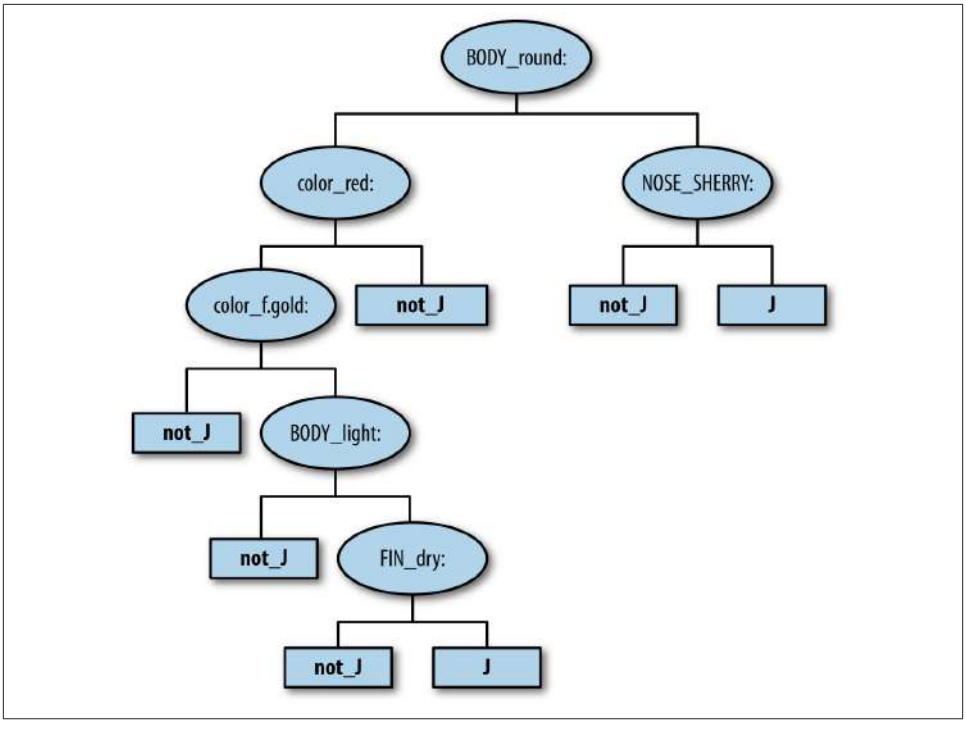

Figure 6-14. The decision tree learned from cluster J on the Scotches data. In this tree, the right branches from each node correspond to the presence of the node's attribute (a value of 1 in the binary representation). The left branches correspond to the absence of the node's attribute (a value of 0). So, the leftmost leaf corresponds to the segment of the population with round body and sherry nose, and the whiskeys in this segment are mostly from cluster J.

From this tree we concentrate only on the leaves labeled **J** (ignoring the ones labeled **not_J**). There are only two such leaves. Tracing paths from the root to these leaves, we can extract the two rules:

1. `(ROUND_BODY = 1) AND (NOSE_SHERRY = 1)` ⇒ `J`
2. `(ROUND_BODY = 0) AND (color_red = 0) AND (color_f.gold = 1) AND (BODY_light = 1) AND (FIN_dry = 1)` ⇒ `J`

Translating these loosely into English, the **J** cluster is distinguished by Scotches having either:

1. A round body and a sherry nose, or

2. A full gold (but not red) color with a light (but not round) body and a dry finish.

Is this description of cluster J better than the one given by Lapointe and Legendre, above? You can decide which you prefer, but it's important to point out they are different *types* of descriptions. Lapointe and Legendre's is a **characteristic** description; it describes what is typical or characteristic of the cluster, ignoring whether other clusters might share some of these characteristics. The one generated by the decision tree is a **differential** description; it describes only what differentiates this cluster from the others, ignoring the characteristics that may be shared by whiskeys within it. To put it another way: characteristic descriptions concentrate on intergroup commonalities, whereas differential descriptions concentrate on intragroup differences. Neither is inherently better —it depends on what you're using it for.

Stepping Back: Solving a Business Problem Versus Data Exploration

We now have seen various examples of our fundamental concepts of data science in action. You may have realized that the clustering examples seem somehow different from the predictive modeling examples, and even the examples of finding similar objects. Let's examine why.

In our predictive modeling examples, as well as our examples of using similarity directly, we focused on solving a very specific business problem. As we have emphasized, one of the fundamental concepts of data science is that one should work to define as precisely as possible the goal of any data mining. Recall the CRISP data mining process, replicated in Figure 6-15. We should spend as much time as we can in the business understanding/data understanding mini-cycle, until we have a concrete, specific definition of the problem we are trying to solve. In predictive modeling applications, we are aided by our need to define the target variable precisely, and we will see in Chapter 7 that we can get more and more precise about defining the problem as we get more sophisticated in our understanding of data science. In our similarity-matching examples, again we had a very concrete notion of what exactly we were looking for: we want to find similar companies to optimize our efforts, and we will define specifically what it means to be similar. We want to find similar whiskeys—specifically in terms of taste—and we again work to gather and represent the data so that we can find exactly these. Later in the book we will discuss how we often expend considerable effort applying data science frameworks to decompose business problems into multiple, well-defined components, each of which we might apply data science methods to solve.

However, not all problems are so well defined. What do we do when in the business understanding phase we conclude: *we would like to explore our data, possibly with only a vague notion of the exact problem we are solving*? The problems to which we apply clustering often fall into this category. We want to perform *unsupervised* segmentation:

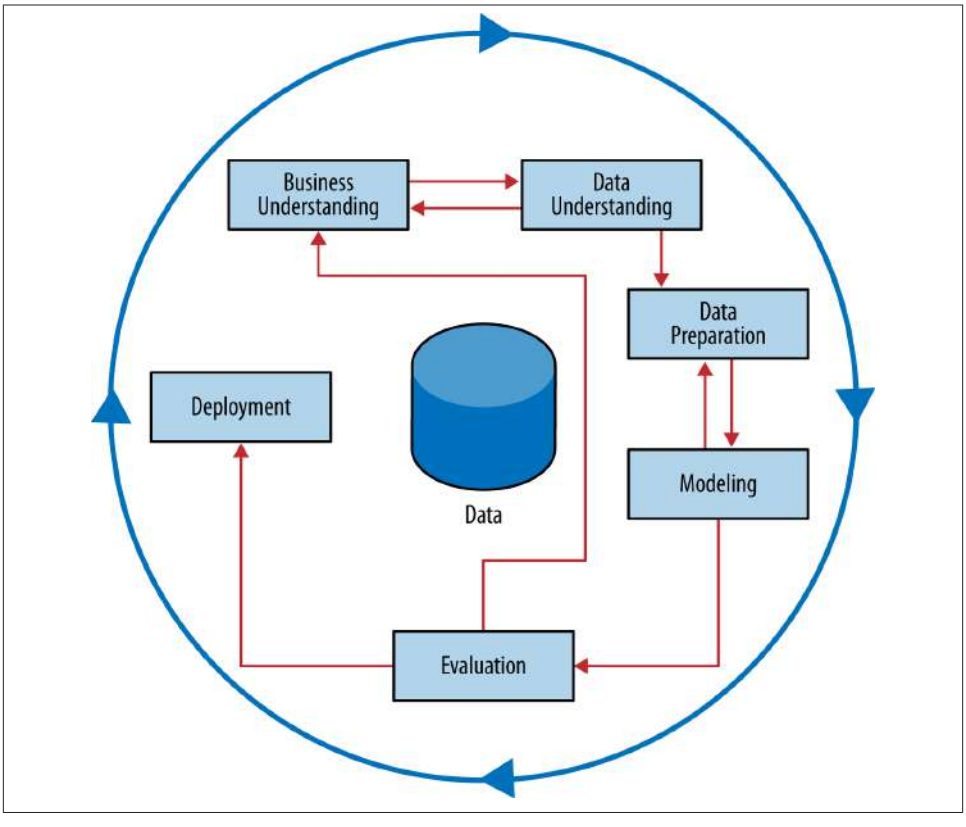

Figure 6-15. The CRISP data mining process.

finding groups that "naturally" occur (subject, of course, to how we define our similarity measures).

For the sake of discussion, let's simplify by separating our problems into supervised (e.g., predictive modeling) and unsupervised (e.g., clustering). The world is not so cut-and-dried and just about any of the data mining techniques we have presented could be used for data exploration, but the discussion will be much clearer if we do simply separate into supervised versus unsupervised. There is a direct trade-off in where and how effort is expended in the data mining process. For the supervised problems, since we spent so much time defining precisely the problem we were going to solve, in the Evaluation stage of the data mining process we already have a clear-cut evaluation question: do the results of the modeling seem to solve the problem we have defined? For example, if we had defined our goal as improving prediction of defection when a customer's contract is about to expire, we could assess whether our model has done this.

In contrast, unsupervised problems often are much more exploratory. We may have a notion that if we could cluster companies, news stories, or whiskeys, we would understand our business better, and therefore be able to improve something. However, we may not have a precise formulation. We should not let our desire to be concrete and precise keep us from making important discoveries from data. But there is a trade-off. The tradeoff is that for problems where we did not achieve a precise formulation of the problem in the early stages of the data mining process, we have to spend more time later in the process—in the Evaluation stage.

For clustering, specifically, it often is difficult even to understand what (if anything) the clustering reveals. Even when the clustering does seem to reveal interesting information, it often is not clear how to use that to make better decisions. Therefore, for clustering, additional creativity and business knowledge must be applied in the Evaluation stage of the data mining process.

Ira Haimowitz and Henry Schwartz (1997) show a concrete example of how clustering was used to improve decisions about how to set credit lines for new credit customers. They clustered existing GE Capital customers based on similarity in their use of their cards, payment of their bills, and profitability to the company. After some work, they settled on five clusters that represented very different consumer credit behavior (e.g., those who spend a lot but pay off their cards in full each month versus those who spend a lot and keep their balance near their credit limit). These different sorts of customers can tolerate very different credit lines (in the two examples, extra care must be taken with the latter to avoid default). The problem with using this clustering immediately for decision making is that the data are not available when the initial credit line is set. Briefly, Haimowitz and Schwarz took this new knowledge and cycled back to the beginning of the data mining process. They used the knowledge to define a precise predictive modeling problem: using data that *are* available at the time of credit approval, predict the probability that a customer will fall into each of these clusters. This predictive model then can be used to improve initial credit line decisions.

Summary

The fundamental concept of similarity between data items occurs throughout data mining. In this chapter we first discussed a wide variety of uses of similarity ranging from finding similar entities (or objects) based on their data descriptions, to predictive modeling, to clustering entities. We discussed these various uses and illustrated with examples.

A very common proxy for the similarity of two entities is the distance between them in the instance space defined by their feature vector representation. We presented similarity and distance computations, generally and in technical detail. We also introduced a family of methods, called nearest-neighbor methods, that perform prediction tasks by calculating explicitly the similarity between a new example and a set of training

examples (with known values for the target). Once we can retrieve a set of nearest neighbors (most similar examples) we can use these for various data mining tasks: classification, regression and instance scoring. Finally, we showed how the same fundamental concept—similarity—also underlies the most common methods for unsupervised data mining: clustering.

We also discussed another important concept that raises its head once we begin to look seriously at methods (such as clustering) that are employed for more exploratory data analysis. When exploring the data, especially with unsupervised methods, we usually end up spending less time at the outset in the business understanding phase of the data mining process, but more time in the evaluation stage, and in iterating around the cycle. To illustrate, we discussed a variety of methodologies for understanding the results of clusterings.

CHAPTER 7

Decision Analytic Thinking I: What Is a Good Model?

Fundamental concepts: *Careful consideration of what is desired from data science results; Expected value as a key evaluation framework; Consideration of appropriate comparative baselines.*

Exemplary techniques: *Various evaluation metrics; Estimating costs and benefits; Calculating expected profit; Creating baseline methods for comparison.*

Recall from the beginning of Chapter 5: as a manager at MegaTelCo, you wanted to assess whether the model my consulting firm had produced was any good. Overfitting aside, how would you go about measuring that?

For data science to add value to an application, it is important for the data scientists and other stakeholders to consider carefully what they would like to achieve by mining data. This sounds obvious, so it is sometimes surprising how often it is ignored. Both data scientists themselves and the people who work with them often avoid—perhaps without even realizing it—connecting the results of mining data back to the goal of the undertaking. This may manifest itself in the reporting of a statistic without a clear understanding of why it is the right statistic, or in the failure to figuring out how to measure performance in a meaningful way.

We should be careful with such a criticism, though. Often it is not possible to measure perfectly one's ultimate goal, for example because the systems are inadequate, or because it is too costly to gather the right data, or because it is difficult to assess causality. So, we might conclude that we need to measure some surrogate for what we'd really like to measure. It is nonetheless crucial to think carefully about what we'd really like to measure. If we have to choose a surrogate, we should do it via careful, data-analytic thinking.

A challenge with writing a chapter on this topic is that every application is different. We cannot offer a single evaluation metric that is "right" for any classification problem, or regression problem, or whatever problem you may encounter. Nevertheless, there are

various common issues and themes in evaluation, and frameworks and techniques for dealing with them.

We will work through a set of such frameworks and metrics for tasks of classification (in this chapter) and instance scoring (e.g., ordering consumers by their likelihood of responding to an offer), and class probability estimation (in the following chapter). The specific techniques should be seen as examples illustrating the general concept of thinking deeply about the needs of the application. Fortunately, these specific techniques do apply quite broadly. We also will describe a very general framework for thinking about evaluation, using expected value, that can cover a very wide variety of applications. As we will show in a later chapter, it also can be used as an organizational tool for data-analytic thinking generally, all the way back to problem formulation.

Evaluating Classifiers

Recall that a classification model takes an instance for which we do not know know the class and predicts its class. Let's consider binary classification, for which the classes often are simply called "positive" and "negative." How shall we evaluate how well such a model performs? In Chapter 5 we discussed how for evaluation we should use a holdout test set to assess the generalization performance of the model. But how should we measure generalization performance?

Sidebar: Bad Positives and Harmless Negatives

In discussing classifiers, we often refer to a bad outcome as a "positive" example, and a normal or good outcome as "negative." This may seem odd to you, given the everyday definitions of positive and negative. Why, for example, is a case of fraud considered positive and a legitimate case considered negative? This terminology is conventional in many fields, including machine learning and data mining, and we will use it throughout this book. Some explanation may be useful to avoid confusion.

It is useful to think of a positive example as one worthy of attention or *alarm*, and a negative example as uninteresting or benign. For example, a medical test (which is a type of classifier) performed on a biological sample tries to detect disease or an unusual condition by examining certain aspects of the sample. If the test comes back positive it means the disease or condition is present; if the test is negative there is no cause for alarm and usually no need for treatment. Similarly, if a fraud detector finds unusual activity on a customer account and decides there is cause for alarm, this is called a positive. On the other hand, negatives (accounts with only legitimate activity) are profitable but from a fraud detection perspective they are unworthy of attention.

There are advantages to maintaining this general orientation rather than redefining the meaning of positive and negative for every domain we introduce. You can think of a classifier as sifting through a large population consisting mostly of negatives—the un-

interesting cases—looking for a small number of positive instances. By convention, then, the positive class is often rare, or at least rarer than the negative class. In consequence, the *number* of mistakes made on negative examples (the *false positive* errors) may dominate, though the *cost* of each mistake made on a positive example (a *false negative* error) will be higher.

Plain Accuracy and Its Problems

Up to this point we have assumed that some simple metric, such as classifier error rate or accuracy, was being used to measure a model's performance.

Classification accuracy is a popular metric because it's very easy to measure. Unfortunately, it is usually too simplistic for applications of data mining techniques to real business problems. This section discusses it and some of the alternatives.

The term "classifier accuracy" is sometimes used informally to mean any general measure of classifier performance. Here we will reserve *accuracy* for its specific technical meaning as the proportion of correct decisions:

$$\text{accuracy} = \frac{\text{Number of correct decisions made}}{\text{Total number of decisions made}}$$

This is equal to 1–*error rate*. Accuracy is a common evaluation metric that is often used in data mining studies because it reduces classifier performance to a single number and it is very easy to measure. Unfortunately, it is simplistic and has some well-known problems (Provost, Fawcett, & Kohavi, 1998). To understand these problems we need a way to decompose and count the different types of correct and incorrect decisions made by a classifier. For this we use the confusion matrix.

The Confusion Matrix

To evaluate a classifier properly it is important to understand the notion of *class confusion* and the *confusion matrix*, which is one sort of contingency table. A confusion matrix for a problem involving n classes is an $n \times n$ matrix with the columns labeled with actual classes and the rows labeled with predicted classes. Each example in a test set has an actual class label as well as the class predicted by the classifier (the predicted class), whose combination determines which matrix cell the instance counts into. For simplicity we will deal with two-class problems having 2×2 confusion matrices.

A confusion matrix separates out the decisions made by the classifier, making explicit how one class is being confused for another. In this way different sorts of errors may be dealt with separately. Let's differentiate between the true classes and the classes predicted by the model by using different symbols. We will consider two-class problems, and will denote the true classes as **p**(ositive) and **n**(egative), and the classes predicted by the

model (the "predicted" classes) as **Y**(es) and **N**(o), respectively (think: the model says "**Y**es, it is a positive" or "**N**o, it is not a positive").

Table 7-1. The layout of a 2 × 2 confusion matrix showing the names of the correct predictions (main diagonal) and errors (off-diagonal) entries.

	p	n
Y	True positives	False positives
N	False negatives	True negatives

In the confusion matrix, the main diagonal contains the counts of correct decisions. The errors of the classifier are the **false positives** (negative instances classified as positive) and **false negatives** (positives classified as negative).

Problems with Unbalanced Classes

As an example of how we need to think carefully about model evaluation, consider a classification problem where one class is rare. This is a common situation in applications, because classifiers often are used to sift through a large population of normal or uninteresting entities in order to find a relatively small number of unusual ones; for example, looking for defrauded customers, checking an assembly line for defective parts, or targeting consumers who actually would respond to an offer. Because the unusual or interesting class is rare among the general population, the class distribution is unbalanced or *skewed* (Ezawa, Singh, & Norton, 1996; Fawcett & Provost, 1996; Japkowicz & Stephen, 2002).

Unfortunately, as the class distribution becomes more skewed, evaluation based on accuracy breaks down. Consider a domain where the classes appear in a 999:1 ratio. A simple rule—always choose the most prevalent class—gives 99.9% accuracy. Presumably this is not satisfactory if a nontrivial solution is sought. Skews of 1:100 are common in fraud detection, and skews greater than $1{:}10^6$ have been reported in other classifier learning applications (Clearwater & Stern, 1991; Attenberg & Provost, 2010). Chapter 5 mentioned the "base rate" of a class, which corresponds to how well a classifier would perform by simply choosing that class for every instance. With such skewed domains the base rate for the majority class could be very high, so a report of 99.9% accuracy may tell us little about what data mining has really accomplished.

Even when the skew is not so great, in domains where one class is more prevalent than another accuracy can be greatly misleading. Consider again our cellular-churn example. Let's say you are a manager at MegaTelCo and as an analyst I report that our churn-prediction model generates 80% accuracy. This sounds good, but is it? My coworker reports that her model generates an accuracy of 37%. That's pretty bad, isn't it?

You might say, wait—we need more information about the data. And you would be exactly right to do so (and would be engaging in data-analytic thinking). What do we

need? Considering the line of discussion so far in this subsection, you might rightly say: we need to know what is the proportion of churn in the population we are considering. Let's say you know that in these data the baseline churn rate is approximately 10% per month. Let's consider a customer who churns to be a positive example, so within our population of customers we expect a positive to negative class ratio of 1:9. So if we simply classify everyone as negative we could achieve a base rate accuracy of 90%!

Digging deeper, you discover that my coworker and I evaluated on two different datasets. This would not be suprising if we had not coordinated our data analysis efforts. My coworker calculated the accuracy on a representative sample from the population, whereas I created artificially balanced datasets for training and testing (both common practices). Now my coworker's model looks really bad—she could have achieved 90% accuracy, but only got 37%. However, when she applies her model to my balanced data set, she also sees an accuracy of 80%. Now it's really confusing.

The bottom line is that accuracy simply is the wrong thing to measure. In this admittedly contrived example, my coworker's model (call it Model A) achieves 80% accuracy on the balanced sample by correctly identifying all positive examples but only 30% of the negative examples. My model (Model B) does this, conversely, by correctly identifying all the negative examples but only 30% of the positive examples.

Let's look at these two models more carefully, using confusion matrices as a conceptual tool. In a training population of 1,000 customers, the confusion matrices are as follows. Recall that a model's predicted classes are denoted **Y** and **N**.

Table 7-2. Confusion matrix of A

	churn	not churn
Y	500	200
N	0	300

Table 7-3. Confusion matrix of B

	churn	not churn
Y	300	0
N	200	500

Figure 7-1 illustrates these classifications on a balanced population and on a representative population. As mentioned, both models correctly classify 80% of the balanced population, but the confusion matrices and the figure show that they operate very differently. Classifier A often falsely predicts that customers will churn when they will not, while classifier B makes many opposite errors of predicting that customers will not churn when in fact they will. When applied to the original, unbalanced population of customers, model A's accuracy declines to 37% while model B's rises to 93%. This is a huge change. So which model is better?

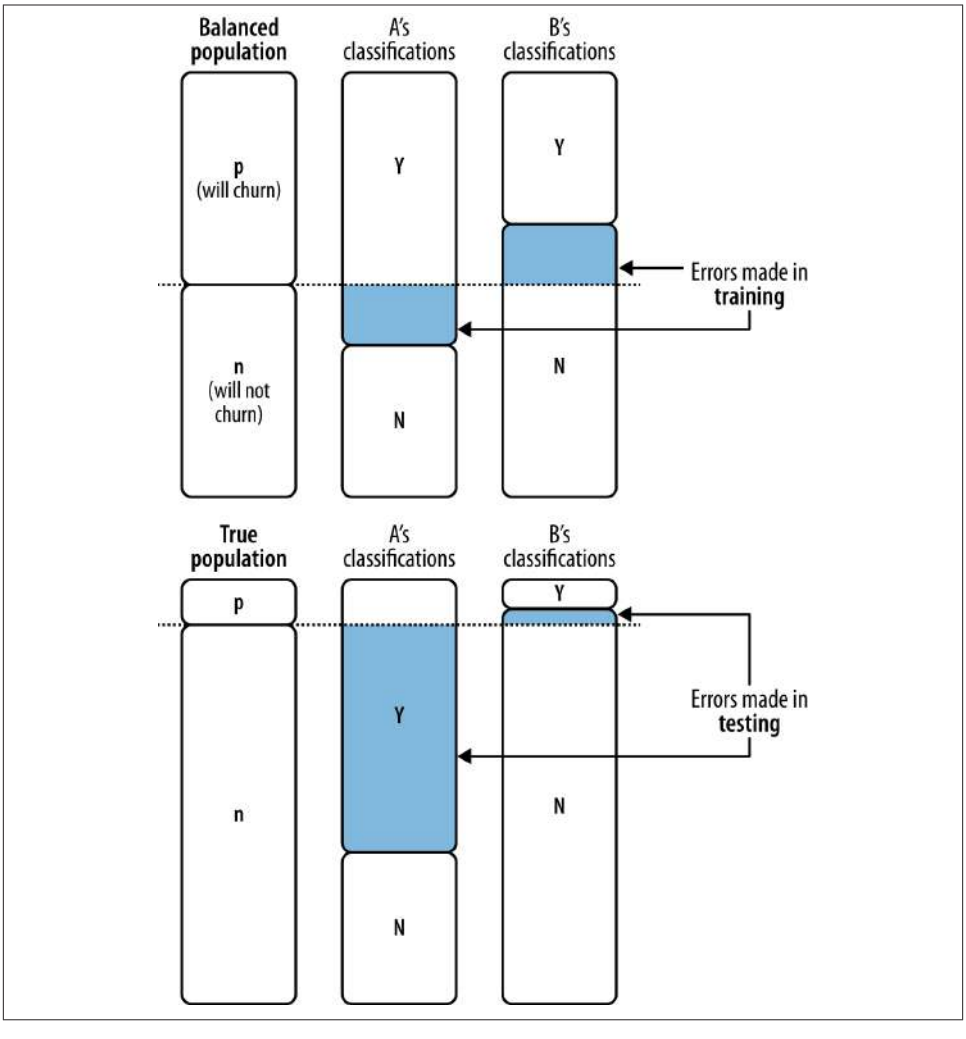

Figure 7-1. Two churn models, A and B, can make an equal number of errors on a balanced population used for training (top) but a very different number of errors when tested against the true population (bottom).

My model (B) now appears to be better than A because B seems to have greater performance on the population we care about—the 1:9 mix of customers we expect to see. But we still can't say for sure because of another problem with accuracy: we don't know how much we *care* about the different errors and correct decisions. This issue is the topic of the next section.

Problems with Unequal Costs and Benefits

Another problem with simple classification accuracy as a metric is that it makes no distinction between false positive and false negative errors. By counting them together, it makes the tacit assumption that both errors are equally important. With real-world domains this is rarely the case. These are typically very different kinds of errors with very different costs because the classifications have consequences of differing severity.

Consider a medical diagnosis domain where a patient is wrongly informed he has cancer when he does not. This is a false positive error. The result would likely be that the patient would be given further tests or a biopsy, which would eventually disconfirm the initial diagnosis of cancer. This mistake might be expensive, inconvenient, and stressful for the patient, but it would not be life threatening. Compare this with the opposite error: a patient who has cancer but she is wrongly told she does not. This is a false negative. This second type of error would mean a person with cancer would miss early detection, which could have far more serious consequences. These two errors are very different, should be counted separately, and should have different costs.

Returning to our cellular-churn example, consider the cost of giving a customer a retention incentive which still results in departure (a false positive error). Compare this with the cost of losing a customer because no incentive was offered (a false negative). Whatever costs you might decide for each, it is unlikely they would be equal; and the errors should be counted separately regardless.

Indeed, it is hard to imagine any domain in which a decision maker can safely be indifferent to whether she makes a false positive or a false negative error. Ideally, we should estimate the cost or benefit of each decision a classifier can make. Once aggregated, these will produce an *expected profit* (or *expected benefit* or *expected cost*) estimate for the classifier.

Generalizing Beyond Classification

We have been using classification modeling to illustrate many data science issues concretely. Most of these issues are applicable beyond classification.

The general principle we're developing here is that when we are applying data science to an actual application it is vital to return to the question: what is important in the application? What is the goal? Are we assessing the results of data mining appropriately given the actual goal?

As another example, let's apply this thinking to regression modeling rather than classification. Say our data science team has to build a movie recommendation model. It predicts how much a given customer will like a given movie, which we will use to help us provide personalized recommendations. Let's say each customer rates a movie by giving it one to five stars, and the recommendation model predicts how many stars a

user will give an unseen movie. One of our analysts describes each model by reporting the mean-squared-error (or root-mean-squared-error, or R^2, or whatever metric) for the model. We should ask: mean-squared-error of what? The analyst replies: the value of the target variable, which is the number of stars that a user would give as a rating for the movie. Why is the mean-squared-error on the predicted number of stars an appropriate metric for our recommendation problem? Is it meaningful? Is there a better metric? Hopefully, the analyst has thought this through carefully. It is surprising how often one finds that an analyst has not, and is simply reporting some measure he learned about in a class in school.

A Key Analytical Framework: Expected Value

We now are ready to discuss a very broadly useful conceptual tool to aid data analytic thinking: expected value. The expected value computation provides a framework that is extremely useful in organizing thinking about data-analytic problems. Specifically, it decomposes data-analytic thinking into (i) the structure of the problem, (ii) the elements of the analysis that can be extracted from the data, and (iii) the elements of the analysis that need to be acquired from other sources (e.g., business knowledge of subject matter experts).

In an expected value calculation the possible outcomes of a situation are enumerated. The expected value is then the weighted average of the values of the different possible outcomes, where the weight given to each value is its probability of occurrence. For example, if the outcomes represent different possible levels of profit, an expected profit calculation weights heavily the highly likely levels of profit, while unlikely levels of profit are given little weight. For this book, we will assume that we are considering repeated tasks (like targeting a large number of consumers, or diagnosing a large number of problems) and we are interested in maximizing expected profit.[1]

The expected value framework provides structure to an analyst's thinking (i) via the general form shown in Equation 7-1 .

Equation 7-1. The general form of an expected value calculation

$$EV = p(o_1) \cdot v(o_1) + p(o_2) \cdot v(o_2) + p(o_3) \cdot v(o_3) \ldots$$

Each o_i is a possible decision outcome; $p(o_i)$ is its probability and $v(o_i)$ is its value. The probabilities often can be estimated from the data (ii), but the business values often need to be acquired from other sources (iii). As we will see in Chapter 11, data-driven mod-

1. A course in decision theory would lead you into a thicket of interesting related issues.

eling may help estimate business values, but usually the values must come from external domain knowledge.

We now will illustrate the use of expected value as an analytical framework with two different data science scenarios. The two scenarios are often confused but it is vital to be able to distinguish them. To do so, recall the difference from Chapter 2 between the *mining* (or induction) of a model, and the model's *use*.

Using Expected Value to Frame Classifier Use

In use, we have many individual cases for which we would like to predict a class, which may then lead to an action. In targeted marketing, for example, we may want to assign each consumer a class of *likely responder* versus *not likely responder*, then we could target the likely responders. Unfortunately, for targeted marketing often the probability of response for any individual consumer is very low—maybe one or two percent—so no consumer may seem like a likely responder. If we choose a "common sense" threshold of 50% for deciding what a likely responder is, we would probably not target anyone. Many inexperienced data miners are surprised when the application of data mining models results in everybody being classified as *not likely responder* (or a similar negative class).

However, with the expected value framework we can see the crux of the problem. Let's walk through a targeted marketing scenario.[2] Consider that we have an offer for a product that, for simplicity, is only available via this offer. If the offer is not made to a consumer, the consumer will not buy the product. We have a model, mined from historical data, that gives an estimated probability of response $p_R(\mathbf{x})$ for any consumer whose feature vector description $\mathbf{x}$ is given as input. The model could be a classification tree or a logistic regression model or some other model we haven't talked about yet. Now we would like to decide whether to target a particular consumer described by feature vector $\mathbf{x}$.

Expected value provides a framework for carrying out the analysis. Specifically, let's calculate the expected benefit (or cost) of targeting consumer $\mathbf{x}$:

$$\text{Expected benefit of targeting} = p_R(\mathbf{x}) \cdot v_R + [1 - p_R(\mathbf{x})] \cdot v_{NR}$$

where v_R is the value we get from a response and v_{NR} is the value we get from no response. Since everyone either responds or does not, our estimate of the probability of not responding is just $(1 - p_R(\mathbf{x}))$. As mentioned, the probabilities came from the historical

2. We use targeted marketing here, rather than the churn example, because the expected value framework actually reveals an important complexity to the churn example that we're not ready to deal with. We will return to that later in Chapter 11 when we're ready to deal with it.

data, as summarized in our predictive model. The benefits v_R and v_{NR} need to be determined separately, as part of the Business Understanding step (recall Chapter 2). Since a customer can only purchase the product by responding to the offer (as discussed above), the expected benefit of not targeting her conveniently is zero.

To be concrete, let's say that a consumer buys the product for $200 and our product-related costs are $100. To target the consumer with the offer, we also incur a cost. Let's say that we mail some flashy marketing materials, and the overall cost including postage is $1, yielding a value (profit) of v_R = $99 if the consumer responds (buys the product). Now, what about v_{NR}, the value to us if the consumer does not respond? We still mailed the marketing materials, incurring a cost of $1 or equivalently a benefit of -$1.

Now we are ready to say precisely whether we want to target this consumer: do we expect to make a profit? Technically, is the expected value (profit) of targeting greater than zero? Mathematically, this is:

$$p_R(\mathbf{x}) \cdot \$99 - [1 - p_R(\mathbf{x})] \cdot \$1 > 0$$

A little rearranging of the equation gives us a decision rule: Target a given customer **x** only if:

$$p_R(\mathbf{x}) \cdot \$99 > [1 - p_R(\mathbf{x})] \cdot \$1$$

$$p_R(\mathbf{x}) > 0.01$$

With these example values, we should target the consumer as long as the estimated probability of responding is greater than 1%.

This shows how an expected value calculation can express how we will *use* the model. Making this explicit helps to organize problem formulation and analysis. We will return to this in Chapter 11. Now, let's move on to the other important application of the expected value framework, to organize our analysis of whether the model we have induced from the data is any good.

Using Expected Value to Frame Classifier Evaluation

At this point we want to shift our focus from individual decisions to collections of decisions. Specifically, we need to evaluate the set of decisions made by a model when applied to a set of examples. Such an evaluation is necessary in order to compare one model to another. For example, does our data-driven model perform better than the hand-crafted model suggested by the marketing group? Does a classification tree work better than a linear discriminant model for a particular problem? Do any of the models do substantially better than a baseline "model," such as randomly choosing consumers

to target? It is likely that each model will make some decisions better than the other model. What we care about is, *in aggregate*, how well does each model do: what is its *expected* value.

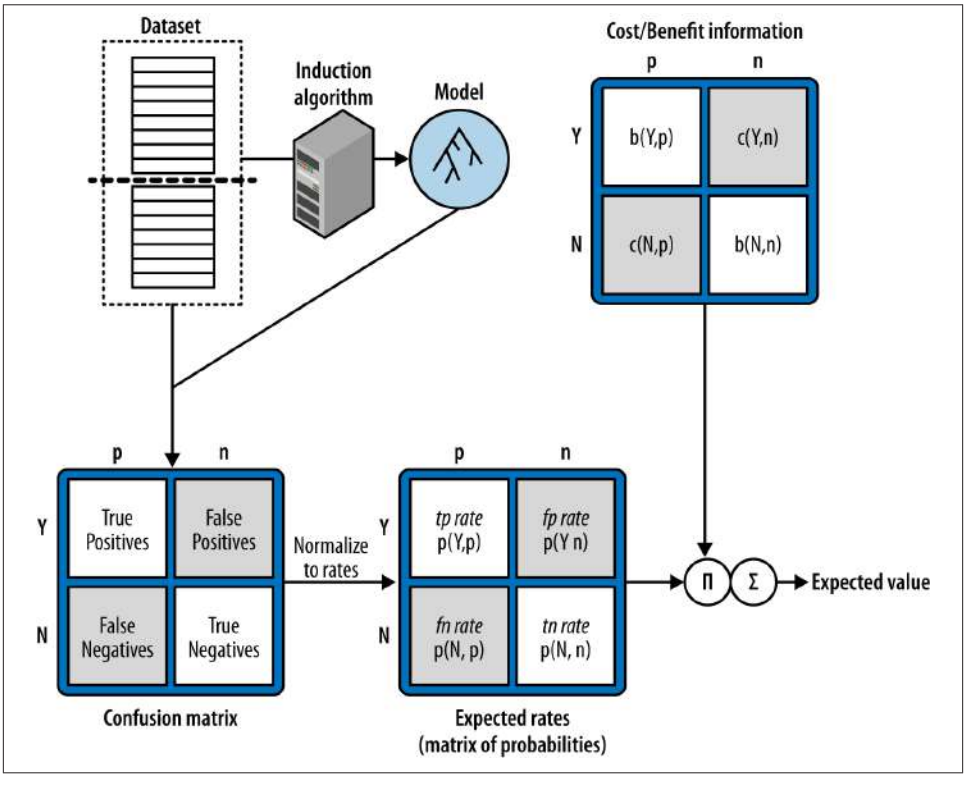

Figure 7-2. A diagram of the expected value calculation. The Π and Σ refer to the multiplication and summation in the expected value calculation.

We can use the expected value framework just described to determine the best decisions for each particular model, and then use the expected value in a different way to compare the models. If we are to calculate the expected profit for a model in aggregate, each o_i in Equation 7-1 corresponds to one of the possible combinations of the class we predict, and the actual class. We want to aggregate all the different possible cases: overall, when we decide to target consumers, what is the probability that they respond? What is the probability that they do not? What about when we do not target consumers, would they have responded? Fortunately, as you may recall, we already have the counts necessary to calculate all these—in the confusion matrix. Each o_i corresponds to one cell of the confusion matrix. For example, what is the probability associated with the particular combination of a consumer being *predicted to churn* and *actually does not churn*? That

would be estimated by the number of test-set consumers who fell into the confusion matrix cell (**Y**,**n**), divided by the total number of test-set consumers.

Let's walk through an entire expected profit calculation at the aggregate (model) level, in the process computing these probabilities. Figure 7-2 shows a schematic diagram of the expected value calculation in the context of model induction and evaluation. At the top left of the diagram, a training portion of a dataset is taken as input by an induction algorithm, which produces the model that we will evaluate. That model is applied to a holdout (test) portion of the data, and the counts for the different cells of the confusion matrix are tallied. Let's consider a concrete example of a classifier confusion matrix in Table 7-4.

Table 7-4. A sample confusion matrix with counts.

	p	n
Y	56	7
N	5	42

Error rates

When calculating expected values for a business problem, the analyst is often faced with the question: where do these probabilities actually come from? When evaluating a model on testing data, the answer is straightforward: these probabilities (of errors and correct decisions) can be estimated from the tallies in the confusion matrix by computing the rates of the errors and correct decisions. Each cell of the confusion matrix contains a count of the number of decisions corresponding to the corresponding combination of (predicted, actual), which we will express as $count(h,a)$ (we use h for "hypothesized" since p is already being used). For the expected value calculation we reduce these counts to rates or estimated probabilities, $p(h,a)$. We do this by dividing each count by the total number of instances:

$$p(h, a) = count(h, a) / T$$

Here are the calculations of the rates for each of the raw statistics in the confusion matrix. These rates are estimates of the probabilities that we will use in the expected value computation of Equation 7-1.

$T = 110$

$p(\mathbf{Y},\mathbf{p}) = 56/110 = 0.51$ $p(\mathbf{Y},\mathbf{n}) = 7/110 = 0.06$

$p(\mathbf{N},\mathbf{p}) = 5/110 = 0.05$ $p(\mathbf{N},\mathbf{n}) = 42/110 = 0.38$

Costs and benefits

To compute expected profit (recall Equation 7-1), we also need the cost and benefit values that go with each decision pair. These will form the entries of a cost-benefit matrix

with the same dimensions (rows and columns) as the confusion matrix. However, the cost-benefit matrix specifies, for each (predicted,actual) pair, the cost or benefit of making such a decision (see Figure 7-3). Correct classifications (true positives and true negatives) correspond to the benefits $b(\mathbf{Y},\mathbf{p})$ and $b(\mathbf{N}, \mathbf{n})$, respectively. Incorrect classifications (false positives and false negatives) correspond to the "benefit" $b(\mathbf{Y},\mathbf{n})$ and $b(\mathbf{N}, \mathbf{p})$, respectively, which may well actually be a cost (a negative benefit), and often are explicitly referred to as costs $c(\mathbf{Y}, \mathbf{n})$ and $c(\mathbf{N}, \mathbf{p})$.

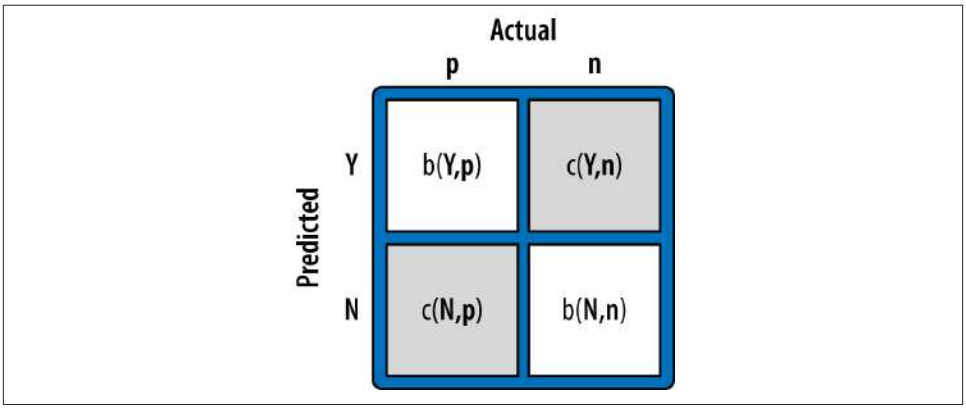

Figure 7-3. A cost-benefit matrix.

While the probabilities can be estimated from data, *the costs and benefits often cannot*. They generally depend on external information provided via analysis of the consequences of decisions in the context of the specific business problem. Indeed, specifying the costs and benefits may take a great deal of time and thought. In many cases they cannot be specified exactly but only as approximate ranges. Chapter 8 will return to address what we might do when these values are not known exactly. For example, in our churn problem, how much is it really worth us to retain a customer? The value depends on future cell phone usage and probably varies a great deal between customers. It may be that data on customers' prior usage can be helpful in this estimation. In many cases, average estimated costs and benefits are used rather than individual-specific costs and benefits, for simplicity of problem formulation and calculation. Therefore, we will ignore customer-specific cost/benefit calculations for the rest of our example, but will return to it in Chapter 11.

So, let's return to our targeted marketing example. What are the costs and benefits? We will express all values as *benefits*, with costs being negative benefits, so the function we're specifying is b(predicted, actual). For simplicity, all numbers will be expressed as dollars.

- A *false positive* occurs when we classify a consumer as a likely responder and therefore target her, but she does not respond. We've said that the cost of preparing and

mailing the marketing materials is a fixed cost of $1 per consumer. The benefit in this case is negative: $b(\mathbf{Y}, \mathbf{n}) = -1$.

- A *false negative* is a consumer who was predicted not to be a likely responder (so was not offered the product), but would have bought it if offered. In this case, no money was spent and nothing was gained, so $b(\mathbf{N}, \mathbf{p}) = 0$.
- A *true positive* is a consumer who is offered the product and buys it. The benefit in this case is the profit from the revenue ($200) minus the product-related costs ($100) and the mailing costs ($1), so $b(\mathbf{Y}, \mathbf{p}) = 99$.
- A *true negative* is a consumer who was not offered a deal and who would not have bought it even if it had been offered. The benefit in this case is zero (no profit but no cost), so $b(\mathbf{N}, \mathbf{n}) = 0$.

These cost-benefit estimations can be summarized in a 2 × 2 cost-benefit matrix, as in Figure 7-4. Note that the rows and columns are the same as for our confusion matrix, which is exactly what we'll need to compute the overall expected value for the classification model.

		Actual	
		p	**n**
Predicted	**Y**	99	-1
	N	0	0

Figure 7-4. A cost-benefit matrix for the targeted marketing example.

Given a matrix of costs and benefits, these are multiplied cell-wise against the matrix of probabilities, then summed into a final value representing the total expected profit. The result is:

$$\begin{aligned} \textit{Expected profit} = {} & p(\mathbf{Y}, \mathbf{p}) \cdot b(\mathbf{Y}, \mathbf{p}) + p(\mathbf{N}, \mathbf{p}) \cdot b(\mathbf{N}, \mathbf{p}) + \\ & p(\mathbf{N}, \mathbf{n}) \cdot b(\mathbf{N}, \mathbf{n}) + p(\mathbf{Y}, \mathbf{n}) \cdot b(\mathbf{Y}, \mathbf{n}) \end{aligned}$$

Using this equation, we can now compute and compare the expected profits for various models and other targeting strategies. All we need is to be able to compute the confusion matrices over a set of test instances, and to generate the cost-benefit matrix.

This equation is sufficient for comparing classifiers, but let's continue along this path a little further, because an alternative calculation of this equation is often used in practice. This alternative view is closely related to some techniques used to visualize classifier performance (see Chapter 8). Furthermore, by examining the alternative formulation we can see exactly how to deal with the model comparison problem we introduced at

the beginning of the chapter—where one analyst had reported performance statistics over a representative (but unbalanced) population, and another had used a class-balanced population.

A common way of expressing expected profit is to factor out the probabilities of seeing each class, often referred to as the *class priors*. The class priors, $p(\mathbf{p})$ and $p(\mathbf{n})$, specify the likelihood of seeing positive and negative instances, respectively. Factoring these out allows us to separate the influence of class imbalance from the fundamental predictive power of the model, as we will discuss in more detail in Chapter 8.

A rule of basic probability is:

$$p(x, y) = p(y) \cdot p(x \mid y)$$

This says that the probability of two different events both occurring is equal to the probability of one of them occurring times the probability of the other occurring if we know that the first occurs. Using this rule we can re-express our expected profit as:

$$\begin{aligned}\text{Expected profit} = \; & p(\mathbf{Y} \mid \mathbf{p}) \cdot p(\mathbf{p}) \cdot b(\mathbf{Y}, \mathbf{p}) + p(\mathbf{N} \mid \mathbf{p}) \cdot p(\mathbf{p}) \cdot b(\mathbf{N}, \mathbf{p}) + \\ & p(\mathbf{N} \mid \mathbf{n}) \cdot p(\mathbf{n}) \cdot b(\mathbf{N}, \mathbf{n}) + p(\mathbf{Y} \mid \mathbf{n}) \cdot p(\mathbf{n}) \cdot b(\mathbf{Y}, \mathbf{n})\end{aligned}$$

Factoring out the class priors $p(\mathbf{y})$ and $p(\mathbf{n})$, we get the final equation:

Equation 7-2. Expected profit equation with priors $p(\mathbf{p})$ and $p(\mathbf{n})$ factored.

$$\begin{aligned}\text{Expected profit} = \; & p(\mathbf{p}) \cdot [p(\mathbf{Y} \mid \mathbf{p}) \cdot b(\mathbf{Y}, \mathbf{p}) + p(\mathbf{N} \mid \mathbf{p}) \cdot c(\mathbf{N}, \mathbf{p})] + \\ & p(\mathbf{n}) \cdot [p(\mathbf{N} \mid \mathbf{n}) \cdot b(\mathbf{N}, \mathbf{n}) + p(\mathbf{Y} \mid \mathbf{n}) \cdot c(\mathbf{Y}, \mathbf{n})]\end{aligned}$$

From this mess, notice that we now have one component (the first one) corresponding to the expected profit from the positive examples, and another (the second one) corresponding to the expected profit from the negative examples. Each of these is weighted by the probability that we see that sort of example. So, if positive examples are very rare, their contribution to the overall expected profit will be correspondingly small.[3] In this alternative formulation, the quantities $p(\mathbf{Y}|\mathbf{p})$, $p(\mathbf{Y}|\mathbf{n})$, etc. correspond directly to the true positive rate, the false positive rate, etc., that also can be calculated directly from the confusion matrix (see "Sidebar: Other Evaluation Metrics" on page 203).

3. This could be extended to any number of classes, though to keep things simple(r) we'll use two.

Here again is our sample confusion matrix in Table 7-5.

Table 7-5. Our sample confusion matrix (raw counts)

	p	n
Y	56	7
N	5	42

Table 7-6 shows the class priors and various error rates we need.

Table 7-6. The class priors and the rates of true positives, false positives, and so on.

T = 110	
P = 61	N = 49
$p(\mathbf{p}) = 0.55$	$p(\mathbf{n}) = 0.45$
tp rate = 56/61 = 0.92	*fp rate* = 7/49 = 0.14
fn rate = 5/61 = 0.08	*tn rate* = 42/49 = 0.86

Returning to the targeted marketing example, what is the expected profit of the model learned? We can calculate it using Equation 7-2:

$$\begin{aligned}
\text{expected profit} &= p(\mathbf{p}) \cdot [p(\mathbf{Y} \mid \mathbf{p}) \cdot b(\mathbf{Y}, \mathbf{p}) + p(\mathbf{N} \mid \mathbf{p}) \cdot c(\mathbf{N}, \mathbf{p})] + \\
&\quad p(\mathbf{n}) \cdot [p(\mathbf{N} \mid \mathbf{n}) \cdot b(\mathbf{N}, \mathbf{n}) + p(\mathbf{Y} \mid \mathbf{p}) \cdot c(\mathbf{Y}, \mathbf{n})] \\
&= 0.55 \cdot [0.92 \cdot b(\mathbf{Y}, \mathbf{p}) + 0.08 \cdot b(\mathbf{N}, \mathbf{p})] + \\
&\quad 0.45 \cdot [0.86 \cdot b(\mathbf{N}, \mathbf{n}) + 0.14 \cdot p(\mathbf{Y}, \mathbf{n})] \\
&= 0.55 \cdot [0.92 \cdot 99 + 0.08 \cdot 0] + \\
&\quad 0.45 \cdot [0.86 \cdot 0 + 0.14 \cdot \ -1] \\
&= 50.1 - 0.063 \\
&\approx \$\mathbf{50.04}
\end{aligned}$$

This expected value means that if we apply this model to a population of prospective customers and mail offers to those it classifies as positive, we can expect to make an average of about $50 profit per consumer.

We now can see one way to deal with our motivating example from the beginning of the chapter. Instead of computing accuracies for the competing models, we would compute expected values. Furthermore, using this alternative formulation, we can compare the two models even though one analyst tested using a representative distribution and the other tested using a class-balanced distribution. In each calculation, we simply can replace the priors. Using a balanced distribution corresponds to priors of $p(\mathbf{p}) = 0.5$ and $p(\mathbf{n}) = 0.5$. The mathematically savvy reader is encouraged to convince herself that the other factors in the equation will not change if the test-set priors change.

To close this section on estimated profit, we emphasize two pitfalls that are common when formulating cost-benefit matrices:

- It is important to make sure the signs of quantities in the cost-benefit matrix are consistent. In this book we take benefits to be positive and costs to be negative. In many data mining studies, the focus is on minimizing cost rather than maximizing profit, so the signs are reversed. Mathematically, there is no difference. However, it is important to pick one view and be consistent.
- An easy mistake in formulating cost-benefit matrices is to "double count" by putting a benefit in one cell and a negative cost *for the same thing* in another cell (or vice versa). A useful practical test is to compute the *benefit improvement* for changing the decision on an example test instance.

  For example, say you've built a model to predict which accounts have been defrauded. You've determined that a fraud case costs $1,000 on average. If you decide that the benefit of catching fraud is therefore +$1,000/case on average, *and* the cost of missing fraud is -$1,000/case, then what would be the *improvement in benefit* for catching a case of fraud? You would calculate:

  b(**Y**,**p**) - b(**N**,**p**) = $1000 - (-$1000) = $2000

  But intuitively you know that this improvement should only be about $1,000, so this error indicates double counting. The solution is to specify either that the benefit of catching fraud is $1,000 *or* that the cost of missing fraud is -$1,000, but not both. One should be zero.

Sidebar: Other Evaluation Metrics

There are many evaluation metrics you will likely encounter in data science. All of them are fundamentally summaries of the confusion matrix. Referring to the counts in the confusion matrix, let's abbreviate the number of true positives, false positives, true negatives, and false negatives by *TP*, *FP*, *TN*, and *FN*, respectively. We can describe various evaluation metrics using these counts. *True positive rate* and *False negative rate* refer to the frequency of being correct and incorrect, respectively, when the instance is actually positive: $TP/(TP + FN)$ and $FN/(TP + FN)$. The *True negative rate* and *False positive rate* are analogous for the instances that are actually negative. These are often taken as estimates of the probability of predicting **Y** when the instance is actually **p**, that is $p(\mathbf{Y}|\mathbf{p})$, etc. We will continue to explore these measures in Chapter 8.

The metrics *Precision* and *Recall* are often used, especially in text classification and information retrieval. Recall is the same as true positive rate, while precision is $TP/(TP$

+ *FP*), which is the accuracy over the cases predicted to be positive. The *F-measure* is the harmonic mean of precision and recall at a given point, and is:

$$\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Practitioners in many fields such as statistics, pattern recognition, and epidemiology speak of the sensitivity and specificity of a classifier:

$$\text{Sensitivity} = TN / (TN + FP) = \text{True negative rate} = 1 - \text{False positive rate}$$
$$\text{Specificity} = TP / (TP + FN) = \text{True positive rate}$$

You may also hear about the *positive predictive value*, which is the same as precision.

Accuracy, as mentioned before, is simply the count of correct decisions divided by the total number of decisions, or:

$$Accuracy = \frac{TP + TN}{P + N}$$

Swets (1996) lists many other evaluation metrics and their relationships to the confusion matrix.

Evaluation, Baseline Performance, and Implications for Investments in Data

Up to this point we have talked about model evaluation in isolation. In some cases just demonstrating that a model generates *some* (nonzero) profit, or a positive return on investment, will be informative by itself. Nevertheless, another fundamental notion in data science is: *it is important to consider carefully what would be a reasonable baseline against which to compare model performance.* This is important for the data science team in order to understand whether they indeed are improving performance, and is equally important for demonstrating to stakeholders that mining the data has added value. So, what is an appropriate baseline for comparison?

The answer of course depends on the actual application, and coming up with suitable baselines is one task for the business understanding phase of the data mining process. However, there are some general principles that can be very helpful.

For classification models it is easy to simulate a completely random model and measure its performance. The visualization frameworks we will discuss in Chapter 8 have natural baselines showing what random classification should achieve. This is useful for very

difficult problems or initial explorations. Comparison against a random model establishes that there is some information to be extracted from the data.

However, beating a random model may be easy (or may seem easy), so demonstrating superiority to it may not be very interesting or informative. A data scientist will often need to implement an alternative model, usually one that is simple but not simplistic, in order to justify continuing the data mining effort.

In Nate Silver's book on prediction, *The Signal and the Noise* (2012), he mentions the baseline issue with respect to weather forecasting:

> There are two basic tests that any weather forecast must pass to demonstrate its merit: It must do better than what meteorologists call persistence: the assumption that the weather will be the same tomorrow (and the next day) as it was today. It must also beat climatology, the long-term historical average of conditions on a particular date in a particular area.

In other words, weather forecasters have two simple—but not simplistic—baseline models that they compare against. One (persistence) predicts that the weather tomorrow is going to be whatever it was today. The other (climatology) predicts whatever the average historical weather has been on this day from prior years. Each model performs considerably better than random guessing, and both are so easy to compute that they make natural baselines of comparison. Any new, more complex model must beat these.

What are some general guidelines for good baselines? For classification tasks, one good baseline is the *majority classifier*, a naive classifier that always chooses the majority class of the training dataset (see Note: Base rate in "Holdout Data and Fitting Graphs" on page 113). This may seem like advice so obvious it can be passed over quickly, but it is worth spending an extra moment here. There are many cases where smart, analytical people have been tripped up in skipping over this basic comparison. For example, an analyst may see a classification accuracy of 94% from her classifier and conclude that it is doing fairly well—when in fact only 6% of the instances are positive. So, the simple majority prediction classifier also would have an accuracy of 94%. Indeed, many beginning data science students are surprised to find that the models they build from the data simply predict everything to be of the majority class. Note that this may make sense if the modeling procedure is set up to build maximum accuracy models—it may be hard to beat 94% accuracy. The answer, of course, is to apply the central idea of this chapter: consider carefully what is desired from the data mining results. Maximizing simple prediction accuracy is usually not an appropriate goal. If that's what our algorithm is doing, we're using the wrong algorithm. For regression problems we have a directly analogous baseline: predict the average value over the population (usually the mean or median).

In some applications there are multiple simple averages that one may want to combine. For example, when evaluating recommender systems that internally predict how many "stars" a particular customer would give to a particular movie, we have the average number of stars a movie gets across the population (how well liked it is) and the average

number of stars a particular customer gives to movies (what that customer's overall bias is). A simple prediction based on these two may do substantially better than using one or the other in isolation.

Moving beyond these simple baseline models, a slightly more complex alternative is a model that only considers a very small amount of feature information. For example, recall from Chapter 3 our very first example of a data mining procedure: finding informative variables. If we find the one variable that correlates best with the target, we can build a classification or regression model that uses just that variable, which gives another view of baseline performance: how well does a simple "conditional" model perform? "Conditional" here means that it predicts differently based on, or conditioned on, the value of the feature(s). The overall population average is therefore sometimes called the "unconditional" average.

One example of mining such single-feature predictive models from data is to use tree induction to build a "**decision stump**" — a decision tree with only one internal node, the root node. A tree limited to one internal node simply means that the tree induction selects the single most informative feature to make a decision. In a well-known paper in machine learning, Robert Holte (1993) showed that decision stumps often produce quite good baseline performance on many of the test datasets used in machine learning research. A decision stump is an example of the strategy of choosing the single most informative piece of information available (recall Chapter 3) and basing all decisions on it. In some cases most of the leverage may be coming from a single feature, and this method assesses whether and to what extent that is the case.

This idea can be extended to data *sources*, and relates to our fundamental principle from Chapter 1 that we should regard data as an asset to be invested in. If you are considering building models that integrate data from various sources, you should compare the result to models built from the individual sources. Often there are substantial costs to acquiring new sources of data. In some cases these are actual monetary costs; in other cases they involve commitments of personnel time for managing relationships with data providers and monitoring data feeds. To be thorough, for each data source the data science team should compare a model that uses the source to one that does not. Such comparisons help to justify the cost of each source by quantifying its value. If the contribution is negligible, the team may be able to reduce costs by eliminating it.

Beyond comparing simple models (and reduced-data models), it is often useful to implement simple, inexpensive models based on domain knowledge or "received wisdom" and evaluate their performance. For example, in one fraud detection application it was commonly believed that most defrauded accounts would experience a sudden increase in usage, and so checking accounts for sudden jumps in volume was sufficient for catching a large proportion of fraud. Implementing this idea was straightforward and it provided a useful baseline for demonstrating the benefit of data mining. (This essentially was a single-feature predictive model.) Similarly, an IBM team that used data

mining to direct sales efforts chose to implement a simple sales model that prioritized existing customers by the size of previous revenue and other companies by annual sales.[4] They were able to demonstrate that their data mining added significant value beyond this simpler strategy. Whatever the data mining group chooses as a baseline for comparison, it should be something the stakeholders find informative, and hopefully persuasive.

Summary

A vital part of data science is arranging for proper evaluation of models. This can be surprisingly difficult to get right and will often require multiple iterations. It is tempting to use simple measures, such as classification accuracy, since these are simple to calculate, are used in many research papers, and may be what one learned in school. However, in real-world domains simplistic measures rarely capture what is actually important for the problem at hand, and often mislead. Instead, the data scientist should give careful thought to how the model will be used in practice and devise an appropriate metric.

The *expected value* calculation is a good framework for organizing this thinking. It will help to frame the evaluation, and in the event that the final deployed model produces unacceptable results, it will help identify what is wrong.

The characteristics of the data should be taken into account carefully when evaluating data science results. For example, real classification problems often present data with very unbalanced class distributions (that is, the classes will not occur with equal prevalence). Adjusting class proportions may be useful (or even necessary) to learn a model from the data; however, evaluation should use the original, realistic population so that the results reflect what will actually be achieved.

To calculate the overall expected value of a model, the costs and benefits of decisions must be specified. If this is possible, the data scientist can calculate an expected cost per instance for each model and choose whichever model produces the lowest expected cost or greatest profit.

It also is vital to consider what one should compare a data-driven model against, to judge whether it performs well or better. The answer to this question is intimately tied to the business understanding, but there are a variety of general best practices that data science teams should follow.

We illustrated the ideas of this chapter with applications of the concepts presented in the chapters that preceded. The concepts are more general of course, and relate to our very first fundamental concept: data should be considered an asset and we need to

4. They refer to these as Willy Sutton models, after the famed bank robber who robbed banks because "that's where the money is."

consider how to invest. We illustrated this point by discussing briefly that one not only can compare different models and different baselines, but also compare results with different data sources. Different data sources may have different associated costs, and careful evaluation may show which can be chosen to maximize the return on investment.

As a final summary point, this chapter has discussed single quantitative numbers as summary estimates of model performance. They can answer questions like "How much profit can I expect to make?" and "Should I use model A or model B?" Such answers are useful but provide only "single-point values" that hold under a specific set of assumptions. It is often revealing to visualize model behavior under a broad range of conditions. The next chapter discusses graphical views of model behavior that can do just this.

CHAPTER 8

Visualizing Model Performance

Fundamental concepts: *Visualization of model performance under various kinds of uncertainty; Further consideration of what is desired from data mining results.*

Exemplary techniques: *Profit curves; Cumulative response curves; Lift curves; ROC curves.*

The previous chapter introduced basic issues of model evaluation and explored the question of what makes for a good model. We developed detailed calculations based on the expected value framework. That chapter was much more mathematical than previous ones, and if this is your first introduction to that material you may have felt overwhemed by the equations. Though they form the basis for what comes next, by themselves they may not be very intuitive. In this chapter we will take a different view to increase our understanding of what they are revealing.

The expected profit calculation of Equation 7-2 takes a specific set of conditions and generates a single number, representing the expected profit in that scenario. Stakeholders outside of the data science team may have little patience for details, and will often want a higher-level, more intuitive view of model performance. Even data scientists who are comfortable with equations and dry calculations often find such single estimates to be impoverished and uninformative, because they rely on very stringent assumptions (e.g., of precise knowledge of the costs and benefits, or that the models' estimates of probabilities are accurate). In short, it is often useful to present *visualizations* rather than just calculations, and this chapter presents some useful techniques.

Ranking Instead of Classifying

"A Key Analytical Framework: Expected Value" on page 194 discussed how the score assigned by a model can be used to compute a decision for each individual case based on its expected value. A different strategy for making decisions is to *rank* a set of cases by these scores, and then take actions on the cases at the top of the ranked list. Instead

of deciding each case separately, we may decide to take the top *n* cases (or, equivalently, all cases that score above a given threshold). There are several practical reasons for doing this.

It may be that the model gives a score that ranks cases by their likelihood of belonging to the class of interest, but which is not a true probability (recall our discussion in Chapter 4 of the distance from the separating boundary as a classifier score). More importantly, for some reason we may not be able to obtain accurate probability estimates from the classifier. This happens, for example, in targeted marketing applications when one cannot get a sufficiently representative training sample. The classifier scores may still be very useful for deciding which prospects are better than others, even if a 1% probability estimate doesn't exactly correspond to a 1% probability of responding.

A common situation is where you have a *budget* for actions, such as a fixed marketing budget for a campaign, and so you want to target the most promising candidates. If one is going to target the highest expected value cases using costs and benefits that are constant for each class, then ranking cases by likelihood of the target class is sufficient. There is no great need to care about the precise probability estimates. The only caveat is that the budget be small enough so that the actions do not go into negative expected-value territory. For now, we will leave that as a business understanding task.

It also may be that costs and benefits cannot be specified precisely, but nevertheless we would like to take actions (and are happy to do so on the highest likelihood cases). We'll return to this situation in the next section.

If *individual* cases have different costs and benefits, then our expected value discussion in "A Key Analytical Framework: Expected Value" on page 194 should make it clear that simply ranking by likelihood will not be sufficient.

When working with a classifier that gives scores to instances, in some situations the classifier decisions should be very conservative, corresponding to the fact that the classifier should have high certainty before taking the positive action. This corresponds to using a high threshold on the output score. Conversely, in some situations the classifier can be more permissive, which corresponds to lowering the threshold.[1]

This introduces a complication for which we need to extend our analytical framework for assessing and comparing models. "The Confusion Matrix" on page 189 stated that a classifier produces a confusion matrix. With a ranking classifier, a classifier *plus a*

1. Indeed, in some applications, scores from the same model may be used in several places with different thresholds to make different decisions. For example, a model may be used first in a decision to grant or deny credit. The same model may be used later in setting a new customer's credit line.

threshold produces a single confusion matrix. Whenever the threshold changes, the confusion matrix may change as well because the numbers of true positives and false positives change.

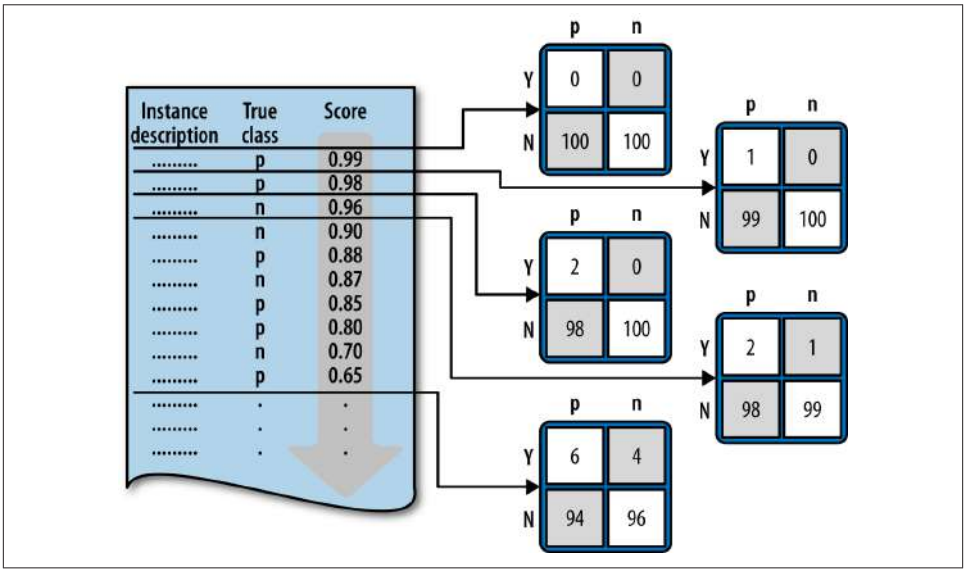

Figure 8-1. Thresholding a list of instances sorted by scores. Here, a set of test instances is scored by a model and sorted decreasing by these scores. We then apply a series of thresholds (represented by each horizontal line) to classify all instances above it as positive and those below it as negative. Each threshold results in a specific confusion matrix.

Figure 8-1 illustrates this basic idea. As the threshold is lowered, instances move up from the **N** row into the **Y** row of the confusion matrix: an instance that was considered a negative is now classified as positive, so the counts change. Which counts change depends on the example's true class. If the instance was a positive (in the "**p**" column) it moves up and becomes a true positive (**Y,p**). If it was a negative (**n**), it becomes a false positive (**Y,n**). Technically, each different threshold produces a different classifier, represented by its own confusion matrix.

This leaves us with two questions: how do we compare different *rankings*? And, how do we choose a proper *threshold*? If we have accurate probability estimates and a well-specified cost-benefit matrix, then we already answered the second question in our discussion of expected value: we determine the threshold where our expected profit is above a desired level (usually zero). Let's explore and extend this idea.

Profit Curves

From "A Key Analytical Framework: Expected Value" on page 194, we know how to compute expected profit, and we've just introduced the idea of using a model to rank instances. We can combine these ideas to construct various performance visualizations in the form of curves. Each curve is based on the idea of examining the effect of thresholding the value of a classifier at successive points, implicitly dividing the list of instances into many successive sets of predicted positive and negative instances. As we move the threshold "down" the ranking, we get additional instances predicted as being positive rather than negative. Each threshold, i.e., each set of predicted positives and negatives, will have a corresponding confusion matrix. The previous chapter showed that once we have a confusion matrix, along with knowledge of the cost and benefits of decisions, we can generate an expected value corresponding to that confusion matrix.

More specifically, with a ranking classifier, we can produce a list of instances and their predicted scores, ranked by decreasing score, and then measure the expected profit that would result from choosing each successive cut-point in the list. Conceptually, this amounts to ranking the list of instances by score from highest to lowest and sweeping down through it, recording the expected profit after each instance. At each cut-point we record the percentage of the list predicted as positive and the corresponding estimated profit. Graphing these values gives us a *profit curve*. Three profit curves are shown in Figure 8-2.

This graph is based on a test set of 1,000 consumers—say, a small random population of people to whom you test-marketed earlier. (When interpreting results, we normally will talk about percentages of consumers, so as to generalize to the population as a whole.) For each curve, the consumers are ordered from highest to lowest probability of accepting an offer based on some model. For this example, let's assume our profit margin is small: each offer costs $5 to make and market, and each accepted offer earns $9, for a profit of $4. The cost matrix is thus:

	p	n
Y	$4	-$5
N	$0	$0

The curves show that profit can go negative—not always, but sometimes they will, depending on the costs and the class ratio. In particular, this will happen when the profit margin is thin and the number of responders is small, because the curves show you "going into the red" by working too far down the list and making offers to too many people who won't respond, thereby spending too much on the costs of the offers.[2]

2. For simplicity in the example we will ignore inventory and other realistic issues that would require a more complicated profit calculation.

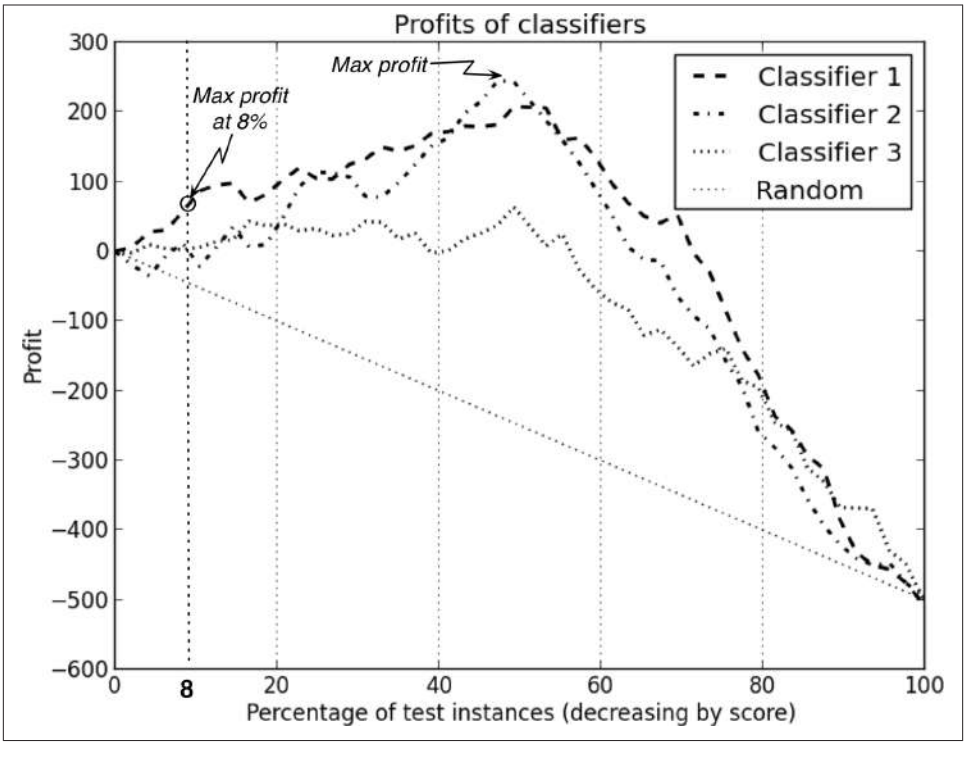

Figure 8-2. Profit curves of three classifiers. Each curve shows the expected cumulative profit for that classifier as progressively larger proportions of the consumer base are targeted.

Notice that all four curves begin and end at the same point. This should make sense because, at the left side, when no customers are targeted there are no expenses and zero profit; at the right side everyone is targeted, so every classifier performs the same. In between, we'll see some differences depending on how the classifiers order the customers. The random classifier performs worst because it has an even chance of choosing a responder or a nonresponder. Among the classifiers tested here, the one labeled Classifier 2 produces the maximum profit of $200 by targeting the top-ranked 50% of consumers. If your goal was simply to maximize profit and you had unlimited resources, you should choose Classifier 2, use it to score your population of customers, and target the top half (highest 50%) of customers on the list.

Now consider a slightly different but very common situation where you're constrained by a *budget*. You have a fixed amount of money available and you must plan how to spend it before you see any profit. This is common in situations such as marketing campaigns. As before, you still want to target the highest-ranked people, but now you

have a budgetary constraint[3] that may affect your strategy. Say you have 100,000 total customers and a budget of $40,000 for the marketing campaign. You want to use the modeling results (the profit curves in Figure 8-2) to figure out how best to spend your budget. What do you do in this case? Well, first you figure out how many offers you can afford to make. Each offer costs $5 so you can target at most $40,000/$5 = 8,000 customers. As before, you want to identify the customers most likely to respond, but each model ranks customers differently. Which model should you use for this campaign? 8,000 customers is 8% of your total customer base, so check the performance curves at x=8%. The best-performing model at this performance point is Classifier 1. You should use it to score the entire population, then send offers to the highest-ranked 8,000 customers.

In summary, from this scenario we see that adding a budgetary constraint causes not only a change in the operating point (targeting 8% of the population instead of 50%) but also a change in the choice of classifier to do the ranking.

ROC Graphs and Curves

Profit curves are appropriate when you know fairly certainly the conditions under which a classifier will be used. Specifically, there are two critical conditions underlying the profit calculation:

1. The *class priors*; that is, the proportion of positive and negative instances in the target population, also known as the *base rate* (usually referring to the proportion of positives). Recall that Equation 7-2 is sensitive to $p(\mathbf{p})$ and $p(\mathbf{n})$.
2. The *costs and benefits*. The expected profit is specifically sensitive to the relative levels of costs and benefits for the different cells of the cost-benefit matrix.

If both class priors and cost-benefit estimates are known and are expected to be stable, profit curves may be a good choice for visualizing model performance.

However, in many domains these conditions are uncertain or unstable. In fraud detection domains, for example, the amount of fraud changes from place to place, and from one month to the next (Leigh, 1995; Fawcett & Provost, 1997). The amount of fraud influences the priors. In the case of mobile phone churn management, marketing campaigns can have different budgets and offers may have different costs, which will change the expected costs.

3. Another common situation is to have a *workforce constraint*. It's the same idea: you have a fixed allocation of resources (money or personnel) available to address a problem and you want the most "bang for the buck." An example might be that you have a fixed workforce of fraud analysts, and you want to give them the top-ranked cases of potential fraud to process.

One approach to handling uncertain conditions is to generate many different expected profit calculations for each model. This may not be very satisfactory: the sets of models, sets of class priors, and sets of decision costs multiply in complexity. This often leaves the analyst with a large stack of profit graphs that are difficult to manage, difficult to understand the implications of, and difficult to explain to a stakeholder.

Another approach is to use a method that can accomodate uncertainty by showing the entire space of performance possibilities. One such method is the Receiver Operating Characteristics (ROC) graph (Swets, 1988; Swets, Dawes, & Monahan, 2000; Fawcett, 2006). A ROC graph is a two-dimensional plot of a classifier with false positive rate on the *x* axis against true positive rate on the *y* axis. As such, a ROC graph depicts relative trade-offs that a classifier makes between benefits (true positives) and costs (false positives). Figure 8-3 shows a ROC graph with five classifiers labeled **A** through **E**.

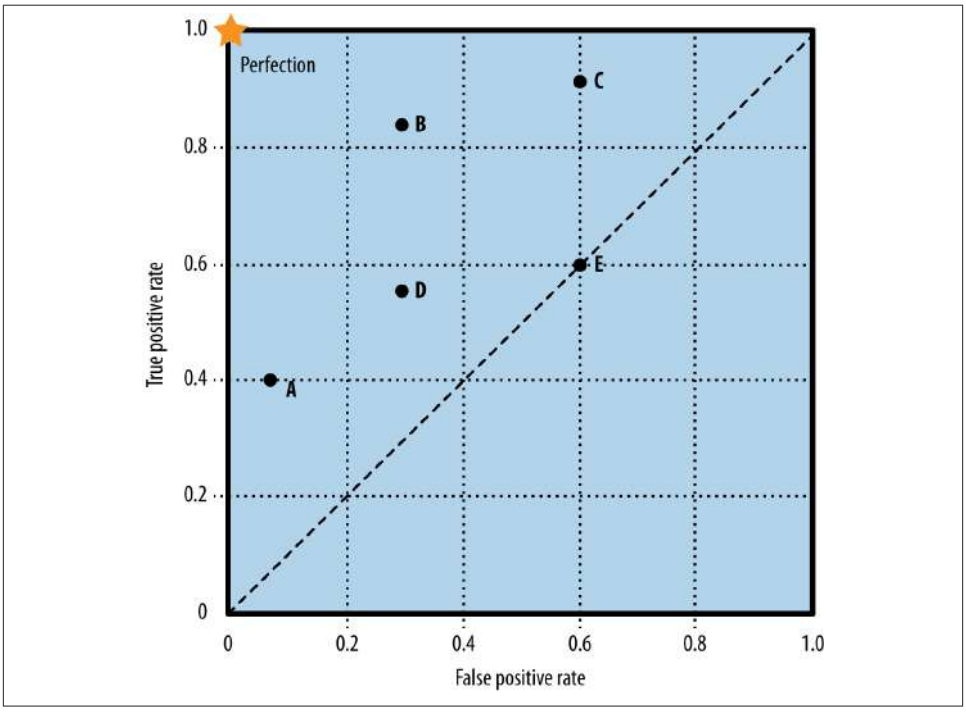

Figure 8-3. ROC space and five different classifiers (A-E) with their performance shown.

A *discrete* classifier is one that outputs only a class label (as opposed to a ranking). As already discussed, each such classifier produces a confusion matrix, which can be summarized by certain statistics regarding the numbers and rates of true positives, false positives, true negatives, and false negatives. Note that although the confusion matrix

contains four numbers, we really only need two of the rates: either the true positive rate or the false negative rate, and either the false positive rate or the true negative rate. Given one from either pair the other can be derived since they sum to one. It is conventional to use the true positive rate (*tp rate*) and the false positive rate (*fp rate*), and we will keep to that convention so the ROC graph will make sense. Each discrete classifier produces an (*fp rate*, *tp rate*) pair corresponding to a single point in ROC space. The classifiers in Figure 8-3 are all discrete classifiers. Importantly for what follows, the *tp rate* is computed using only the actual positive examples, and the *fp rate* is computed using only the actual negative examples.

Remembering exactly what statistics the *tp rate* and *fp rate* refer to can be confusing for someone who does not deal with such things on a daily basis. It can be easier to remember by using less formal but more intuitive names for the statistics: the *tp rate* is sometimes referred to as the *hit rate*—what percent of the actual positives does the classifier get right. The *fp rate* is sometimes referred to as the *false alarm rate*—what percent of the actual negative examples does the classifier get wrong (i.e., predict to be positive).

Several points in ROC space are important to note. The lower left point (0, 0) represents the strategy of never issuing a positive classification; such a classifier commits no false positive errors but also gains no true positives. The opposite strategy, of unconditionally issuing positive classifications, is represented by the upper right point (1, 1). The point (0, 1) represents perfect classification, represented by a star. The diagonal line connecting (0, 0) to (1, 1) represents the policy of guessing a class. For example, if a classifier randomly guesses the positive class half the time, it can be expected to get half the positives and half the negatives correct; this yields the point (0.5, 0.5) in ROC space. If it guesses the positive class 90% of the time, it can be expected to get 90% of the positives correct but its false positive rate will increase to 90% as well, yielding (0.9, 0.9) in ROC space. Thus a random classifier will produce a ROC point that moves back and forth on the diagonal based on the frequency with which it guesses the positive class. In order to get away from this diagonal into the upper triangular region, the classifier must exploit some information in the data. In Figure 8-3, **E**'s performance at (0.6, 0.6) is virtually random. **E** may be said to be guessing the positive class 60% of the time. Note that no classifier should be in the lower right triangle of a ROC graph. This represents performance that is worse than random guessing.

One point in ROC space is superior to another if it is to the northwest of the first (*tp rate* is higher and *fp rate* is no worse; *fp rate* is lower and *tp rate* is no worse, or both are better). Classifiers appearing on the lefthand side of a ROC graph, near the *x* axis, may be thought of as "conservative": they raise alarms (make positive classifications) only with strong evidence so they make few false positive errors, but they often have low true

positive rates as well. Classifiers on the upper righthand side of a ROC graph may be thought of as "permissive": they make positive classifications with weak evidence so they classify nearly all positives correctly, but they often have high false positive rates. In Figure 8-3, **A** is more conservative than **B**, which in turn is more conservative than **C**. Many real-world domains are dominated by large numbers of negative instances (see the discussion in "Sidebar: Bad Positives and Harmless Negatives" on page 188), so performance in the far left-hand side of the ROC graph is often more interesting than elsewhere. If there are very many negative examples, even a moderate false alarm *rate* can be unmanageable. A ranking model produces a set of points (a curve) in ROC space. As discussed previously, a ranking model can be used with a threshold to produce a discrete (binary) classifier: if the classifier output is above the threshold, the classifier produces a **Y**, else an **N**. Each threshold value produces a different point in ROC space, as shown in Figure 8-4.

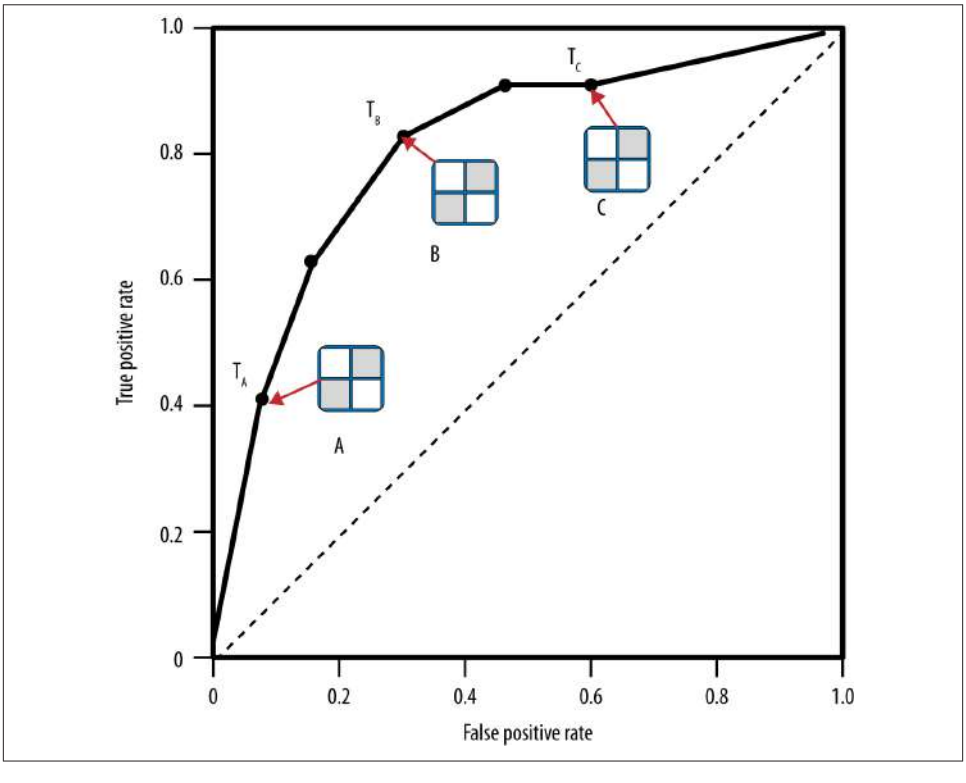

Figure 8-4. Each different point in ROC space corresponds to a specific confusion matrix.

Conceptually, we may imagine sorting the instances by score and varying a threshold from $-\infty$ to $+\infty$ while tracing a curve through ROC space, as shown in Figure 8-5.

Whenever we pass a positive instance, we take a step upward (increasing true positives); whenever we pass a negative instance, we take a step rightward (increasing false positives). Thus the "curve" is actually a step function for a single test set, but with enough instances it appears smooth.[4]

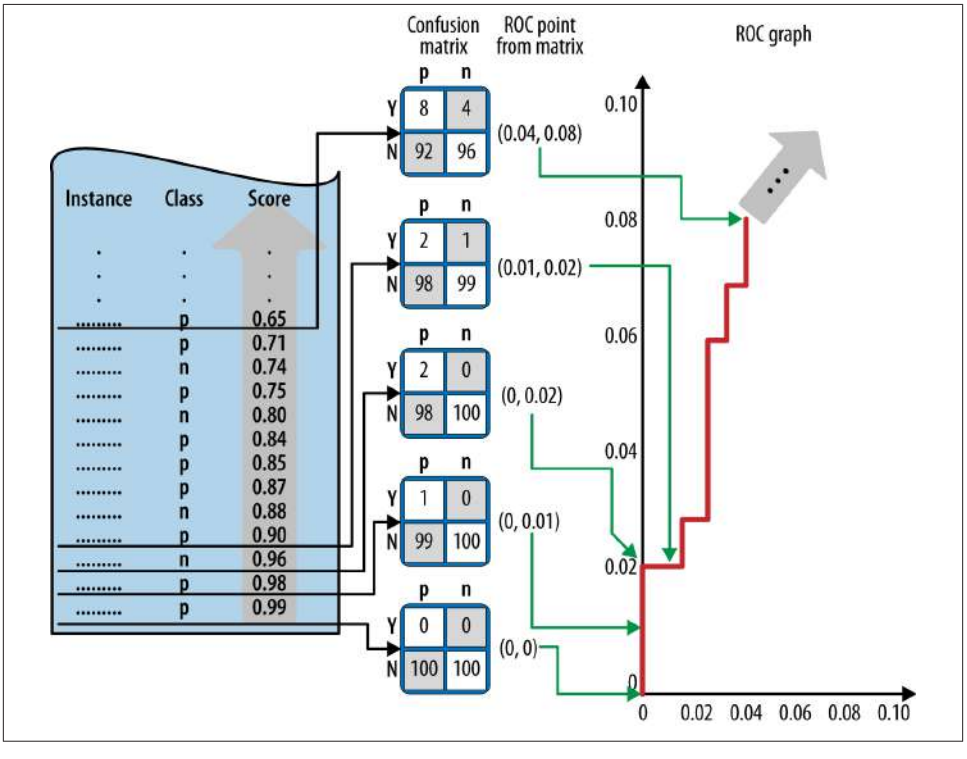

*Figure 8-5. An illustration of how a ROC "curve" (really, a stepwise graph) is constructed from a test set. The example set, at left, consists of 100 positives and 100 negatives. The model assigns a score to each instance and the instances are ordered decreasing from bottom to top. To construct the curve, start at the bottom with an initial confusion matrix where everything is classified as **N**. Moving upward, every instance moves a count of 1 from the **N** row to the **Y** row, resulting in a new confusion matrix. Each confusion matrix maps to a (fp rate, tp rate) pair in ROC space.*

An advantage of ROC graphs is that they decouple classifier performance from the conditions under which the classifiers will be used. Specifically, they are independent of the class proportions as well as the costs and benefits. A data scientist can plot the

4. Technically, if there are runs of examples with the same score, we should count the positive and negatives across the entire run, and thus the ROC curve will have a sloping step rather than square step.

performance of classifiers on a ROC graph as they are generated, knowing that the positions and relative performance of the classifiers will not change. The region(s) on the ROC graph that are of interest may change as costs, benefits, and class proportions change, but the curves themselves should not.

Both Stein (2005) and Provost & Fawcett (1997, 2001) show how the operating conditions of the classifier (the class priors and error costs) can be combined to identify the region of interest on its ROC curve. Briefly, knowledge about the range of possible class priors can be combined with knowledge about the cost and benefits of decisions; together these describe a family of tangent lines that can identify which classifier(s) should be used under those conditions. Stein (2005) presents an example from finance (loan defaulting) and shows how this technique can be used to choose models.

The Area Under the ROC Curve (AUC)

An important summary statistic is the *area under the ROC curve* (AUC). As the name implies, this is simply the area under a classifier's curve expressed as a fraction of the unit square. Its value ranges from zero to one. Though a ROC curve provides more information than its area, the AUC is useful when a single number is needed to summarize performance, or when nothing is known about the operating conditions. Later, in "Example: Performance Analytics for Churn Modeling" on page 223, we will show a use of the AUC statistic. For now it is enough to realize that it's a good general summary statistic of the predictiveness of a classifier.

As a technical note, the AUC is equivalent to the Mann-Whitney-Wilcoxon measure, a well-known ordering measure in Statistics (Wilcoxon, 1945). It is also equivalent to the Gini Coefficient, with a minor algebraic transformation (Adams & Hand, 1999; Stein, 2005). Both are equivalent to the probability that a randomly chosen positive instance will be ranked ahead of a randomly chosen negative instance.

Cumulative Response and Lift Curves

ROC curves are a common tool for visualizing model performance for classification, class probability estimation, and scoring. However, as you may have just experienced if you are new to all this, ROC curves are not the most intuitive visualization for many business stakeholders who really ought to understand the results. It is important for the data scientist to realize that clear communication with key stakeholders is not only a primary goal of her job, but also is essential for doing the right modeling (in addition to doing the modeling right). Therefore, it can be useful also to consider visualization frameworks that might not have all of the nice properties of ROC curves, but are more intuitive. (It is important for the business stakeholder to realize that the theoretical

properties that are sacrificed sometimes are important, so it may be necessary in certain circumstances to pull out the more complex visualizations.)

One of the most common examples of the use of an alternate visualization is the use of the "cumulative response curve," rather than the ROC curve. They are closely related, but the cumulative response curve is more intuitive. Cumulative response curves plot the hit rate (*tp* rate; *y* axis), *i.e.*, the *percentage of positives correctly classified*, as a function of the percentage of the population that is targeted (*x* axis). So, conceptually as we move down the list of instances ranked by the model, we target increasingly larger proportions of all the instances. Hopefully in the process, if the model is any good, when we are at the top of the list we will target a larger proportion of the actual positives than actual negatives. As with ROC curves, the diagonal line $x=y$ represents random performance. In this case, the intuition is clear: if you target 20% of all instances completely randomly, you should target 20% of the positives as well. Any classifier above the diagonal is providing some advantage.

The cumulative response curve is sometimes called a *lift curve*, because one can see the increase over simply targeting randomly as how much the line representing the model performance is lifted up over the random performance diagonal. We will call these curves cumulative response curves, because "lift curve" also refers to a curve that specifically plots the numeric lift.

Intuitively, the lift of a classifier represents the advantage it provides over random guessing. The lift is the degree to which it "pushes up" the positive instances in a list above the negative instances. For example, consider a list of 100 customers, half of whom churn (positive instances) and half who do not (negative instances). If you scan down the list and stop halfway (representing 0.5 targeted), how many positives would you expect to have seen in the first half? If the list were sorted randomly, you would expect to have seen only half the positives (0.5), giving a lift of 0.5/0.5 = 1. If the list had been ordered by an effective ranking classifier, more than half the positives should appear in the top half of the list, producing a lift greater than 1. If the classifier were *perfect*, all positives would be ranked at the top of the list so by the midway point we would have seen all of them (1.0), giving a lift of 1.0/0.5 = 2.

Figure 8-6 shows four sample cumulative response curves, and Figure 8-7 shows the lift curves of the same four.

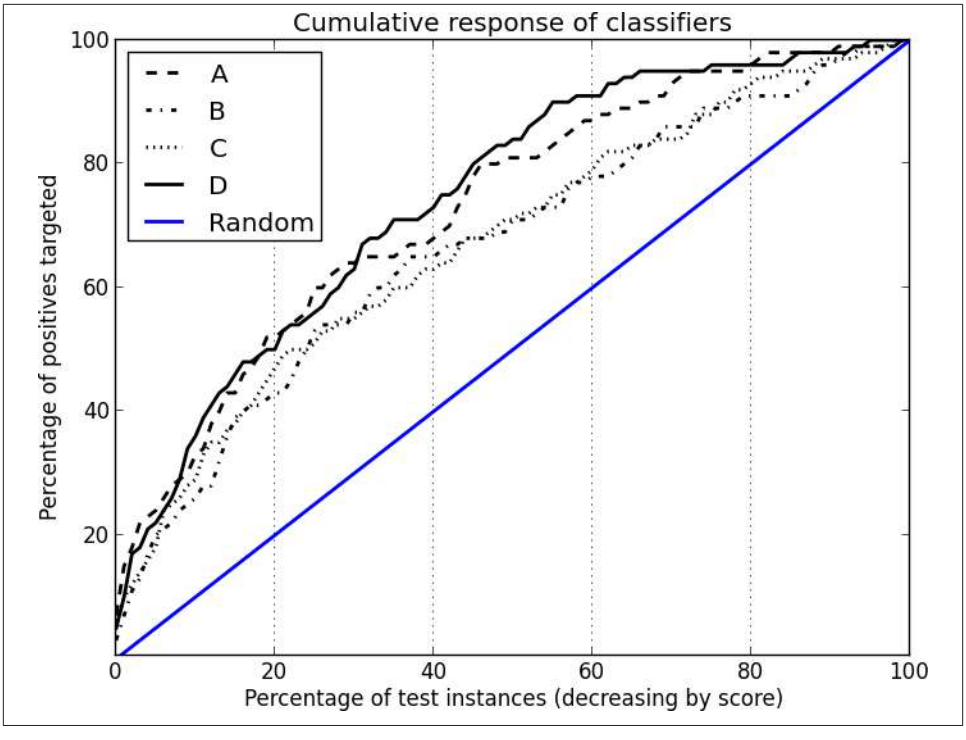

Figure 8-6. Four example classifiers (A–D) and their cumulative response curves.

The lift curve is essentially the value of the cumulative response curve at a given x point divided by the diagonal line ($y=x$) value at that point. The diagonal line of a cumulative response curve becomes a horizontal line at $y=1$ on the lift curve.

Sometimes you will hear claims like "our model gives a two times (or a 2X) lift"; this means that at the chosen threshold (often not mentioned), the lift curve shows that the model's targeting is twice as good as random. On the cumulative response curve, the corresponding *tp rate* for the model will be twice the *tp rate* for the random-performance diagonal. (You might also compute a version of lift with respect to some other baseline.) The lift curve plots this numeric lift on the y axis, against the percent of the population targeted on the x axis (the same x axis as the cumulative response curve).

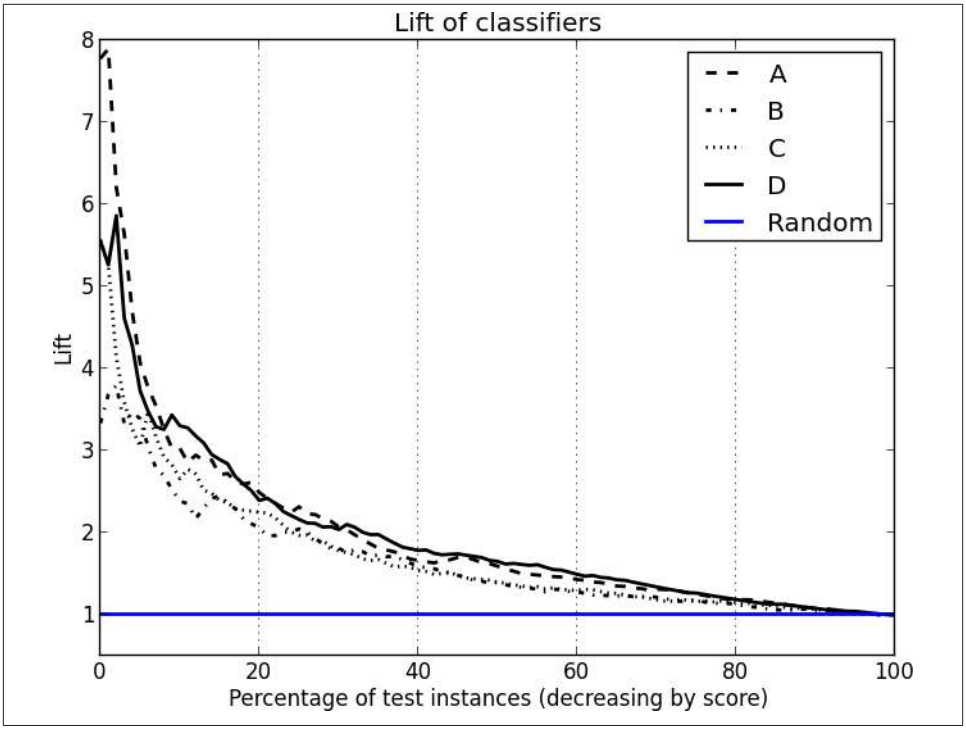

Figure 8-7. The four classifiers (A–D) of Figure 8-6 and their lift curves.

Both lift curves and cumulative response curves must be used with care if the exact proportion of positives in the population is unknown or is not represented accurately in the test data. Unlike for ROC curves, these curves assume that the test set has exactly the same target class priors as the population to which the model will be applied. This is one of the simplifying assumptions that we mentioned at the outset, that can allow us to use a more intuitive visualization.

As an example, in online advertising the base rate of observed response to an advertisement may be very small. One in ten million (1:10^7) is not unusual. Modelers may not want to have to manage datasets that have ten million nonresponders for every responder, so they down-sample the nonresponders, and create a more balanced dataset for modeling and evaluation. When visualizing classifier performance with ROC curves, this will have no effect (because as mentioned above, the axes each correspond only to proportions of one class). However, lift and cumulative response curves will be different—the basic shapes of the curves may still be informative, but the relationships between the values on the axes will not be valid.

Example: Performance Analytics for Churn Modeling

The last few chapters have covered a lot of territory in evaluation. We've introduced several important methods and issues in evaluating models. In this section we tie them together with a single application case study to show the results of different evaluation methods. The example we'll use is our ongoing domain of cell phone churn. However, in this section we use a different (and more difficult) churn dataset than was used in previous chapters. It is a dataset from the 2009 KDD Cup data mining competition. We did not use this dataset in earlier examples, such as Table 3-2 and Figure 3-18, because these attribute names and values have been anonymized extensively to preserve customer privacy. This leaves very little meaning in the attributes and their values, which would have interfered with our discussions. However, we can demonstrate the model performance analytics with the sanitized data. From the website:

> The KDD Cup 2009 offers the opportunity to work on large marketing databases from the French Telecom company Orange to predict the propensity of customers to switch provider (churn), buy new products or services (appetency), or buy upgrades or add-ons proposed to them to make the sale more profitable (up-selling). The most practical way, in a CRM system, to build knowledge on customer is to produce scores.
>
> A score (the output of a model) is an evaluation for all instances of a target variable to explain (i.e., churn, appetency or up-selling). Tools which produce scores allow to project, on a given population, quantifiable information. The score is computed using input variables which describe instances. Scores are then used by the information system (IS), for example, to personalize the customer relationship.

Little of the dataset is worth describing because it has been thoroughly sanitized, but its class skew is worth mentioning. There are about 47,000 instances altogether, of which about 7% are marked as churn (positive examples) and the remaining 93% are not (negatives). This is not severe skew, but it's worth noting for reasons that will become clear.

We emphasize that the intention is not to propose good solutions for this problem, or to suggest which models might work well, but simply to use the domain as a testbed to illustrate the ideas about evaluation we've been developing. Little effort has been done to tune performance. We will train and test several models: a classification tree, a logistic regression equation, and a nearest-neighbor model. We will also use a simple Bayesian classifier called Naive Bayes, not discussed until Chapter 9. For the purpose of this section, details of the models are unimportant; all the models are "black boxes" with different performance characteristics. We're using the evaluation and visualization techniques introduced in the last chapters to understand their characteristics.

Let's begin with a very naive evaluation. We'll train on the complete dataset and then test on the *same* dataset we trained on. We'll also measure simple classification accuracies. The results are shown in Table 8-1.

Table 8-1. Accuracy values of four classifiers trained and tested on the complete KDD Cup 2009 churn problem.

Model	Accuracy
Classification tree	95%
Logistic regression	93%
k-Nearest Neighbor	100%
Naive Bayes	76%

Several things are striking here. First, there appears to be a wide range of performance —from 76% to 100%. Also, since the dataset has a base rate of 93%, any classifier should be able to achieve at least this minimum accuracy. This makes the Naive Bayes result look strange since it's significantly worse. Also, at 100% accuracy, the *k*-Nearest Neighbor classifier looks suspiciously good.[5]

But this test was performed on the training set, and by now (having read Chapter 5) you realize such numbers are unreliable, if not completely meaningless. They are more likely to be an indication of how well each classifier can memorize (overfit) the training set than anything else. So instead of investigating these numbers further, let's redo the evaluation properly using separate training and test sets. We could just split the dataset in half, but instead we'll use the cross-validation procedure discussed in "From Holdout Evaluation to Cross-Validation" on page 126. This will not only ensure proper separation of datasets but also provide a measure of variation in results. The results are shown in Table 8-2.

Table 8-2. Accuracy and AUC values of four classifiers on the KDD Cup 2009 churn problem. These values are from ten-fold cross-validation.

Model	Accuracy (%)	AUC
Classification Tree	91.8 ± 0.0	0.614 ± 0.014
Logistic Regression	93.0 ± 0.1	0.574 ± 0.023
k-Nearest Neighbor	93.0 ± 0.0	0.537 ± 0.015
Naive Bayes	76.5 ± 0.6	0.632 ± 0.019

5. Optimism can be a fine thing, but as a rule of thumb in data mining, any results that show perfect performance on a real-world problem should be mistrusted.

Each number is an average of ten-fold cross validation followed by a "±" sign and the standard deviation of the measurements. Including a standard deviation may be regarded as a kind of "sanity check": a large standard deviation indicates the test results are very erratic, which could be the source of various problems such as the dataset being too small or the model being a very poor match to a portion of the problem.

The accuracy numbers have all dropped considerably, except for Naive Bayes, which is still oddly low. The standard deviations are fairly small compared to the means so there is not a great deal of variation in the performance on the folds. This is good.

At the far right is a second value, the Area Under the ROC Curve (commonly abbreviated AUC). We briefly discussed this AUC measure back in "The Area Under the ROC Curve (AUC)" on page 219, noting it as a good general summary statistic of the predictiveness of a classifier. It varies from zero to one. A value of 0.5 corresponds to randomness (the classifier cannot distinguish at all between positives and negatives) and a value of one means that it is perfect in distinguishing them. One of the reasons accuracy is a poor metric is that it is misleading when datasets are skewed, which this one is (93% negatives and 7% positives).

Recall that we introduced fitting curves back in "Overfitting Examined" on page 113 as a way to detect when a model is overfitting. Figure 8-8 shows fitting curves for the classification tree model on this churn domain. The idea is that as a model is allowed to get more and more complex it typically fits the data more and more closely, but at some point it is simply memorizing idiosyncracies of the particular training set rather than learning general characteristics of the population. A fitting curve plots model complexity (in this case, the number of nodes in the tree) against a performance measure (in this case, AUC) using two datasets: the set it was trained upon and a separate holdout set. When performance on the holdout set starts to decrease, overfitting is occurring, and Figure 8-8 does indeed follow this general pattern.[6] The classification tree definitely *is* overfitting, and the other models probably are too. The "sweet spot" where holdout performance is maximum is at about 100 tree nodes, beyond which the performance on the holdout data declines.

6. Note that the *x* axis is log scale so the righthand side of the graph looks compressed.

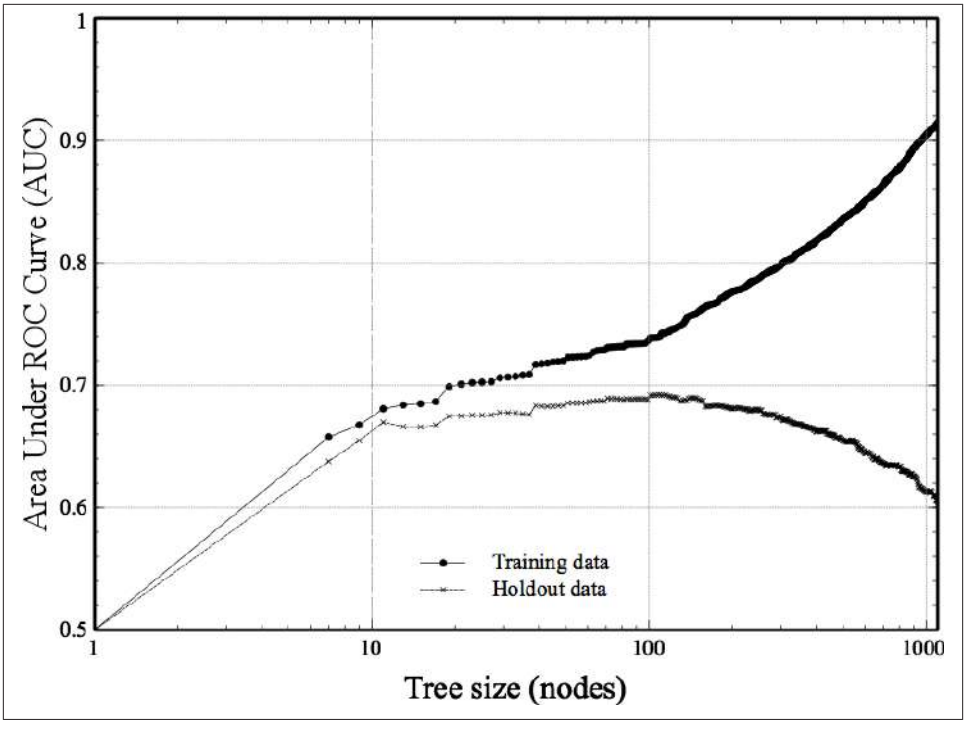

Figure 8-8. Fitting curves for a classification tree on the churn data: the change in the area under the ROC curve (AUC) as we increase the allowed complexity (size) of the tree. The performance on the training data (upper curve) continues to increase whereas the performance on the holdout data peaks and then declines.

Let's return to the model comparison figures in Table 8-2. These values are taken from a reasonably careful evaluation using holdout data, so they are less suspicious. However, they do raise some questions. There are two things to note about the AUC values. One is that they are all fairly modest. This is unsurprising with real-world domains: many datasets simply have little signal to be exploited, or the data science problem is formulated after the easier problems have already been solved. Customer churn is a difficult problem so we shouldn't be too surprised by these modest AUC scores. Even modest AUC scores may lead to good business results.

The second interesting point is that Naive Bayes, which has the *lowest* accuracy of the group, has the *highest* AUC score in Table 8-2. What's going on here? Let's take a look at a sample confusion matrix of Naive Bayes, with the highest AUC and lowest accuracy, and compare it with the confusion matrix of *k*-NN (lowest AUC and high accuracy) on the same dataset. Here is the Naive Bayes confusion matrix:

	p	n
Y	127 (3%)	848 (18%)
N	200 (4%)	3518 (75%)

Here is the *k*-Nearest Neighbors confusion matrix on the same test data:

	p	n
Y	3 (0%)	15 (0%)
N	324 (7%)	4351 (93%)

We see from the *k*-NN matrix that it rarely predicts churn—the **Y** row is almost empty. In other words, it is performing very much like a base-rate classifier, with a total accuracy of just about 93%. On the other hand, the Naive Bayes classifier makes more mistakes (so its accuracy is lower) but it identifies many more of the churners. Figure 8-9 shows the ROC curves of a typical fold of the cross-validation procedure. Note that the curves corresponding to Naive Bayes (NB) and Classification Tree (Tree) are somewhat more "bowed" than the others, indicating their predictive superiority.

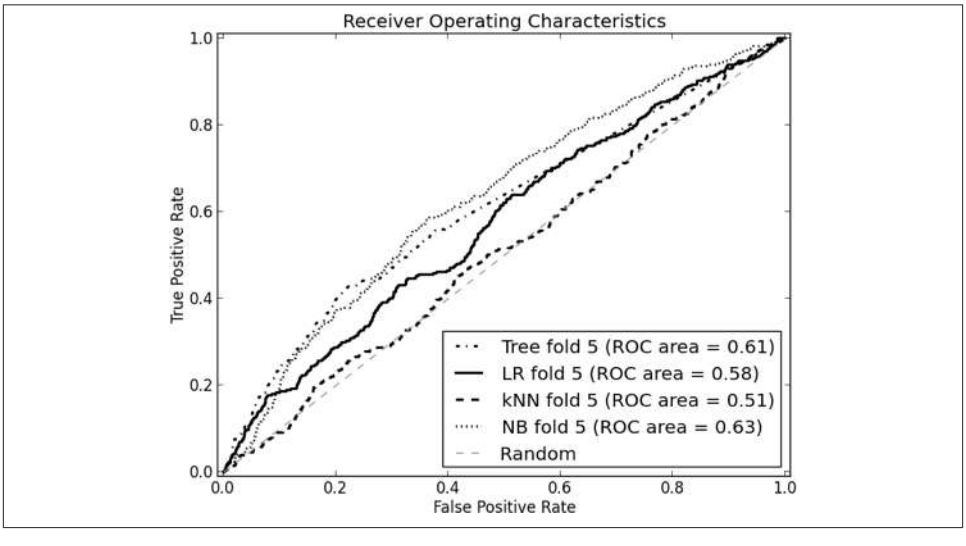

Figure 8-9. ROC curves of the classifiers on one fold of cross-validation for the churn problem.

As we said, ROC curves have a number of nice technical properties but they can be hard to read. The degree of "bowing" and the relative superiority of one curve to another can be difficult to judge by eye. Lift and profit curves are sometimes preferable, so let's examine these.

Lift curves have the advantage that they don't require us to commit to any costs yet so we begin with those, shown in Figure 8-10.

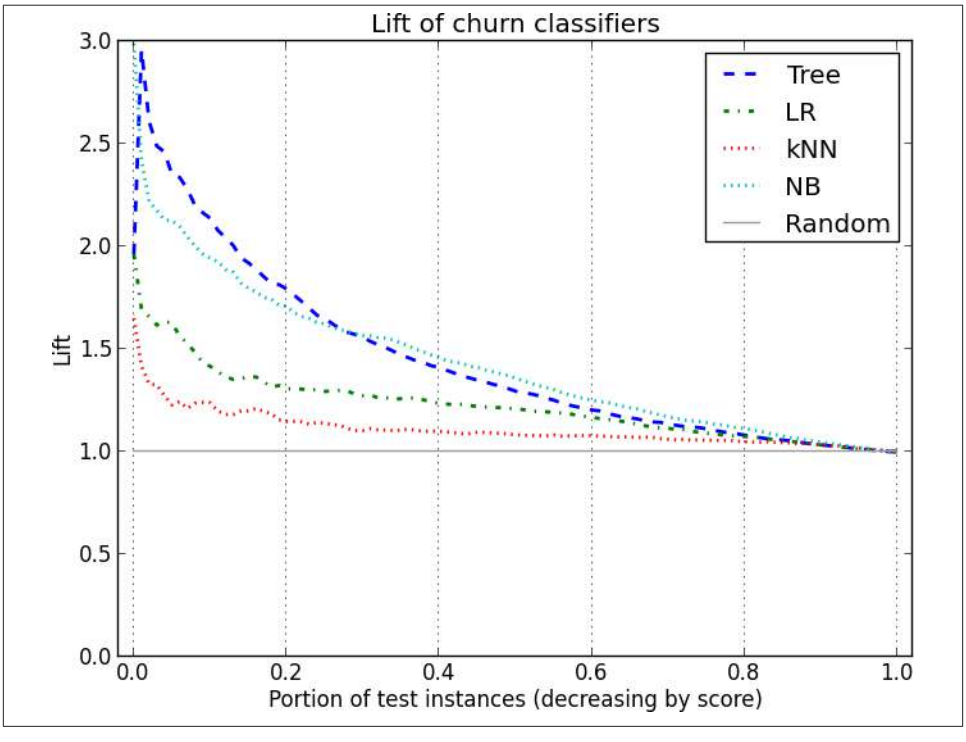

Figure 8-10. Lift curves for the churn domain.

These curves are averaged over the 10 test sets of the cross-validation. The classifiers generally peak very early then trail off down to random performance (Lift=1). Both Tree (Classification tree) and NB (Naive Bayes) perform very well. Tree is superior up through about the first 25% of the instances, after which it is dominated by NB. Both *k*-NN and Logistic Regression (LR) perform poorly here and have no regions of superiority. Looking at this graph, if you wanted to target the top 25% or less of customers, you'd choose the classification tree model; if you wanted to go further down the list you should choose NB. Lift curves are sensitive to the class proportions, so if the ratio of churners to nonchurners changed these curves would change also.

A note on combining classifiers

Looking at these curves, you might wonder, *"If Tree is best for the top 25%, and NB is best for the remainder, why don't we just use Tree's top 25% then switch to NB's list for the rest?"* This is a clever idea, but you won't necessarily get the best of both classifiers that way. The reason, in short, is that the two orderings are different and you can't simply pick-and-choose segments from each and expect the result to be optimal. The evaluation curves are only valid for each model individually, and all bets are off when you start mixing orderings from each.

But classifiers *can* be combined in principled ways, such that the combination outperforms any individual classifier. Such combinations are called ensembles, and they are discussed in "Bias, Variance, and Ensemble Methods" on page 306.

Although the lift curve shows you the relative advantage of each model, it does *not* tell you how much profit you should expect to make—or even whether you'd make a profit at all. For that purpose we use a profit curve, which incorporates assumptions about costs and benefits and displays expected value.

Let's ignore the actual details of churn in wireless for the moment (we will return to these explicitly in Chapter 11). To make things interesting with this dataset, let's make two sets of assumptions about costs and benefits. In the first scenario, let's assume an expense of $3 for each offer and a gross benefit of $30, so a true positive gives us a net profit of $27 and a false positive gives a net loss of $3. This is a 9-to-1 profit ratio. The resulting profit curves are shown in Figure 8-11. The classification tree is superior for the highest cutoff thresholds, and Naive Bayes dominates for the remainder of the possible cutoff thresholds. Maximum profit would be achieved in this scenario by targeting roughly the first 20% of the population.

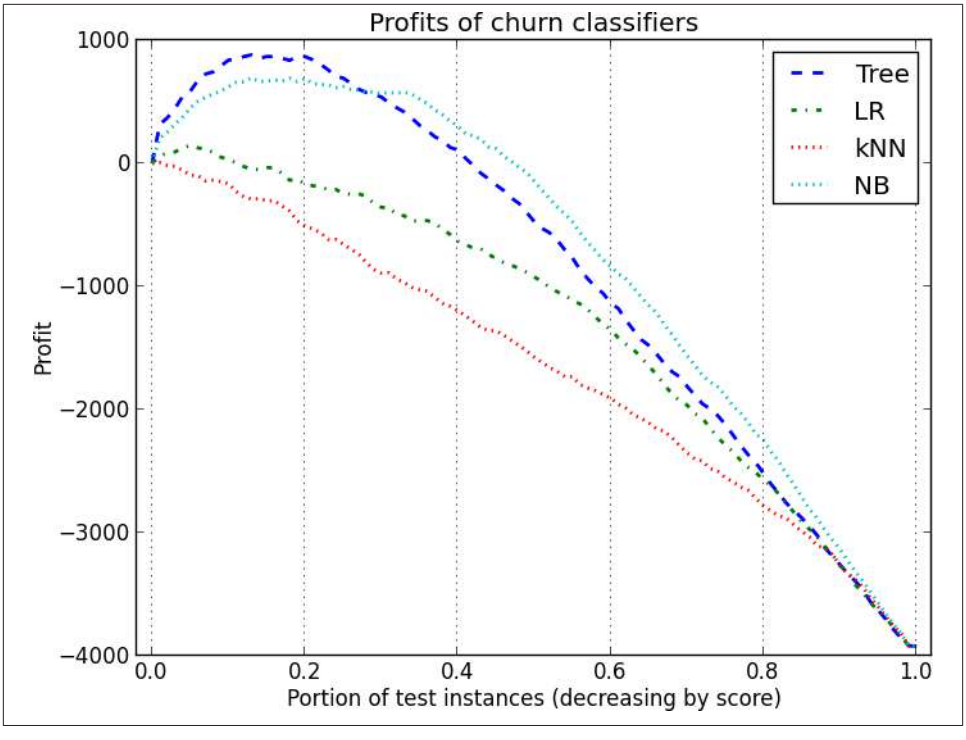

Figure 8-11. Profit curves of four classifiers on the churn domain, assuming a 9-to-1 ratio of benefit to cost.

In the second scenario, we assume the same expense of $3 for each offer (so the false positive cost doesn't change) but we assume a higher gross benefit ($39), so a true positive now nets us a profit of $36. This is a 12-to-1 profit ratio. The curves are shown in Figure 8-12. As you might expect, this scenario has much higher maximum profit than the previous scenario. More importantly it demonstrates different profit *maxima*. One peak is with the Classification Tree at about 20% of the population and the second peak, slightly higher, occurs when we target the top 35% of the population with NB. The crossover point between Tree and LR occurs at the same place on both graphs, however: at about 25% of the population. This illustrates the sensitivity of profit graphs to the particular assumptions about costs and benefits.

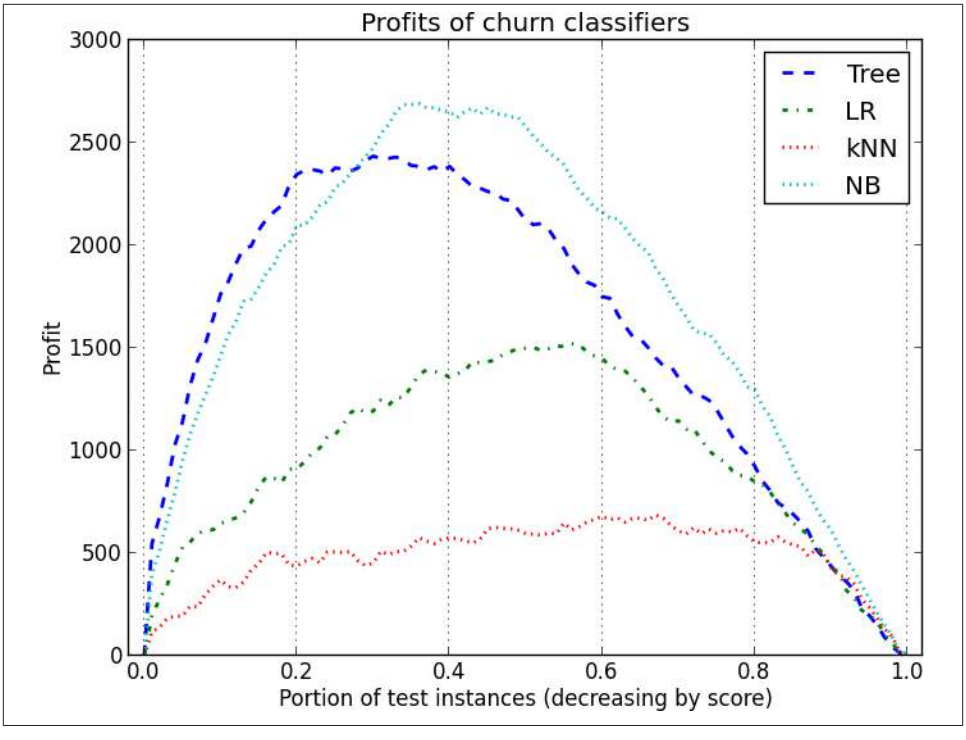

Figure 8-12. Profit curves of four classifiers on the churn domain. These curves assume a more lucrative 12-to-1 ratio (compare with Figure 8-11).

We conclude this section by reiterating that these graphs are just meant to illustrate the different techniques for model evaluation. Little effort was made to tune the induction methods to the problem, and no general conclusions should be drawn about the relative merits of these model types or their suitability for churn prediction. We deliberately produced a range of classifier performance to illustrate how the graphs could reveal their differences.

Summary

A critical part of the data scientist's job is arranging for proper evaluation of models and conveying this information to stakeholders. Doing this well takes experience, but it is vital in order to reduce surprises and to manage expectations among all concerned. Visualization of results is an important piece of the evaluation task.

When building a model from data, adjusting the training sample in various ways may be useful or even necessary; but evaluation should use a sample reflecting the original, realistic population so that the results reflect what will actually be achieved.

When the costs and benefits of decisions can be specified, the data scientist can calculate an expected cost per instance for each model and simply choose whichever model produces the best value. In some cases a basic *profit* graph can be useful to compare models of interest under a range of conditions. These graphs may be easy to comprehend for stakeholders who are not data scientists, since they reduce model performance to their basic "bottom line" cost or profit.

The disadvantage of a profit graph is that it requires that operating conditions be known and specified exactly. With many real-world problems, the operating conditions are imprecise or change over time, and the data scientist must contend with uncertainty. In such cases other graphs may be more useful. When costs and benefits cannot be specified with confidence, but the class mix will likely not change, a *cumulative response* or *lift* graph is useful. Both show the relative advantages of classifiers, independent of the value (monetary or otherwise) of the advantages.

Finally, ROC curves are a valuable visualization tool for the data scientist. Though they take some practice to interpret readily, they separate out performance from operating conditions. In doing so they convey the fundamental trade-offs that each model is making.

A great deal of work in the Machine Learning and Data Mining communities involves comparing classifiers in order to support various claims about learning algorithm superiority. As a result, much has been written about the methodology of classifier comparison. For the interested reader a good place to start is Thomas Dietterich's (1998) article "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," and the book *Evaluating Learning Algorithms: A Classification Perspective* (Japkowicz & Shah, 2011).

CHAPTER 9
Evidence and Probabilities

Fundamental concepts: *Explicit evidence combination with Bayes' Rule; Probabilistic reasoning via assumptions of conditional independence.*

Exemplary techniques: *Naive Bayes classification*; *Evidence lift*.

So far we have examined several different methods for using data to help draw conclusions about some unknown quantity of a data instance, such as its classification. Let's now examine a different way of looking at drawing such conclusions. We could think about the things that we know about a data instance as *evidence* for or against different values for the target. The things that we know about the data instance are represented as the features of the instance. If we knew the strength of the evidence given by each feature, we could apply principled methods for combining evidence probabilistically to reach a conclusion as to the value for the target. We will determine the strength of any particular piece of evidence from the training data.

Example: Targeting Online Consumers With Advertisements

To illustrate, let's consider another business application of classification: targeting online display advertisements to consumers, based on what webpages they have visited in the past. As consumers, we have become used to getting a vast amount of information and services on the Web seemingly for free. Of course, the "for free" part is very often due to the existence or promise of revenue from online advertising, similar to how broadcast television is "free." Let's consider *display advertising*—the ads that appear on the top, sides, and bottom of pages full of content that we are reading or otherwise consuming.

Display advertising is different from search advertising (e.g., the ads that appear with the results of a Google search) in an important way: for most webpages, the user has not typed in a phrase related to exactly what she is looking for. Therefore, the targeting of an advertisement to the user needs to be based on other sorts of inference. For several

chapters now we have been talking about a particular sort of inference: inferring the value of an instance's target variable from the values of the instance's features. Therefore, we could apply the techniques we already have covered to infer whether a particular user would be interested in an advertisement. In this chapter we will introduce a different way of looking at the problem, that has wide applicability and is quite easy to apply.

Let's define our ad targeting problem more precisely. What will be an instance? What will be the target variable? What will be the features? How will we get the training data?

Let's assume that we are working for a very large content provider (a "publisher") who has a wide variety of content, sees many online consumers, and has many opportunities to show advertisements to these consumers. For example, Yahoo! has a vast number of different advertisement-supported web "properties," which we can think of as different "content pieces." In addition, recently (as of this writing) Yahoo! agreed to purchase the blogging site Tumblr, which has 50 billion blog posts across over 100 million blogs. Each of these might also be seen as a "content piece" that gives some view into the interests of a consumer who reads it. Similarly, Facebook might consider each "Like" that a consumer makes as a piece of evidence regarding the consumer's tastes, which might help target ads as well.

For simplicity, assume we have one advertising campaign for which we would like to target some subset of the online consumers that visit our sites. This campaign is for the upscale hotel chain, Luxhote. The goal of Luxhote is for people to book rooms. We have run this campaign in the past, selecting online consumers randomly. We now want to run a targeted campaign, hopefully getting more bookings per dollar spent on ad impressions.[1]

Therefore, we will consider a consumer to be an instance. Our target variable will be: did/will the consumer book a Luxhote room within one week after having seen the Luxhote advertisement? Through the magic of browser cookies,[2] in collaboration with Luxhote we can observe which consumers book rooms. For training, we will have a binary value for this target variable for each consumer. In use, we will estimate the probability that a consumer will book a room after having seen an ad, and then, as our budget allows, target some subset of the highest probability consumers.

We are left with a key question: what will be the features we will use to describe the consumers, such that we might be able to differentiate those that are more or less likely to be good customers for Luxhote? For this example, we will consider a consumer to be described by the set of content pieces that we have observed her to have viewed (or Liked) previously, again as recorded via browser cookies or some other mechanism. We

1. An ad *impression* is when an ad is displayed somewhere on a page, regardless of whether a user clicks it.
2. A browser exchanges small amounts of information ("cookies") with the sites that are visited, and saves site-specific information that can be retrieved later by the same website.

have many different kinds of content: finance, sports, entertainment, cooking blogs, etc. We might pick several thousand content pieces that are very popular, or we may consider hundreds of millions. We believe that some of these (e.g., finance blogs) are more likely to be visited by good prospects for Luxhote, while other content pieces are seen as less likely (e.g., a tractor-pull fan page).

However, for this exercise we do not want to rely on our presumptions about such content, nor do we have the resources to estimate the evidence potential for each content piece manually. Furthermore, while humans are quite good at using our knowledge and common sense to recognize whether evidence is likely to be "for" or "against," humans are notoriously bad at estimating the precise *strength* of the evidence. We would like our historical data to estimate both the direction and the strength of the evidence. We next will describe a very broadly applicable framework both for evaluating the evidence, and for combining it to estimate the resulting likelihood of class membership (here, the likelihood that a consumer will book a room after having seen the ad).

It turns out that there are many other problems that fit the mold of our example: classification/class probability estimation problems where each instance is described by a set of pieces of evidence, possibly taken from a very large total collection of possible evidence. For example, text document classification fits exactly (which we'll discuss next in Chapter 10). Each document is a collection of words, from a very large total vocabulary. Each word can possibly provide some evidence for or against the classification, and we would like to combine the evidence. The techniques that we introduce next are exactly those used in many spam detection systems: an instance is an email message, the target classes are **spam** or **not-spam**, and the features are the words and symbols in the email message.

Combining Evidence Probabilistically

More math than usual ahead

To discuss the ideas of combining evidence probabilistically, we need to introduce some probability notation. You do not have to have learned (or remember) probability theory—the notions are quite intuitive, and we will not get beyond the basics. The notation allows us to be precise. It might look like there's a lot of math in what follows, but you'll see that it's quite straightforward.

We are interested in quantities such as the probability of a consumer booking a room after being shown an ad. We actually need to be a little more specific: some particular consumer? Or just any consumer? Let's start with just any consumer: what is the probability that if you show an ad to just any consumer, she will book a room? As this is our desired classification, let's call this quantity *C*. We will represent the probability of an

event C as $p(C)$. If we say $p(C) = 0.0001$, that means that if we were to show ads randomly to consumers, we would expect about 1 in 10,000 to book rooms.[3]

Now, we are interested in the probability of C given some evidence E, such as the set of websites visited by a *particular* consumer. The notation for this quantity is $p(C|E)$, which is read as "the probability of C given E," or "the probability of C conditioned on E." This is an example of a conditional probability, and the "|" is sometimes called the "conditioning bar." We would expect that $p(C|E)$ would be different based on different collections of evidence E—in our example, different sets of websites visited.

As mentioned above, we would like to use some labeled data, such as the data from our randomly targeted campaign, to associate different collections of evidence E with different probabilities. Unfortunately, this introduces a key problem. For any particular collection of evidence E, we probably have not seen enough cases with exactly that same collection of evidence to be able to infer the probability of class membership with any confidence. In fact, we may not have seen this particular collection of evidence at all! In our example, if we are considering thousands of different websites, what is the chance that in our training data we have seen a consumer with *exactly* the same visiting patterns as a consumer we will see in the future? It is infinitesimal. Therefore, what we will do is to consider the different pieces of evidence separately, and then combine evidence. To discuss this further, we need a few facts about combining probabilities.

Joint Probability and Independence

Let's say we have two events, A and B. If we know $p(A)$ and $p(B)$, can we say what is the probability that both A and B occur? Let's call that $p(AB)$. This is called the *joint* probability.

There is one special case when we can: if events A and B are *independent*. A and B being independent means that knowing about one of them tells you nothing about the likelihood of the other. The typical example used to illustrate independence is rolling a fair die; knowing the value of the first roll tells you nothing about the value of the second. If event A is "roll #1 shows a six" and event B is "roll #2 shows a six", then $p(A) = 1/6$ and $p(B) = 1/6$, and importantly, even if we *know* that roll #1 shows a six, still $p(B) = 1/6$. In this case, the events are independent, and in the case of independent events, $p(AB) = p(A) \cdot p(B)$—we can calculate the probability of the "joint" event AB by multiplying the probabilities of the individual events. In our example, $p(AB) = 1/36$.

However, we cannot in general compute the probabilities of joint events in this way. If this isn't clear, think about the case of rolling a trick die. In my pocket I have six trick dice. Each trick die has one of the numbers from one to six on all faces—all faces show

3. This is not necessarily a reasonable response rate for any particular advertisement, just an illustrative example. Purchase rates attributable to online advertisements generally seem very small to those outside the industry. It is important to realize that the cost of placing one ad often is quite small as well.

the same number. I pull a die at random from my pocket, and then roll it twice. In this case, $p(A) = p(B) = 1/6$ (because I could have pulled any of the six dice out with equal likelihood). However, $p(AB) = 1/6$ as well, because the events are completely dependent! If the first roll is a six, so will be the second (and vice versa).

The general formula for combining probabilities that takes care of dependencies between events is:

Equation 9-1. Joint probability using conditional probability

$$p(AB) = p(A) \cdot p(B \mid A)$$

This is read as: the probability of A and B is the probability of A times the probability of B *given* A. In other words, given that you know A, what is the probability of B? Take a minute to make sure that has sunk in.

We can illustrate with our two dice examples. In the independent case, since knowing A tells us nothing about $p(B)$, then $p(B|A) = p(B)$, and we get our formula from above, where we simply multiply the individual probabilities. In our trick die case, $p(B|A) = 1.0$, since if the first roll was a six, then the second roll is guaranteed to be a six. Thus, $p(AB) = p(A) \cdot 1.0 = p(A) = 1/6$, just as expected. In general, events may be completely independent, completely dependent, or somewhere in between. In the latter case, knowing something about one event changes the likelihood of the other. In all cases, our formula $p(AB) = p(A) \cdot p(B|A)$ combines the probabilities properly.

We've gone through this detail for a very important reason. This formula is the basis for one of the most famous equations in data science, and in fact in science generally.

Bayes' Rule

Notice that in $p(AB) = p(A)p(B|A)$ the order of A and B seems rather arbitrary—and it is. We could just as well have written:

$$p(AB) = p(B) \cdot p(A \mid B)$$

This means:

$$p(A) \cdot p(B \mid A) = p(AB) = p(B) \cdot p(A \mid B)$$

And so:

$$p(A) \cdot p(B \mid A) = p(B) \cdot p(A \mid B)$$

If we divide both sides by $p(A)$ we get:

$$p(B \mid A) = \frac{p(A \mid B) \cdot p(B)}{p(A)}$$

Now, let's consider B to be some hypothesis that we are interested in assessing the likelihood of, and A to be some evidence that we have observed. Renaming with H for hypothesis and E for evidence, we get:

$$p(H \mid E) = \frac{p(E \mid H) \cdot p(H)}{p(E)}$$

This is the famous *Bayes' Rule*, named after the Reverend Thomas Bayes who derived a special case of the rule back in the 18th century. Bayes' Rule says that we can compute the probability of our hypothesis H given some evidence E by instead looking at the probability of the evidence given the hypothesis, as well as the unconditional probabilities of the hypothesis and the evidence.

Note: Bayesian methods

Bayes' Rule, combined with the important fundamental principle of thinking carefully about conditional independencies, are the foundation for a vast amount of more advanced data science techniques that we will not cover in this book. These include Bayesian networks, probabilistic topic models, probabilistic relational models, Hidden Markov Models, Markov random fields, and others.

Importantly, the last three quantities may be easier to determine than the quantity of ultimate interest—namely, $p(H|E)$. To see this, consider a (simplified) example from medical diagnosis. Assume you're a doctor and a patient arrives with red spots. You guess (hypothesize) that the patient has measles. We would like to determine the probability of our hypothesized diagnosis (H = measles), given the evidence (E = red spots). In order to directly estimate p(measles|red spots) we would need to think through all the different reasons a person might exhibit red spots and what proportion of them would be measles. This is likely impossible even for the most broadly knowledgeable physician.

However, consider instead the task of estimating this quantity using the righthand side of Bayes' Rule.

- $p(E|H)$ is the probability that one has red spots given that one has measles. An expert in infectious diseases may well know this or be able to estimate it relatively accurately.

- $p(H)$ is simply the probability that someone has measles, without considering any evidence; that's just the prevalence of measles in the population.
- $p(E)$ is the probability of the evidence: what's the probability that someone has red spots—again, simply the prevalence of red spots in the population, which does not require complicated reasoning about the different underlying causes, just observation and counting.

Bayes' Rule has made estimating $p(H|E)$ much easier. We need three pieces of information, but they're much easier to estimate than the original value is.

$p(E)$ may still be difficult to compute. In many cases, though, it does not have to be computed, because we are interested in comparing the probabilities of different hypotheses given the same evidence. We will see this later.

Applying Bayes' Rule to Data Science

It is possibly quite obvious now that Bayes' Rule should be critical in data science. Indeed, a very large portion of data science is based on "Bayesian" methods, which have at their core reasoning based on Bayes' Rule. Describing Bayesian methods broadly is well beyond the scope of this book. We will introduce the most fundamental ideas, and then show how they apply in the most basic of Bayesian techniques—which is used a great deal. Let's rewrite Bayes' Rule yet again, but now returning to classification.

Equation 9-2. Bayes Rule for classification

$$p(C = c \mid \mathbf{E}) = \frac{p(\mathbf{E} \mid C = c) \cdot p(C = c)}{p(\mathbf{E})}$$

In Equation 9-2, we have four quantities. On the lefthand side is the quantity we would like to estimate. In the context of a classification problem, this is the probability that the target variable C takes on the class of interest c *after* taking the evidence $\mathbf{E}$ (the **vector** of feature values) into account. This is called the *posterior* probability.

Bayes' Rule decomposes the posterior probability into the three quantities that we see on the righthand side. We would like to be able to compute these quantities from the data:

1. $p(C = c)$ is the *"prior"* probability of the class, i.e., the probability we would assign to the class before seeing any evidence. In Bayesian reasoning generally, this could come from several places. It could be (i) a "subjective" prior, meaning that it is the

belief of a particular decision maker based on all her knowledge, experience, and opinions; (ii) a "prior" belief based on some previous application(s) of Bayes' Rule with other evidence, or (iii) an unconditional probability inferred from data. The specific method we introduce below takes approach (iii), using as the *class prior* the "base rate" of c—the prevalence of c in the population as a whole. This is calculated easily from the data as the percentage of all examples that are of class c.

2. $p(\mathbf{E}|C = c)$ is the *likelihood* of seeing the evidence **E**—the particular features of the example being classified—when the class $C = c$. One might see this as a "generative" question: if the world (the "data generating process") generated an instance of class c, how often would it look like **E**? This likelihood might be calculated from the data as the percentage of examples of class c that have feature vector **E**.
3. Finally, $p(\mathbf{E})$ is the likelihood of the evidence: how common is the feature representation **E** among all examples? This might be calculated from the data as the percentage occurrence of **E** among all examples.

Estimating these three values from training data, we could calculate an estimate for the posterior $p(C = c|\mathbf{E})$ for a particular example in use. This could be used directly as an estimate of class probability, possibly in combination with costs and benefits as described in Chapter 7. Alternatively, $p(C = c|\mathbf{E})$ could be used as a score to rank instances (e.g., estimating those that are most likely to respond to our advertisement). Or, we could choose as the classification the maximum $p(C = c|\mathbf{E})$ across the different values c.

Unfortunately, we return to the major difficulty we mentioned above, which keeps Equation 9-2 from being used directly in data mining. Consider **E** to be our usual vector of attribute values $<e_1, e_2, \cdots, e_k>$, a possibly large, specific collection of conditions. Applying Equation 9-2 directly would require knowing the $p(\mathbf{E}|c)$ as $p(e_1 \wedge e_2 \wedge \cdots \wedge e_k|c)$.[4] This is very specific and very difficult to measure. We may never see a specific example in the training data that exactly matches a given **E** in our testing data, and even if we do it may be unlikely we'll see enough of them to estimate a probability with any confidence.

Bayesian methods for data science deal with this issue by making assumptions of probabilistic independence. The most broadly used method for dealing with this complication is to make a particularly strong assumption of independence.

Conditional Independence and Naive Bayes

Recall from above the notion of independence: two events are independent if knowing one does not give you information on the probability of the other. Let's extend that notion ever so slightly.

4. The ∧ operator means "and."

Conditional independence is the same notion, except using conditional probabilities. For our purposes, we will focus on the class of the example as the condition (since in Equation 9-2 we are looking at the probability of the evidence *given* the class). Conditional independence is directly analogous to the unconditional independence we discussed above. Specifically, without assuming independence, to combine probabilities we need to use Equation 9-1 from above, augmented with the $|C$ condition:

$$p(AB \mid C) = p(A \mid C) \cdot p(B \mid AC)$$

However, as above, if we assume that A and B are conditionally independent given C,[5] we can now combine the probabilities much more easily:

$$p(AB \mid C) = p(A \mid C) \cdot p(B \mid C)$$

This makes a huge difference in our ability to compute the probabilities from the data. In particular, for the conditional probability $p(\mathbf{E} \mid C{=}c)$ in Equation 9-2, let's assume that the attributes are conditionally independent, given the class. In other words, in $p(e_1 \wedge e_2 \wedge \cdots \wedge e_k \mid c)$, each e_i is independent of every other e_j given the class c:

$$\begin{aligned} p(\mathbf{E} \mid c) &= p(e_1 \wedge e_2 \wedge \cdots \wedge e_k \mid c) \\ &= p(e_1 \mid c) \cdot p(e_2 \mid c) \cdots p(e_k \mid c) \end{aligned}$$

Each of the $p(e_i \mid c)$ terms can be computed directly from the data, since now we simply need to count up the proportion of the time that we see individual feature e_i in the instances of class c, rather than looking for an entire matching feature vector. There are likely to be relatively many occurrences of e_i.[6] Combining this with Equation 9-2 we get the *Naive Bayes equation* as shown in Equation 9-3.

Equation 9-3. Naive Bayes equation

$$p(c \mid \mathbf{E}) = \frac{p(e_1 \mid c) \cdot p(e_2 \mid c) \cdots p(e_k \mid c) \cdot p(c)}{p(\mathbf{E})}$$

This is the basis of the *Naive Bayes classifier*. It classifies a new example by estimating the probability that the example belongs to each class and reports the class with highest probability.

5. This is a weaker assumption than assuming unconditional independence, by the way.

6. And in the cases where there are not we can use a statistical correction for small counts. The difference is that we will not be doing that for all the evidence, as we would have considering the entire E.

If you will allow two paragraphs on a technical detail: at this point you might notice the $p(\mathbf{E})$ in the denominator of Equation 9-3 and say, whoa there—if I understand you, isn't that going to be almost as difficult to compute as $p(\mathbf{E}\,|C)$? It turns out that generally $p(\mathbf{E})$ never actually has to be calculated, for one of two reasons. First, if we are interested in classification, what we mainly care about is: of the different possible classes c, for which one is $p(C|\,\mathbf{E})$ the greatest? In this case, **E** is the same for all, and we can just look to see which numerator is larger.

In cases where we would like the actual probability estimates, we still can get around computing $p(\mathbf{E})$ in the denominator. This is because the classes often are mutually exclusive and exhaustive, meaning that every instance will belong to one and only one class. In our Luxhote example, a consumer either books a room or does not. Informally, if we see evidence **E** it belongs either to c_0 or c_1. Mathematically:

$$\begin{aligned} p(\mathbf{E}) &= p(\mathbf{E} \wedge c_0) + p(\mathbf{E} \wedge c_1) \\ &= p(\mathbf{E} \mid c_0) \cdot p(c_0) + p(\mathbf{E} \mid c_1) \cdot p(c_1) \end{aligned}$$

Our independence assumption allows us to rewrite this as:

$$\begin{aligned} p(\mathbf{E}) &= p(e_1 \mid c_0) \cdot p(e_2 \mid c_0) \cdots p(e_k \mid c_0) \cdot p(c_0) \\ &\quad + p(e_1 \mid c_1) \cdot p(e_2 \mid c_1) \cdots p(e_k \mid c_1) \cdot p(c_1) \end{aligned}$$

$$p(\mathbf{E}) = p(e_1 \mid c_0) \cdot p(e_2 \mid c_0) \cdots p(e_k \mid c_0) \cdot p(c_0) + p(e_1 \mid c_1) \cdot p(e_2 \mid c_1) \cdots p(e_k \mid c_1) \cdot p(c_1)$$

Combining this with Equation 9-3, we get a version of the Naive Bayes equation with which we can compute the posterior probabilities easily from the data:

$$p(c_0 \mid \mathbf{E}) = \frac{p(e_1 \mid c_0) \cdot p(e_2 \mid c_0) \cdots p(e_k \mid c_0) \cdot p(c_0)}{p(e_1 \mid c_0) \cdot p(e_2 \mid c_0) \cdots p(e_k \mid c_0) + p(e_1 \mid c_1) \cdot p(e_2 \mid c_1) \cdots p(e_k \mid c_1)}$$

Although it has lots of terms in it, each one is either the evidence "weight" of some particular individual piece of evidence, or a class prior.

Advantages and Disadvantages of Naive Bayes

Naive Bayes is a very simple classifier, yet it still takes all the feature evidence into account. It is very efficient in terms of storage space and computation time. Training consists only of storing counts of classes and feature occurrences as each example is seen. As mentioned, $p(c)$ can be estimated by counting the proportion of examples of class c among all examples. $p(e_i|c)$ can be estimated by the proportion of examples in class c for which feature e_i appears.

In spite of its simplicity and the strict independence assumptions, the Naive Bayes classifier performs surprisingly well for classification on many real-world tasks. This is because the violation of the independence assumption tends not to hurt classification performance, for an intuitively satisfying reason. Specifically, consider that two pieces of evidence are actually strongly dependent—what does that mean? Roughly, that means that when we see one we're also likely to see the other. Now, if we treat them as being independent, we're going to see one and say "there's evidence for the class" and see the other and say "there's more evidence for the class." So, to some extent we'll be double-counting the evidence. *However*, as long as the evidence is generally pointing us in the right direction, for classification the double-counting won't tend to hurt us. In fact, it will tend to make the probability estimates more extreme in the correct direction: the probability will be overestimated for the correct class and underestimated for the incorrect class(es). But for classification we're choosing the class with the greatest probability estimate, so making them more extreme in the correct direction is OK.

This does become a problem, though, if we're going to be using the probability estimates themselves—so Naive Bayes should be used with caution for actual decision-making with costs and benefits, as discussed in Chapter 7. Practitioners do use Naive Bayes regularly for ranking, where the actual values of the probabilities are not relevant—only the relative values for examples in the different classes.

Another advantage of Naive Bayes is that it is naturally an "incremental learner." An incremental learner is an induction technique that can update its model one training example at a time. It does not need to reprocess all past training examples when new training data become available.

Incremental learning is especially advantageous in applications where training labels are revealed in the course of the application, and we would like the model to reflect this new information as quickly as possible. For example, consider creating a personalized junk email classifier. When I receive a piece of junk email, I can click the "junk" button in my browser. Besides removing this email from my Inbox, this also creates a training data point: a positive instance of spam. It would be quite useful if the model that is classifying my email could be updated on the fly, and thereby immediately start classifying other similar emails as being spam. Naive Bayes is the basis of many personalized spam detection systems, such as the one in Mozilla's Thunderbird.

Naive Bayes is included in nearly every data mining toolkit and serves as a common baseline classifier against which more sophisticated methods can be compared. We have discussed Naive Bayes using binary attributes. The basic idea presented above can be extended easily to multi valued categorical attributes, as well as to numeric attributes, as you can read about in a textbook on data mining algorithms.

A Model of Evidence "Lift"

"Cumulative Response and Lift Curves" on page 219 presented the notion of *lift* as a metric for evaluating a classifier. Intuitively, lift is the amount by which a classifier concentrates the positive examples above the negative examples. Lift measures how much more prevalent the positive class is in the selected subpopulation over the prevalence in the population as a whole. If the prevalence of hotel bookings in a randomly targeted set of consumers 0.01% and in our selected population it is 0.02%, then the classifier gives us a lift of 2—the selected population has double the booking rate.

With a slight modification, we can adapt our Naive Bayes equation to model the different lifts attributable to the different pieces of evidence, along with a very straightforward way of combining them. The slight modification is to assume full feature independence, rather than the weaker assumption of conditional independence used for Naive Bayes. Let's call this Naive-Naive Bayes, since it's making stronger simplifying assumptions about the world. Assuming full feature independence, Equation 9-3 becomes the following for Naive-Naive Bayes:

$$p(c \mid \mathbf{E}) = \frac{p(e_1 \mid c) \cdot p(e_2 \mid c) \cdots p(e_k \mid c) \cdot p(c)}{p(e_1) \cdot p(e_2) \cdots p(e_k)}$$

The terms in this equation can be rearranged to yield:

Equation 9-4. Probability as a product of evidence lifts

$$p(C = c \mid \mathbf{E}) = p(C = c) \cdot \text{lift}_c(e_1) \cdot \text{lift}_c(e_2) \cdots$$

where $\text{lift}_c(x)$ is defined as:

$$\text{lift}_c(x) = \frac{p(x \mid c)}{p(x)}$$

Consider how we'll use our Bayesian classifier to classifier a new example $\mathbf{E} = <e_1, e_2, \cdots, e_k>$. Starting at the prior probability, each piece of evidence—each feature e_i—raises or lowers the probability of the class by a factor equal to that piece of evidence's lift (which may be less than one).

Conceptually, we start off with a number—call it z—set to the prior probability of class c. We go through our example, and for each new piece of evidence e_i we multiply z by $\text{lift}_c(e_i)$. If the lift is greater than one, the probability z is increased; if less than one, z is diminished.

In the case of our Luxhote example, z is the probability of booking, and it is initialized to 0.0001 (the prior probability, before seeing evidence, that a website visitor will book a room). Visited a finance site? Multiply the probability of booking by a factor of two. Visit a truck-pull site? Multiply the probability by a factor of 0.25. And so on. After processing all of the e_i evidence bits of **E**, the resulting product (call that z_f) is the final probability (belief) that **E** is a member of class c—in this case, that visitor **E** will book a room.

Considered this way, it may become clearer what the independence assumption is doing. We are treating each bit of evidence e_i as independent of the others, so we can just multiply z by their individual lifts. But any dependencies among them will result in some distortion of the final value, z_f. It will end up either higher or lower than it properly should be. Thus the evidence lifts and their combining are very useful for understanding the data, and for *comparing* between instances, but the actual final value of the probability should be taken with a large grain of salt.

Example: Evidence Lifts from Facebook "Likes"

Let's examine some evidence lifts from real data. To freshen things up a little, let's consider a brand new domain of application. Researchers Michal Kosinski, David Stillwell, and Thore Graepel recently published a paper (Kosinski et al., 2013) in the *Proceedings of the National Academy of Sciences* showing some striking results. What people "Like" on Facebook [7] is quite predictive of all manner of traits that usually are not directly apparent:

- How they score on intelligence tests
- How they score on psychometric tests (e.g., how extroverted or conscientious they are)
- Whether they are (openly) gay
- Whether they drink alcohol or smoke
- Their religion and political views
- And many more.

We encourage you to read the paper to understand their experimental design. You should be able to understand most of the results now that you have read this book. (For

7. For those unfamiliar with Facebook, it is a social networking site that allows people to share a wide variety of information on their interests and activities. Each user has a unique page, and Facebook encourages people to connect with other friends on the site. Facebook also has pages devoted to special interests such as TV shows, movies, bands, hobbies, and so on. Each such page has a "Like" button, and users can declare themselves to be fans by clicking it. Such "Likes" can usually be seen by other friends.

example, for evaluating how well they can predict many of the binary traits they report the area under the ROC curve, which you can now interpret properly.)

What we would like to do is to look to see what are the Likes that give strong evidence lifts for "high IQ," or more specifically for scoring high on an IQ test. Taking a sample of the Facebook population, if we define our target variable as the binary variable *IQ>130*, about 14% of the sample is positive (has *IQ>130*).

So let's examine the Likes that give the highest evidence lifts…[8]

Table 9-1. Some Facebook page "Likes" and corresponding lifts.

Like	Lift	Like	Lift
Lord Of The Rings	1.69	Wikileaks	1.59
One Manga	1.57	Beethoven	1.52
Science	1.49	NPR	1.48
Psychology	1.46	*Spirited Away*	1.45
The Big Bang Theory	1.43	Running	1.41
Paulo Coelho	1.41	Roger Federer	1.40
The Daily Show	1.40	*Star Trek*	1.39
Lost	1.39	Philosophy	1.38
Lie to Me	1.37	*The Onion*	1.37
How I Met Your Mother	1.35	*The Colbert Report*	1.35
Doctor Who	1.34	*Star Trek*	1.32
Howl's Moving Castle	1.31	Sheldon Cooper	1.30
Tron	1.28	*Fight Club*	1.26
Angry Birds	1.25	*Inception*	1.25
The Godfather	1.23	*Weeds*	1.22

So, recalling Equation 9-4 above, and the independence assumptions made, we can calculate the probability that someone has very high intelligence based on the things they Like. If I Like nothing, then my estimated probability of *IQ>130* (let's call that **High-IQ**) is just the base rate in the population: 14%. What if on Facebook I had Liked one item, Sheldon Cooper. Then using Equation 9-4, my estimated probability would increase by 30% to $0.14 \times 1.3 = 18\%$. If I have three Likes—Sheldon Cooper, Star Trek, and Lord of the Rings—then my estimated probability of **High-IQ** increases to $0.14 \times 1.3 \times 1.39 \times 1.69 = 43\%$.

Of course, there are also Likes that would drag *down* my probabability of **High-IQ**. So as not to depress you, we won't list them here.

8. Thanks to Wally Wang for his generous help with generating these results.

This example also illustrates how it is important to think carefully about exactly what the results mean in light of the data generating process. This does not really mean that liking *The Lord of the Rings* is a strong indication that I have very high IQ. It means clicking "Like" on Facebook's page called **The Lord of the Rings** is a strong indication that I have very high IQ. This difference is important: the act of declaring publicly that you Like something is different from simply liking it, and the data we have represent an instance of the former and not the latter.

Evidence in Action: Targeting Consumers with Ads

In spite of the math that appears in this chapter, the calculations are quite simple to implement—so simple they can be implemented directly in a spreadsheet. So instead of presenting a static example here, we have prepared a spreadsheet with a simple numerical example illustrating Naive Bayes and evidence lift on a toy version of the online ad-targeting example. You'll see how straightforward it is to use these calculations, because they just involve counting things, computing proportions, and multiplying and dividing.

The spreadsheet can be downloaded here.

The spreadsheet lays out all the "evidence" (website visits for multiple visitors) and shows the intermediate calculations and final probability of a ficticious advertising response. You can experiment with the technique by tweaking the numbers, adding or deleting visitors, and seeing how the estimated probabilities of response and the evidence lifts adjust in response.

Summary

Prior chapters presented modeling techniques that basically asked the question: "*What is the best way to distinguish (segment) target values?*" Classification trees and linear equations both create models this way, trying to minimize loss or entropy, which are functions of discriminability. These are termed *discriminative* methods, in that they try directly to discriminate different targets.

This chapter introduced a new family of methods that essentially turns the question around and asks: "How do different targets *generate* feature values?" They attempt to model how the data were generated. In the use phase, when faced with a new example to be classified, they apply Bayes' Rule to their models to answer the question: "Which class most likely generated this example?" Thus, in data science this approach to modeling is called *generative*, and it forms the basis for a large family of popular methods

known as *Bayesian* methods, because they depend critically on Bayes' Rule. The literature on Bayesian methods is both broad and deep, and you will encounter these methods often in data science.

This chapter focused primarily on a particularly common and simple but very useful Bayesian method called the Naive Bayes classifier. It is "naive" in the sense that it models each feature's effect on the target independently, so it takes no feature interactions into account. Because of its simplicity it is very fast and efficient, and in spite of its naïveté it is surprisingly (almost embarrassingly) effective. In data science it is so simple as to be a common "baseline" method—one of the first methods to be applied to any new problem.

We also discussed how Bayesian reasoning using certain independence assumptions can allow us to compute "evidence lifts" to examine large numbers of possible pieces of evidence for or against a conclusion. As an example, we showed that "Liking" *Fight Club*, *Star Trek*, or *Sheldon Cooper* on Facebook each increases by about 30% our estimation of the probability that you have a high IQ. If you were to Like all three of those, it would more than double our estimate that you have a high IQ.

CHAPTER 10
Representing and Mining Text

Fundamental concepts: *The importance of constructing mining-friendly data representations; Representation of text for data mining.*

Exemplary techniques: *Bag of words representation; TFIDF calculation; N-grams; Stemming; Named entity extraction; Topic models.*

Up to this point we've ignored or side-stepped an important stage of the data mining process: data preparation. The world does not always present us with data in the feature vector representation that most data mining methods take as input. Data are represented in ways natural to problems from which they were derived. If we want to apply the many data mining tools that we have at our disposal, we must either engineer the data representation to match the tools, or build new tools to match the data. Top-notch data scientists employ both of these strategies. It generally is simpler to first try to engineer the data to match existing tools, since they are well understood and numerous.

In this chapter, we will focus on one particular sort of data that has become extremely common as the Internet has become a ubiquitous channel of communication: text data. Examining text data allows us to illustrate many real complexities of data engineering, and also helps us to understand better a very important type of data. We will see in Chapter 14 that although in this chapter we focus exclusively on text data, the fundamental principles indeed generalize to other important sorts of data.

We've encountered text once before in this book, in the example involving clustering news stories about Apple Inc. ("Example: Clustering Business News Stories"). There we deliberately avoided a detailed discussion of how the news stories were prepared because the focus was on clustering, and text preparation would have been too much of a digression. This chapter is devoted to the difficulties and opportunities of dealing with text.

In principle, text is just another form of data, and text processing is just a special case of representation engineering. In reality, dealing with text requires dedicated preprocessing steps and sometimes specific expertise on the part of the data science team.

Entire books and conferences (and companies) are devoted to text mining. In this chapter we can only scratch the surface, to give a basic overview of the techniques and issues involved in typical business applications.

First, let's discuss why text is so important and why it's difficult.

Why Text Is Important

Text is everywhere. Many legacy applications still produce or record text. Medical records, consumer complaint logs, product inquiries, and repair records are still mostly intended as communication between people, not computers, so they're still "coded" as text. Exploiting this vast amount of data requires converting it to a meaningful form.

The Internet may be the home of "new media," but much of it is the same form as old media. It contains a vast amount of text in the form of personal web pages, Twitter feeds, email, Facebook status updates, product descriptions, Reddit comments, blog postings—the list goes on. Underlying the search engines (Google and Bing) that we use everyday are massive amounts of text-oriented data science. Music and video may account for a great deal of traffic volume, but when people communicate with each other on the Internet it is usually via text. Indeed, the thrust of Web 2.0 was about Internet sites allowing users to interact with one another as a community, and to generate much added content of a site. This user-generated content and interaction usually takes the form of text.

In business, understanding customer feedback often requires understanding text. This isn't always the case; admittedly, some important consumer attitudes are represented explicitly as data or can be inferred through behavior, for example via five-star ratings, click-through patterns, conversion rates, and so on. We can also pay to have data collected and quantified through focus groups and online surveys. But in many cases if we want to "listen to the customer" we'll actually have to read what she's written—in product reviews, customer feedback forms, opinion pieces, and email messages.

Why Text Is Difficult

Text is often referred to as "unstructured" data. This refers to the fact that text does not have the sort of structure that we normally expect for data: tables of records with fields having fixed meanings (essentially, collections of feature vectors), as well as links between the tables. Text of course has plenty of structure, but it is *linguistic* structure—intended for human consumption, not for computers.

Words can have varying lengths and text fields can have varying numbers of words. Sometimes word order matters, sometimes not.

As data, text is relatively *dirty*. People write ungrammatically, they misspell words, they run words together, they abbreviate unpredictably, and punctuate randomly. Even when

text is flawlessly expressed it may contain synonyms (multiple words with the same meaning) and homographs (one spelling shared among multiple words with different meanings). Terminology and abbreviations in one domain might be meaningless in another domain—we shouldn't expect that medical recordkeeping and computer repair records would share terms in common, and in the worst case they would conflict.

Because text is intended for communication between people, *context* is important, much more so than with other forms of data. Consider this movie review excerpt:

> "The first part of this movie is far better than the second. The acting is poor and it gets out-of-control by the end, with the violence overdone and an incredible ending, but it's still fun to watch."

Consider whether the overall sentiment is for or against the film. Is the word *incredible* positive or negative? It is difficult to evaluate any particular word or phrase here without taking into account the entire context.

For these reasons, text must undergo a good amount of preprocessing before it can be used as input to a data mining algorithm. Usually the more complex the featurization, the more aspects of the text problem can be included. This chapter can only describe some of the basic methods involved in preparing text for data mining. The next few subsections describe these steps.

Representation

Having discussed how difficult text can be, let's go through the basic steps to transform a body of text into a set of data that can be fed into a data mining algorithm. The general strategy in text mining is to use the simplest (least expensive) technique that works. Nevertheless, these ideas are the key technology underlying much of web search, like Google and Bing. A later example will demonstrate basic query retrieval.

First, some basic terminology. Most of this is borrowed from the field of Information Retrieval (IR). A *document* is one piece of text, no matter how large or small. A document could be a single sentence or a 100 page report, or anything in between, such as a YouTube comment or a blog posting. Typically, all the text of a document is considered together and is retrieved as a single item when matched or categorized. A document is composed of individual *tokens* or *terms*. For now, think of a token or term as just a word; as we go on we'll show how they can be different from what are customarily thought of as words. A collection of documents is called a *corpus*.[1]

1. Latin for "body." The plural is *corpora*.

Bag of Words

It is important to keep in mind the purpose of the text representation task. In essence, we are taking a set of documents—each of which is a relatively free-form sequence of words—and turning it into our familiar feature-vector form. Each document is one instance but we don't know in advance what the features will be.

The approach we introduce first is called "bag of words." As the name implies, the approach is to treat every document as just a collection of individual words. This approach ignores grammar, word order, sentence structure, and (usually) punctuation. It treats every word in a document as a potentially important keyword of the document. The representation is straightforward and inexpensive to generate, and tends to work well for many tasks.

Note: Sets and bags

The terms *set* and *bag* have specific meanings in mathematics, neither of which we exactly mean here. A set allows only one instance of each item, whereas we want to take into account the number of occurrences of words. In mathematics a *bag* is a *multiset*, where members are allowed to appear more than once. The bag-of-words representation initially treats documents as bags—multisets—of words, thereby ignoring word order and other linguistic structure. However, the representation used for mining the text often is more complex than just counting the number of occurrences, as we will describe.

So if every word is a possible feature, what will be the feature's value in a given document? There are several approaches to this. In the most basic approach, each word is a token, and each document is represented by a one (if the token is present in the document) or a zero (the token is not present in the document). This approach simply reduces a document to the set of words contained in it.

Term Frequency

The next step up is to use the word count (frequency) in the document instead of just a zero or one. This allows us to differentiate between how many times a word is used; in some applications, the importance of a term in a document should increase with the number of times that term occurs. This is called the *term frequency* representation. Consider the three very simple sentences (documents) shown in Table 10-1.

Table 10-1. Three simple documents.

d1	jazz music has a swing rhythm
d2	swing is hard to explain
d3	swing rhythm is a natural rhythm

Each sentence is considered a separate document. A simple bag-of-words approach using term frequency would produce a table of term counts shown in Table 10-2.

Table 10-2. Term count representation.

	a	explain	hard	has	is	jazz	music	natural	rhythm	swing	to
d1	1	0	0	1	0	1	1	0	1	1	0
d2	0	1	1	0	1	0	0	0	0	1	1
d3	1	0	0	0	1	0	0	1	2	1	0

Usually some basic processing is performed on the words before putting them into the table. Consider this more complex sample document:

> Microsoft Corp and Skype Global today announced that they have entered into a definitive agreement under which Microsoft will acquire Skype, the leading Internet communications company, for $8.5 billion in cash from the investor group led by Silver Lake. The agreement has been approved by the boards of directors of both Microsoft and Skype.

Table 10-3 shows a reduction of this document to a term frequency representation.

Table 10-3. Terms after normalization and stemming, ordered by frequency

Term	Count	Term	Count	Term	Count	Term	Count
skype	3	microsoft	3	agreement	2	global	1
approv	1	announc	1	acquir	1	lead	1
definit	1	lake	1	communic	1	internet	1
board	1	led	1	director	1	corp	1
compani	1	investor	1	silver	1	billion	1

To create this table from the sample document, the following steps have been performed:

- First, the case has been normalized: every term is in lowercase. This is so that words like Skype and SKYPE are counted as the same thing. Case variations are so common (consider iPhone, iphone, and IPHONE) that case normalization is usually necessary.
- Second, many words have been *stemmed*: their suffixes removed, so that verbs like *announces*, *announced* and *announcing* are all reduced to the term `announc`. Similarly, stemming transforms noun plurals to the singular forms, which is why *directors* in the text becomes `director` in the term list.
- Finally, *stopwords* have been removed. A stopword is a very common word in English (or whatever language is being parsed). The words *the*, *and*, *of*, and *on* are considered stopwords in English so they are typically removed.

Note that the "$8.5" in the story has been discarded entirely. Should it have been? Numbers are commonly regarded as unimportant details for text processing, but the

purpose of the representation should decide this. You can imagine contexts where terms like "4TB" and "1Q13" would be meaningless, and others where they could be critical modifiers.

Note: Careless Stopword Elimination

A word of caution: stopword elimination is not always a good idea. In titles, for example, common words may be very significant. For example, *The Road*, Cormac McCarthy's story of a father and son surviving in a post-apocalyptic world, is very different from John Kerouac's famous novel *On the Road*— though careless stopword removal may cause them to be represented identically. Similarly, the recent movie thriller *Stoker* should not be confused with the 1935 film comedy *The Stoker*.[2]

Table 10-3 shows raw counts of terms. Instead of raw counts, some systems perform a step of normalizing the term frequencies with respect to document length. The purpose of term frequency is to represent the relevance of a term to a document. Long documents usually will have more words—and thus more word occurrences—than shorter ones. This doesn't mean that the longer document is necessarily more important or relevant than the shorter one. In order to adjust for document length, the raw term frequencies are normalized in some way, such as by dividing each by the total number of words in the document.

Measuring Sparseness: Inverse Document Frequency

So term *frequency* measures how prevalent a term is in a single document. We may also care, when deciding the weight of a term, how common it is in the entire corpus we're mining. There are two opposing considerations.

First, a term should not be too *rare*. For example, say the unusual word *prehensile* occurs in only one document in your corpus. Is it an important term? This may depend on the application. For retrieval, the term may be important since a user may be looking for that exact word. For clustering, there is no point keeping a term that occurs only once: it will never be the basis of a meaningful cluster. For this reason, text processing systems usually impose a small (arbitrary) lower limit on the number of documents in which a term must occur.

Another, opposite consideration is that a term should not be too *common*. A term occurring in every document isn't useful for classification (it doesn't distinguish anything) and it can't serve as the basis for a cluster (the entire corpus would cluster together).

2. Both of these examples appeared in recent search results on the film review site of a popular search engine. Not everyone is careful with stopword elimination.

Overly common terms are typically eliminated. One way to do this is to impose an arbitrary upper limit on the number (or fraction) of documents in which a word may occur.

In addition to imposing upper and lower limits on term frequency, many systems take into account the distribution of the term over a corpus as well. The fewer documents in which a term occurs, the more significant it likely is to be to the documents is does occur in. This sparseness of a term *t* is measured commonly by an equation called *inverse document frequency* (IDF), which is shown in Equation 10-1.

Equation 10-1. Inverse Document Frequency (IDF) of a term

$$\text{IDF}(t) = 1 + \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing } t}\right)$$

IDF may be thought of as the boost a term gets for being rare. Figure 10-1 shows a graph of IDF(t) as a function of the number of documents in which t occurs, in a corpus of 100 documents. As you can see, when a term is very rare (far left) the IDF is quite high. It decreases quickly as t becomes more common in documents, and asymptotes at 1.0. Most stopwords, due to their prevalence, will have an IDF near one.

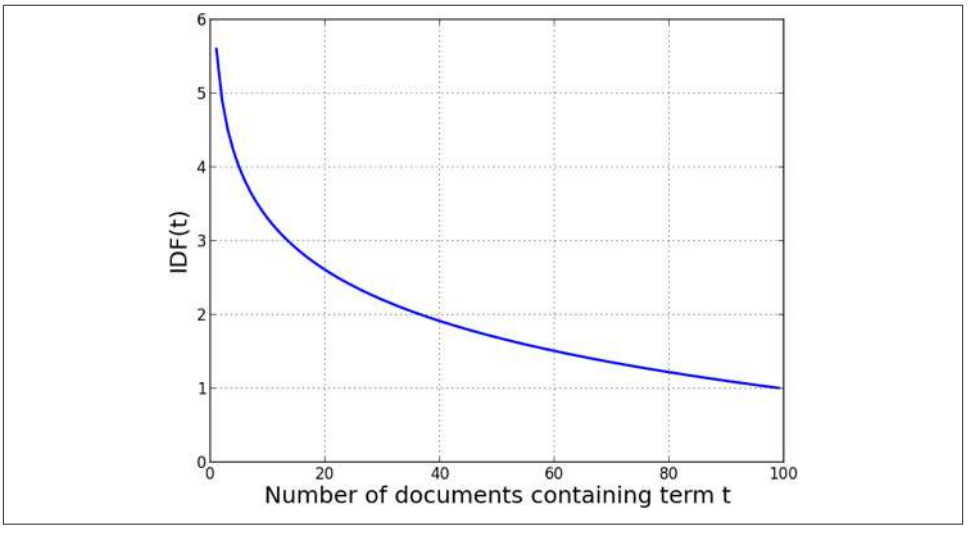

Figure 10-1. IDF of a term t within a corpus of 100 documents.

Combining Them: TFIDF

A very popular representation for text is the product of Term Frequency (TF) and Inverse Document Frequency (IDF), commonly referred to as TFIDF. The TFIDF value of a term t in a given document d is thus:

$$\text{TFIDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

Note that the TFIDF value is specific to a single document (d) whereas IDF depends on the entire corpus. Systems employing the bag-of-words representation typically go through steps of stemming and stopword elimination before doing term counts. Term counts within the documents form the TF values for each term, and the document counts across the corpus form the IDF values.

Each document thus becomes a feature vector, and the corpus is the set of these feature vectors. This set can then be used in a data mining algorithm for classification, clustering, or retrieval.

Because there are very many potential terms with text representation, feature selection is often employed. Systems do this in various ways, such as imposing minimum and maximum thresholds of term counts, and/or using a measure such as information gain[3] to rank the terms by importance so that low-gain terms can be culled.

The bag-of-words text representation approach treats every word in a document as an independent potential keyword (feature) of the document, then assigns values to each document based on frequency and rarity. TFIDF is a very common value representation for terms, but it is not necessarily optimal. If someone describes mining a text corpus using bag of words it just means they're treating each word individually as a feature. Their values could be binary, term frequency, or TFIDF, with normalization or without. Data scientists develop intuitions about how best to attack a given text problem, but they'll typically experiment with different representations to see which produces the best results.

Example: Jazz Musicians

Having introduced a few basic concepts, let's now illustrate them with a concrete example: representing jazz musicians. Specifically, we're going to look at a small corpus of 15 prominent jazz musicians and excerpts of their biographies from Wikipedia. Here are excerpts from a few jazz musician biographies:

3. See "Example: Attribute Selection with Information Gain" on page 56.

Charlie Parker

> Charles "Charlie" Parker, Jr., was an American jazz saxophonist and composer. Miles Davis once said, "You can tell the history of jazz in four words: Louis Armstrong. Charlie Parker." Parker acquired the nickname "Yardbird" early in his career and the shortened form, "Bird," which continued to be used for the rest of his life, inspired the titles of a number of Parker compositions, […]

Duke Ellington

> Edward Kennedy "Duke" Ellington was an American composer, pianist, and big-band leader. Ellington wrote over 1,000 compositions. In the opinion of Bob Blumenthal of *The Boston Globe*, "in the century since his birth, there has been no greater composer, American or otherwise, than Edward Kennedy Ellington." A major figure in the history of jazz, Ellington's music stretched into various other genres, including blues, gospel, film scores, popular, and classical.[…]

Miles Davis

> Miles Dewey Davis III was an American jazz musician, trumpeter, bandleader, and composer. Widely considered one of the most influential musicians of the 20th century, Miles Davis was, with his musical groups, at the forefront of several major developments in jazz music, including bebop, cool jazz, hard bop, modal jazz, and jazz fusion.[…]

Even with this fairly small corpus of fifteen documents, the corpus and its vocabulary are too large to show here (nearly 2,000 features after stemming and stopword removal) so we can only illustrate with a sample. Consider the sample phrase *"Famous jazz saxophonist born in Kansas who played bebop and latin."* We could imagine it being typed as a query to a search engine. How would it be represented? It is treated and processed just like a document, and goes through many of the same steps.

First, basic stemming is applied. Stemming methods are not perfect, and can produce terms like `kansa` and `famou` from "Kansas" and "famous." Stemming perfection usually isn't important as long as it's consistent among all the documents. The result is shown in Figure 10-2.

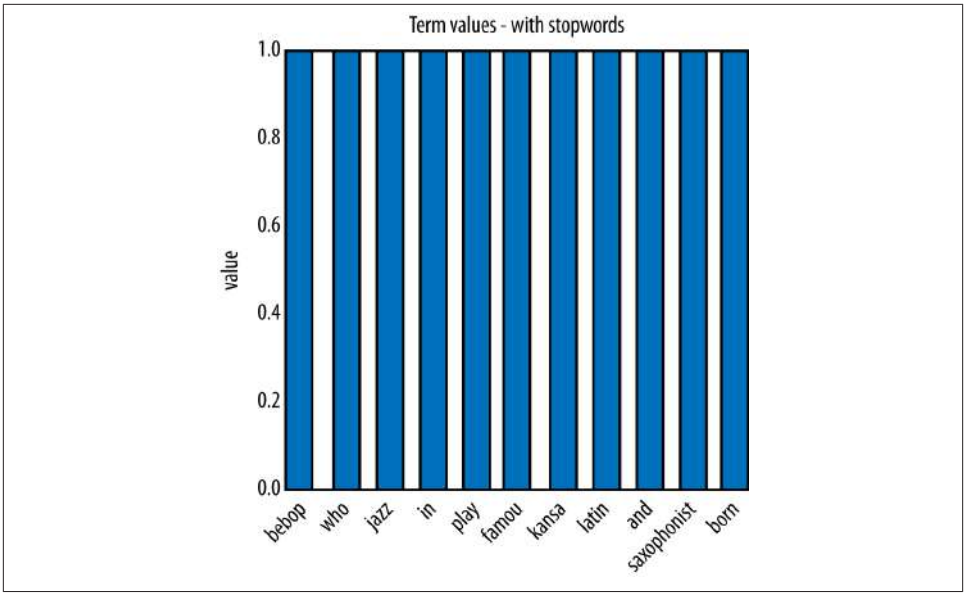

Figure 10-2. Representation of the query "Famous jazz saxophonist born in Kansas who played bebop and latin" after stemming.

Next, stopwords (`in` and `and`) are removed, and the words are normalized with respect to document length. The result is shown in Figure 10-3.

These values would typically be used as the Term Frequency (TF) feature values if we were to stop here. Instead, we'll generate the full TFIDF representation by multiplying each term's TF value by its IDF value. As we said, this boosts words that are rare.

Jazz and *play* are very frequent in this corpus of jazz musician biographies so they get no boost from IDF. They are almost stopwords in this corpus.

The terms with the highest TFIDF values ("latin," "famous," and "kansas") are the rarest in this corpus so they end up with the highest weights among the terms in the query. Finally, the terms are renormalized, producing the final TFIDF weights shown in Figure 10-4. This is the feature vector representation of this sample "document" (the query).

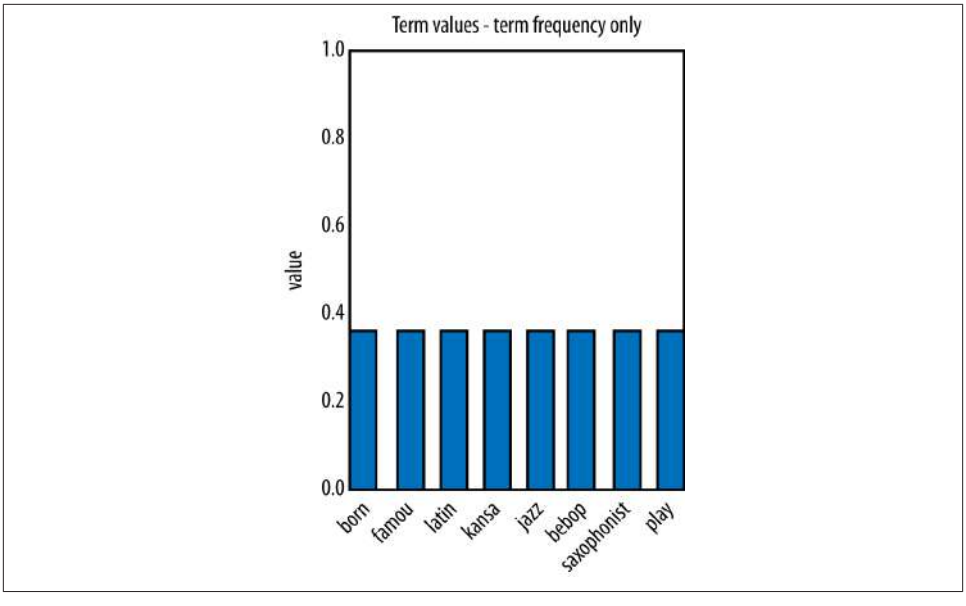

Figure 10-3. Representation of the query "Famous jazz saxophonist born in Kansas who played bebop and latin" after stopword removal and term frequency normalization.

Having shown how this small "document" would be represented, let's use it for something. Recall in Chapter 6, we discussed doing nearest-neighbor retrievals by employing a distance metric, and we showed how similar whiskies could be retrieved. We can do the same thing here. Assume our sample phrase *"Famous jazz saxophonist born in Kansas who played bebop and latin"* was a search query typed by a user and we were implementing a simple search engine. How might it work? First, we would translate the query to its TFIDF representation, as shown graphically in Figure 10-4. We've already computed TFIDF representations of each of our jazz musician biography documents. Now all we need to do is to compute the similarity of our query term to each musician's biography and choose the closest one!

For doing this matching, we'll use the Cosine Similarity function (Equation 6-5) discussed back in the starred section "* Other Distance Functions" on page 158. Cosine similarity is commonly used in text classification to measure the distance between documents.

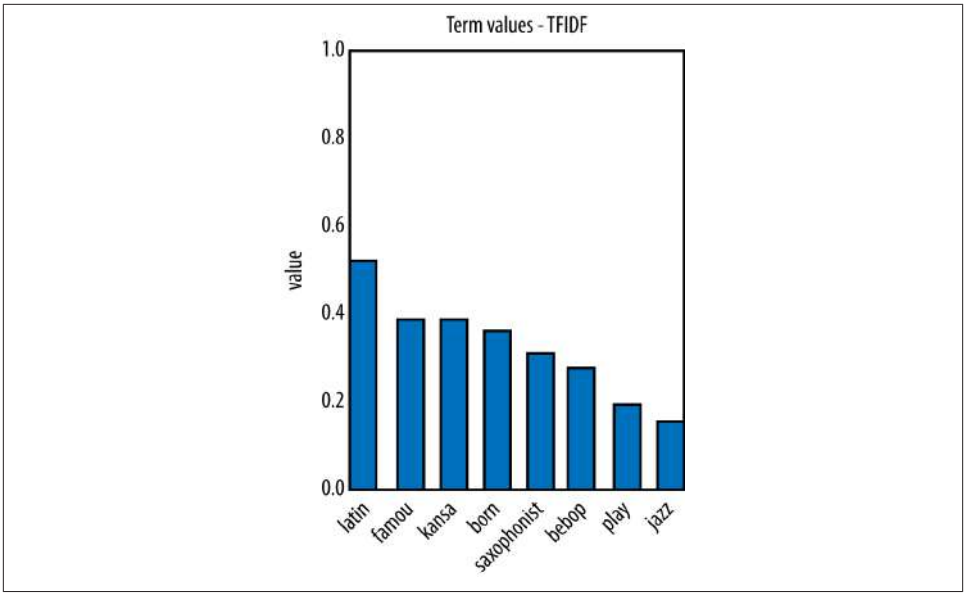

Figure 10-4. Final TFIDF representation of the query "Famous jazz saxophonist born in Kansas who played bebop and latin."

Table 10-4. Similarity of each musician's text to the query 'Famous jazz saxophonist born in Kansas who played bebop and latin,' ordered by decreasing similarity.

Musician	Similarity	Musician	Similarity
Charlie Parker	0.135	Count Basie	0.119
Dizzie Gillespie	0.086	John Coltrane	0.079
Art Tatum	0.050	Miles Davis	0.050
Clark Terry	0.047	Sun Ra	0.030
Dave Brubeck	0.027	Nina Simone	0.026
Thelonius Monk	0.025	Fats Waller	0.020
Charles Mingus	0.019	Duke Ellington	0.017
Benny Goodman	0.016	Louis Armstrong	0.012

As you can see, the closest matching document is Charlie Parker—who was, in fact, a saxophonist born in Kansas and who played the bebop style of jazz. He sometimes combined other genres, including Latin, a fact that is mentioned in his biography.

* The Relationship of IDF to Entropy

Back in "Selecting Informative Attributes" on page 49, we introduced the entropy measure when we began discussing predictive modeling. The curious reader (with a long memory) may notice that Inverse Document Frequency and entropy are somewhat similar—they both seem to measure how "mixed" a set is with respect to a property. Is there any connection between the two? Maybe they're the same? They are not identical, but they are related, and this section will show the relationship. If you're not curious about this you can skip this section.

Figure 10-5 shows some graphs related to the equations we're going to talk about. To begin, consider a term t in a document set. What is the probability that a term t occurs in a document set? We can estimate it as:

$$p(t) = \frac{\text{Number of documents containing } t}{\text{Total number of documents}}$$

To simplify things, from here on we'll refer to this estimate $p(t)$ simply as p. Recall that the definition of IDF of some term t is:

$$\text{IDF}(t) = 1 + \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing } t}\right)$$

The **1** is just a constant so let's discard it. We then notice that IDF(t) is basically $log(1/p)$. You may recall from algebra that $log(1/p)$ is equal to $-log(p)$.

Consider again the document set with respect to a term t. Each document either contains t (with probability p) or does not contain it (with probability $1-p$). Let's create a pseudo, mirror-image term *not_t* that, by definition, occurs in every document that does *not* contain t. What's the IDF of this new term? It is:

$$\text{IDF}(not\_T) = \log 1/(1-p) = -\log(1-p)$$

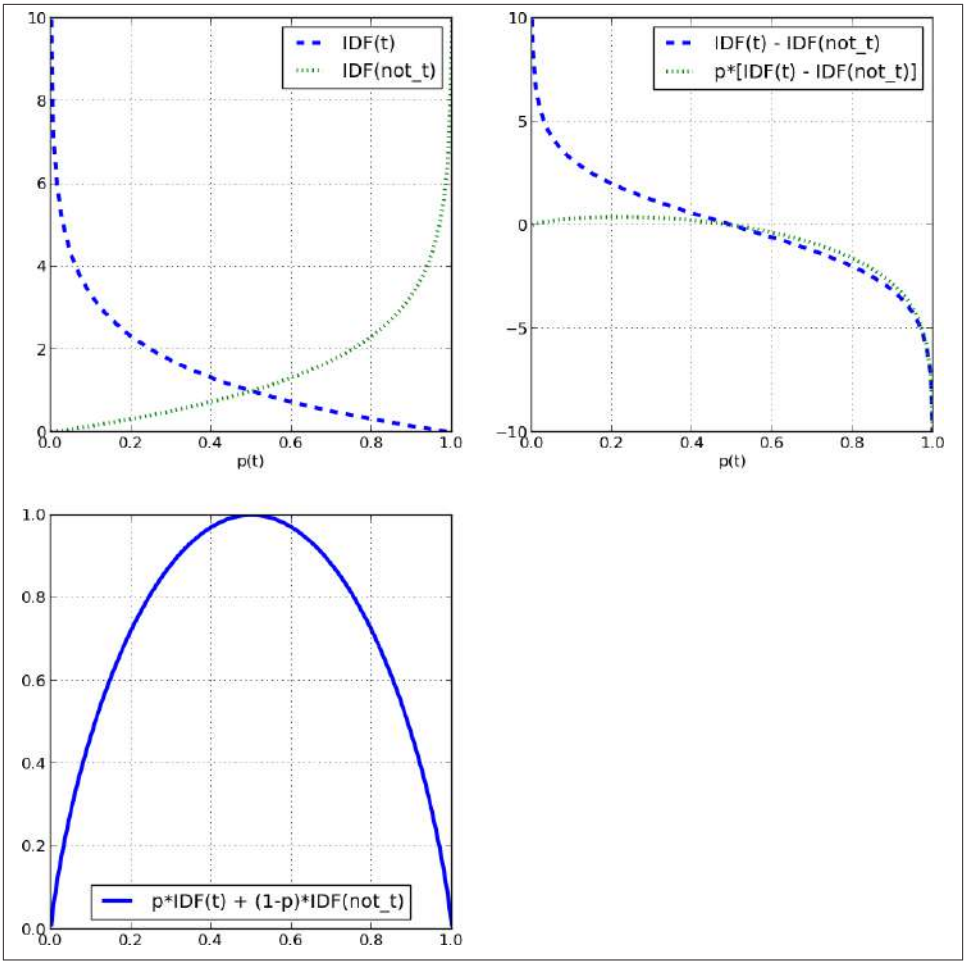

Figure 10-5. Plots of various values related to IDF(t) and IDF(not_t).

See the upper left graph of Figure 10-5. The two graphs are mirror images of each other, as we might expect. Now recall the definition of entropy from Equation 3-1. For a binary term where $p_2=1-p_1$, the entropy becomes:

$$\text{entropy} = -p_1 \log(p_1) - p_2 \log(p_2)$$

In our case, we have a binary term t that either occurs (with probability p) or does not (with probability $1-p$). So the definition of entropy of a set partitioned by t reduces to:

$$\text{entropy}(t) = -p \log(p) - (1-p) \log(1-p)$$

Now, given our definitions of IDF(t) and IDF(not_t), we can start substituting and simplifying (for reference, various of these subexpressions are plotted in the top right graph of Figure 10-5).

$$\begin{aligned} \text{entropy}(t) &= -p \log(p) - (1 - p) \log(1 - p) \\ &= p \cdot \text{IDF}(t) - (1 - p)[-\text{IDF}(not\_t)] \\ &= p \cdot \text{IDF}(t) + (1 - p)[\text{IDF}(not\_t)] \end{aligned}$$

Note that this is now in the form of an *expected value* calculation! We can express entropy as the expected value of IDF(t) and IDF(not_t) based on the probability of its occurrence in the corpus. Its graph at the bottom left in Figure 10-5 does match the entropy curve of Figure 3-3 back in Chapter 3.

Beyond Bag of Words

The basic bag of words approach is relatively simple and has much to recommend it. It requires no sophisticated parsing ability or other linguistic analysis. It performs surprisingly well on a variety of tasks, and is usually the first choice of data scientists for a new text mining problem.

Still, there are applications for which bag of words representation isn't good enough and more sophisticated techniques must be brought to bear. Here we briefly discuss a few of them.

N-gram Sequences

As presented, the bag-of-words representation treats every individual word as a term, discarding word order entirely. In some cases, word order is important and you want to preserve some information about it in the representation. A next step up in complexity is to include *sequences* of adjacent words as terms. For example, we could include pairs of adjacent words so that if a document contained the sentence *"The quick brown fox jumps."* it would be transformed into the set of its constitutent words {`quick`, `brown`, `fox`, `jumps`}, plus the tokens `quick_brown`, `brown_fox`, and `fox_jumps`.

This general representation tactic is called *n-grams*. Adjacent pairs are commonly called bi-grams. If you hear a data scientist mention representing text as "bag of n-grams up to three" it simply means she's representing each document using as features its individual words, adjacent word pairs, and adjacent word triples.

N-grams are useful when particular phrases are significant but their component words may not be. In a business news story, the appearance of the tri-gram `exceed_analyst_expectation` is more meaningful than simply knowing that the individual words `analyst`, `expectation`, and `exceed` appeared somewhere in a story. An advantage of

using n-grams is that they are easy to generate; they require no linguistic knowledge or complex parsing algorithm.

The main disadvantage of n-grams is that they greatly increase the size of the feature set. There are far more word pairs than individual words, and still more word triples. The number of features generated can quickly get out of hand. Data mining using n-grams almost always needs some special consideration for dealing with massive numbers of features, such as a feature selection stage or special consideration to computational storage space.

Named Entity Extraction

Sometimes we want still more sophistication in phrase extraction. We want to be able to recognize common named entities in documents. *Silicon Valley*, *New York Mets*, *Department of the Interior*, and *Game of Thrones* are significant phrases. Their component words mean one thing, and may not be significant, but in sequence they name unique entities with interesting identities. The basic bag-of-words (or even n-grams) representation may not capture these, and we'd want a preprocessing component that knows when word sequences constitute proper names.

Many text-processing toolkits include a named entity extractor of some sort. Usually these can process raw text and extract phrases annotated with terms like `person` or `organization`. In some cases normalization is done so that, for example, phrases like "HP," "H-P," and "Hewlett-Packard" all link to some common representation of the Hewlett-Packard Corporation.

Unlike bag of words and n-grams, which are based on segmenting text on whitespace and punctuation, named entity extractors are knowledge intensive. To work well, they have to be trained on a large corpus, or hand coded with extensive knowledge of such names. There is no linguistic principle dictating that the phrase "*oakland raiders*" should refer to the Oakland Raiders professional football team, rather than, say, a group of aggressive California investors. This knowledge has to be learned, or coded by hand. The quality of entity recognition can vary, and some extractors may have particular areas of expertise, such as industry, government, or popular culture.

Topic Models

So far we've dealt with models created directly from words (or named entities) appearing from a document. The resulting model—whatever it may be—refers directly to words. Learning such direct models is relatively efficient, but is not always optimal. Because of the complexity of language and documents, sometimes we want an additional layer between the document and the model. In the context of text we call this the *topic* layer (see Figure 10-6).

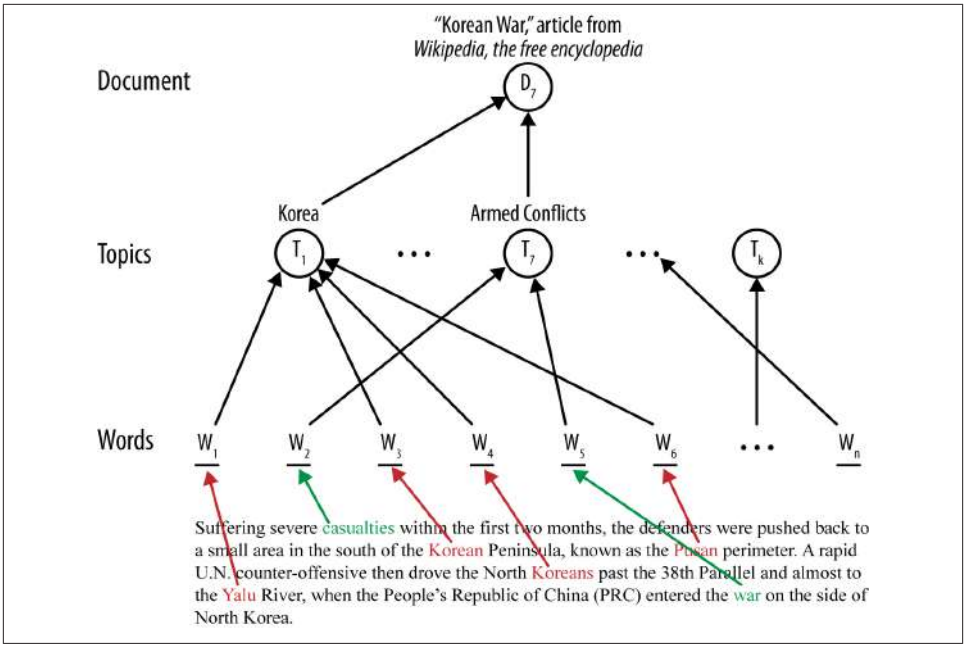

Figure 10-6. Modeling documents with a topic layer.

The main idea of a topic layer is first to model the set of topics in a corpus separately. As before, each document constitutes a sequence of words, but instead of the words being used directly by the final classifier, the words map to one or more topics. The topics also are learned from the data (often via unsupervised data mining). The final classifier is defined in terms of these intermediate topics rather than words. One advantage is that in a search engine, for example, a query can use terms that do not exactly match the specific words of a document; if they map to the correct topic(s), the document will still be considered relevant to the search.

General methods for creating topic models include matrix factorization methods, such as Latent Semantic Indexing and Probabilistic Topic Models, such as Latent Dirichlet Allocation. The math of these approaches is beyond the scope of this book, but we can think of the topic layer as being a clustering of words. In topic modeling, the terms associated with the topic, and any term weights, are *learned* by the topic modeling process. As with clusters, the topics emerge from statistical regularities in the data. As such, they are not necessarily intelligible, and they are not guaranteed to correspond to topics familiar to people, though in many cases they are.

Note: Topics as Latent Information

Topic models are a type of *latent information* model, which we'll discuss a bit more in Chapter 12 (along with a movie recommendation example). You can think of latent information as a type of intermediate, unobserved layer of information inserted between the inputs and outputs. The techniques are essentially the same for finding latent topics in text and for finding latent "taste" dimensions of movie viewers. In the case of text, words map to topics (unobserved) and topics map to documents. This makes the entire model more complex and more expensive to learn, but can yield better performance. In addition, the latent information is often interesting and useful in its own right (as we will see again in the movie recommendation example in Chapter 12).

Example: Mining News Stories to Predict Stock Price Movement

To illustrate some issues in text mining, we introduce a new predictive mining task: we're going to predict stock price fluctuations based on the text of news stories. Roughly speaking, we are going to "predict the stock market" based on the stories that appear on the news wires. This project contains many common elements of text processing and of problem formulation.

The Task

Every trading day there is activity in the stock market. Companies make and announce decisions—mergers, new products, earnings projections, and so forth—and the financial news industry reports on them. Investors read these news stories, possibly change their beliefs about the prospects of the companies involved, and trade stock accordingly. This results in stock price changes. For example, announcements of acquisitions, earnings, regulatory changes, and so on can all affect the price of a stock, either because it directly affects the earnings potential or because it affects what traders think other traders are likely to pay for the stock.

This is a very simplified view of the financial markets, of course, but it's enough to lay out a basic task. We want to predict stock price changes based on financial news. There are many ways we could approach this based on the ultimate purpose of the project. If we wanted to make *trades* based on financial news, ideally we'd like to predict—in advance and with precision—the change in a company's stock price based on the stream of news. In reality there are many complex factors involved in stock price changes, many of which are not conveyed in news stories.

Instead, we'll mine the news stories for a more modest purpose, that of *news recommendation*. From this point of view, there is a huge stream of market news coming in—some interesting, most not. We'd like predictive text mining to recommend interesting news stories that we should pay attention to. What's an interesting story? Here we'll define it as *news that will likely result in a significant change in a stock's price*.

We have to simplify the problem further to make it more tractable (in fact, this task is a good example of problem formulation as much as it is of text mining). Here are some of the problems and simplifying assumptions:

1. It is difficult to predict the effect of news far in advance. With many stocks, news arrives fairly often and the market responds quickly. It is unrealistic, for example, to predict what price a stock will have a week from now based on a news release today. Therefore, we'll try to predict what effect a news story will have on stock price the *same day*.
2. It is difficult to predict exactly what the stock price will be. Instead, we will be satisfied with the *direction* of movement: up, down, or no change. In fact, we'll simplify this further into **change** and **no change**. This works well for our example application: recommending a news story if it looks like it will trigger, or indicate, a subsequent change in the stock price.
3. It is difficult to predict small changes in stock price, so instead we'll predict *relatively large* changes. This will make the signal a bit cleaner at the expense of yielding fewer events. We will deliberately ignore the subtlety of small fluctuations.
4. It is difficult to associate a specific piece of news with a price change. In principle, any piece of news could affect any stock. If we accepted this idea it would leave us with a huge problem of credit assignment: how do you decide which of today's thousands of stories are relevant? We need to narrow the "causal radius."

   We will assume that only news stories mentioning a specific stock will affect that stock's price. This is inaccurate, of course—companies are affected by the actions of their competitors, customers, and clients, and it's rare that a news story will mention all of them. But for a first pass this is an acceptable simplifying assumption.

We still have to nail some of this down. Consider issue two. What is a "relatively large" change? We can (somewhat arbitrarily) place a threshold of 5%. If a stock's price increases by five percent or more, we'll call it a **surge**; if it declines by five percent or more, we'll call it a **plunge**. What if it changes by some amount in between? We could call any value in between **stable**, but that's cutting it a little close—a 4.9% change and a 5% change shouldn't really be distinct classes. Instead, we'll designate some "gray zones" to make the classes more separable (see Figure 10-7). Only if a stock's price stays between 2.5% and −2.5% will it be called **stable**. Otherwise, for the zones between 2.5% to 5% and −2.5% to −5%, we'll refuse to label it.

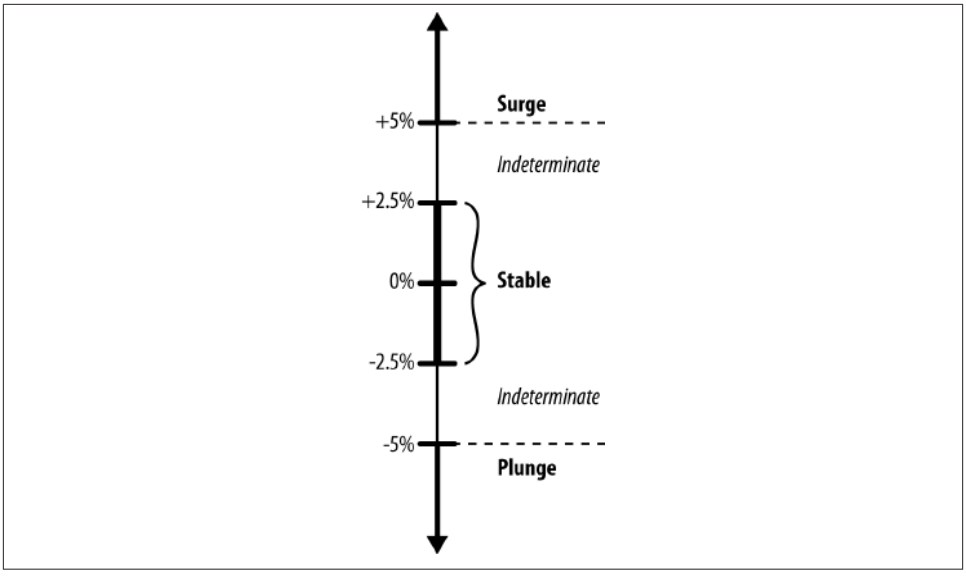

Figure 10-7. Percentage change in price, and corresponding label.

For the purpose of this example, we'll create a two-class problem by merging surge and plunge into a single class, **change**. It will be the positive class, and **stable** (**no change**) will be the negative class.

The Data

The data we'll use comprise two separate time series: the stream of news stories (text documents), and a corresponding stream of daily stock prices. The Internet has many sources of financial data, such as Google Finance and Yahoo! Finance. For example, to see what news stories are available about Apple Computer, Inc., see the corresponding Yahoo! Finance page. Yahoo! aggregates news stories from a variety of sources such as Reuters, PR Web, and Forbes. Historical stock prices can be acquired from many sources, such as Google Finance.

The data to be mined are historical data from 1999 for stocks listed on the New York Stock Exchange and NASDAQ. This data was used in a prior study (Fawcett & Provost, 1999). We have open and close prices for stocks on the major exchanges, and a large compendium of financial news stories throughout the year—nearly 36,000 stories altogether. Here is a sample news story from the corpus:

```
1999-03-30 14:45:00
WALTHAM, Mass.--(BUSINESS WIRE)--March 30, 1999--Summit Technology,
Inc. (NASDAQ:BEAM) and Autonomous Technologies Corporation
(NASDAQ:ATCI) announced today that the Joint Proxy/Prospectus for
Summit's acquisition of Autonomous has been declared effective by the
```

```
Securities and Exchange Commission. Copies of the document have been
mailed to stockholders of both companies. "We are pleased that these
proxy materials have been declared effective and look forward to the
shareholder meetings scheduled for April 29," said Robert Palmisano,
Summit's Chief Executive Officer.
```

As with many text sources, there is a lot of miscellaneous material since it is intended for human readers and not machine parsing (see "Sidebar: The News Is Messy" on page 270 for more details). The story includes the date and time, the news source (Reuters), stock symbols and link (NASDAQ:BEAM), as well as background material not strictly germane to the news. Each such story is tagged with the stock mentioned.

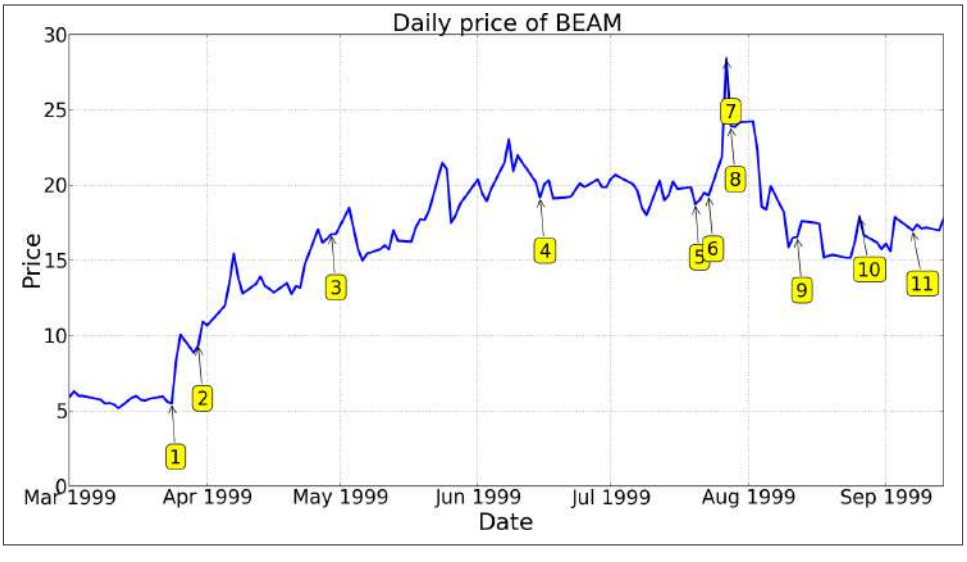

Figure 10-8. Graph of stock price of Summit Technologies, Inc., (NASDAQ:BEAM) annotated with news story summaries.

1. Summit Tech announces revenues for the three months ended Dec 31, 1998 were $22.4 million, an increase of 13%.
2. Summit Tech and Autonomous Technologies Corporation announce that the Joint Proxy/Prospectus for Summit's acquisition of Autonomous has been declared effective by the SEC.
3. Summit Tech said that its procedure volume reached new levels in the first quarter and that it had concluded its acquisition of Autonomous Technologies Corporation.
4. Announcement of annual shareholders meeting.
5. Summit Tech announces it has filed a registration statement with the SEC to sell 4,000,000 shares of its common stock.
6. A US FDA panel backs the use of a Summit Tech laser in LASIK procedures to correct nearsightedness with or without astigmatism.
7. Summit up 1-1/8 at 27-3/8.

8 Summit Tech said today that its revenues for the three months ended June 30, 1999 increased 14%…

9 Summit Tech announces the public offering of 3,500,000 shares of its common stock priced at $16/share.

10 Summit announces an agreement with Sterling Vision, Inc. for the purchase of up to six of Summit's state of the art, Apex Plus Laser Systems.

11 Preferred Capital Markets, Inc. initiates coverage of Summit Technology Inc. with a Strong Buy rating and a 12-16 month price target of $22.50.

Sidebar: The News Is Messy

The financial news corpus is actually far messier than this one story implies, for several reasons.

First, financial news comprises a wide variety of stories, including earnings announcements, analysts' assessments ("We are reiterating our Buy rating on Apple"), market commentary ("Other stocks featured in this morning's MarketMovers include Lycos Inc. and Staples Inc."), SEC filings, financial balance sheets, and so on. Companies are mentioned for many different reasons and a single document ("story") may actually comprise multiple unrelated news blurbs of the day.

Second, stories come in different formats, some with tabular data, some in a multiparagraph "lead stories of the day" format, and so on. Much of the meaning is imparted by context. Our text processing won't pick this up.

Finally, stock tagging is not perfect. It tends to be overly permissive, such that stories are included in the news feed of stocks that were not actually referenced in the story. As an extreme example, American blogger Perez Hilton uses the expression "cray cray" to mean crazy or disgusting, and some of his blog postings end up in the story feed of Cray Computer Corporation.

In short, the relevance of a stock to a document may not be clear without a careful reading. With deep parsing (or at least story segmentation) we could eliminate some of the noise, but with bag of words (or even named entity extraction) we cannot hope to remove all of it.

Figure 10-8 shows the kind of data we have to work with. They are basically two linked time series. At the top is a graph of the stock price of Summit Technologies, Inc., a manufacturer of excimer laser systems for use in laser vision correction. Some points on the graph are annotated with story numbers on the date the story was released. Below the graph are summaries of each story.

Data Preprocessing

As mentioned, we have two streams of data. Each stock has an opening and closing price for the day, measured at 9:30 am EST and 4:00 pm EST, respectively. From these values

we can easily compute a percentage change. There is one minor complication. We're trying to predict stories that produce a substantial change in a stock's value. Many events occur outside of trading hours, and fluctuations near the opening of trading can be erratic. For this reason, instead of measuring the opening price at the opening bell (9:30 am EST) we measure it at 10:00 am, and track the difference between the day's prices at 4 pm and 10 am. Divided by the stock's closing price, this becomes the daily percent change.

The stories require much more care. The stories are pre-tagged with stocks, which are mostly accurate ("Sidebar: The News Is Messy" on page 270 goes into some details on why this is a difficult text mining problem). Almost all stories have timestamps (those without are discarded) so we can align them with the correct day and trading window. Because we want a fairly tight association of a story with the stock(s) it might affect, we reject any stories mentioning more than two stocks. This gets rid of many stories that are just summaries and news aggregations.

The basic steps outlined in "Bag of Words" on page 252 were applied to reduce each story to a TFIDF representation. In particular, each word was case-normalized and stemmed, and stopwords were removed. Finally, we created n-grams up to two, such that every individual term and pair of adjacent terms were used to represent each story.

Subject to this preparation, each story is tagged with a label (**change** or **no change**) based on the associated stock(s) price movement, as depicted in Figure 10-7. This results in about 16,000 usable tagged stories. For reference, the breakdown of stories was about 75% no change, 13% surge, and 12% plunge. The surge and plunge stories were merged to form **change**, so 25% of the stories were followed by a significant price change to the stocks involved, and 75% were not.

Results

Before we dig into results, a short digression.

Previous chapters (particularly Chapter 7) stressed the importance of thinking carefully about the business problem being solved in order to frame the evaluation. With this example we have not done such careful specification. If the purpose of this task were to trigger stock trades, we might propose an overall trading strategy involving thresholds, time limits, and transaction costs, from which we could produce a complete cost-benefit analysis.[4] But the purpose is news recommendation (answering "which stories lead to substantial stock price changes?") and we've left this pretty open, so we won't specify exact costs and benefits of decisions. For this reason, expected value calculations and profit graphs aren't really appropriate here.

4. Some researchers have done this, evaluating their systems by simulating stock trades and calculating the return on investment. See, for example, Schumaker & Chen's (2010) work on AZFinText.

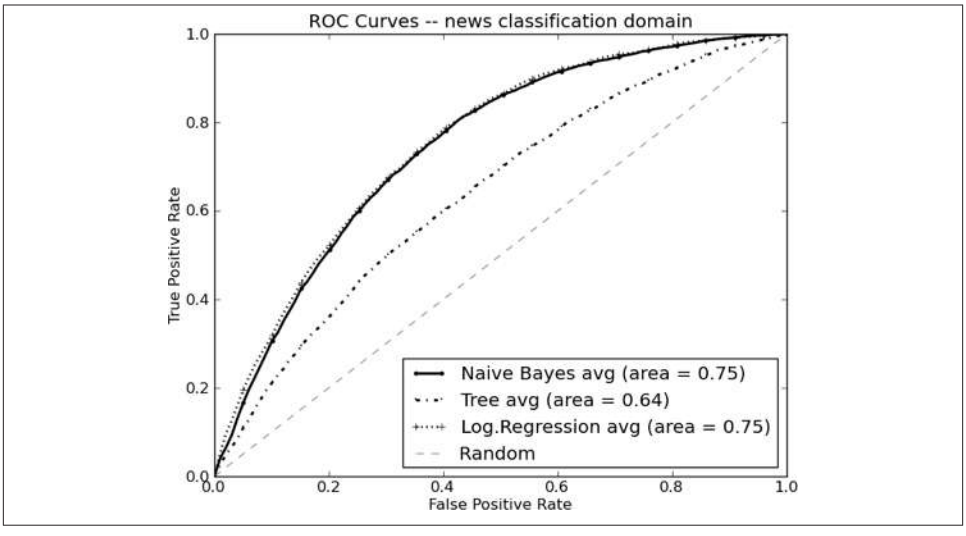

Figure 10-9. ROC curves for the stock news classification task.

Instead, let's look at predictability, just to get a sense of how well this problem can be solved. Figure 10-9 shows the ROC curves of three sample classifiers: Logistic Regression, Naive Bayes, and a Classification Tree. These curves are averaged from ten-fold cross-validation, using **change** as the positive class and **no change** as the negative class. Several things are apparent. First, there is a significant "bowing out" of the curves away from the diagonal (Random) line, and the ROC curve areas (AUCs) are all substantially above 0.5, so there *is* predictive signal in the news stories. Second, logistic regression and Naive Bayes perform similarly, whereas the classification tree (Tree) is considerably worse. Finally, there is no obvious region of superiority (or deformity) in the curves. Bulges or concavities can sometimes reveal characteristics of the problem, or flaws in the data representation, but we see none here.

Figure 10-10 shows the corresponding lift curves of these three classifiers, again averaged from ten-fold cross-validation. Recall that one in four (25%) of the stories in our population is positive, (i.e., it is followed by a significant change in stock price). Each curve shows the lift in precision[5] we would get if we used the model to score and order the news stories. For example, consider the point at x=0.2, where the lifts of Logistic Regression and Naive Bayes are both around 2.0. This means that, if you were to score all the news stories and take the top 20% (x=0.2), you'd have *twice* the precision (lift of two) of finding a positive story in that group than in the population as a whole. Therefore, among the top 20% of the stories as ranked by the model, *half* are significant.

5. Recall from Chapter 7, precision is the percentage of the cases that are above the classification threshold that are actually positive examples, and the lift is how many times more this is than you would expect by chance.

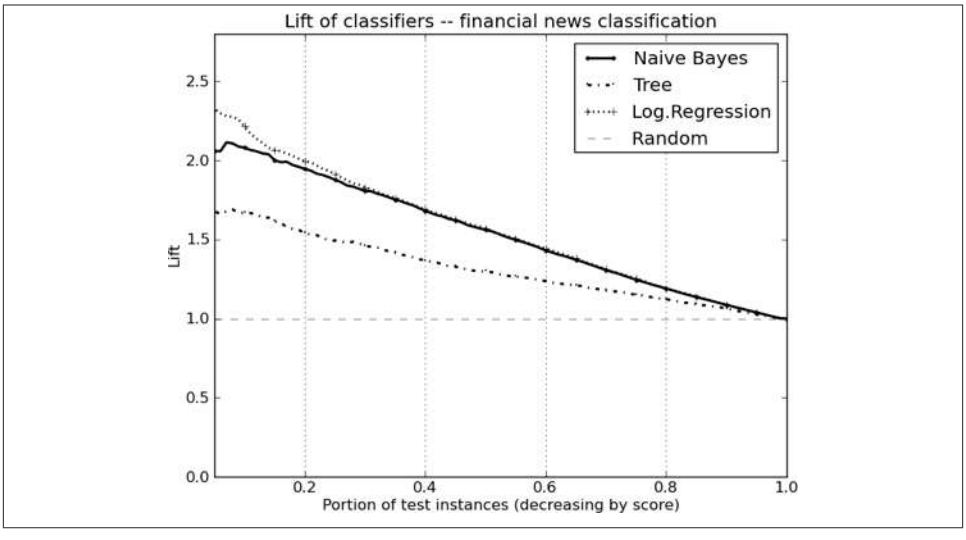

Figure 10-10. Lift curves for the stock news prediction task.

Before concluding this example, let's look at some of the important terms found from this task. The goal of this example was not to create intelligible rules from the data, but prior work on the same corpus by Macskassy et al. (2001) did just that. Here is a list of terms with high information gain[6] taken from their work. Each boldface term is either a word or a stem followed by suffixes in parentheses:

```
alert(s,ed), architecture, auction(s,ed,ing,eers), average(s,d), award(s,ed),
bond(s), brokerage, climb(ed,s,ing), close(d,s), comment(ator,ed,ing,s),
commerce(s), corporate, crack(s,ed,ing), cumulative, deal(s), dealing(s),
deflect(ed,ing), delays, depart(s,ed), department(s), design(ers,ing),
economy, econtent, edesign, eoperate, esource, event(s), exchange(s),
extens(ion,ive), facilit(y,ies), gain(ed,s,ing), higher, hit(s), imbalance(s),
index, issue(s,d), late(ly), law(s,ful), lead(s,ing), legal(ity,ly), lose,
majority, merg(ing,ed,es), move(s,d), online, outperform(s,ance,ed),
partner(s), payments, percent, pharmaceutical(s), price(d), primary,
recover(ed,s), redirect(ed,ion), stakeholder(s), stock(s), violat(ing,ion,ors)
```

Many of these are suggestive of significant announcements of good or bad news for a company or its stock price. Some of them (`econtent`, `edesign`, `eoperate`) are also suggestive of the "Dotcom Boom" of the late 1990s, from which this corpus is taken, when the e- prefix was in vogue.

Though this example is one of the most complex presented in this book, it is still a fairly simple approach to mining financial news stories. There are many ways this project

6. Recall Chapter 3.

could be extended and refined. The bag-of-words representation is primitive for this task; named entity recognition could be used to better extract the names of the companies and people involved. Better still, event parsing should provide real leverage, since news stories usually report events rather than static facts about companies. It is not clear from individual words who are the subjects and objects of the events, and important modifiers like *not*, *despite*, and *expect* may not be adjacent to the phrases they modify, so the bag of words representation is at a disdvantage. Finally, to calculate price changes we considered only daily opening and closing stock prices, rather than hourly or instantaneous ("tick level") price changes. The market responds quickly to news, and if we wanted to trade on the information we'd need to have fine-grained, reliable timestamps on both stock prices and news stories.

Sidebar: Prior Work on Predicting Stock Prices from Financial News

The problem of relating financial news stories to market activity has been tackled by many people in the past 15 years or so. Your authors even did some early work on the task (Fawcett & Provost, 1999). Most of the prior work has been published outside the data mining literature, so the data mining community may remain largely unaware of the task and the work. We mention a few articles here for anyone interested in pursuing the topic.

A survey by Mittermayer and Knolmayer (2006) is a good place to start, though it is a bit dated by now. It provides a good overview of approaches up to that point.

Most researchers view the problem as predicting the stock market from news. In this chapter, we've taken an inverse view as that of recommending news stories based on their future effects. This task was termed *information triage* by Macskassy et al. (2001).

Early work looked at the effect of financial news in the mainstream media. Later work takes into account opinions and sentiment from other sources on the Internet, such as Twitter updates, blog postings, and search engine trends. A paper by Mao et al. (2011) provides a good analysis and comparison of the effect of these additional sources.

Finally, though it's not text mining per se, let us mention the paper "Legislating Stock Prices" by Cohen, Diether, and Malloy (2012). These researchers examined the relationship of politicians, legislation, and firms affected by the legislation. Obviously, these three groups are interrelated and should affect each other, but surprisingly, the relationship had not been exploited by Wall Street. From publicly available data the researchers discovered a "simple, yet previously undetected impact on firm stock prices" that they report to be able to trade upon profitably. This suggests that there are undiscovered relationships remaining to be mined.

Summary

Our problems do not always present us with data in a neat feature vector representation that most data mining methods take as input. Real-world problems often require some form of data representation engineering to make them amenable to mining. Generally it is simpler to first try to engineer the data to match existing tools. Data in the form of text, images, sound, video, and spatial information usually require special preprocessing —and sometimes special knowledge on the part of the data science team.

In this chapter, we discussed one especially prevalent type of data that requires preprocessing: text. A common way to turn text into a feature vector is to break each document into individual words (its "bag of words" representation), and assign values to each term using the TFIDF formula. This approach is relatively simple, inexpensive and versatile, and requires little knowledge of the domain, at least initially. In spite of its simplicity, it performs surprisingly well on a variety of tasks. In fact, we will revisit these ideas on a completely different, nontext task in Chapter 14.

CHAPTER 11

Decision Analytic Thinking II: Toward Analytical Engineering

Fundamental concept: *Solving business problems with data science starts with analytical engineering: designing an analytical solution, based on the data, tools, and techniques available.*

Exemplary technique: *Expected value as a framework for data science solution design.*

Ultimately, data science is about extracting information or knowledge from data, based on principled techniques. However, as we've discussed throughout the book, seldom does the world provide us with important business problems perfectly aligned with these techniques, or with data represented such that the techniques can be applied directly. Ironically, this fact often is better accepted by the business users (for whom it is often obvious) than by entry-level data scientists—because academic programs in statistics, machine learning, and data mining often present students with problems ready for the application of the tools that the programs teach.

Reality is much messier. Business problems rarely are classification problems or regression problems or clustering problems. They're just business problems. Recall the mini-cycle in the first stages of the data mining process, where we focus on business understanding and data understanding. In these stages we must *design* or *engineer* a solution to the business problem. As with engineering more broadly, the data science team considers the needs of the business as well as the tools that might be brought to bear to solve the problem.

In this chapter, we will illustrate such *analytical engineering* with two case studies. In these case studies, we will see the application of the fundamental principles presented throughout the book, as well as some of the specific techniques that we have introduced. One common theme that runs through these case studies is how our expected value framework (recall from Chapter 7) helps to decompose each of the business problems into subproblems, such that the subproblems can be attacked with tried-and-true data

science techniques. Then the expected value framework guides the recombination of the results into a solution to the original problem.

Targeting the Best Prospects for a Charity Mailing

A classic business problem for applying data science principles and techniques is targeted marketing. Targeted marketing makes for a perfect case study for two reasons. First, a very large number of businesses have problems that look similar to targeted marketing problems—traditional targeted (database) marketing, customer-specific coupon offers, online ad targeting, and so on. Second, the fundamental structure of the problem occurs in many other problems as well, such as our running example problem of churn management.

For this case study, let's consider a real example of targeted marketing: targeting the best prospects for a charity mailing. Fundraising organizations (including those in universities) need to manage their budgets and the patience of their potential donors. In any given campaign segment, they would like to solicit from a "good" subset of the donors. This could be a very large subset for an inexpensive, infrequent campaign, or a smaller subset for a focused campaign that includes a not-so-inexpensive incentive package.

The Expected Value Framework: Decomposing the Business Problem and Recomposing the Solution Pieces

We would like to "engineer" an analytic solution to the problem, and our fundamental concepts will provide the structure to do so. To frame our data-analytic thinking, we begin by using the data-mining process (Chapter 2) to provide structure to the overall analysis: we start with business and data understanding. More specifically, we need to focus using one of our fundamental principles: what exactly is the business problem that we would like to solve (Chapter 7)?

So let's get specific. A data miner might immediately think: we want to model the probability that each prospective customer, a prospective donor in this case, will respond to the offer. However, thinking carefully about the business problem we realize that in this case, the response can vary—some people might donate $100 while others might donate $1. We need to take this into account.

Would we like to maximize the total amount of donations? (The amount could be either in this particular campaign or over the lifetime of the donor prospects; let's assume the first for simplicity.) What if we did that by targeting a massive number of people, and these each give just $1, and our costs are about $1 per person? We would make almost no money. So let's revise our thinking.

Focusing on the business problem that we want to solve may have given us our answer right away, because to a business-savvy person it may seem rather obvious: we would like to maximize our donation *profit*—meaning the net after taking into account the

costs. However, while we have methods for estimating the probability of response (that's a clear application of class probability estimation over a binary outcome), it is not clear that we have methods to estimate profit.

Again, our fundamental concepts allow us to structure our thinking and engineer a data-analytic solution. Applying another one of our fundamental notions, we can structure this data analysis using the framework of expected value. We can apply the concepts introduced in Chapter 7 to our problem formulation: we can use expected value as a framework for structuring our approach to engineering a solution to the problem. Recall our formulation of the expected benefit (or cost) of targeting consumer $\mathbf{x}$:

$$\text{Expected benefit of targeting} = p(R \mid \mathbf{x}) \cdot v_R + [1 - p(R \mid \mathbf{x})] \cdot v_{NR}$$

where $p(R \mid \mathbf{x})$ is the probability of response given consumer $\mathbf{x}$, v_R is the value we get from a response, and v_{NR} is the value we get from no response. Since everyone either responds or does not, our estimate of the probability of not responding is just $(1 - p(R \mid \mathbf{x}))$. As we discussed in Chapter 7, we can model the probabilities by mining historical data using one of the many techniques discussed through the book.

However, the expected value framework helps us realize that this business problem is slightly different from problems we have considered up to this point. In this case, the value varies from consumer to consumer, and we do not know the value of the donation that any particular consumer will give until after she is targeted! Let's modify our formulation to make this explicit:

$$\text{Expected benefit of targeting} = p(R \mid \mathbf{x}) \cdot v_R(\mathbf{x}) + [1 - p(R \mid \mathbf{x})] \cdot v_{NR}(\mathbf{x})$$

where $v_R(\mathbf{x})$ is the value we get from a response from consumer $\mathbf{x}$ and $v_{NR}(\mathbf{x})$ is the value we get if consumer $\mathbf{x}$ does not respond. The value of a response, $v_R(\mathbf{x})$, would be the consumer's donation minus the cost of the solicitation. The value of no response, $v_{NR}(\mathbf{x})$, in this application would be zero minus the cost of the solicitation. To be complete, we also want to estimate the benefit of *not* targeting, and then compare the two to make the decision of whether to target or not. The expected benefit of not targeting is simply zero—in this application, we do not expect consumers to donate spontaneously without a solicitation. That may not always be the case, but let's assume it is here.

Why exactly does the expected value framework help us? Because we may be able to estimate $v_R(\mathbf{x})$ and/or $v_{NR}(\mathbf{x})$ from the data as well. Regression modeling estimates such values. Looking at historical data on consumers who have been targeted, we can use regression modeling to estimate how much a consumer will respond. Moreover, the expected value framework gives us even more precise direction: $v_R(\mathbf{x})$ is the value we would predict to get *if a consumer were to respond* — this would be estimated using a model trained only on consumers who have responded. This turns out to be a more

useful prediction problem than the problem of estimating the response from a targeted consumer generally, because in this application the vast majority of consumers do not respond at all, and so the regression modeling would need somehow to differentiate between the cases where the value is zero because of non-response or the value is small because of the characteristics of the consumer.

Stepping back for a moment, this example illustrates why the expected value framework is so useful for decomposing business problems: as discussed in Chapter 7, the expected value is a summation of products of probabilities and values, and data science gives us methods to estimate both probabilities and values. To be clear, we may not need to estimate some of these quantities (like $v_{NR}(\mathbf{x})$, which we assume in this example is always zero), and estimating them well may be a nontrivial undertaking. The point is that the expected value framework provides a helpful decomposition of possibly complicated business problems into subproblems that we understand better how to solve. The framework also shows exactly how to put the pieces together. For our example problem (chosen for its straightforward derivation), the answer works out to the intuitively satisfying result: mail to those people whose estimated expected donation is greater than the cost associated with mailing! Mathematically, we simply look for those whose expected benefit of targeting is greater than zero, and simplify the inequality algebraically. Let $d_R(\mathbf{x})$ be the estimated donation if consumer $\mathbf{x}$ were to respond, and let c be the mailing cost. Then:

$$\text{Expected benefit of targeting} = p(R \mid \mathbf{x}) \cdot v_R(\mathbf{x}) + [1 - p(R \mid \mathbf{x})] \cdot v_{NR}(\mathbf{x})$$

We always want this benefit to be greater than zero, so:

$$\begin{aligned} p(R \mid \mathbf{x}) \cdot (d_R(\mathbf{x}) - c) + [1 - p(R \mid \mathbf{x})] \cdot (-c) &> 0 \\ p(R \mid \mathbf{x}) \cdot d_R(\mathbf{x}) - p(R \mid \mathbf{x}) \cdot c - c + p(R \mid \mathbf{x}) \cdot c &> 0 \\ p(R \mid \mathbf{x}) \cdot d_R(\mathbf{x}) &> c \end{aligned}$$

That is, the expected donation (lefthand side) should be greater than the solicitation cost (righthand side).

A Brief Digression on Selection Bias

This example brings up an important data science issue whose detailed treatment is beyond the scope of this book, but nevertheless is important to discuss briefly. For modeling the predicted donation, notice that the data may well be biased—meaning that they are not a random sample from the population of all donors. Why? Because the data are from past donations—from the individuals who *did respond* in the past. This is similar to the idea of modeling creditworthiness based on the experience with past credit customers: those are likely the people whom you had deemed to be creditworthy

in the past! However, you want to apply the model to the general population to find good prospects. Why would those who happened to have been selected in the past be a good sample from which to model the general population? This is an example of *selection bias*—the data were not selected randomly from the population to which you intend to apply the model, but instead were biased in some way (by who happened to donate, and perhaps by those who were targeted using past methods; by who was granted credit in the past).

One important question for the data scientist is: do you expect the particular selection procedure that biases the data also to have a bearing on the value of the target variable? In modeling creditworthiness, the answer is absolutely *yes*—the past customers were selected precisely because they were predicted to be creditworthy. The donation case is not as straightforward, but it seems reasonable to expect that people who donate larger sums do not donate as often. For example, some people may donate $10 each and every time they're asked. Others may give $100 and then feel they need not donate for a while, ignoring many subsequent campaigns. The result would be that those who happened to donate in some past campaign will be biased towards those who donate *less*.

Fortunately, there are data science techniques to help modelers deal with selection bias. They are beyond the scope of this book, but the interested reader might start by reading (Zadrozny & Elkan, 2001; Zadrozny, 2004) for an illustration of dealing with selection bias in this exact donation solicitation case study.

Our Churn Example Revisited with Even More Sophistication

Let's return to our example of churn and apply what we've learned to examine it data-analytically. In our prior forays, we did not treat the problem as comprehensively as we might. That was by design, of course, because we had not learned everything we needed yet, and the intermediate attempts were illustrative. But now let's examine the problem in more detail, applying the exact same fundamental data science concepts as we just applied to the case of soliciting donations.

The Expected Value Framework: Structuring a More Complicated Business Problem

First, what exactly is the business problem we would like to solve? Let's keep our basic example problem setting: we're having a serious problem with churn in our wireless business. Marketing has designed a special retention offer. Our task is to target the offer to some appropriate subset of our customer base.

Initially, we had decided that we would try to use our data to determine which customers would be the most likely to defect shortly after their contracts expire. Let's continue to focus on the set of customers whose contracts are about to expire, because this is where

most of the churn occurs. However, do we really want to target our offer to those with the highest probability of defection?

We need to go back to our fundamental concept: what exactly is the business problem we want to solve. Why is churn a problem? Because it causes us to lose money. The real business problem is losing money. If a customer actually were costly to us rather than profitable, we may not mind losing her. We would like to limit the amount of money we are losing—not simply to keep the most customers. Therefore, as in the donation problem, we want to take the *value* of the customer into account. Our expected value framework helps us to frame that analysis, similar to how it did above. In the case of churn, the value of an individual may be much easier to estimate: these are our customers, and since we have their billing records we can probably forecast their future value pretty well (contingent on their staying with the company) with a simple extrapolation of their past value. However, in this case we have not completely solved our problem, and framing the analysis using expected value shows why.

Let's apply our expected value framework to really dig down into the business understanding/data understanding segment of the data mininig process. Is there any problem with treating this case exactly as we did the donation case? As with the donation case study, we might represent the expected benefit of targeting a customer with the special offer as:

$$\text{Expected benefit of targeting} = p(S \mid \mathbf{x}) \cdot v_S(\mathbf{x}) + [1 - p(S \mid \mathbf{x})] \cdot v_{NS}(\mathbf{x})$$

where $p(S \mid \mathbf{x})$ is the probability that the customer will **S**tay with the company after being targeted, $v_S(\mathbf{x})$ is the value we get if consumer $\mathbf{x}$ stays with the company and $v_{NS}(\mathbf{x})$ is the value we get if consumer $\mathbf{x}$ does not stay (defects or churns).

Can we use this to target customers with the special offer? All else being equal, targeting those with the highest value seems like it simply targets those with the highest probability of *staying*, rather than the highest probability of leaving! To see this let's oversimplify by assuming that the value if the customer does not stay is zero. Then our expected value becomes:

$$\text{Expected benefit of targeting} = p(S \mid \mathbf{x}) \cdot v_s(\mathbf{x})$$

That does not jibe with our prior intuition that we want to target those who have the highest probability of leaving. What's wrong? Our expected value framework tells us exactly—let's be more careful. We don't want to just apply what we did in the donation problem, but to think carefully about this problem. We don't want to target those with the highest value if they were to stay. We want to target those where we would lose the most value if they were to leave. That's complicated, but our expected value framework can help us to work through the thinking systematically, and as we will see that will cast

an interesting light on the solution. Recall that in the donation example we said, "To be complete, we would also want to assess the expected benefit of not targeting, and then compare the two to make the decision of whether to target or not." We allowed ourselves to ignore this in the donation setting because we assumed that consumers were not going to donate spontaneously without a solicitation. However, in the business understanding phase we need to think through the specifics of each particular business problem.

Let's think about the "not targeting" case of the churn problem. Is the value zero if we don't target? No, not necessarily. If we do not target and the customer stays anyway, then we actually achieve higher value because we did not expend the cost of the incentive!

Assessing the Influence of the Incentive

Let's dig even deeper, calculating both the benefit of targeting a customer with the incentive and of not targeting her, and making the cost of the incentive explicit. Let's call $u_S(\mathbf{x})$ the profit from customer $\mathbf{x}$ if she stays, not including the incentive cost; and $u_{NS}(\mathbf{x})$ the profit from customer $\mathbf{x}$ if she leaves, not including the incentive cost. Furthmore, for simplicity, let's assume that we incur the incentive cost c no matter whether the customer stays or leaves.

For churn this is not completely realistic, as the incentives usually include a large cost component that is contingent upon staying, such as a new phone. Expanding the analysis to include this small complication is straightforward, and we would draw the same qualitative conclusions. Try it.

So let's compute separately the expected benefit if we target or if we do not target. In doing so, we need to clarify that there (hopefully) will be different estimated probabilities of staying and churning depending on whether we target (i.e., hopefully the incentive actually has an effect), which we indicate by conditioning the probability of staying on the two possibilities (target, T, or not target, $notT$). The expected benefit of targeting is:

$$EB_T(\mathbf{x}) = p(S \mid \mathbf{x}, T) \cdot (u_s(\mathbf{x}) - c) + [1 - p(S \mid \mathbf{x}, T)] \cdot (u_{NS}(\mathbf{x}) - c)$$

The expected benefit of not targeting is:

$$EB_{notT}(\mathbf{x}) = p(S \mid \mathbf{x}, notT) \cdot (u_s(\mathbf{x}) - c) + [1 - p(S \mid \mathbf{x}, notT)] \cdot (u_{NS}(\mathbf{x}) - c)$$

So, now to complete our business problem formulation, we would like to target those customers for whom we would see the greatest expected benefit *from targeting them*.

These are specifically those customers where $EB_T(\mathbf{x}) - EB_{notT}(\mathbf{x})$ is the largest. This is a substantially more complex problem formulation than we have seen before—but the expected value framework structures our thinking so we can think systematically and engineer our analysis focusing precisely on the goal.

The expected value framework also allows us to see what is different about this problem structure than those that we have considered in the past. Specifically, we need to consider what would happen if we did *not target* (looking at both EB_T and EB_{notT}), as well as what is the actual *influence* of the incentive (taking the difference of EB_T and EB_{notT}).[1]

Let's take another brief mathematical digression to illustrate. Consider the conditions under which this "value of targeting," $VT = EB_T(\mathbf{x}) - EB_{notT}(\mathbf{x})$, would be the largest. Let's expand the equation for VT, but at the same time simplify by assuming that we get no value from a customer if she does not stay.

Equation 11-1. VT decomposition

$$\begin{aligned} VT &= p(S \mid \mathbf{x}, T) \cdot u_S(\mathbf{x}) - p(S \mid \mathbf{x}, notT) \cdot u_S(\mathbf{x}) - c \\ &= [p(S \mid \mathbf{x}, T) - p(S \mid \mathbf{x}, notT)] \cdot u_S(\mathbf{x}) - c \\ &= \Delta(p) \cdot u_S(\mathbf{x}) - c \end{aligned}$$

where $\Delta(p)$ is the difference in the predicted probabilities of staying, depending on whether the customer is targeted or not. Again we see an intuitive result: we want to target those customers with the greatest change in their probability of staying, moderated by their value if they were to stay! In other words, target those with the greatest change in their expected value as a result of targeting. (The $-c$ is the same for everyone in our scenario, and including it here simply assures that the VT is not expected to be a monetary loss.)

It's important not to lose track: this was all work in our Business Understanding phase. Let's turn to the implications for the rest of the data mining process.

From an Expected Value Decomposition to a Data Science Solution

The prior discussion and specifically the decomposition highlighted in Equation 11-1 guide us in our data understanding, data formulation, modeling, and evaluation. In

1. This also is an essential starting point for *causal analysis*: create a so-called counterfactual situation assessing the difference in expected values between two otherwise identical settings. These settings are often called the "treated" and "untreated" cases, in analogy to medical inference, where one often wants to assess the causal influence of the treatment. The many different frameworks for causal analysis, from randomized experimentation, to regression-based causal analysis, to more modern causal modeling approaches, all have this difference in expected values at their core. We will discuss causal data analysis further in Chapter 12.

particular, from the decomposition we can see precisely what models we will want to build: models to estimate $p(S \mid \mathbf{x}, T)$ and $p(S \mid \mathbf{x}, notT)$ the probability that a customer will stay if targeted and the probability that a customer will stay anyway, even if not targeted. Unlike our prior data mining solutions, here we want to build two separate probability estimation models. Once these models are built, we can use them to compute the expected value of targeting.

Importantly, the expected value decomposition focuses our Data Understanding efforts. What data do we need to build these models? In both cases, we need samples of customers who have reached contract expiration. Indeed, we need samples of customers who have gone far enough beyond contract expiration that we are satisfied with concluding they have definitely "stayed" or "left." For the first model we need a sample of customers who were targeted with the offer. For the second model, we need a sample of customers who were *not* targeted with the offer. Hopefully this would be a representative sample of the customer base to which the model was applied (see the above discussion of selection bias). Developing our Data Understanding, let's think more deeply about each of these in turn.

How can we obtain a sample of such customers who have not been targeted with the offer? First, we should assure ourselves that nothing substantial has changed in the business environment that would call into question the use of historical data for churn prediction (e.g., the introduction of the iPhone only to AT&T customers would have been such an event for the other phone companies). Assuming there has been no such event, gathering the requisite data should be relatively straightforward: the phone company keeps substantial data on customers for many months, for billing, fraud detection, and other purposes. Given that this is a new offer, none of them would have been targeted with it. We would want to double-check that none of our customers was made some other offer that would affect the likelihood of churning.

The situation with modeling $p(S \mid \mathbf{x}, T)$ is quite different, and again highlights how the expected value framework can focus our thinking early, highlighting issues and challenges that we face. What's the challenge here? This is a new offer. No one has seen it yet. We do not have the data to build a model to estimate $p(S \mid \mathbf{x}, T)$!

Nonetheless, business exigencies may force us to proceed. We need to reduce churn; Marketing has confidence in this offer, and we certainly have some data that might inform how we proceed. This is not an uncommon situation in the application of data science to solving a real business problem. The expected value decomposition can lead us to a complex formulation that helps us to understand the problem, but for which we are not willing or able to address the full complexity. It may be that we simply do not have the resources (data, human, or computing). In our churn example, we do not have the data necessary.

A different scenario might be that we do not believe that the added complexity of the full formulation will add substantially to our effectiveness. For example, we might con-

clude, "*Yes, the formulation of Equation 11-1 helps me understand what I should do, but I believe I will do just about as well with a simpler or cheaper formulation.*" For example, what if we were to assume that when given the offer, everyone would Stay with certainty, $p(S \mid \mathbf{x}, T) = 1$? This is obviously an oversimplification, but it may allow us to act—and in business we need to be ready to act even without ideal information. You could verify via Equation 11-1 that the result of applying this assumption would be simply to target those customers with the largest $1 - p(S \mid \mathbf{x}, notT) \cdot u\ S\ (\mathbf{x})$— i.e., the customers with the largest expected loss if they were to leave. That makes a lot of sense if we do not have data on the actual differential effect of the offer.

Consider an alternative course of action in a case such as this, where sufficient data are not available on a modeling target. One can instead label the data with a "proxy" for the target label of interest. For example, perhaps Marketing had come up with a similar, but not identical, offer in the past. If this offer had been made to customers in a similar situation (and recall the selection bias concern discussed above), it may be useful to build a model using the proxy label. [2]

The expected value decomposition highlights yet another option. What would we need to do to model $p(S \mid \mathbf{x}, T)$? We need to *obtain* data. Specifically, we need to obtain data for customers who are targeted. That means we have to target customers. However, this would incur a cost. What if we target poorly and waste money targeting customers with lower probabilities of responding? This situation relates back to our very first fundamental principle of data science: data should be treated as an asset. We need to think not only about taking advantage of the assets that we already have, but also about investing in data assets from which we can generate important returns. Recall from Chapter 1 the situation Signet Bank faced in "Data and Data Science Capability as a Strategic Asset" on page 9. They did not have data on the differential response of customers to the various new sorts of offers they had designed. So they invested in data, taking losses by making offers broadly, and the data assets they acquired is considered to be the reason they became the wildly successful Capital One. Our situation may not be so grand, in that we have a single offer, and in making the offer we are not likely to lose the sort of money that Signet Bank did when their customers defaulted. Nonetheless, the lesson is the same: if we are willing to invest in data on how people will respond to this offer, we may be able to better target the offer to future customers.

2. For some applications, proxy labels might come from completely different events from the event on which the actual target label is based. For example, for building models to predict who will purchase after being targeted with an advertisement, data on actual conversions are scarce. It is surprisingly effective to use visiting the compaign's brand's website as a modeling proxy for purchasing (Dalessandro, Hook, Perlich, & Provost, 2012).

It's worth reiterating the importance of deep business understanding. Depending on the structure of the offer, we may not lose that much if the offer is not taken, so the simpler formulation above may be quite satisfactory.

Note that this investment in data can be managed carefully, also applying conceptual tools developed through the book. Recall the notion of visualizing performance via the learning curve, from Chapter 8. The learning curve helps us to understand the relationship between the amount of data—in this case, the amount of investment in data so far—and the resultant improvement in generalization performance. We can easily extend the notion of generalization performance to include the improvement in performance over a baseline (recall our fundamental concept: think carefully about what you will compare to). That baseline could be our alternative, simple churn model. Thus, we would slowly invest in data, examining whether increasing our data is improving our performance, and whether extrapolating the curve indicates that there are more improvements to come. If this analysis suggests that the investment is not worthwhile, it can be aborted.

Importantly, that does not mean the investment was wasteful. We invested in information: here, information about whether the additional data would pay off for our ultimate task of cost-effective churn reduction.

Furthermore, framing the problem using expected value allows extensions to the formulation to provide a structured way to approach the question of: *what is the right offer to give*. We could expand the formulation to include multiple offers, and judge which gives the best value for any particular customer. Or we could parameterize the offers (for example with a variable discount amount) and then work to optimize what discount will yield the best expected value. This would likely involve additional investment in data, running experiments to judge different customers' probabilities of staying or leaving at different offer levels—again similar to what Signet Bank did in becoming Capital One.

Summary

By following through the donation and churn examples, we have seen how the expected value framework can help articulate the true business problem and the role(s) data mining will play in its solution.

It is possible to keep elaborating the business problem into greater and greater detail, uncovering additional complexity in the problem (and greater demands on its solution). You may wonder, "*Where does this all end? Can't I keep pushing the analysis on forever?*" In principle, yes, but modeling always involves making some simplifying assump-

tions to keep the problem tractable. There will always be points in analytical engineering at which you should conclude:

- We can't get data on this event,
- It would be too expensive to model this aspect accurately,
- This event is so improbable we're just going to ignore it, or
- This formulation seems sufficient for the time being, and we should proceed with it.

The point of analytical engineering is not to develop complex solutions by addressing every possible contingency. Rather, the point is to promote thinking about problems data analytically so that the role of data mining is clear, the business constraints, cost, and benefits are considered, and any simplifying assumptions are made consciously and explicitly. This increases the chance of project success and reduces the risk of being blindsided by problems during deployment.

CHAPTER 12
Other Data Science Tasks and Techniques

Fundamental concepts: *Our fundamental concepts as the basis of many common data science techniques; The importance of familiarity with the building blocks of data science.*

Exemplary techniques: *Association and co-occurrences; Behavior profiling; Link prediction; Data reduction; Latent information mining; Movie recommendation; Bias-variance decomposition of error; Ensembles of models; Causal reasoning from data.*

As discussed in the previous chapter, a useful way to think of a team approaching a business problem data analytically is that they are faced with an *engineering* problem—not mechanical engineering or even software engineering, but *analytical engineering*. The business problem itself provides the goal as well as constraints on its solution. The data and domain knowledge provide raw materials. And data science provides frameworks for decomposing the problem into subproblems, as well as tools and techniques for solving them. We have discussed some of the most valuable conceptual frameworks and some of the most common building blocks for solutions. However, data science is a vast field, with entire degree programs devoted to it, so we cannot hope to be exhaustive in a book like this. Fortunately, the fundamental principles we have discussed undergird most of data science.

As with other engineering problems, it is often more efficient to cast a new problem into a set of problems for which we already have good tools, rather than trying to build a custom solution completely from scratch. Analytical engineering is not different: data science provides us with an abundance of tools to solve particular, common tasks. So we have illustrated the fundamental principles with some of the most common tools, methods for finding correlations/finding informative variables, finding similar entities, classification, class-probability estimation, regression, and clustering.

These are tools for the most common data science tasks, but as described in Chapter 2 there are others as well. Fortunately, the same fundamental concepts that underlie the tasks we have used for illustration also underlie these others. So now that we've

presented the fundamentals, let's briefly discuss some of the other tasks and techniques we haven't yet discussed.

Co-occurrences and Associations: Finding Items That Go Together

Co-occurrence grouping or *association discovery* attempts to find associations between entities based on transactions involving them. Why would we want to find such co-occurrences? There are many applications. Consider a consumer-facing application. Let's say that we run an online retailer. Based on shopping cart data, we might tell a customer, "Customers who bought the new eWatch also bought the eBracelet Bluetooth speaker companion." If the associations indeed capture true consumer preferences, this might increase revenue from cross-selling. It also could enhance the consumer experience (in this case, by allowing stereo music listening from their otherwise monaural eWatch), and thus leverage our data asset to create additional customer loyalty.

Consider an operations application where we ship products to online customers from many distribution centers across the globe. Not every distribution center stocks every product. Indeed, the smaller, regional distribution centers only stock the more frequently purchased products. We built these regional distribution centers to reduce shipping expense, but in practice we see that for many orders we end up either having to ship from the main distribution center anyway, or to make multiple deliveries for many orders. The reason is that even when people order popular items, they often include less-popular items as well. This is a business problem we can try to address by mining associations from our data. If there are particular less-popular items that co-occur often with the most-popular items, these also could be stocked in the regional distribution centers, achieving a substantial reduction in our shipping costs.

The co-occurrence grouping is simply a search through the data for combinations of items whose statistics are "interesting." There are different ways of framing the task, but let's think of the co-occurrence as a rule: *"If A occurs then B is likely to occur as well."* So *A* might be the sale of an eWatch, and *B* the sale of the eBracelet.[1] The statistics on "interesting" generally follow our fundamental principles.

First, we need to consider complexity control: there are likely to be a tremendous number of cooccurrences, many of which might simply be due to chance, rather than to a generalizable pattern. A simple way to control complexity is to place a constraint that such rules must apply to some minimum percentage of the data—let's say that we require rules to apply to at least 0.01% of all transactions. This is called the *support* of the association.

1. *A* and *B* could be multiple items as well. We will presume that they are single items for the moment. The Facebook Likes example below generalizes to multiple items.

We also have the notion of "likely" in the association. If a customer buys the eWatch then she is likely to buy the eBracelet. Again, we may want to require a certain minimum degree of likelihood for the associations we find. We can quantify this notion again using the same notions we have already seen. The probability that B occurs when A occurs we've seen before; it is $p(B|A)$, which in association mining is called the *confidence* or *strength* of the rule. Let's call that "strength," so as not to confuse it with statistical confidence. So we might say we require the strength to be above some threshold, such as 5% (so that 5% or more of the time, a buyer of A also buys B).

Measuring Surprise: Lift and Leverage

Finally, we would like the association to be in some sense "surprising." There are many notions of surprisingness that have been pursued in data mining, but unfortunately most of them involve matching the discovered knowledge to our prior background knowledge, intuition, and common sense. In other words, an association is surprising if it contradicts something we already knew or believed. Researchers study how to address this difficult-to-codify knowledge, but dealing with it automatically is not common in practice. Instead, data scientists and business users pore over long lists of associations, culling the unsurprising ones.

However, there is a weaker but nonetheless intuitive notion of surprisingness that can be computed from the data alone, and which we already have encountered in other contexts: **lift** — how much more frequently does this association occur than we would expect by chance? If associations from supermarket shopping cart data revealed that bread and milk are often bought together, we might say: "Of course." Many people buy milk and many people buy bread. So we would expect them to occur together frequently just by chance. We would be more surprised if we found associations that occur much more frequently than chance would dictate. Lift is calculated simply by applying basic notions of probability.

Equation 12-1. Lift

$$\text{Lift}(A,\ B) = \frac{p(A,\ B)}{p(A)p(B)}$$

In English, the lift of the co-occurrence of A and B is the probability that we actually see the two together, compared to the probability that we would see the two together if they were unrelated to (independent of) each other. As with other uses of lift we've seen, a lift greater than one is the factor by which seeing A "boosts" the likelihood of seeing B as well.

This is only one possible way to compute how much more likely than chance a discovered association is. An alternative is to look at the difference of these quantities rather than their ratio. This measure is called **leverage**.

Equation 12-2. Leverage

$$\text{Leverage}(A, B) = p(B, A) - p(A)p(B)$$

Take a minute to convince yourself that one of these would be better for associations that are very unlikely to occur by chance, and one better for those rather likely to occur by chance.

Example: Beer and Lottery Tickets

As we've already seen from the "eWatch and eBracelet" example, association discovery is often used in market basket analysis to find and analyze co-ocurrences of bought items. Let's work through a concrete example.

Suppose we operate a small convenience store where people buy groceries, liquor, lottery tickets, and so on. Let's say we analyze all of our transactions over a year's time. We discover that people often buy beer and lottery tickets together. However, we know that in our store, people buy beer often and people buy lottery tickets often. Let's say we find that 30% of all transactions involve beer, and 20% of the transactions include both beer and lottery tickets! Is this co-occurrence an interesting one? Or is it simply due to the commonality of these two purchases? Association statistics can help us.

First, let's state an association rule representing this belief: "Customers who buy beer are also likely to buy lottery tickets"; or more tersely, "beer ⇒ lottery tickets." Next, let's calculate the lift of this association. We already know one value we need: p(beer)=0.3. Let's say that lottery tickets also are very popular: p(lottery tickets)=0.4. If these two items were completely unrelated (independent), the chance that they would be bought together would be the product of these two: p(beer) × p(lottery tickets)=0.12.

We also have the actual probability (frequency in the data) of people buying the two items together, p(lottery tickets, beer), which we found by combing through the register receipt data looking for all transactions including beer and lottery tickets. As mentioned above, 20% of the transactions included both, and this is our probability: p(lottery tickets, beer) = 0.2. So the lift is 0.2 / 0.12, which is about 1.67. This means that buying lottery tickets and beer together is about 1 2/3 times more likely than one would expect by chance. We might conclude that there is some relationship there, but much of the co-occurrence is due to the fact that these are each very popular items.

What about leverage? This is p(lottery tickets, beer) – p(lottery tickets) × p(beer), which is 0.2 – 0.12, or 0.08. Whatever is driving the co-occurrence results in an eight

percentage-point increase in the probability of buying both together over what we would expect simply because they are popular items.

There are two other significant statistics we should calculate too: the support and the strength. The *support* of the association is just the prevalence in the data of buying the two items together, *p*(lottery tickets, beer), which is 20%. The *strength* is the conditional probability, *p*(lottery tickets|beer), which is 67%.

Associations Among Facebook Likes

Although finding associations is often used with market basket data—and sometimes is even called *market-basket analysis*—the technique is much more general. We can use our example from Chapter 9 of "Likes" on Facebook to illustrate. Recall that we have data on the things that were "Liked" by a large collection of users of Facebook (Kosinski, Stillwell, & Graepel, 2013). By analogy to market basket data, we can consider each of these users to have a "basket" of Likes, by aggregating all the Likes of each user. Now we can ask, do certain Likes tend to co-occur more frequently than we would expect by chance? We will use this simply as an interesting example to illustrate association finding, but the process could actually have an important business application. If you are a marketer looking to understand the consumers in a particular market, you might be interested in finding patterns of things people Like. If you are thinking data-analytically, you will apply exactly the sort of thinking we've illustrated so far in this chapter: you'll want to know what things co-occur more frequently than you would expect by chance.

Before we get to the mining of the data, let's introduce one more useful idea for association finding. Since we're using the market basket as an analogy at this point, we should consider broadening our thinking of what might be an item. Why can't we put just about anything we might be interested in finding associations with into our "basket"? For example, we might put a user's location into the basket, and then we could see associations between Likes and locations. For actual market basket data, these sometimes are called *virtual* items, to distinguish from the actual items that people put into their basket in the store. For our Facebook data, recall that we might obtain psychometric data on many of the consumers, such as their degree of extroversion or agreeableness, or their score on an IQ test. It may be interesting to allow the association search to find associations with these psychometric characteristics as well.

Note: Supervised Versus Unsupervised?

We should keep in mind our distinction between supervised and unsupervised data mining. If we want specifically to understand what correlates most with agreeableness or with Liking our brand, we should formulate this as a supervised problem, with the corresponding target variable. This is what we did when looking at the evidence lifts in Chapter 9, and at supervised segmentation throughout the book. If we want to explore the data without such a specific goal, then association finding may be more appropriate. See the discussion in Chapter 6 on the differences between supervised and unsupervised mining—there in the context of clustering, but the fundamental concepts apply to association mining as well.

OK, so let's see what associations we get among Facebook Likes.[2] These associations were found using the popular association mining system Magnum Opus.[3] Magnum Opus allows searching for associations that give the highest lift or highest leverage, while filtering out associations that cover too few cases to be interesting. The list below shows some of the highest lift associations among Facebook Likes with the constraint that they have to cover at least 1% of the users in the dataset. Do these associations make sense? Do they give us a picture of the relationships among the users' tastes? Note that the lifts are all above 20, meaning that all of these associations are at least *20 times more likely* than we would expect by chance:

```
Family Guy & The Daily Show -> The Colbert Report
Support=0.010; Strength=0.793; Lift=31.32; Leverage=0.0099

Spirited Away -> Howl's Moving Castle
Support=0.011; Strength=0.556; Lift=30.57; Leverage=0.0108

Selena Gomez -> Demi Lovato
Support=0.010; Strength=0.419; Lift=27.59; Leverage=0.0100

I really hate slow computers & Random laughter when remembering something ->
    Finding Money In Your Pocket
Support=0.010; Strength=0.726; Lift=25.80; Leverage=0.0099

Skittles & Glowsticks -> Being Hyper!
Support=0.011; Strength=0.529; Lift=25.53; Leverage=0.0106

Linkin Park & Disturbed & System of a Down & Korn -> Slipknot
Support=0.011; Strength=0.862; Lift=25.50; Leverage=0.0107
```

2. Thanks to Wally Wang for help with this.
3. See this page.

```
Lil Wayne & Rihanna -> Drake
Support=0.011; Strength=0.619; Lift=25.33; Leverage=0.0104

Skittles & Mountain Dew -> Gatorade
Support=0.010; Strength=0.519; Lift=25.23; Leverage=0.0100

SpongeBob SquarePants & Converse -> Patrick Star
Support=0.010; Strength=0.654; Lift=24.94; Leverage=0.0097

Rihanna & Taylor Swift -> Miley Cyrus
Support=0.010; Strength=0.490; Lift=24.90; Leverage=0.0100

Disturbed & Three Days Grace -> Breaking Benjamin
Support=0.012; Strength=0.701; Lift=24.64; Leverage=0.0117

Eminem & Lil Wayne -> Drake
Support=0.014; Strength=0.594; Lift=24.30; Leverage=0.0131

Adam Sandler & System of a Down & Korn -> Slipknot
Support=0.010; Strength=0.819; Lift=24.23; Leverage=0.0097

Pink Floyd & Slipknot & System of a Down -> Korn
Support=0.010; Strength=0.810; Lift=24.05; Leverage=0.0097

Music & Anime -> Manga
Support=0.011; Strength=0.675; Lift=23.99; Leverage=0.0110

Medium IQ & Sour Gummy Worms -> I Love Cookie Dough
Support=0.012; Strength=0.568; Lift=23.86; Leverage=0.0118

Rihanna & Drake -> Lil Wayne
Support=0.011; Strength=0.849; Lift=23.55; Leverage=0.0104

I Love Cookie Dough -> Sour Gummy Worms
Support=0.014; Strength=0.569; Lift=23.28; Leverage=0.0130

Laughing until it hurts and you can't breathe! & I really hate slow computers ->
    Finding Money In Your Pocket
Support=0.010; Strength=0.651; Lift=23.12; Leverage=0.0098

Evanescence & Three Days Grace -> Breaking Benjamin
Support=0.012; Strength=0.656; Lift=23.06; Leverage=0.0117

Disney & Disneyland -> Walt Disney World
Support=0.011; Strength=0.615; Lift=22.95; Leverage=0.0103

i finally stop laughing... look back over at you and start all over again ->
    That awkward moment when you glance at someone staring at you.
Support=0.011; Strength=0.451; Lift=22.92; Leverage=0.0104

Selena Gomez -> Miley Cyrus
Support=0.011; Strength=0.443; Lift=22.54; Leverage=0.0105
```

```
Reese's & Starburst -> Kelloggs Pop-Tarts
Support=0.011; Strength=0.493; Lift=22.52; Leverage=0.0102

Skittles & SpongeBob SquarePants -> Patrick Star
Support=0.012; Strength=0.590; Lift=22.49; Leverage=0.0112

Disney & DORY & Toy Story -> Finding Nemo
Support=0.011; Strength=0.777; Lift=22.47; Leverage=0.0104

Katy Perry & Taylor Swift -> Miley Cyrus
Support=0.011; Strength=0.441; Lift=22.43; Leverage=0.0101

AKON & Black Eyed Peas -> Usher
Support=0.010; Strength=0.731; Lift=22.42; Leverage=0.0097

Eminem & Drake -> Lil Wayne
Support=0.014; Strength=0.807; Lift=22.39; Leverage=0.0131
```

Most association mining examples use domains (such as Facebook Likes) where readers already have a fair knowledge of the domain. This is because otherwise, since the mining is unsupervised, evaluation depends much more critically on domain knowledge validation (recall the discussion in Chapter 6)—we do not have a well-defined target task for an objective evaluation. However, one interesting practical use of association mining is to explore data that we do not understand so well. Consider going into a new job. Exploring the company's customer transaction data and examining the strong co-occurrences can quickly give broad overview of the taste relationships in the customer base. So, with that in mind, look back at the co-occurrences in the Facebook Likes and pretend that this was not a domain of popular culture: these and others like them (there are huge numbers of such associations) would give you a very broad view of the related tastes of the customers.

Profiling: Finding Typical Behavior

Profiling attempts to characterize the typical behavior of an individual, group, or population. An example profiling question might be: *What is the typical credit card usage of this customer segment?* This could be a simple average of spending, but such a simple description might not represent the behavior well for our business task. For example, fraud detection often uses profiling to characterize normal behavior and then looks for instances that deviate substantially from the normal behavior—especially in ways that previously have been indicative of fraud (Fawcett & Provost, 1997; Bolton & Hand, 2002). Profiling credit card usage for fraud detection might require a complex description of weekday and weekend averages, international usage, usage across merchant and product categories, usage from suspicious merchants, and so on. Behavior can be described generally over an entire population, at the level of small groups, or even for each individual. For example, each credit card user might be profiled with respect to his

international usage, so as not to create many false alarms for an individual who commonly travels abroad.

Profiling combines concepts discussed previously. Profiling can essentially involve clustering, if there are subgroups of the population with different behaviors. Many profiling methods seem complicated, but in essence are simply instantiations of the fundamental concept introduced in Chapter 4: define a numeric function with some parameters, define a goal or objective, and find the parameters that best meet the objective.

So let's consider a simple example from business operations management. Businesses would like to use data to help to understand how well their call centers are supporting their customers.[4] One aspect of supporting customers well is to not leave them sitting on hold for long periods of time. So how might we profile the typical wait time of our customers who call into the call center? We might calculate the mean and standard deviation of the wait time.

That seems like exactly what a manager with basic statistical training might do—it turns out to be a simple instance of model fitting. Here's why. Let's assume that customer wait times follow a Normal or Gaussian distribution. Saying such things can cause a non-mathematical person to fear what's to come, but that just means the distribution follows a bell curve with some particularly nice properties. Importantly, it is a "profile" of the wait times that (in this case) has only two important parameters: the mean and the standard deviation. When we calculate the mean and standard deviation, we are finding the "best" profile or model of wait time under the assumption that it is Normally distributed. In this case "best" is the same notion that we discussed for logistic regression, for example, the mean we calculate from the spending gives us the mean of the Gaussian distribution that is most likely to have generated the data (the "maximum likelihood" model).

This view illustrates why a data science perspective can help even in simple scenarios: it is much clearer now what we are doing when we are calculating averages and standard deviations, even if our memory of the details from statistics classes is hazy. We also need to keep in mind our fundamental principles introduced in Chapter 4 and elaborated in Chapter 7: we need to consider carefully what we desire from our data science results. Here we would like to profile the "normal" wait time of our customers. If we plot the data and they do not look like they came from a Gaussian (a symmetric bell curve that goes to zero very quickly in the "tails"), we might want to reconsider simply reporting the mean and standard deviation. We might instead report the median, which is not so sensitive to the skew, or possibly even better, fit a different distribution (maybe after talking to a statistically oriented data scientist about what might be appropriate).

4. The interested reader is encouraged to read Brown et al. (2005) for a technical treatment and details on this application.

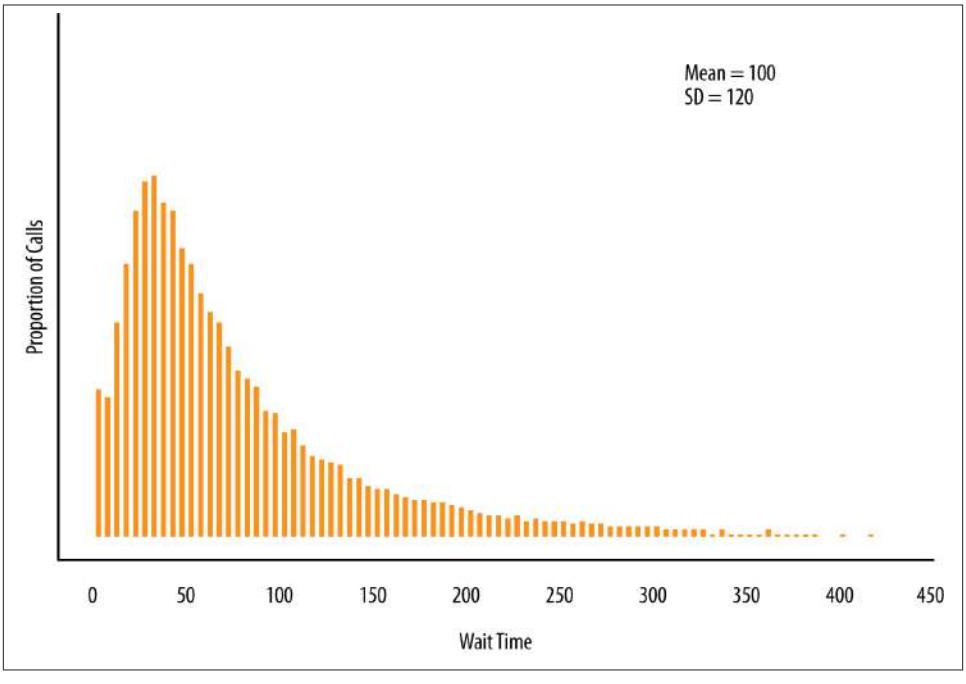

Figure 12-1. A distribution of wait times for callers into a bank's call center.

To illustrate how a data science savvy manager might proceed, let's look at a distribution of wait times for callers into a bank's call center over a couple of months. Figure 12-1 shows such a distribution. Importantly, we see how visualizing the distribution should cause our data science radar to issue an alert. The distribution is *not* a symmetric bell curve. We should then worry about simply profiling wait times by reporting the mean and standard deviation. For example, the mean (100) does not seem to satisfy our desire to profile how long our customers normally wait; it seems too large. Technically, the long "tail" of the distribution skews the mean upward, so it does not represent faithfully where most of the data really lie. It does not represent faithfully the normal wait time of our customers.

To give more depth to what our data science-savvy manager might do, let's go a little further. We will not get into the details here, but a common trick for dealing with data that are skewed in this way is to take the logarithm (log) of the wait times. Figure 12-2 shows the same distribution as Figure 12-1, except using the logarithms of the wait times. We now see that after the simple transformation, the wait times look very much like the classic bell curve.

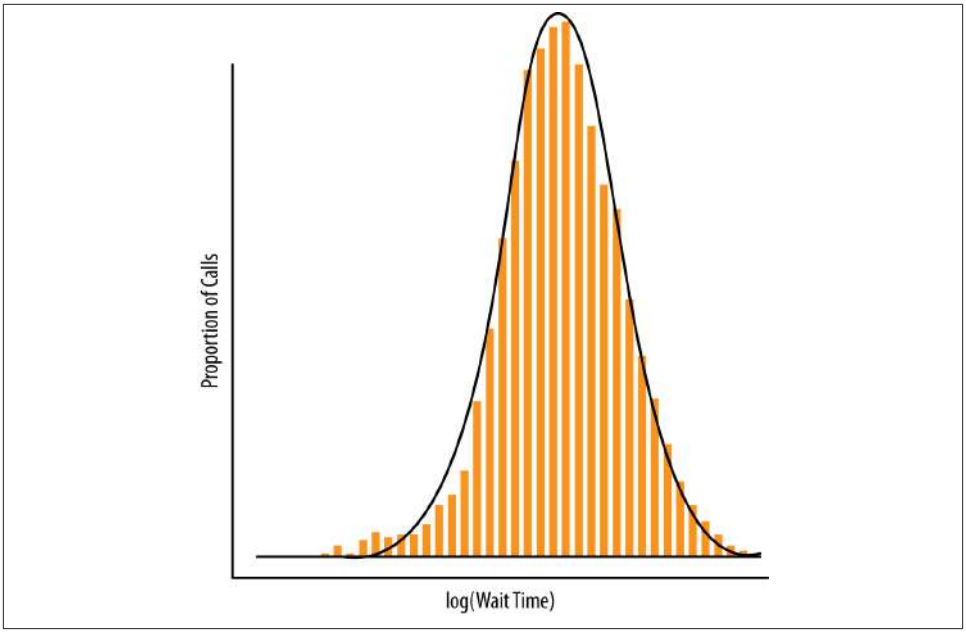

Figure 12-2. The distribution of wait times for callers into a bank's call center after a quick redefinition of the data.

Indeed, Figure 12-2 also shows an actual Gaussian distribution (the bell curve) fit to the bell-shaped distribution, as described above. It fits very well, and thus we have a justification for reporting the mean and standard deviation as summary statistics of the profile of (log) wait times.[5]

This simple example extends nicely to more complex situations. Shifting contexts, let's say we want to profile customer behavior in terms of their spending and their time on our website. We believe these to be correlated, but not perfectly, as with the points plotted in Figure 12-3. Again, a very common tack is to apply the fundamental notion of Chapter 4: choose a parameterized numeric function and an objective, and find parameters that maximize the objective. For example, we can choose a two-dimensional Gaussian, which is essentially a bell *oval* instead of a bell curve—an oval-shaped blob that is very dense in the center and thins out toward the edges. This is represented by the contour lines in Figure 12-3.

5. A statistically trained data scientist might have noticed immediately the shape of the distribution of the original data, shown in Figure 12-1. This is a so-called log-normal distribution, which just means that the logs of the quantities in question are normally distributed.

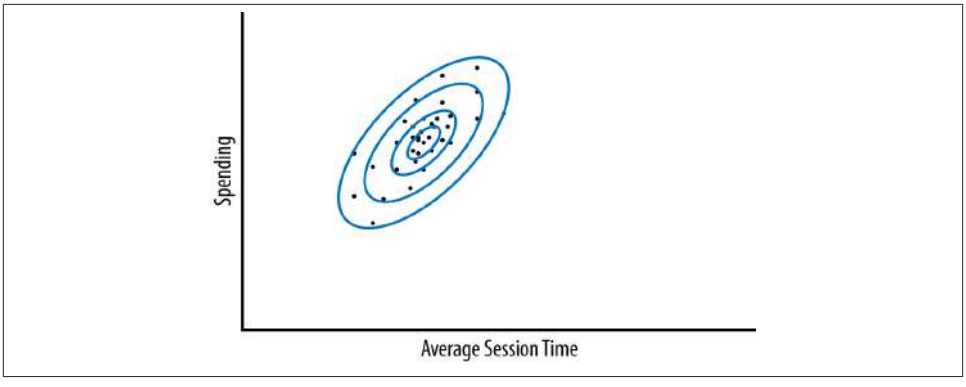

Figure 12-3. A profile of our customers with respect to their spending and the time they spend on our web site, represented as a two-dimensional Gaussian fit to the data.

We can keep extending the idea to more and more sophisticated profiling. What if we believe there are different subgroups of customers with different behaviors? We may not be willing to simply fit a Gaussian distribution to the behavior. However, maybe we are comfortable assuming that there are k groups of customers, each of whose behavior is normally distributed. We can fit a model with multiple Gaussians, called a *Gaussian Mixture Model* (GMM). Applying our fundamental concept again, finding the maximum-likelihood parameters identifies the k Gaussians that fit the data best (with respect to this particular objective function). We see an example with k=2 in Figure 12-4. The figure shows how the fitting procedure identifies two different groups of customers, each modeled by a two-dimensional Gaussian distribution.

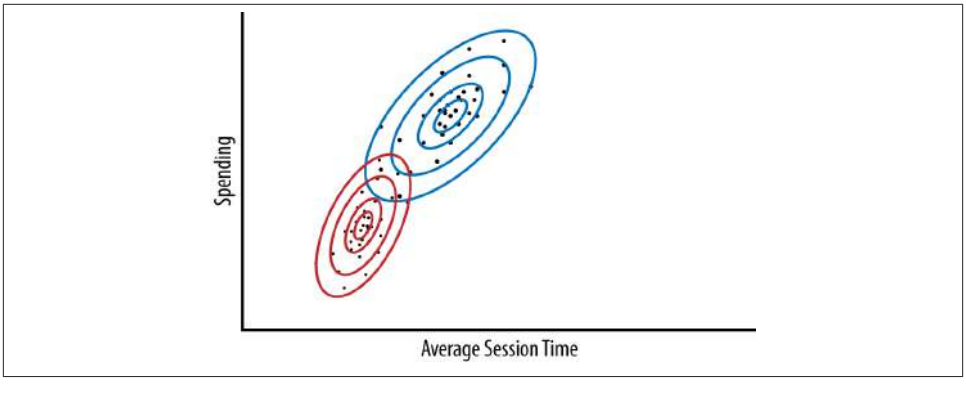

Figure 12-4. A profile of our customers with respect to their spending and the time they spend on our web site, represented as a Gaussian Mixture Model (GMM), with 2 two-dimensional Gaussians fit to the data. The GMM provides a "soft" clustering of our customers along two these two dimensions.

Now we have a rather sophisticated profile, which we can understand as a surprisingly straightforward application of our fundamental principles. An interesting side note is that this GMM has produced for us a clustering, but in a different way from the clusterings presented in Chapter 6. This illustrates how fundamental principles, rather than specific tasks or algorithms, form the basis for data science. In this case, clustering can be done in many different ways, just as classification and regression can be.

Note: "Soft" Clustering

Incidentally, you may notice that the clusters in the GMM overlap with each other. The GMM provides what is called a "soft" or probabilistic clustering. Each point does not strictly belong to a single cluster, but instead has a degree or probability of membership in each cluster. In this particular clustering, we can think that a point is more likely to have come from some clusters than others. However, there still is a possibility, perhaps remote, that the point may have come from any of them.

Link Prediction and Social Recommendation

Sometimes, instead of predicting a property (target value) of a data item, it is more useful to predict *connections* between data items. A common example of this is predicting that a link should exist between two individuals. Link prediction is common in social networking systems: *Since you and Karen share 10 friends, maybe you'd like to be Karen's friend?* Link prediction can also estimate the strength of a link. For example, for recommending movies to customers one can think of a graph between customers and the movies they've watched or rated. Within the graph, we search for links that do *not* exist between customers and movies, but that we predict should exist and should be strong. These links form the basis for recommendations.

There are many approaches to link prediction, and even an entire chapter of this book would not do them justice. However, we can understand a wide variety of approaches using our fundamental concepts of data science. Let's consider the social network case. Knowing what you know now, if you had to predict either the presence or the strength of a link between two individuals, how would you go about framing the problem? We have several alternatives. We could presume that links should be between similar individuals. We know then that we need to define a similarity measure that takes into account the important aspects of our application.

Could we define a similarity measure between two individuals that would indicate that they might like to be friends? (Or are already friends, depending on the application.) Sure. Using the example above directly, we could consider the similarity to be the number of shared friends. Of course, the similarity measure could be more sophisticated: we could weight the friends by the amount of communication, geographical proximity,

or some other factor, and then find or devise a similarity function that takes these strengths into account. We could use this friend strength as one aspect of similarity while also including others (since after Chapter 6 we are comfortable with multivariate similarity), such as shared interests, shared demographics, etc. In essence, we could apply knowledge of "finding similar data items" to people, by considering the different ways in which we could represent the people as data.

That is one way to attack the link prediction problem. Let's consider another, just to continue to illustrate how the fundamental principles apply to other tasks. Since we want to *predict* the existence (or strength) of a link, we might well decide to cast the task as a predictive modeling problem. So we can apply our framework for thinking about predictive modeling problems. As always, we start with business and data understanding. What would we consider to be an instance? At first, we might think: wait a minute—here we are looking at the *relationship* between *two* instances. Our conceptual framework comes in very handy: let's stick to our guns, and define an instance for prediction. What exactly is it that we want to predict? We want to predict the existence of a relationship (or its strength, but let's just consider the existence here) between two people. So, an instance should be a pair of people!

Once we have defined an instance to be a pair of people, we can proceed smoothly. Next, what would be the target variable? Whether the relationship exists, or would be formed if recommended. Would this be a supervised task? Yes, we can get training data where links already do or do not exist, or if we wanted to be more careful we could invest in acquiring labels specifically for the recommendation task (we would need to spend a bit more time than we have here on defining the exact semantics of the link). What would be the features? These would be features *of the pair of people*, such as how many common friends the two individuals have, what is the similarity in their interests, and so on. Now that we have cast the problem in the form of a predictive modeling task, we can start to ask what sorts of models we would apply and how we would evaluate them. This is the same conceptual procedure we go through for any predictive modeling task.

Data Reduction, Latent Information, and Movie Recommendation

For some business problems, we would like to take a large set of data and replace it with a smaller set that preserves much of the important information in the larger set. The smaller dataset may be easier to deal with or to process. Moreover, the smaller dataset may better reveal the information contained within it. For example, a massive dataset on consumer movie-viewing preferences may be reduced to a much smaller dataset revealing the consumer taste preferences that are latent in the viewing data (for example, viewer preferences for movie genre). Such data reduction usually involves sacrificing some information, but what is important is the trade-off between the insight or manageability gained against the information lost. This is often a trade worth making.

As with link prediction, data reduction is a general task, not a particular technique. There are many techniques, and we can use our fundamental principles to understand them. Let's discuss a popular technique as an example.

Let's continue to talk about movie recommendations. In a now famous (at least in data science circles) contest, the movie rental company Netflix™ offered a million dollars to the individual or team that could best predict how consumers would rate movies. Specifically, they set a prediction performance goal on a holdout data set and awarded the prize to the team that first reached this goal.[6] Netflix made available historical data on movie ratings assigned by their customers. The winning team[7] produced an extremely complicated technique, but much of the success is attributed to two aspects of the solution: (i) the use of ensembles of models, which we will discuss in "Bias, Variance, and Ensemble Methods" on page 306, and (ii) data reduction. The main data reduction technique that the winners used can be described easily using our fundamental concepts.

The problem to be solved was essentially a link prediction problem, where specifically we would like to predict the strength of the link between a user and a movie—the strength representing how much the user would like it. As we just discussed, this can be cast as a predictive modeling problem. However, what would the features be for the relationship between a user and a movie?

One of the most popular approaches for providing recommendations, described in detail in a very nice article by several of the Netflix competition winners (Koren, Bell, & Volinsky, 2009), is to base the model on *latent* dimensions underlying the preferences. The term "latent," in data science, means "relevant but not observed explicitly in the data." Chapter 10 discussed topic models, another form of latent model, where the latent information is the set of topics in the documents. Here the latent dimensions of movie preference include possible characterizations like serious versus escapist, comedy versus drama, orientation towards children, or gender orientation. Even if these are not represented explicitly in the data, they may be important for judging whether a particular user will like the movie. The latent dimensions also could include possibly ill-defined things like depth of character development or quirkiness, as well as dimensions never explicitly articulated, since the latent dimensions will emerge from the data.

Again, we can understand this advanced data science approach as a combination of fundamental concepts. The idea of the latent dimension approaches to recommendation is to represent each movie as a feature vector using the latent dimensions, and also to represent each user's preferences as a feature vector using the latent dimensions. Then it is easy to find movies to recommend to any user: compute a similarity score between the user and all the movies; the movies that best match the users' preferences would be

6. There are some technicalities to the rules of the Netflix Challenge, which you can find on the Wikipedia page.

7. The winning team, Bellkor's Pragmatic Chaos, had seven members. The history of the contest and the team evolution is complicated and fascinating. See this Wikipedia page on the Netflix Prize.

those movies most similar to the user, when both are represented by the same latent dimensions.

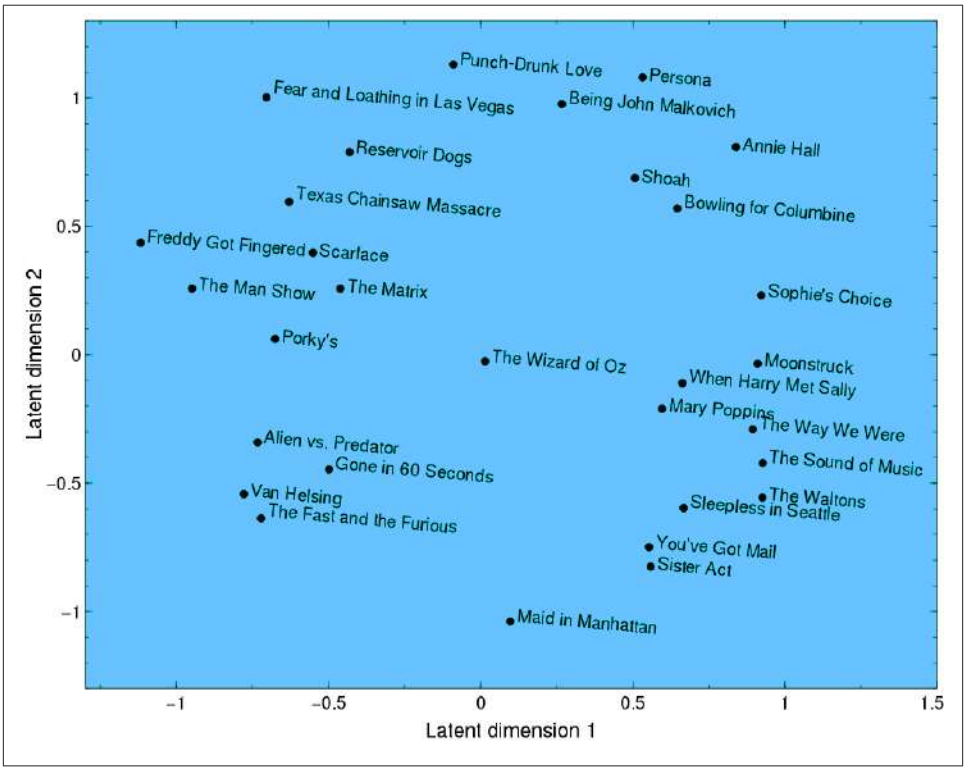

Figure 12-5. A collection of movies placed in a "taste space" defined by the two strongest latent dimensions mined from the Netflix Challenge data. See the text for a detailed discussion. A customer would also be placed somewhere in the space, based on the movies she has previously viewed or rated. A similarity-based recommendation approach would suggest the closest movies to the customer as candidate recommendations.

Figure 12-5 shows a two-dimensional latent space actually mined from the Netflix movie data,[8] as well as a collection of movies represented in this new space. The interpretation of such latent dimensions mined from data must be inferred by the data scientists or business users. The most common way is to observe how the dimensions separate the movies, then apply domain knowledge.

In Figure 12-5, the latent dimension represented by the horizontal axis seems to separate the movies into drama-oriented films on the right and action-oriented films on the left.

8. Thanks to one of the members of the winning team, Chris Volinsky, for his help here.

At the extremes, on the far right we see films of the heart such as *The Sound of Music*, *Moonstruck*, and *When Harry Met Sally*. On the far left we see whatever is the opposite of films of the heart (films of the gut?), including films focusing on the stereotypical likes of men and adolescent boys (*The Man Show*, *Porky's*), killing (*Texas Chainsaw Massacre*, *Reservoir Dogs*), speed (*The Fast and the Furious*), and monster hunting (*Van Helsing*). The latent dimension represented by the vertical axis seems to separate the movies by intellectual appeal versus emotional appeal, with movies like *Being John Malkovich*, *Fear and Loathing in Los Vegas*, and *Annie Hall* at one extreme, and *Maid in Manhattan*, *The Fast and the Furious*, and *You've Got Mail* at the other. Feel free to disagree with those interpretations of the dimensions—they are completely subjective. One thing is clear, though: *The Wizard of Oz* captures an unusual balance of whatever tastes are represented by the latent dimensions.

To use this latent space for recommendation, a customer also would be placed somewhere in the space, based on the movies she has rented or rated. The closest movies to the position of the customer would be good candidates for making recommendations. Note that for making recommendations, as always we need to keep thinking back to our business understanding. For example, different movies have different profit margins, so we may want to combine this knowledge with the knowledge of the most similar movies.

But how do we *find* the right latent dimensions in the data? We apply the fundamental concept introduced in Chapter 4: we represent the similarity calculation between a user and a movie as a mathematical formula using some number *d* of as-yet-unknown latent dimensions. Each dimension would be represented by a set of weights (the coefficients) on each movie and a set of weights on each customer. A high weight would mean this dimension is strongly associated with the movie or the customer. The meaning of the dimension would be purely implicit in the weights on the movies and customers. For example, we might look at the movies that are weighted highly on some dimension versus low-weighted movies and decide, "the highly rated movies are all 'quirky.'" In this case, we could think of the dimension as the degree of *quirkiness* of the movie, although it is important to keep in mind that this interpretation of the dimension was imposed by us. The dimension is simply some way in which the movies clustered in the data on how customers rated movies.

Recall that to fit a numeric-function model to data, we find the optimal set of parameters for the numeric function. Initially, the *d* dimensions are purely a mathematical abstraction; only after the parameters are selected to fit the data can we try to formulate an interpretation of the meaning of the latent dimensions (and sometimes such an effort is fruitless). Here, the parameters of the function would be the (unknown) weights for each customer and each movie along these dimensions. Intuitively, data mining is determining simultaneously (i) how quirky the movie is, and (ii) how much this viewer likes quirky movies.

We also need an objective function to determine what is a good fit. We define our objective function for training based on the set of movie ratings we have observed. We find a set of weights that characterizes the users and the movies along these dimensions. There are different objective functions used for the movie-recommendation problem. For example, we can choose the weights that allow us to best predict the observed ratings in the training data (subject to regularization, as discussed in Chapter 4). Alternatively, we could choose the dimensions that best explain the variation in the observed ratings. This is often called "matrix factorization," and the interested reader might start with the paper about the Netflix Challenge (Koren, Bell, & Volinsky, 2009).

The result is that we have for each movie a representation along some reduced set of dimensions—maybe how quirky it is, maybe whether it is a "tear-jerker" or a "guy flick," or whatever—the best d latent dimensions that the training finds. We also have a representation of each user in terms of their preferences along these dimensions. We can now look back at Figure 12-5 and the associated discussion. These are the two latent dimensions that best fit the data, i.e., the dimensions that result from fitting the data with d=2.

Bias, Variance, and Ensemble Methods

In the Netflix competition, the winners also took advantage of another common data science technique: they built lots of different recommendation models and combined them into one super model. In data mining parlance, this is referred to as creating an *ensemble* model. Ensembles have been observed to improve generalization performance in many situations—not just for recommendations, but broadly across classification, regression, class probability estimation, and more.

Why is a collection of models often better than a single model? If we consider each model as a sort of "expert" on a target prediction task, we can think of an ensemble as a collection of experts. Instead of just asking one expert, we find a group of experts and then somehow combine their predictions. For example, we could have them vote on a classification, or we could average their numeric predictions. Notice that this is a generalization of the method introduced in Chapter 6 to turn similarity computations into "nearest neighbor" predictive models. To make a k-NN prediction, we find a group of similar examples (very simple experts) and then apply some function to combine their individual predictions. Thus a k-nearest-neighbor model is a simple ensemble method. Generally, ensemble methods use more complex predictive models as their "experts"; for example, they may build a group of classification trees and then report an average (or weighted average) of the predictions.

When might we expect ensembles to improve our performance? Certainly, if each expert knew exactly the *same* things, they would all give the same predictions, and the ensemble would provide no advantage. On the other hand, if each expert was knowledgeable in a slightly different aspect of the problem, they might give complementary predictions,

and the whole group might provide more information than any individual expert. Technically, we would like the experts to make different sorts of *errors*—we would like their errors to be as unrelated as possible, and ideally to be completely independent. In averaging the predictions, the errors would then tend to cancel out, the predictions indeed would be complementary, and the ensemble would be superior to using any one expert.

Ensemble methods have a long history and are an active area of research in data science. Much has been written about them. The interested reader may want to start with the review article by Dietterich (2000).

One way to understand why ensembles work is to understand that the errors a model makes can be characterized by three factors:

1. Inherent randomness,
2. Bias, and
3. Variance.

The first, inherent randomness, simply covers cases where a prediction is not "deterministic," (i.e., we simply do not always get the same value for the target variable every time we see the same set of features). For example, customers described by a certain set of characteristics may not always either purchase our product or not. The prediction may simply be inherently probabilistic given the information we have. Thus, a portion of the observed "error" in prediction is simply due to this inherent probabilistic nature of the problem. We can debate whether a particular data-generating process is truly probabilistic—as opposed to our simply not seeing all the requisite information—but that debate is largely academic,[9] because the process may be essentially probabilistic based on the data we have available. Let's proceed assuming that we've reduced the randomness as much as we can, and there simply is some theoretical maximum accuracy that we can achieve for this problem. This accuracy is called the *Bayes rate*, and it is generally unknown. For the rest of this section we will consider the Bayes rate as being "perfect" accuracy.

Beyond inherent randomness, models make errors for two other reasons. The modeling procedure may be "biased." What this means can be understood best in reference to learning curves (recall "Learning Curves" on page 130). Specifically, a modeling procedure is biased if no matter how much training data we give it, the learning curve will

9. The debate sometimes can bear fruit. For example, thinking whether we have all the requisite information might reveal a new attribute that could be obtained that would increase the possible predictability.

never reach perfect accuracy (the Bayes rate). For example, we learn a (linear) logistic regression to predict response to an advertising campaign. If the true response really is more complex than the linear model can represent, the model will never achieve perfect accuracy.

The other source of error is due to the fact that we do not have an infinite amount of training data; we have some finite sample that we want to mine. Modeling procedures usually give different models from even slightly different samples. These different models will tend to have different accuracies. How much the accuracy tends to vary across different training sets (let's say, of the same size) is referred to as the modeling procedure's variance. Procedures with more variance tend to produce models with larger errors, all else being equal.

You might see now that we would like to have a modeling procedure that has no bias and no variance, or at least low bias and low variance. Unfortunately (and intuitively), there typically is a trade-off between the two. Lower variance models tend to have higher bias, and vice versa. As a very simple example, we might decide that we want to estimate the response to our advertising campaign simply by ignoring all the customer features and simply predict the (average) purchase rate. This will be a very low-variance model, because we will tend to get about the same average from different datasets of the same size. However, we have no hope to get perfect accuracy if there are customer-specific differences in propensity to purchase. On the other hand, we might decide to model customers based on one thousand detailed variables. We may now have the opportunity to get much better accuracy, but we would expect there to be much greater variance in the models we obtain based on even slightly different training sets. Thus, we won't necessarily expect the thousand-variable model to be better; we don't know exactly which source of error (bias or variance) will dominate.

You may be thinking: *Of course. As we learned in Chapter 5, the thousand-variable model will overfit. We should apply some sort of complexity control, such as selecting a subset of the variables to use*. That is exactly right. More complexity generally gives us lower bias but higher variance. Complexity control generally tries to manage the (usually unknown) trade-off between bias and variance, to find a "sweet spot" where the combination of the errors from each is smallest. So, we could apply variable selection to our thousand-variable problem. If there truly are customer-specific differences in purchase rate, and we have enough training data, hopefully the variable selection will not throw away all variables, which would leave us with just the average over the population. Hopefully, instead we would get a model with a subset of the variables that allow us to predict as well as possible, given the training data available.

Technically, the accuracies we discuss in this section are the expected values of the accuracies of the models. We omit that qualification because the discussion otherwise becomes technically baroque. The reader interested in understanding bias, variance, and the trade-off between them might start with the technical but quite readable article by Friedman (1997).

Now we can see why ensemble techniques might work. If we have a modeling method with high variance, averaging over multiple predictions reduces the variance in the predictions. Indeed, ensemble methods tend to improve the predictive ability more for higher-variance methods, such as in cases where you would expect more overfitting (Perlich, Provost, & Simonoff, 2003). Ensemble methods are often used with tree induction, as classification and regression trees tend to have high variance. In the field you may hear about random forests, bagging, and boosting. These are all ensemble methods popular with trees (the latter two are more general). Check out Wikipedia to find out more about them.

Data-Driven Causal Explanation and a Viral Marketing Example

One important topic that we have only touched on in this book (in Chapter 2 and Chapter 11) is *causal explanation* from data. Predictive modeling is extremely useful for many business problems. However, the sort of predictive modeling that we have discussed so far is based on correlations rather than on knowledge of causation. We often want to look more deeply into a phenomenon and ask what influences what. We may want to do this simply to understand our business better, or we may want to use data to improve decisions about how to intervene to cause a desired outcome.

Consider a detailed example. Recently there has been much attention paid to "viral" marketing. One common interpretation of viral marketing is that consumers can be helped to influence each other to purchase a product, and so a marketer can get significant benefit by "seeding" certain consumers (e.g., by giving them the product for free), and they then will be "influencers"— they will cause an increase in the likelihood that the people they know will purchase the product. The holy grail of viral marketing is to be able to create campaigns that spread like an epidemic, but the critical assumption behind "virality" is that consumers actually influence each other. How much do they? Data scientists work to measure such influence, by observing in the data whether once a consumer has the product, her social network neighbors indeed have increased likelihood to purchase the product.

Unfortunately, a naive analysis of the data can be tremendously misleading. For important sociological reasons (McPherson, Smith-Lovin, & Cook, 2001), people tend to

cluster in social networks with people who are similar to them. Why is this important? This means that social network neighbors are likely to have similar product preferences, and we would *expect* the neighbors of people who choose or like a product also to choose or like the product, *even in the absence of any causal influence* among the consumers! Indeed, based on the careful application of causal analysis, it was shown in the *Proceedings of the National Academy of Sciences* (Aral, Muchnik, & Sundararajan, 2009) that traditional methods for estimating the influence in viral marketing analysis overestimated the influence by at least 700%!

There are various methods for careful causal explanation from data, and they can all be understood within a common data science framework. The point of discussing this here toward the end of the book is that understanding these sophisticated techniques requires a grasp of the fundamental principles presented so far. Careful causal data analysis requires the understanding of investments in acquiring data, of similarity measurements, of expected value calculations, of correlation and finding informative variables, of fitting equations to data, and more.

Chapter 11 gave a taste of this more sophisticated causal analysis when we returned to the telecommunications churn problem and asked: shouldn't we be targeting those customers whom the special offer is most likely to influence? This illustrated the key role the expected value framework played, along with several other concepts. There are other techniques for causal understanding that use similarity matching (Chapter 6) to simulate the "counterfactual" that someone might both receive a "treatment" (e.g., an incentive to stay) and not receive the treatment. Still other causal analysis methods fit numeric functions to data and interpret the coefficients of the functions.[10]

The point is that we cannot understand causal data science without first understanding the fundamental principles. Causal data analysis is just one such example; the same applies to other more sophisticated methods you may encounter.

Summary

There are very many specific techniques used in data science. To achieve a solid understanding of the field, it is important to step back from the specifics and think about the sorts of tasks to which the techniques are applied. In this book, we have focused on a collection of the most common tasks (finding correlations and informative attributes, finding similar data items, classification, probability estimation, regression, clustering), showing that the concepts of data science provide a firm foundation for understanding

10. It is beyond the scope of this book to explain the conditions under which this can be given a causal interpretation. But if someone presents a regression equation to you with a causal interpretation of the equation's parameters, ask questions about exactly what the coefficients mean and why one can interpret them causally until you are satisfied. For such analyses, comprehension by decision makers is paramount; insist that you understand any such results.

both the tasks and the methods for solving the tasks. In this chapter, we presented several other important data science tasks and techniques, and illustrated that they too can be understood based on the foundation provided by our fundamental concepts.

Specifically, we discussed: finding interesting co-occurrences or associations among items, such as purchases; profiling typical behavior, such as credit card usage or customer wait time; predicting links between data items, such as potential social connections between people; reducing our data to make it more manageable or to reveal hidden information, such as latent movie preferences; combining models as if they were experts with different expertise, for example to improve movie recommendations; and drawing causal conclusions from data, such as whether and to what extent the fact that socially connected people buy the same products is actually because they influence each other (necessary for viral campaigns), or simply because socially connected people have very similar tastes (which is well known in sociology). A solid understanding of the basic principles helps you to understand more complex techniques as instances or combinations of them.

CHAPTER 13

Data Science and Business Strategy

Fundamental concepts: *Our principles as the basis of success for a data-driven business; Acquiring and sustaining competitive advantage via data science; The importance of careful curation of data science capability.*

In this chapter we discuss the interaction between data science and business strategy, including a high-level perspective on choosing problems to be solved with data science. We see that the fundamental concepts of data science allow us to think clearly about strategic issues. We also show how, taken as a whole, the array of concepts is useful for thinking about tactical business decisions such as evaluating proposals for data science projects from consultants or internal data science teams. We also discuss in detail the curation of data science capability.

Increasingly we see stories in the press about how yet another aspect of business has been addressed with a data science-based solution. As we discussed in Chapter 1, a confluence of factors has led contemporary businesses to be strikingly data rich, as compared to their predecessors. But the availability of data alone does not ensure successful data-driven decision-making. How does a business ensure that it gets the most from the wealth of data? The answer of course is manifold, but two important factors are: (i) the firm's management must think data-analytically, and (ii) the management must create a culture where data science, and data scientists, will thrive.

Thinking Data-Analytically, Redux

Criterion (i) does not mean that the managers have to be data scientists. However, managers have to understand the fundamental principles well enough to envision and/or appreciate data science opportunities, to supply the appropriate resources to the data science teams, and to be willing to invest in data and experimentation. Furthermore, unless the firm has on its management team a seasoned, practical data scientist, often the management must steer the data science team carefully to make sure that the team stays on track toward an eventually useful business solution. This is very difficult if the

managers don't really understand the principles. Managers need to be able to ask probing questions of a data scientist, who often can get lost in technical details. We need to accept that each of us has strengths and weaknesses, and as data science projects span so much of a business, a diverse team is essential. Just as we can't expect a manager necessarily to have deep expertise in data science, we can't expect a data scientist *necessarily* to have deep expertise in business solutions. However, an effective data science team involves collaboration between the two, and each needs to have some understanding of the fundamentals of the other's area of responsibility. Just as it would be a Sisyphean task to manage a data science team where the team had no understanding of the fundamental concepts of business, it likewise is extremely frustrating at best, and often a tremendous waste, for data scientists to struggle under a management that does not understand basic principles of data science.

For example, it is not uncommon for data scientists to struggle under a management that (sometimes vaguely) sees the potential benefit of predictive modeling, but does not have enough appreciation for the process to invest in proper training data or in proper evaluation procedures. Such a company may "succeed" in engineering a model that is predictive enough to produce a viable product or service, but will be at a severe disadvantage to a competitor who invests in doing the data science well.

A solid grounding in the fundamentals of data science has much more far-reaching strategic implications. We know of no systematic scientific study, but broad experience has shown that as executives, managers, and investors increase their exposure to data science projects, they see more and more opportunities in turn. We see extreme cases in companies like Google and Amazon (there is a vast amount of data science underlying web search, as well as Amazon's product recommendations and other offerings). Both of these companies eventually built subsequent products offering "big data" and data-science related services to other firms. Many, possibly most, data-science oriented start-ups use Amazon's cloud storage and processing services for some tasks. Google's "Prediction API" is increasing in sophistication and utility (we don't know how broadly used it is).

Those are extreme cases, but the basic pattern is seen in almost every data-rich firm. Once the data science capability has been developed for one application, other applications throughout the business become obvious. Louis Pasteur famously wrote, "Fortune favors the prepared mind." Modern thinking on creativity focuses on the juxtaposition of a new way of thinking with a mind "saturated" with a particular problem. Working through case studies (either in theory or in practice) of data science applications helps prime the mind to see opportunities and connections to new problems that could benefit from data science.

For example, in the late 1980s and early 1990s, one of the largest phone companies had applied predictive modeling—using the techniques we've described in this book—to the problem of reducing the cost of repairing problems in the telephone network and

to the design of speech recognition systems. With the increased understanding of the use of data science for helping to solve business problems, the firm subsequently applied similar ideas to decisions about how to allocate a massive capital investment to best improve its network, and how to reduce fraud in its burgeoning wireless business. The progression continued. Data science projects for reducing fraud discovered that incorporating features based on social-network connections (via who-calls-whom data) into fraud prediction models improved the ability to discover fraud substantially. In the early 2000s, telecommunications firms produced the first solutions using such social connections to improve marketing—and improve marketing it did, showing huge performance lifts over traditional targeted marketing based on socio-demographic, geographic, and prior purchase data. Next, in telecommunications, such social features were added to models for churn prediction, with equally beneficial results. The ideas diffused to the online advertising industry, and there was a subsequent flurry of development of online advertising based on the incorporation of data on online social connections (at Facebook and at other firms in the online advertising ecosystem).

This progression was driven both by experienced data scientists moving among business problems as well as by data science savvy managers and entrepreneurs, who saw new opportunities for data science advances in the academic and business literature.

Achieving Competitive Advantage with Data Science

Increasingly, firms are considering whether and how they can obtain competitive advantage from their data and/or from their data science capability. This is important strategic thinking that should not be superficial, so let's spend some time digging into it.

Data and data science capability are (complementary) strategic assets. Under what conditions can a firm achieve competitive advantage from such an asset? First of all, the asset has to be valuable to the firm. This seems obvious, but note that the value of an asset to a firm depends on the other strategic decisions that the firm has made. Outside of the context of data science, in the personal computer industry in the 1990s, Dell famously got substantial competitive advantage early over industry leader Compaq from using web-based systems to allow customers to configure computers to their personal needs and liking. Compaq could not get the same value from web-based systems. One main reason was that Dell and Compaq had implemented different strategies: Dell already was a direct-to-customer computer retailer, selling via catalogs; web-based systems held tremendous value given this strategy. Compaq sold computers mainly via retail outlets; web-based systems were not nearly as valuable given this alternative strategy. When Compaq tried to replicate Dell's web-based strategy, it faced a severe backlash from its retailers. The upshot is that the value of the new asset (web-based systems) was dependent on each company's other strategic decisions.

The lesson is that we need to think carefully in the business understanding phase as to how data and data science can provide value in the context of our business strategy, and also whether it would do the same in the context of our competitors' strategies. This can identify both possible opportunities and possible threats. A direct data science analogy of the Dell-Compaq example is Amazon versus Borders. Even very early, Amazon's data on customers' book purchases allowed personalized recommendations to be delivered to customers while they were shopping online. Even if Borders were able to exploit its data on who bought what books, its brick-and-mortar retail strategy did not allow the same seamless delivery of data science-based recommendations.

So, a prerequisite for competetive advantage is that the asset be valuable in the context of our strategy. We've already begun to talk about the second set of criteria: in order to gain competitive advantage, competitors either must not possess the asset, or must not be able to obtain the same value from it. We should think both about the data asset(s) and the data science capability. Do we have a unique data asset? If not, do we have an asset the utilization of which is better aligned with our strategy than with the stragegy of our competitors? Or are we better able to take advantage of the data asset due to our better data science capability?

The flip side of asking about achieving competitive advantage with data and data science is asking whether we are at a competitive disadvantage. It may be that the answers to the previous questions are affirmative for our competitors and not for us. In what follows we will assume that we are looking to achieve competitive advantage, but the arguments apply symmetrically if we are trying to achieve parity with a data-savvy competitor.

Sustaining Competitive Advantage with Data Science

The next question is: even if we can achieve competitive advantage, can we *sustain* it? If our competitors can easily duplicate our assets and capabilities, our advantage may be short-lived. This is an especially critical question if our competitors have greater resources than we do: by adopting our strategy, they may surpass us if they have greater resources.

One strategy for competing based on data science is to plan to always keep one step ahead of the competition: always be investing in new data assets, and always be developing new techniques and capabilities. Such a strategy can provide for an exciting and possibly fast-growing business, but generally few companies are able to execute it. For example, you must have confidence that you have one of the best data science teams, since the effectiveness of data scientists has a huge variance, with the best being much more talented than the average. If you have a great team, you may be willing to bet that you can keep ahead of the competition. We will discuss data science teams more below.

The alternative to always keeping one step ahead of the competition is to achieve sustainable competitive advantage due to a competitor's inability to replicate, or their ele-

vated expense of replicating, the data asset or the data science capability. There are several avenues to such sustainability.

Formidable Historical Advantage

Historical circumstances may have placed our firm in an advantageous position, and it may be too costly for competitors to reach the same position. Amazon again provides an outstanding example. In the "Dotcom Boom" of the 1990s, Amazon was able to sell books below cost, and investors continued to reward the company. This allowed Amazon to amass tremendous data assets (such as massive data on online consumers' buying preferences and online product reviews), which then allowed them to create valuable data-based products (such as recommendations and product ratings). These historical circumstances are gone: it is unlikely today that investors would provide the same level of support to a competitor that was trying to replicate Amazon's data asset by selling books below cost for years on end (not to mention that Amazon has moved far beyond books).

This example also illustrates that the data products themselves can increase the cost to competitors of replicating the data asset. Consumers value the data-driven recommendations and product reviews/ratings that Amazon provides. This creates switching costs: competitors would have to provide extra value to Amazon's customers to entice them to shop elsewhere—either with lower prices or with some other valuable product or service that Amazon does not provide. Thus, when the data acquisition is tied directly to the value provided by the data, the resulting virtuous cycle creates a catch-22 for competitors: competitors need customers in order to acquire the necessary data, but they need the data in order to provide equivalent service to attract the customers.

Entrepreneurs and investors might turn this strategic consideration around: what historical circumstances now exist that may not continue indefinitely, and which may allow me to gain access to or to build a data asset more cheaply than will be possible in the future? Or which will allow me to build a data science team that would be more costly (or impossible) to build in the future?

Unique Intellectual Property

Our firm may have unique intellectual property. Data science intellectual property can include novel techniques for mining the data or for using the results. These might be patented, or they might just be trade secrets. In the former case, a competitor either will be unable to (legally) duplicate the solution, or will have an increased expense of doing so, either by licensing our technology or by developing new technology to avoid infringing on the patent. In the case of a trade secret, it may be that the competitor simply does not know how we have implemented our solution. With data science solutions, the actual mechanism is often hidden; with only the result being visible.

Unique Intangible Collateral Assets

Our competitors may not be able to figure out how to put our solution into practice. With successful data science solutions, the actual source of good performance (for example with effective predictive modeling) may be unclear. The effectiveness of a predictive modeling solution may depend critically on the problem engineering, the attributes created, the combining of different models, and so on. It often is not clear to a competitor how performance is achieved in practice. Even if our algorithms are published in detail, many implementation details may be critical to get a solution that works in the lab to work in production.

Furthermore, success may be based on intangible assets such as a company culture that is particularly suitable to the deployment of data science solutions. For example, a culture that embraces business experimentation and the (rigorous) supporting of claims with data will naturally be an easier place for data science solutions to succeed. Alternatively, if developers are encouraged to understand data science, they are less likely to screw up an otherwise top-quality solution. Recall our maxim: *Your model is not what your data scientists design, it's what your engineers implement.*

Superior Data Scientists

Maybe our data scientists simply are much better than our competitors'. There is a huge variance in the quality and ability of data scientists. Even among well-trained data scientists, it is well accepted within the data science community that certain individuals have the combination of innate creativity, analytical acumen, business sense, and perseverence that enables them to create remarkably better solutions than their peers.

This extreme difference in ability is illustrated by the year-after-year results in the KDD Cup data mining competition. Every year, the top professional society for data scientists, the ACM SIGKDD, holds its annual conference (the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining). Each year the conference holds a data mining competition. Some data scientists love to compete, and there are many competitions. The Netflix competition, discussed in Chapter 12, is one of the most famous, and such competitions have even been turned into a crowd-sourcing business (see Kaggle). The KDD Cup is the granddaddy of data mining competitions and has been held every year since 1997. Why is this relevant? Some of the best data scientists in the world participate in these competitions. Depending on the year and the task, hundreds or thousands of competitors try their hand at solving the problem. If data science talent were evenly distributed, then one would think it unlikely to see the same individuals repeatedly winning the competitions. But that's exactly what we see. There are individuals who have been on winning teams repeatedly, sometimes multiple years in a row and for multiple tasks each year (sometimes the competition has more than

one task).[1] The point is that there is substantial variation in the ability even of the best data scientists, and this is illustrated by the "objective" results of the KDD Cup competitions. The upshot is that because of the large variation in ability, the best data scientists can pick and choose the employment opportunities that suit their desires with respect to salary, culture, advancement opportunities, and so on.

The variation in the quality of data scientists is amplified by the simple fact that top-notch data scientists are in high demand. Anyone can call himself a data scientist, and few companies can really evaluate data scientists well as potential hires. This leads to another catch: you need at least one top-notch data scientist to truly evaluate the quality of prospective hires. Thus, if our company has managed to build a strong data science capability, we have a substantial and sustained advantage over competitors who are having trouble hiring data scientists. Further, top-notch data scientists like to work with other top-notch data scientists, which compounds our advantage.

We also must embrace the fact that data science is in part a craft. Analytical expertise takes time to acquire, and all the great books and video lectures alone will not turn someone into a master. The craft is learned by experience. The most effective learning path resembles that in the classic trades: aspiring data scientists work as apprentices to masters. This could be in a graduate program with a top applications-oriented professor, in a postdoctoral program, or in industry working with one of the best industrial data scientists. At some point the apprentice is skilled enough to become a "journeyman," and will then work more independently on a team or even lead projects of her own. Many high-quality data scientists happily work in this capacity for their careers. Some small subset become masters themselves, because of a combination of their talent at recognizing the potential of new data science opportunities (more on that in a moment) and their mastery of theory and technique. Some of these then take on apprentices. Understanding this learning path can help to focus on hiring efforts, looking for data scientists who have apprenticed with top-notch masters. It also can be used tactically in a less obvious way: if you can hire one master data scientist, top-notch aspiring data scientists may come to apprentice with her.

In addition to all this, a top-notch data scientist needs to have a strong professional network. We don't mean a network in the sense of what one might find in an online professional networking system; an effective data scientist needs to have deep connections to other data scientists throughout the data science community. The reason is simply that the field of data science is immense and there are far too many diverse topics for any individual to master. A top-notch data scientist is a master of some area of technical expertise, and is familiar with many others. (Beware of the "jack-of-all-trades, master of none.") However, we do not want the data scientist's mastery of some area of

1. This is not to say that one should look at the KDD Cup winners as necessarily the best data miners in the world. Many top-notch data scientists have never competed in such a competition; some compete once and then focus their efforts on other things.

technical expertise to turn into the proverbial hammer for which all problems are nails. A top-notch data scientist will pull in the necessary expertise for the problem at hand. This is facilitated tremendously by strong and deep professional contacts. Data scientists call on each other to help in steering them to the right solutions. The better a professional network is, the better will be the solution. And, the best data scientists have the best connections.

Superior Data Science Management

Possibly even more critical to success for data science in business is having good *management* of the data science team. Good data science managers are especially hard to find. They need to understand the fundamentals of data science well, possibly even being competent data scientists themselves. Good data science managers also must possess a set of other abilities that are rare in a single individual:

- They need to truly understand and appreciate the needs of the business. What's more, they should be able to anticipate the needs of the business, so that they can interact with their counterparts in other functional areas to develop ideas for new data science products and services.
- They need to be able to communicate well with and be respected by both "techies" and "suits"; often this means translating data science jargon (which we have tried to minimize in this book) into business jargon, and vice versa.
- They need to coordinate technically complex activities, such as the integration of multiple models or procedures with business constraints and costs. They often need to understand the technical architectures of the business, such as the data systems or production software systems, in order to ensure that the solutions the team produces are actually useful in practice.
- They need to be able to anticipate outcomes of data science projects. As we have discussed, data science is more similar to R&D than to any other business activity. Whether a particular data science project will produce positive results is highly uncertain at the outset, and possibly even well into the project. Elsewhere we discuss how it is important to produce proof-of-concept studies quickly, but neither positive nor negative outcomes of such studies are highly predictive of success or failure of the larger project. They just give guidance to investments in the next cycle of the data mining process (recall Chapter 2). If we look to R&D management for clues about data science management, we find that there is only one reliable predictor of the success of a research project, and it is *highly* predictive: the prior success of the investigator. We see a similar situation with data science projects. There are individuals who seem to have an intuitive sense of which projects will pay off. We do not know of a careful analysis of why this is the case, but experience shows that it is. As with data science competitions, where we see remarkable repeat performances by the same individuals, we also see individuals repeatedly envisioning new data

science opportunities and managing them to great success—and this is particularly impressive as many data science managers never see even one project through to great success.

- They need to do all this within the culture of a particular firm.

Finally, our data science capability may be difficult or expensive for a competitor to duplicate because *we can hire data scientists and data science managers better*. This may be due to our reputation and brand appeal with data scientists—a data scientist may prefer to work for a company known as being friendly to data science and data scientists. Or our firm may have a more subtle appeal. So let's examine in a little more detail what it takes to attract top-notch data scientists.

Attracting and Nurturing Data Scientists and Their Teams

At the beginning of the chapter, we noted that the two most important factors in ensuring that our firm gets the most from its data assets are: (i) the firm's management must think data-analytically, and (ii) the firm's management must create a culture where data science, and data scientists, will thrive. As we mentioned above, there can be a huge difference between the effectiveness of a great data scientist and an average data scientist, and between a great data science team and an individually great data scientist. But how can one confidently engage top-notch data scientists? How can we create great teams?

This is a very difficult question to answer in practice. At the time of this writing, the supply of top-notch data scientists is quite thin, resulting in a very competitive market for them. The best companies at hiring data scientists are the IBMs, Microsofts, and Googles of the world, who clearly demonstrate the value they place in data science via compensation, perks, and/or intangibles, such as one particular factor not to be taken lightly: data scientists like to be around other top-notch data scientists. One might argue that they *need* to be around other top-notch data scientists, not only to enjoy their day-to-day work, but also because the field is vast and the collective mind of a group of data scientists can bring to bear a much broader array of particular solution techniques.

However, just because the market is difficult does not mean all is lost. Many data scientists want to have more individual influence than they would have at a corporate behemoth. Many want more responsibility (and the concomitant experience) with the broader process of producing a data science solution. Some have visions of becoming Chief Scientist for a firm, and understand that the path to Chief Scientist may be better paved with projects in smaller and more varied firms. Some have visions of becoming entrepreneurs, and understand that being an early data scientist for a startup can give them invaluable experience. And some simply will enjoy the thrill of taking part in a fast-growing venture: working in a company growing at 20% or 50% a year is much different from working in a company growing at 5% or 10% a year (or not growing at all).

In all these cases, the firms that have an advantage in hiring are those that create an environment for nurturing data science and data scientists. If you do not have a critical mass of data scientists, be creative. Encourage your data scientists to become part of local data science technical communities and global data science academic communities.

A note on publishing

Science is a social endeavor, and the best data scientists often want to stay engaged in the community by publishing their advances. Firms sometimes have trouble with this idea, feeling that they are "giving away the store" or tipping their hand to competitors by revealing what they are doing. On the other hand, if they do not, they may not be able to hire or retain the very best. Publishing also has some advantages for the firm, such as increased publicity, exposure, external validation of ideas, and so on. There is no clear-cut answer, but the issue needs to be considered carefully. Some firms file patents aggressively on their data science ideas, after which academic publication is natural if the idea is truly novel and important.

A firm's data science presence can be bolstered by engaging academic data scientists. There are several ways of doing this. For those academics interested in practical applications of their work, it may be possible to fund their research programs. Both of your authors, when working in industry, funded academic programs and essentially extended the data science team that was focusing on their problems and interacting. The best arrangement (by our experience) is a combination of data, money, and an interesting business problem; if the project ends up being a portion of the Ph.D. thesis of a student in a top-notch program, the benefit to the firm can far outweigh the cost. Funding a Ph.D. student might cost a firm in the ballpark of $50K/year, which is a fraction of the fully loaded cost of a top data scientist. A key is to have enough understanding of data science to select the right professor—one with the appropriate expertise for the problem at hand.

Another tactic that can be very cost-effective is to take on one or more top-notch data scientists as scientific advisors. If the relationship is structured such that the advisors truly interact on the solutions to problems, firms that do not have the resources or the clout to hire the very best data scientists can substantially increase the quality of the eventual solutions. Such advisors can be data scientists at partner firms, data scientists from firms who share investors or board members, or academics who have some consulting time.

A different tack altogether is to hire a third party to conduct the data science. There are various third-party data science providers, ranging from massive firms specializing in business analytics (such as IBM), to data-science-specific consulting firms (such as Elder

Research), to boutique data science firms who take on a very small number of clients to help them develop their data science capabilities (such as Data Scientists, LLC).[2] You can find a large list of data-science service companies, as well as a wide variety of other data science resources, at KDnuggets. A caveat about engaging data science consulting firms is that their interests are not always well aligned with their customers' interests; this is obvious to seasoned users of consultants, but not to everyone.

Savvy managers employ all of these resources tactically. A chief scientist or empowered manager often can assemble for a project a substantially more powerful and diverse team than most companies can hire.

Examine Data Science Case Studies

Beyond building a solid data science team, how can a manager ensure that her firm is best positioned to take advantage of opportunities for applying data science? Make sure that there is an understanding of and appreciation for the fundamental principles of data science. Empowered employees across the firm often see novel applications.

After gaining command of the fundamental principles of data science, the best way to position oneself for success is to work through many examples of the application of data science to business problems. Read case studies that actually walk through the data mining process. Formulate your own case studies. Actually mining data is helpful, but even more important is working through the connection between the business problem and the possible data science solutions. The more, different problems you work through, the better you will be at naturally seeing and capitalizing on opportunities for bringing to bear the information and knowledge "stored" in the data—often the same problem formulation from one problem can be applied by analogy to another, with only minor changes.

It is important to keep in mind that the examples we have presented in this book were chosen or designed for illustration. In reality, the business and data science team should be prepared for all manner of mess and contraints, and must be flexible in dealing with them. Sometimes there is a wealth of data and data science techniques available to be brought to bear. Other times the situation seems more like the critical scene from the movie *Apollo 13*. In the movie, a malfunction and explosion in the command module leave the astronauts stranded a quarter of a million miles from Earth, with the CO_2 levels rising too rapidly for them to survive the return trip. In a nutshell, because of the constraints placed by what the astronauts have on hand, the engineers have to figure out how to use a large cubic filter in place of a narrower cylindrical filter (to literally put a square peg in a round hole). In the key scene, the head engineer dumps out onto a table all the "stuff" that's there in the command module, and tells his team: "OK, people …

2. Disclaimer: The authors have a relationship with Data Scientists, LLC.

we got to find a way to make *this* fit into the hole for *this*, using nothing but *that*." Real data science problems often seem more like the Apollo 13 situation than a textbook situation.

For example, Perlich et al. (2013) describe a study of just such a case. For targeting consumers with online display advertisements, obtaining an adequate supply of the ideal training data would have been prohibitively expensive. However, data were available at much lower cost from various other distributions and for other target variables. Their very effective solution cobbled together models built from these surrogate data, and "transferred" these models for use on the desired task. The use of these surrogate data allowed them to operate with a substantially reduced investment in data from the ideal (and expensive) training distribution.

Be Ready to Accept Creative Ideas from Any Source

Once different role players understand fundamental principles of data science, creative ideas for new solutions can come from any direction—such as from executives examining potential new lines of business, from directors dealing with profit and loss responsibility, from managers looking critically at a business process, and from line employees with detailed knowledge of exactly how a particular business process functions. Data scientists should be encouraged to interact with employees throughout the business, and part of their performance evaluation should be based on how well they produce ideas for improving the business with data science. Incidentally, doing so can pay off in unintended ways: the data processing skills possessed by data scientists often can be applied in ways that are not so sophisticated but nevertheless can help other employees without those skills. Often a manager may have no idea that particular data can even be obtained—data that might help the manager directly, without sophisticated data science.

Be Ready to Evaluate Proposals for Data Science Projects

Ideas for improving business decisions through data science can come from any direction. Managers, investors, and employees should be able to formulate such ideas clearly, and decision makers should be prepared to evaluate them. Essentially, we need to be able to formulate solid proposals and to evaluate proposals.

The data mining process, described in Chapter 2, provides a framework to direct this. Each stage in the process reveals questions that should be asked both in formulating proposals for projects and in evaluating them:

- Is the business problem well specified? Does the data science solution solve the problem?
- Is it clear how we would evaluate a solution?

- Would we be able see evidence of success before making a huge investment in deployment?
- Does the firm have the data assets it needs? For example, for supervised modeling, are there actually labeled training data? Is the firm ready to invest in the assets it does not have yet?

Appendix A provides a starting list of questions for evaluating data science proposals, organized by the data mining process. Let's walk through an illustrative example. (In Appendix B you will find another example proposal to evaluate, focusing on our running churn problem.)

Example Data Mining Proposal

Your company has an installed user base of 900,000 current users of your Whiz-bang® widget. You now have developed Whiz-bang® 2.0, which has substantially lower operating costs than the original. Ideally, you would like to convert ("migrate") your entire user base over to version 2.0; however, using 2.0 requires that users master the new interface, and there is a serious risk that in attempting to do so, the customers will become frustrated and not convert, become less satisfied with the company, or in the worst case, switch to your competitor's popular Boppo® widget. Marketing has designed a brand-new migration incentive plan, which will cost $250 per selected customer. There is no guarantee that a customer will choose to migrate even if she takes this incentive.

An external firm, Big Red Consulting, is proposing a plan to target customers carefully for Whiz-bang® 2.0, and given your demonstrated fluency with the fundamentals of data science, you are called in to help assess Big Red's proposal. Do Big Red's choices seem correct?

> **Targeted Whiz-bang Customer Migration—prepared by Big Red Consulting, Inc.**
>
> We will develop a predictive model using modern data-mining technology. As discussed in our last meeting, we assume a budget of $5,000,000 for this phase of customer migration; adjusting the plan for other budgets is straightforward. Thus we can target 20,000 customers under this budget. Here is how we will select those customers:
>
> We will use data to build a model of whether or not a customer will migrate given the incentive. The dataset will comprise a set of attributes of customers, such as the number and type of prior customer service interactions, level of usage of the widget, location of the customer, estimated technical sophistication, tenure with the firm, and other loyalty indicators, such as number of other firm products and services in use. The target will be whether or not the customer will migrate to the new widget if he/she is given the incentive. Using these data, we will build a linear regression to estimate the target variable. The model will be evaluated based on its accuracy on these data; in particular, we want to ensure that the accuracy is substantially greater than if we targeted randomly.
>
> To use the model: for each customer we will apply the regression model to estimate the target variable. If the estimate is greater than 0.5, we will predict that the customer will migrate; otherwise, we will say the customer will not migrate. We then will select at ran-

dom 20,000 customers from those predicted to migrate, and these 20,000 will be the recommended targets.

Flaws in the Big Red Proposal

We can use our understanding of the fundamental principles and other basic concepts of data science to identify flaws in the proposal. Appendix A provides a starting guide for reviewing such proposals, with some of the main questions to ask. However, this book as a whole really can be seen as a proposal review guide. Here are some of the most egregious flaws in Big Data's proposal:

Business Understanding

- The target variable definition is imprecise. For example, over what time period must the migration occur? (Chapter 3)
- The formulation of the data mining problem could be better-aligned with the business problem. For example, what if certain customers (or everyone) were likely to migrate anyway (without the incentive)? Then we would be wasting the cost of the incentive in targeting them. (Chapter 2, Chapter 11)

Data Understanding/Data Preparation

- There aren't any labeled training data! This is a brand-new incentive. We should invest some of our budget in obtaining labels for some examples. This can be done by targeting a (randomly) selected subset of customers with the incentive. One also might propose a more sophisticated approach (Chapter 2, Chapter 3, Chapter 11).
- If we are worried about wasting the incentive on customers who are likely to migrate without it, we also should observe a "control group" over the period where we are obtaining training data. This should be easy, since everyone we don't target to gather labels would be a "control" subject. We can build a separate model for migrate or not given no incentive, and combine the models in an expected value framework. (Chapter 11)

Modeling

- Linear regression is not a good choice for modeling a categorical target variable. Rather one should use a classification method, such as tree induction, logistic regression, k-NN, and so on. Even better, why not try a bunch of methods and evaluate them experimentally to see which performs best? (Chapter 2, Chapter 3, Chapter 4, Chapter 5, Chapter 6, Chapter 7, Chapter 8)

Evaluation

- The evaluation shouldn't be on the training data. Some sort of holdout approach should be used (e.g., cross-validation and/or a staged approach as discussed above). (Chapter 5)
- Is there going to be any domain-knowledge validation of the model? What if it is capturing some weirdness of the data collection process? (Chapter 7, Chapter 11, Chapter 14)

Deployment

- The idea of randomly selecting customers with regression scores greater than 0.5 is not well considered. First, it is not clear that a regression score of 0.5 really corresponds to a probability of migration of 0.5. Second, 0.5 is rather arbitrary in any case. Third, since our model is providing a ranking (e.g., by likelihood of migration, or by expected value if we use the more complex formulation), we should use the ranking to guide our targeting: choose the top-ranked candidates, as the budget will allow. (Chapter 2, Chapter 3, Chapter 7, Chapter 8, Chapter 11)

Of course, this is just one example with a particular set of flaws. A different set of concepts may need to be brought to bear for a different proposal that is flawed in other ways.

A Firm's Data Science Maturity

For a firm to realistically plan data science endeavors it should assess, frankly and rationally, its own *maturity* in terms of data science capability. It is beyond the scope of this book to provide a self-assessment guide, but a few words on the topic are important.

Firms vary widely in their data science capabilities along many dimensions. One dimension that is very important for strategic planning is the firm's "maturity," specifically, how systematic and well founded are the processes used to guide the firm's data science projects.[3]

At one end of the maturity spectrum, a firm's data science processes are completely ad hoc. In many firms, the employees engaged in data science and business analytics endeavors have no formal training in these areas, and the managers involved have little understanding of the fundamental principles of data science and data analytic thinking.

3. The reader interested in this notion of the maturity of a firm's capabilities is encouraged to read about the Capability Maturity Model for software engineering, which is the inspiration for this discussion.

A note on "immature" firms

Being "immature" does *not* mean that a firm is destined to failure. It means that success is highly variable and is much more dependent on luck than in a mature firm. Project success will depend upon the heroic efforts by individuals who happen to have a natural acuity for data-analytic thinking. An immature firm may implement not-so-sophisticated data science solutions at a large scale, or may implement sophisticated solutions at a small scale. Rarely, though, will an immature firm implement sophisticated data science solutions at a large scale.

A firm with a medium level of maturity employs well-trained data scientists, as well as business managers and other stakeholders who understand the fundamental principles of data science. Both sides can think clearly about how to solve business problems with data science, and both sides participate in the design and implementation of solutions that directly address the problems of the business.

At the high end of maturity are firms who continually work to improve their data science *processes* (and not just the solutions). Executives at such firms continually challenge the data science team to instill processes that will align their solutions better with the business problems. At the same time they realize that pragmatic trade-offs may favor the choice of a suboptimal solution that can be realized today over a theoretically much better solution that won't be ready until next year. Data scientists at such a firm should have the confidence that when they propose an investments to improve data science processes, their suggestions will be met with open and informed minds. That's not to say that every such request will be approved, but that the proposal will be evaluated on its own merits in the context of the business.

Note: Data science is neither operations nor engineering.

There is some danger in making an analogy to the Capability Maturity Model from software engineering—danger that the analogy will be taken too literally. Trying to apply the same sort of processes that work for software engineering, or worse for manufacturing or operations, will fail for data science. Moreover, misguided attempts to do so will send a firm's best data scientists out the door before the management even knows what happened. The key is to understand *data science processes* and how to data science well, and work to establish consistency and support. Remember that data science is more like R&D than like engineering or manufacturing. As a concrete example, management should consistently make available the resources needed for solid evaluation of data science projects early and often. Sometimes this involves investing in data that would not otherwise have been obtained. Often this involves assigning engineering resources to sup-

> port the data science team. The data science team should in return work to provide management with evaluations that are as well aligned with the actual business problem(s) as possible.

As a concrete example, consider yet again our telecom churn problem and how firms of varying maturity might address it:

- An immature firm will have (hopefully) analytically adept employees implementing ad hoc solutions based on their intuitions about how to manage churn. These may work well or they may not. In an immature firm, it will be difficult for management to evaluate these choices against alternatives, or to determine when they've implemented a nearly optimal solution.
- A firm of medium maturity will have implemented a well-defined framework for testing different alternative solutions. They will test under conditions that mimic as closely as possible the actual business setting—for example, running the latest production data through a testbed platform that compares how different methods "would have done," and considering carefully the costs and benefits involved.
- A very mature organization may have deployed the exact same methods as the medium-maturity firm for identifying the customers with the highest probability of leaving, or even the highest expected loss if they were to churn. They would also be working to implement the processes, and gather the data, necessary to judge also the effect of the incentives and thereby work towards finding those individuals for which the incentives will produce the largest expected increase in value (over not giving the incentive). Such a firm may also be working to integrate such a procedure into an experimentation and/or optimization framework for assessing different offers or different parameters (like the level of discount) to a given offer.

A frank self-assessment of data science maturity is difficult, but it is essential to getting the best out of one's current capabilities, and to improving one's capabilities.

CHAPTER 14
Conclusion

If you can't explain it simply, you don't understand it well enough.

—Albert Einstein

The practice of data science can best be described as a combination of analytical engineering and exploration. The business presents a problem we would like to solve. Rarely is the business problem directly one of our basic data mining tasks. We decompose the problem into subtasks that we think we can solve, usually starting with existing tools. For some of these tasks we may not know how *well* we can solve them, so we have to mine the data and conduct evaluation to see. If that does not succeed, we may need to try something completely different. In the process we may discover knowledge that will help us to solve the problem we had set out to solve, or we may discover something unexpected that leads us to other important successes.

Neither the analytical engineering nor the exploration should be omitted when considering the application of data science methods to solve a business problem. Omitting the engineering aspect usually makes it much less likely that the results of mining data will actually solve the business problem. Omitting the understanding of process as one of exploration and discovery often keeps an organization from putting the right management, incentives, and investments in place for the project to succeed.

The Fundamental Concepts of Data Science

Both the analytical engineering and the exploration and discovery are made more systematic and thereby more likely to succeed by the understanding and embracing of the fundamental concepts of data science. In this book we have introduced a collection of the most important fundamental concepts. Some of these concepts we made into headliners for the chapters, and others were introduced more naturally through the discus-

sions (and not necessarily labeled as fundamental concepts). These concepts span the process from envisioning how data science can improve business decisions, to applying data science techniques, to deploying the results to improve decision-making. The concepts also undergird a large array of business analytics.

We can group our fundamental concepts roughly into three types:

1. General concepts about how data science fits in the organization and the competitive landscape, including ways to attract, structure, and nurture data science teams, ways for thinking about how data science leads to competitive advantage, ways that competitive advantage can be sustained, and tactical principles for doing well with data science projects.
2. General ways of thinking data-analytically, which help us to gather appropriate data and consider appropriate methods. The concepts include the *data mining process*, the collection of different *high-level data science tasks*, as well as principles such as the following.
    - *Data should be considered an asset, and therefore we should think carefully about what investments we should make to get the best leverage from our asset*
    - *The expected value framework can help us to structure business problems so we can see the component data mining problems as well as the connective tissue of costs, benefits, and constraints imposed by the business environment*
    - *Generalization and overfitting: if we look too hard at the data, we will find patterns; we want patterns that generalize to data we have not yet seen*
    - *Applying data science to a well-structured problem versus exploratory data mining require different levels of effort in different stages of the data mining process*
3. General concepts for actually extracting knowledge from data, which undergird the vast array of data science techniques. These include concepts such as the following.
    - *Identifying informative attributes—those that correlate with or give us information about an unknown quantity of interest*
    - *Fitting a numeric function model to data by choosing an objective and finding a set of parameters based on that objective*
    - *Controlling complexity is necessary to find a good trade-off between generalization and overfitting*
    - *Calculating similarity between objects described by data*

Once we think about data science in terms of its fundamental concepts, we see the same concepts underlying many different data science strategies, tasks, algorithms, and processes. As we have illustrated throughout the book, these principles not only allow us to understand the theory and practice of data science much more deeply, they also allow us to understand the methods and techniques of data science very broadly, because these methods and techniques are quite often simply particular instantiations of one or more of the fundamental principles.

At a high level we saw how structuring business problems using the expected value framework allows us to decompose problems into data science tasks that we understand better how to solve, and this applies across many different sorts of business problems.

For extracting knowledge from data, we saw that our fundamental concept of determining the similarity of two objects described by data is used directly, for example to find customers similar to our best customers. It is used for classification and for regression, via nearest-neighbor methods. It is the basis for clustering, the unsupervised grouping of data objects. It is the basis for finding documents most related to a search query. And it is the basis for more than one common method for making recommendations, for example by casting both customers and movies into the same "taste space," and then finding movies most similar to a particular customer.

When it comes to measurement, we see the notion of *lift*—determining how much more likely a pattern is than would be expected by chance—appearing broadly across data science, when evaluating very different sorts of patterns. One evaluates algorithms for targeting advertisements by computing the lift one gets for the targeted population. One calculates lift for judging the weight of evidence for or against a conclusion. One calculates lift to help judge whether a repeated co-occurrence is interesting, as opposed to simply being a natural consequence of popularity.

Understanding the fundamental concepts also facilitates communication between business stakeholders and data scientists, not only because of the shared vocabulary, but because both sides actually understand better. Instead of missing important aspects of a discussion completely, we can dig in and ask questions that will reveal critical aspects that otherwise would not have been uncovered.

For example, let's say your venture firm is considering investing in a data science-based company producing a personalized online news service. You ask how exactly they are personalizing the news. They say they use support vector machines. Let's even pretend that we had not talked about support vector machines in this book. You should feel confident enough in your knowledge of data science now that you should not simply say "Oh, OK." You should be able to confidently ask: "What's that exactly?" If they really do know what they are talking about, they should give you some explanation based upon our fundamental principles (as we did in Chapter 4). You also are now prepared to ask, "What exactly are the training data you intend to use?" Not only might that impress data scientists on their team, but it actually is an important question to be asked to see

whether they are doing something credible, or just using "data science" as a smokescreen to hide behind. You can go on to think about whether you really believe building any predictive model from these data—regardless of what sort of model it is—would be likely to solve the business problem they're attacking. You should be ready to ask whether you really think they will have reliable training labels for such a task. And so on.

Applying Our Fundamental Concepts to a New Problem: Mining Mobile Device Data

As we've emphasized repeatedly, once we think about data science as a collection of concepts, principles, and general methods, we will have much more success both understanding data science activities broadly, and also applying data science to new business problems. Let's consider a fresh example.

Recently (as of this writing), there has been a marked shift in consumer online activity from traditional computers to a wide variety of mobile devices. Companies, many still working to understand how to reach consumers on their desktop computers, now are scrambling to understand how to reach consumers on their mobile devices: smart phones, tablets, and even increasingly mobile laptop computers, as WiFi access becomes ubiquitous. We won't talk about most of the complexity of that problem, but from our perspective, the data-analytic thinker might notice that mobile devices provide a new sort of data from which little leverage has yet been obtained. In particular, mobile devices are associated with data on their location.

For example, in the mobile advertising ecosystem, depending on my privacy settings, my mobile device may broadcast my exact GPS location to those entities who would like to target me with advertisements, daily deals, and other offers. Figure 14-1 shows a scatterplot of a small sample of locations that a potential advertiser might see, sampled from the mobile advertising ecosystem. Even if I do not broadcast my GPS location, my device broadcasts the IP address of the network it currently is using, which often conveys location information.

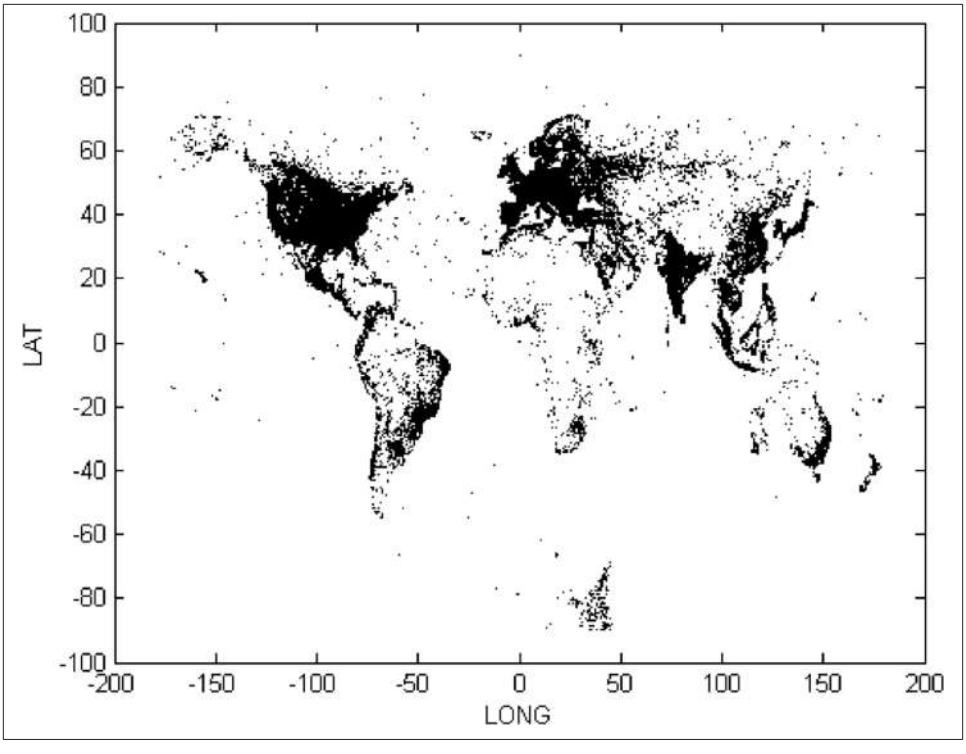

Figure 14-1. A scatterplot of a sample of GPS locations captured from mobile devices.

As an interesting side point, this is just a scatterplot of the latitude and longitudes broadcast by mobile devices; *there is no map!* It gives a striking picture of population density across the world. And it makes us wonder what's going on with mobile devices in Antarctica.

How might we use such data? Let's apply our fundamental concepts. If we want to get beyond exploratory data analysis (as we've started with the visualization in Figure 14-1), we need to think in terms of some concrete business problem. A particular firm might have certain problems to solve, and be focused on one or two. An entrepreneur or investor might scan across different possible problems she sees that businesses or consumers currently have. Let's pick one related to these data.

Advertisers face the problem that in this new world, we see a variety of different devices and a particular consumer's behavior may be fragmented across several. In the desktop world, once the advertisers identify a good prospect, perhaps via a cookie in a particular consumer's browser or a device ID, they can then begin to take action accordingly; for example, by presenting targeted ads. In the mobile ecosystem, this consumer's activity

is fragmented across devices. Even if a good prospect is found on one device, how can she be targeted on her other devices?

One possibility is to use the location data to winnow the space of possible other devices that could belong to this prospect. Figure 14-1 suggests that a huge portion of the space of possible alternatives would be eliminated if we could profile the location visitation behavior of a mobile device. Presumably, my location behavior on my smart phone will be fairly similar to my location behavior on my laptop, especially if I am considering the WiFi locations that I use.[1] So I may want to draw on what I know about assessing the similarity of data items (Chapter 6).

When working through our data-understanding phase, we need to decide how exactly we will represent devices and their locations. Once we are able to step back from the details of algorithms and applications, and think instead about the fundamentals, we might notice that the ideas discussed in the example of problem formulation for text mining (Chapter 10) would apply very well here—even though this example has nothing to do with text. When mining data on documents, we often ignore much of the structure of the text, such as its sequence. For many problems we can simply treat each document as a collection of words from a potentially large vocabulary. The same thinking will apply here. Obviously there is considerable structure to the locations one visits, such as the sequence in which they are visited, but for data mining a simplest-first strategy is often best. Let's just consider each device to be a "bag of locations," in analogy to the bag-of-words representation discussed in Chapter 10.

If we are looking to try to find other instances of the same user, we might also profitably apply the ideas of TFIDF for text to our locations. WiFi locations that are very popular (like the Starbucks on the corner of Washington Square Park) are unlikely to be so informative in a similarity calculation focused on finding the same user on different devices. Such a location would get a low IDF score (think of the "D" as being for "**D**evice" rather than "Document"). At the other end of the spectrum, for many people their apartment WiFi networks would have few different devices, and thereby be quite discriminative. TFIDF on location would magnify the importance of these locations in a similarity calculation. In between these two in discriminability might be an office WiFi network, which might get a middle-of-the-road IDF score.

Now, if our device profile is a TFIDF representation based on our bag of locations, as with using similarity over the TFIDF formulation for our search query for the jazz musician example in Chapter 10, we might look for the devices most similar to the one that we had identified as a good prospect. Let's say that my laptop was the device identified as a good prospect. My laptop is observed on my apartment WiFi network and on my work WiFi network. The only other devices that are observed there are my phone, my tablet, and possibly the mobile devices of my wife and a few friends and colleagues

1. Which incidentally can be anonymized if I am concerned about invasions of privacy. More on that later.

(but note that these will get low TF scores at one or the other location, as compared to my devices). Thus, it is likely that my phone and tablet will be strongly similar—possibly most similar—to the one identified as a prospect. If the advertiser had identified my laptop as a good prospect for a particular ad, then this formulation would also identify my phone and tablet as good prospects for the same ad.

This example isn't meant to be a definitive solution to the problem of finding corresponding users on different mobile devices;[2] it shows how having a conceptual toolkit can be helpful in thinking about a brand-new problem. Once these ideas are conceptualized, data scientists would dig in to figure out what really works and how to flesh out and extend the ideas, applying many of the concepts we have discussed (such as how to evaluate alternative implementation options).

Changing the Way We Think about Solutions to Business Problems

The example also provides a concrete illustration of yet another important fundamental concept (we haven't exhausted them even after this many pages of a detailed book). It is quite common that in the business understanding/data understanding subcycle of the data mining process, our notion of *what is the problem* changes to fit what we actually can do with the data. Often the change is subtle, but it is very important to (try to) notice when it happens. Why? Because all stakeholders are not involved with the data science problem formulation. If we forget that we have changed the problem, especially if the change is subtle, we may run into resistance down the line. And it may be resistance due purely to misunderstanding! What's worse, it may be perceived as due to stubbornness, which might lead to hard feelings that threaten the success of the project.

Let's look back at the mobile targeting example. The astute reader might have said: *Wait a minute. We started by saying that we were going to find the same users on different devices. What we've done is to find very similar users in terms of their location information. I may be willing to agree that the set of these similar users is very likely to contain the same user—more likely than any alternative I can think of—but that's not the same as finding the same user on different devices.* This reader would be correct. In working through our problem formulation the problem changed slightly. We now have made the identification of the same user probabilistic: there may be a very high probability that the subset of devices with very similar location profiles will contain other instances of the same user, but it is not guaranteed. This needs to be clear in our minds, and clarified to stakeholders.

It turns out that for targeting advertisements or offers, this change probably will be acceptable to all stakeholders. Recalling our cost/benefit framework for evaluating data mining solutions (Chapter 7), it's pretty clear that for many offers targeting some false

2. It is however the essence of a real-world solution to the problem implemented by one of the most advanced mobile advertising companies.

positives will be of relatively low cost as compared to the benefit of hitting more true positives. What's more, for many offers targeters may actually be happy to "miss," if each miss constitutes hitting other people with similar interests. And my wife and close friends and colleagues are pretty good hits for many of my tastes and interests![3]

What Data Can't Do: Humans in the Loop, Revisited

This book has focused on how, why, and when we can get business value from data science by enhancing data-driven decision-making. It is important also to consider the limits of data science and data-driven decision-making.

There are things computers are good at and things people are good at, but often these aren't the same things. For example, humans are much better at identifying—from everything out in the world—small sets of relevant aspects of the world from which to gather data in support of a particular task. Computers are much better at sifting through a massive collection of data, including a huge number of (possibly) relevant variables, and quantifying the variables' relevance to predicting a target.

New York Times Op-Ed columnist David Brooks has written an excellent essay entitled "What Data Can't Do" (Brooks, 2013). You should read this if you are considering the magical application of data science to solve your problems.

Data science involves the judicious integration of human knowledge and computer-based techniques to achieve what neither of them could achieve alone. (And beware of any tool vendor who suggests otherwise!) The data mining process introduced in Chapter 2 helps direct the combination of humans and computers. The structure imposed by the process emphasizes the interaction of humans early, to ensure that the application of data science methods are focused on the right tasks. Examining the data mining process also reveals that task selection and specification is not the only place where human interaction is critical. As discussed in Chapter 2, one of the places where human creativity, knowledge, and common sense adds value is in selecting the right data to mine—which is far too often overlooked in discussions of data mining, especially considering its importance.

3. In an article in the *Proceedings of the National Academy of Sciences*, Crandall et al. (2010) show that geographic co-ocurrences between individuals are very strongly predictive of the individuals being friends: "The knowledge that two people were proximate at just a few distinct locations at roughly the same times can indicate a high conditional probability that they are directly linked in the underlying social network." This means that even "misses" due to location similarity may still contain some of the advantage of social network targeting —which has been shown to be extremely effective for marketing (Hill et al., 2006).

Human interaction is also critical in the evaluation stage of the process. The combination of the right data and data science techniques excels at finding models that optimize some objective criterion. Only humans can tell what is the best objective criterion to optimize for a particular problem. This involves substantial subjective human judgment, because often the true criterion to be optimized cannot be measured, so the humans have to pick the best proxy or proxies possible—and keep these decisions in mind as sources of risk when the models are deployed. And then we need careful, and sometimes creative, attention to whether the resultant models or patterns actually do help solve the problem.

We also need to keep in mind that the data to which we will apply data science techniques are the product of some process that involved human decisions. We should not fall prey to thinking that the data represent objective truth.[4] Data incorporate the beliefs, purposes, biases, and pragmatics of those who designed the data collection systems. The meaning of data is colored by our own beliefs.

Consider the following simple example. Many years ago, your authors worked together as data scientists at one of the largest telephone companies. There was a terrible problem with fraud in the wireless business, and we applied data science methods to massive data on cell phone usage, social calling patterns, locations visited, etc. (Fawcett & Provost, 1996, 1997). A seemingly well-performing component of a model for detecting fraud indicated that "calling from cellsite number zero provides substantially increased risk of fraud." This was verified through careful holdout evaluation. Fortunately (in this instance), we followed good data science practice and in the evaluation phase worked to ensure domain-knowledge validation of the model. We had trouble understanding this particular model component. There were many cellsites that indicated elevated probability of fraud,[5] but cellsite zero stood out. Furthermore, the other cellsites made sense because when you looked up their locations, there at least was a good story—for example, the cellsite was in a high-crime area. Looking up cellsite zero resulted in nothing at all. It wasn't in the cellsite lists. We went to the top data guru to divine the answer. Indeed, *there was no cellsite zero*. But the data clearly have many fraudulent calls from cellsite zero!

4. The philosophically minded should read W. V. O. Quine's (1951) classic essay, "Two Dogmas of Empiricism," in which he presents a biting criticism of the common notion that there is a dichotomy between the empirical and the analytical.

5. Technically, the models were most useful if there was a significant *change* in behavior to more calling from these cellsites. If you are interested, the papers describe this in detail.

To make a quite long story short, our understanding of the data was wrong. Briefly, when fraud was resolved on a customer's account, often a substantial amount of time passed between their bill being printed, sent out, received by the customer, opened, read, and acted upon. During this time, fraudulent activity continued. Now that fraud had been detected, these calls should not appear on the customer's next bill, so they were removed from the billing system. They were not discarded however, but (fortunately for the data mining efforts) were kept in a different database. Unfortunately, whoever designed that database decided that it was not important to keep certain fields. One of these was the cellsite. Thus, when the data science effort asked for data on all the fraudulent calls, in order to build training and test sets, these calls were included. As they did not have a cellsite, another design decision (conscious or not) led the fields to be filled with zeros. Thus, many of the fraudulent calls seemed to be from cellsite zero!

This is a "leak," as introduced in Chapter 2. You might think that should have been easy to spot. It wasn't, for several reasons. Consider how many phone calls are made by *tens of millions* of customers over many months, and for each call there was a very large number of possible descriptive attributes. There was no possibility to manually examine the data. Further, the calls were grouped by customer, so there wasn't a bulk of cellsite-zero calls; they were interspersed with each customer's other calls. Finally, and possibly most importantly, as part of the data preparation the data were scrubbed to improve the quality of the target variable. Some calls credited as fraud to an account were not actually fraudulent. Many of these, in turn, could be identified by seeing that the customer called them in a prior, nonfraud period. The result was that calls from cellsite zero had an elevated probability of fraud, but were not a perfect predictor of fraud (which would have been a red flag).

The point of this mini-case study is to illustrate that "what the data is" is an interpretation that we place. This interpretation often changes through the process of data mining, and we need to embrace this malleability. Our fraud detection example showed a change in the interpretation of a data item. We often also change our understanding of how the data were sampled as we uncover biases in the data collection process. For example, if we want to model consumer behavior in order to design or deliver a marketing campaign, it is essential to understand exactly what was the consumer base from which the data were sampled. This again sounds obvious in theory, but in practice it may involve in-depth analysis of the systems and businesses from which the data came.

Finally we need to be discerning in *the sorts of problems for which data science, even with the integration of humans, is likely to add value*. We must ask: are there really sufficient data pertaining to the decision at hand? Very high-level strategic decisions may be placed in a unique context. Data analyses, as well as theoretical simulations, may provide insight, but often for the highest-level decisions the decision makers must rely on their experience, knowledge, and intuition. This applies certainly to strategic decisions such as whether to acquire a particular company: data analysis can support the

decision, but ultimately each situation is unique and the judgment of an experienced strategist will be necessary.

This idea of unique situations should be carried through. At an extreme we might think of Steve Jobs' famous statement: "It's really hard to design products by focus groups. A lot of times, people don't know what they want until you show it to them… That doesn't mean we don't listen to customers, but it's hard for them to tell you what they want when they've never seen anything remotely like it." As we look to the future we may hope that with the increasing ability to do careful, automated experimentation we may move from asking people what they would like or would find useful to observing what they like or find useful. To do this well, we need to follow our fundamental principle: consider data as an asset, in which we may need to invest. Our Capital One case from Chapter 1 is an example of creating many products and investing in data and data science to determine both which ones people would want, and for each product which people would be appropriate (i.e., profitable) customers.

Privacy, Ethics, and Mining Data About Individuals

Mining data, especially data about individuals, raises important ethical issues that should not be ignored. There recently has been considerable discussion in the press and within government agencies about privacy and data (especially online data), but the issues are much broader. Most consumer-facing large companies collect or purchase detailed data on all of us. These data are used directly to make decisions regarding many of the business applications we have discussed through the book: should we be granted credit? If so, what should be our credit line? Should we be targeted with an offer? What content would we like to be shown on the website? What products should be recommended to us? Are we likely to defect to a competitor? Is there fraud on our account?

The tension between privacy and improving business decisions is intense because there seems to be a direct relationship between increased use of personal data and increased effectiveness of the associated business decisions. For example, a study by researchers at the University of Toronto and MIT showed that after particularly stringent privacy protection was enacted in Europe, online advertising became significantly less effective. In particular, "the difference in stated purchase intent between those who were exposed to ads and those who were not dropped by approximately 65%. There was no such change for countries outside Europe" (Goldfarb & Tucker, 2011).[6] This is not a phenomenon restricted to online advertising: adding fine-grained social network data (e.g., who communicates with whom) to more traditional data on individuals substantially increases the effectiveness of fraud detection (Fawcett & Provost, 1997) and targeted marketing (Hill et al., 2006). Generally, the more fine-grained data you collect on in-

6. See Mayer and Narayanan's web site for a criticism of this and other research claims about the value of behaviorally targeted online advertising.

dividuals, the better you can predict things about them that are important for business decision-making. This seeming direct relationship between reduced privacy and increased business performance elicits strong feelings from both the privacy and the business perspectives (sometimes within the same person).

It is far beyond the scope of this book to resolve this problem, and the issues are extremely complicated (for example, what sort of "anonymization" would be sufficient?) and diverse. Possibly the biggest impediment to the reasoned consideration of privacy-friendly data science designs is the difficulty with even defining what privacy is. Daniel Solove is a world authority on privacy. His article "A Taxonomy of Privacy" (2006) starts:

> Privacy is a concept in disarray. Nobody can articulate what it means. As one commentator has observed, privacy suffers from "an embarrassment of meanings."

Solove's article goes on to spend over 80 pages giving a taxonomy of privacy. Helen Nissenbaum is another world authority on privacy, who has concentrated recently specifically on the relationship of privacy and massive databases (and the mining thereof). Her book on this topic, *Privacy in Context*, is over 300 pages (and well worth reading). We bring this up to emphasize that privacy concerns are not some easy-to-understand or easy-to-deal-with issues that can be quickly dispatched, or even written about well as a section or chapter of a data science book. If you are either a data scientist or a business stakeholder in data science projects, you should care about privacy concerns, and you will need to invest serious time in thinking carefully about them.

To help frame our thinking about data science and privacy, we have posted an online appendix (see our page here) explaining some of the various concepts and issues, and pointing to additional reading material where we all can further expand our thinking.

Is There More to Data Science?

Although this book is fairly thick, we have tried hard to pick the most relevant fundamental concepts to help the data scientist and the business stakeholder to understand data science and to communicate well. Of course, we have not covered all the fundamental concepts of data science, and any given data scientist may dispute whether we have included exactly the right ones. But all should agree that these are some of the most important concepts and that they underlie a vast amount of the science.

There are all manner of advanced topics and closely related topics that build upon the fundamentals presented here. We will not try to list them—if you're interested, peruse the programs of recent top-notch data mining research conferences, such as the *ACM SIGKDD International Conference on Data Mining and Knowledge Discovery*, or the *IEEE International Conference on Data Mining*. Both of these conferences have top-notch Industry Tracks as well, focusing on applications of data science to business and government problems.

Let us give just one more concrete example of the sort of topic one might find when exploring further. Recall our first principle of data science: data (and data science capability) should be regarded as assets, and should be candidates for investment. Through the book we have discussed in increasing complexity the notion of investing in data. If we apply our general framework of considering the costs and benefits in data science projects explicitly, it leads us to new thinking about investing in data.

Final Example: From Crowd-Sourcing to Cloud-Sourcing

The connectivity between businesses and "consumers" brought about by the Internet has changed the economics of labor. Web-based systems like Amazon's Mechanical Turk and oDesk (among others) facilitate a type of crowd-sourcing that might be called "cloud labor"—harnessing via the Internet a vast pool of independent contractors. One sort of cloud labor that is particularly relevant to data science is "micro-outsourcing": the outsourcing of large numbers of very small, well-defined tasks. Micro-outsourcing is particularly relevant to data science, because it changes the economics, as well as the practicalities, of investing in data.[7]

As one example, recall the requirements for applying supervised modeling. We need to have specified a target variable precisely, and we need to actually have values for the target variable ("labels") for a set of training data. Sometimes we can specify the target variable precisely, but we find we do not have any labeled data. In certain cases, we can use micro-outsourcing systems such as Mechanical Turk to label data.

For example, advertisers would like to keep their advertisements off of objectionable web pages, like those that contain hate speech. However, with billions of pages to put their ads on, how can they know which ones are objectionable? It would be far too costly to have employees look at them all. We might immediately recognize this as a possible candidate for text classification (Chapter 10): we can get the text of the page, represent it as feature vectors as we have discussed, and build a hate-speech classifier. Unfortunately, we have no representative sample of hate speech pages to use as training data. However, if this problem is important enough[8] then we should consider investing in labeled training data to see whether we can build a model to identify pages containing hate speech.

7. The interested reader can go to Google Scholar and query on "data mining mechanical turk" or more broadly on "human computation" to find papers on the topic, and to follow the forward citation links ("Cited by") to find even more.

8. In fact, the problem of ads appearing on objectionable pages was reported to be a $2 billion problem (Winterberry Group, 2010).

Cloud labor changes the economics of investing in data in our example of getting labeled training data. We can engage very inexpensive labor via the Internet to invest in data in various ways. For example, we can have workers on Amazon Mechanical Turk label pages as objectionable or not, providing us with target labels, much more cheaply than hiring even student workers.

> The rate of completion, when done by a trained intern, was 250 websites per hour, at a cost of $15/hr. When posted on Amazon Mechanical Turk, the labeling rate went up to 2,500 websites per hour and the overall cost remained the same. (Ipeirotis et al., 2010)

The problem is that you get what you pay for, and low cost sometimes means low quality. There has been a surge of research over the past half decade on the problems of maintaining quality while taking advantage of cloud labor. Note that page labeling is just one example of enhancing data science with cloud labor. Even in this case study there are other options, such as using cloud labor to *search for* positive examples of hate speech, instead of labeling pages that we give them (Attenberg & Provost, 2010), or cloud laborers can be challenged in a game-like system to find cases where the current model makes mistakes—to "beat the machine" (Attenberg et al., 2011).

Final Words

Your authors have been working on applying data science to real business problems for more than two decades. You would think that it would all become second nature. It is striking how useful it still can be even for us to have this set of explicit fundamental concepts in hand. So many times when you reach a seeming impasse in thinking, pulling out the the fundamental concepts makes the way clear. "Well, let's go back to our business and data understanding…what exactly is the problem we are trying to solve" can resolve many problems, whether we then decide to work through the implications of the expected value framework, or to think more carefully about how the data are gathered, or about whether the costs and benefits are specified well, or about further investing in data, or to consider whether the target variable has been defined appropriately for the problem to be solved, etc. Knowing what are the different sorts of data science tasks helps to keep the data scientist from treating all business problems as nails for the particular hammers that he knows well. Thinking carefully about what is important to the business problem, when considering evaluation and "baselines" for comparison, brings interactions with stakeholders to life. (Compare that with the chilling effect of reporting some statistic like mean-squared error when it is meaningless to the problem at hand.) This facilitation of data-analytic thinking applies not just to the data scientists, but to everyone involved.

If you are a business stakeholder rather than a data scientist, don't let so-called data scientists bamboozle you with jargon: the concepts of this book plus knowledge of your own business and data systems should allow you to understand 80% or more of the data science at a reasonable enough level to be productive for your business. After having

read this book, if you don't understand what a data scientist is talking about, be wary. There are of course many, many more complex concepts in data science, but a good data scientist should be able to describe the fundamentals of the problem and its solution at the level and in the terms of this book.

If you are a data scientist, take this as our challenge: think deeply about exactly why your work is relevant to helping the business and be able to present it as such.

APPENDIX A

Proposal Review Guide

Effective data analytic thinking should allow you to assess potential data mining projects systematically. The material in this book should give you the necessary background to assess proposed data mining projects, and to uncover potential flaws in proposals. This skill can be applied both as a self-assessment for your own proposals and as an aid in evaluating proposals from internal data science teams or external consultants.

What follows contains a set of questions that one should have in mind when considering a data mining project. The questions are framed by the data mining process discussed in detail in Chapter 2, and used as a conceptual framework throughout the book. After reading this book, you should be able to apply these conceptually to a new business problem. The list that follows is not meant to be exhaustive (in general, the book isn't meant to be exhaustive). However, the list contains a selection of some of the most important questions to ask.

Throughout the book we have concentrated on data science projects where the focus is to mine some regularities, patterns, or models from the data. The proposal review guide reflects this. There may be data science projects in an organization where these regularities are not so explicitly defined. For example, many data visualization projects initially do not have crisply defined objectives for modeling. Nevertheless, the data mining process can help to structure data-analytic thinking about such projects—they simply resemble unsupervised data mining more than supervised data mining.

Business and Data Understanding

- What exactly is the business problem to be solved?
- Is the data science solution formulated appropriately to solve this business problem? *NB: sometimes we have to make judicious approximations.*
- What business entity does an instance/example correspond to?

- Is the problem a supervised or unsupervised problem?
  - If supervised,
    - Is a *target* variable defined?
    - If so, is it defined precisely?
    - Think about the values it can take.
- Are the attributes defined precisely?
  - Think about the values they can take.
- For supervised problems: will modeling this target variable improve the stated business problem? An important subproblem? If the latter, is the rest of the business problem addressed?
- Does framing the problem in terms of expected value help to structure the subtasks that need to be solved?
- If unsupervised, is there an "exploratory data analysis" path well defined? (That is, *where is the analysis going?*)

Data Preparation

- Will it be practical to get values for attributes and create feature vectors, and put them into a single table?
- If not, is an alternative data format defined clearly and precisely? Is this taken into account in the later stages of the project? (Many of the later methods/techniques assume the dataset is in feature vector format.)
- If the modeling will be supervised, is the target variable well defined? Is it clear how to get values for the target variable (for training and testing) and put them into the table?
- How exactly will the values for the target variable be acquired? Are there any costs involved? If so, are the costs taken into account in the proposal?
- Are the data being drawn from the similar population to which the model will be applied? If there are discrepancies, are the selection biases noted clearly? Is there a plan for how to compensate for them?

Modeling

- Is the choice of model appropriate for the choice of target variable?
  - Classification, class probability estimation, ranking, regression, clustering, etc.

- Does the model/modeling technique meet the other requirements of the task?
  - Generalization performance, comprehensibility, speed of learning, speed of application, amount of data required, type of data, missing values?
  - Is the choice of modeling technique compatible with prior knowledge of problem (e.g., is a linear model being proposed for a definitely nonlinear problem)?
- Should various models be tried and compared (in evaluation)?
- For clustering, is there a similarity metric defined? Does it make sense for the business problem?

Evaluation and Deployment

- Is there a plan for domain-knowledge validation?
  - Will domain experts or stakeholders want to vet the model before deployment? If so, will the model be in a form they can understand?
- Is the evaluation setup and metric appropriate for the business task? Recall the original formulation.
  - Are business costs and benefits taken into account?
  - For classification, how is a classification threshold chosen?
  - Are probability estimates used directly?
  - Is ranking more appropriate (e.g., for a fixed budget)?
  - For regression, how will you evaluate the quality of numeric predictions? Why is this the right way in the context of the problem?
- Does the evaluation use holdout data?
  - Cross-validation is one technique.
- Against what baselines will the results be compared?
  - Why do these make sense in the context of the actual problem to be solved?
  - Is there a plan to evaluate the baseline methods objectively as well?
- For clustering, how will the clustering be understood?
- Will deployment as planned actually (best) address the stated business problem?
- If the project expense has to be justified to stakeholders, what is the plan to measure the final (deployed) business impact?

APPENDIX B

Another Sample Proposal

Appendix A presented a set of guidelines and questions useful for evaluating data science proposals. Chapter 13 contained a sample proposal ("Example Data Mining Proposal" on page 325) for a "customer migration" campaign and a critique of its weaknesses ("Flaws in the Big Red Proposal" on page 326).

We've used the telecommunications churn problem as a running example throughout this book. Here we present a second sample proposal and critique, this one based on the churn problem.

Scenario and Proposal

You've landed a great job with Green Giant Consulting (GGC), managing an analytical team that is just building up its data science skill set. GGC is proposing a data science project with TelCo, the nation's second-largest provider of wireless communication services, to help address their problem of customer churn. Your team of analysts has produced the following proposal, and you are reviewing it prior to presenting the proposed plan to TelCo. Do you find any flaws with the plan? Do you have any suggestions for how to improve it?

> **Churn Reduction via Targeted Incentives — A GGC Proposal**
> We propose that TelCo test its ability to control its customer churn via an analysis of churn prediction. The key idea is that TelCo can use data on customer behavior to predict when customers will leave, and then can target these customers with special incentives to remain with TelCo. We propose the following modeling problem, which can be carried out using data already in TelCo's possession.
>
> We will model the probability that a customer will (or will not) leave within 90 days of contract expiration, with the understanding that there is a separate problem of retaining customers who are continuing their service month-to-month, long after contract expiration. We believe that predicting churn in this 90-day window is an appropriate starting point, and the lessons learned may apply to other churn-prediction cases as well. The

model will be built on a database of historical cases of customers who have left the company. Churn probability will be predicted based on data 45 days prior to contract expiration, in order for TelCo to have sufficient lead time to affect customer behavior with an incentive offer. We will model churn probability by building an ensemble of trees (random forest) model, which is known to have high accuracy for a wide variety of estimation problems.

We estimate that we will be able to identify 70% of the customers who will leave within the 90-day time window. We will verify this by running the model on the database to verify that indeed the model can reach this level of accuracy. Through interactions with TelCo stakeholders, we understand that it is very important that the V.P. of Customer Retention sign off on any new customer retention procedures, and she has indicated that she will base her decision on her own assessment that the procedure used for identifying customers makes sense and on the opinions about the procedure from selected firm experts in customer retention. Therefore, we will give the V.P. and the experts access to the model, so that they can verify that it will operate effectively and appropriately. We propose that every week, the model be run to estimate the probabilities of churn of the customers whose contracts expire in 45 days (give or take a week). The customers will be ranked based on these probabilities, and the top *N* will be selected to receive the current incentive, with *N* based on the cost of the incentive and the weekly retention budget.

Flaws in the GGC Proposal

We can use our understanding of the fundamental principles and other basic concepts of data science to identify flaws in the proposal. Appendix A provides a starting "guide" for reviewing such proposals, with some of the main questions to ask. However, this book as a whole really can be seen as a proposal review guide. Here are some of the most egregious flaws in Green Giant's proposal:

1. The proposal currently only mentions modeling based on "customers who have left the company." For training (and testing) we will also want to have customers who did *not* leave the company, in order for the modeling to find discriminative information. (Chapter 2, Chapter 3, Chapter 4, Chapter 7)
2. Why rank customers by the highest probability of churn? Why not rank them by expected loss, using a standard expected value computation? (Chapter 7, Chapter 11)
3. Even better, should we not try to model those customers who are most likely to be influenced (positively) by the incentive? (Chapter 11, Chapter 12)
4. If we're going to proceed as in (3), we have the problem of not having the training data we need. We'll have to invest in obtaining training data. (Chapter 3, Chapter 11)

Note that the current proposal may well be just a first step toward the business goal, but this would need to be spelled out explicitly: *see if we can estimate the probabilities well*. If we can, then it makes sense to proceed. If not, we may need to rethink our investment in this project.

5. The proposal says nothing about assessing *generalization* performance (i.e., doing a holdout evaluation). It sounds like they are going to test on the training set ("… running the model on the database…"). (Chapter 5)
6. The proposal does not define (nor even mention) what attributes are going to be used! Is this just an omission? Is this because the team hasn't even thought about it? What is the plan? (Chapter 2, Chapter 3)
7. How does the team estimate that the model will be able to identify 70% of the customers who will leave? There is no mention that any pilot study already has been conducted, nor learning curves having been produced on data samples, nor any other support for this claim. It seems like a guess. (Chapter 2, Chapter 5, Chapter 7)
8. Furthermore, without discussing the error rate or the notion of false positives and false negatives, it's not clear what "identify 70% of the customers who will leave" really means. If I say nothing about the false-positive rate, I can identify 100% of them simply by saying everyone will leave. So talking about true-positive rate only makes sense if you also talk about false-positive rate. (Chapter 7, Chapter 8)
9. Why choose one particular model? With modern toolkits, we can easily compare various models on the same data. (Chapter 4, Chapter 7, Chapter 8)
10. The V.P. of Customer Retention must sign off on the procedure, and has indicated that she will examine the procedure to see if it makes sense (domain knowledge validation). However, ensembles of trees are black-box models. The proposal says nothing about how she is going to understand how the procedure is making its decisions. Given her desire, it would be better to sacrifice some accuracy to build a more comprehensible model. Once she is "on board" it may be possible to use less-comprehensible techniques to achieve higher accuracies. (Chapter 3, Chapter 7, Chapter 12)

Glossary

Note: This glossary is an extension to one compiled by Ron Kohavi and Foster Provost (1998), used with kind permission of Springer Science and Business Media.

a priori
: *A priori* is a term borrowed from philosophy meaning "prior to experience." In data science, an *a priori* belief is one that is brought to the problem as background knowledge, as opposed to a belief that is formed after examining data. For example, you might say, "There is no *a priori* reason to believe that this relationship is linear." After examining data you might decide that two variables have a linear relationship (and so linear regression should work fairly well), but there was no reason to believe, from prior knowledge, that they should be so related. The opposite of *a priori* is *a posteriori*.

Accuracy (error rate)
: The rate of correct (incorrect) predictions made by the model over a dataset (cf. coverage). Accuracy is usually estimated using an independent (holdout) dataset that was not used at any time during the learning process. More complex accuracy estimation techniques, such as cross-validation and the bootstrap, are commonly used, especially with datasets containing a small number of instances.

Association mining
: Techniques that find conjunctive implication rules of the form "X and Y → A and B" (associations) that satisfy given criteria.

Attribute (field, variable, feature)
: A quantity describing an instance. An attribute has a domain defined by the attribute type, which denotes the values that can be taken by an attribute. The following domain types are common:

  - **Categorical (symbolic)**: A finite number of discrete values. The type *nominal* denotes that there is no ordering between the values, such as last names and colors. The type *ordinal* denotes that there is an ordering, such as in an attribute taking on the values low, medium, or high.
  - **Continuous (quantitative)**: Commonly, subset of real numbers, where there is a measurable difference between the possible values. Integers are usually treated as continuous in practical problems.

  We do not differentiate in this book, but often the distinction is made that a feature is the specification of an attribute and its

value. For example, color is an attribute. "Color is blue" is a feature of an example. Many transformations to the attribute set leave the feature set unchanged (for example, regrouping attribute values or transforming multivalued attributes to binary attributes). In this book we follow the practice of many authors and practitioners, and use feature as a synonym for *attribute*.

Class (label)
: One of a small, mutually exclusive set of labels used as possible values for the target variable in a classification problem. Labeled data has one class label assigned to each example. For example, in a dollar bill classification problem the classes could be *legitimate* and *counterfeit*. In a stock assessment task the classes might be *will gain substantially, will lose substantially, and _will maintain its value*.

Classifier
: A mapping from unlabeled instances to (discrete) classes. Classifiers have a form (e.g., classification tree) plus an interpretation procedure (including how to handle unknown values, etc.). Most classifiers also can provide probability estimates (or other likelihood scores), which can be thresholded to yield a discrete class decision thereby taking into account a cost/benefit or utility function.

Confusion matrix
: A matrix showing the predicted and actual classifications. A confusion matrix is of size $l \times l$, where l is the number of different label values. A variety of classifier evaluation metrics are defined based on the contents of the confusion matrix, including *accuracy, true positive rate, false positive rate, true negative rate, false negative rate, precision, recall, sensitivity, specificity, positive predictive value,* and *negative predictive value*.

Coverage
: The proportion of a dataset for which a classifier makes a prediction. If a classifier does not classify all the instances, it may be important to know its performance on the set of cases for which it is confident enough to make a prediction.

Cost (utility/loss/payoff)
: A measurement of the cost to the performance task (and/or benefit) of making a prediction $\hat{y}$ when the actual label is y. The use of accuracy to evaluate a model assumes uniform costs of errors and uniform benefits of correct classifications.

Cross-validation
: A method for estimating the accuracy (or error) of an inducer by dividing the data into k mutually exclusive subsets (the "folds") of approximately equal size. The inducer is trained and tested k times. Each time it is trained on the dataset minus one of the folds and tested on that fold. The accuracy estimate is the average accuracy for the k folds or the accuracy on the combined ("pooled") testing folds.

Data cleaning/cleansing
: The process of improving the quality of the data by modifying its form or content, for example by removing or correcting data values that are incorrect. This step usually precedes the modeling step, although a pass through the data mining process may indicate that further cleaning is desired and may suggest ways to improve the quality of the data.

Data mining
: The term data mining is somewhat overloaded. It sometimes refers to the whole data mining process and sometimes to the specific application of modeling techniques to data in order to build models or find other patterns/regularities.

Dataset
: A schema and a set of instances matching the schema. Generally, no ordering on instances is assumed. Most data mining work uses a single fixed-format table or collection of feature vectors.

Dimension
: An attribute or several attributes that together describe a property. For example, a

geographical dimension might consist of three attributes: country, state, city. A time dimension might include 5 attributes: year, month, day, hour, minute.

Error rate
: See Accuracy (error rate).

Example
: See Instance (example, case, record).

Feature
: See Attribute (field, variable, feature).

Feature vector (record, tuple)
: A list of features describing an instance.

Field
: See Attribute.

i.i.d. sample
: A set of independent and identically distributed instances.

Induction
: Induction is the process of creating a general model (such as a classification tree or an equation) from a set of data. Induction may be contrasted with deduction: deduction starts with a general rule or model and one or more facts, and creates other specific facts from them. Induction goes in the other direction: induction takes a collection of facts and creates a general rule or model. In the context of this book, model induction is synonymous with *learning* or *mining* a model, and the rules or models are generally statistical in nature.

Instance (example, case, record)
: A single object of the world from which a model will be learned, or on which a model will be used (*e.g.*, for prediction). In most data science work, instances are described by feature vectors; some work uses more complex representations (*e.g.*, containing relations between instances or between parts of instances).

KDD
: originally was an abbreviation for Knowledge Discovery from Databases. It is now used to cover broadly the discovery of knowledge from data, and often is used synonymously with data mining.

Knowledge discovery
: The nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. This is the definition used in "Advances in Knowledge Discovery and Data Mining," by Fayyad, Piatetsky-Shapiro, & Smyth (1996).

Loss
: See Cost (utility/loss/payoff).

Machine learning
: In data science, machine learning is most commonly used to mean the application of induction algorithms to data. The term often used synonymously with the modeling stage the data mining process. Machine Learning is the field of scientific study that concentrates on induction algorithms and on other algorithms that can be said to learn.

Missing value
: The situation where the value for an attribute is not known or does not exist. There are several possible reasons for a value to be missing, such as: it was not measured; there was an instrument malfunction; the attribute does not apply, or the attribute's value cannot be known. Some algorithms have problems dealing with missing values.

Model
: A structure and corresponding interpretation that summarizes or partially summarizes a set of data, for description or prediction. Most inductive algorithms generate models that can then be used as classifiers, as regressors, as patterns for human consumption, and/or as input to subsequent stages of the data mining process.

Model deployment
: The use of a learned model to solve a real-world problem. Deployment often is used specifically to contrast with the "use" of a model in the Evaluation stage of the data mining process. In the latter, deployment

usually is simulated on data where the true answer is known.

OLAP (MOLAP, ROLAP)
: Online Analytical Processing. Usually synonymous with MOLAP (multidimensional OLAP). OLAP engines facilitate the exploration of data along several (predetermined) dimensions. OLAP commonly uses intermediate data structures to store precalculated results on multidimensional data, allowing fast computations. ROLAP (relational OLAP) refers to performing OLAP using relational databases.

Record
: See Feature vector (record, tuple).

Schema
: A description of a dataset's attributes and their properties.

Sensitivity
: True positive rate (see Confusion matrix).

Specificity
: True negative rate (see Confusion matrix).

Supervised learning
: Techniques used to learn the relationship between independent attributes and a designated dependent attribute (the label). Most induction algorithms fall into the supervised learning category.

Tuple
: See Feature vector (record, tuple).

Unsupervised learning
: Learning techniques that group instances without a pre-specified target attribute. Clustering algorithms are usually unsupervised.

Utility
: See Cost (utility/loss/payoff).

Bibliography

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, *7*(1), 39–59. Available: *http://www.iiia.csic.es/People/enric/AICom.html*.

Adams, N. M., & Hand, D. J. (1999). Comparing classifiers when the misallocations costs are uncertain. *Pattern Recognition*, *32*, 1139–1147.

Aha, D. W. (Ed.). (1997). *Lazy learning*. Kluwer Academic Publishers, Norwell, MA, USA.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, *6*, 37–66.

Aggarwal, C., & Yu, P. (2008). *Privacy-preserving Data Mining: Models and Algorithms.* Springer, USA.

Aral, S., Muchnik, L., & Sundararajan, A. (2009). Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proceedings of the National Academy of Sciences*, *106*(51), 21544-21549.

Arthur, D., & Vassilvitskii, S. (2007). K-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035.

Attenberg, J., Ipeirotis, P., & Provost, F. (2011). Beat the machine: Challenging workers to find the unknown unknowns. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence.*

Attenberg, J., & Provost, F. (2010). Why label when you can search?: Alternatives to active learning for applying human resources to build classification models under extreme class imbalance. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 423–432. ACM.

Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository. *http://archive.ics.uci.edu/ml*. Irvine, CA: University of California, School of Information and Computer Science.

Bolton, R., & Hand, D. (2002). Statistical Fraud Detection: A Review. *Statistical Science*, *17*(3), 235-255.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Wadsworth International Group, Belmont, CA.

Brooks, D. (2013). What Data Can't Do. *New York Times*, Feb. 18.

Brown, L., Gans, N., Mandelbaum, A., Sakov, A., Shen, H., Zeltyn, S., & Zhao, L. (2005). Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of the American Statistical Association*, *100*(469), 36-50.

Brynjolfsson, E., & Smith, M. (2000). Frictionless commerce? A comparison of internet and conventional retailers. *Management Science*, *46*, 563–585.

Brynjolfsson, E., Hitt, L. M., & Kim, H. H. (2011). Strength in numbers: How does data-driven decision making affect firm performance? Tech. rep., available at SSRN: *http://ssrn.com/abstract=1819486* or *http://dx.doi.org/10.2139/ssrn.1819486*.

Business Insider (2012). The Digital 100: The world's most valuable private tech companies. *http://www.businessinsider.com/2012-digital-100*.

Ciccarelli, F. D., Doerks, T., Von Mering, C., Creevey, C. J., Snel, B., & Bork, P. (2006). Toward automatic reconstruction of a highly resolved tree of life. *Science*, *311* (5765), 1283–1287.

Clearwater, S., & Stern, E. (1991). A rule-learning program in high energy physics event classification. *Comp Physics Comm*, *67*, 159–182.

Clemons, E., & Thatcher, M. (1998). Capital One: Exploiting and Information-based Strategy. In *Proceedings of the 31st Hawaii International Conference on System Sciences*.

Cohen, L., Diether, K., & Malloy, C. (2012). Legislating Stock Prices. Harvard Business School Working Paper, No. 13–010.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, *13*(1), 21–27.

Crandall, D., Backstrom, L., Cosley, D., Suri, S., Huttenlocher, D., & Kleinberg, J. (2010). Inferring social ties from geographic coincidences. *Proceedings of the National Academy of Sciences*, 107(52), 22436-22441.

Deza, E., & Deza, M. (2006). *Dictionary of distances*. Elsevier Science.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, *10*, 1895–1923.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *Multiple Classifier Systems*, 1-15.

Duhigg, C. (2012). How Companies Learn Your Secrets. *New York Times*, Feb. 19.

Elmagarmid, A., Ipeirotis, P., & Verykios, V. (2007). Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, *19*(1), 1–16.

Evans, R., & Fisher, D. (2002). Using decision tree induction to minimize process delays in the printing industry. In Klosgen, W., & Zytkow, J. (Eds.), *Handbook of Data Mining and Knowledge Discovery*, pp. 874–881. Oxford University Press.

Ezawa, K., Singh, M., & Norton, S. (1996). Learning goal oriented Bayesian networks for telecommunications risk management. In Saitta, L. (Ed.), *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 139–147. San Francisco, CA. Morgan Kaufmann.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874.

Fawcett, T., & Provost, F. (1996). Combining data mining and machine learning for effective user profiling. In Simoudis, Han, & Fayyad (Eds.), *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 8–13. Menlo Park, CA. AAAI Press.

Fawcett, T., & Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*, *1* (3), 291–316.

Fayyad, U., Piatetsky-shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, *17*, 37–54.

Frank, A., & Asuncion, A. (2010). UCI machine learning repository.

Friedman, J. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, *1*(1), 55-77.

Gandy, O. H. (2009). *Coming to Terms with Chance: Engaging Rational Discrimination and Cumulative Disadvantage.* Ashgate Publishing Company.

Goldfarb, A. & Tucker, C. (2011). Online advertising, behavioral targeting, and privacy. *Communications of the ACM 54*(5), 25-27.

Haimowitz, I., & Schwartz, H. (1997). Clustering and prediction for credit line optimization. In Fawcett, Haimowitz, Provost, & Stolfo (Eds.), *AI Approaches to Fraud Detection and Risk Management*, pp. 29–33. AAAI Press. Available as Technical Report WS-97-07.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. & Witten, I. (2009). The WEKA data mining software: An update. *SIGKDD Explorations*, 11 (1).

Hand, D. J. (2008). *Statistics: A Very Short Introduction*. Oxford University Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Second Edition edition). Springer.

Hays, C. L. (2004). What they know about you. *The New York Times*.

Hernández, M. A., & Stolfo, S. J. (1995). The merge/purge problem for large databases. *SIGMOD Rec.*, *24*, 127–138.

Hill, S., Provost, F., & Volinsky, C. (2006). Network-based marketing: Identifying likely adopters via consumer networks. *Statistical Science*, *21* (2), 256–276.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, *11*, 63–91.

Ipeirotis, P., Provost, F., & Wang, J. (2010). Quality management on Amazon Mechanical Turk. In *Proceedings of the 2010 ACM SIGKDD Workshop on Human Computation*, pp. 64-67. ACM.

Jackson, M. (1989). *Michael Jackson's Malt Whisky Companion: a Connoisseur's Guide to the Malt Whiskies of Scotland*. Dorling Kindersley, London.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis*, *6* (5), 429–450.

Japkowicz, N., & Shah, M. (2011). *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press.

Jensen, D. D., & Cohen, P. R. (2000). Multiple comparisons in induction algorithms. *Machine Learning*, *38*(3), 309–338.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, *29*(2), 119–127.

Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *6*(4), 15.

Kohavi, R., Brodley, C., Frasca, B., Mason, L., & Zheng, Z. (2000). KDD-cup 2000 organizers' report: Peeling the onion. *ACM SIGKDD Explorations*. *2*(2).

Kohavi, R., Deng, A., Frasca, B., Longbotham, R., Walker, T., & Xu, Y. (2012). Trustworthy online controlled experiments: Five puzzling outcomes explained. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 786–794. ACM.

Kohavi, R., & Longbotham, R. (2007). Online experiments: Lessons learned. *Computer*, *40* (9), 103–105.

Kohavi, R., Longbotham, R., Sommerfield, D., & Henne, R. (2009). Controlled experiments on the web: Survey and practical guide. *Data Mining and Knowledge Discovery*, *18*(1), 140-181.

Kohavi, R., & Parekh, R. (2003). Ten supplementary analyses to improve e-commerce web sites. In *Proceedings of the Fifth WEBKDD workshop*.

Kohavi, R., & Provost, F. (1998). Glossary of terms. *Machine Learning*, *30*(2-3), 271-274.

Kolodner, J. (1993). *Case-Based Reasoning*. Morgan Kaufmann, San Mateo.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42* (8), 30-37.

Kosinski, M., Stillwell, D., & Graepel, T. (2013). Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, doi: *10.1073/pnas.1218772110*.

Lapointe, F.-J., & Legendre, P. (1994). A classification of pure malt Scotch whiskies. *Applied Statistics*, *43* (1), 237–257.

Leigh, D. (1995). Neural networks for credit scoring. In Goonatilake, S., & Treleaven, P. (Eds.), *Intelligent Systems for Finance and Business*, pp. 61–69. John Wiley and Sons Ltd., West Sussex, England.

Letunic, & Bork (2006). Interactive tree of life (iTOL): an online tool for phylogenetic tree display and annotation. *Bioinformatics*, *23* (1).

Lin, J.-H., & Vitter, J. S. (1994). A theory for memory-based learning. *Machine Learning*, *17*, 143–167.

Lloyd, S. P. (1982). Least square quantization in PCM. *IEEE Transactions on Information Theory*, *28* (2), 129–137.

MacKay, D. (2003). *Information Theory, Inference and Learning Algorithms*, Chapter 20. An Example Inference Task: Clustering. Cambridge University Press.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297. University of California Press.

Malin, B. & Sweeney, L. (2004). How (not) to protect genomic data privacy in a distributed network: Using trail re-identification to evaluate and design anonymity protection systems. *Journal of Biomedical Informatics*, 37(3), 179-192.

Martens, D., & Provost, F. (2011). Pseudo-social network targeting from consumer transaction data. Working paper CeDER-11-05, New York University – Stern School of Business.

McDowell, G. (2008). *Cracking the Coding Interview: 150 Programming Questions and Solutions*. CareerCup LLC.

McNamee, M. (2001). Credit Card Revolutionary. *Stanford Business 69* (3).

McPherson, M., Smith-Lovin, L., & Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, *27*:415-444.

Mittermayer, M., & Knolmayer, G. (2006). Text mining systems for market response to news: A survey. Working Paper No.184, Institute of Information Systems, University of Bern.

Muoio, A. (1997). They have a better idea … do you? *Fast Company*, *10*.

Nissenbaum, H. (2010). *Privacy in context.* Stanford University Press.

Papadopoulos, A. N., & Manolopoulos, Y. (2005). *Nearest Neighbor Search: A Database Perspective*. Springer.

Pennisi, E. (2003). A tree of life. Available online only: *http://www.sciencemag.org/site/feature/data/tol/*.

Perlich, C., Provost, F., & Simonoff, J. (2003). Tree Induction vs. Logistic Regression: A Learning-Curve Analysis. *Journal of Machine Learning Research*, *4*, 211-255.

Perlich, C., Dalessandro, B., Stitelman, O., Raeder, T., & Provost, F. (2013). Machine learning for targeted display advertising: Transfer learning in action. *Machine Learning* (in press; published online: 30 May 2013. DOI 10.1007/s10994-013-5375-2).

Poundstone, W. (2012). *Are You Smart Enough to Work at Google?: Trick Questions, Zen-like Riddles, Insanely Difficult Puzzles, and Other Devious Interviewing Techniques You Need to Know to Get a Job Anywhere in the New Economy*. Little, Brown and Company.

Provost, F., & Fawcett, T. (1997). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pp. 43–48 Menlo Park, CA. AAAI Press.

Provost, F., & Fawcett, T. (2001). Robust classification for imprecise environments. *Machine learning*, *42*(3), 203–231.

Provost, F., Fawcett, T., & Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In Shavlik, J. (Ed.), *Proceedings of ICML-98*, pp. 445–453 San Francisco, CA. Morgan Kaufmann.

Pyle, D. (1999). *Data Preparation for Data Mining*. Morgan Kaufmann.

Quine, W.V.O. (1951). Two dogmas of empiricism, *The Philosophical Review 60*: 20-43. Reprinted in his 1953 *From a Logical Point of View*. Harvard University Press.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, *1* (1), 81–106.

Raeder, T., Dalessandro, B., Stitelman, O., Perlich, C., & Provost, F. (2012). Design principles of massive, robust prediction systems. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Rosset, S., & Zhu, J. (2007). Piecewise linear regularized solution paths. *The Annals of Statistics*, *35*(3), 1012–1030.

Schumaker, R., & Chen, H. (2010). A Discrete Stock Price Prediction Engine Based on Financial News Keywords. *IEEE Computer*, *43*(1), 51–56.

Sengupta, S. (2012). Facebook's prospects may rest on trove of data.

Shakhnarovich, G., Darrell, T., & Indyk, P.(Eds., 2005). *Nearest-Neighbor Methods in Learning and Vision*. Neural Information Processing Series. The MIT Press, Cambridge, Massachusetts, USA.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*, 379–423.

Shearer, C. (2000). The CRISP-DM model: The new blueprint for data mining. *Journal of Data Warehousing*, *5*(4), 13–22.

Shmueli, G. (2010). To explain or to predict?. *Statistical Science*, *25*(3), 289–310.

Silver, N. (2012). *The Signal and the Noise*. The Penguin Press HC.

Solove, D. (2006). A taxonomy of privacy. *University of Pennsylvania Law Review*, *154*(3), 477-564.

Stein, R. M. (2005). The relationship between default prediction and lending profits: Integrating ROC analysis and loan pricing. *Journal of Banking and Finance*, *29*, 1213–1236.

Sugden, A. M., Jasny, B. R., Culotta, E., & Pennisi, E. (2003). Charting the evolutionary history of life. *Science*, *300*(5626).

Swets, J. (1988). Measuring the accuracy of diagnostic systems. *Science*, *240*, 1285–1293.

Swets, J. A. (1996). *Signal Detection Theory and ROC Analysis in Psychology and Diagnostics: Collected Papers*. Lawrence Erlbaum Associates, Mahwah, NJ.

Swets, J. A., Dawes, R. M., & Monahan, J. (2000). Better decisions through science. *Scientific American*, *283*, 82–87.

Tambe, P. (2013). Big Data Investment, Skills, and Firm Value. Working Paper, NYU Stern. Available: *http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2294077*.

WEKA (2001). Weka machine learning software. Available: *http://www.cs.waikato.ac.nz/~ml/index.html*.

Wikipedia (2012). Determining the number of clusters in a data set. *Wikipedia, the free encyclopedia*. *http://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set* [Online; accessed 14-February-2013].

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, *1*(6), 80–83. Available: *http://sci2s.ugr.es/keel/pdf/algorithm/articulo/wilcoxon1945.pdf*.

Winterberry Group (2010). Beyond the grey areas: Transparency, brand safety and the future of online advertising. White Paper, Winterberry Group LLC. *http://www.winterberrygroup.com/ourinsights/wp*

Wishart, D. (2006). *Whisky Classified*. Pavilion.

Witten, I., & Frank, E. (2000). *Data mining: Practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann, San Francisco. Software available from *http://www.cs.waikato.ac.nz/~ml/weka/*.

Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. In *Proceedings of the Twenty-first International Conference on Machine Learning*, pp. 903-910.

Zadrozny, B., & Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 204–213. ACM.

Index

Symbols

A

We'd like to hear your suggestions for improving our indexes. Send email to index@oreilly.com.

B

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

About the Authors

Foster Provost is Professor and NEC Faculty Fellow at the NYU Stern School of Business where he teaches in the Business Analytics, Data Science, and MBA programs. His award-winning research is read and cited broadly. Prior to joining NYU, he worked as a research data scientist for five years for what's now Verizon. Over the past decade, Professor Provost has co-founded several successful data-science-driven companies.

Tom Fawcett holds a Ph.D. in machine learning and has worked in industry R&D for more than two decades (GTE Laboratories, NYNEX/Verizon Labs, HP Labs, etc.). His published work has become standard reading in data science both on methodology (e.g., evaluating data mining results) and on applications (e.g., fraud detection and spam filtering).

Colophon

The cover font is Adobe ITC Garamond. The text font is Adobe Minion Pro and the heading font is Adobe Myriad Condensed.