The databases could be internal (e.g., past purchases made by customers) or external (credit ratings). While data mining deals with very large databases, usually the analysis to be done requires only thousands or tens of thousands of records.

- *Explore, clean, and preprocess the data.* This step involves verifying that the data are in reasonable condition. How should missing data be handled? Are the values in a reasonable range, given what you would expect for each variable? Are there obvious outliers? The data are reviewed graphically: for example, a matrix of scatterplots showing the relationship of each variable with every other variable. We also need to ensure consistency in the definitions of fields, units of measurement, time periods, and so on. In this step, new variables are also typically created from existing ones. For example, "duration" can be computed from start and end dates.
- *Reduce the data dimension, if necessary.* Dimension reduction can involve operations such as eliminating unneeded variables, transforming variables (e.g., turning "money spent" into "spent > $100" vs. "spent ≤ $100"), and creating new variables (e.g., a variable that records whether at least one of several products was purchased). Make sure that you know what each variable means and whether it is sensible to include it in the model.
- *Determine the data mining task.* (classification, prediction, clustering, etc.). This involves translating the general question or problem of Step 1 into a more specific data mining question.
- *Partition the data (for supervised tasks).* If the task is supervised (classification or prediction), randomly partition the dataset into three parts: training, validation, and test datasets.
- *Choose the data mining techniques to be used.* (regression, neural nets, hierarchical clustering, etc.).
- *Use algorithms to perform the task.* This is typically an iterative process—trying multiple variants, and often using multiple variants of the same algorithm (choosing different variables or settings within the algorithm). Where appropriate, feedback from the algorithm's performance on validation data is used to refine the settings.
- *Interpret the results of the algorithms.* This involves making a choice as to the best algorithm to deploy, and where possible, testing the final choice on the test data to get an idea as to how well it will perform. (Recall that each algorithm may also be tested on the validation data for tuning purposes; in this way, the validation data become a part of the fitting process and are likely to underestimate the error in the deployment of the model that is finally chosen.)
- *Deploy the model.* This step involves integrating the model into operational systems and running it on real records to produce decisions or actions. For example, the model might be applied to a purchased list of possible customers, and the action might be "include in the mailing if the predicted amount of purchase is > $10." A key step here is "scoring" the new records, or using the chosen model to predict the outcome value ("score") for each new record.

The foregoing steps encompass the steps in SEMMA, a methodology developed by the software company SAS:

Sample	Take a sample from the dataset; partition into training, validation, and test datasets.
Explore	Examine the dataset statistically and graphically.
Modify	Transform the variables and impute missing values.
Model	Fit predictive models (e.g., regression tree, neural network).
Assess	Compare models using a validation dataset.

IBM SPSS Modeler (previously SPSS-Clementine) has a similar methodology, termed CRISP-DM (CRoss-Industry Standard Process for Data Mining). All these frameworks include the same main steps involved in predictive modeling.

2.4 Preliminary Steps

Organization of Datasets

Datasets are nearly always constructed and displayed so that variables are in columns and records are in rows. We will illustrate this with home values in West Roxbury, Boston, in 2014. 14 variables are recorded for over 5000 homes. The spreadsheet is organized so that each row represents a home—the first home's assessed value was $344,200, its tax was $4430, its size was 9965 ft^2, it was built in 1880, and so on. In supervised learning situations, one of these variables will be the outcome variable, typically listed in the first or last column (in this case it is TOTAL VALUE, in the first column).

Predicting Home Values in the West Roxbury Neighborhood

The Internet has revolutionized the real estate industry. Realtors now list houses and their prices on the web, and estimates of house and condominium prices have become widely available, even for units not on the market. At this time of writing, Zillow (www.zillow.com) is the most popular online real estate information site in the United States[1], and in 2014 they purchased their major rival, Trulia. By 2015, Zillow had become the dominant platform for checking house prices and, as such, the dominant online advertising venue for realtors. What used to be a comfortable 6% commission structure for realtors, affording them a handsome surplus (and an oversupply of realtors), was being rapidly eroded by an increasing need to pay for advertising on Zillow. (This, in fact, is the key to Zillow's business model—redirecting the 6% commission away from realtors and to itself.)

Zillow gets much of the data for its "Zestimates" of home values directly from publicly available city housing data, used to estimate property values for tax assessment. A competitor seeking to get into the market would likely take the same approach. So might realtors seeking to develop an alternative to Zillow.

A simple approach would be a naive, model-less method—just use the assessed values as determined by the city. Those values, however, do not necessarily include all properties, and they might not include changes warranted by remodeling, additions, etc. Moreover, the assessment methods used by cities may not be transparent or always reflect true market values. However, the city property data can be used as a starting point to build a model, to which additional data (such as that collected by large realtors) can be added later.

Let's look at how Boston property assessment data, available from the city of Boston, might be used to predict home values. The data in *WestRoxbury.csv* includes information on single family owner-occupied homes in West Roxbury, a neighborhood in southwest Boston, MA, in 2014. The data include values for various predictor variables, and for an outcome-assessed home value ("total value"). This dataset has 14 variables and includes 5802 homes. A sample of the data[2] is shown in Table 2.2, and the "data dictionary" describing each variable is in Table 2.1.[3]

Table 2.1 Description of Variables in West Roxbury (Boston) Home Value Dataset

TOTAL VALUE	Total assessed value for property, in thousands of USD
TAX	Tax bill amount based on total assessed value multiplied by the tax rate, in USD
LOT SQ FT	Total lot size of parcel in square feet
YR BUILT	Year the property was built
GROSS AREA	Gross floor area
LIVING AREA	Total living area for residential properties (ft^2)
FLOORS	Number of floors
ROOMS	Total number of rooms
BEDROOMS	Total number of bedrooms
FULL BATH	Total number of full baths
HALF BATH	Total number of half baths
KITCHEN	Total number of kitchens
FIREPLACE	Total number of fireplaces
REMODEL	When the house was remodeled (Recent/Old/None)

Table 2.2 First 10 Records in the West Roxbury home values dataset

TOTAL VALUE	TAX	LOT SQ FT	YR BUILT	GROSS AREA	LIVING AREA	FLOORS	ROOMS	BED ROOMS	FULL BATH	HALF BATH	KIT CHEN	FIRE PLACE	RE
344.2	4330	9965	1880	2436	1352	2	6	3	1	1	1	0	
412.6	5190	6590	1945	3108	1976	2	10	4	2	1	1	0	I
330.1	4152	7500	1890	2294	1371	2	8	4	1	1	1	0	
498.6	6272	13,773	1957	5032	2608	1	9	5	1	1	1	1	

331.5	4170	5000	1910	2370	1438	2	7	3	2	0	1	0	
337.4	4244	5142	1950	2124	1060	1	6	3	1	0	1	1	
359.4	4521	5000	1954	3220	1916	2	7	3	1	1	1	0	
320.4	4030	10,000	1950	2208	1200	1	6	3	1	0	1	0	
333.5	4195	6835	1958	2582	1092	1	5	3	1	0	1	1	I
409.4	5150	5093	1900	4818	2992	2	8	4	2	0	1	0	

As we saw earlier, below the header row, each row in the data represents a home. For example, the first home was assessed at a total value of $344.2 thousand (TOTAL VALUE). Its tax bill was $4330. It has a lot size of 9965 square feet (ft^2), was built in the year 1880, has two floors, six rooms, and so on.

Loading and Looking at the Data in R

To load data into R, we will typically want to have the data available as a csv (comma separated values) file. If the data are in an xls (or xlsx) file, we can save that same file in Excel as a csv file: go to File > Save as > Save as type: CSV (Comma delimited) (*.csv) > Save.

Note: When dealing with .csv files in Excel, beware of two things:

- Opening a .csv file in Excel strips off leading 0's, which corrupts zipcode data.
- Saving a .csv file in Excel saves only the digits that are displayed; if you need precision to a certain number of decimals, you need to ensure they are displayed before saving.

Once we have R and RStudio installed on our machine and the *West Roxbury.csv* file saved as a csv file, we can run the code in Table 2.3 to load the data into R.

Table 2.3 Working with Files in R

```
To start, open RStudio, go to File > New File > R Script.  It opens a new tab.
Then save your Untitled1.R file into the directory where your csv is saved. Give it the name
WestRoxbury.R.
From the Menu Bar, go to Session > Set Working Directory >
To Source File Location; This sets the working directory as the place where both the R file and
csv file are saved.
```


code for loading and creating subsets from the data

```
housing.df <- read.csv("WestRoxbury.csv", header = TRUE)  # load data
dim(housing.df)  # find the dimension of data frame
head(housing.df)  # show the first six rows
View(housing.df)  # show all the data in a new tab

# Practice showing different subsets of the data
housing.df[1:10, 1]  # show the first 10 rows of the first column only
housing.df[1:10, ]  # show the first 10 rows of each of the columns
housing.df[5, 1:10]  # show the fifth row of the first 10 columns
housing.df[5, c(1:2, 4, 8:10)]  # show the fifth row of some columns
housing.df[, 1]  # show the whole first column
housing.df$TOTAL_VALUE  # a different way to show the whole first column
housing.df$TOTAL_VALUE[1:10]  # show the first 10 rows of the first column
length(housing.df$TOTAL_VALUE)  # find the length of the first column
mean(housing.df$TOTAL_VALUE)  # find the mean of the first column
summary(housing.df)  # find summary statistics for each column
```

Data from a csv file is stored in R as a data frame (e.g., *housing.df*). If our csv file has column headers, these headers get automatically stored as the column names of our data. A data frame is the fundamental object almost all analyses begin with in R. A data frame has rows and columns. The rows are the observations for each case (e.g., house), and the columns are the variables of interest (e.g., TOTAL VALUE, TAX). The code in Table 2.3 walks you through some basic steps you will want to perform prior to doing any analysis: finding the size and dimension of your data (number

basic steps you will want to perform prior to doing any analysis: finding the size and dimension of your data (number of rows and columns), viewing all the data, displaying only selected rows and columns, and computing summary statistics for variables of interest. Note that comments are preceded with the # symbol.

Sampling from a Database

Typically, we perform data mining on less than the complete database. Data mining algorithms will have varying limitations on what they can handle in terms of the numbers of records and variables, limitations that may be specific to computing power and capacity as well as software limitations. Even within those limits, many algorithms will execute faster with smaller samples.

Accurate models can often be built with as few as several thousand records. Hence, we will want to sample a subset of records for model building. Table 2.4 provides code for sampling in R.

Table 2.4 Sampling in R

code for sampling and over/under-sampling

```
# random sample of 5 observations
s <- sample(row.names(housing.df), 5)
housing.df[s,]

# oversample houses with over 10 rooms
s <- sample(row.names(housing.df), 5, prob = ifelse(housing.df$ROOMS>10, 0.9, 0.01))
housing.df[s,]
```

Oversampling Rare Events in Classification Tasks

If the event we are interested in classifying is rare, for example, customers purchasing a product in response to a mailing, or fraudulent credit card transactions, sampling a random subset of records may yield so few events (e.g., purchases) that we have little information on them. We would end up with lots of data on nonpurchasers and non-fraudulent transactions but little on which to base a model that distinguishes purchasers from nonpurchasers or fraudulent from non-fraudulent. In such cases, we would want our sampling procedure to overweight the rare class (purchasers or frauds) relative to the majority class (nonpurchasers, non-frauds) so that our sample would end up with a healthy complement of purchasers or frauds.

Assuring an adequate number of responder or "success" cases to train the model is just part of the picture. A more important factor is the costs of misclassification. Whenever the response rate is extremely low, we are likely to attach more importance to identifying a responder than to identifying a non-responder. In direct-response advertising (whether by traditional mail, e-mail, or web advertising), we may encounter only one or two responders for every hundred records—the value of finding such a customer far outweighs the costs of reaching him or her. In trying to identify fraudulent transactions, or customers unlikely to repay debt, the costs of failing to find the fraud or the nonpaying customer are likely to exceed the cost of more detailed review of a legitimate transaction or customer.

If the costs of failing to locate responders are comparable to the costs of misidentifying responders as non-responders, our models would usually achieve highest overall accuracy if they identified everyone as a non-responder (or almost everyone, if it is easy to identify a few responders without catching many nonresponders). In such a case, the misclassification rate is very low—equal to the rate of responders—but the model is of no value.