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.

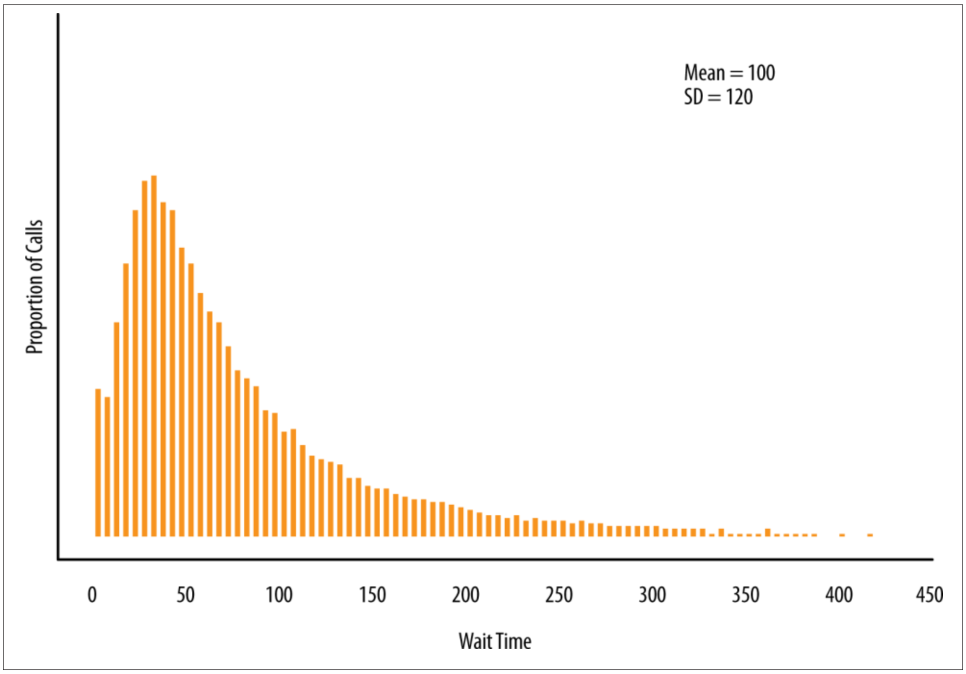

Figure 12-1. A distribution of wait times for callers into a bank's call center.

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.

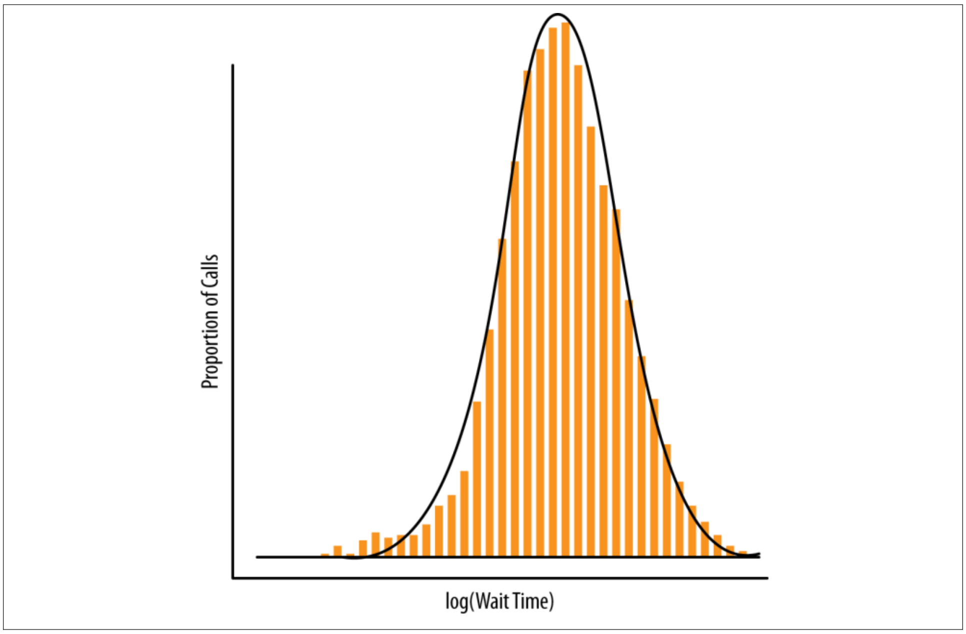

Figure 12-2. The distribution of wait times for callers into a bank's call center after a quick redefinition of the data.

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.

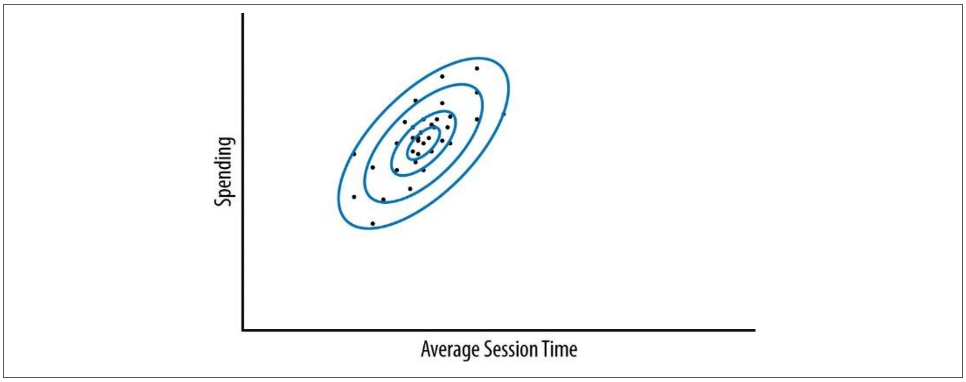

Figure 12-3. A profile of our customers with respect to their spending and the time they spend on our web site, represented as a two-dimensional Gaussian fit to the data.

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.

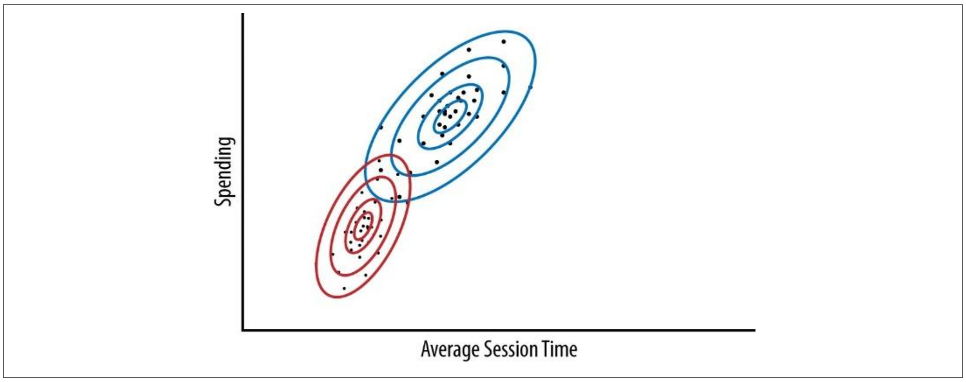

Figure 12-4. A profile of our customers with respect to their spending and the time they spend on our web site, represented as a Gaussian Mixture Model (GMM), with 2 two-dimensional Gaussians fit to the data. The GMM provides a "soft" clustering of our customers along two these two dimensions.

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 probabilis- tic clustering. Each point does not strictly belong to a single cluster, but instead has a degree or probability of membership in each clus-ter. In this particular clustering, we can think that a point is more likelyto 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.

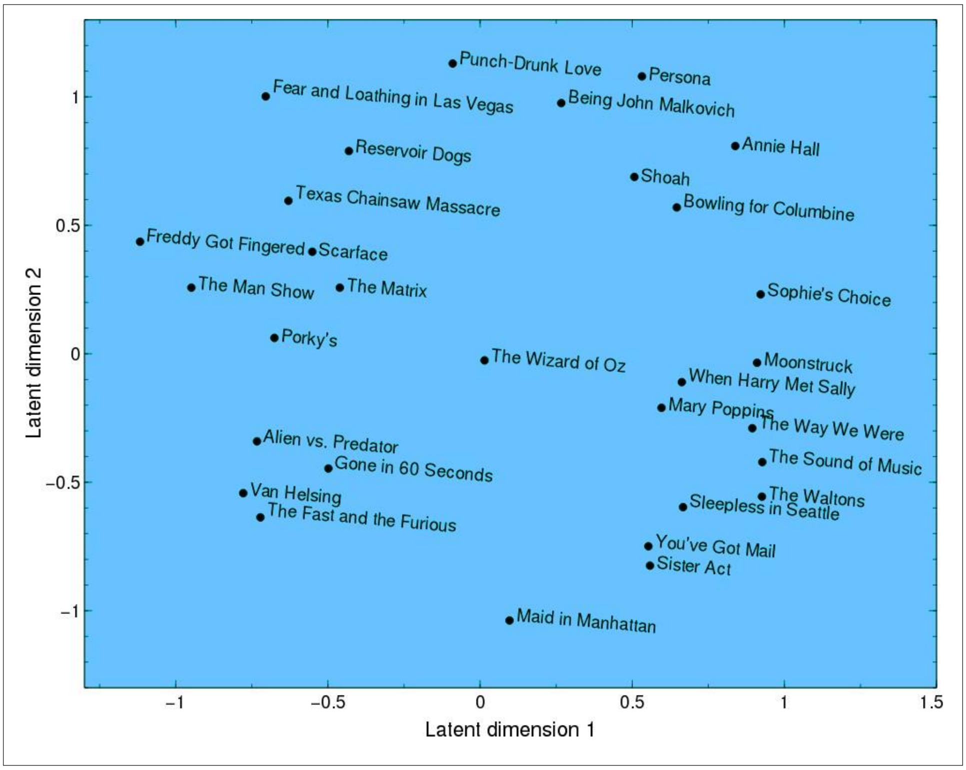

Figure 12-5. A collection of movies placed in a "taste space" defined by the two strongest latent dimensions mined from the Netflix Challenge data. See the text for a detailed discussion. A customer would also be placed somewhere in the space, based on the movies she has previously viewed or rated. A 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 over-estimated 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 severalother 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 cus- tomer wait time; predicting links between data items, such as potential social connec-tions between people; reducing our data to make it more manageable or to reveal hiddeninformation, such as latent movie preferences; combining models as if they were experts with different expertise, for example to improve movie recommendations; and drawingcausal conclusions from data, such as whether and to what extent the fact that sociallyconnected 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 combi- nations of them.