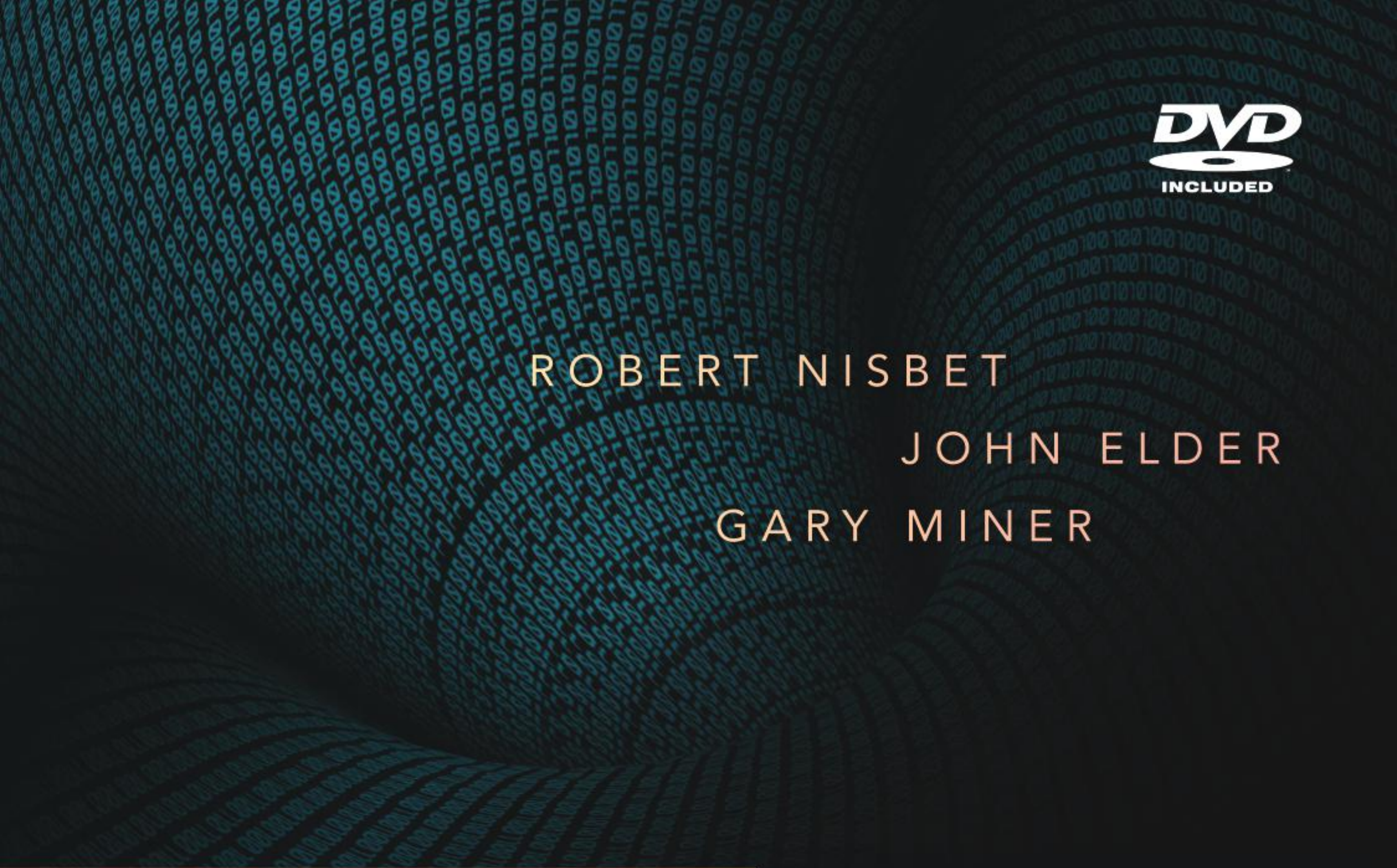

DVD INCLUDED

ROBERT NISBET

JOHN ELDER

GARY MINER

HANDBOOK OF Statistical Analysis & Data Mining Applications

www.elsevierdirect.com/datamining ISBN: 978-0-123747655

CHAPTER

20

Top 10 Data Mining Mistakes

OUTLINE

PREAMBLE

Mining data to extract useful and enduring patterns remains a skill arguably more art than science. Pressure enhances the appeal of early apparent results, but it is all too easy to fool yourself. How can you resist the siren songs of the data and maintain an analysis discipline that will lead to robust results? It is essential to *not* lack (proper) data, focus on training, rely on one technique, ask the wrong question, listen (only) to the data, accept leaks from the future, discount pesky cases, extrapolate (practically and theoretically), answer every inquiry, sample casually, or believe the best model.

INTRODUCTION

It has been said that good judgment comes through experience, but experience seems to come through bad judgment! In two decades of mining data from diverse fields, we have made many mistakes, which may yet lead to wisdom. In the following sections, we briefly describe, and illustrate from examples, what we believe are the "Top 10" mistakes of data mining, in terms of frequency and seriousness. Most are basic, though a few are subtle. All have, when undetected, left analysts worse off than if they'd never looked at their data.

After compiling the list, we realized that an even more basic problem—mining without (proper) data—must be addressed as well. So, numbering like a computer scientist (with an overflow problem), here are mistakes 0 to 10.[1]

0. LACK DATA

To really make advances with an analysis, you must have labeled cases, i.e., an output variable. With input variables only, all you can do is look for subsets with similar characteristics (cluster) or find the dimensions that best capture the data variation (principal components). These unsupervised techniques are much less useful than a good (supervised) prediction or classification model. Even with an output variable, though, the most interesting class or type of observation is usually the most rare by orders of magnitude. For instance, roughly 1/10 of "risky" individuals given credit will default within 2 years, 1/100 people mailed a catalog will respond with a purchase, and perhaps 1/10,000 banking transactions of a certain size require auditing. The less probable the interesting events, the more data it takes to obtain enough to generalize a model to unseen cases. Some projects probably should not proceed until enough critical data are gathered to make them worthwhile.

For example, on a project to discover fraud in government contracting, known fraud cases were so rare that strenuous effort could initially only reduce the size of the haystack in which the needles were hiding.[2] That is, modeling served to assure that the great majority of contracts were almost surely not fraudulent, which did enable auditors to focus their effort. But more known fraud cases—good for data miners, but bad for taxpayers—could have provided the modeling traction needed to automatically flag suspicious new cases much sooner. This was certainly the situation on another project, which sought to discover collusion on tax fraud. Unfortunately (for honest taxpayers), there were plenty of training

[1] Most examples are from our own and our colleagues' experiences, but some identifying details are mercifully withheld.

[2] Virtually all known cases were government workers who had, out of guilt, turned themselves in. Most, it seems, meant to pay back what they had fraudulently obtained (but how?). One audacious fraudster was discovered, however, after coworkers realized that the clerk had been driving a different sports car to work every day of the week!

examples, but their presence did lead to stronger, immediate modeling results, which was ultimately beneficial to taxpayers.

You can't mine without data, but not just any data will work. Many data mining projects have to make do with "found" data, not the results of an experiment designed to illuminate the question studied. It's like making a salad out of weeds found in the yard.

One sophisticated credit-issuing company realized this when seeking to determine if there was a market for its products in the class of applicants previously routinely dismissed as being too risky. Perhaps a low-limit card would be profitable, and even help a deserving subset of applicants pull themselves up in their credit rating?[3] But the company had no data on such applicants by which to distinguish the truly risky from those worth a try; its traditional filters excluded such individuals from even initial consideration. So the company essentially gave (small amounts of) credit almost randomly to thousands of risky applicants and monitored their repayments for 2 years. Then it built models to forecast defaulters (those late on payments by 90+ days) trained only on initial application information. This large investment in *creating* relevant data paid off in allowing the company to rationally expand its customer base.

1. FOCUS ON TRAINING

Only out-of-sample results matter; otherwise, a lookup table would always be the best model. Researchers at the MD Anderson medical center in Houston a decade ago used neural networks to detect cancer. Their out-of-sample results were reasonably good, though worse than training, which is typical. They supposed that longer training of the network would improve it—after all, that's the way it works with doctors—and were astonished to find that running the neural network for a week (rather than a day) led to only slightly better training results and much worse evaluation results. This was a classic case of overfit,

[3] For a decade now, the credit industry has mailed over a billion offers a year to American households; the high-risk market was one of the few places not saturated a few years ago. Credit profits are nonlinear with risk, and remind us of the triage system established during the Napoleonic wars, when the *levee en masse* swelled the battlefields and, combined with the new technology of cannons, etc., led to an army's medical resources being completely overwhelmed. Battlefield wounds were classified into three levels: the most minor to be passed by and treated later (if at all), more serious to receive immediate attention, but the most serious were judged not likely to be worth a physician's time. (We can envision a combatant, aware of hovering between the latter two classes insisting, like the Black Knight in the *Monty Python* movie, "What? The leg gone? It's just a flesh wound!") Likewise, credit companies make the most profit on individuals in the middle category of "woundedness"—those who can't pay off their balance, but keep trying. But they lose 5–10 times as much on clients just a little worse off, who eventually give up trying altogether. So, for models to be profitable at this edge of the return cliff, they have to forecast very fine distinctions. Recent downturns in the economy have severely punished the stocks of companies that aggressively sought that customer niche—especially if they did not give obsessive attention to model quality.

where obsession with getting as much as possible out of training cases focuses the model too much on the peculiarities of that data to the detriment of inducing general lessons that will apply to similar, but unseen, data. Early machine learning work often sought, in fact, to continue “learning” (refining and adding to the model) until achieving exact results on known data—which, at the least, insufficiently respects the incompleteness of our knowledge of a situation.

The most important way to avoid overfit is to reserve data. But since data—especially cases of interest—are precious, you must use resampling tools, such as bootstrap, cross-validation, jackknife, or leave-one-out. Traditional statistical significance tests are a flimsy defense when the model structure is part of the search process, though the strongest of penalty-based metrics, such as Bayesian Information Criterion or Minimum Description Length, can be useful in practice.

With resampling, multiple modeling experiments are performed, with different samples of the data, to illuminate the distribution of results. If you were to split the data into training and evaluation subsets a single time, the evaluation accuracy result might largely be due to luck (either good or bad). By splitting it, say, 10 different ways and training on the 90% sets and evaluating on the out-of-sample 10% sets, you have 10 different accuracy estimates. The mean of this distribution of evaluation results tends to be more accurate than a single experiment, and it also provides, in its standard deviation, a confidence measure.

Note that resampling evaluates whatever is held constant throughout its iterations, or “folds.” That is, you can set the structure (terms) of a model and search for its parameter values over multiple data subsets; then the accuracy results would apply to that fixed model structure. Or you could automate multiple stages of the process—e.g., outlier detection, input selection, interaction discovery—and put that whole process inside a resampling loop. Then it’s the accuracy distribution of that full process that is revealed.

In the end, you have multiple overlapping models; which is the model to use? One approach is to choose a single model (perhaps by its beauty rather than its accuracy). Another is to rerun the model-building process with all the data and assume that the resulting model inherits the accuracy properties measured by the cross-validation folds. A third is to use the variety of models in an ensemble, as discussed on Mistake 10, and in Chapter 13.

2. RELY ON ONE TECHNIQUE

“To a little boy with a hammer, all the world’s a nail.” All of us have had colleagues (at least) for whom the best solution for a problem happens to be the type of analysis in which they are most skilled! For many reasons, most researchers and practitioners focus too narrowly on one type of modeling technique. But, for best results, you need a whole toolkit. At the very least, be sure to compare any new and promising method against a stodgy conventional one, such as linear regression (LR) or linear discriminant analysis (LDA). In a study of articles in a neural network journal over a 3-year period (about a decade ago), only

17% of the articles avoided mistakes 1 and 2. That is, five of six referred articles either looked only at training data or didn't compare results against a baseline method, or made both of those mistakes. We can only assume that conference papers and unpublished experiments, subject to less scrutiny, are even less rigorous.

Using only one modeling method leads us to credit (or blame) it for the results. Most often, it is more accurate to blame the data. It is unusual for the particular modeling technique to make more difference than the expertise of the practitioner or the inherent difficulty of the data—and when the method will matter strongly is hard to predict. It is best to employ a handful of good tools. Once the data are made useful—which usually eats most of your time—running another algorithm with which you are familiar and analyzing its results adds only 5–10% more effort. (But to your client, boss, or research reviewers, it looks like twice the work!)

The true variety of modeling algorithms is much less than the apparent variety, as many devolve to variations on a handful of elemental forms. But there are real differences in how that handful builds surfaces to "connect the dots" of the training data, as illustrated in Figure 18.3 for five different methods—*decision tree, polynomial network, Delaunay triangles* (Elder, 1993), *adaptive kernels*, and *nearest neighbors*—on (different) two-dimensional input data. Surely some surfaces have characteristics more appropriate than others for a given problem.

Figure 18.1 (after Elder & Lee, 1997) reveals this performance issue graphically. The relative error of five different methods—*neural network, logistic regression, linear vector quantization, projection pursuit regression,* and *decision tree*—is plotted for six different problems from the Machine Learning Repository.[4] Note that "every dog has its day"; that is, that every method wins or nearly wins on at least one problem.[5] On this set of experiments, *neural networks* came out best, but how do you predict beforehand which technique will work best for your problem?[6] Best to try several and even use a combination (as covered in Mistake 10 and Chapter 13).

[4] The worst out-of-sample error for each problem is shown as a value near 1 and the best as near 0. The problems, along the x-axis, are arranged left-to-right by increasing proportion of error variance. So the methods differed least on the Pima Indians Diabetes data and most on the (toy) Investment data. The models were built by advocates of the techniques (using *S* implementations), reducing the "tender loving care" factor of performance differences. Still, the UCI ML repository data are likely overstudied; there are likely fewer cases in those data sets than there are papers employing them!

[5] When one of us used this colloquialism in a presentation in Santiago, Chile, the excellent translator employed a quite different Spanish phrase, roughly "Tell the pig Christmas is coming!" We had meant every method has a situation in which it celebrates; the translation conveyed the concept on the flip-side: "You think you're something, eh pig? Well, soon you'll be dinner!"

[6] An excellent comparative study examining nearly two dozen methods (though nine are variations on *decision trees*) against as many problems is (Michie et al., 1994) reviewed by (Elder, 1996). Armed with the matrix of results, the authors even built a decision tree to predict which method would work best on a problem with given data characteristics.

3. ASK THE WRONG QUESTION

It is first important to have the right project goal; that is, to aim at the right target. This was exemplified by a project at Shannon Labs, led by Daryl Pregibon, to detect fraud in international calls. Rather than use a conventional approach, which would have tried to build a model to distinguish (rare but expensive) fraud from (vast examples of) nonfraud, for any given call, the researchers characterized normal calling patterns for *each account* (customer) separately. When a call departed from what was the normal pattern for that account, an extra level of security, such as an operator becoming involved, was initiated. For instance, if one typically called a few particular countries each week, briefly, during weekdays, a call to a different region of the world on the weekend would bear scrutiny. Efficiently reducing historical billing information to its key features, creating a mechanism for the proper level of adaptation over time, and implementing the models in real time for vast streams of data provided interesting research challenges. Still, the key to success was asking the right question of the data. The ongoing "account signature" research won technical awards (at KDD, for example) but, more importantly, four researchers, part time in a year, were able to save their company enough money to pay the costs of the entire Shannon Labs (of 400 people) for the next year[7]—an impressive example of data mining return on investment (ROI).

Even with the right project goal, it is essential to also have an appropriate model goal. You want the computer to "feel" about the problem like you do—to share your multifactor score function, just as stock grants or options are supposed to give key employees a similar stake as owners in the fortunes of a company.[8] But analysts and tool vendors almost always use squared error as the criterion, rather than one tailored to the problem. Lured by the incredible speed and ease of using squared error in algorithms, we are like drunks looking for lost keys under the lamppost—where the light is better—rather than at the bar where they were likely dropped.

For instance, imagine that we're trying to decide whether to invest in our company's stock as a pension option, and we build a model using squared error. Say it forecasts that the price will rise from $10 to $11 in the next quarter, and it goes on to actually rise to $14. We've enjoyed a positive surprise; we expected a 10% gain but got 40%.[9] But when we're entering in that data for the next go-round, the computer has a different response; it sees an error of $3, between the truth and the estimate, and squares that to a penalty of 9. It would have more

[7] They were rewarded, as we techno-nerds like, with bigger toys. The group got a "Data Wall"—a 10′ × 20′ computer screen, complete with couch, with which to visualize data. As it was often commandeered by management for demonstrations, the research group was eventually provided a second one to actually use.

[8] Unfortunately, the holder of an option has a different score function from the owner of the stock. The option is very valuable if the company thrives, but only worthless if it doesn't. Yet, the owner can be seriously hurt by a downturn. Thus, a manager's rational response to having options is to take on increased risk—to "shoot the moon" for the potential up-side reward. To better align owner and management interests, it is better to grant stock outright, rather than (cheaper, but more inflammatory) options.

[9] Of course, our joy is short-lived, as we kick ourselves for not mortgaging the house and betting even more! Fear and greed are always at war when dealing with the markets.

than twice "preferred" it if the price had dropped –$1 to $9; then its squared error would only have been 4. A criterion that instead punishes negative errors much more than positive errors would better reflect our preferences.

Though conventional squared error can often put a model into a serviceable region of performance, the function being optimized has a thorough effect on the suitability of the final model. "Inspect what you expect," a retired IBM friend often says about managing projects. Similarly, you won't produce the best-spelling students if your grading has focused on penmanship. When performance is critical, have the computer do not what's easiest for it (and thereby, us) but what's most useful. To best handle custom metrics, analysts need a strong multidimensional (and preferably, multimodal) optimization algorithm; still, use of even simple random search with a custom score function is usually better than not customizing.[10]

4. LISTEN (ONLY) TO THE DATA

Inducing models from data has the virtue of looking at the data afresh, not constrained by old hypotheses. But, while "letting the data speak," don't tune out received wisdom. Experience has taught these once brash analysts that those familiar with the domain are usually more vital to the solution of the problem than the technology we bring to bear.

Often, nothing *inside* the data will protect you from significant, but wrong, conclusions. Table 20.1 contains two variables about high school, averaged by state: cost and average SAT score (from about 1994). Our task, say, is to model their relationship to advise the legislature of the costs of improving our educational standing relative to nearby states. Figure 20.1 illustrates how the relationship between the two is significant: the LR *t*-statistic is over 4, for example, suggesting that such a strong relationship occurs randomly only 1/10,000 times. However, the sign of the relationship is the opposite of what was expected. That is, to improve our standing (lower our SAT ranking), the graph suggests we need to reduce school funding!

Observers of this example will often suggest adding further data—perhaps, for example, local living costs, or percent of the population in urban or rural settings—to help explain what is happening. But the real problem is one of self-selection. The high-SAT/low-cost states are clustered mainly in the Midwest, where the test required for state universities (the best deal for one's dollar) is not the SAT but the ACT. Only those students aspiring to attend (presumably more prestigious) out-of-state schools go to the trouble of taking an extra standardized test, and their resulting average score is certainly higher than the larger population's would be. Additional variables in the database, in fact (other than proportion of students taking the SAT), would make the model more complex and might obscure the fact that information external to the data is vital.

[10] (Elder, 1993) introduced a global search algorithm for multimodal surfaces that updates a piecewise planar model of the score surface as information is gathered. It is very efficient, in terms of function evaluations, but its required overhead restricts it to a handful of dimensions (simultaneous factors) in practice. The need remains for efficient, higher-capacity global search methods.

TABLE 20.1 Spending and Rank of Average SAT Score by State

USA State	SAT Rank	$ Spent
AK	31	7877
AL	14	3648
AR	17	3334
AZ	25	4231
CA	34	4826
CO	23	4809
CT	35	7914
DC	49	8210
DE	37	6016
FL	40	5154
GA	50	4860
HI	44	5008
IA	1	4839
ID	22	3200
IL	10	5062
IN	47	5051
KS	6	5009
KY	18	4390
LA	16	4012
MA	33	6351
MD	32	6184
ME	41	5894
MI	20	5257
MN	3	5260
MO	13	4415
MS	12	3322
MT	19	5184
NB	8	4381
NC	48	4802
ND	2	3685
NH	28	5504
NJ	39	9159
NM	15	4446
NV	29	4564
NY	42	8500
OH	24	5639
OK	11	3742
OR	26	5291
PA	45	6534
RI	43	6989
SC	51	4327
SD	5	3730
TN	9	3707
TX	46	4238
UT	4	2993
VA	38	5360
VT	36	5740
WA	30	5045
WI	7	5946
WV	27	5046
WY	21	5255

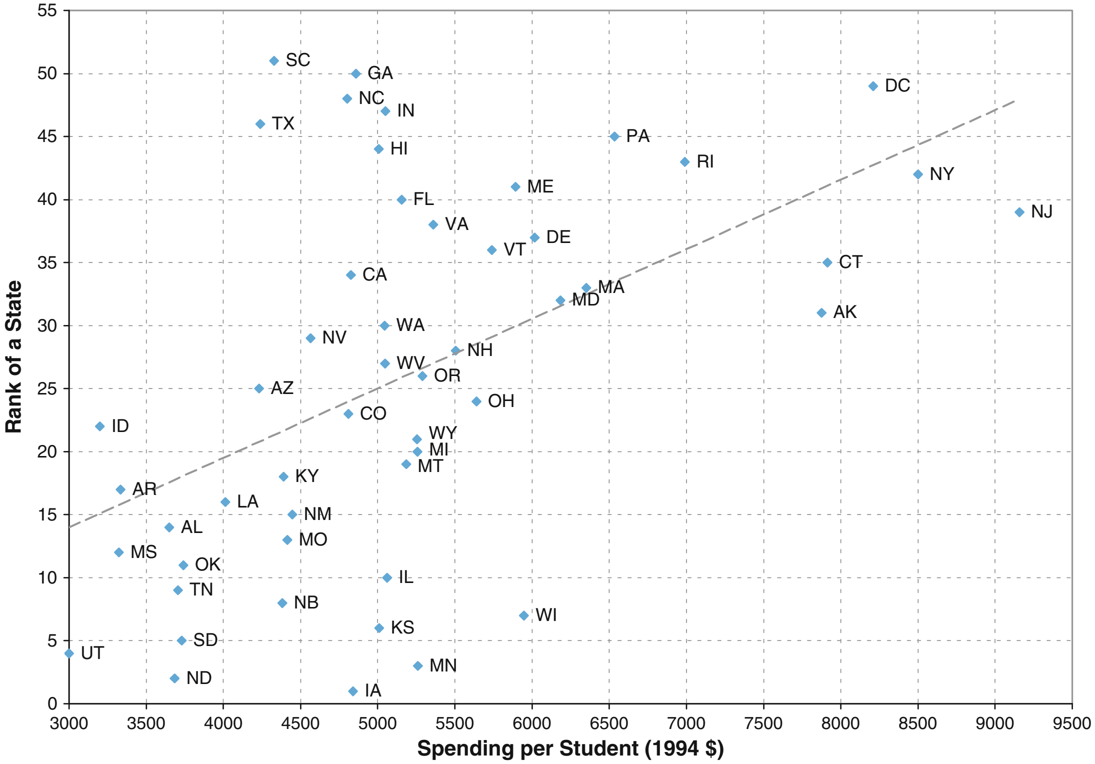

FIGURE 20.1 Rank of a state (in average SAT score) versus its spending per student (circa 1994) and the least-squares regression estimate of their relationship.

The preceding example employed typical "opportunistic," or found, data. But even data generated by a designed experiment need external information. A DoD project from the early days of neural networks attempted to distinguish aerial images of forests with and without tanks in them. Perfect performance was achieved on the training set, and then on an out-of-sample set of data that had been gathered at the same time but not used for training. This was celebrated but, wisely, a confirming study was performed. New images were collected on which the models performed extremely poorly. This drove investigation into the features driving the models and revealed them to be magnitude readings from specific locations of the images; i.e., background pixels. It turns out that the day the tanks had been photographed was sunny, and that for nontanks, cloudy![11] Even resampling the original data wouldn't have protected against this error, as the flaw was inherent in the generating experiment.

[11] PBS featured this project in a 1991 documentary series *The Machine That Changed the World*: Episode IV, "The Thinking Machine."

A second tanks and networks example: A colleague had worked at a San Diego defense contractor, where researchers sought to distinguish tanks and trucks from any aspect angle. Radars and mechanized vehicles are bulky and expensive to move around, so they fixed the radar installation and rotated a tank and a truck on separate large, rectangular platforms. Signals were beamed at different angles, and the returns were extensively processed—using polynomial network models of subsets of principal components of Fourier transforms of the signals—and great accuracy in classification was achieved. However, seeking transparency (not easy for complex, multistage models), the colleague discovered, much to his chagrin, that the source of the key distinguishing features determining vehicle type turned out to be the bushes beside one platform and not another![12] Further, it is suspected that the angle estimation accuracy came from the signal reflecting from the platform corners—not a feature you will encounter in the field. Again, no modeling technology alone could correct for flaws in the data, and it took careful study of how the model worked to discover its weakness.

5. ACCEPT LEAKS FROM THE FUTURE

One of us often evaluates promising investment systems for possible implementation. A Ph.D. consultant, with a couple of books under his belt, had prepared a neural network model for a Chicago bank to forecast interest rate changes. The model was 95% accurate—astonishing given the importance of such rates for much of the economy. The bank board was cautiously ecstatic and sought a second opinion. My colleagues found that a version of the output variable had accidentally been made a candidate input. Thus, the output could be thought of as only losing 5% of its information as it traversed the network.

One investment system we were called in to examine was 70% accurate in forecasting the direction a market index would move the next day. Its developers were quite secretive, but after a great deal of work on behalf of the client considering investing, we eventually duplicated its actions exactly with a simple moving average of 3 days of prices. This simplicity was disappointing, but much worse was that the 3 days were centered on *today*. That is, tomorrow's price was one of the inputs! (They'd have had 100% accuracy if they had have just dropped one of the input variables.) Another trading system, developed with monumental effort over several years and involving the latest research in Genetic Algorithms (GA), focused on commodities. Eventually, it was 99.9% matched by, essentially, two lines of code, which made obvious that its use was impractical. That is, the complex GA devolved to a simple model (the flaws of which then became quite clear), in a manner impossible to discern by examining the extremely complex modeling machinery. In all these cases, the model's author was the chief one deceived.

[12] This excellent practice of trying to break your own work is so hard to do even if you are convinced of its need that managers should perhaps pit teams with opposite reward metrics against one another in order to proof-test solutions.

One trick is to look hardest at any input variable that works too well. For instance, on a cross-sell project—trying to identify clients of an auto club who would be good prospects for a more profitable insurance product—we found a code that was present about 25% of the time but was always associated with insurance purchasers. After extended inquiry (as the meaning of data fields are often lost to the mists of time), we found that the code was the type of insurance cancellation; that is, that it really represented the fact that about a quarter of purchasers canceled their insurance each year. Dorian Pyle, author of a thorough book on *Data Preparation for Data Mining*, has recounted privately that he's encountered problems that required seven such "decapitation" passes, where the best variable turns out to be a leak from the future.

In general, data warehouses are built to hold the best information known to date on each customer; they are not naturally able to pull out what was known at the time that you wish to study. So, when you are storing data for future mining, it's important to date-stamp records and to archive the full collection at regular intervals. Otherwise, re-creating realistic information states will be extremely difficult and will lead to wrong conclusions. For instance, imagine you wished to study whether dot-com companies were, in aggregate, really a bad bet. Using a price-quoting service, you pull down all the histories of current such companies and study their returns. Quite likely, they would have been a great bet, despite the horrible shakeout in that market sector that started in roughly March 2000. Why? Were their early gains so great as to absorb later massive losses? Actually, you would have made a study error—"survivor bias"—by looking back from *current* companies, which is all most data services carry. A re-creation of the set of companies that existed at the earlier time, including the doomed ones, would provide a much more realistic (i.e., negative) result.

6. DISCOUNT PESKY CASES

Outliers and leverage points can greatly affect summary results and cloud general trends. Yet you must not routinely dismiss them; they could *be* the result. The statistician John Aitchison recalled how a spike in radiation levels over the Antarctic was thrown out for years, as an assumed error in measurement, when in fact it revealed a hole in the ozone layer that proved to be an impressive finding. To the degree possible, visualize your data to help decide whether outliers are mistakes to be purged or findings to be explored.

We find the most exciting phrase in research not to be a triumphal (and rare) "Aha!" of discovery, but the muttering of puzzlement, "That's odd . . ." To be surprised, though, you must have expectations. So we urge colleagues to make hypotheses of how results will turn out from their upcoming experiments. After the fact, virtually everything can and will be plausibly interpreted. One master's engineering student at the University of Virginia was working with medical data (often an extremely tough domain) and presented some interim findings as a graph on an unlabeled transparency to the nurse and doctor leading the research. They were happily interpreting the results when he realized, to his horror, that the foil was upside-face-down, that is, that the relationship between the variables was reversed. He sheepishly set it right and in only seconds the medical experts exclaimed, “That makes sense too!” and continued interpreting its (new and completely opposite) nuances.[13]

Humans are, and likely will remain, the best pattern-recognizers in existence—for the low dimensions in which we operate. But we are perhaps too good; we tend to see patterns even when they don’t exist. Two colleagues (and expert data miners, Dustin Hux and Steve Gawtry) worked at the Virginia State Climatology Office when a citizen sent in a videotape of purported cloud phenomena: “Could the weather experts explain the astonishing phenomena?” The un-narrated 3-hour tape contained nothing but typical summer (cumulus humulus) clouds. The citizen had seen “dragons in the clouds,” where there (almost certainly) weren’t any.

A valuable step early in analysis is to seek to validate your data internally: do the variables agree with one another? Finding, as we did on one data set, that “95% of the husbands are male” is useless in itself, but reveals something about the data’s quality, and provides audit questions and flags observations. Reliable analysis depends so strongly on the quality of the data that internal inconsistencies can hobble your work, or they can be clues to problems with the flow of information within the company and reveal a key process obstacle. We worked closely with a direct mail client and dove deeply into the data, looking for relationships between what was known about a potential customer and resulting orders. We actually endangered our appearance of competence to the client by persisting in questioning about unexpectedly low numbers of catalogs being sent to some customers. Eventually, it was found that the “Merge/Purge house” was treating overseas purchasers the opposite of how they were instructed, and erroneously deleting from the mailing lists some of the best prospects. This finding was probably more helpful to the client’s bottom line than most of our high-tech modeling work.[14]

7. EXTRAPOLATE

Modeling “connects the dots” between known cases to build up a plausible estimate of what will happen in related, but unseen, locations in data space. Obviously, models—and especially nonlinear ones—are very unreliable outside the bounds of any known data. (Boundary checks are the very minimum protection against “overanswering,” as discussed in the next section.)

But there are other types of extrapolations that are equally dangerous. We tend to learn too much from our first few experiences with a technique or problem. The hypotheses we form—which our brains are desperate to do to simplify our world—are irrationally hard

[13] Those who don’t regularly research with computers seem to give more credence to their output, we’ve noticed. Perhaps like sausage being enjoyed most by those least familiar with how it’s made.

[14] The analysis work, though, combined with operational changes such as higher-quality catalog paper, did result in a doubling of the client’s average sales per catalog within a year.

to dethrone when conflicting data accumulate. Similarly, it is very difficult to "unlearn" things we've come to believe after an upstream error in our process is discovered. (This is not a problem for our obedient and blindingly fast assistant: the computer. It blissfully forgets everything except what it's presented at the moment.) The only antidote to retaining outdated stereotypes about our data seems to be regular communication with colleagues and clients about our work, to uncover and organize the unconscious hypotheses guiding our explorations.[15]

Extrapolating also from small dimensions, *d*, to large is fraught with danger, as intuition gained on low-*d* is useless, if not counterproductive, in high-*d*. (That is, an idea may make sense on a white board, and not work on a many-columned database.) For instance, take the intuitive *nearest neighbor* algorithm, where the output value of the closest known point is taken as the answer for a new point. In high-*d*, no point is typically actually close to another; that is, the distances are all very similar and, by a univariate scale, not small. "If the space is close, it's empty; it it's not empty; it's not close" is how Scott (1992) describes this aspect of the "curse of dimensionality."

Friedman (1994) illustrates four properties of high-*d* space:

1. Sample sizes yielding the same density increase exponentially with *d*.
2. Radiuses enclosing a given fraction of data are disproportionately large.
3. Almost every point is closer to an edge of the sample space than to even the nearest other point.
4. Almost every point is an outlier in its own projection.[16]

As our most powerful technique—visualization—and our deep intuition about spatial relationships (in low-*d*) are rendered powerless in high-*d*, researchers are forced to employ much more simplistic tools at the early stages of a problem until the key variables can be identified and the dimensions thereby reduced.

The last extrapolation is philosophical. Most researchers in data mining, machine learning, artificial intelligence, etc., hold the theory of evolution as an inspiration, if not motivating faith. The idea that the awesome complexity observed of life might have self-organized through randomization and indirect optimization can bolster the belief that something similar might be accomplished in software (and many orders of magnitude faster). This deep belief can easily survive evidence to the contrary. We have heard many early users of *neural networks*, for instance, justify their belief that their technique will eventually provide the answer since "that's how the brain works."[17] Others have such faith in their mining algorithm that they concentrate only on obtaining all the raw materials that collectively contain the information about a problem and don't focus sufficiently on

[15] This is so critical that, if you don't have a colleague, rent one! A tape-recorder or a dog will even be preferable to keeping all of your dialog internal.

[16] That is, each point, when projecting itself onto the distribution of other points, thinks of itself as weird . . . kind of like junior high.

[17] Though research from even a decade ago argues instead that each human neuron (of which there are billions) is more like a supercomputer than a simple potentiometer.

creating higher-order features of the raw data. They feed, say, the intensity values of each pixel of an image into an algorithm, in hopes of classifying the image—which is almost surely doomed to fail—instead of calculating higher-order features—such as edges, regions of low variance, or matches to templates—which might give the algorithms a chance.

A better mental model of the power and limitations of data mining is small-scale, rather than large-scale, evolution. We can observe, for instance, that one can take a population of mutts and, through selective breeding over several generations, create a specialized breed such as a greyhound. But it is a bold and unproven hypothesis that one could do so, even with infinite time, beginning instead with pond scum. Likewise, the features you extract from raw data strongly impact the success of your model. As a rule, use all the domain knowledge and creativity your team can muster to generate a rich set of candidate data features. Data mining algorithms are strong at sifting through alternative building blocks, but not at coming up with them in the first place.

The March 25, 1996, cover of *Time* magazine, provocatively asks: “Can Machines Think? They already do, say scientists. So what (if anything) is special about the human mind?”[18] Magazine covers can perhaps be forgiven for hyperbole; they’re crafted to sell copies. But inside, someone who should know better (an MIT computer science professor) was quoted as saying, “Of course machines can think. After all, humans are just machines made of meat.” This is an extreme version of the “high-AI (artificial intelligence)” view (or perhaps,

[18] *Time* magazine was reporting on the previous month’s first chess match between Gary Kasparov (often called the best chess player in history) and “Deep Blue,” a specialized IBM chess computer. Kasparov lost the first game—the first time a Grand Master had been beaten by a program—but handily won the full match. Still, that was to be the high-water mark of human chess achievement. A year later, “Deeper Blue” won the re-match, and it’s likely humans will never reign again. (IBM enjoyed the publicity and didn’t risk a requested third match and, of course, computer power has grown by over two orders of magnitude since then.) Unlike checkers, chess is still nearly infinite enough that computers can’t play it perfectly, but they can simply march through a decision tree of possibilities as deep as time allows (routinely to a dozen or more plies, or paired move combinations). Though it seems like a good test of intelligence, the game of chess actually plays well to the strengths of a finite state machine: the world of possibilities is vast, but bounded, the pieces have precise properties, and there is close consensus on many of the game trade-offs (i.e., a bishop is worth about three times as much as an unadvanced pawn). There are also vast libraries of carefully worked special situations, such as openings, and end-game scenarios, where a computer can play precisely and not err from the known best path. The automation component with the greatest uncertainty is the precise trade-off to employ between the multiple objectives—such as attack position (strong forward center?), defense strength (take time to castle?), and the pursuit of materiel (capture that pawn?)—that vie for control of the next move. To define this score function by which to sort the leaf nodes of the decision tree, the Deep Blue team employed supervised learning. They took the best role models available (human Grand Masters) and trained on the choices made by the GMs over many thousands of recorded games to discover what parameter values for the move optimizer would best replicate this “gold standard” collection of choices. Lastly, the designers had the luxury of studying many of Kasparov’s games and purportedly devised special anti-Kasparov moves. (Incidentally, Kasparov was refused the chance to study prior Deep Blue games.) Given how “computational” chess is then, it’s a wonder any human does well against a machine! But our major point is that chess skill is a poor metric for “thinking.”

the "low-human" view). But anyone who's worked hard with computers knows that the analytic strengths of computers and humans are more complementary than alike. Humans are vastly superior at tasks like image recognition and speech understanding, which require context and "common sense" or background knowledge to interpret the data, but computers can operate in vast numbers of dimensions—very simply, but with great precision. It's clear to us that the great promise being fulfilled by data mining is to vastly augment the productivity of—but not to replace—skilled human analysts. To believe otherwise—at the extreme, in an eventual "singularity event" in time where humans and machines will merge to create a type of immortal consciousness—is an extrapolation more like faith than science.

8. ANSWER EVERY INQUIRY

Early in our careers, one of us demonstrated a model estimating rocket thrust that used engine temperature, *T*, as an input. A technical gate-keeper for the potential client suggested we vary some inputs and tell what ensued. "Try $T = 98.6$ degrees." Naively, we complied—making mistake 7, as that was far outside its training bounds. The output (of a nonlinear polynomial network) was ridiculous, as expected, but no amount of calm technical explanation around that nonsurprising (to us) result could erase, in the decision-maker's mind, the negative impact of the breathtaking result that had briefly flashed by. We never heard from that company again. Obviously, a model should answer "don't know" for situations in which its training has no standing!

But how do we know where the model is valid, that is, has enough data close to the query by which to make a useful decision? The simplest approach is to note whether the new point is outside the bounds, on any dimension, of the training data. Yet, especially in high-*d*, the volume of the populated space is only a small fraction of the volume of the rectangle defined by the univariate bounds. With most real data, inputs are very far from mutually independent, so the occupied fraction of space is very small, even in low-*d*. (The data often look like an umbrella packed corner-to-corner diagonally in a box.) A second approach, more difficult and rare, is to calculate the convex hull of the sample—essentially, a "shrink wrap" of the data points. Yet even this does not always work to define the populated space. Figure 20.2 illustrates a 2-*d* problem similar to one we encountered in practice (in higher-*d*) in an aeronautical application. There, fundamental constraints on joint values of physical variables (e.g., height, velocity, pitch, and yaw) caused the data to be far from i.i.d. (independent and identically distributed). We found that even the sample mean of the data, μ, was outside the true region of populated space.

One approach that has helped the few times we've tried it, is to fit a very responsive, nonlinear model to the data, for instance through a polynomial network (Elder and Brown, 2000). High-order polynomials quickly go toward infinity outside the bounds of the training data. If the output estimate resulting from an unbounded, nonlinear (and even overfit) model is beyond the output bounds, then it is very likely the input point is outside the

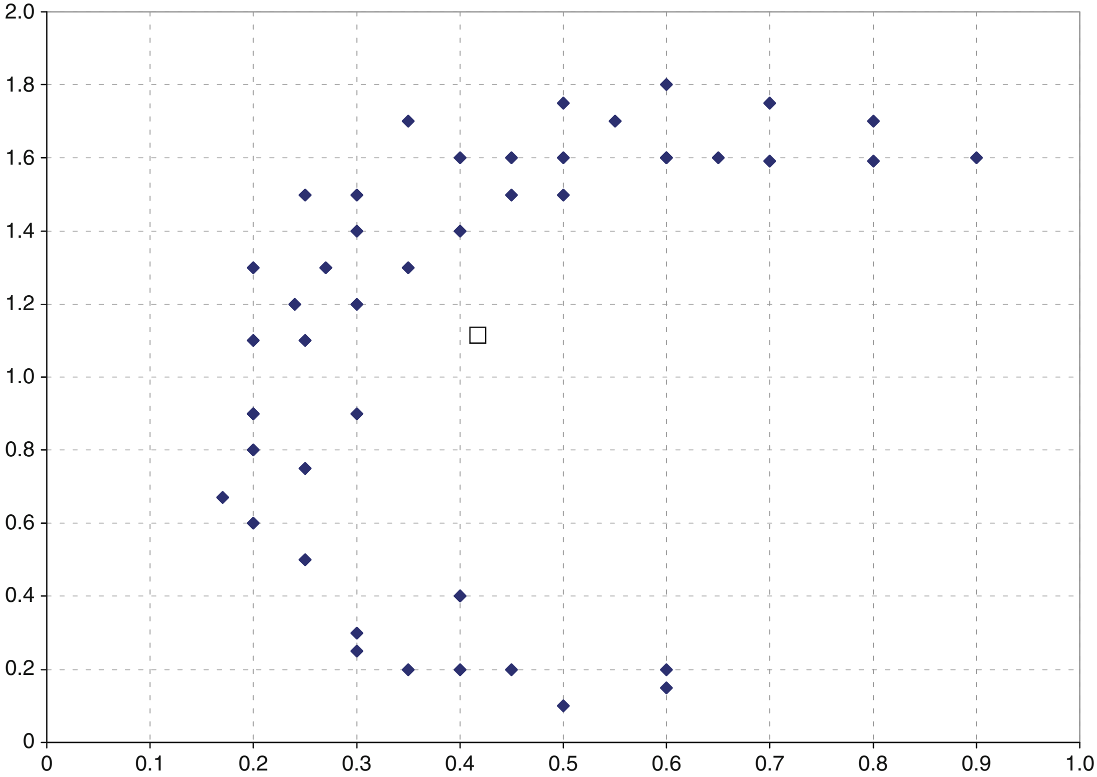

FIGURE 20.2 Sample two-dimensional problem for which the data mean (open box symbol) is outside the bounds of the (crescent-shaped) valid space.

training data. If a training data point had been near that input point, it would have better constrained the model's estimate.

Just as it is essential to know where a model has standing—i.e., in what regions of input space its estimates might be valid—it is also useful to know the uncertainty of estimates. Most techniques provide some measure of spread, such as σ, for the overall accuracy result (e.g., +/− 3% for a political survey), but it is rare indeed to have a conditional standard deviation, $\sigma(x)$, to go with the conditional $\mu(x)$. A great area of research, in our opinion, would be to develop robust methods of estimating certainty for estimates conditioned on where in input space you are inquiring.

One estimation algorithm, Delaunay Triangles, which does depend strongly on $\sigma(x)$ was developed to make optimal use of experimental information for global optimization (Elder, 1993). For experiments where results are expensive to obtain (core samples of soil, for instance), the challenge is to find, as efficiently as possible, the input location with the best result. If several samples and their results are known, you can model the score surface (relationship between input vector and output score) and rapidly ask the model for the best location to next probe (i.e., experimental settings to employ). If that result isn't yet good enough (and budget remains to keep going), its information could be used to update the model for the next probe location. The overall estimation surface consists of piecewise planes, as shown in Figure 20.3, where each region's plane has a quadratic variance "canopy" over it, as in Figure 20.4, revealing how the uncertainty of the estimation grows as you depart from the known points (the corners).[19] This approach worked very well, for low (fewer than about 10) dimensions, and the resulting multimodal search

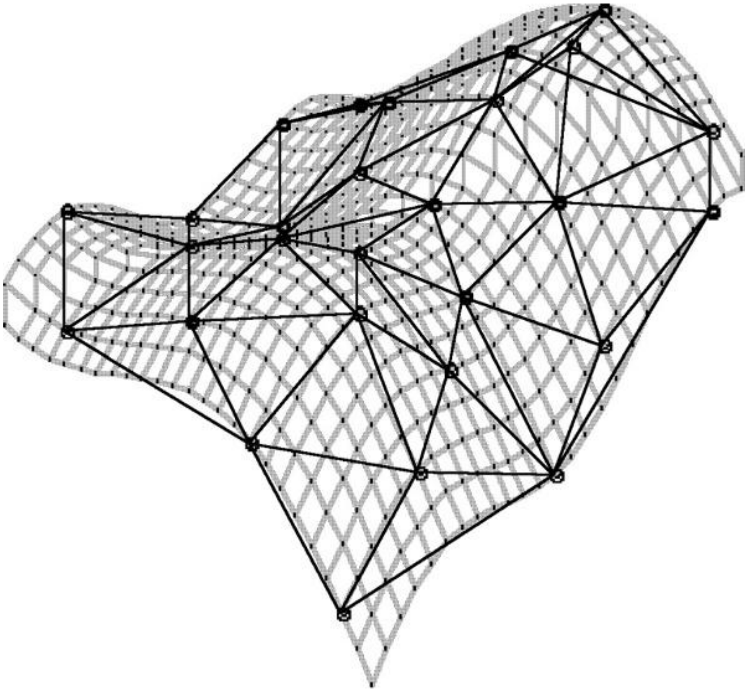

FIGURE 20.3 Estimation surface of Delaunay Triangle method (Elder, 1993) is piecewise planar. (The underlying functional surface is represented here by a mesh.)

[19] The modeling technique developed for GROPE was driven by the special requirements of optimizing an unknown function—especially that the response surface model had to agree exactly with the known samples. If you assume the least about the response surface—that there is Brownian motion (or a random walk) between the known points—then the ideal estimator turns out to be a plane. So, $\mu(x)$ is a piecewise planar collection of simplices (e.g., triangles when there are two input dimensions). The tiling or tessellation of the input space is done in such a way as to create the most uniform simplices (those with the greatest minimum angle), which is performed by Delaunay triangulation (a dual of nearest neighbor mapping). The key, though, was to pair this with an estimate of the standard deviation of $\mu(x)$, conditioned on x, $\sigma(x)$. (The Brownian motion assumption drives this to be the square root of a quadratic function of distance from the known corners.) Now, with both parts, $\mu(x)$ and $\sigma(x)$, you can rapidly calculate the location, x, where the probability of exceeding your result goal is the greatest. So, the model would suggest a probe location, you would perform the experiment, and the result would update the model, with greater clarity on the mean estimates (piecewise planes) and reduced variance (piecewise quadratic "bubbles" over each plane) with each iteration.

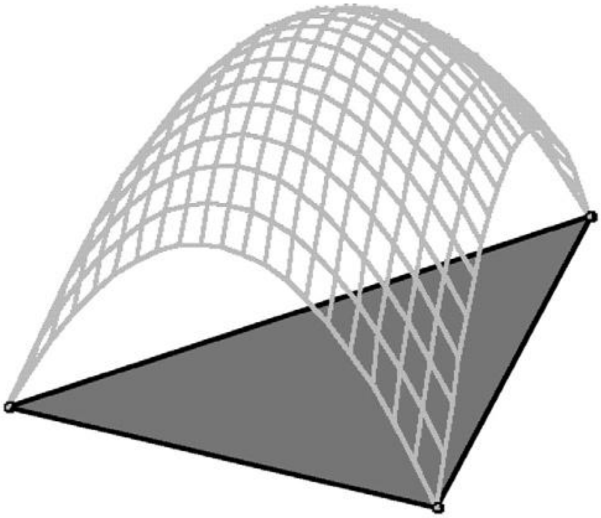

FIGURE 20.4 Each simplex (e.g., triangle in two dimensions) of the Delaunay method (Elder, 1993) pairs a planar estimation of $\mu(x)$ with a quadratic estimation of $\sigma^2(x)$.

algorithm, GROPE (Global R^d Optimization when Probes are Expensive), took the fewest probes of all then-existing algorithms to converge close to the answer on a standard suite of test problems. By having, for every location, x, an estimate of the mean, $u(x)$, along with its uncertainty, $\sigma(x)$, the algorithm could, with every new result, refine its estimates and reduce its uncertainty, and thereby zero in on the locations with the greatest potential.

9. SAMPLE CASUALLY

The interesting cases for many data mining problems are rare, and the analytic challenge is akin to "finding needles in a haystack." However, many algorithms don't perform well in practice, if the ratio of hay to needles is greater than about 10 to 1. To obtain a near-enough balance, you must either *undersample*—removing most common cases—or *oversample*—duplicating rare cases. Yet it is a mistake to do either casually.

A direct marketing firm in Maryland had too many (99%) nonresponders (NR) for a decision tree model to properly predict who would give to a client charity. To better balance the data, the firm kept all responders and every tenth NR from the data set of over a million cases until it had exactly 100K cases. The firm's decision tree model then predicted that everyone in certain Alaskan cities (such as Ketchikan, Wrangell, and Ward Cove) would respond. And, it was right, on the training data. What had happened? It turns out the original data had been sorted by ZIP code, and since it was over 1M cases, the decimating (sampling every tenth) had stopped (at 100K) before reaching the end. Thus, only responders (all of whom had been taken at first) were sampled from the bottom of the file, where the highest (Alaskan) ZIP codes were.

Clearly, a good strategy is to "shake before baking," that is, to randomize the order of a file before sampling. Actually, first record the original case number, so you can reverse the process, but also because there may be useful information in the original order. One credit scoring problem we tackled, for instance, had very useful information in the case number. Perhaps the first cases in the file were the first to respond (most desperate for credit?) or were the top candidates to be given offers—in any case, the file position was a very significant predictor of creditworthiness. Unfortunately, no one could remember how the file was formed, so we lost the opportunity to use that clue to focus on related features that would improve the edge of the model.

Since case number is not always random, we recommend appending some truly random variables to the data to act as "canaries in the mine"[20] to indicate when your variable selection has pushed too far. If the model starts using those variables, you know that overfit is very likely. (Be sure to label them *random1*, *random2*, etc., instead of *R1*, *R2*, or someone will call you months later and ask where he or she might find the *R1* values, needed for the "improved" model!)

It is possible to make mistakes even more readily with up-sampling. Again, on a credit scoring problem, we had too few known defaulters to properly train several algorithms. (Some work well with case weights and variable misclassification costs, but most don't, and we wanted to make sure their training data were exactly equivalent for comparison.) We ran several experiments with cross-validation and many modeling cycles. Oddly, for several experiments, it was much harder than anticipated to obtain overfit; that is, results tended to improve with model complexity far beyond where we expected. Eventually, we noticed that when we separated the performance on new data of the rare cases of default from the common cases of creditworthiness, performance on the former (rare cases) kept improving with complexity, while the latter curve turned upward (higher error) as expected. It turns out that we had duplicated the defaults *before* splitting the data into cross-validation sets. This meant that copies of each rare case appeared in most of the data subsets, and thus no rare case was truly ever out-of-sample. The lesson learned is to split into sets first and then up-sample the rare cases in training only.

Note that you may need to define the unit of sampling at a higher granularity than the case. That is, some cases may need to stay bundled together and never be separated across data subsets. For instance, with medical data, each case often represents a doctor visit. In predicting outcomes, you must almost certainly keep all the records from one patient together and not use some for training and some for evaluation.

Lastly, remember that a stratified sample will almost always save you from trouble. Carefully consider which variables need to be represented in each data subset and sample them separately; e.g., take 10% of the red and 10% of the blue cases for each sample, instead of just a simple 10% of the whole. Good stratification is how fairly accurate political projections, for instance, can be made with very few (~1,000) interviews.

[20] Canaries are particularly susceptible to carbon monoxide and methane, so if the caged canary stopped singing and keeled over, it was time to evacuate quickly!

10. BELIEVE THE BEST MODEL

As George Box said, "All models are wrong, but some are useful." Unfortunately, we seem to have a great need to believe in our models. We want them to reveal deep underlying truth, rather than to just be useful leading indicators.[21] However, reading too much into models may do more harm than good. Usually, too much attention is paid to the particular variables used by the "best" data mining model—which likely barely won out over hundreds of others of the millions (to billions) tried, using a score function only approximating your goals, and on finite data scarcely representing the underlying data-generating mechanism. It is better to build several models and interpret the resulting distribution of variables rather than accept the set chosen by the single best model.[22]

Usually, many very similar variables are available, and the particular structure and variables selected by the best model can vary chaotically, that is, change greatly due to very small changes in the input data or algorithm settings. A decision tree, for instance, can change its root node due to one case value change, and that difference cascades through the rest of the tree. One polynomial network we built two decades ago changed drastically in appearance when only 999 of the 1,000 cases sent with it were used to test it. This structural variability is troubling to most researchers. But that is perhaps due to their reading too much into the "best" model. In cases we have examined, the functional similarity of competing models—that is, their vector of estimate values—is often much more similar than their structural form. That is, competing models often *look* more different than they *act*, and it's the latter we believe that matters.

Lastly, as argued in Chapter 13, the best model is likely to be an ensemble of competing, distinct, and individually good models. In Chapter 18, Figure 18.1 showed the relative performance of five algorithms on six test problems. Figure 18.2 went on to reveal that any of the four different ways of ensembling those models greatly reduced the error on out-of-sample data compared to the individual models.

Bundling models reduces the clarity of a model's details; yet, as Leo Breiman argued, only somewhat tongue-in-cheek:

$$\text{Interpretability} * \text{Accuracy} < \beta \ (\text{Breiman's constant})$$

meaning increased interpretability of a model comes, inevitably, at the cost of reduced accuracy. Some problems do require strict interpretability. For instance, insurance models have to be approved by state authorities. Also, credit applicants have to, by law, be able to be told the top five factors under their influence that hurt their credit score. But for the applications we're most familiar with, such as investment prediction or fraud detection, there is

[21] The ancient Egyptians believed the dog star/god, Sirius, was responsible for the seasonal flooding of the Nile, so essential to their survival. While not causal, the star's rise was a useful leading indicator, accurate enough for important decision making.

[22] In a similar vein, Box also said, "Statisticians are like artists; they fall in love with their models." To be great data miners, we need to work as hard to break our models as we did to build them. The real world will certainly break them if we don't!

a huge return to small but significant increases in accuracy, and we are content to worry about interpretation only after the model has proven it is worthy of the attention.

HOW SHALL WE THEN SUCCEED?

Fancier modern tools and harder analytic challenges mean we can now "shoot ourselves in the foot" with greater accuracy and power than ever before! Success is improved by learning, which best comes from experience, which seems to be most memorable due to mistakes. So go out and make mistakes early in your career!

We've found a useful PATH to success to be

- **P**ersistence: Attack a data mining problem repeatedly, from different angles. Automate the essential steps, especially so you can perform resampling tests. Externally check your work. A great idea is to hire someone to break your model, since you often won't have the heart.
- **A**ttitude: An optimistic, "can-do" attitude can work wonders for results, especially in a team setting.
- **T**eamwork: Business and statistical experts must cooperate closely to make the best progress. Does everyone want the project to succeed? Sometimes, passive-aggressive partners, such as the purported providers of data, can secretly see only danger in a project delving into their domain, so be sure that each partner can advance his or her career through the project's success.
- **H**umility: Learning from others requires vulnerability. When we data miners visit a client, we know the least about the subject at hand. However, we do know a lot about analysis and the mistakes thereof. Also be humble about the powers of technology. As shown in this chapter especially, data mining is no "silver bullet." It still requires a human expert who can step back, see the big picture, and ask the right questions. We think of data mining as something of an "Iron Man" suit, tremendously augmenting, for good or ill, the powers of the wearer.

POSTSCRIPT

So go out and do well, while doing good, with these powerful modern tools!

References

Elder, J. F., IV (1993). *Efficient optimization through response surface modeling: A GROPE algorithm*. Dissertation. Charlottesville: School of Engineering and Applied Science, University of Virginia.

Elder, J. F., IV (1996). A review of *Machine Learning, Neural and Statistical Classification* (Michie, Spiegelhalter, & Taylor, Eds., 1994). *Journal of the American Statistical Association*, *91*(433), 436–437.

Elder, J. F., IV & Lee, S. S. (1997). *Bundling Heterogeneous Classifiers with Advisor Perceptrons*. University of Idaho Technical Report, October, 14.

Elder, J. F., IV, & Brown, D. (2000). Induction and Polynomial Networks. In M. Fraser (Ed.), *Network Models for Control and Processing* (Chapter 6). Bristol, UK: Intellect Press.

Friedman, J. H. (1994). An Overview of Computational Learning and Function Approximation. In V. Cherkassky, J. H. Friedman, & H. Wechsler (Eds.), *From Statistics to Neural Networks: Theory and Pattern Recognition Applications* (Chapter 1). New York: Springer-Verlag.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*. New York: Ellis Horwood.

Scott, D. W. (1992). *Multivariate Density Estimation: Theory, Practice, and Visualization*. New York: John Wiley.