CHAPTER

17

Preparing Data for Mining

As a translucent amber fluid, gasoline—the power behind the transportation industry—barely resembles the gooey black ooze pumped up through oil wells. The difference between the two liquids is the result of multiple steps of refinement that distill useful products from the raw material.

Data preparation is a very similar process. The raw material comes from operational systems that have often accumulated crud, in the form of eccentric business rules and layers of system enhancements and fixes, over the course of time. Fields in the data are used for multiple purposes. Values become obsolete. Errors are fixed on an ongoing basis, so interpretations change over time. The process of preparing data is like the process of refining oil. Valuable stuff lurks inside the goo of operational data. Half the battle is refinement. The other half is converting its energy to a useful form—the equivalent of running an engine on gasoline.

The proliferation of data is a feature of modern business. Our challenge is to make sense of the data, to refine the data so that the engines of data mining can extract value. One of the challenges is the sheer volume of data. A customer may call the call center several times a year, pay a bill once a month, turn the phone on once a day, make and receive phone calls several times a day. Over the course of time, hundreds of thousands or millions of customers are generating hundreds of millions of records of their behavior. Even on today's computers, this is a lot of data processing. Fortunately, computer systems have become powerful enough that the problem is really one of having an adequate

budget for buying hardware and software; technically, processing such vast quantities of data is possible.

Data comes in many forms, from many systems, and in many different types. Data is always dirty, incomplete, sometimes incomprehensible and incompatible. This is, alas, the real world. And yet, data is the raw material for data mining. Oil starts out as a thick tarry substance, mixed with impurities. It is only by going through various stages of refinement that the raw material becomes usable—whether as clear gasoline, plastic, or fertilizer. Just as the most powerful engines cannot use crude oil as a fuel, the most powerful algorithms (the engines of data mining) are unlikely to find interesting patterns in unprepared data.

After more than a century of experimentation, the steps of refining oil are quite well understood—better understood than the processes of preparing data. This chapter illustrates some guidelines and principles that, based on experience, should make the process more effective. It starts with a discussion of what data should look like once it has been prepared, describing the customer signature. It then dives into what data actually looks like, in terms of data types and column roles. Since a major part of successful data mining is in the derived variables, ideas for these are presented in some detail. The chapter ends with a look at some of the difficulties presented by dirty data and missing values, and the computational challenge of working with large volumes of commercial data.

What Data Should Look Like

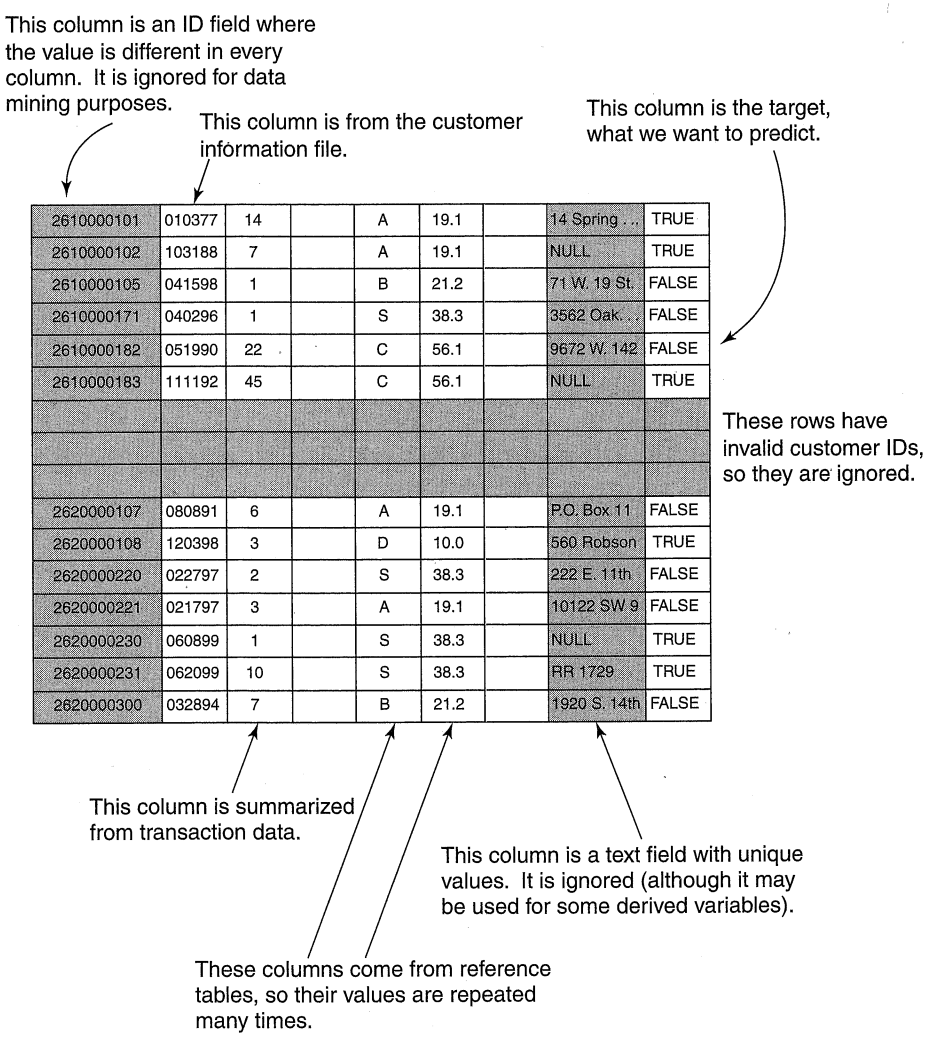
The place to start the discussion on data is at the end: what the data should look like. All data mining algorithms want their inputs in tabular form—the rows and columns so common in spreadsheets and databases. Unlike spreadsheets, though, each column must mean the same thing for all the rows.

Some algorithms need their data in a particular format. For instance, market basket analysis (discussed in Chapter 9) usually looks at only the products purchased at any given time. Also, link analysis (see Chapter 10) needs references between records in order to connect them. However, most algorithms, and especially decision trees, neural networks, clustering, and statistical regression, are looking for data in a particular format called the *customer signature.*

The Customer Signature

The customer signature is a snapshot of customer behavior that captures both current attributes of the customers and changes in behavior over time. Like

a signature on a check, each customer's signature is theoretically unique—capturing the unique characteristics of the individual. Unlike a signature on a check, though, the customer signature is used for analysis and not identification; in fact, often customer signatures have no more identifying information than a string of seemingly random digits representing a household, individual, or account number. Figure 17.1 shows that a customer signature is simply a row of data that represents the customer and whatever might be useful for data mining.



2610000101	010377	14		A	19.1		14 Spring . ..	TRUE
2610000102	103188	7		A	19.1		NULL	TRUE
2610000105	041598	1		B	21.2		71 W. 19 St.	FALSE
2610000171	040296	1		S	38.3		3562 Oak. . .	FALSE
2610000182	051990	22		C	56.1		9672 W. 142	FALSE
2610000183	111192	45		C	56.1		NULL	TRUE
2620000107	080891	6		A	19.1		P.O. Box 11	FALSE
2620000108	120398	3		D	10.0		560 Robson	TRUE
2620000220	022797	2		S	38.3		222 E. 11th	FALSE
2620000221	021797	3		A	19.1		10122 SW 9	FALSE
2620000230	060899	1		S	38.3		NULL	TRUE
2620000231	062099	10		S	38.3		RR 1729	TRUE
2620000300	032894	7		B	21.2		1920 S. 14th	FALSE

Figure 17.1 Each row in the customer signature represents one customer (the unit of data mining) with fields describing that customer.

It is perhaps unfortunate that there is no big database sitting around with up-to-date customer signatures, ready for all modeling applications. Such a system might at first sight seem very useful. However, the lack of such a system is an opportunity because modeling efforts require understanding data. No single customer signature works for all modeling efforts, although some customer signatures work well for several applications

The "customer" in customer signature is the unit of data mining. This book focuses primarily on customers, so the unit of data mining is typically an account, an individual, or a household. There are other possibilities. Chapter 11 has a case study on clustering towns—because that was the level of action for developing editorial zones for a newspaper. Acquisition modeling often takes place at the geographic level, census block groups or zip codes. And applications outside customer relationship management are even more disparate. *Mastering Data Mining*, for instance, has a case study where the signatures are press runs in plants that print magazines.

The Columns

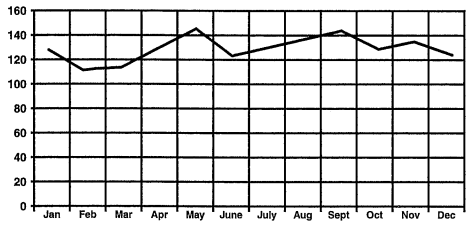
The columns in the data contain values that describe aspects of the customer. In some cases, the columns come directly from existing business systems; more often, the columns are the result of some calculation—so called *derived variables*.

Each column contains values. The *range* refers to the set of allowable values for that column. Table 17.1 shows range characteristics for typical types of data used for data mining.

Table 17.1 Range Characteristics for Typical Types of Data Used for Data Mining

VARIABLE TYPE	TYPICAL RANGE CHARACTERISTICS
Categorical variables	List of acceptable values
Numeric	Minimum and maximum values
Dates	Earliest and latest dates, often latest date is less than or equal to current date
Monetary amounts	Greater than or equal to 0
Durations	Greater than or equal to 0 (or perhaps strictly greater than 0)
Binned or quantiled values	The number of quantiles
Counts	Greater than or equal to 0 (or perhaps greater than or equal to 1)

Histograms, such as those in Figure 17.2, shows how often each value or range of values occurs in some set of data. The vertical axis is a count of records, and the horizontal axis is the values in the column. The shape of this histogram shows the *distribution* of the values (strictly speaking, in a distribution, the counts are divided by the total number of records so the area under the curve is one). If we are working with a sample, and the sample is randomly chosen, then the distribution of values in the subset should be about the same as the distribution in the original data.

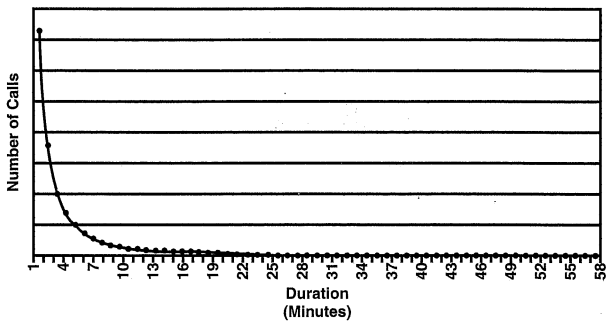


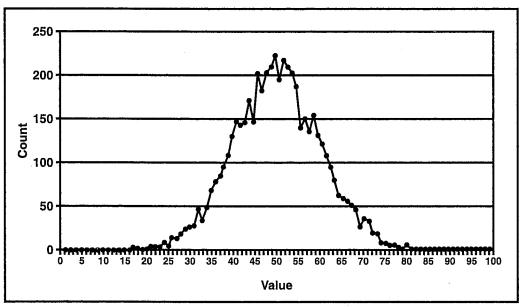
This histogram is for the month of claim for a set of insurance claims.

This is an example of a typically uniform distribution. That is, the number of claims is roughly the same for each month.

This histogram shows the number of telephone calls made for different durations.

This is an example of an exponentially decreasing distribution.





This histogram shows a normal distribution with a mean of 50 and a standard deviation of 10. Notice that high and low values are very rare.

Figure 17.2 Histograms show the distribution of data values.

The distribution of the values provides important insights into the data. It shows which values are common and which are less common. Just looking at the distribution of values brings up questions—such as why an amount is negative or why some categorical values are not present. Although statisticians tend to be more concerned with distributions than data miners, it is still important to look at variable values. Here, we illustrate some special cases of distributions that are important for data mining purposes, as well as the special case of variables synonymous with the target.

Columns with One Value

The most degenerate distribution is a column that has only one value. Unary-valued columns, as they are more formally known, do not contain any information that helps to distinguish between different rows. Because they lack any information content, they should be ignored for data mining purposes.

Having only one value is sometimes a property of the data. It is not uncommon, for instance, for a database to have fields defined in the database that are not yet populated. The fields are only placeholders for future values, so all the values are uniformly something such as "null" or "no" or "0."

Before throwing out unary variables, check that NULLs are being counted as values. Appended demographic variables sometimes have only a single value or NULL when the value is not known. For instance, if the data provider knows that someone is interested in golf—say because the person subscribes to a golfing magazine or belongs to a country club—then the "golf-enthusiast" flag would be set to "Y."When there is no evidence, many providers set the flag to NULL—meaning unknown—rather than "N."

TIP When a variable has only one value, be sure (1) that NULL is being included in the count of the number of values and (2) that other values were not inadvertently left out when selecting rows.

Unary-valued columns also arise when the data mining effort is focused on a subset of customers, and the field used to filter the records is retained in the resulting table. The fields that define this subset may all contain the same value. If we are building a model to predict the loss-ratio (an insurance measure) for automobile customers in New Jersey, then the state field will always have "NJ" filled in. This field has no information content for the sample being used, so it should be ignored for modeling purposes.

Columns with Almost Only One Value

In "almost-unary" columns, almost all the records have the same value for that column. There may be a few outliers, but there are very few. For example, retail

data may summarize all the purchases made by each customer in each department. Very few customers may make a purchase from the automotive department of a grocery store or the tobacco department of a department store. So, almost all customers will have a $0 for total purchases from these departments.

Purchased data often comes in an "almost-unary" format, as well. Fields such as "people who collect porcelain dolls" or "amount spent on greens fees" will have a null or $0 value for all but very few people. Or, some data, such as survey data, is only available for a very small subset of the customers. These are all extreme examples of data skew, shown in Figure 17.3.

The big question with "almost-unary" columns is, "When can they be ignored?" To justify ignoring them, the values must have two characteristics. First, almost all the records must have the same value. Second, there must be so few records with a different value, that they constitute a negligible portion of the data.

What is a negligible portion of the data? It is a group so small that even if the data mining algorithms identified it perfectly, the group would be too small to be significant.

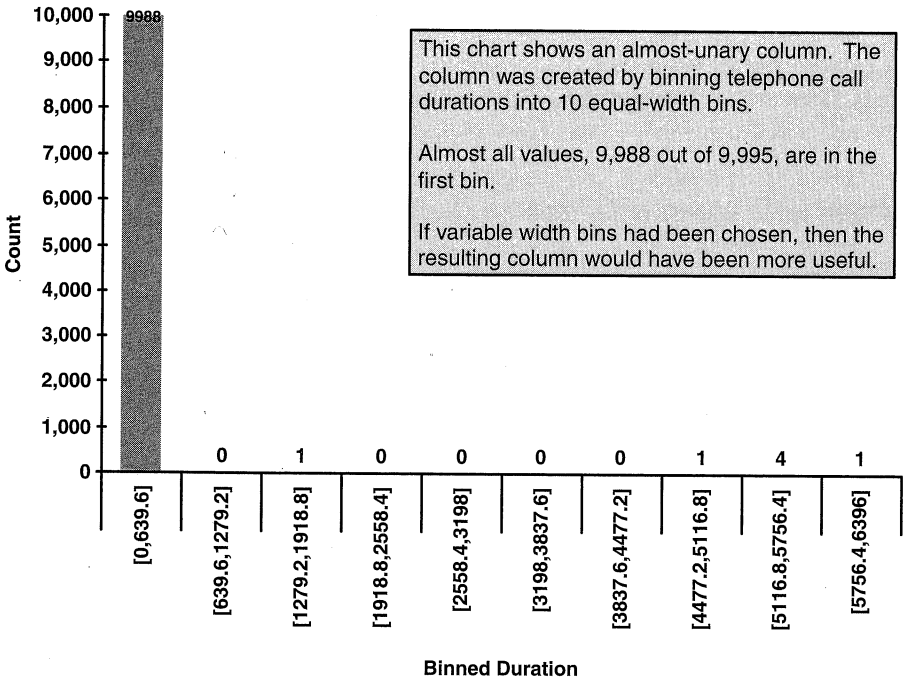


Figure 17.3 An almost-unary field, such as the bins produced by equal-width bins in this case, is useless for data mining purposes

Before ignoring a column, though, it is important to understand why the values are so heavily skewed. What does this column tell us about the business? Perhaps few people ever buy automotive products because only a handful of the stores in question even sell them. Identifying customers as "automotive-product-buyers," in this case, may not be useful.

In other cases, an event might be rare for other reasons. The number of people who cancel their telephone service on any given day is negligible, but over time the numbers accumulate. So the cancellations need to be accumulated over a longer time period, such as a month, quarter, or year. Or, the number of people who collect porcelain dolls may be very rare in itself, but when combined with other fields, this might suggest an important segment of collectors.

The rule of thumb is that, even if a column proves to be very informative, it is unlikely to be useful for data mining if it is almost-unary. That is, fully understanding the rows with different values does not yield actionable results. As a general rule of thumb, if 95 to 99 percent of the values in the column are identical, the column—in isolation—is likely to be useless without some work. For instance, if the column in question represents the target variable for a model, then stratified sampling can create a sample where the rare values are more highly populated. Another approach is to combine several such columns for creating derived variables that might prove to be valuable. As an example, some census fields are sparsely populated, such as those for particular occupations. However, combining some of these fields into a single field—such as "high status occupation"—can prove useful for modeling purposes.

Columns with Unique Values

At the other extreme are categorical columns that take on a different value for every single row—or almost every row. These columns identify each customer uniquely (or close enough), for example:

- Customer name
- Address
- Telephone number
- Customer ID
- Vehicle identification number

These columns are also not very helpful. Why? They do not have predictive value, because they uniquely identify each row. Such variables cause overfitting.

One caveat—which will be investigated later in this chapter. Sometimes these columns contain a wealth of information. Lurking inside telephone numbers and addresses is important geographical information. Customers' first names give an indication of gender. Customer numbers may be sequentially assigned, telling us which customers are more recent—and hence show up as important

variables in decision trees. These are cases where the important features (such as geography and customer recency) should be extracted from the fields as derived variables. However, data mining algorithms are not yet powerful enough to extract such information from values; data miners need to do the extraction.

Columns Correlated with Target

When a column is too highly correlated with the target column, it can mean that the column is just a synonym. Here are two examples:

- "Account number is NULL" may be synonymous with failure to respond to a marketing campaign. Only responders opened accounts and were assigned account numbers.
- "Date of churn is not NULL" is synonymous with having churned.

Another danger is that the column reflects previous business practices. For instance, the data may show that all customers with call forwarding also have call waiting. This is a result of product bundling; call forwarding is sold in a product bundle that always includes call waiting. Or the data may show that almost all customers reside in the wealthiest areas, because this where customer acquisition campaigns in the past were targeted. This illustrates that data miners need to know historical business practices. Columns synonymous with the targets should be ignored.

TIP **An easy way to find columns synonymous with the target is to build decision trees. The decision tree will choose one synonymous variable, which can then be ignored. If the decision tree tool lets you see alternative splits, then all such variables can be found at once.**

Model Roles in Modeling

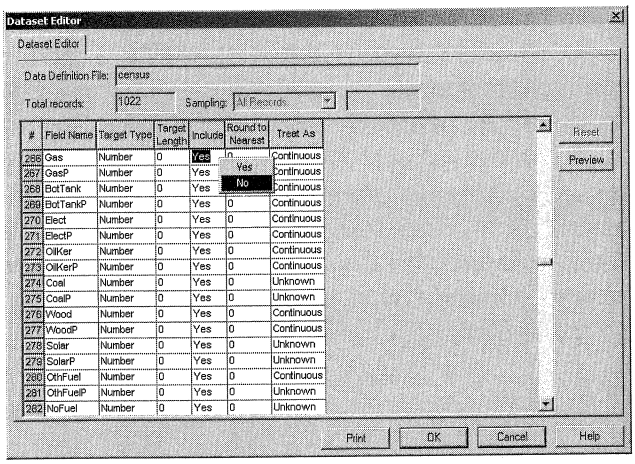
Columns contain data with data types. In addition, columns have roles with respect to the data mining algorithms. Three important roles are:

Input columns. These are columns that are used as input into the model.

Target column(s). This column or set of columns is only used when building predictive models. These are what is interesting, such as propensity to buy a particular product, likelihood to respond to an offer, or probability of remaining a customer. When building undirected models, there does not need to be a target.

Ignored columns. These are columns that are not used.

Different tools have different names for these roles. Figure 17.4 shows how a column is removed from consideration in Angoss Knowledge Studio.



#	Field Name	Target Type	Target Length	Include	Round to Nearest	Treat As
266	Gas	Number	0	Yes	In	Continuous
267	GasP	Number	0	Yes	Yes	Continuous
268	BotTank	Number	0	Yes	No	Continuous
269	BotTankP	Number	0	Yes	0	Continuous
270	Elect	Number	0	Yes	0	Continuous
271	ElectP	Number	0	Yes	0	Continuous
272	OilKer	Number	0	Yes	0	Continuous
273	OilKerP	Number	0	Yes	0	Continuous
274	Coal	Number	0	Yes	0	Unknown
275	CoalP	Number	0	Yes	0	Unknown
276	Wood	Number	0	Yes	0	Continuous
277	WoodP	Number	0	Yes	0	Continuous
278	Solar	Number	0	Yes	0	Unknown
279	SolarP	Number	0	Yes	0	Unknown
280	OthFuel	Number	0	Yes	0	Continuous
281	OthFuelP	Number	0	Yes	0	Unknown
282	NoFuel	Number	0	Yes	0	Unknown

Figure 17.4 Angoss Knowledge Studio supports several model roles, such as ignoring a column when building a model.

TIP **Ignored columns play a very important role in clustering. Since ignored columns are not used to build the clusters, their distribution in the clusters can be very informative. By ignoring columns such as customer profitability or response flags, we can see how these "ignored" columns are distributed in the clusters. And we might just discover something very interesting about customer profit or responders.**

There are some more advanced roles as well, which are used under specific circumstances. Figure 17.5 shows the many model roles available in SAS Enterprise Miner. These model roles include:

Identification column. These are columns that uniquely identify each row. In general, these columns are ignored for data mining purposes, but are important for scoring.

Weight column. This is a column that specifies a "weight" to be applied to each row. This is a way of creating a weighted sample by including the weight in the data.

Cost column. The cost column specifies a cost associated with a row. For instance, if we are building a customer retention model, then the "cost" might include an estimate of each customer's value. Some tools can use this information to optimize the models that they are building.

The additional model roles available in the tool are specific to SAS Enterprise Miners.

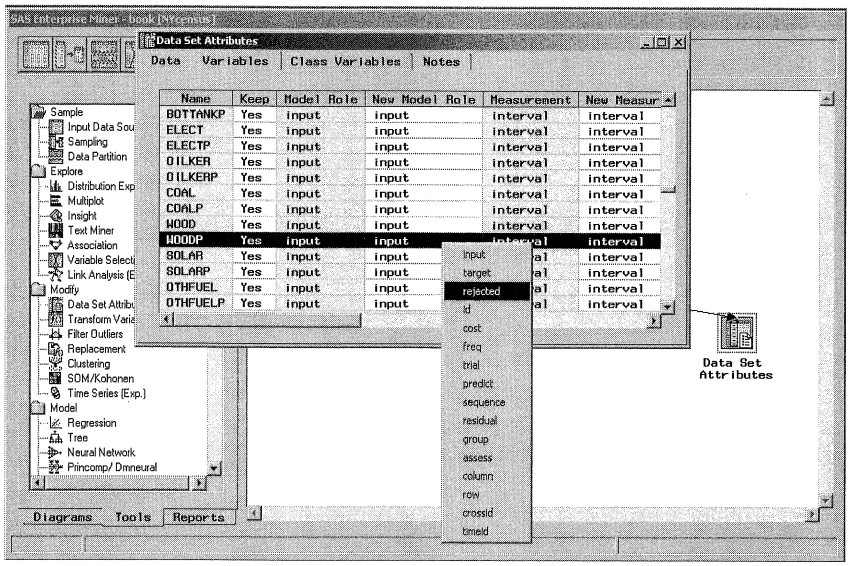


Figure 17.5 SAS Enterprise Miner has a wide range of available model roles.

Variable Measures

Variables appear in data and have some important properties. Although databases are concerned with the type of variables (and we'll return to this topic in a moment), data mining is concerned with the *measure* of variables. It is the measure that determines how the algorithms treat the values. The following measures are important for data mining:

- *Categorical* variables can be compared for equality but there is no meaningful ordering. For example, state abbreviations are categorical. The fact that Alabama is next to Alaska alphabetically does not mean that they are closer to each other than Alabama and Tennessee, which share a geographic border but appear much further apart alphabetically.
- *Ordered* variables can be compared with equality and with greater than and less than. Classroom grades, which range from A to F, are an example of ordered values.
- *Interval* variables are ordered and support the operation of subtraction (although not necessarily any other mathematical operation such as addition and multiplication). Dates and temperatures are examples of intervals.

- *True numeric* variables are interval variables that support addition and other mathematical operations. Monetary amounts and customer tenure (measured in days) are examples of numeric variables.

The difference between true numerics and intervals is subtle. However, data mining algorithms treat both of these the same way. Also, note that these measures form a hierarchy. Any ordered variable is also categorical, any interval is also categorical, and any numeric is also interval.

There is a difference between measure and data type. A numeric variable, for instance, might represent a coding scheme—say for account status or even for state abbreviations. Although the values look like numbers, they are really categorical. Zip codes are a common example of this phenomenon.

Some algorithms expect variables to be of a certain measure. Statistical regression and neural networks, for instance, expect their inputs to be numeric. So, if a zip code field is included and stored as a number, then the algorithms treat its values as numeric, generally not a good approach. Decision trees, on the other hand, treat all their inputs as categorical or ordered, even when they are numbers.

Measure is one important property. In practice, variables have associated types in databases and file layouts. The following sections talk about data types and measures in more detail.

Numbers

Numbers usually represent quantities and are good variables for modeling purposes. Numeric quantities have both an ordering (which is used by decision trees) and an ability to perform arithmetic (used by other algorithms such as clustering and neural networks). Sometimes, what looks like a number really represents a code or an ID. In such cases, it is better to treat the number as a categorical value (discussed in the next two sections), since the ordering and arithmetic properties of the numbers may mislead data mining algorithms attempting to find patterns.

There are many different ways to transform numeric quantities. Figure 17.6 illustrates several common methods:

Normalization. The resulting values are made to fall within a certain range, for example, by subtracting the minimum value and dividing by the range. This process does not change the form of the distribution of the values. Normalization can be useful when using techniques that perform mathematical operations such as multiplication directly on the values, such as neural networks and K-means clustering. Decision trees are unaffected by normalization, since the normalization does not change the order of the values.

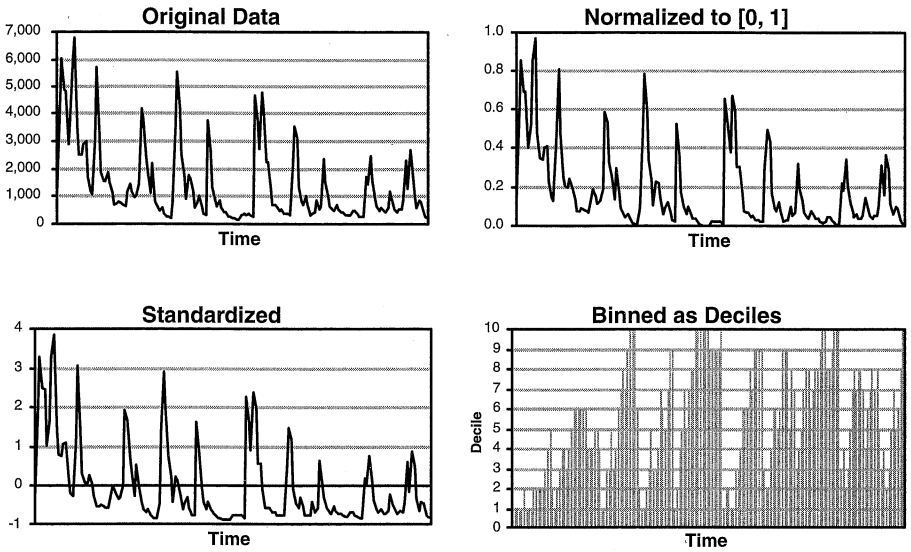


Figure 17.6 Normalization, standardization, and binning are typical ways to transform a numeric variable.

Standardization. This transforms the values into the number of standard deviations from the mean, which gives a good sense of how unexpected the value is. The arithmetic is easy—subtract the average value and divide by the standard deviation. These standardized values are also called *z-scores*. As with normalization, standardization does not affect the ordering, so it has no effect on decision trees.

Equal-width binning. This transforms the variables into ranges that are fixed in width. The resulting variable has roughly the same distribution as the original variable. However, binning values affects all data mining algorithms.

Equal-height binning. This transforms the variables into n-tiles (such as quintiles or deciles) so that the same number of records falls into each bin. The resulting variable has a uniform distribution.

Perhaps unexpectedly, binning values can improve the performance of data mining algorithms. In the case of neural networks, binning is one of several ways of reducing the influence of outliers, because all outliers are grouped together into the same bin. In the case of decision trees, binned variables may result in child nodes having more equal sizes at high levels of the tree (that is, instead of one child getting 5 percent of the records and the other 95 percent, with the corresponding binned variable one might get 20 percent and the other 80 percent). Although the split on the binned variables is not optimal, subsequent splits may produce better trees.

Dates and Times

Dates and times are the most common examples of interval variables.. These variables are very important, because they introduce the time element into data analysis. Often, the importance of date and time variables is that they provide sequence and timestamp information for other variables, such as cause and resolution of the last complaint call.

Because there is a myriad of different formats, working with dates and time stamps can be difficult. Excel has fifteen different date formats prebuilt for cells, and the ability to customize many more. One typical internal format for dates and times is as a single number—the number of days or seconds since some date in the past. When this is the case, data mining algorithms treat dates as numbers. This representation is adequate for the algorithms to detect what happened earlier and later. However, it misses other important properties, which are worth adding into the data:

- Time of day
- Day of the week, and whether it is a workday or weekend
- Month and season
- Holidays

In his book *The Data Warehouse Toolkit* (Wiley, 2002), Ralph Kimball strongly recommends that a calendar be one of the first tables built for a data warehouse. We strongly agree with this recommendation, since the attributes of the calendar are often important for data mining work.

One challenge when working with dates and times is time zones. Especially in the interconnected world of the Web, the time stamp is generally the time stamp from the server computer, rather than the time where the customer is. It is worth remembering that the customer who is visiting the Web site repeatedly in the wee hours of the morning might actually be a Singapore lunchtime surfer rather than a New York night owl.

Fixed-Length Character Strings

Fixed-length character strings usually represent categorical variables, which take on a known set of values. It is always worth comparing the actual values that appear in the data to the list of legal values—to check for illegal values, to verify that the field is always populated, and to see which values are most and least frequent.

Fixed-length character strings often represent codes of some sort. Helpfully, there are often reference tables that describe what these codes mean. The reference tables can be particularly useful for data mining, because they provide hierarchies and other attributes that might not be apparent just looking at the code itself.

Character strings do have an ordering—the alphabetical ordering. However, as the earlier example with Alabama and Alaska shows, this ordering might be useful for librarians, but it is less useful for data miners. When there is a sensible ordering, it makes sense to replace the codes with numbers. For instance, one company segmented customers into three groups: NEW customers with less than 1 year of tenure, MARGINAL customers with between 1 and 2 years, and CORE customers with more than 2 years. These categories clearly have an ordering. In practice, one way to incorporate the ordering would be to map the groups into the numbers 1, 2, and 3. A better way would be to include that actual tenure for data mining purposes, although reports could still be based on the tenure groups.

Data mining algorithms usually perform better when there are fewer categories rather than more. One way to reduce the number of categories is to use attributes of the codes, rather than the codes themselves. For instance, a mobile phone company is likely to have customers with hundreds of different handset equipment codes (although just a few popular varieties will account for the vast bulk of customers). Instead of using each model independently, include features such as handset weight, original release date of the handset, and the features it provides.

Zip codes in the United States provide a good example of a potentially useful variable that takes on many values. One way to reduce the number of values is to use only the first three characters (digits). These are the sectional center facility (SCF), which is usually at the center of a county or large town. They maintain most of the geographic information in the zip code but at a higher level. Even though the SCF and zip codes are numbers, they need to be treated as codes. One clue is that the leading "0" in the zip code is important—the zip code of Data Miners, Inc. is 02114, and it would not make sense without the leading "0".

Some businesses are regional; consequently almost all customers are located in a small number of zip codes. However, there still may be many other customers spread thinly in many other places. In this case, it might be best to group all the rare values into a single "other" category. Another and often better approach, is to replace the zip codes with information about the zip code. There could be several items of information, such as median income and average home value (from the census bureau), along with penetration and response rate to a recent marketing campaign. Replacing string values with descriptive numbers is a powerful way to introduce business knowledge into modeling.

TIP Replacing categorical variables with numeric summaries of the categories—such as product penetration within a zip code—improves data mining models and solves the problem of working with categoricals that have too many values.

Neural networks and K-means clustering are examples of algorithms that want their inputs to be intervals or true numerics. This poses a problem for strings. The naïve approach is to assign a number to each value. However, the numbers have additional information that is not present in the codes, such as ordering. This spurious ordering can hide information in the data. A better approach is to create a set of flags, called *indicator variables*, for each possible value. Although this increases the number of variables, it eliminates the problem of spurious ordering and improves results. Neural network tools often do this automatically.

In summary, there are several ways to handle fixed-length character strings:

- If there are just a few values, then the values can be used directly.
- If the values have a useful ordering, then the values can be turned into rankings representing the ordering.
- If there are reference tables, then information describing the code is likely to be more useful.
- If a few values predominate, but there are many values, then the rarer values can be grouped into an "other" category.
- For neural networks and other algorithms that expect only numeric inputs, values can be mapped to indicator variables.

A general feature of these approaches is that they incorporate domain information into the coding process, so the data mining algorithms can look for unexpected patterns rather than finding out what is already known.

IDs and Keys

The purpose of some variables is to provide links to other records with more information. IDs and keys are often stored as numbers, although they may also be stored as character strings. As a general rule, such IDs and keys should not be used directly for modeling purposes.

A good example of a field that should generally be ignored for data mining purposes are account numbers. The irony is that such fields may improve models, because account numbers are not assigned randomly. Often, they are assigned sequentially, so older accounts have lower account numbers; possibly they are based on acquisition channel, so all Web accounts have higher numbers than other accounts. It is better to include the relevant information explicitly in the customer signature, rather than relying on hidden business rules.

In some cases, IDs do encode meaningful information. In these cases, the information should be extracted to make it more accessible to the data mining algorithms. Here are some examples.

Telephone numbers contain country codes, area codes, and exchanges—these all contain geographical information. The standard 10-digit number in North

American starts with a three-digit area code followed by a three-digit exchange and a four-digit line number. In most databases, the area code provides good geographic information. Outside North America, the format of telephone numbers differs from place to place. In some cases, the area codes and telephone numbers are of variable length making it more difficult to extract geographic information.

Uniform product codes (Type A UPC) are the 12-digit codes that identify many of the products passed in front of scanners. The first six digits are a code for the manufacturer, the next five encode the specific product. The final digit has no meaning. It is a check digit used to verify the data.

Vehicle identification numbers are the 17-character codes inscribed on automobiles that describe the make, model, and year of the vehicle. The first character describes the country of origin. The second, the manufacturer. The third is the vehicle type, with 4 to 8 recording specific features of the vehicle. The 10th is the model year; the 11th is the assembly plant that produced the vehicle. The remaining six are sequential production numbers.

Credit card numbers have 13 to 16 digits. The first few digits encode the card network. In particular, they can distinguish American Express, Visa, MasterCard, Discover, and so on. Unfortunately, the use of the rest of the numbers depends on the network, so there are no uniform standards for distinguishing gold cards from platinum cards, for instance. The last digit, by the way, is a check digit used for rudimentary verification that the credit card number is valid. The algorithm for check digit is called the Luhn Algorithm, after the IBM researcher who developed it.

National ID numbers in some countries (although not the United States) encode the gender and data of birth of the individual. This is a good and accurate source of this demographic information, when it is available.

Names

Although we want to get to know the customers, the goal of data mining is not to actually meet them. In general, names are not a useful source of information for data mining. There are some cases where it might be interesting to classify names according to ethnicity (such as Hispanic names or Asian names) when trying to reach a particular market or by gender for messaging purposes. However, such efforts are at best very rough approximations and not widely used for modeling purposes.

Addresses

Addresses describe the geography of customers, which is very important for understanding customer behavior. Unfortunately, the post office can understand many different variations on how addresses are written. Fortunately, there are service bureaus and software that can standardize address fields.

One of the most important uses of an address is to understand when two addresses are the same and when they are different. For instance, is the delivery address for a product ordered on the Web the same as the billing address of the credit card? If not, there is a suggestion that the purchase is a gift (and the suggestion is even stronger if the distance between the two is great and the giver pays for gift wrapping!).

Other than finding exact matches, the entire address itself is not particularly useful; it is better to extract useful information and present it as additional fields. Some useful features are:

- Presence or absence of apartment numbers
- City
- State
- Zip code

The last three are typically stored in separate fields. Because geography often plays such an important role in understanding customer behavior, we recommend standardizing address fields and appending useful information such as census block group, multi-unit or single unit building, residential or business address, latitude, longitude, and so on.

Free Text

Free text poses a challenge for data mining, because these fields provide a wealth of information, often readily understood by human beings, but not by automated algorithms. We have found that the best approach is to extract features from the text intelligently, rather than presenting the entire text fields to the computer.

Text can come from many sources, such as:

- Doctors' annotations on patient visits
- Memos typed in by call-center personnel
- Email sent to customer service centers
- Comments typed into forms, whether Web forms or insurance forms
- Voice recognition algorithms at call centers

Sources of text in the business world have the property that they are ungrammatical and filled with misspellings and abbreviations. Human beings generally understand them, but it is very difficult to automate this understanding. Hence, it is quite difficult to write software that automatically filters spam even though people readily recognize spam.

Our recommended approach is to look for specific features by looking for specific substrings. For instance, once upon a time, a Jewish group was boycotting a company because of the company's position on Israel. Memo fields typed in by call-center service reps were the best source of information on why customers were stopping. Unfortunately, these fields did not uniformly say "Cancelled due to Israel policy." In fact, many of the comments contained references to "Isreal," "Is rael," "Palistine" [sic], and so on. Classifying the text memos required looking for specific features in the text (in this case, the presence of "Israel," "Isreal," and "Is rael" were all used) and then analyzing the result.

Binary Data (Audio, Image, Etc.)

Not surprisingly, there are other types of data that do not fall into these nice categories. Audio and images are becoming increasingly common. And data mining tools do not generally support them.

Because these types of data can contain a wealth of information, what can be done with them? The answer is to extract features into derived variables. However, such feature extraction is very specific to the data being used and is outside the scope of this book.

Data for Data Mining

Data mining expects data to be in a particular format:

- All data should be in a single table.
- Each row should correspond to an entity, such as a customer, that is relevant to the business.
- Columns with a single value should be ignored.
- Columns with a different value for every column should be ignored—although their information may be included in derived columns.
- For predictive modeling, the target column should be identified and all synonymous columns removed.

Alas, this is not how data is found in the real world. In the real world, data comes from source systems, which may store each field in a particular way. Often, we want to replace fields with values stored in reference tables, or to extract features from more complicated data types. The next section talks about putting this data together into a customer signature.

Constructing the Customer Signature

Building the customer signature, especially the first time, is a very incremental process. At a minimum, customer signatures need to be built at least two times—once for building the model and once for scoring it. In practice, exploring data and building models suggests new variables and transformations, so the process is repeated many times. Having a repeatable process simplifies the data mining work.

The first step in the process, shown in Figure 17.7, is to identify the available sources of data. After all, the customer signature is a summary, at the customer level, of what is known about each customer. The summary is based on available data. This data may reside in a data warehouse. It might equally well reside in operational systems and some might be provided by outside vendors. When doing predictive modeling, it is particularly important to identify where the target variable is coming from.

The second step is identifying the customer. In some cases, the customer is at the account level. In others, the customer is at the individual or household level. In some cases, the signature may have nothing to do with a person at all. We have used signatures for understanding products, zip codes, and counties, for instance, although the most common use is for accounts and households.

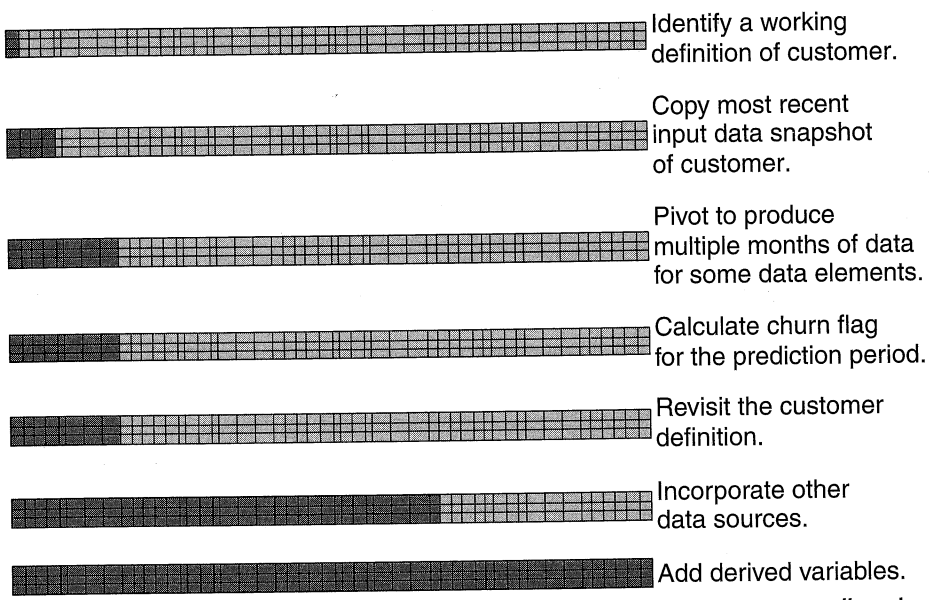


Figure 17.7 Building customer signatures is an iterative process; start small and work through the process step-by-step, as in this example for building a customer signature for churn prediction.

Once the customer has been identified, data sources need to be mapped to the customer level. This may require additional lookup tables—for instance, to convert accounts into households. It may not be possible to find the customers in the available data. Such a situation requires revisiting the customer definition.

The key to building customer signatures is to start simple and build up. Prioritize the data sources by the ease with which they map to the customer. Start with the easiest one, and build the signature using it. You can use a signature before all the data is put into it. While awaiting more complicated data transformations, get your feet wet and understand what is available. When building customer signatures out of transactions, be sure to get all the transactions associated with a particular customer.

Cataloging the Data

The data mining group at a mobile telecommunications company wants to develop a churn model in-house. This churn model will predict churn for one month, given a one-month lag time. So, if the data is available for February, then the churn prediction is for April. Such a model provides time for gathering the data and scoring new customers, since the February data is available sometime in March.

At this company, there are several potential sources of data for the customer signatures. All of these are kept in a data repository with 18 months of history. Each file is an end-of-the-month snapshot—basically a dump of an operational system into a data repository.

The UNIT_MASTER file contains a description of every telephone number in service and a snapshot of what is known about the telephone number at the end of the month. Examples of fields in this file are the telephone number, billing account, billing plan, handset model, last billed date, and last payment.

The TRANS_MASTER file contains every transaction that occurs on a particular telephone number during the course of the month. These are account-level transactions, which include connections, disconnections, handset upgrades, and so on.

The BILL_MASTER file describes billing information at the account level. Multiple handsets might be attached to the same billing account—particularly for business customers and customers on family billing plans.

Although other sources of data were available in the company, these were not immediately highlighted for use for the customer signature. One source, for instance, was the call detail records—a record of every telephone call—that is useful for predicting churn. Although this data was eventually used by the data mining group, it was not part of this initial effort.

Identifying the Customer

The data is typical of the real world. Although the focus might be on one type of customer or another, the data has multiple groups. The sidebar "Residential Versus Business Customers" talks about distinguishing between these two segments.

The business problem being addressed in this example is churn. As shown in Figure 17.8, the customer data model is rather complex, resulting in different options for the definition of customer:

- Telephone number
- Customer ID
- Billing account

This being the real world, though, it is important to remember that these relationships are complex and change over time. Customers might change their telephone numbers. Telephones might be added or removed from accounts. Customers change handsets, and so on. For the purposes of building the signature, the decision was to use the telephone number, because this was how the business reported churn.

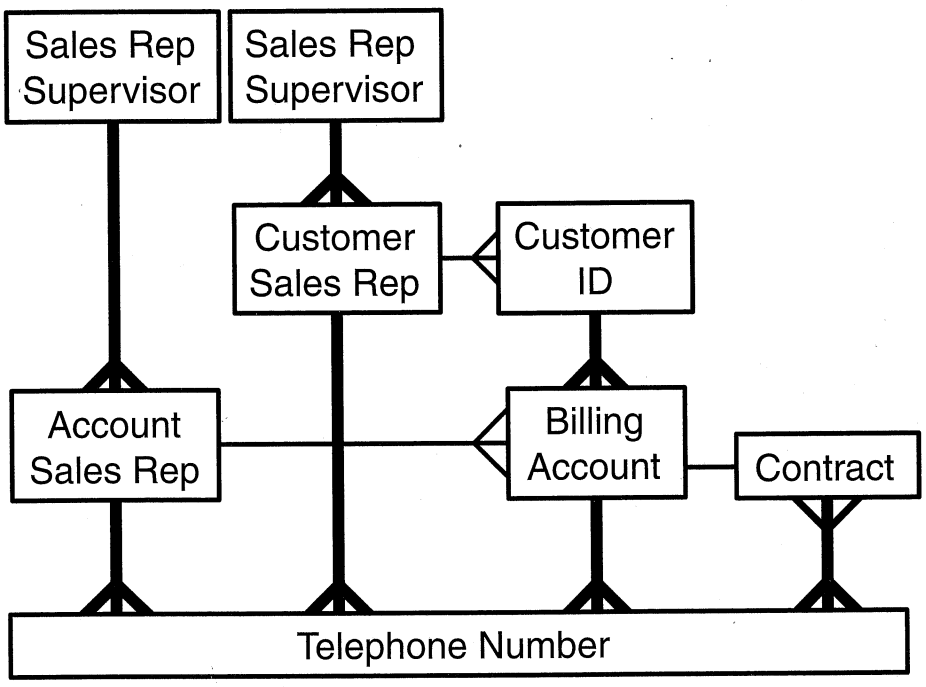


Figure 17.8 The customer model is complicated and takes into account sales, billing, and business hierarchy information.

RESIDENTIAL VERSUS BUSINESS CUSTOMERS

Often data mining efforts focus on one type of customer—such as residential customers or small businesses. However, data for all customers is often mixed together in operational systems and data warehouses. Typically, there are multiple ways to distinguish between these types of customers:

- **Often there is a customer type field, which has values like "residential" and "small business."**
- **There might be a sales hierarchy; some sales channels are business-only while others are residential-only.**
- **Some billing plans are only for businesses; others are only for residential customers.**
- **There might be business rules, so any customer with more than two lines is considered business.**

These examples illustrate the fact that there are typically several different rules for distinguishing between different types of customers. Given the opportunity to be inconsistent, most data sources will not fail. The different rules select different subsets of customers.

Is this a problem? That depends on the particular model being worked on. The hope is that the rules are all very close, so the customers included (or missed) by one rule are essentially the same as those included by the others. It is important to investigate whether or not this is true, and when the rules disagree.

What usually happens in practice is that one of the rules is predominant, because that is the way the business is organized. So, although the customer type might be interesting, the sales hierarchy is probably more important, since it corresponds to people who have responsibility for different customer segments.

The distinction between businesses and residences is important for prospects as well as customers. A long-distance telephone company sees many calls traversing its network that were originated by customers of other carriers. Their switches create call detail records containing the originating and destination telephone numbers. Any domestic number that does not belong to an existing customer belongs to a prospect. One long-distance company builds signatures to describe the behavior of the unknown telephone numbers over time by tracking such things as how frequently the number is seen, what times of day and days of the week it is typically active, and the typical call duration. Among other things, this signature is used to score the unknown telephone numbers for the likelihood that they are businesses because business and residential customers are attracted by different offers.

One simplification would be to focus only on customers whose accounts have only one telephone number. Since the purpose is to build a model for residential customers, this was a good way of simplifying the data model for getting started. If the purpose were to build a model for business customers, a better choice for the customer level would be the billing account level, since

business customers often turn handsets and telephone numbers on and off. However, churn in this case would mean the cancelation of the entire account, rather than the cancelation of a single telephone number. These two situations are the same for those residential customers who have only one line.

First Attempt

The first attempt to build the customer signature needs to focus on the simplest data source. In this case, the simplest data source is the UNIT_MASTER file, which conveniently stores data at the telephone number level, the level being used for the customer signature.

It is worth pointing out two problems with this file and the customer definition:

- Customers may change their telephone number.
- Telephone numbers may be reassigned to new customers.

These problems will be addressed later; the first customer signature is at the telephone number level to get started. The process used to build the signature has four steps: identifying the time frames, creating a recent snapshot, pivoting columns, and calculating the target.

Identifying the Time Frames

The first attempt at building the customer signature needs to take into account the time frame for the data, as discussed in Chapter 3. Figure 17.9 shows a model time chart for this data. The ultimate model set should have more than one time frame in it. However, the first attempt focuses on only one time frame.

The time frame defined churn during 1 month—August. All of the input data come from at least 1 month before. The cutoff date is June 30, in order to provide 1 month of latency.

Taking a Recent Snapshot

The most recent snapshot of data is defined by the cutoff date. These fields in the signature describe the most recent information known about a customer before he or she churned (or did not churn).

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov
SCORE						4	3	2	1		P
MODEL SET				4	3	2	1		P		
MODEL SET			4	3	2	1		P			

Figure 17.9 A model time chart shows the time frame for the input columns and targets when building a customer signature.

This is a set of fields from the UNIT_MASTER file for June—fields such as the handset type, billing plan, and so on. It is important to keep the time frame in mind when filling the customer signature. It is a good idea to use a naming convention to avoid confusion. In this case, all the fields might have a suffix of "_01," indicating that they are from the most recent month of input data.

TIP Use a naming convention when building the customer signature to indicate the time frame for each variable. For instance, the most recent month of input data would have a "_01" suffix; the month before, "_02"; and so on.

At this point, presumably not much is known about the fields, so descriptive information is useful. For instance, the billing plan might have a description, monthly base, per-minute cost, and so on. All of these features are interesting and of potential value for modeling—so it is reasonable to bring them into the model set. Although descriptions are not going to be used for modeling (codes are much better), they help the data miners understand the data.

Pivoting Columns

Some of the fields in UNIT_MASTER represent data that is reported in a regular time series. For instance, bill amount has a value for every month, and each of these values needs to be put into a separate column. These columns come from different UNIT_MASTER records, one for June, one for May, one for April, and so on. Using a naming convention, the fields would be, for example:

- Last_billed_amount_01 for June (which may already be in the snapshot)
- Last_billed_amount_02 for May
- Last_billed_amount_03 for April

At this point, the customer signature is starting to take shape. Although the input fields only come from one source, the appropriate fields have been chosen as input and aligned in time.

Calculating the Target

A customer signature for predictive modeling would not be useful without a target variable. Since the customer signature is going to be used for churn modeling, the target needs to be whether or not the customer churned in August. This is in the account status field for the August UNIT_MASTER record. Note that only customers who were active on or before June 30 are included in the model set. A customer that starts in July and cancels in August is not included.

Making Progress

Although quite rudimentary, the customer signature is ready for use in a model set. Having a well-defined time frame, a target variable, and input variables, it is functional, even if minimally so. Although useful and a place to get started, the signature is missing a few things.

First, the definition of customer does not take into account changes in telephone numbers. The TRANS_MASTER file solves this problem, because it keeps track of these types of changes on customers' accounts. To fix the definition of customer requires creating a table, which has the original telephone number on the account (with perhaps a counter, since a telephone number can actually be reused). A typical row in this table would have the following columns:

- Telephone Number
- Effective Date
- End Date
- Unique Customer Identifier

With this table, the customer identifier can be used instead of the telephone number, so the customer signatures are robust with respect to changes in telephone number.

Another shortcoming of the customer signature is its reliance on only one data source. Additional data sources should be added in, one at a time, to build a richer signature of customer behavior. The model set only has one time frame of data. Additional time frames make models that are more stable. This customer signature also lacks derived variables, which are the subject of much of the rest of this chapter.

Practical Issues

There are some practical issues encountered when building customer signatures. Customer signatures often bring together the largest sources of data and perform complex operations on them. This becomes an issue in terms of computing resources. Although the resulting model set probably has at most tens or hundreds of megabytes, the data being summarized could be thousands of times larger. For this reason, it is often a good idea to do as much of the processing as possible in relational databases, because these can take advantage of multiple processors and multiple disks at the same time.

Although the resulting queries are complicated, much of the work of putting together the signatures can be done in SQL or in the database's scripting language. This is useful not only because it increases efficiency, but also because the code then resides in only one place—reducing the possibility of

error and increasing the ability to find bugs when they occur. Alternatively, the data can be extracted from the source and pieced together. Increasingly, data mining tools are becoming better at manipulating data. However, this generally requires some amount of programming, in a language such as SAS, SPSS, S-Plus, or Perl. The additional processing not only adds time to the effort, but it also introduces a second level where bugs might creep in.

It is important when creating signatures to realize that data mining is an iterative process that often requires rebuilding the signature. A good approach is to create a template for pulling one time frame of data from the data sources, and then to do multiple such pulls to create the model set. For the score set, the same process can be used, since the score set closely resembles the model set.

Exploring Variables

Data exploration is critically intertwined with the data mining process. In many ways, data mining and data exploration are complementary ways of achieving the same goal. Where data mining tends to highlight the interesting algorithms for finding patterns, data exploration focuses more on presenting data so that people can intuit the patterns. When it comes to communicating results, pretty pictures that *show* what is happening are often much more effective than dry tables of numbers. Similarly, when preparing data for data mining, seeing the data provides insight into what is happening, and this insight can help improve models.

Distributions Are Histograms

The place to start when looking at data is with histograms of each field; histograms show the distribution of values in fields. Actually, there is a slight difference between histograms and distributions, because histograms count occurrences, whereas distributions are normalized. However, for our purposes, the similarities are much more important—histograms and distributions (or strictly speaking, the density function associated with the distribution) have similar shapes; it is only the scale of the Y-axis that changes.

Most data mining tools provide the ability to look at the values that a single variable takes on as a histogram. The vertical axis is the number of times each value occurs in the sample; the horizontal axis shows the various values.

Numeric variables are often binned when creating histograms. For the purpose of exploring the variables, these bins should be of equal width and not of equal height. Remember that equal-height binning creates bins that all contain the same number of values. Bins containing similar numbers of records are useful for modeling; however, they are less useful for understanding the variables themselves.

Changes over Time

Perhaps the most revealing information becomes apparent when the time element is incorporated into a histogram. In this case, only a single value of a variable is used. The chart shows how the frequency of this value changes over time.

As an example, the chart in Figure 17.10 shows fairly clearly that something happened during one March with respect to the value "DN." This type of pattern is important. In this case, the "DN" represents duplicate accounts that needed to be canceled when two different systems were merged. In fact, we stumbled across the explanation only after seeing such a patterns and asking questions about what was happening during this time period.

The top chart shows the raw values, and that can be quite useful. The bottom one shows the standardized values. The curves in the two charts have the same shape; the only difference is the vertical scale. Remember that standardizing values converts them into the number of standard deviations from the mean, so values outside the range of –2 to 2 are unusual; values less then –3 or greater than 3 should be very rare. Visualizing the same data shows that the peaks are many standard deviations outside expected values—and 14 standard deviations is highly suspect. The likelihood of this happening randomly is so remote that the chart suggests that something external is affecting the variable—something external like the one-time even of merging of two computer systems, which is how the duplicate accounts were created.

Creating one cross-tabulation by time is not very difficult. Unfortunately, however, there is not much support in data mining tools for this type of diagram. They are easy to create in Excel or with a bit of programming in SAS, SPSS, S-Plus, or just about any other programming language. The challenge is that many such diagrams are needed—one for each value taken on by each categorical variable. For instance, it is useful to look at:

- Different types of accounts opened over time.
- Different reasons why customers stop over time.
- Performance of certain geographies over time.
- Performance of different channels over time.

Because these charts explicitly go back in time, they bring up issues of what happened when. They can be useful for spotting particularly effective combinations that might not otherwise be obvious—such as "oh, our Web banner click-throughs go up after we do email campaigns."

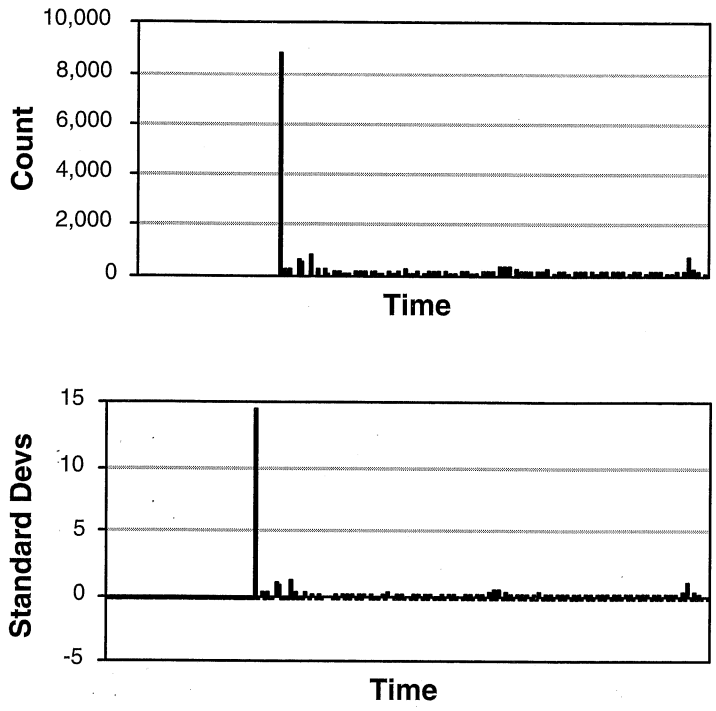


Figure 17.10 This histogram suggests that something unusual was happening with this stop code. The top diagram is the raw data; in the bottom one, the values are standardized.

Crosstabulations

Looking at variables over time is one example of a cross-tabulation. In general, cross-tabulations show how frequently two variables occur with respect to each other. Figure 17.11 shows a cross-tabulation between two variables, channel and credit card payment. The size of the bubble shows the proportion of customers starting in the channel with that payment method. This is the same data shown in Table 17.2.

Cross-tabulations without time show static images rather than trends. This is useful, but trend information is usually even more useful.

Table 17.2 Cross Tabulation of Channels by Payment Method

	CREDIT CARD	DIRECT BILL
DM	69,126	51,481
TM	50,105	249,208
WEB	67,830	29,608

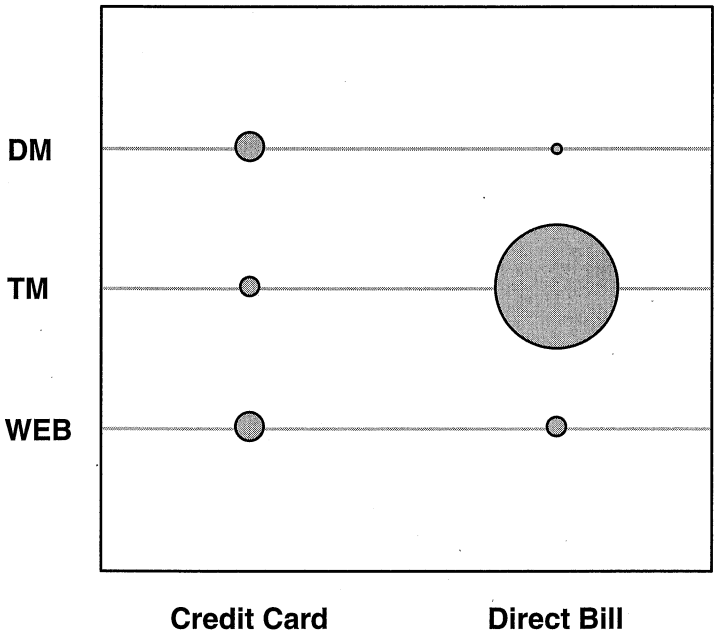


Figure 17.11 Cross-tabulations show relationships between variables.

Deriving Variables

There have been many examples of derived variables in this chapter and throughout this book. Such variables are predigested, making it easier for data mining algorithms to incorporate them into models. Perhaps more important, derived variables make it possible to incorporate domain knowledge into the data mining process. Put the domain information into the data so that the data mining algorithms can use it to find patterns.

Because adding variables is central to any successful data mining project, it is worth looking at the six basic ways that derived variables are calculated in a bit of detail. These six methods are:

- Extracting features from a single value
- Combining values within a record (used, among other things, for capturing trends)
- Looking up auxiliary information in another table
- Pivoting time-dependent data into multiple columns
- Summarizing transactional records
- Summarizing fields across the model set

The following sections discuss these methods, giving examples of derived variables and highlighting important points about computing them.

Extracting Features from a Single Value

Computationally, parsing values is a very simple operation because all the data needed is present in a single value. Even though it is so simple, it is quite useful, as these examples show:

- Calculating the day of the week from a date
- Extracting the credit card issuer code from a credit card number
- Taking the SCF (first three digits) of a zip code
- Determining the vehicle manufacturer code from the VIN
- Adding a flag when a field is missing

These operations generally require rudimentary operations that data mining tools should be able to handle. Unfortunately, many statistical tools focus more on numeric data types than on the strings, dates, and times often encountered in business data—so string operations and date arithmetic can be difficult. In such cases, these variables may need to be added during a preprocessing phase or as data is extracted from data sources.

Combining Values within a Record

As with the extraction of features from a single value, combining values within a record is computationally simple—instead of using one variable, there are several variables. Most data mining tools support adding derived variables that combine values from several fields, particularly for numeric fields. This can be very useful, for adding ratios, sums, averages, and so on. Such derived values are often more useful for modeling purposes than the raw data because these variables start to capture underlying customer behavior. Date fields are often combined. Taking the difference of two dates to calculate duration is an especially common and useful example.

It is not usually necessary to combine string fields, unless the fields are somehow related. For instance, it might be useful to combine a "credit card payment flag" with a "credit card type," so there is one field representing the payment type.

Looking Up Auxiliary Information

Looking up auxiliary information is a more complicated process than the previous two calculations. A lookup is an example of joining two tables together (to use relational database terminology), with the simplifying assumption that one table is big and the other table is relatively small.

When the lookup table is small enough, such as Table 17.3, which describes the mapping between initial digits of a credit card number and the credit card type, then a simple formula can suffice for the lookup.

The more common situation is having a secondary table or file with the information. This table might, for instance, contain:

- Populations and median household incomes of zip codes (usefully provided for downloading for the United States by the U.S. Census Bureau at www.census.gov)
- Hierarchies for product codes
- Store type information about retail locations

Unfortunately data mining tools do not, as a rule, make it easy to do lookups without programming. Tools that do provide this facility, such as I-Miner from Insightful, usually require that both tables be sorted by the field or fields used for the lookup; an example of this is shown in Figure 17.12. This is palatable for one such field, but it is cumbersome when there are many different fields to be looked up. In general, it is easier to do these lookups outside the tool, especially when the lookup tables and original data are both coming from databases.

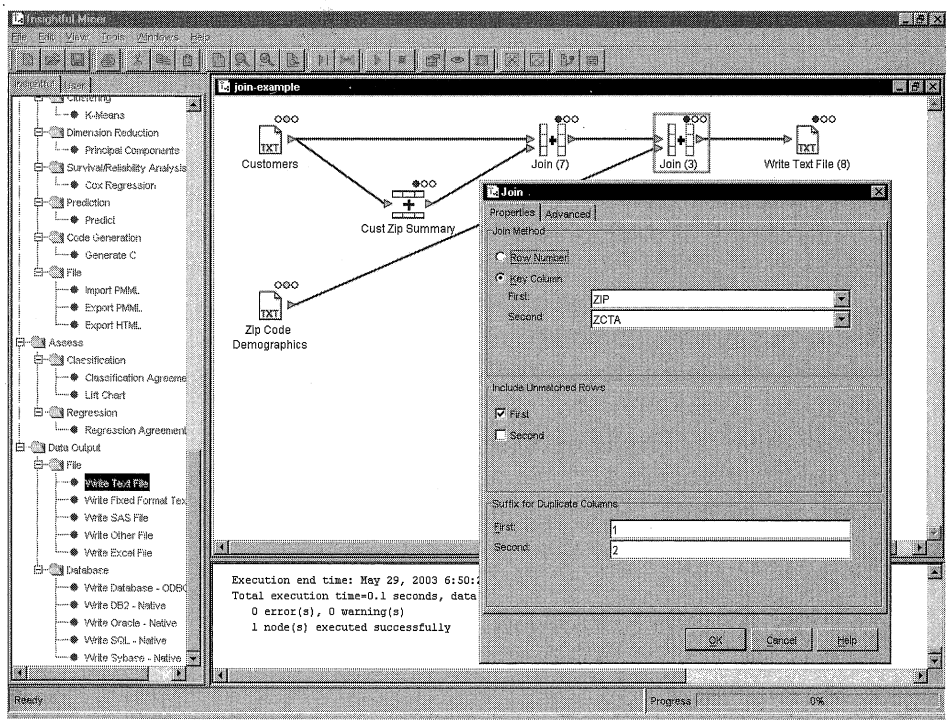


Figure 17.12 Insightful Miner enables users to use and create lookup tables from the graphical user interface.

Sometimes, the lookup tables already exist. Other times, they must be created as needed. For instance, one useful predictor of customer attrition is the historical attrition rate by zip code. To add this to a customer signature requires calculating the historical attrition rate for each zip code and then using the result as a lookup table.

WARNING When using database joins to look up values in a lookup table, always use a left outer join to ensure that no customer rows are lost in the process! An outer join in SQL looks like:

```
SELECT c.*, l.value
FROM (customer c left outer join lookup l on c.code = l.code)
```

Table 17.3 Credit Card Prefixes

CARD TYPE	PREFIX	LENGTH
MasterCard	51	16
MasterCard	52	16
MasterCard	53	16
MasterCard	54	16
MasterCard	55	16
Visa	4	13
Visa	4	16
American Express	34	15
American Express	37	15
Diners Club	300	14
Diners Club	301	14
Diners Club	302	14
Diners Club	303	14
Diners Club	304	14
Diners Club	305	14
Discover	6011	16
enRoute	2014	15
enRoute	2149	15
JCB	3	16
JCB	2131	15
JCB	1800	15

Pivoting Regular Time Series

Data about customers is often stored at a monthly level, where each month has a separate row of data. For instance, billing data is often stored this way, since most subscription-based companies bill customers once a month. This data is an example of a regular time series, because the data occurs at fixed, defined intervals. Figure 17.13 illustrates the process needed to put this data into a customer signature. The data must be pivoted, so values that start out in rows end up in columns.

This is generally a cumbersome process, because neither data mining tools nor SQL makes it easy to do pivoting. Data mining tools generally require programming for pivoting. To accomplish this, the customer file needs to be sorted by customer ID, and the billing file needs to be sorted by the customer ID and the billing date. Then, special-purpose code is needed to calculate the pivoting columns. In SAS, proc TRANSPOSE is used for this purpose. The sidebar "Pivoting Data in SQL" shows how it is done in SQL.

Most businesses store customer data on a monthly basis, usually by calendar month. Some industries, though, show strong weekly cyclical patterns, because customers either do or do not do things over the weekend. For instance, Web sites might be most active during weekdays, and newspaper subscriptions generally start on Mondays or Sundays.

Such weekly cycles interfere with the monthly data, because some months are longer than others. Consider a Web site where most activity is on weekdays. Some months have 20 weekdays; others have up to 23 (not including holidays). The difference between successive months could be 15 percent, due solely to the difference in the number of weekdays. To take this into account, divide the monthly activity by the number of weekdays during the month, to get an "activity per weekday." This only makes sense, though, when there are strong weekly cycles.

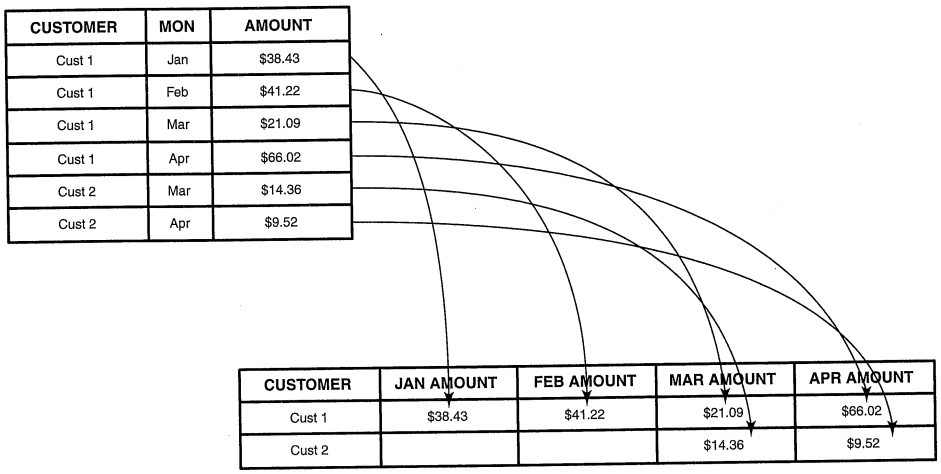


Figure 17.13 Pivoting a field takes values stored in one or more rows for each customer and puts them into a single row for each customer, but in different columns.

PIVOTING DATA IN SQL

SQL does not have great support for pivoting data (although some databases may have nonstandard extensions with this capability). However, when using standard SQL it is possible to pivot data.

Assume that the data consists of billing records and that each has a sequential billing number assigned to it. The first billing record has a "1," the second "2," and so on. The following SQL fragment shows how to pivot this data:

```
SELECT customer_id,
       sum(case when bill_seq = 1 then bill_amt end) as bill_1,
       sum(case when bill_seq = 2 then bill_amt end) as bill_2,
       sum(case when bill_seq = 3 then bill_amt end) as bill_3,
       . . .
FROM billing
GROUP BY customer_id
```

One problem with this fragment is that different customers have different numbers of billing periods. However, the query can only take a fixed number. When a customer has fewer billing periods than the query wants, then the later periods are filled with NULLs.

Actually, this code fragment is not generally what is needed for customer signatures because the signature wants the most recent billing periods—such as the last 12 or 24. For customers who are active, this is the most recent period. However, for customers who have stopped, this requires considering their stop date instead. The following code fragment takes this into account:

```
SELECT customer_id,
       sum(case when trunc(months_between(bill_date, cutoff) = 1
                then bill_amt else 0 end) as bill_1,
       sum(case when trunc(months_between(bill_date, cutoff) = 2
                then bill_amt else 0 end) as bill_2,
       . . .
FROM billing b,
     (select customer_id,
             (case when status = 'ACTIVE' then sysdate
                   else stop_date end) as cutoff
      from customer) c
where b.customer_id = c.customer_id
GROUP BY customer_id
```

This code fragment does use some extensions to SQL for the date calculations (these are expressed as Oracle functions in this example). However, most databases have similar functions.

The above code is an example of a killer query, because it is joining a big table (the customer table) with an even bigger table (the customer billing table) and then doing a grouping operation. Fortunately, modern databases can take advantage of multiple processors and multiple disks to perform this query in a reasonable amount of time.

Summarizing Transactional Records

Transactional records are an example of an irregular time series—that is, the records can occur at any point in time. Such records are generated by customer interactions, as is the case with:

- Automated teller machine transactions
- Telephone calls
- Web site visits
- Retail purchases

There are several challenges when working with irregular time series. First, the transaction volumes are very, very large. Working with such voluminous data requires sophisticated tools and powerful computers. Second, there is no standard way of working with them. The regular time series data has a natural way of pivoting. For irregular time series, it is necessary to determine how best to summarize the data.

One way is to transform the irregular time series into regular time series and then to pivot the series. For instance, calculate the number of calls per month or the amount withdrawn from ATMs each month, and then pivot the sums by month. When working with transactions, these calculations can be more complex, such as the number of calls longer than 10 minutes or the number of withdrawals less than $50. These specialized summaries can be quite useful. More complicated examples that describe customer behavior are provided just after the next section.

Another approach is to define a set of data transformations that are run on the transactional data as it is being collected. This is an approach taken in the telecommunications industry, where the volume of data is vast. Some variables may be as simple as minutes of use, others may be a complex as a score for whether the calling number is a business or residence. This approach hard-codes the calculations, and such calculations are hard to change. Although such variables can be useful, a more flexible environment for summarizing transactional data is strategically more useful.

Summarizing Fields across the Model Set

The last method for deriving variables is summarizing values across fields in the customer signature itself. There are several examples of such fields:

- Binning values into equal sized bins requires calculating the breakpoints for the bins.
- Standardizing a value (subtracting the mean and dividing by the standard deviation) requires calculating the mean and standard deviation for the field and then doing the calculation.

- Ranking a value (so the smallest value has a value of 1, the second smallest 2, and so on) requires sorting all the values to get the ranking.

Although these are complicated operations, they are performed directly on the model set. Data mining tools provide support for these operations, especially for binning numeric values, which is the most important of the three.

One type of binning that would be very useful is not readily available. This is binning for codes based on frequency. That is, it would be useful to keep all codes that have at least, say, 1,000 instances in the model set and to place all other codes in a single "other" category. This is useful for working with outliers, such as the many old and unpopular handsets that show up in mobile telephone data although few customers use them. One way to handle this is to identify the handsets to keep and to add a new field "handset for analysis" that keeps these handsets and places the rest into an "other" category. A more automated way is to create a lookup table to map the handsets. However, perhaps a better way is to replace the handset ID itself with information such as the date the handset was released, its weight, and the features it uses—information that is probably available in a lookup table already.

Examples of Behavior-Based Variables

The real power of derived variables comes from the ability to summarize customer behaviors along known dimensions. This section builds on the ideas already presented and gives three examples of useful behavior-based variables.

Frequency of Purchase

Once upon a time, catalogers devised a clever method for characterizing customer behavior using three dimensions—recency, frequency, and monetary value. RFM, which relies on these three variables, has been used at least since the 1970s. Of these three descriptions of customer behavior, recency is usually the most predictive, but frequency is the most interesting. Recency simply means the length of time since a customer made a purchase. Monetary value is traditionally the total amount purchased (although we have found the average purchase value more useful since the total is highly correlated with frequency).

In traditional RFM analysis, frequency is just the number of purchases. However, a simple count does not do a good job of characterizing customer behavior. There are other approaches to determining frequency, and these can be applied to other areas not related to catalog purchasing—frequency of complaints, frequency of making international telephone calls, and so on. The important point is that customers may perform an action at irregular intervals, and we want to characterize this behavior pattern because it provides potentially useful information about customers.

One method of calculating frequency would be to take the length of time indicated by the historical data and divide it by the number of times the customer made a purchase. So, if the catalog data goes back 6 years and a customer made a single purchase, then that frequency would be once every 6 years.

Although simple, this approach misses an important point. Consider two customers:

- John made a purchase 6 years ago and has received every catalog since then.
- Mary made a purchase last month when she first received the catalog.

Does it make sense that both these customers have the same frequency? No. John more clearly has a frequency of no more than once every 6 years. Mary only had the opportunity to make one purchase in the past month, so her frequency would more accurately be described as once per month. The first point about frequency is that it should be measured from the first point in time that a customer had an opportunity to make a purchase.

There is another problem. What we really know about John and Mary is that their frequencies are no more than once every 6 years and no more than once per month, respectively. Historically, one observation does not contain enough information to deduce a real frequency. This is really a time to event problem, such as those discussed in Chapter 12.

Our goal here is to characterize frequency as a derived variable, rather than predict the next event (which is best approached using survival analysis). To do this, let's assume that there are two or more events, so the average time between events is the total span of time divided by the number of events minus one, as shown in Figure 17.14. This provides the average time between events for the period when the events occurred.

There is no perfect solution to the question of frequency, because customer events occur irregularly and we do not know what will happen in the future—the data is censored. Taking the time span from the first event to the most recent event runs into the problem that customers whose events all took place long ago may have a high frequency. The alternative is to take the time since the first event, in essence pretending that the present is an event. This is unsatisfying, because the next event is not known, and care must be taken when working with censored data. In practice, taking the number of events since the first event could have happened and dividing by the total span of time (or the span when the customer was active) is the best solution.

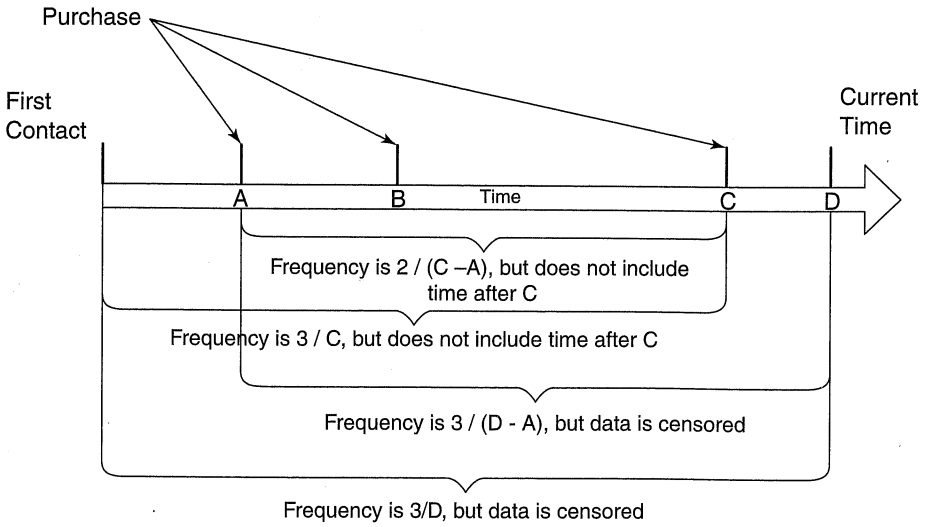


Figure 17.14 There is no perfect way to estimate frequency, but these four ways are all reasonable.

Declining Usage

In telecommunications, one significant predictor of churn is declining usage—customers who use services less and less over time are more likely to leave than other customers. Customers who have declining usage are likely to have many variables indicating this:

- Billing measures, such as recent amounts spent are quite small.
- Usage measures, such as recent amounts used are quite small or always at monthly minimums.
- Optional services recently have no usage.
- Ratios of recent measures to older measures are less than 1, often significantly less than one, indicating recent usage is smaller than historical usage.

The existence of so many different measures for the same underlying behavior suggests a situation where a derived variable might be useful to capture the behavior in a single variable. The goal is to incorporate as much information as possible into a "declining usage" indicator.

TIP When many different variables all suggest a single customer behavior, then it is likely that a derived variable that incorporates this information will do a better job for data mining.

Fortunately, mathematics provides an elegant solution, in the form of the best fit line, as shown in Figure 17.15. The goodness of fit is described by the R^2 statistic, which varies from 0 to 1, with values near 0 being poor fit and values near 1 being very good. The slope of the line provides the average rate of increase or decrease in some variable over time. In statistics, this slope is called the beta function and is calculated according to the following formula:

```
Sum of (x-average(x))*(y-average(y)) / sum((x-average(x))^2)
```

To give an example of how this might be used, consider the following data for the customer shown in the previous figure. Table 17.4 walks through the calculation for a typical customer.

Table 17.4 Example of Calculating the Slope for a Time Series

MONTH (*X* –VALUE)	*X* – AVG(X)	(*X* – AVG (X))^2	*Y* (FROM CUST A)	Y – AVG(Y)	(*X* – AVG(X)) * (*Y* – AVG(*Y*))
1	–5.5	30.25	53.47	3.19	–17.56
2	–4.5	20.25	46.61	–3.67	16.52
3	–3.5	12.25	47.18	–3.10	10.84
4	–2.5	6.25	49.54	–0.74	1.85
5	–1.5	2.25	48.71	–1.57	2.35
6	–0.5	0.25	52.04	1.76	–0.88
7	0.5	0.25	48.45	–1.83	–0.91
8	1.5	2.25	54.16	3.88	5.83
9	2.5	6.25	54.47	4.19	10.47
10	3.5	12.25	53.69	3.42	11.95
11	4.5	20.25	45.93	–4.35	–19.59
12	5.5	30.25	49.10	–1.18	–6.51
TOTAL		143			14.36
SLOPE					0.1004

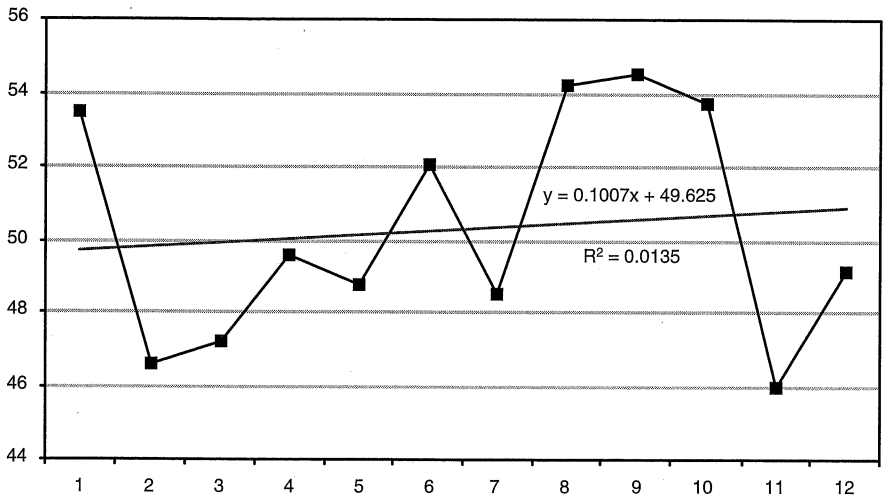


Figure 17.15 The slope of the line of best fit provides a good measure of changes over time.

This example shows a very typical use for calculating the slope—finding the slope over the previous year's usage or billing patterns. The tabular format shows the calculation in a way most suitable for a spreadsheet. However, many data mining tools provide a function to calculate beta values directly from a set of variables in a single row. When such a function is not available, it is possible to express it using more basic arithmetic functions.

Although monthly data is often the most convenient for such calculations, remember that different months have different numbers of days. This issue is particularly significant for businesses that have strong weekly cycles. Some months have five full weekends, for instance, while others only have four. Different months have between 20 and 23 working days (not including holidays). These differences can account for up to 25 percent of the difference between months. When working with data that has such cycles, it is a good idea to calculate the "average per weekend" or "average per working day" to see how the chosen measure is changing over time.

TIP When working with data that has weekly cycles but must be reported by month, consider variables such as "average per weekend day" or "average per work day" so that comparisons between months are more meaningful.

Revolvers, Transactors, and Convenience Users: Defining Customer Behavior

Often, business people can characterize different groups of customers based on their behavior over time. However, translating an informal business description into a form useful for data mining is challenging. Faced with such a challenge, the best response is to determine measures of customer behavior that match the business understanding.

This example is about a credit card group at a major retail bank, which has found that profitable customers come in three flavors:

- *Revolvers* are customers who maintain large balances on their credit cards. These are highly profitable customers because every month they pay interest on large balances.
- *Transactors* are customers who have high balances every month, but pay them off. These customers do not pay interest, but the processing fee charged on each transaction is an important source of revenue. One component of the transaction fee is based on a percentage of the transaction value.
- *Convenience users* are customers who periodically charge large amounts, for vacations or large purchases, for example, and then pay them off over several months. Although not as profitable as revolvers, they are lower risk, while still paying significant amounts of interest.

The marketing group believes that these three types of customers are motivated by different needs. So, understanding future customer behavior would allow future marketing campaigns to send the most appropriate message to each customer segment. The group would like to predict customer behavior 6 months in the future.

The interesting part of this example is not the prediction, but the definition of the segments. The training set needs examples where customers are already classified into the three groups. Obtaining this classification proves to be a challenge.

Data

The data available for this project consisted of 18 months of billing data, including:

- Credit limit
- Interest rate
- New charges made during each month
- Minimum payment
- Amount paid
- Total balance in each month
- Amount paid in interest and related charges each month

The rules for these credit cards are typical. When a customer has paid off the balance, there is no interest on new charges (for 1 month). However, when there is an outstanding balance, then interest is charged on both the balance and on new charges. What does this data tell us about customers?

Segmenting by Estimating Revenue

Estimated revenue is a good way of understanding the value of customers. (By itself, this value does not provide much insight into customer behavior, so it is not very useful for messaging.) Basing customer value on revenue alone assumes that the costs for all customers are the same. This is not true, but it is a useful approximation, since a full profitability model is quite complicated, difficult to develop, and beyond the scope of this example.

Table 17.5 illustrates 1 month of billing for six customers. The last column is the estimated revenue, which has two components. The first is the amount of interest paid. The second is the transaction fee on new transactions, which is estimated to be 1 percent of the new transaction volume for this example.

Table 17.5 Six Credit Card Customers and 1 Month of Data

	CREDIT LIMIT	RATE	NEW CHARGES	BEGINNING BALANCE	MIN PAYMENT	AMOUNT PAID	INTEREST	TRANSACTION REVENUE	EST. REVENUE
Customer 1	$500	14.9%	$50	$400	$15	$15	$4.97	$0.50	$5.47
Customer 2	$5,000	4.9%	$0	$4,500	$135	$135	$18.38	$0.00	$18.38
Customer 3	$6,000	11.9%	$100	$3,300	$99	$1,000	$32.73	$1.00	$33.73
Customer 4	$10,000	14.9%	$2,500	$0	$0	$75	$0.00	$25.00	$25.00
Customer 5	$8,000	12.9%	$6,500	$0	$0	$6,500	$0.00	$65.00	$65.00
Customer 6	$5,000	17.9%	$0	$4,500	$135	$135	$67.13	$0.00	$67.13

Estimated revenue is a good way to compare different customers with a single number. The table clearly shows that someone who rarely uses the credit card (Customer 1) has very little estimated revenue. On the other hand, those who make many charges or pay interest create a larger revenue stream.

However, estimated revenue does not differentiate between different types of customers. In fact, a transactor (Customer 5) has very high revenue. So, does a revolver who has no new charges (Customer 6). This example shows that estimated revenue has little relationship to customer behavior. Frequent users of the credit card and infrequent users both generate a lot of revenue. And this is to be expected, since there are different types of profitable customers.

The real world is more complicated than this simplified example. Each customer has a risk of bankruptcy, where the outstanding balance must be written off. Different types of cards have different rules. For instance, many co-branded cards have the transaction fee going to the co-branded institution. And, the cost of servicing different customers varies, depending on whether the customer uses customer service, disputes charges, pays bills online, and so on.

In short, estimating revenue is a good way of understanding which customers are valuable. But, it does not provide much insight into customer behavior.

Segmentation by Potential

In addition to actual revenue, each customer has a potential revenue. This is the maximum amount of revenue that the customer could possibly bring in each month. The maximum revenue is easy to calculate. Simply assume that the entire credit line is used either in new charges (hence transaction revenue) or in carry-overs (hence interest revenue). The greater of these is the potential revenue.

Table 17.6 compares the potential revenue with the actual revenue for the same six customers during one month. This table shows some interesting characteristics. Some not-so-profitable customers are already saturating their potential. Without increasing their credit limits or interest rate, it is not possible to increase their value.

Table 17.6 Potential of Six Credit Card Customers

	CREDIT LIMIT	RATE	INTEREST	TRANSACTION	POTENTIAL REVENUE	ACTUAL	POTENTIAL
Customer 1	$500	14.9%	$6.21	$5.00	$6.21	$5.47	88%
Customer 2	$5,000	4.9%	$20.42	$50.00	$50.00	$18.38	37%
Customer 3	$6,000	11.9%	$59.50	$60.00	$60.00	$33.73	56%
Customer 4	$10,000	14.9%	$124.17	$100.00	$124.17	$25.00	20%
Customer 5	$8,000	12.9%	$86.00	$80.00	$86.00	$65.00	76%
Customer 6	$5,000	17.9%	$74.58	$50.00	$74.58	$67.13	90%

There is another aspect of comparing actual revenue to potential revenue; it normalizes the data. Without this normalization, wealthier customers appear to have the most potential, although this potential is not fully utilized. So, the customer with a $10,000 credit line is far from meeting his or her potential. In fact, it is Customer 1, with the smallest credit line, who comes closest to achieving his or her potential value. Such a definition of value eliminates the wealth effect, which may or may not be appropriate for a particular purpose.

Customer Behavior by Comparison to Ideals

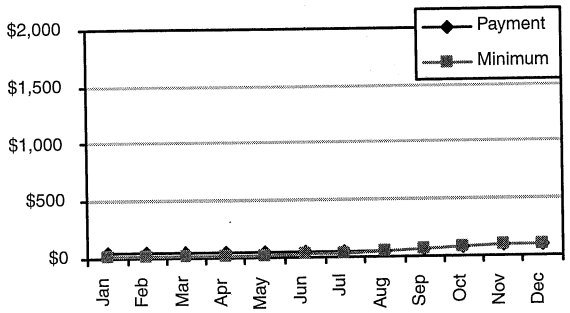
Since estimating revenue and potential does not differentiate among types of customer behavior, let's go back and look at the definitions in more detail. First, what is it inside the data that tells us who is a revolver? Here are some definitions of a revolver:

- Someone who pays interest every month
- Someone who pays more than a certain amount of interest every month (say, more than $10)
- Someone who pays more than a certain amount of interest, almost every month (say, more than $10 in 80 percent of the months)

All of these have an ad hoc quality (and the marketing group had historically made up definitions similar to these on the fly). What about someone who pays very little interest, but does pay interest every month? Why $10? Why 80 percent of the months? These definitions are all arbitrary, often the result of one person's best guess at a definition at a particular time.

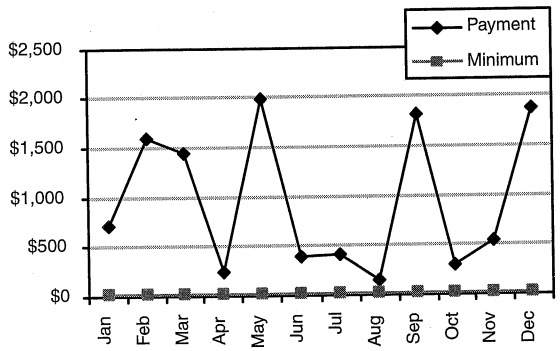
From the customer perspective, what is a revolver? It is someone who only makes the minimum payment every month. So far, so good. For comparing customers, this definition is a bit tricky because the minimum payments change from month to month and from customer to customer.

Figure 17.16 shows the actual and minimum payments made by three customers, all of whom have a credit line of $2,000. The revolver makes payments that are very close to the minimum payment each month. The transactor makes payments closer to the credit line, but these monthly charges vary more widely, depending on the amount charged during the month. The convenience user is somewhere in between. Qualitatively, the shapes of the curves provide insight into customer behavior.



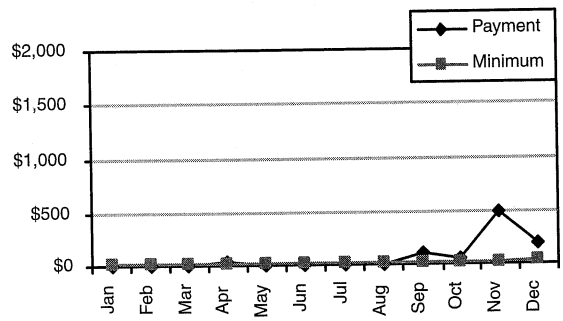
A typical revolver only pays on or near the minimum balance every month.

This revolver has maintained an average balance of $1,070, with new charges of about $200 dollars.



A typical transactor pays off the bill every month. The payment is typically much larger than the minimum payment, except in months with few charges.

This transactor has an average balance of $1,196.



A typical convenience user uses the card when necessary and pays off the balance over several months.

This convenience user has an average balance of $524.

Figure 17.16 These three charts show actual and minimum payments for three credit card customers with a credit line of $2,000.

Manually looking at shapes is an inefficient way to categorize the behavior of several million customers. Shape is a vague, qualitative notion. What is needed is a score. One way to create a score is by looking at the area between the "minimum payment" curve and the actual "payment" curve. For our purposes, the area is the sum of the differences between the payment and the minimum. For the revolver, this sum is $112; for the convenience user, $559.10; and for the transactor, a whopping $13,178.90.

This score makes intuitive sense. The lower it is, the more the customer looks like a revolver. However, the score does not work for comparing two cardholders with different credit lines. Consider an extreme case. If a cardholder has a credit line of $100 and was a perfect transactor, then the score would be no more than $1,200. And yet an imperfect revolver with a credit line of $2,000 has a much larger score.

The solution is to normalize the value by dividing each month's difference by the total credit line. Now, the three scores are 0.0047, 0.023, and 0.55, respectively. When the normalized score is close to 0, the cardholder is close to being a perfect revolver. When it is close to 1, the cardholder is close to being a perfect transactor. Numbers in between represent convenience users. This provides a revolver-transactor score for each customer, with convenience users falling in the middle.

This score for customer behavior has some interesting properties. Someone who never uses their card would have a minimum payment of 0 and an actual payment of 0. These people look like revolvers. That might not be a good thing. One way to resolve this would be to include the estimated revenue potential with the behavior score, in effect, describing the behavior using two numbers.

Another problem with this score is that as the credit line increases, a customer looks more and more like a revolver, unless the customer charges more. To get around this, the ratios could instead be the monthly balance to the credit line. When nothing is owed and nothing paid, then everything has a value of 0.

Figure 17.17 shows a variation on this. This score uses the ratio of the amount paid to the minimum payment. It has some nice features. Perfect revolvers now have a score of 1, because their payment is equal to the minimum payment. Someone who does not use the card has a score of 0. Transactors and convenience users both have scores higher than 1, but it is hard to differentiate between them.

This section has shown several different ways of measuring the behavior of a customer. All of these are based on the important variables relevant to the customer and measurements taken over several months. Different measures are more valuable for identifying various aspects of behavior.

The Ideal Convenience User

The measures in the previous section focused on the extremes of customer behavior, as typified by revolvers and transactors. Convenience users were just assumed to be somewhere in the middle. Is there a way to develop a score that is optimized for the ideal convenience user?

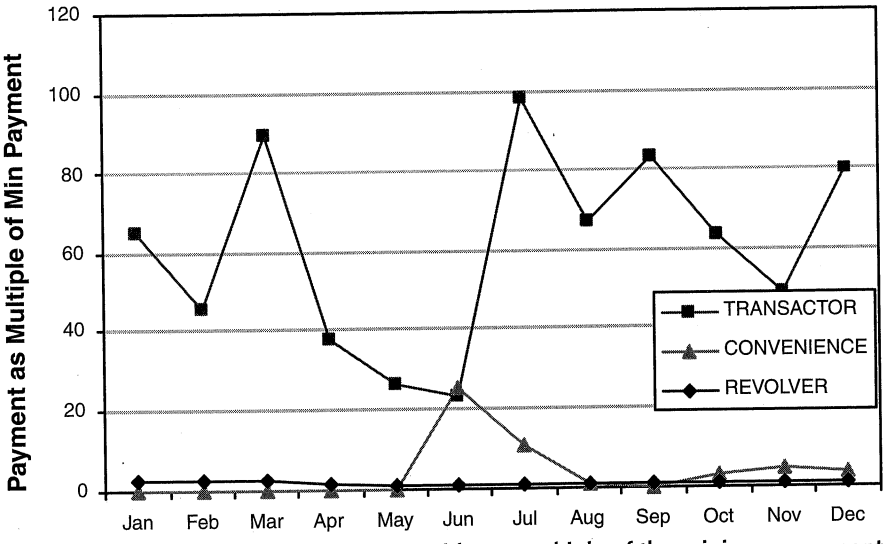


Figure 17.17 Comparing the amount paid as a multiple of the minimum payment shows distinct curves for transactors, revolvers, and convenience users.

First, let's define the ideal convenience user. This is someone who, twice a year, charges up to his or her credit line and then pays the balance off over 4 months. There are few, if any, additional charges during the other 10 months of the year. Table 17.7 illustrates the monthly balances for two convenience users as a ratio of their credit lines.

This table also illustrates one of the main challenges in the definition of convenience users. The values describing their behavior have no relationship to each other in any given month. They are out of phase. In fact, there is a fundamental difference between convenience users on the one hand and transactors and revolvers on the other. Knowing that someone is a transactor exactly describes their behavior in any given month—they pay off the balance. Knowing that someone is a convenience user is less helpful. In any given month, they may be paying nothing, paying off everything, or making a partial payment.

Table 17.7 Monthly Balances of Two Convenience Users Expressed as a Percentage of Their Credit Lines

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	NOV	DEC
Conv1	80%	60%	40%	20%	0%	0%	0%	60%	30%	15%	70%
Conv2	0%	0%	83%	50%	17%	0%	67%	50%	17%	0%	0%

Does this mean that it is not possible to develop a measure to identify convenience users? Not at all. The solution is to sort the 12 months of data by the balance ratio and to create the convenience-user measure using the sorted data.

Figure 17.18 illustrates this process. It shows the two convenience users, along with the profile of the ideal convenience user. Here, the data is sorted, with the largest values occurring first. For the first convenience user, month 1 refers to January. For the second, it refers to March.

Now, using the same idea of taking the area between the ideal and the actual produces a score that measures how close a convenience user is to the ideal. Notice that revolvers would have outstanding balances near the maximum for all months. They would have high scores, indicating that they are far from the ideal convenience user. For convenience users, the scores are much smaller.

This case study has shown several different ways of segmenting customers. All make use of derived variables to describe customer behavior. Often, it is possible to describe a particular behavior and then to create a score that measures how each customer's behavior compares to the ideal.

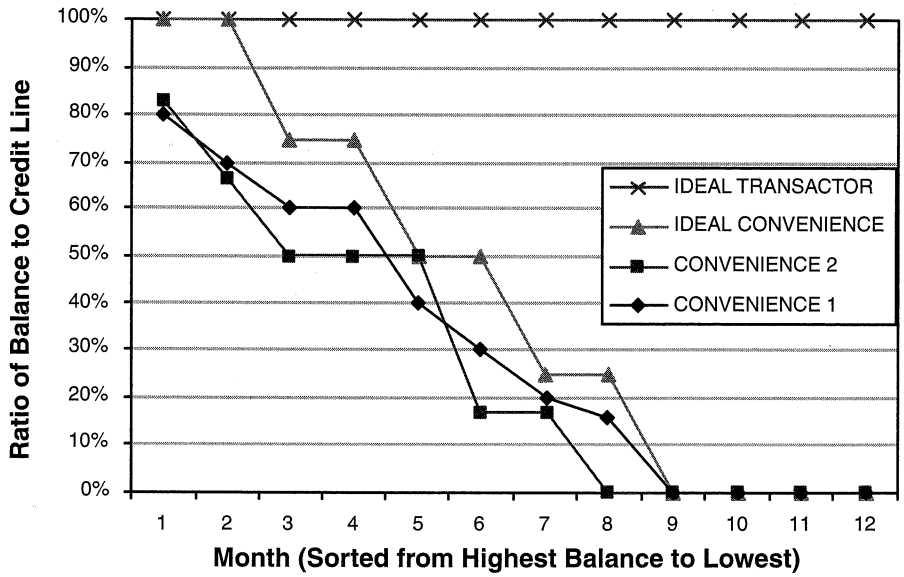


Figure 17.18 Comparison of two convenience users to the ideal, by sorting the months by the balance ratio.

The Dark Side of Data

Working with data is a critical part of the data mining process. What does the data mean? There are many ways to answer this question—through written documents, in database schemas, in file layouts, through metadata systems, and, not least, via the database administrators and systems analysis who know what is really going on. No matter how good the documentation, the real story lies in the data.

There is a misconception that data mining requires perfect data. In the world of business analysis, the perfect is definitely the enemy of the sufficiently good. For one thing, exploring data and building models highlights data issues that are otherwise unknown. Starting the process with available data may not result in the best models, but it does start a process that can improve over time. For another thing, waiting for perfect data is often a way of delaying a project so that nothing gets done.

This section covers some of the important issues that make working with data a sometimes painful process.

Missing Values

Missing values refer to data that should be there but is not. In many cases, missing values are represented as NULLs in the data source, making it easy to identify them. However, be careful: NULL is sometimes an acceptable value. In this case, we say that the value is empty rather than missing, although the two look the same in source data. For instance, the stop code of an account might be NULL, indicating that the account is still active. This information, which indicates whether data is censored or not, is critical for survival analysis.

Another time when NULL is an acceptable value is when working with overlay data describing demographics and other characteristics of customers and prospects. In this case, NULL often has one of two meanings:

- There is not enough evidence to indicate whether the field is true for the individual. For instance, lack of subscriptions to golfing magazines suggests the person is not a golfer, but does not prove it.
- There is no matching record for the individual in the overlay data.

TIP When working with ovelay data, it is useful to replace NULLs with alternative values, one meaning that the record does not match and the other meaning that the value is unknown.

It is worth distinguishing between these situations. One way is to separate the data where the records do not match, creating two different model sets. The other is to replace the NULL values with alternative values, indicating whether the failure to match is at the record level or the field level.

Because customer signatures use so much aggregated data, they often contain "0" for various features. So, missing data in the customer signatures is not the most significant issue for the algorithms. However, this can be taken too far. Consider a customer signature that has 12 months of billing data. Customers who started in the past 12 months have missing data for the earlier months. In this case, replacing the missing data with some arbitrary value is not a good idea. The best thing is to split the model set into two pieces—those with 12 months of tenure and those who are more recent.

When missing data is a problem, it is important to find its cause. For instance, one database we encountered had missing data for customers' start dates. With further investigation, it turned out that these were all customers who had started and ended their relationship prior to March 1999. Subsequent use of this data source focused on either customers who started after this date or who were active on this date. In another case, a transaction table was missing a particular type of transaction before a certain date. During the creation of the data warehouse, different transactions were implemented at different times. Only carefully looking at crosstabulations of transaction types by time made it clear that one type was implemented much later than the rest.

In another case, the missing data in a data warehouse was just that—missing because the data warehouse had failed to load it properly. When there is such a clear cause, the database should be fixed, especially since misleading data is worse than no data at all.

One approach to dealing with missing data is to try to fill in the values—for example, with the average value or the most common value. Either of these substitutions changes the distribution of the variable and may lead to poor models. A more clever variation of this approach is to try to calculate the value based on other fields, using a technique such as regression or neural networks. We discourage such an approach as well, unless absolutely necessary, since the field no longer means what it is supposed to mean.

WARNING **One of the worst ways to handle missing values is to replace them with some "special" value such as 9999 or –1 that is supposed to stick out due to its unreasonableness. Data mining algorithms will happily use these values as if they were real, leading to incorrect results.**

Usually data is missing for systematic reasons, as in the new customers scenario mentioned earlier. A better approach is to split the model set into parts, eliminating the missing fields from one data set. Although one data set has more fields, neither will have missing values.

It is also important to understand whether the data is going to be missing in the future. Sometimes the right approach is to build models on records that have complete data (and hope that these records are sufficiently representative of all records) and to have someone fix the data sources, eliminating this headache in the future.

Dirty Data

Dirty data refers to fields that contain values that might look correct, but are not. These can often be identified because such values are outliers. For instance, once upon a time, a company thought that it was very important for their call-center reps to collect the birth dates of customers. They thought it was so important that the input field on the screen was mandatory. When they looked at the data, they were surprised to see that more than 5 percent of their customers were born in 1911; and not just in 1911, but on November 11th. It turns out that not all customers wanted to share their birth date, so the call-center reps quickly learned that typing six "1"s was the quickest way to fill the field (the day, month, and year eachtook two characters). The result: many customers with the exact same birthday.

The attempt to collect accurate data often runs into conflict with efforts to manage the business. Many stores offer discounts to customers who have membership cards. What happens when a customer does not have a card? The business rules probably say "no discount." What may really happen is that a store employee may enter a default number, so that customer can still qualify. This friendly gesture leads to certain member numbers appearing to have exceptionally high transaction volumes.

One company found several customers in Elizabeth, NJ with the zip code 07209. Unfortunately, the zip code does not exist, which was discovered when analyzing the data by zip code and appending zip code information. The error had not been discovered earlier because the post office can often figure out how to route incorrectly addressed mail. Such errors can be fixed by using software or an outside service bureau to standardize the address data.

What looks like dirty data might actually provide insight into the business. A telephone number, for instance, should consist only of numbers. The billing system for one regional telephone company stored the number as a string (this is quite common actually). The surprise was several hundred "telephone numbers" that included alphabetic characters. Several weeks (!) after being asked about this, the systems group determined that these were essentially calling card numbers, not attached to a telephone line, that were used only for third-party billing services.

Another company used media codes to determine how customers were acquired. So, media codes starting with "W" indicated that customers came from the Web, "D" indicated response to direct mail, and so on. Additional characters in the code distinguished between particular banner ads and particular email campaigns. When looking at the data, it was surprising to discover Web customers starting as early as the 1980s. No, these were not bleeding-edge customers. It turned out that the coding scheme for media codes was created in October 1997. Earlier codes were essentially gibberish. The solution was to create a new channel for analysis, the "pre-1998" channel.

WARNING Wthe most pernicious data problem are the ones you don't know about. For this reason, data mining cannot be performed in a vacuum; input from business people and data analysts are critical for success.

All of these cases are examples where dirty data could be identified. The biggest problems in data mining, though, are the unknown ones. Sometimes, data problems are hidden by intervening systems. In particular, some data warehouse builders abhor missing data. So, in an effort to clean data, they may impute values. For instance, one company had more than half their loyal customers enrolling in a loyalty program in 1998. The program has been around longer, but the data was loaded into the data warehouse in 1998. Guess what? For the participants in the initial load, the data warehouse builders simply put in the current date, rather than the date when the customers actually enrolled.

The purpose of data mining is to find patterns in data, preferably interesting, actionable patterns. The most obvious patterns are based on how the business is run. Usually, the goal is to gain an understanding of customers more than an understanding of how the business is run. To do this, it is necessary to understand what was happening when the data was created.

Inconsistent Values

Once upon a time, computers were expensive, so companies did not have many of them. That time is long past, and there are now many systems for many different purposes. In fact, most companies have dozens or hundreds of systems, some on the operational side, some on the decision-support side. In such a world, it is inevitable that data in different systems does not always agree.

One reason that systems disagree is that they are referring to different things. Consider the start date for mobile telephone service. The order-entry system might consider this the date that customer signs up for the service. An operational system might consider it the date that the service is activated. The billing system might consider it the effective date of the first bill. A downstream decision-support system might have yet another definition. All of these dates should be close to each other. However, there are always exceptions. The best solution is to include all these dates, since they can all shed light on the business. For instance, when are there long delays between the time a customer signs up for the service and the time the service actually becomes effective? Is this related to churn? A more common solution is to choose one of the dates and call that the start date.

Another reason has to do with the good intentions of systems developers. For instance, a decision-support system might keep a current snapshot of customers, including a code for why the customer stopped. One code value might indicate that some customers stopped for nonpayment; other code values might represent other reasons—going to a competitor, not liking the service,

and so on. However, it is not uncommon for customers who have stopped voluntarily to not pay their last bill. In this data source, the actual stop code was simply overwritten. The longer ago that a customer stopped, greater the chance that the original stop reason was subsequently overwritten when the company determines—at a later time—that a balance is owed. The problem here is that one field is being used for two different things—the stop reason and nonpayment information. This is an example of poor data modeling that comes back to bite the analysts.

A problem that arises when using data warehouses involves the distinction between the initial loads and subsequent incremental loads. Often, the initial load is not as rich in information, so there are gaps going back in time. For instance, the start date may be correct, but there is no product or billing plan for that date. Every source of data has its peculiarities; the best advice is to get to know the data and ask lots of questions.

Computational Issues

Creating useful customer signatures requires considerable computational power. Fortunately, computers are up to the task. The question is more which system to use. There are several possibilities for doing the transformation work:

- Source system, typically in databases of some sort (either operational or decision support)
- Data extraction tools (used for populating data warehouses and data marts)
- Special-purpose code (such as SAS, SPSS, S-Plus, Perl)
- Data mining tools

Each of these has its own advantages and disadvantages.

Source Systems

Source systems are usually relational databases or mainframe systems. Often, these systems are highly restricted, because they have many users. Such source systems are not viable platforms for performing data transformations. Instead, data is dumped (usually as flat files) from these systems and manipulated elsewhere.

In other cases, the databases may be available for ad hoc query use. Such queries are useful for generating customer signatures because of the power of relational databases. In particular, databases make it possible to:

- Extract features from individual fields, even when these fields are dates and strings

- Combine multiple fields using arithmetic operations
- Look up values in reference tables
- Summarize transactional data

Relational databases are not particularly good at pivoting fields, although as shown earlier in this chapter, they can be used for that as well.

On the downside, expressing transformations in SQL can be cumbersome, to say the least, requiring considerable SQL expertise. The queries may extend for hundreds of lines, filled with subqueries, joins, and aggregations. Such queries are not particularly readable, except by whoever constructed them. These queries are also killer queries, although databases are becoming increasingly powerful and able to handle them. On the plus side, databases do take advantage of parallel hardware, a big advantage for transforming data.

Extraction Tools

Extraction tools (often called ETL tools for extract-transform-load) are generally used for loading data warehouses and data marts. In most companies, business users do not have ready access to these tools, and most of their functionality can be found in other tools. Extraction tools are generally on the expensive side because they are intended for large data warehousing projects.

In *Mastering Data Mining* (Wiley, 1999), we discuss a case study using a suite of tools from Ab Initio, Inc., a company that specializes in parallel data transformation software. This case study illustrates the power of such software when working on very large volumes of data, something to consider in an environment where such software might be available.

Special-Purpose Code

Coding is the tried-and-true way of implementing data transformations. The choice of tool is really based on what the programmer is most familiar with and what tools are available. For the transformations needed for a customer signature, the main statistical tools all have sufficient functionality.

One downside of using special-purpose code is that it adds an extra layer to the data transformation process. Data must still be extracted from source systems (one possible source of error) and then passed through code (another source of error). It is a good idea to write code that is well documented and reusable.

Data Mining Tools

Increasingly, data mining tools have the ability to transform data within the tool. Most tools have the ability to extract features from fields and to combine multiple fields in a row, although the support for non-numeric data types

varies from tool to tool and release to release. Some tools also support summarizations within the customer signature, such as binning variables (where the binning breakpoints are determined first by looking at the entire set of data) and standardization.

However, data mining tools are generally weak on looking up values and doing aggregations. For this reason, the customer signature is almost always created elsewhere and then loaded into the tool. Tools from leading vendors allow the embedding of programming code inside the tool and access to databases using SQL. Using these features is a good idea because such features reduce the number of things to keep track of when transforming data.

Lessons Learned

Data is the gasoline that powers data mining. The goal of data preparation is to provide a clean fuel, so the analytic engines work as efficiently as possible. For most algorithms, the best input takes the form of customer signatures, a single row of data with fields describing various aspects of the customer. Many of these fields are input fields, a few are targets used for predictive modeling.

Unfortunately, customer signatures are not the way data is found in available systems—and for good reason, since the signatures change over time. In fact, they are constantly being built and rebuilt, with newer data and newer ideas on what constitutes useful information.

Source fields come in several different varieties, such as numbers, strings, and dates. However, the most useful values are usually those that are added in. Creating derived values may be as simple as taking the sum of two fields. Or, they may require much more sophisticated calculations on very large amounts of data. This is particularly true when trying to capture customer behavior over time, because time series, whether regular or irregular, must be summarized for the signature.

Data also suffers (and causes us to suffer along with it) from problems—missing values, incorrect values, and values from different sources that disagree. Once such problems are identified, it is possible to work around them. The biggest problems are the unknown ones—data that looks correct but is wrong for some reason.

Many data mining efforts have to use data that is less than perfect. As with old cars that spew blue smoke but still manage to chug along the street, these efforts produce results that are good enough. Like the vagabonds in Samuel Beckett's play *Waiting for Godot*, we can choose to wait until perfection arrives. That is the path to doing nothing; the better choice is to plow ahead, to learn, and to make incremental progress.