3

SOME GENERAL STRATEGIES USED IN DATA MINING

CROSS-VALIDATION

Data dredging—searching through data until one finds statistically significant relations—is deplored by conventional methods textbooks, which instruct students to generate hypotheses before beginning their statistical analyses. The DM approach raises data dredging to new heights—but to its credit DM does not follow the conventional paradigm's bad example regarding significance testing when there are multiple predictors. It focuses instead on an alternative way of avoiding false-positive results or type I error: it emphasizes replication rather than significance testing, through a procedure known as *cross-validation*.

Prior to beginning an analysis involving cross-validation, DM software separates the cases within a dataset into different groups, randomly assigning each case or observation to one group or another. (*Random* assignment is critical here.) DM software typically allows the user to choose what proportion of cases from the original dataset should go into each group.

- One group or random subset of cases or observations is known as the *training sample* or *estimation sample*. This is the group of cases that will be analyzed first, to create a predictive model.
- In some but not all DM methods a second random sample is created, known as the *tuning sample* (it is sometimes called the *validation sample*). It is used to estimate certain modeling parameters that will yield an optimal prediction. For

example, some DM techniques combine separate predictive models into one final best effort at prediction. This requires a decision about how to weight the prediction from each of the models when they are combined. In this context, this second random sample of cases, the tuning-sample data, would be used to calculate alternative weights so that the final weighting scheme yields the most accurate prediction. (This is known as *optimization*.) In other DM contexts, the tuning sample is instead used to decide how many predictors ought to be entered into a model.

- A third randomly selected group of observations is central to cross-validation. This is the *test sample,* sometimes called the *holdout sample*. The test sample is not used in any way during the creation of the predictive model; it is deliberately kept completely separate (held back).

During the last step in a DM analysis, a predictive model that has been generated using the data in the training sample (and sometimes involving the tuning sample data too) is applied to the new test sample data. The model generates predicted values of the target for these new test cases and those predicted values are compared to the actual observed values of the target in the test data. Fit statistics are calculated for this test sample, documenting how accurately the previously estimated model predicts this fresh set of observations.

Cross-validation fulfills a function in DM analogous to that provided by significance testing in the conventional approach: it is a way of assessing the generalizability of research findings. You might also think of cross-validation as a kind of quality control for data mining models.

The difference between the two approaches to generalization is that, in the conventional paradigm, tests for statistical significance speak to whether findings from one particular sample generalize to the *population* from which the sample was randomly drawn. Moreover, the assessment of generalizability is a theoretical or hypothetical one: the researcher doesn't have actual data for that entire population. In contrast, the test of generalizability in DM is an empirical one. A model that was developed and performed well for the training or estimation sample is applied to a different sample *of real data* (the test sample) and the goodness of that fit informs the researcher's judgment as to how well the model generalizes to the new data. In the DM case, generalizability is not from one sample to a population but rather from one random sample to another random sample (i.e., from the training to the test sample).

There are several variants of cross-validation. The simplest is known as the *holdout method* and is ideally suited to analyzing big data with many observations. It randomly divides a dataset into two or three subsamples (the training, tuning, and testing samples). The test sample is held out and is not used in training the predictive model. If the original dataset is very large, this random division of the original sample into two or three parts does not result in a problematic loss of statistical power when estimating predictive models; there will be lots of cases left in the training subsample. Note that in this holdout method, each observation is randomly assigned to either the training, the tuning, or the test subsample. Each subsample therefore contains completely separate cases or observations.

Cross-validation does not *require* a very large dataset, however. A different kind of cross-validation procedure known as k-*fold cross-validation* works with small as well as large datasets. The procedure begins by creating a chosen number (k) of random subsamples; often 10 are created. Cases or observations are drawn at random from the original sample and are assigned to each subsample, until one has k randomly selected subsamples in all. Each contains one-kth of the original sample.

One of those k subsamples will initially function as the test dataset, while the other $k - 1$ subsamples are pooled together to form a training set. A model is estimated for that pooled training set, and afterwards this predictive model is tested on the single test set subsample, yielding a fit statistic or a measure of error.

This procedure is then repeated k times in all, such that each of the k subsamples serves just once as the test dataset, while the remaining folds combined are the training data. The final fit statistic that the software reports is the average of the fit statistics for the test samples across all k runs.

Whichever of these forms of cross-validation is chosen (and there are additional variants), the crucial point to remember is that when assessing the predictive accuracy of a model you should always look at the fit statistics for the *holdout* or *test* sample. Some software reports fit statistics for the training sample as well, but *the fit statistic for the holdout or test sample is always the important one.*

To understand why data miners depend solely upon the fit statistics for the test sample requires a digression into another important phenomenon, known as *overfitting*.

OVERFITTING

DM has its own weak spots, and overfitting is one of those. Some DM applications are *too* effective at building a predictive model; they construct something too complicated, that will not generalize to other samples. This is easiest to appreciate graphically (see figure 3.1).

A simple model would explain the relation between X and Y in this diagram by fitting a straight line. This line represents the predicted value of Y for various values of X. The vertical distance from each data point to the straight line represents the error of estimation for each data point in that simple model, the difference between the predicted value of Y and the observed value of Y for each value of X.

A much more complicated model might reduce the amount of prediction error. The wavy line represents an equation such as $Y = a + bX + cX^2 + dX^3 + eX^4 + fX^5 \ldots$. As you can see in the diagram, this more complex line passes right through all the data points, implying no prediction error.

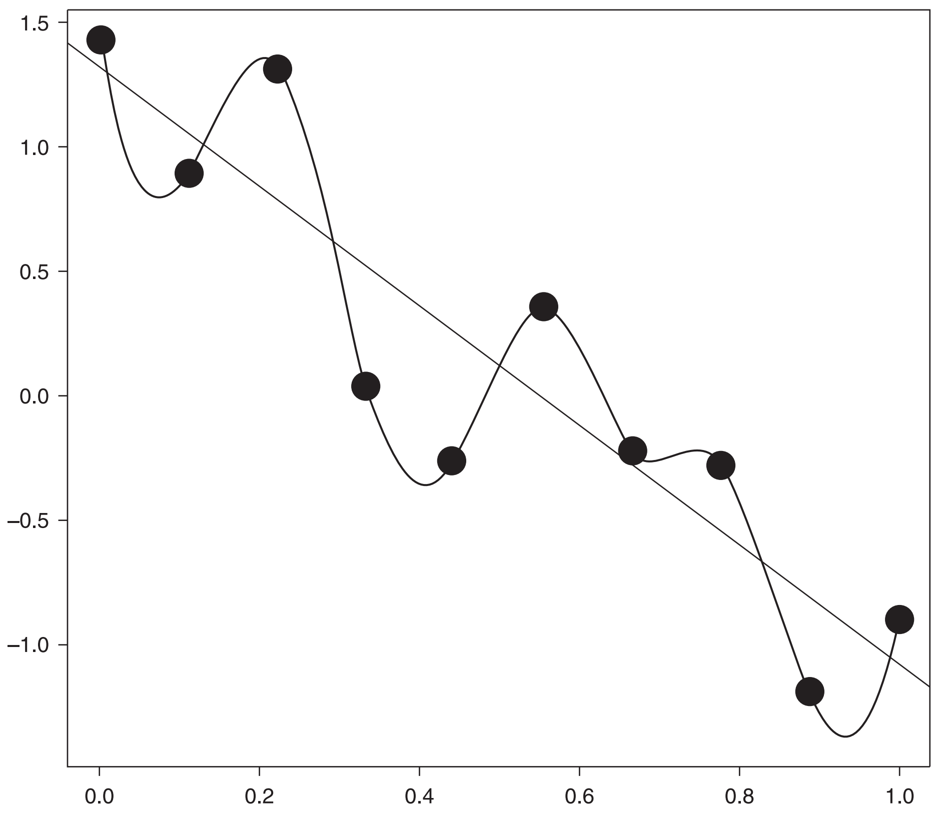

FIGURE 3.1
Overfitting data.

What could be wrong with choosing a more complex model if that reduces error and yields a stronger prediction? Data miners would caution that some of the distance from each data point to the straight line is probably due to measurement error, that is, to *noise*. Using a highly complicated model, like the wavy line, to fit those data points exactly is not just fitting the signal; it is also fitting the noise. The complex wavy model is, in DM jargon, *overfitting* the data. Overfitting is undesirable because it means that the complex model will not perform very well once it is applied to other data, such as the test data. The model has been tailored to the noisy training data and as a result won't fit other data very well.

How can one know whether any given model or equation is overfitted or not? When the predictive model (usually in the form of an equation) derived from a particular training sample is applied to a completely separate test sample, containing different observations or cases, then one can compare the predicted values obtained from the model to the observed values in the new dataset and determine how well they fit. This second step provides a trustworthy assessment of how well the predictive model works for data which were not used before.

The overfitting will "drop out" or fail to help in prediction of the test or holdout data because the part of the model that described chance patterns in the training data (the overfitted part) will fail to predict anything useful in the second or test dataset. There will be random noise in the second random test sample too, but by definition if it is random it isn't going to be the same noise as in the first dataset. So it won't have the same pattern; in fact it won't have any pattern.

Typically, therefore, a fit statistic for a model calculated for a training sample will be better than the fit of the same model applied to a test sample (because the test data will not overfit). If there is a large difference in the fit statistic between a training sample and a test sample, that is a strong indication that there was considerable overfitting in the former case. Conversely, if the fit of a model in a training sample and the fit of the same model applied to a test sample are pretty close, then this suggests that there was not much overfitting in the former: the model generalizes well.

To conclude, data mining's use of cross-validation appears to be a more rigorous approach to avoiding type I error (false-positive findings) than the significance-testing strategy common in conventional social research. *One evaluates the accuracy of a DM model by examining fit statistics obtained for a randomly selected test or holdout sample, and this provides a trustworthy measure of the generalizability of the findings.*

Figure 3.2 gives a visual illustration of how cross-validation can be used to avoid overfitting. The two top quadrants in the diagram are analyses of the same data points. The top-left quadrant of the diagram shows a complex model fitted to these data, a curve with many inflection points where the model (represented by the line) closely fits all the data points, so it will yield a very good prediction for the *training data*. However, we are told in that quadrant that cross-validation (CV) informs us that this is a very bad model because its fit statistics when applied to *test data* were greatly reduced. The original model must have been overfitted.

The top right of the diagram shows a much simpler model fitted to the same data points. The fit is not as good as previously, since the line does not go right through each of the points. However, when applied to test data, via cross-validation, we are told this simpler model fits quite well. There is not much of a reduction in the fit statistics when comparing the model applied to the test data with the training data. So a data miner would opt for the top-right model.

The two bottom quadrants also refer to a single dataset, but it's a different dataset from the top half of the diagram. On the left, a complex DM model is fitted to the data; however, we are told that the cross-validation fit statistics for this model are nearly as good for the test sample as for the training sample. So we conclude that, although it is complicated, this is a generalizable model; it is not overfitted. Just to be prudent, we also try a simpler model on the same data. The bottom-right quadrant shows this. With this new model we find that the fit statistics for the test sample are not as good as for the training sample.

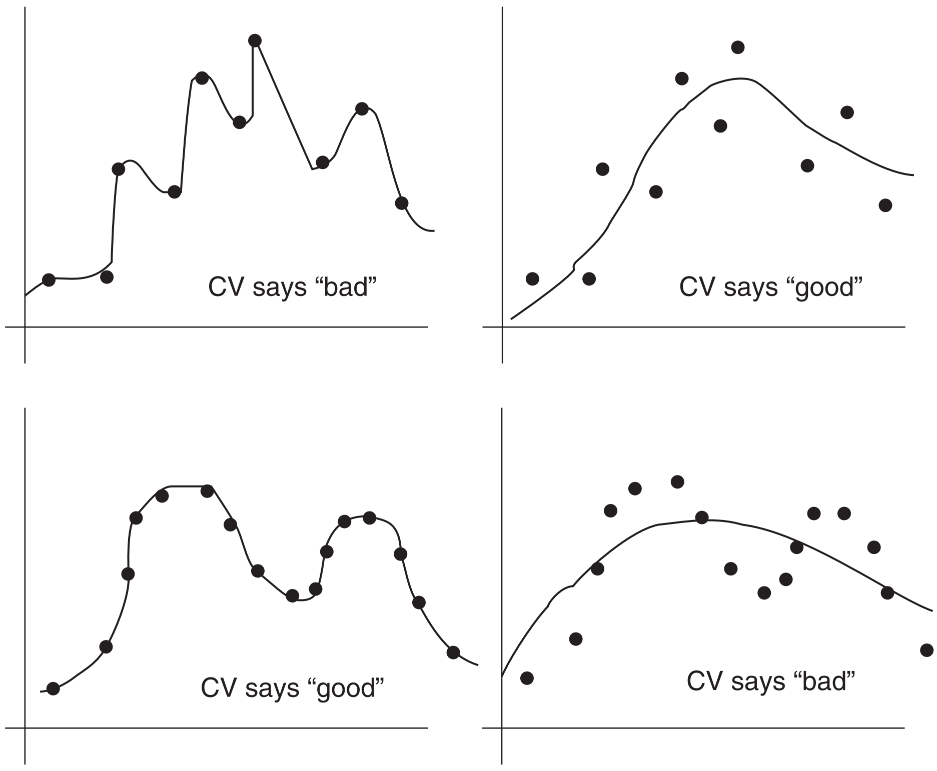

FIGURE 3.2
Cross-validation (from www.cs.cmu.edu/~schneide/tut5/node42.html#figcv0).

Three lessons can be drawn from these illustrations.

- Simplicity in a model is not always good (though we usually favor it); and
- complexity in a model is not always evidence of overfitting. However,
- cross-validation is an objective procedure that prevents one from accepting overfitted models.

BOOSTING

We have claimed that DM emphasizes the importance of accurate prediction and, compared to the conventional paradigm, is less accepting of models that can explain only a small percentage of the variance in a dependent variable. Because enhanced prediction is such a strong motive for data miners, they have developed novel strategies that improve prediction. Some of these seem quite strange when viewed from the perspective of conventional social science modeling. But, as we shall demonstrate, these strategies often outperform conventional models when it comes to prediction.

Boosting is one such strategy. It treats model creation as a series of steps. One might begin, for example, by estimating a regression model to predict an observed target variable, Y. The model fit is not perfect, so for each observation there will be a residual or prediction error, the difference between the observed and predicted value on Y for each case, or $Y - \hat{Y}$.

In a second step, another predictive model is estimated, using a different modeling method, but this time predicting the *residuals* from the first model, rather than predicting the original dependent variable, Y. This second model also yields predicted values, but some errors of prediction remain. So the residuals from this second model can, in turn, be predicted by a third model, and so on for many iterations.

The final stage in a boosted analysis is to combine the prediction equations obtained from each step (Ridgeway 1999). This is sometimes accomplished by providing diminishing weights to successive models, and then aggregating the predictions to get a single best prediction of Y.

Boosting can produce a substantial improvement in the final fit or predictive accuracy of a DM model, compared to a conventional single-step approach. Matthias Schonlau (2005) has written a Stata program called boost that applies a popular boosting algorithm. He reports its performance for two examples: a conventional linear regression and a stepwise logistic regression. In the former context, the conventional ordinary least squares (OLS) regression model predicted 21.3% of the variance (the R^2), while the identical predictors and data in a boosted regression explained 93.8% of the variance. For a stepwise logistic regression, the conventional Stata program correctly classified 54.1% of cases in test data, but boosting correctly predicted 76.0% of cases in a test sample. These are huge increases in predictive power thanks to boosting.

You will recall that a common explanation given for why conventional (non-DM) statistical models often explain only a small percentage of the variance is the presence of measurement error and/or the notion that some important factors have not been measured and are therefore omitted from the model. However, here we see that a single DM technique—boosting—can dramatically raise the percentage of variance explained, compared to a conventional model, while using exactly the same predictors and data as the conventional model. In this case, the claim that measurement error and omitted variables are what reduce explained variance rings hollow.

Evidently, there is something about the predictive performance of these conventional models that is inferior to the DM approach. Boosting was able to find much more structure in the data than the conventional approach could. This was *not* due to overfitting, because these impressive fit statistics are not for the original random sample of *training* data that created the predictive model but for a completely separate random sample of data, the *test* data.

Schonlau's paper used artificially constructed data. We performed a similar analysis to see whether boosting performed as well with real-world data. A conventional OLS

TABLE 3.1 An Ordinary Least Squares Regression Predicting the Log of Personal Income

Total personal income (log)	Estimate	SE	p
Age	0.0674	0.0007	<.0001
Age squared (centered)	–0.0006	<0.0001	<.0001
Female	–0.5387	0.0021	<.0001
Occupational prestige	0.0211	<0.0001	<.0001
Black	–0.0626	0.0037	<.0001
Native American	–0.1804	0.0135	<.0001
Asian	–0.0489	0.0056	<.0001
Other race (ref = white)	–0.0162	0.0042	<.0001
Widowed	0.0707	0.0063	<.0001
Divorced	–0.0522	0.0028	<.0001
Separated	–0.1544	0.0061	<.0001
Married, spouse absent (ref = married, spouse present)	–0.1833	0.0070	<.0001
Non-citizen	–0.1609	0.0045	<.0001
Naturalized citizen (ref = citizen by birth)	0.0213	0.0042	<.0001
Less than HS	–0.2350	0.0041	<.0001
Some college, no degree	0.1019	0.0030	<.0001
Associate's	0.1528	0.0039	<.0001
Bachelor's	0.2913	0.0035	<.0001
More than bachelor's	0.4460	0.0042	<.0001
New England	0.0971	0.0051	<.0001
Mid-Atlantic	0.0951	0.0036	<.0001
East North Central	–0.0090	0.0034	<.0001
West North Central	–0.0172	0.0047	<.0001
East South Central	–0.0827	0.0047	<.0001
West South Central	–0.0050	0.0038	0.1820
Mountain	–0.0460	0.0047	<.0001
Pacific (ref = South Atlantic)	0.0712	0.0036	<.0001

NOTE: N of observations = 1,226,925; constant = 8.077; R^2 = 0.2882

regression is reported in table 3.1 above, in which the logarithm of personal income is predicted by several sociodemographic variables, using data from the U.S. Census Bureau's 2010 American Community Survey. Despite a large sample, numerous predictors, and technically high-quality data collection, the explained variance as represented by the regression R^2 is only 29%.

In table 3.2, this conventional model is compared with several DM models that used the same data. The first row reiterates the R^2 for the conventional OLS regression above, while the other rows report R^2 statistics for four different data mining models using the identical

TABLE 3.2 Performance of Standard OLS Regression versus Data Mining Models

Model type	Test sample R^2
OLS	.288
Partition tree	.442
Bootstrap forest	.438
Boosted tree	.436
Neural network	.481

data and variables. In each case, the DM approach explains considerably more variance than the conventional regression: it has much better predictive power (though we did not see as large an improvement as in Schonlau's example). These results use real data, but are presented here solely for illustrative purposes. If we had "tweaked" the DM models further, by adjusting various parameters, we could probably have increased the R^2 even further.

CALIBRATING

Calibrating is yet another DM strategy for improving model prediction that also diverges from conventional practices. One of the statistical assumptions underlying conventional regression modeling is that, across the spectrum of values of the dependent variable Y, the best estimate of Y is always the prediction (called $\hat{Y}$ or *Y-hat*) provided by the regression equation. Consequently, a plot of the predicted values of Y against the observed values of Y should be a straight line. If so, the model is said to be *calibrated*.

Unfortunately, in real-world data analyses, a plot or graph of the relation between Y and $\hat{Y}$ is often linear across much of the range of values of Y but diverges from a straight line at either high or low values of Y or both. The line curves. In this case, the regression model is *uncalibrated*—the model does not predict as accurately at extreme values of Y as it does in the midrange. In the conventional approach, a researcher would try to identify variables producing this curved pattern and add those to the regression model, hopefully causing the curvature to disappear.

DM sometimes uses a different approach. If a model is uncalibrated, as indicated by a curved plot of Y against $\hat{Y}$, then the researcher may fit a polynomial to Y ($Y = \hat{Y} + \hat{Y}^2 + \hat{Y}^3 + \ldots$) or use some other smoothing function, such as a spline. This procedure does not add anything to the substantive understanding of the relation between the various predictors and the dependent variable, because the researcher has not discovered why the curve is present. Nevertheless, this procedure does enhance the accuracy of prediction of Y and improves the fit of the model.

Table 3.3 provides an illustration of the effects of calibration on explained variance, using an OLS regression model predicting the log of earnings, where the predictors are

TABLE 3.3 The Effect of Calibration on Model Fit

	R^2	RMSE
Basic OLS regression model	.5237	2.28
Above + quadratic term: $\hat{Y}^2$	.5929	2.11
Above + cubic term: $\hat{Y}^3$	.5939	2.11
Above + quartic term: $\hat{Y}^4$	.5949	2.11

age, age squared, educational attainment (as a set of dummy variables), region, race, hours worked, and weeks worked. Again, the data are from the 2010 American Community Survey.

The addition of $\hat{Y}^2$, $\hat{Y}^3$, and $\hat{Y}^4$ terms in the regression equation increases the explained variance from 0.52 to 0.59, showing that calibration can produce an improvement in predictive accuracy.

Boosting and calibrating are common strategies in DM. Both illustrate the strong emphasis that DM places on improving prediction, and the way that this results in novel analytical strategies.

MEASURING FIT: THE CONFUSION MATRIX AND ROC CURVES

Data miners use the term *fit* to refer to the accuracy of a predictive model, and specifically the extent to which predicted values of a target or dependent variable are close to the observed values of that variable. The simplest measure of fit for a predictive model with a continuous dependent variable is the model's R^2 or adjusted R^2: the percentage of variance explained by the model. But when a predictive model has a dichotomous dependent variable, such as yes/no or zero/one, we need a different way of assessing fit. The most common format provided for assessing fit is called a *confusion matrix,* which is just a two-by-two table. The confusion matrix informs us how accurately the predictive model we have constructed performs in classifying cases. It compares the predicted outcome (yes/no) with the observed or actual outcome (yes/no).

In a real confusion matrix, there would be numbers in the four cells. In the example shown in table 3.4, we have represented the numbers as n_{11}, n_{12}, n_{21}, and n_{22}, just so we can point to specific cells. Note the following with regard to this table.

- Correctly predicted or correctly classified observations appear on the diagonal of the matrix: those cases that were predicted as negative and were actually observed as negative plus those that were predicted as positive and were observed as positive. For an accurate model, most of its cases should ideally appear on the diagonal.
- The percentage of observations correctly classified by the model is $(n_{11} + n_{22})/(n_{11} + n_{12} + n_{21} + n_{22})$.

TABLE 3.4 A Confusion Matrix

		Predicted outcome	
		Negative (0)	*Positive (1)*
Actual outcome	Negative (0)	n_{11}	n_{12}
	Positive (1)	n_{21}	n_{22}

- However, publications commonly report an overall classification *error rate* instead: $(n_{21} + n_{12})/(n_{11} + n_{12} + n_{21} + n_{22})$.
- Some articles report a measure called *sensitivity*, defined as $n_{22}/(n_{21} + n_{22})$.
- Some also report a measure known as *specificity*, defined as $n_{11}/(n_{11} + n_{12})$.
- The *false-positive rate* is defined as the proportion of predicted positives that were in reality negative: $n_{12}/(n_{12} + n_{22})$.
- The *false-negative rate* is defined as the proportion of predicted negatives that were in reality positive: $n_{21}/(n_{11} + n_{21})$.

In all predictive situations, there is an unavoidable trade-off in prediction between false-positive rate and false-negative rate, or between sensitivity and specificity. Minimizing the false-negative rate will necessarily increase the number of false positives. Conversely, avoiding false positives means that the rate of false negatives will increase.

USING A CONFUSION MATRIX FOR CLASSIFICATION DECISIONS

A logistic regression model with a zero/one dependent variable can report for each observation or case in the dataset the predicted probability that $Y = 1$. Those predicted probabilities will take a continuous range of values from zero to one. But where should a researcher set the "bar" or threshold probability above which an observation should be assumed to have a value of $Y = 1$ and, below that threshold, predicted to be $Y = 0$?

In statistical software, the bar is usually set at $p = .5$. So a logistic regression program treats all the observations with a predicted probability of .5 or greater as predictions that $Y = 1$, and all with probabilities less than .5 as predictions that $Y = 0$.

However, for most real-world decisions, one *would not* assume that the .5 cutoff is the best one for predicting, because there is often an asymmetry in the "costs" of false-positive predictions versus the costs of false-negative predictions. A false positive may cost you far more than a false negative, or vice versa, and this should inform your decision-point.

So how would you determine the decision-point, the predicted probability from your model at which to classify a case as $Y = 1$? Here is one example of the logic to use. Consider a banking situation where a decision has to be made as to whether to offer a $5,000 loan (table 3.5). The model is constructed to predict whether someone will default (not

TABLE 3.5 Adding Cost/Benefit Consideration to a Confusion Matrix

		Decision (prediction from model)	
		Withhold loan, fearing default	*Offer loan*
P_D	Defaults	\$0	–\$5,000
$1 - P_D$	Does not default	–\$200	+\$200

pay back the loan). Call P_D the probability that the applicant will default; then $1 - P_D$ is the probability that they will not.

In each cell is the *cost* of the decision regarding each outcome. This information has to be derived from outside the prediction model, from someone who understands the real-world context the model is operating in. If your predictive model indicates that the loan applicant will default and therefore you withhold the loan, you have lost nothing; hence \$0 is written in the top-left cell in the table. If your model predicts that the person will not default, so you give them a loan, but in fact they do default, you will have lost the \$5,000 you loaned out; hence \$5,000 in the top-right cell. If your model predicts that the person will default, so you refuse them the loan, but in fact they would have paid it off, then you will have missed out on \$200 in profit from interest (–\$200, lower-left cell). And finally, if the model correctly predicts that the person will not default, and you therefore grant them the loan, you will make \$200 in profit from interest (lower-right cell).

The expected value is $0(P_D) - 200(1 - P_D) - 5000(P_D) + 200(1 - P_D)$.

Thus, the decision point is where $-200(1 - P_D) = -5000(P_D) + 200(1 - P_D)$.

Rearranging and solving this equation gives $P_D = .074$. The profitable decision-point is to refuse the loan (expecting default) for any predicted P_D of .074 or greater. Note how different this is from assuming that any probability over .5 should be classified as a default, as the confusion matrix for most logistic regression software reports.

For further reading about including cost considerations in classification models, consult Witten, Eibe, and Hall (2001, 163). Adding cost considerations to a confusion matrix in order to decide on the optimum cut point is usually straightforward when costs and benefits have straight monetary value. Unfortunately, the tradeoff between false positives and false negatives, or between sensitivity and specificity, is sometimes difficult to express that way. Deciding where to set the cutoff for a new diagnostic health test is fraught with difficulty, since one has to balance the turmoil produced when a patient is falsely told they have some serious medical problem against the consequences of failing to identify that problem when it really is present.

ROC CURVES AS MEASURES OF FIT

A receiver operating characteristic (ROC) curve is a visual way of deciding which of several models best classifies cases. It is used in contexts where an outcome is zero/one or

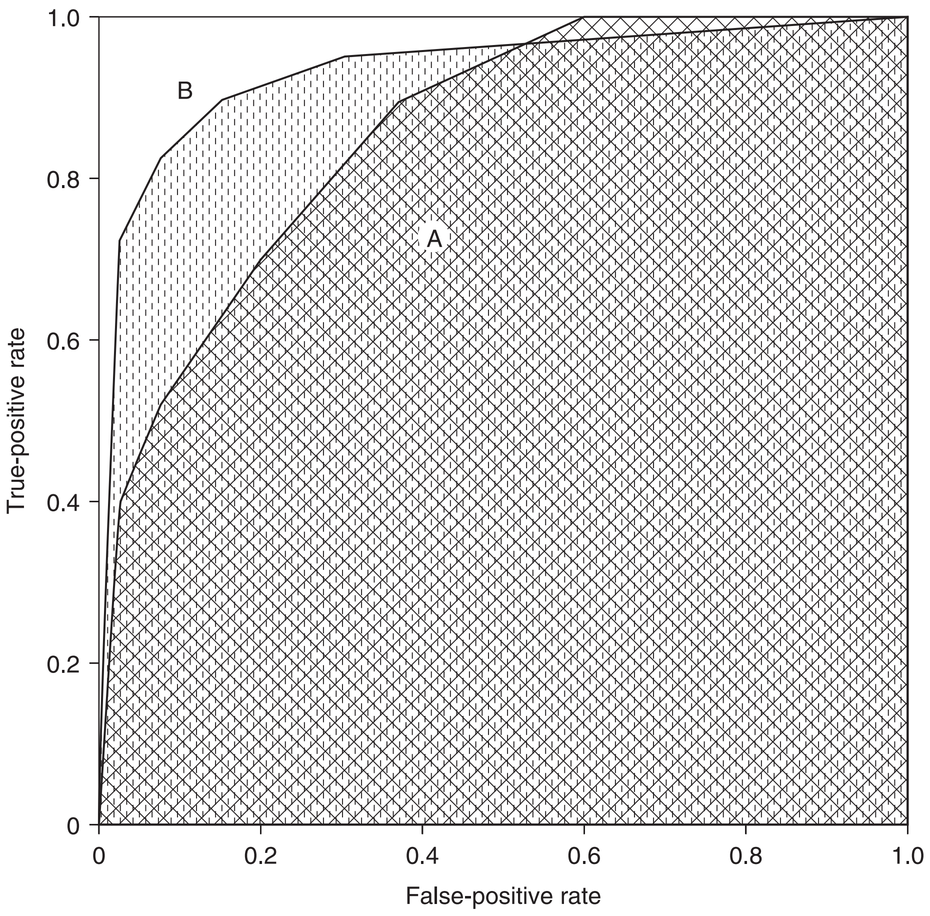

FIGURE 3.3
Examples of receiver operating characteristic curves (from Fawcett 2006).

yes/no and where a model provides a predicted probability of yes or $Y = 1$ for each case. (In DM terms this is a *binary classifier*. Many medical tests are binary classifiers, for example.) The ROC curve plots the rate of true positives (specificity) on the Y axis against false positives (1 – specificity) on the X axis. It therefore depicts the trade-off between true positives (benefit) and false positives (cost)—see figure 3.3.

In the figure, the model represented by line B is generally superior at classification to that represented by line A. But we can also see that where the false-positive rate is very high (greater than 0.6), model A has a better predictive performance (Fawcett 2006).

ROC curves are often used to understand the accuracy of diagnostic tests of some disease, such as a blood test. In addition to the predicted probability of having the disease, obtained from the blood test, one needs some separate objective information as to whether the person actually has the disease. The latter is called the "gold standard" in the medical literature.

An ideal model closely follows the *Y* axis on the left and then sharply turns parallel to the *X* axis. It gets as close to the top-left corner of the ROC diagram as possible. The area under such a curve is nearly 1. A very bad model would follow the 45-degree line: this is no better than chance, and the area under this line is 0.5. That diagnostic test would tell you nothing useful.

To summarize this section, DM often constructs predictive models, and the data miner wants some method for assessing the accuracy of a given model. First, the data miner applies a predictive model, which was derived from training data, to other data (other cases or observations) that were set aside as test data. Next, the data miner uses a confusion matrix or an ROC curve to understand the accuracy of the model in predicting a target. One important measure of fit or classification accuracy is the percentage of observations correctly classified (or conversely, the overall error rate). But the researcher often wishes to determine both the false-positive and false-negative rates, and sometimes uses this information, along with cost data, to decide the most suitable cutoff value to use with the predicted probability when classifying cases.

IDENTIFYING STATISTICAL INTERACTIONS AND EFFECT HETEROGENEITY IN DATA MINING

One desideratum underlying a conventional regression model is that the same pattern of association or correlation between variables holds for all the observations in a dataset. When the opposite situation occurs—when a dataset contains groups of observations for which there are very different relations between variables—then regression models can produce highly misleading coefficients. This is known colloquially as the "apples and oranges" problem or more technically as *heterogeneous effects*. For example, if the factors predicting graduation among students in community colleges are quite different from the factors associated with graduation at highly selective four-year colleges, then estimating a single statistical model for a dataset containing both types of students will yield misleading results.

The problem is not that there are different groups within a dataset. That will always be the case. The problem arises when certain subgroups or clusters of cases within a dataset have very different patterns of association between variables than other groups do. One dramatic example is known as Simpson's paradox (sometimes the Yule-Simpson paradox), the amalgamation paradox, or the reversal paradox (Blyth 1972). Two groups of observations in a dataset might both exhibit a *positive* relation between two variables, say *X* and *Y*. But when both groups are analyzed together, in the same model, the direction of the relation between *X* and *Y* reverses direction: *X* might appear to be *negatively* associated with *Y*.

An example is provided in table 3.6, in which outcomes are given for a hypothetical medical trial. The trial was carried out at two locations (sites A and B) and involved giving

TABLE 3.6 Blyth's Formulation of Simpson's Paradox

	Overall		Site A		Site B	
	Standard treatment	*Experimental treatment*	*Standard treatment*	*Experimental treatment*	*Standard treatment*	*Experimental treatment*
Total number	11,000	10,100	1,000	10,000	10,000	100
Number dead	5,950	9,005	950	9,000	5,000	5
Number alive	5,050	1,095	50	1,000	5,000	95
Survival rate	46%	11%	5%	10%	50%	95%

SOURCE: Blyth (1972).

some patients a new, experimental treatment and others the standard treatment. The first two columns compare the outcomes for the experimental treatment and the standard treatment for the trial as a whole, across the two sites. The clearly demonstrate that the survival rate was substantially lower for those receiving the experimental treatment. If we inspected only these two columns we would conclude that the experimental treatment is much worse than the standard treatment and that it therefore ought to be abandoned.

But when we move to the four columns to the right, we find that at *each of the two individual sites* the survival rate was substantially *higher* for those receiving the experimental treatment. This suggests that the experimental treatment is much more effective than the standard treatment. How can we reconcile this with what we see in the aggregated data? The answer is that the experimental technique was administered most frequently at a site with low survival rates for both groups, and the standard technique was administered disproportionately at a site with a much higher survival rates. When the data are combined, then, the higher survival rates achieved in the experimental groups disappear. Or, put another way, the observed negative bivariate relation between exposure to the experimental treatment and the odds of survival reverses when we condition on the site where one received treatment.

A less extreme but more common situation occurs when an observed regression coefficient for a predictor *X* appears to be small or even not statistically significant. This sometimes occurs because *X* is strongly positively associated with *Y* for one group or cluster of cases within the sample, while for another group the same predictor *X* may have no relation or even a negative relation with *Y*. Averaging these two effects, as regression does when analyzing the whole sample, results in a misleadingly small coefficient.

Datasets are often heterogeneous in this fashion, but the researcher does not normally know what the subgroups or clusters of cases are in advance, so the "apples and oranges" problem is endemic. Therefore, as a preliminary step in DM analysis, identifying groups or clusters of cases within the dataset is desirable, so that a researcher can subsequently either run separate analyses for each distinctive group or add interaction terms that model different slopes for each group or cluster (Melamed, Breiger, and Schoon 2013).

Several DM clustering techniques can identify clusters of observations with heterogeneous relations between one or more *X* variables and an outcome, *Y*. Robert Haralick and colleagues have developed a method using linear and nonlinear manifold clustering (Haralick and Harpaz 2007). Melamed, Breiger, and Schoon offer another solution using singular value decomposition. Unfortunately, these techniques are not yet available in any of the main DM software packages.

A third, more readily available solution uses a technique known as *latent class regression* or *latent class cluster models*. The term *latent class* is used because the heterogeneous groups within a dataset may not be defined by a single measured variable. (If men differ from women in some regression model, or if older and younger respondents show a different pattern of association between variables, that can be relatively easy to identify, since those are single observed variables.) When the subgroups in the data are defined in more complex ways, however, we conceptualize the subgroups as having different values on some unobserved (hence "latent") variable. So how does one identify such subgroups?

Statistical Innovations provides a software package called Latent GOLD that accomplishes this kind of analysis in a clear and user-friendly fashion. The software is described on their website (http://statisticalinnovations.com/products/latentgold.html). Among statisticians, this topic is often called *finite mixture modeling*, and in the last decade considerable advances have been made in developing this technique. Collins and Lanza (2009) have written a useful book about the statistical ideas behind the method.

After identifying distinctive clusters or groups of observations within a dataset, a researcher may decide to analyze all the clusters in separate models. Alternatively, clusters can be represented by a nominal variable and new terms added to the model representing interactions between each cluster and particular predictors. A single model that includes these interaction terms can then be calculated to allow for the groups' heterogeneous effects.

BAGGING AND RANDOM FORESTS

The end product in the conventional statistical paradigm is usually a single regression or similar model that summarizes the relations in a dataset. That model may have gone through several cycles of improvement and modification, but at the end of the day a single model represents the best that a researcher is able to come up with.

By contrast, a DM analysis often follows a different logic, generating several *different* predictive models and combining their results to provide the best possible prediction, a process known in DM as *ensemble learning* (Berk 2006). There are alternative strategies within DM for creating these multiple models and for combining them. One of these—*bagging* (not to be confused with *binning*)—treats a dataset as if it were a population rather than a sample. It draws multiple random samples with replacement from the dataset. For each of those random samples, the DM application fits a model, and from that model calculates a predicted value of the outcome variable for each case or observation. Predictions from those different models can then be averaged to yield a best possible prediction, either for the original dataset or for new out-of-sample observations.

A related approach known as *random forests* is used to aggregate the results of several decision trees. The basic idea is to generate several tree models and average their results to obtain a best prediction. The novel aspect of random forests is that the researcher forces a different subset of predictors to be included in each model, so that each model cannot have an identical structure or content to the previous one. The varied predictions obtained from those multiple models are then combined to yield a best estimate.

Bagging and random forests are optional procedures within JMP and several of the other DM suites discussed above. Examples will be provided in a later chapter.

One rationale for this DM practice of estimating several models and averaging their findings is that in some cases model-building may be *path dependent*. In several kinds of DM models, an algorithm checks each feature to discover which one is the single most powerful predictor of a target. It keeps the most powerful one and then cycles back through the remaining predictors, to select the second-most powerful predictor, and so on for several iterations until it has selected a set of features or variables that collectively maximize the overall predictive power of the model.

This is a widely used and appropriate method for variable or feature selection. However, it has one potential pitfall. Once such an algorithm has chosen a first predictor to be entered into the model, this makes it more likely that some variables will be chosen as the second predictor, compared with some others. For example, the algorithm is unlikely to choose as the second predictor a variable that is very highly correlated with the first variable it chose, since the addition of a second highly related predictor will not enhance the predictive power very much. In other words, the choice of the first variable to some degree sets a path for the remaining iterations of the program, hence the term *path dependence*.

Path dependence implies that certain valuable predictors might be overlooked or omitted in any single model. Hence it makes sense to estimate several models, which are constrained to pick different predictors in each case, thereby avoiding the possibility that certain predictors are overlooked. That is what random forests accomplish.

A related logic informs DM procedures that involve iterative approximations that converge on an optimal solution. One weakness of such algorithms is that, depending upon where the first "guess" is located, the program may sometimes converge upon a local optimum solution that is not the best solution overall.

This is best understood graphically. In figure 3.4, the *Y* axis represents some measure of error, so the program seeks a solution with the lowest possible value on the *Y* axis. The *X* axis represents the value of some parameter being estimated, and the curve represents the path that a program might follow in seeking a solution, a best estimate of *X*. If the program's initial guess or estimate is on the left of the diagram, at a low value of *X*, an iterative process that chooses each subsequent solution to be slightly lower on *Y* will

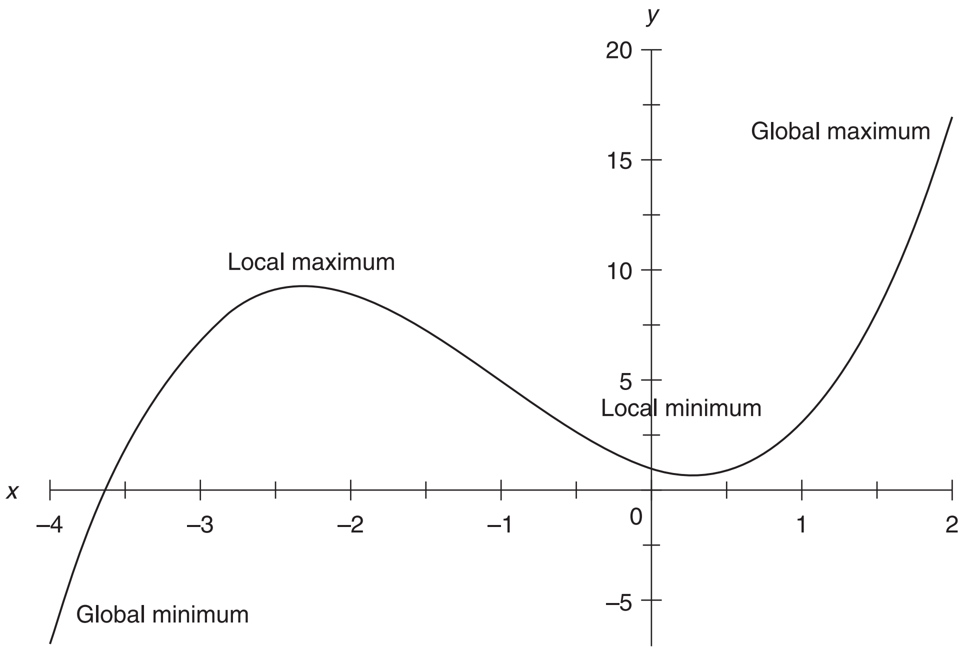

FIGURE 3.4
Global and local minima.

follow the line downwards until it reaches the global minimum—the best possible answer. It will not move up the line when the line turns upward because it is programmed to keep looking for lower values of *Y* and to stop when it cannot find a lower value. The algorithm will stop at the lowest *Y*, the global minimum, where the estimate of *X* is about –4.

But if the program's first guess or estimate of *X* happens to be on the right side of the diagram, at a higher value of *X* (say 1.5), the iterative process will lead down-slope to a *local minimum*. Because the algorithm always tries to decrease *Y*, it will not move back up the line after it reaches the first low point around $X = 0.3$, and so it will miss the next "valley" where the global minimum resides. It will settle therefore on the local minimum (around $X = 0.3$), "believing" erroneously that this is the best solution, the estimate of *X* that minimizes *Y*.

One antidote to this problem with iterative methods is to estimate several different models, each of which begins at a very different starting point (an initial guess or estimate), and to aggregate the predictions from all these different models to determine one's final prediction. This procedure will not prevent some solutions from being suboptimal (because they have settled on a local minimum), but it does ensure that there are many other chances to reach the true or optimal solution (the global minimum) and that those will predominate.

JMP Pro and other DM software packages ask the user how many starting points to use. They then run separate models starting at very different starting points, to ensure that they do not get fooled by local minima. The cost, of course, is that they have to run many models instead of one, which may take a lot of processing time with big datasets.

· · ·

To summarize this section on bagging and random forests, researchers working within the DM paradigm have discovered the value of analyzing data several times over, using slightly different samples, or different sets of predictors, or different beginning points. Each individual analysis yields an estimate, and the most robust and accurate prediction is found to be some combination of those estimates. To use an analogy, this is decision by committee, and the claim is made that "voting," averaging, or in some other fashion combining multiple models (also known as *ensemble learning*) provides a more accurate prediction than depending upon a single model or analysis. However, repeating analyses in this fashion and then combining their results is predicated on having fast, capacious computers that can calculate models many times over, a requirement for many DM methods.

THE LIMITS OF PREDICTION

Two eminent DM practitioners have published best-selling books about the limitations of data mining and prediction. Nassim Nicholas Taleb, a financial analyst and statistical modeler, is the author of *The Black Swan* (2005) and *Fooled by Randomness* (2007). Nate Silver, a developer of predictive software in baseball who is also a leading election poll analyst (see the *FiveThirtyEight* blog of the *New York Times*), authored *The Signal and the Noise* (2012).

Both authors provide cautionary points, such as the following:

- Not all natural or social phenomena have an underlying structure that can be discovered. In general, the greater the ratio of noise to signal, the greater the risk that data miners may be misled by overfitting. They may be "fooled by randomness" into seeing a mirage or finding nonexistent structure. (That's why cross-validation and replication are so important.)
- Very dynamic interconnected systems are affected by multiple causes, some of which can trigger positive feedback loops that result in rapid, large-scale, unanticipated change. "Does the Flap of a Butterfly's Wings in Brazil Set Off a Tornado in Texas?" was the title of a famous presentation about *chaos theory*, a theory about nonlinear systems credited to Edward Lorenz. As the imagery suggests, a very small-scale change in one place can trigger very large-scale consequences elsewhere. This imagery might be taken to imply that nonlinear systems cannot be successfully modeled at all. On the contrary, nonlinear systems such as the weather can be predicted, within certain limits (argues Silver), but

only within a time frame shortly before the predicted event. Predictions made earlier than this will be completely inaccurate. In other words, one cannot trace any actual tornado back to the flap of a butterfly's wings. One can *imagine* the link, but one cannot *model* that far back. One can, however, fairly accurately predict a hurricane from evidence collected a few days before it happens.

- Other natural systems manifest regularities but are beyond our current predictive understanding. The timing and severity of earthquakes is one example that Silver describes. We can identify certain patterns about the magnitudes of earthquakes, but we cannot accurately predict *when* large earthquakes will happen. The pattern, if there is one, eludes us.
- Certain social phenomena are unsuited to prediction because the actors involved are scanning their environment and responding to any hints of change from the status quo. Under that circumstance, once a hint of movement is perceived—for example, a few stock prices start to rise or fall—many people will anticipate that the market is changing its direction and will jump on the bandwagon. The movement may then become a self-fulfilling prophecy as more people buy or sell off stocks. In such volatile contexts, yesterday's behavior is a poor predictor of tomorrow's action, since participants may gain monetarily from anticipating a change in direction. In this context, a herd mentality can undermine prediction.
- In other contexts, however, the opinions of a collectivity can rival the predictions of individual experts. The average of people's predictions on many issues is frequently superior to most individuals' estimates. That is why information "markets" where numerous individuals bet on outcomes often perform as well as expert decision-makers.
- Experts are often overconfident of their predictions. Equally, individuals with a stake in the continuation of a status quo tend to understate or overlook the risk of change.

These authors' two most crucial points, in our opinion, are these:

- Any prediction should always be accompanied by a probability or confidence interval that represents the uncertainty of the prediction.
- Statistically highly unlikely outcomes, though infrequent, do occur and should therefore be expected and planned for. A one-in-a-million occurrence will happen at some point. Taleb refers to such events as *black swans* and notes their destructive impact on people who make decisions assuming that statistically rare outcomes won't ever happen. One of the causes behind market crashes and bankruptcies is the tendency among quantitatively skilled decision-makers to act as if only the most statistically probable events will occur. Their plans (and fortunes) are devastated when a black swan—an unlikely event—comes to pass.

BIG DATA IS NEVER BIG ENOUGH

The portrait we have drawn of DM suggests that the combination of brute computing power and very large datasets enables data miners to discover structures in data that would not have been revealed by applying conventional statistical approaches to datasets containing smaller numbers of cases. We stand by that claim, but it is also important to acknowledge a paradox that data miners continually face and which molds their entire enterprise, namely that even the largest social science datasets—for example the five-million-person, multi-year census files available from the American Community Survey—are not large enough to allow a comprehensive or exhaustive search for structure, and even the biggest, fastest computers find certain empirical tasks intractable.

One consequence of this paradox is that data mining frequently has to make simplifying assumptions to keep problems tractable, or select subsets of variables, because even DM cannot handle all the available measures in one model. Why, given the huge computation and data resources available, does DM still have to make compromises or cut corners or find ingenious ways to estimate or approximate, rather than measuring things directly and exhaustively?

A thought experiment can demonstrate what is at stake. Imagine a situation where we have a dichotomous (yes/no) target or dependent variable and we have determined through some process of exploratory research that, taken together, 10 variables or features provide a good prediction of this yes/no target. Now, for argument's sake, imagine that each of these 10 predictors takes values from 0 through 9. (Say we took continuous predictors such as age or income and split each of them into 10 bins, turning each into an ordinal variable with 10 possible values.)

Hypothetically, one could construct a table, with one row for every possible combination of these 10 ten-valued predictor variables or features. Each of the cases or observations in a training dataset could be sorted, ending up in the single row that represented that case's values on all 10 predictors. After the training data were all sorted in this fashion, the proportion of yes responses for that row could be counted.

This table of training data could then serve as a predictive model. To predict the target (yes or no) for each new case in a test sample, one would just look up in the table the particular row that corresponded to the individual pattern of independent variables for that new case or observation. (For example, look up the row for men, age 65–70, income between $75K and $80K, living in New England, etc., for all 10 of a person's variables.) The proportion of yes cases in that row would then provide the predicted probability of yes for a particular new case in the test data. This look-up process could be repeated for each case in the new data file.

Why isn't this kind of exhaustive empirical prediction strategy feasible with big data? Consider what data miners would call the *measurement space*: the size of the table needed to represent all combinations of 10 variables, each with 10 values. It would have to be 10^{10} in size: ten billion rows in total. Also, consider what amount of training data would be needed such that (say) a hundred cases or observations would be available for each row within the table from which to count the proportion of yes answers for prediction to new data. One would need a training data set of a hundred times ten billion cases—a trillion cases—to fill up the table sufficiently to allow for a purely empirical look-up strategy.

Even big data is not big enough for an *exhaustive* prediction strategy; short of astronomy, we are unlikely to find datasets with a trillion cases. So data mining is unable to handle an exhaustive direct measurement strategy for an imagined problem of 10 variables, each with 10 values.

What this thought experiment is intended to communicate is that DM faces important limitations stemming from the size of training data and also from computational load. On the other hand, DM has several strategies that successfully avoid these problems and enable DM methods to successfully analyze data with hundreds of variables.

- First, DM places a high premium on reducing the number of variables that will go into any model. One approach involves *feature selection*: scanning through large numbers of variables to discover which small subset is the most powerful for predicting a given target, and dropping the rest.
- A second approach involves combining some predictor variables into indices, scales, or factors, a process called *feature extraction*.
- A third approach avoids a huge measurement space by realizing that many of the possible combinations of values of variables will not be important in practice, either because there aren't many cases with those particular combinations, or because one can obtain good estimates of the effects of individual variables without considering all their possible interactions or combinations of values.
- *Data partitioning* (or *decision trees*) is one example. These methods search for statistical interactions between variables, but they do not exhaustively consider the entire measurement space with its millions of possible interactions or cells. Instead, they work one variable at a time, initially picking the single variable that best partitions the data on Y, then recursively finding additional variables to partition the data further. These techniques do find interactions that matter in terms of predicting Y, but by proceeding one variable at a time they identify maybe a hundred important combinations of values of variables (or interactions), rather than billions. Tree or recursive partitioning methods select a subset of predictors and interactions from a larger number of possible predictors and interactions.
- *Neural network* models play a similar role. They can incorporate complex interactions between predictors into their predictive models in an automated fashion, without the data analyst having to specify those ahead of time.
- Finally, some methods take advantage of the fact that a huge measurement space can be drastically reduced if we make a simplifying assumption, namely that

> each variable affects a dependent variable independently of each other variable, or more precisely that predictors are *conditionally independent*. This is equivalent to saying that interactions between predictors do not matter. Methods that follow this simplifying assumption, including the naive Bayesian classifier (discussed in a later chapter), are fairly accurate in some contexts.

Let's summarize our argument with respect to "big data is never big enough." We began by noting that, in principle, DM methods could take an "exhaustive" approach to discovering structure and prediction, for example by considering every possible interaction between predictors, or by using every available predictor. We used a thought experiment to show that an exhaustive strategy often isn't possible as a practical matter, because the number of combinations or interactions between predictors becomes astronomically large, so large that no dataset is going to have enough cases to cover all the combinations. Faced with that, DM methods adopt search strategies that are not exhaustive. Although they still try out many possible models, they usually don't try out *all* possibilities. In practice, DM reduces the measurement space or the number of possibilities considered. This is accomplished in several ways: (1) by initially selecting a subset of important predictors from a larger list—*feature selection*; (2) by combining variables into scales or other composites—*feature extraction*; (3) by sometimes ignoring interactions among predictors to obtain an simpler but perhaps still accurate prediction; and (4) by looking for clusters of similar cases in the data, and analyzing each cluster or group separately.