choice of a machine learning model. When the model type that we choose is unable to fit our dataset well, the resulting error is bias.
- *Variance* is the type of error that occurs when the dataset that we use to train our machine learning model is not representative of the entire universe of possible data.
- *Irreducible error*, or noise, occurs independently of the machine learning algorithm and training dataset that we use. It is error inherent in the problem that we are trying to solve.

When we are attempting to solve a specific machine learning problem, we cannot do much to address irreducible error, so we focus our efforts on the two remaining sources of error: bias and variance. Generally speaking, an algorithm that exhibits high variance will have low bias, while a low-variance algorithm will have higher bias, as shown in Figure 1.11. Bias and variance are intrinsic characteristics of our models and coexist. When we modify our models to improve one, it comes at the expense of the other. Our goal is to find an optimal balance between the two.

In cases where we have high bias and low variance, we describe the model as *underfitting* the data. Let's take a look at a few examples that might help illustrate this point. Figure 1.12 shows a few attempts to use a function of two variables to predict a third variable. The leftmost graph in Figure 1.12 shows a linear model that underfits the data. Our data points are distributed in a curved manner, but our choice of a straight line (a linear model) limits the ability of the model to fit our dataset. There is no way that you can draw a straight line that will fit this dataset well. Because of this, the majority of the error in our approach is due to our choice of model and our dataset exhibits high bias.

The middle graph in Figure 1.12 illustrates the problem of *overfitting*, which occurs when we have a model with low bias but high variance. In this case, our model fits the training dataset *too* well. It's the equivalent of studying for a specific test (the training dataset) rather than learning a generalized solution to the problem. It's highly likely that when this model is used on a different dataset, it will not work well. Instead of learning the underlying knowledge, we studied the answers to a past exam. When we faced a new exam, we didn't have the knowledge necessary to figure out the answers.

The balance that we seek is a model that optimizes both bias and variance, such as the one shown in the rightmost graph of Figure 1.12. This model matches the curved nature of the distribution but does not closely follow the specific data points in the training

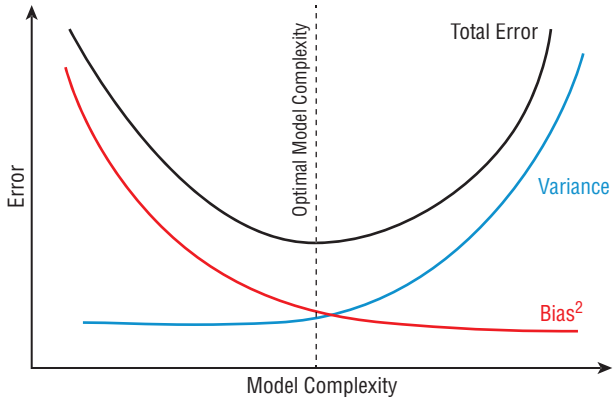


Figure 1.11 The bias/variance trade-off

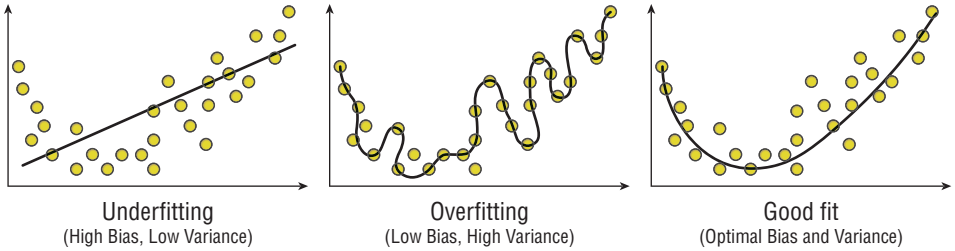


Figure 1.12 Underfitting, overfitting, and optimal fit

dataset. It aligns with the dataset much better than the underfit model but does not closely follow specific points in the training dataset as the overfit model does.

Partitioning Datasets

When we evaluate a machine learning model, we can protect against variance errors by using validation techniques that expose the model to data other than the data used to create the model. The point of this approach is to address the overfitting problem. Look back at the overfit model in Figure 1.12. If we used the training dataset to evaluate this model, we would find that it performed extremely well because the model is highly tuned to perform well on that specific dataset. However, if we used a new dataset to evaluate the model, we'd likely find that it performs quite poorly.

We can explore this issue by using a *test dataset* to assess the performance of our model. The test dataset is set aside at the beginning of the model development process specifically for the purpose of model assessment. It is not used in the training process, so it is not possible for the model to overfit the test dataset. If we develop a generalizable model that does not overfit the training dataset, it will also perform well on the test dataset. On the other hand, if our model overfits the training dataset, it will not perform well on the test dataset.

We also sometimes need a separate dataset to assist with the model development process. These datasets, known as *validation datasets*, are used to help develop the model in an iterative process, adjusting the parameters of the model during each iteration until we find an approach that performs well on the validation dataset. While it may be tempting to use the test dataset as the validation dataset, this approach reintroduces the potential of overfitting the test dataset, so we should use a third dataset for this purpose.

Holdout Method

The most straightforward approach to test and validation datasets is the *holdout method*. In this approach, illustrated in Figure 1.13, we set aside portions of the original dataset for validation and testing purposes at the beginning of the model development process. We use the validation dataset to assist in model development and then use the test dataset to evaluate the performance of the final model.

Cross-Validation Methods

There are also a variety of more advanced methods for creating validation datasets that perform repeated sampling of the data during an iterative approach to model development. These approaches, known as *cross-validation* techniques, are particularly useful for smaller datasets where it is undesirable to reserve a portion of the dataset for validation purposes.

Figure 1.14 shows an example of cross-validation. In this approach, we still set aside a portion of the dataset for testing purposes, but we use a different portion of the training dataset for validation purposes during each iteration of model development.

If this sounds complicated now, don't worry about it. We discuss the holdout method and cross-validation in greater detail when we get to Chapter 9. For now, you should just have a passing familiarity with these techniques.

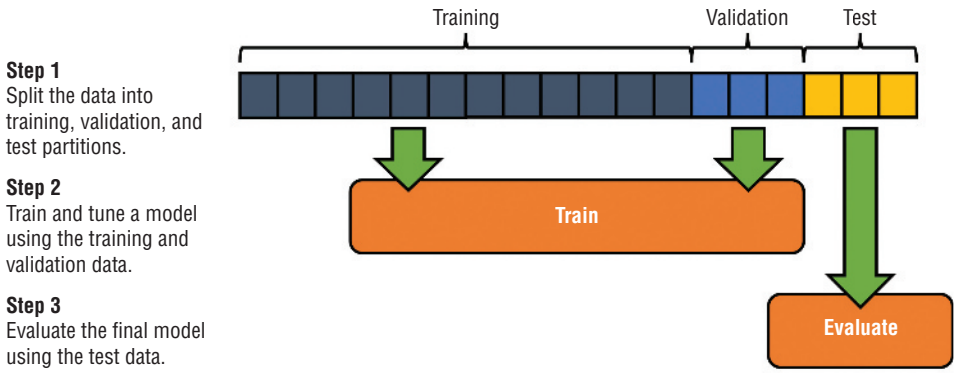


Figure 1.13 Holdout method

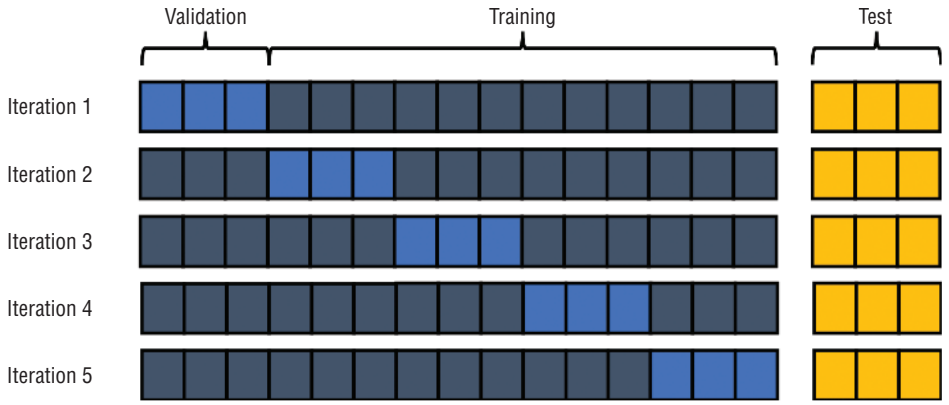


Figure 1.14 Cross-validation method

EXERCISES

1. Consider each of the following machine learning problems. Would the problem be best approached as a classification problem or a regression problem? Provide a rationale for your answer.

a. Predicting the number of fish caught on a commercial fishing voyage
b. Identifying likely adopters of a new technology
c. Using weather and population data to predict bicycle rental rates
d. Predicting the best marketing campaign to send a specific person

2. You developed a machine learning algorithm that assesses a patient's risk of heart attack (a positive event) based on a number of diagnostic criteria. How would you describe each of the following events?

a. Your model identifies a patient as likely to suffer a heart attack, and the patient does suffer a heart attack.
b. Your model identifies a patient as likely to suffer a heart attack, and the patient does not suffer a heart attack.
c. Your model identifies a patient as not likely to suffer a heart attack, and the patient does not suffer a heart attack.
d. Your model identifies a patient as not likely to suffer a heart attack, and the patient does suffer a heart attack.