SAMPLE CHAPTER

REAL-WORLD Machine Learning

Henrik Brink
Joseph W. Richards
Mark Fetherolf

FOREWORD BY Beau Cronin

MANNING

Real-World Machine Learning

by Henrik Brink
Joseph W. Richards
Mark Fetherolf

Chapter 1

brief contents

PART 1 THE MACHINE-LEARNING WORKFLOW 1

1 ■ What is machine learning? 3
2 ■ Real-world data 27
3 ■ Modeling and prediction 52
4 ■ Model evaluation and optimization 77
5 ■ Basic feature engineering 106

PART 2 PRACTICAL APPLICATION 127

6 ■ Example: NYC taxi data 129
7 ■ Advanced feature engineering 146
8 ■ Advanced NLP example: movie review sentiment 172
9 ■ Scaling machine-learning workflows 196
10 ■ Example: digital display advertising 214

1 What is machine learning?

This chapter covers

- Machine-learning basics
- Advantages of machine learning over traditional approaches
- Overview of the basic machine-learning workflow
- Overview of advanced methods for improving model performance

In 1959, an IBM computer scientist named Arthur Samuel wrote a computer program to play checkers. Each board position was assigned a score based on its likelihood of leading to a win. At first, scores were based on a formula using factors such as the number of pieces on each side and the number of kings. It worked, but Samuel had an idea about how to improve its performance. He had the program play thousands of games against itself and used the results to refine the positional scoring. By the mid-1970s, the program had achieved the proficiency of a respectable amateur player.[1]

[1] Jonathan Schaeffer, *One Jump Ahead: Computer Perfection at Checkers* (New York: Springer, 2009).

Samuel had written a computer program that was able to improve its own performance through experience. It learned—and machine learning (ML) was born.

The aim of this book isn't to describe the gory mathematical details of machine-learning algorithms (although we'll peel back a few layers of the onion to provide insight into the inner workings of the most common ones). Rather, the book's primary purpose is to instruct non-experts on important aspects and common challenges when integrating machine learning into real-world applications and data pipelines. In this first chapter, we present a real business problem—reviewing loan applications—to demonstrate the advantages of using machine learning over some of the most common alternatives.

1.1 Understanding how machines learn

When we talk about human learning, we distinguish between rote learning, or memorization, and true intelligence. Memorizing a telephone number or a set of instructions is undoubtedly learning. But when we say *learning*, we frequently mean something more.

When children play in groups, they observe how others respond to their actions. Their future social behaviors are informed by this experience. But they don't rewind and replay their past. Rather, certain recognizable features of their interactions—playground, classroom, Mom, Dad, siblings, friends, strangers, adults, children, indoors, outdoors—provide clues. They assess each new situation based on the features it has in common with past situations. Their learning is more than gathering knowledge. They're building what might be called *insight*.

Imagine teaching a child the difference between dogs and cats by using flashcards. You show a card, the child makes a choice, and you place the card in one of two piles for right and wrong choices, respectively. As the child practices, his performance improves. Interestingly, it isn't necessary to first teach the child techniques for cat and dog recognition. Human cognition has built-in classification mechanisms. All that's needed are *examples*. After the child is proficient with the flashcards, he'll be able to classify not only the images on the flashcards, but also most any cat or dog image, not to mention the real thing. This ability to *generalize*, to apply knowledge gained through training to new unseen examples, is a key characteristic of both human and machine learning.

Of course, human learning is far more sophisticated than even the most advanced machine-learning algorithms, but computers have the advantage of greater capacity to memorize, recall, and process data. Their experience comes in the form of historical data that's processed—using the techniques described in this book—to create and optimize, through experience, algorithms that embody, if not true insight, at least the ability to generalize.

Analogies between human and machine learning naturally bring to mind the term *artificial intelligence* (AI) and the obvious question, "What's the difference between AI and machine learning?" There's no clear consensus on this matter, but most (not all) agree that ML is one form of AI, and that AI is a far broader subject encompassing such areas as robotics, language processing, and computer vision systems. To increase the ambiguity even further, machine learning is being applied in many of these adjacent AI fields with increasing frequency. We can say that the discipline of machine learning refers to *a specific body of knowledge and an associated set of techniques*. It's fairly clear what is, and what isn't, machine learning, whereas the same can't always be said for artificial intelligence. Paraphrasing Tom Mitchell's often-cited definition, a computer program is said to learn if its performance of a certain task, as measured by a computable score, improves with experience.[2]

Kaggle, a machine-learning consultancy, ran a competition for the most accurate program for classifying whether images depicted a dog or cat.[3] Competitors were provided 25,000 example images for training. Each was labeled to indicate the species depicted. After all the competitors had trained their algorithms, they were tested on their ability to classify 12,500 unlabeled test images.

When we explain the Kaggle competition to people, they often respond by reflecting on the sorts of rules one might apply to accomplish dog and cat recognition. Cats' ears are triangular and stand up; dogs' ears are floppy—but not always. Try to imagine how you might explain to a person who had never seen a dog or a cat how to tell the difference, without showing any examples.

People use a variety of methods involving shapes, colors, textures, proportions, and other features to learn, and to generalize, from examples. Machine learning also employs a variety of strategies, in various combinations, depending on the problem at hand.

These strategies are embodied in collections of algorithms developed over the course of recent decades by academics and practitioners in disciplines ranging from statistics, computer science, robotics, and applied mathematics, to online search, entertainment, digital advertising, and language translation. They are diverse and have various strengths and weaknesses. Some of them are classifiers. Others predict a numeric measurement. Some measure the similarity or difference of comparable entities (for example, people, machines, processes, cats, dogs). What the algorithms have in common is learning from examples (experience) and the capacity to apply what they've learned to new, unseen cases—the ability to generalize.

In the cats and dogs competition, during the learning phase, competitors' programs tried over and over to perform correct classifications using many algorithms. In each of the millions of iterations of the learning process, the programs performed the classification, measured their results, and then adjusted the process ever so slightly, searching for incremental improvements. The winner classified 98.914% of the unseen test images correctly. That's pretty good, considering the human error rate

[2] Tom Mitchell, *Machine Learning* (McGraw Hill, 1997), 2. "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

[3] See "Dogs vs. Cats" at www.kaggle.com/c/dogs-vs-cats.

is around 7%. Figure 1.1 illustrates the process. The machine-learning process analyzes labeled images and builds a model that is, in turn, used by the *recall* (prediction) process to classify unlabeled images. There's one mislabeled cat in the example.

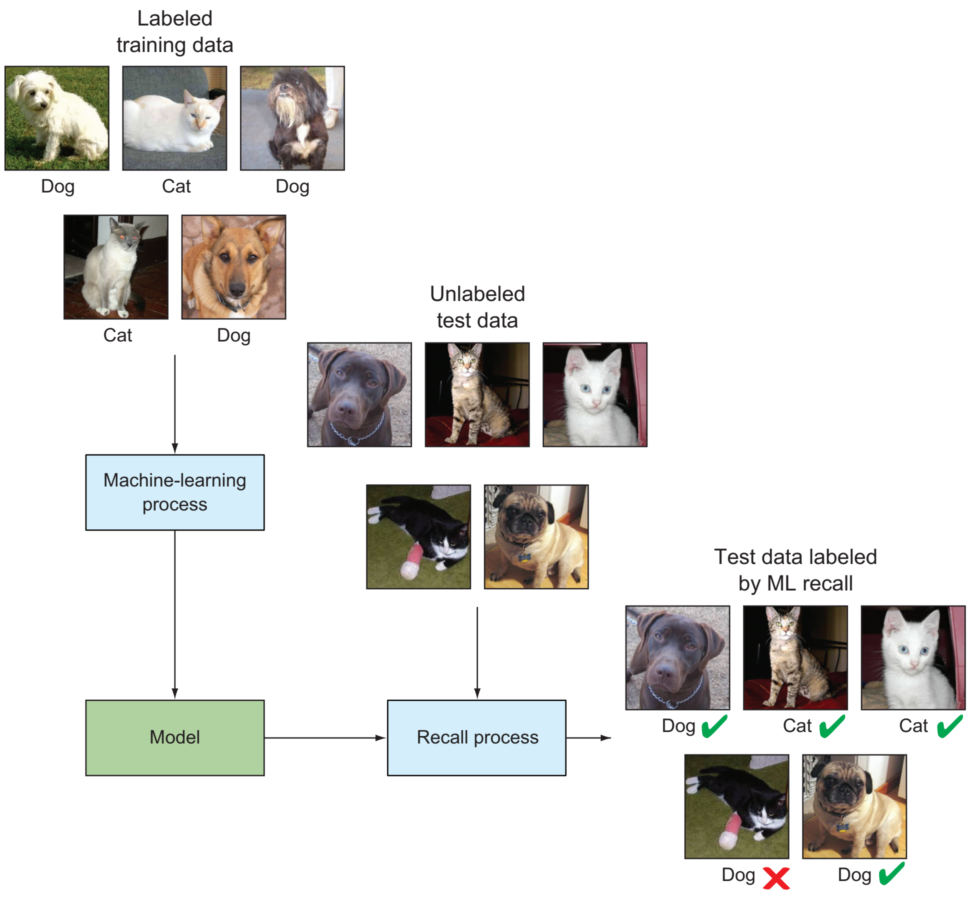


Figure 1.1 Machine-learning process for the cats and dogs competition

Please note that what we've described here is *supervised* machine learning, and it's not the only type of ML. We discuss other types later.

Machine learning can be applied to a wide range of business problems, from fraud detection, to customer targeting and product recommendation, to real-time industrial monitoring, sentiment analysis, and medical diagnosis. It can take on problems that can't be managed manually because of the huge amount of data that must be processed. When applied to large datasets, ML can sometimes find relationships so subtle that no amount of manual scrutiny would ever discover them. And when many such "weak" relationships are combined, they become strong predictors.

The process of learning from data, and subsequently using the acquired knowledge to inform future decisions, is extremely powerful. Indeed, machine learning is rapidly becoming the engine that powers the modern data-driven economy.

Table 1.1 describes widely used supervised machine-learning techniques and some of their practical applications. This isn't an exhaustive list, as the potential use cases could stretch across several pages.

Table 1.1 Use cases for supervised machine learning, organized by the type of problem

Problem	Description	Example use cases
Classification	Determine the discrete class to which each individual belongs, based on input data	Spam filtering, sentiment analysis, fraud detection, customer ad targeting, churn prediction, support case flagging, content personalization, detection of manufacturing defects, customer segmentation, event discovery, genomics, drug efficacy
Regression	Predict the real-valued output for each individual, based on input data	Stock-market prediction, demand forecasting, price estimation, ad bid optimization, risk management, asset management, weather forecasting, sports prediction
Recommendation	Predict which alternatives a user would prefer	Product recommendation, job recruiting, Netflix Prize, online dating, content recommendation
Imputation	Infer the values of missing input data	Incomplete patient medical records, missing customer data, census data

1.2 Using data to make decisions

In the following example, we describe a real-world business problem that can benefit from a machine-learning approach. We'll run through the various alternatives that are commonly used and demonstrate the advantages of the ML approach.

Imagine that you're in charge of a microlending company that provides loans to individuals who want to start small businesses in troubled communities. Early on, the company receives a few applications per week, and you're able in a few days' time to manually read each application and do the necessary background checks on each applicant to decide whether to approve each loan request. The schematic of this process is shown in figure 1.2. Your early borrowers are pleased with your short turnaround time and personal service. Word of your company starts to spread.

As your company continues to gain popularity, the number of applicants begins to increase. Soon you're receiving hundreds of applications per week. You try to stay up with the increased rate of applications by working extra hours, but the backlog of applications continues to grow. Some of your applicants grow weary of waiting and seek loans from your competitors. It's obvious to you that manually processing each application by yourself isn't a sustainable business process and, frankly, isn't worth the stress.

So what should you do? In this section, you'll explore several ways to scale up your application-vetting process to meet your increasing business needs.

1.2.1 Traditional approaches

Let's explore two traditional data analysis approaches as applied to the application-vetting process: manual analysis and business rules. For each approach, we'll walk through the process of implementing the technique and highlight the ways in which it falls short of enabling you to build a scalable business.

HIRE MORE ANALYSTS

You decide to hire another analyst to help you out. You aren't thrilled with the idea of spending some of your profit on a new hire, but with a second person vetting applications, you can process roughly twice as many applications in the same amount of time. This new analyst allows you to flush out the application backlog within a week.

For the first couple of weeks, the two of you stay up with demand. Yet the number of applications continues to grow, doubling within a month to 1,000 per week. To keep up with this increased demand, you now must hire two more analysts. Projecting forward, you determine that this pattern of hiring isn't sustainable: all of your increased revenue from new loan applicants is going directly to your new hires instead of to more-critical areas such as your microlending fund. *Hiring more analysts as demand increases hinders the growth of your business.* Further, you find that the hiring process is lengthy and expensive, sapping your business of more of its revenue. Finally, each new hire is less experienced and slower at processing applications than the last, and the added stress of managing a team of individuals is wearing on you.

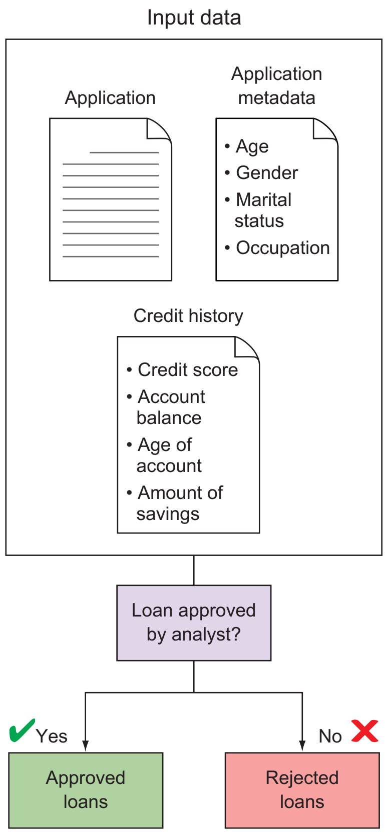


Figure 1.2 The loan-approval process for the microlending example

Aside from the obvious disadvantage of increased cost, people bring all sorts of conscious and unconscious biases to the decision-making process. To ensure consistency, you might develop detailed guidelines for the approval process and implement an extensive training program for new analysts, but this adds still more cost and probably doesn't eliminate the bias.

Employ business rules

Imagine that of the 1,000 loans whose repayment date has passed, 70% were repaid on time. This is shown in figure 1.3.

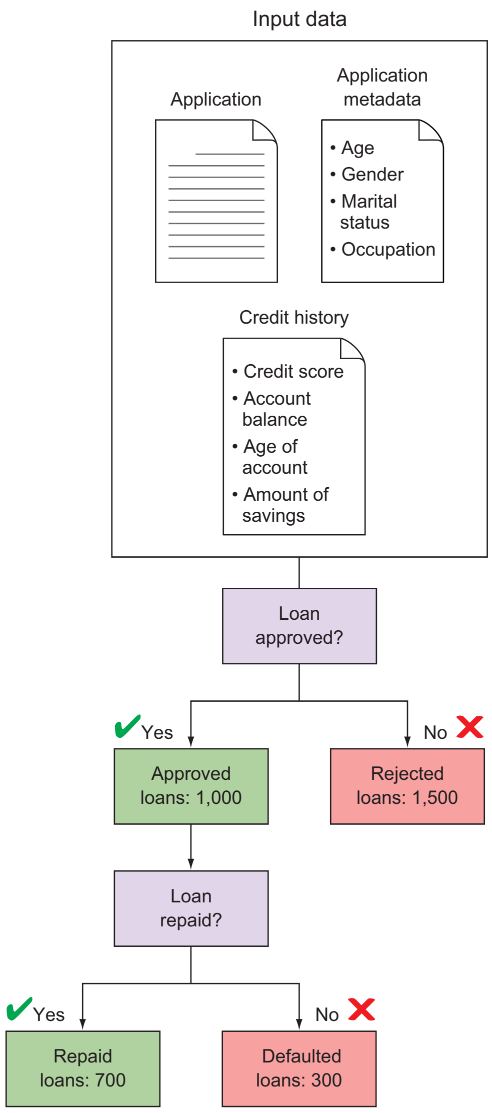


Figure 1.3 After a few months of business and 2,500 loan applications, 1,000 were approved, of which 700 applicants repaid the loan on time and the other 300 defaulted. This initial set of observed information is critical to start building automation into your loan-approval process.

You're now in a position to begin looking for trends between the applicant data and incidence of loan repayment. In particular, you perform a manual search for a set of filtering rules that produces a subset of "good" loans that were primarily paid on time. Through the process of manually analyzing hundreds of applications, you've gained extensive experience about what makes each application good or bad.[4] Through some introspection and back-testing of loan repayment status, you've noticed a few trends in the credit background checks data:[5]

- Most borrowers with a credit line of more than $7,500 defaulted on their loan.
- Most borrowers who had no checking account repaid their loan on time.

Now you can design a filtering mechanism to pare down the number of applications that you need to process manually through those two rules.

Your first filter is to automatically reject any applicant with a credit line of more than $7,500. Looking through your historical data, you find that 44 of the 86 applicants with a credit line of more than $7,500 defaulted on their loan. Roughly 51% of these high-credit-line applicants defaulted, compared to 28% of the rest. This filter seems like a good way to exclude high-risk applicants, but you realize that only 8.6% (86 out of 1,000) of your accepted applicants had a credit line that was so high, meaning that you'll still need to manually process more than 90% of applications. You need to do more filtering to get that number down to something more manageable.

Your second filter is to automatically accept any applicant who doesn't have a checking account. This seems to be an excellent filter, as 348 of the 394 (88%) applicants without a checking account repaid their loans on time. Including this second filter brings the percentage of applications that are automatically accepted or rejected up to 45%. Thus, you need to manually analyze only roughly half of the new incoming applications. Figure 1.4 demonstrates these filtering rules.

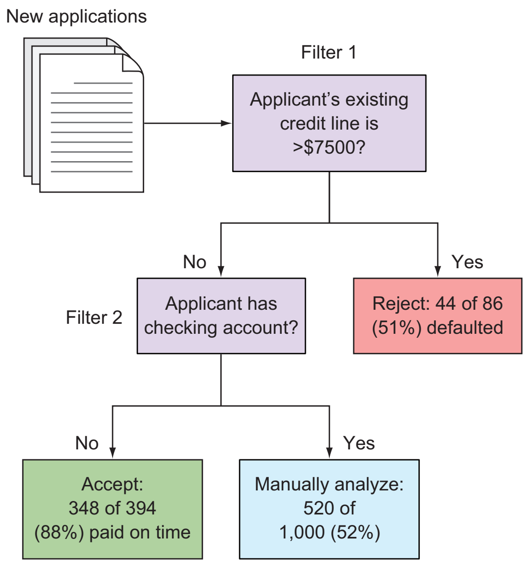

New applications
Filter 1
Applicant's existing credit line is >$7500?
No
Yes
Filter 2
Applicant has checking account?
Reject: 44 of 86 (51%) defaulted
No
Yes
Accept: 348 of 394 (88%) paid on time
Manually analyze: 520 of 1,000 (52%)

Figure 1.4 Filtering new applications through two business rules enables you to reduce manual analysis to only 52% of the incoming applications.

[4] You could also use statistical correlation techniques to determine which input data attributes are most strongly associated with the outcome event of loan repayment.

[5] In this example, we use the German Credit Data dataset. You can download this data from http://mng.bz/95r4.

With these two business rules, you can scale your business up to twice the amount of volume without having to hire a second analyst, because you now need to manually accept or reject only 52% of new applications. Additionally, based on the 1,000 applications with known outcome, you expect your filtering mechanism to erroneously reject 42 out of every 1,000 applications (4.2%) and to erroneously accept 46 of every 1,000 applications (4.6%).

As business grows, you'd like your system to automatically accept or reject a larger and larger percentage of applications without increasing losses from defaults. To do this, you again need to add more business rules. You soon encounter several problems:

- Manually finding effective filters becomes harder and harder—if not impossible—as the filtering system grows in complexity.
- The business rules become so complicated and opaque that debugging them and ripping out old, irrelevant rules becomes virtually impossible.
- The construction of your rules has no statistical rigor. You're pretty sure that better "rules" can be found by better exploration of the data, but can't know for sure.
- As the patterns of loan repayment change over time—perhaps due to changes in the population of applicants—the system doesn't adapt to those changes. To stay up to date, the system needs to be constantly adjusted.

All these drawbacks can be traced to a single debilitating weakness in a business rules approach: the system doesn't automatically learn from data.

Data-driven systems, from simple statistical models to more-sophisticated machine-learning workflows, can overcome these problems.

1.2.2 The machine-learning approach

Finally, you decide to look into an entirely automated, data-driven approach to your microlending application-vetting process. Machine learning is an attractive option because the completely automated nature of the process will allow your operation to keep pace with the increasing inflow of applications. Further, unlike business rules, ML learns the optimal decisions *directly from the data* without having to arbitrarily hard-code decision rules. This graduation from rules-based to ML-based decision making means that your decisions will be more accurate and will improve over time as more loans are made. You can be sure that your ML system produces optimized decisions with minimal handholding.

In machine learning, the data provides the foundation for deriving insights about the problem at hand. To determine whether to accept each new loan application, ML uses historical *training data* to predict the best course of action for each new application. To get started with ML for loan approval, you begin by assembling the training data for the 1,000 loans that have been granted. This training data consists of the input data for each loan application, along with the known outcome of whether each loan was repaid on time. The input data, in turn, consists of a set of *features*—numerical or categorical metrics that capture the relevant aspects of each application—such as the applicant's credit score, gender, and occupation.

In figure 1.5 historical data trains the machine-learning model. Then, as new loan applications come in, predictions of the probability of future repayment are generated instantaneously from the application data.

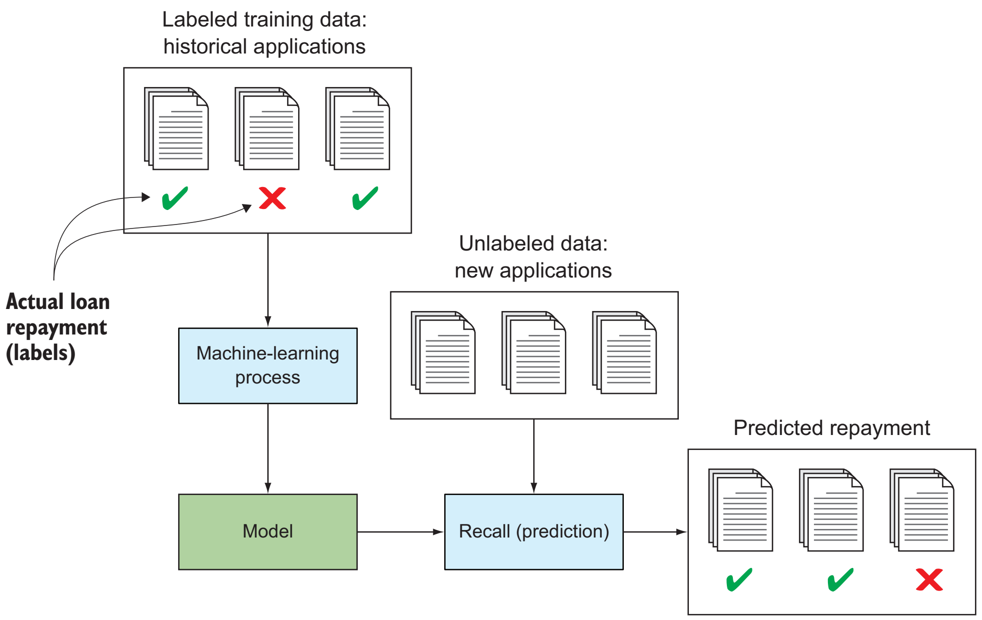

Labeled training data: historical applications
Actual loan repayment (labels)
Machine-learning process
Unlabeled data: new applications
Model
Recall (prediction)
Predicted repayment

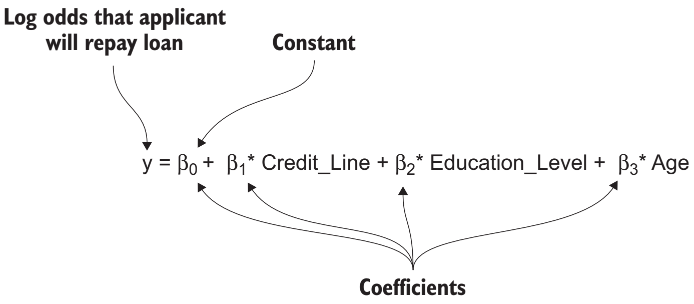
Figure 1.5 Basic ML workflow, as applied to the microloan example

ML modeling, then, determines how the input data for each applicant can be used to *best predict* the loan outcome. By finding and using patterns in the training set, ML produces a model (you can think of this as a black box, for now) that produces a prediction of the outcome for each new applicant, based on that applicant's data.

The next step is to select an ML algorithm to use. Machine learning comes in many flavors, ranging from simple statistical models to more-sophisticated approaches. Here, we compare two examples: the first is a simple parametric model, and the second a nonparametric ensemble of classification trees. Don't let the terminology scare you. Machine learning employs a lot of algorithms and lots of ways to categorize them, as you'll soon see.

Most traditional statistical business models fall into the first category. These parametric models use simple, fixed equations to express the relationship between the outcome and the inputs. Data is then used to learn the best values of the unknown terms in the equation. Approaches such as linear regression, logistic regression, and autoregressive models all fit under this category. Regression models are covered in more detail in chapter 3.

In this example, you could use logistic regression to model the loan-approval process. In logistic regression, the logarithm of the odds (the *log odds*) that each loan is repaid is modeled as a linear function of the input features. For example, if each new application contains three relevant features—the applicant's credit line, education level, and age—then logistic regression attempts to predict the log odds that the applicant will default on the loan (we'll call this y) via this equation:


Log odds that applicant will repay loan
Constant
y = β0 + β1* Credit_Line + β2* Education_Level + β3* Age
Coefficients

Log odds

The odds ratio is one way of expressing probability. You've undoubtedly heard someone say that a (favorite) team's chance of winning is 3 to 1. Odds are the probability of success (for example, winning) divided by the probability of failure (losing). Mathematically, this can be expressed as follows:

Odds(A) = P(A) / P(~A) = The probability of A divided by the probability of not A

So 3-to-1 odds is equivalent to 0.75 / 0.25 = 3 and log(3) = 0.47712…

If A were a fair coin toss, the odds of heads would be 0.5 / 0.5 = 1. Log(1) = 0. It turns out that the log(Odds) can take on any real-valued number. A log odds value near $-\infty$ denotes a highly unlikely event. A value near ∞ indicates near certainty, and log(1) = 0 indicates an even random change. Using log-odds instead of regular probabilities is a mathematical trick that makes certain computations easier, because unlike probabilities, they're not limited to values between 0 and 1.

The optimal values of each coefficient of the equation (in this case, β_0, β_1, β_2, and β_3) are learned from the 1,000 training data examples.

When you can express the relationship between inputs and outputs in a formula like this one, predicting the output (y) from the inputs (credit line, education level, and age) is easy. All you have to do is figure out which values of β_1, β_2, and β_3 yield the best result when using your historical data.

But when the relationship between the inputs and the response are complicated, models such as logistic regression can be limited. Take the dataset in the left panel of figure 1.6, for example. Here, you have two input features, and the task is to classify each data point into one of two classes. The two classes are separated in the two-dimensional feature space by a nonlinear curve, the *decision boundary* (depicted by the curve in the figure). In the center panel, you see the result of fitting a logistic regression model on this dataset. The logistic regression model comes up with a straight line that separates the two regions, resulting in many classification errors (points in the wrong region).

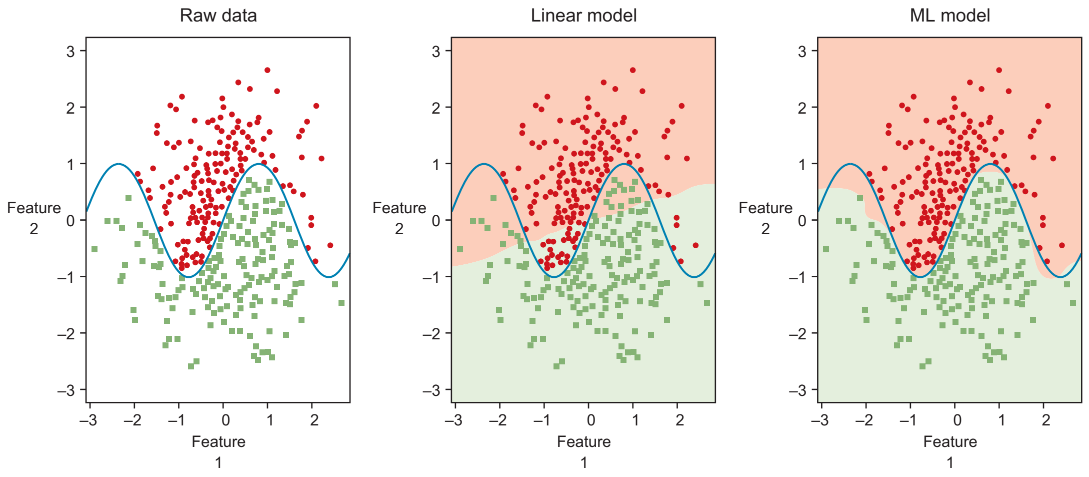


Figure 1.6 In this two-class classification, individual data points can belong to either the round class or the square class. This particular data lies in a two-dimensional feature space having a nonlinear decision boundary that separates the classes, denoted by the curve. Whereas a simple statistical model does quite poorly at accurately classifying the data (center), an ML model (right) is able to discover the true class boundary with little effort.

The problem here is that the model depicted in the center panel is attempting to explain a complicated, nonlinear phenomenon with a simple *parametric* model. The formal definition of parametric versus nonparametric models is complex and too mathematical for this book, but the gist is that parametric models work well when you have prior understanding of the relationship between your inputs and the response you're trying to predict. If you know enough about the nonlinear relationship, you may be able to transform your inputs or response variables so that a parametric model will still work. For example, if the rate at which a certain disease is observed within a population is higher for older people, you might find a linear relationship between the probability of contracting the disease and the square of the subject's age. But in the real world, you're often presented with problems for which such transformations aren't possible to guess.

What you need are more flexible models that can automatically discover complex trends and structure in data without being told what the patterns look like. This is where *nonparametric* machine-learning algorithms come to the rescue. In the right-hand panel of figure 1.6, you see the result of applying a nonparametric learning algorithm (in this case, a *random forest classifier*) to the problem. Clearly, the predicted decision boundary is much closer to the true boundary, and as a result, the classification accuracy is much higher than that of the parametric model.

Because they attain such high levels of accuracy on complicated, high-dimensional, real-world datasets, nonparametric ML models are the approach of choice for many data-driven problems. Examples of nonparametric approaches include some of the most widely used methods in machine learning, such as k-nearest neighbors, kernel smoothing, support vector machines, decision trees, and ensemble methods. We describe all of these approaches later in the book, and the appendix provides an overview of some important algorithms. Linear algorithms have other properties that make them attractive in some cases, though. They can be easier to explain and reason about, and they can be faster to compute and scale to larger datasets.

Further reading

The textbook *An Introduction to Statistical Learning* by Gareth James et al. (Springer, 2013) provides a detailed introduction to the most commonly used approaches in machine learning, at a level that's accessible to readers without a background in statistics or mathematics. A PDF version is available on the author's website (www-bcf.usc.edu/~gareth/ISL/).

Returning to the microlending problem, the best choice for scaling up your business is to employ a nonparametric ML model. The model may find the exact same rules as those you initially found manually, but chances are that they'll be slightly different in order to optimize the statistical gains. Most likely, the ML model will also automatically find other and deeper relationships between input variables and the desired outcome that you otherwise wouldn't have thought about.

In addition to providing an automated workflow, you may also attain higher accuracy, which translates directly to higher business value. Imagine that a nonparametric ML model yields 25% higher accuracy than a logistic regression approach. In this case, your ML model will make fewer mistakes on new applications: accepting fewer applicants who won't repay their loan and rejecting fewer applicants who would have repaid their loan. Overall, this means a higher average return on the loans that you do make, enabling you to make more loans overall and to generate higher revenues for your business.

We hope this gives you a taste of the power that machine learning can bring you. Before we move on to defining our basic machine-learning workflow, we'll enumerate a few advantages of machine learning, as well as a few challenges with this approach.

1.2.3 Five advantages to machine learning

To wrap up our discussion of the microlending example, we list some of the most prominent advantages to using a machine-learning system, as compared to the most common alternatives of manual analysis, hardcoded business rules, and simple statistical models. The five advantages of machine learning are as follows:

- *Accurate*—ML uses data to discover the optimal decision-making engine for your problem. As you collect more data, the accuracy can increase automatically.
- *Automated*—As answers are validated or discarded, the ML model can learn new patterns automatically. This allows users to embed ML directly into an automated workflow.
- *Fast*—ML can generate answers in a matter of milliseconds as new data streams in, allowing systems to react in real time.
- *Customizable*—Many data-driven problems can be addressed with machine learning. ML models are custom built from your own data, and can be configured to optimize whatever metric drives your business.
- *Scalable*—As your business grows, ML easily scales to handle increased data rates. Some ML algorithms can scale to handle large amounts of data on many machines in the cloud.

1.2.4 Challenges

Naturally, achieving these benefits involves a few challenges. Depending on the size and shape of the business problem, the degree of attendant difficulty ranges from child's-play trivial to Hannibal-crossing-the-Alps colossal.

Most prominent is acquiring data in a usable form. It has been estimated that data scientists spend 80% of their time on data preparation.[6] You've undoubtedly heard that businesses capture vastly greater quantities of data than ever before, and they do. You also may have heard this data referred to as the "exhaust" of business processes. In other words, our new treasure trove of data wasn't designed to meet the input needs of our ML systems. Extracting useful data from the residue can be tedious and messy work.

A related challenge is formulating the problem so that machine learning can be applied, and will yield a result that's actionable and measurable. In our example, the goal is clear: predict who will repay and who will default. The classification is easy to apply, and the outcome is easily measured. Fortunately, some real-world problems are this simple; for example, given everything we know about prospective customers (and we have a lot of data), predict whether they'll purchase our product. This is low-hanging fruit.

A more difficult example might be along these lines: find the optimum media mix and combination of advertising units to increase brand awareness for a new product line. Simply formulating the problem requires constructing a way of measuring brand

[6] Steve Lohr, "For Big-Data Scientists, 'Janitor Work' Is Key Hurdle to Insights," *New York Times*, August 17, 2014, http://mng.bz/7W8n.

awareness, an understanding of the alternative media options under consideration, and data that reflects pertinent experience with the alternatives and associated outcomes.

When the outcome you're trying to predict is complicated, choosing the algorithm and how to apply it may be an enormous effort in itself. Cardiology researchers working to predict the likelihood of postoperative complications have a mind-boggling set of data for each patient, but ML algorithms don't naturally slurp up electrocardiography (EKG) data and DNA sequences. *Feature engineering* is the process of transforming inputs such as these into predictive features.

We'd be remiss if we didn't mention the bane of the predictive modeler's existence: a model that fits the training data perfectly, but falls flat on its face when it's used to do real predictions on data that isn't in the training set. The problem is most often *overfitting*.

You'll see that machine learning can solve a great variety of problems, some much more easily than others. You may also notice that the value of the solution isn't always proportional to the effort required. And indeed, ML isn't a silver bullet for any problem. But as you'll see in this book, machine learning is the perfect choice for many real-world, data-driven problems.

1.3 Following the ML workflow: from data to deployment

In this section, we introduce the main workflow for integrating machine-learning models into your applications or data pipelines. The *ML workflow* has five main components: data preparation, model building, evaluation, optimization, and predictions on new data. The application of these steps has an inherent order, but most real-world machine-learning applications require revisiting each step multiple times in an iterative process. These five components are detailed in chapters 2 through 4, but we outline them in this introduction to whet your appetite for getting started. Figure 1.7

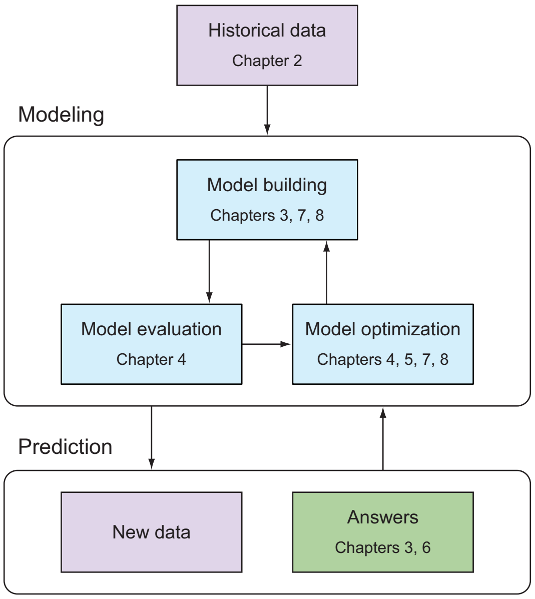
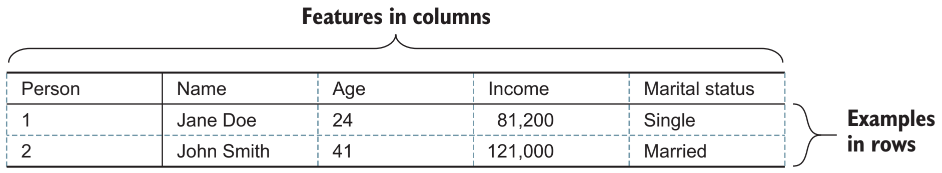


Figure 1.7 The workflow of real-world machine-learning systems. From historical input data you can build a model using an ML algorithm. You then need to evaluate the performance of the model, and optimize accuracy and scalability to fit your requirements. With the final model, you can make predictions on new data.

outlines this workflow, and the following sections introduce these concepts from top to bottom. You'll see this figure a lot throughout the book as we introduce the various components of the ML workflow.

1.3.1 Data collection and preparation

Collecting and preparing data for machine-learning systems usually entails getting the data into a tabular format, if it's not already. Think of the tabular format as a spreadsheet in which data is distributed in rows and columns, with each row corresponding to an *instance* or *example* of interest, and each column representing a measurement on this instance. A few exceptions and variations exist, but it's fair to say that most machine-learning algorithms require data in this format. Don't worry; you'll deal with the exceptions as you encounter them. Figure 1.8 shows a simple dataset in this format.


Features in columns
Examples in rows

Person	Name	Age	Income	Marital status
1	Jane Doe	24	81,200	Single
2	John Smith	41	121,000	Married

Figure 1.8 In a tabular dataset, rows are called *instances* and columns represent *features*.

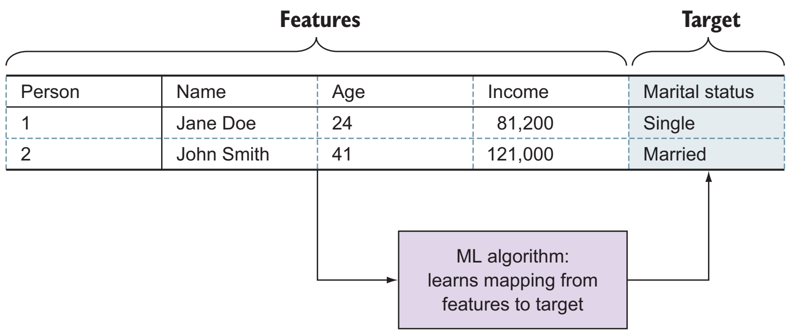
The first thing to notice about tabular data is that individual columns usually include the same type of data, and rows typically include data of various types. In figure 1.8, you can already identify four types of data: *Name* is a string variable, *Age* is an integer variable, *Income* is a floating-point variable, and *Marital status* is a categorical variable (taking on a discrete number of categories). Such a dataset is called *heterogeneous* (in contrast to homogeneous), and in chapter 2 we explain how and why we'll coerce some of these types of data into other types, depending on the particular machine-learning algorithm at hand.

Real-world data can be "messy" in a variety of other ways. Suppose that a particular measurement is unavailable for an instance in the data-gathering phase, and there's no way of going back to find the missing piece of information. In this case, the table will contain a *missing value* in one or more cells, and this can complicate both model building and subsequent predictions. In some cases, humans are involved in the data-gathering phase, and we all know how easy it is to make mistakes in repetitive tasks such as data recording. This can lead to some of the data being flat-out wrong, and you'll have to be able to handle such scenarios, or at least know how well a particular algorithm behaves in the presence of misleading data. You'll look closer at methods for dealing with missing and misleading data in chapter 2.

1.3.2 Learning a model from data

The first part of building a successful machine-learning system is to ask a question that can be answered by the data. With this simple Person table, you could build an ML model that could predict whether a person is married or single. This information would be useful for showing relevant ads, for example.

In this case, you'd use the *Marital status* variable as the *target*, or *label*, and the remaining variables as *features*. The job of the ML algorithm will then be to find how the set of input features can successfully predict the target. Then, for people whose marital status is unknown, you can use the model to predict marital status based on the input variables for each individual. Figure 1.9 shows this process on our toy dataset.



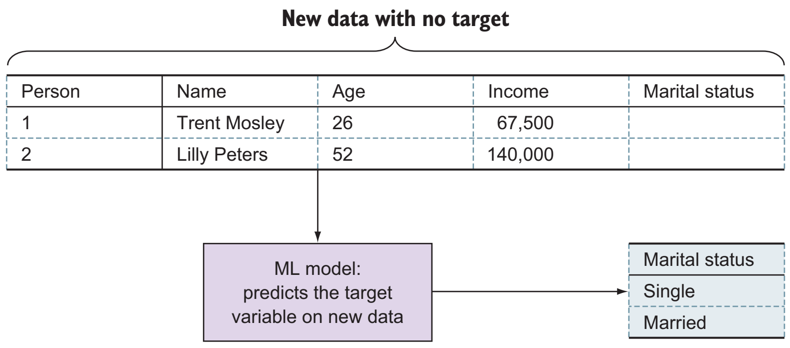
Person	Name	Age	Income	Marital status
1	Jane Doe	24	81,200	Single
2	John Smith	41	121,000	Married

Figure 1.9 The machine-learning modeling process

At this point, think of the ML algorithm as a magical box that performs the mapping from input features to output data. To build a useful model, you'd need more than two rows. One of the advantages of machine-learning algorithms, compared with other widely used methods, is the ability to handle many features. Figure 1.9 shows only four features, of which the *Person* ID and *Name* probably aren't useful in predicting marital status. Some algorithms are relatively immune to uninformative features, whereas others may yield higher accuracy if you leave those features out. Chapter 3 presents a closer look at types of algorithms and their performance on various kinds of problems and datasets.

It's worth noting, however, that valuable information can sometimes be extracted from seemingly uninformative features. A location feature may not be informative in itself, for example, but can lead to informative features such as population density. This type of data enhancement, called *feature extraction*, is important in real-world ML projects and is the topic of chapters 5 and 7.

With our ML model in hand, you can now make predictions on new data—data for which the target variable is unknown. Figure 1.10 shows this process, using the magic-box model built in figure 1.9.



Person	Name	Age	Income	Marital status
1	Trent Mosley	26	67,500	
2	Lilly Peters	52	140,000	

Marital status
Single
Married

Figure 1.10 Using the model for prediction on new data

The target predictions are returned in the same form as they appeared in the original data used to learn the model. Using the model to make predictions can be seen as filling out the blank target column of the new data. Some ML algorithms can also output the probabilities associated with each class. In our married/single example, a *probabilistic* ML model would output two values for each new person: the probability of this person being married and the probability of the person being single.

We left out a few details on the way here, but in principle you've just architected your first ML system. Every machine-learning system is about building models and using those models to make predictions. Let's look at the basic machine-learning workflow in pseudocode to get another view of how simple it is.

Listing 1.1 Initial structure of an ML workflow program

```
data = load_data("data/people.csv")
model = build_model(data, target="Marital status")
new_data = load_data("data/new_people.csv")
predictions = model.predict(new_data)
```

Although we haven't programmed any of these functions yet, the basic structure is in place. By chapter 3, you'll understand these steps; the rest of the book (chapters 4 through 10) is about making sure you're building the best model for the problem at hand.

1.3.3 *Evaluating model performance*

Rarely is an ML system put to use without some kind of validation of the performance of the model. Even though we've skipped a lot of details in this chapter, let's pretend that you know how to build a model and make predictions. You can now apply a clever trick to get some sense of how well your model is working before you use it to predict on new data.

You take out some of the data and pretend that you don't know the target variable. You then build a model on the remaining data and use the held-out data (testing data) to make predictions. Figure 1.11 illustrates this model-testing process.

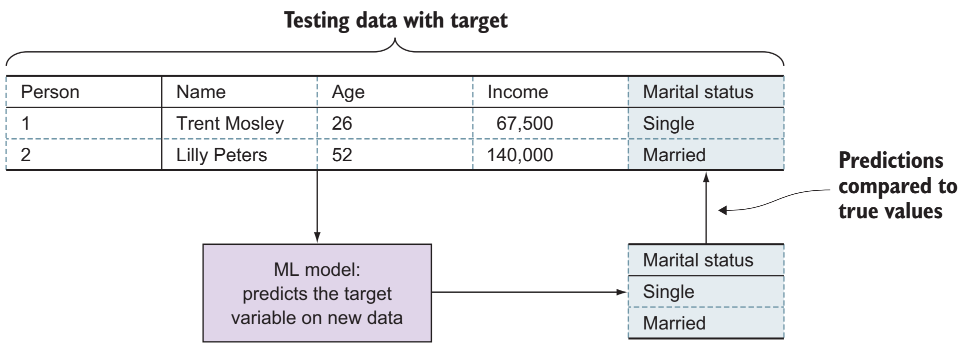


Figure 1.11 When using a testing set to evaluate model performance, you "pretend" that the target variable is unknown and compare the predictions with the true values.

Let's also look at the pseudocode for this workflow.

Listing 1.2 Our ML workflow program with model evaluation

```
data = load_data(...)
training_data, testing_data = split_data(data)
model = build_model(training_data, target="Marital status")
true_values = testing_data.extract_column("Marital status")
predictions = model.predict(testing_data)
accuracy = compare_predictions(predictions, true_values)
```

You can now compare the predicted results with the known "true" values to get a feeling for the accuracy of the model. In the pseudocode, this functionality is hidden behind the `compare_predictions` function, and most of chapter 4 is dedicated to understanding how this function looks for various types of machine-learning problems.

1.3.4 Optimizing model performance

The last piece of the essential machine-learning puzzle is also covered in chapter 4: how to use the results of your model evaluation to go back and make the model better. You can achieve better model accuracy in three ways:

- *Tuning the model parameters*—ML algorithms are configured with parameters specific to the underlying algorithm, and the optimal value of these parameters often depends on the type and structure of the data. The value of each parameter, or any of them combined, can have an impact on the performance of the model. We introduce various ways to find and select the best parameter values, and show how this can help in determining the best algorithm for the dataset in question.

- *Selecting a subset of features*—Many ML problems include a large number of features, and the noise from those features can sometimes make it hard for the algorithm to find the real signal in the data, even though they might still be informative on their own. For many ML problems, having a lot of data is a good thing; but it can sometimes be a curse. And because you don't know beforehand when this will affect your model performance, you have to carefully determine the features that make up the most general and accurate model.
- *Preprocessing the data*—If you search the internet for machine-learning datasets, you'll find easy-to-use datasets that many ML algorithms can be quickly applied to. Most real-world datasets, however, aren't in such a clean state, and you'll have to perform cleaning and processing, a process widely referred to as *data munging* or *data wrangling*. The dataset may include names that are spelled differently, although they refer to the same entity, or have missing or incorrect values, and these things can hurt the performance of the model. It may sound like edge cases, but you'll be surprised how often this happens even in sophisticated, data-driven organizations.

With the machine-learning essentials in place, you'll look briefly at more-advanced features in the next section before learning more details about the main components covered in this section.

1.4 Boosting model performance with advanced techniques

The previous section introduced the essential steps in any real-world machine-learning project, and now you'll look at additional techniques often used to improve model performance even further. Depending on the data and problem at hand, some of these techniques can provide significant gains in accuracy, but sometimes at the cost of speed in both training and prediction. These techniques are explained in more detail in chapters 5 through 10, but this section outlines the main ideas.

1.4.1 Data preprocessing and feature engineering

You'll look at various kinds of data and how to deal with common types of messiness in chapter 2. But in addition to this essential data cleaning, you can go a step further and extract additional value from the data that might improve your model performance.

In any problem domain, specific knowledge goes into deciding the data to collect, and this valuable domain knowledge can also be used to extract value from the collected data, in effect adding to the features of the model before model building. We call this process *feature engineering*, and when the previously introduced essential ML workflow has become second nature to you, you can find yourself spending almost all your time in this part of the optimization process. This is also the creative part of machine learning, where you get to use your knowledge and imagination to come up with ways to improve the model by digging into the data and extracting hidden value. You'll make extensive use of our statistically validated model evaluation and optimization steps to distinguish what seemed like a good idea at the time from what is actually useful. Here are a few important examples of feature engineering:

- *Dates and times*—You'll see a date or time variable in many datasets, but by themselves they're not useful for ML algorithms, which tend to require raw numbers or categories. The information might be valuable, though. If you want to predict which ad to show, it'll certainly be important to know the time of day, the day of the week, and the time of year. With feature engineering, this information can be extracted from the dates and times and made available to the model.

  Also, when dates and times appear in observations of repetitive activity, such as a user's repeated visits to a website over the course of a month or year, they can be used to compute interval durations that may be predictive. For example, on a shopping site, users might visit more frequently just prior to making a purchase to review and compare items and prices.
- *Location*—Location data, such as latitude/longitude coordinates or location names, is available in some datasets. This information can sometimes be used in itself, but you may be able to extract additional information that's useful for a specific problem. For example, if you want to predict election results in a county, you might want to extract the population density, mean income, and poverty rate to use as numbers in your model.
- *Digital media*—This is data such as text, documents, images, and video. The feature engineering that makes this kind of data usable is the difficult part of projects like the dogs and cats competition. Edges, shapes, and color spectra are first extracted from the images. Then these are classified using mathematical transformations, the output of which is a set of features usable by the classification algorithms.

Hopefully it's clear that feature engineering can be important for real-world ML projects. Chapters 5 and 7 go into much more detail, introducing specific feature-engineering techniques; you'll learn how these techniques feed into your ML workflow so your model performance improves without becoming too complex and prone to overfitting. Figure 1.12 illustrates feature-engineering integration into the larger ML workflow introduced in section 1.3.

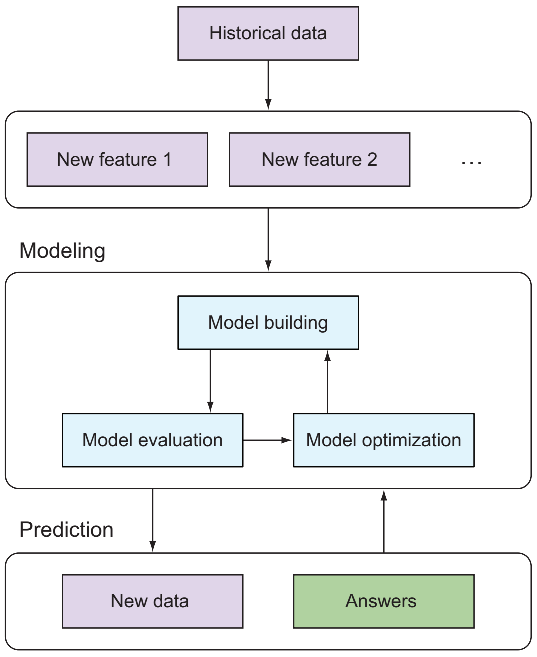


Figure 1.12 Feature-engineering phase inserted in the original ML workflow

1.4.2 Improving models continually with online methods

Most traditional ML models are static or only rarely rebuilt. But in many cases, you'll have data and predictions flowing back into the system, and you want the model to improve with time and adapt to changes in the data. Several ML algorithms support this type of *online learning*; chapter 8 introduces these algorithms and their potential pitfalls. Figure 1.13 shows how continual relearning can be integrated into the ML workflow.

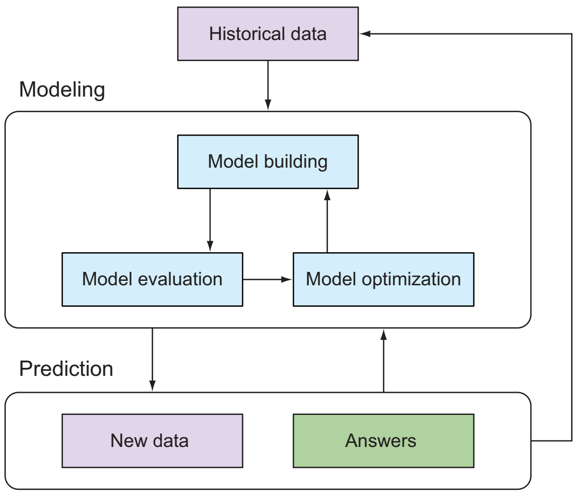


Figure 1.13 In this flow of an online ML system, predictions are fed back to the model for iterative improvements.

1.4.3 Scaling models with data volume and velocity

It's well known that datasets are increasing in size and velocity more quickly than ever. Datasets for supervised methods, in which the target answers are in the training set, have traditionally been relatively small because humans were needed in order to acquire the answers. Today, a lot of data (including answers) is produced directly by sensors, machines, or computers, and we're beginning to see requirements for scalable ML algorithms in order to handle these data volumes.

Chapter 9 presents details of machine-learning methods that are capable of scaling with growing dataset sizes; you'll see how they compare to each other and to nonscaling algorithms.

1.5 Summary

This chapter introduced machine learning as a better, more data-driven approach to making decisions. The main points to take away from this chapter are as follows:

- Machine-learning algorithms are distinguished from rule-based systems in that they create their own models based on data. Supervised ML systems generalize by learning from the features of examples with known results.
- Machine learning is often more accurate, automated, fast, customizable, and scalable than manually constructed rule-based systems.
- Machine-learning challenges include identifying and formulating problems to which ML can be applied, acquiring and transforming data to make it usable, finding the right algorithms for the problem, feature engineering, and overfitting.
- The basic machine-learning workflow consists of data preparation, model building, model evaluation, optimization, and predictions on new data.
- Online learning models continually relearn by using the results of their predictions to update themselves.

1.6 Terms from this chapter

Word	Definition
instance or example	A single object, observation, transaction, or record.
target or label	The numerical or categorical (label) attribute of interest. This is the variable to be predicted for each new instance.
features	The input attributes that are used to predict the target. These also may be numerical or categorical.
model	A mathematical object describing the relationship between the features and the target.
training data	The set of instances with a known target to be used to fit an ML model.
recall	Using a model to predict a target or label.

Word	Definition
supervised machine learning	Machine learning in which, given examples for which the output value is known, the training process infers a function that relates input values to the output.
unsupervised machine learning	Machine-learning techniques that don't rely on labeled examples, but rather try to find hidden structure in unlabeled data.
ML workflow	The stages in the ML process: data preparation, model building, evaluation, optimization, and prediction.
online machine learning	A form of machine learning in which predictions are made, and the model is updated, for each new example.

In chapter 2, you'll get into the practical matters of collecting data, preparing it for machine learning use, and using visualizations to gain the insight needed to choose the best tools and methods.

MACHINE LEARNING/PROGRAMMING

Real-World Machine Learning

Brink • Richards • Fetherolf

Machine learning systems help you find valuable insights and patterns in data, which you'd never recognize with traditional methods. In the real world, ML techniques give you a way to identify trends, forecast behavior, and make fact-based recommendations. It's a hot and growing field, and up-to-speed ML developers are in demand.

Real-World Machine Learning will teach you the concepts and techniques you need to be a successful machine learning practitioner without overdosing you on abstract theory and complex mathematics. By working through immediately relevant examples in Python, you'll build skills in data acquisition and modeling, classification, and regression. You'll also explore the most important tasks like model validation, optimization, scalability, and real-time streaming. When you're done, you'll be ready to successfully build, deploy, and maintain your own powerful ML systems.

What's Inside

- Predicting future behavior
- Performance evaluation and optimization
- Analyzing sentiment and making recommendations

No prior machine learning experience assumed. Readers should know Python.

Henrik Brink, **Joseph Richards**, and **Mark Fetherolf** are experienced data scientists engaged in the daily practice of machine learning.

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit www.manning.com/books/real-world-machine-learning

Free eBook
SEE INSERT

"This is that crucial *other* book that many old hands wish they had back in the day."
—From the Foreword by Beau Cronin, 21 Inc.

"A comprehensive guide on how to prepare data for ML and how to choose the appropriate algorithms."
—Michael Lund, iCodeIT

"Very approachable. Great information on data preparation and feature engineering, which are typically ignored."
—Robert Diana
RSI Content Solutions

ISBN-13: 978-1-61729-192-0
ISBN-10: 1-61729-192-7
54999
9 781617 291920

MANNING $49.99 / Can $57.99 [INCLUDING eBOOK]