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.

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>Phone Number</th>
<th>Home Address</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>2910001010</td>
<td>1234567890</td>
<td>123 Main St.</td>
<td>TRUE</td>
</tr>
<tr>
<td>2910001020</td>
<td>9876543210</td>
<td>456 Elm St.</td>
<td>TRUE</td>
</tr>
<tr>
<td>2910001030</td>
<td>1112223333</td>
<td>789 Lake Dr.</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

These columns come from reference tables, so their values are repeated many times.

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

**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.

![This chart shows an almost-unary column. The column was created by binning telephone call durations into 10 equal-width bins. Almost all values, 9,988 out of 9,995, are in the first bin. If variable width bins had been chosen, then the resulting column would have been more useful.](image)

**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.
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.
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.
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,” “Palestine” [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.
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 overlay 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 each took 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** The 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.