5 Data Exploration and Preparation

5.1 INTRODUCTION

Well-prepared business analysts need to know how to use data to derive business insights and to improve business decisions in a variety of contexts. In one context, the analyst is called upon to *describe* the current situation. For example, the task might be to discover which products contributed most to profit over the past year. In another context, the analyst may be asked to *predict* the future situation. In this case, it may be useful to project sales by product line over the next year. In yet another context, the analyst might be called upon to *prescribe* specific actions the organization should take to achieve its goals. In this case the task might be to recommend a pricing strategy for each product line in order to increase profits. The first of these tasks, describing the past, is simply a matter of extracting and summarizing the appropriate data; it involves no modeling at all. At the other extreme, the pricing task may involve sophisticated management science modeling supported, of course, by relevant data. Whether the task at hand is primarily descriptive, predictive, or prescriptive, an analyst needs to command a range of data-related skills.

In this chapter, we focus on the basic skills needed to understand a data set, to explore individual variables and groups of variables for insights, and to prepare data for more complex analysis. Most of these skills relate to the descriptive tasks mentioned above. However, these same skills are necessary for any predictive or prescriptive task that involves data. Whether simple or complex, any management science model rests on a foundation of numerical assumptions. These numbers are often derived from raw data, which must be understood and explored before reliable estimates can be entered into the model. Furthermore, effective sensitivity analysis relies on the plausible range for a given variable, and this range can often be estimated from data.

In Chapter 6, we present a set of predictive models that rely heavily on data for their effectiveness. Some of these models are used for classification methods, which predict the category for an individual case. For example, we might want to predict which of a set of corporations will default on their bonds, which of our customers will respond to a promotion campaign, or which qualified voters will actually vote on Election Day. Other methods focus on numerical prediction. For example, we might want to predict the size of the next purchase from an existing customer, the value of the stock for a company going public, or the annual donations from members of a nonprofit organization. Both classification and prediction methods are based on past associations between outcomes and the variable to be predicted, and usually require large amounts of data to be effective. In Chapter 7, we present a special class of prediction methods called time-series methods. These rely on past values of the variables themselves to predict future values. Again, well-understood and well-prepared datasets are the keys to success in using these methods.

This chapter covers four main topics: database structure, types of variables, data exploration, and data preparation. We should also say a word about our coverage of software skills. Throughout this and the next two chapters, we use a variety of tools from Excel as well as the Excel add-in XLMiner. In a number of cases, a given data operation can be performed in several alternative ways in Excel and XLMiner. In most cases, we present the one method we think is most straightforward, flexible, and effective. When two alternative methods have competing advantages, we present both, but our intent is not to provide exhaustive coverage.

Finally, we offer an admonition to always maintain a skeptical attitude toward data. Datasets used for teaching purposes are usually carefully selected to be relevant to the problem at hand, free of bias, and without missing values or errors. Datasets encountered in the real world are another story: they may not be relevant to the problem, they may have been collected or processed in biased ways, and they often contain missing values and errors. Thus, when working with real data, it is crucial to explore the dataset carefully and prepare it for analysis, and to never lose sight of the fundamental assumption behind all data analysis: that the future will be enough like the past to make past data relevant.

5.2 DATABASE STRUCTURE

When the only data available to us are a few numbers or a few names, we can store them on slips of paper. But when the data grow to hundreds, thousands, or millions of data points, it is essential to store data in a structured format designed for easy access and manipulation. Most complex organizational data today are stored in relational databases and accessed using sophisticated database management systems, but data analysis is often carried out in spreadsheets. For our purposes, a database is a two-dimensional file (often called a flat file) consisting of rows and columns. Each row, or **record**, contains information on a single entity, which could be an individual customer, product, or company, depending on the database. Each column, or **field**, contains a specific type of information on each record, such as the name of a customer, the cost of a product, or the ticker symbol for a company. Depending on the context, records are also referred to as cases or instances, while fields may be referred to as variables, descriptors, or predictors. Most databases also contain a **data dictionary**, which documents each field in detail.

As examples, we'll work with three databases:

- Analgesics.xlsx, with data on retail sales of painkillers.
- Applicants.xlsx, with data on one year's pool of MBA applicants.
- Executives.xlsx, with data on executive compensation for a sample of companies.

Figure 5.1 shows a portion of the Analgesics database. This database contains 7,517 records, each of which describes sales at one of six different stores for a particular painkiller during a one-week period. The database has eight fields, starting with ID number and ending with SALES. The data dictionary describes the fields as follows:

Field Name	*Description*
ID	Record number
ITEM	Item number
UPC	Uniform Product Code
DESCRIPTION	Description
SIZE	Items per container
STORE	Store number
WEEK	Week number
SALES	Sales volume in cases

We might use this database to answer such questions as the following:

- What were the market shares of the various brands?
- What were the weekly sales volumes at the different stores?

We use the past tense here to emphasize that although we are interested in projecting into the future, the data actually tell us something about the past.

Figure 5.2 shows a portion of the Applicants database. This database contains 2,917 records, each of which describes an applicant to an MBA program. The database

FIGURE 5.1 First Portion of the Analgesics Database*

	A	B	C	D	E	F	G	H
1	ID	ITEM	UPC	DESCRIPTION	SIZE	STORE	WEEK	SALES
2	1	6122741	2586610502	ALEVE CAPLETS	24 CT	101	383	0
3	2	6122741	2586610502	ALEVE CAPLETS	24 CT	101	384	0
4	3	6122741	2586610502	ALEVE CAPLETS	24 CT	101	385	0
5	4	6122741	2586610502	ALEVE CAPLETS	24 CT	101	386	0
6	5	6122741	2586610502	ALEVE CAPLETS	24 CT	101	387	0
7	6	6122741	2586610502	ALEVE CAPLETS	24 CT	101	388	0
8	7	6122741	2586610502	ALEVE CAPLETS	24 CT	101	389	0
9	8	6122741	2586610502	ALEVE CAPLETS	24 CT	101	390	0
10	9	6122741	2586610502	ALEVE CAPLETS	24 CT	101	391	0
11	10	6122741	2586610502	ALEVE CAPLETS	24 CT	101	392	0
12	11	6122741	2586610502	ALEVE CAPLETS	24 CT	103	383	0
13	12	6122741	2586610502	ALEVE CAPLETS	24 CT	103	384	0
14	13	6122741	2586610502	ALEVE CAPLETS	24 CT	103	385	0
15	14	6122741	2586610502	ALEVE CAPLETS	24 CT	103	386	0
16	15	6122741	2586610502	ALEVE CAPLETS	24 CT	103	387	0
17	16	6122741	2586610502	ALEVE CAPLETS	24 CT	103	388	0
18	17	6122741	2586610502	ALEVE CAPLETS	24 CT	103	389	0
19	18	6122741	2586610502	ALEVE CAPLETS	24 CT	103	390	0
20	19	6122741	2586610502	ALEVE CAPLETS	24 CT	103	391	0
21	20	6122741	2586610502	ALEVE CAPLETS	24 CT	103	392	0
22	21	6122751	2586610504	ALEVE CAPLETS	50 CT	102	383	0
23	22	6122751	2586610504	ALEVE CAPLETS	50 CT	102	384	0
24	23	6122751	2586610504	ALEVE CAPLETS	50 CT	102	385	0
25	24	6122751	2586610504	ALEVE CAPLETS	50 CT	102	386	0
26	25	6122751	2586610504	ALEVE CAPLETS	50 CT	102	387	0
27	26	6122751	2586610504	ALEVE CAPLETS	50 CT	102	388	0
28	27	6122751	2586610504	ALEVE CAPLETS	50 CT	102	389	0
29	28	6122751	2586610504	ALEVE CAPLETS	50 CT	102	390	0
30	29	6122751	2586610504	ALEVE CAPLETS	50 CT	102	391	0

5.1

FIGURE 5.2 First Portion of the Applicants Database

	A	B	C	D	E	F	G	H	I
1	ID	ROUND	AGE	SEX	CITZ CODE	1ST CONTACT	JOB MONTHS	INDUSTRY	INDUSTRY DESC.
2	1	1	30	M	U	Email	84	260	Finan Serv-Diversified
3	2	1	29	M	U	Phone call	86	370	Government
4	3	1	27	M	U	Phone call	40	260	Finan Serv-Diversified
5	4	1	30	M	U	Email	48		
6	5	1	32	M	U	Home Page	85	290	Finan Serv-Invest Mgt/Research
7	6	1	26	M	U	Home Page	32	280	Finan Serv-Invest Bk/Brokerage
8	7	1	29	M	U	Phone call	60	10	Accounting
9	8	1	27	F	U		44	220	Entertainment/Leisure/Media
10	9	1	29	M	U	Phone call	66	40	Agribusiness
11	10	1	31	M	U	Letter	78	420	Nonprofit
12	11	1	27	M	U	Phone call	24	460	Retail
13	12	1	27	M	U	Phone call	51	370	Government
14	13	1	28	M	U		26	230	Environmental Services
15	14	1	30	M	U	Phone call	72	20	Advertising/Marketing Services
16	15	1	27	F	U	Phone call	48	420	Nonprofit
17	16	1	25	M	U	Home Page	31	260	Finan Serv-Diversified
18	17	1	24	F	U	Phone call	24	210	Energy/Utilities
19	18	1	28	M	U	Phone call	31	400	Law
20	19	1	27	M	N	Home Page	54	80	Construction
21	20	1	26	M	U	Letter	32	230	Environmental Services
22	21	1	24	M	U	World Wide Web	24	580	Other Services
23	22	1	27	M	U	Phone call	46	260	Finan Serv-Diversified
24	23	1	28	M	U	Home Page	50	160	Consumer Gds-Food/Beverage
25	24	1	28	M	U	Home Page	60	10	Accounting
26	25	1	29	M	U	Phone call	30	370	Government
27	26	1	27	M	U		48	120	Consulting-Strategy/Management
28	27	1	26	M	U	Phone call	38	260	Finan Serv-Diversified
29	28	1	28	F	U	Home Page	44	260	Finan Serv-Diversified
30	29	1	35	M	U	Phone call	146	220	Entertainment/Leisure/Media

5.1 5.2

* To download spreadsheets for this chapter, go to the Student Companion Site at www.wiley.com/college/powell.

contains 20 fields, each of which is described in more detail in the following data dictionary:

Field Name	*Description*
ID	Record number
ROUND	Round number (1–5) in which application was received
AGE	Age
SEX	Gender
CITZ CODE	U (U.S.) or N (non-U.S.)
1ST CONTACT	How first made contact with admissions
JOB MONTHS	Tenure in current (or most recent) job
INDUSTRY	Industry code number
INDUSTRY DESC.	Industry description
DECISION	Admission Committee decision
GMAT	Test score
PCT	Percentile
OLD	Old GMAT score, if previously taken
DEGREE	Previous degree
MJR	Major (abbreviated)
MAJOR	Major
DEG2	2nd previous degree
MJR2	2nd previous major
DEG3	3rd previous degree
MJR3	3rd previous major

We might use this database to answer such questions as the following:

- What proportion of the applicants had nonprofit work experience?
- What was the average GMAT score for accepted applicants?

Figure 5.3 shows a portion of the Executives database. This database contains 100 records, each of which describes the financial background of an executive. The database

FIGURE 5.3 First Portion of the Executives Database

	A	B	C	D	E	F	G	H	I
1	ID	EXECID	GENDER	SALARY	BONUS	OTHER	SHARES	CONAME	TICKER
2	108	01512	MALE	1000.000	2400.000	284.179	617.072	AMERICAN EXPRESS	AXP
3	151	01542	MALE	631.733	1124.421	0.000	48.300	AMGEN INC	AMGN
4	166	01060	MALE	475.400	0.000	25.600	194.583	ANDREW CORP	ANDW
5	217	01538	MALE	980.000	1557.710	14.377	161.445	ATLANTIC RICHFIELD CO	ARC
6	314	00111	MALE	750.000	825.000	294.770	183.031	BAXTER INTERNATIONAL INC	BAX
7	381	02633	MALE	1093.079	1900.800	0.000	21.136	BOEING CO	BA
8	447	01258	MALE	636.500	975.000	3.500	402.261	CIGNA CORP	CI
9	498	02821	MALE	700.000	1000.000	0.000	50.510	CHAMPION INTERNATIONAL CORP	CHA
10	516	00197	MALE	1350.000	1965.000	0.000		CHEVRON CORP	CHV
11	586	00778	MALE	1000.000	8732.474	448.577	17500.746	CITIGROUP INC	C
12	609	02079	MALE	750.000	861.000	0.000	1667.160	COMPUTER ASSOCIATES INTL INC	CA
13	767	02328	MALE	334.511	177.582	0.552	16.445	DOMINION RESOURCES INC	D
14	776	01504	MALE	850.000	1100.000	0.000	308.486	DOVER CORP	DOV
15	791	02272	MALE	425.000	226.250	0.000	2569.562	DOW JONES & CO INC	DJ
16	809	02238	MALE	723.000	376.834	0.000	172.819	DUN & BRADSTREET CORP	DNB
17	822	00315	MALE	771.018	463.200	0.000	224.861	EASTERN ENTERPRISES	EFU
18	921	01337	MALE	800.000	0.000	55.556		FIRST UNION CORP (N C)	FTU
19	988	02741	MALE	1050.000	1560.000	97.036	616.694	GENERAL ELECTRIC CO	GE
20	1016	00418	MALE	620.000	952.468	0.000	334.430	GENUINE PARTS CO	GPC
21	1063	02277	MALE	410.040	92.258	0.000	57.860	GRAINGER (W W) INC	GWW
22	1106	02424	MALE	312.500	177.470	24.000	91.066	HARRIS CORP	HRS
23	1121	02437	MALE	527.300	290.015	12.815		HASBRO INC	HAS
24	1153	01208	MALE	980.769	3000.000	104.984	34796.756	HOME DEPOT INC	HD
25	1211	02567	MALE	260.000	1120.000	0.000	3012.109	INTEL CORP	INTC
26	1214	02569	MALE	323.471	85.266	54.165	50.849	INTERGRAPH CORP	INGR
27	1220	01549	MALE	2000.000	7200.000	66.376	965.986	INTL BUSINESS MACHINES CORP	IBM
28	1307	01305	MALE	877.308	804.169	0.000	167.635	KNIGHT-RIDDER INC	KRI
29	1329	00555	MALE	1185.577	3331.968	0.000	46497.711	LIMITED INC	LTD
30	1351	02793	MALE	180.000	407.457	2.823	75.648	LONGS DRUG STORES INC	LDG

5.1 5.2 5.3

contains 19 fields, each of which is described in more detail in the following data dictionary:

Field Name	*Description*
ID	Record number
EXECID	Executive ID number
GENDER	Male or Female
SALARY	Annual salary
BONUS	End of year bonus
OTHER	Other compensation
SHARES	Shares owned
CONAME	Company name
TICKER	Ticker abbreviation
INDDESC	Industry description
STATE	State where company HQ located
ZIP	Zip code
AREA	Telephone area code
SALES	Annual sales
PRETAX	Pretax profits
ASSETS	Total assets
ROA	Return on assets
MKTVAL	Market value of stock
DIVYIELD	Dividend yield for the year

We might use this database to answer such questions as the following:

- What was the average salary among these executives?
- What proportion of compensation was due to annual bonuses?

5.3 TYPES OF DATA

Despite the almost infinite variety of data in databases, there are only a few common types of data. The most basic type is **nominal data**, which simply names the category of a record. The GENDER field in the Executives database is nominal data: it takes only the two values M (for Male) and F (for Female). Likewise, the CITZ CODE field in the Applicants database is nominal: it takes on the values U (for US) and N (for non-US). The DESCRIPTION field in the Analgesics database is also nominal: it takes over 90 distinct values, from ADVIL to TYLENOL X/STRGH LIQ. Note that nominal data are simply names; they have no natural order and cannot be treated as numbers. In some cases, nominal variables actually appear as numbers: for example, the categories Male and Female might be represented by 0 and 1 respectively. But these numbers should be understood as simply codes for the related words, not as numbers subject to addition or multiplication. An example is the STORE field in the Analgesics database, where the six stores are given the numbers 100 to 105; in this case the *average* store number makes no practical sense.

A second type of data, known as **ordinal data**, also identifies the category of a record, but in this case there is a natural order to the values. A simple example would be a sales variable in which the values were High, Medium, and Low. These values are clearly not numbers, but there is a definite order: we can say that High is greater than Medium, and Medium is greater than Low. Again, we could code these values using numbers, such as 3, 2, and 1. This coding preserves the order of the values, but we should avoid calculating differences or averages because these are meaningless. When we ask survey respondents to rank alternatives from Least Preferred (1) to Most Preferred (7), we are using an ordinal scale: Most Preferred is preferred to Least Preferred, but the difference of 6 is not meaningful.

Nominal variables and ordinal variables are both referred to as **categorical variables**. The other main type of data is **numerical data**. This category itself can be further divided into interval data and ratio data. **Interval data** conveys a sense of the differences between values. The Fahrenheit and Celsius temperature scales provide common examples. We can

say that 80 degrees Fahrenheit is 40 degrees warmer than 40 degrees Fahrenheit, so differences in degrees make sense. However, we cannot say that 80 degrees Fahrenheit is *twice* as warm as 40 degrees Fahrenheit. (These measurements correspond to 25 degrees Celsius and 5 degrees Celsius, so in Celsius the same temperature would appear to be five times as hot.) Interval data is based on a scale with an arbitrary zero point, which is why ratios are not meaningful. **Ratio data**, on the other hand, is based on a scale with a meaningful zero point, so differences, ratios, and any other algebraic calculation make sense. If a bank account increases from $1,000 to $2,000 not only has it increased by $1,000, but it also contains twice as much money. In our sample databases, the field SALARY in the Executives database and the field AGE in the Applicants database are both ratio variables.

5.4 DATA EXPLORATION

A database is a highly structured means for storing raw data, but it does not automatically reveal the patterns and insights that we seek. These must be ferreted out by a process of exploration. Although exploration is a creative process, and somewhat different from one situation to the next, some common approaches can be used in most situations. In this section, we describe a logical approach to the task of exploring a database, along with the software tools needed at each stage. This five-step process proceeds as follows:

- Understand the data.
- Organize and subset the database.
- Examine individual variables and their distributions.
- Calculate summary measures for individual variables.
- Examine relationships among variables.

5.4.1 Understand the Data

Remembering our earlier admonition to be skeptical of the data, the first step in data exploration is to understand the data we have: how are the fields defined, what types of data are represented, and what units are the data in?

For example, in scanning the Analgesics database we observe that the DESCRIPTION field is a nominal variable that defines both the brand (e.g., Aleve) and the type of analgesic (caplet, tablet, etc.). The SIZE field is somewhat ambiguous, so we consult the data dictionary to find that SIZE refers to "Items per container." Does this mean the number of bottles per case or the number of pills in a bottle? We may have to give this question additional thought. Similarly, the SALES field is ambiguous: this is defined as "Sales volume," but is that in units of bottles or cases? It's hard to know for sure; more investigation is needed before we draw final conclusions from this dataset.

Examining the Applicants database, we note that the AGE field is in integers; perhaps the actual age is rounded to the nearest year. The CITZ CODE field takes on the values U for US and N for non-US, but we might want to know more about how this value is assigned. Does it represent the country of birth, the country of citizenship, the country where the applicant currently lives, or perhaps the country where the most recent degree was received? The conclusions we draw might depend on how this variable is coded.

The Executives database gives financial information on 100 executives and their firms. The SALARY field seems clear enough, but what units apply? Most of the values are between 500 and 2,000, so we might infer that these are actual salaries between $500,000 and $2,000,000, and thus the data are in thousands of dollars. We can also infer that the BONUS and OTHER variables are in similar units, but we cannot know for sure.

It is vital not to skip over this first step in data exploration. Most databases contain ambiguities or hidden pitfalls, and it is important not to assume that our data are either defined in the manner we would like or measured in the units we would expect. Furthermore, our application of more sophisticated exploration techniques often depends on the data type and the coding method, so it is worthwhile to understand those issues at the start.

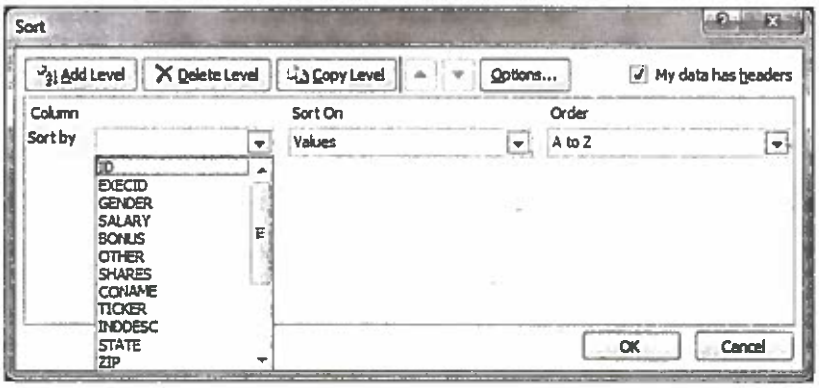


FIGURE 5.4 The Sort Window

5.4.2 Organize and Subset the Database

The next step after understanding the data is to organize the database and to create subsets of it. Two Excel tools are particularly helpful here: **Sort** and **Filter**. These can be found on the Home ribbon in the Editing group and on the Data ribbon in the Sort & Filter group. Sorting and Filtering a database can be particularly helpful when we have a question that applies to a subset of a database, such as those records with the highest values on a variable or those records that satisfy a complex set of restrictions. We give an example of sorting using the Executives database and another using the Applicants database.

Question: In the Executives database, do any duplicate records appear?

Suppose we wish to determine whether any executive (identified by the number in column B) is represented more than once in the sample. We begin by selecting the database. (Exploiting the range name, we do this by clicking the drop-down menu of range names, located directly above column A, and selecting the name Data.) Then, we choose Home▶Editing▶Sort & Filter▶Custom Sort, which opens the Sort window (Figure 5.4). This window has three drop-down windows: `Column`, `Sort On`, and `Order`. Here we sort by the EXECID column, sort on Values, and sort in the order A to Z. (This field was entered as text, so it must be sorted in alphabetical order.) When we click on OK, the sorting procedure is carried out, producing the results shown in Figure 5.5. Then, when we scan the ID numbers in the sorted list, we can see that no two adjacent entries match. Therefore, no executive appears twice in the list.

EXCEL TIP *The Sort Command*

When we sort a list with column headings, we check the box `My data has headers` in the Sort window. If we need to sort by rows instead of columns, we click on the `Options` button and choose `Sort left to right`. Although the Sort operation can be reversed by the Undo command, it is often a good idea to save the data to a new worksheet before sorting, so that the sorted data can be saved and analyzed separately. ■

Question: In the Applicants database, how does work experience vary among the applicants in successive rounds?

In the previous example, we chose one basis for sorting—the executive identification number. Sometimes we want to sort on a second or third criterion. When two sort criteria are specified, ties on the first criterion are broken by the second; when three criteria are specified, ties on the second criterion are broken by the third. To sort on additional criteria, click on `Add Level` in the Sort window. In the Applicants database, for example, we can sort first by Round, then by Industry Description, and finally by Job Months (see Figure 5.6 for the entries in the Sort window) to get a sense of how applications arrive over time from people in different industries and with different lengths of service in their current job. The first portion of the result appears in Figure 5.7. Looking deeper into this worksheet, we might observe, for example, that the applicants with relatively fewer months in advertising tend to apply in the first two rounds.

The Filtering capabilities in Excel allow us to probe a large database and extract a portion of it that deals with the specific records in which we are interested. We may simply

FIGURE 5.5 The Executives Database sorted by the EXECID field

	A	B	C	D	E	F	G	H	I
1	ID	EXECID	GENDER	SALARY	BONUS	OTHER	SHARES	CONAME	TICKER
2	2611	00006	MALE	683.462	412.768	0.000	207.828	ADC TELECOMMUNICATIONS INC	ADCT
3	4666	00029	MALE	750.000	1012.500	0.000	641.691	ALLTEL CORP	AT
4	2702	00074	MALE	875.000	1000.000	22.100	2234.402	ARROW ELECTRONICS INC	ARW
5	5562	00085	MALE	324.389	80.563	0.000	16632.050	ATMEL CORP	ATML
6	314	00111	MALE	750.000	825.000	294.770	183.031	BAXTER INTERNATIONAL INC	BAX
7	2766	00140	MALE	509.734	77.257	0.000	354.541	BOB EVANS FARMS	BOBE
8	2777	00152	MALE	700.000	317.665	0.000	39.928	KEYSPAN CORP	KSE
9	5588	00167	MALE	820.677	500.000	45.687	1208.415	CALLAWAY GOLF CO	ELY
10	516	00197	MALE	1350.000	1965.000	0.000		CHEVRON CORP	CHV
11	2962	00223	MALE	1000.000	0.000	0.000	255.198	CMS ENERGY CORP	CMS
12	2654	00246	MALE	1357.026	2162.508	91.721	1517.168	AFLAC INC	AFL
13	5865	00313	MALE	933.333	5554.350	300.034	6033.567	TRIARC COS INC -CL A	TRY
14	822	00315	MALE	771.018	463.200	0.000	224.861	EASTERN ENTERPRISES	EFU
15	9283	00393	MALE	1000.000	2750.000	143.698	643.966	FREEPRT MCMOR COP&GLD -CL B	FCX
16	5067	00398	MALE	205.000	0.000	56.891	9040.113	HUMANA INC	HUM
17	16384	00415	MALE	550.000	275.000	60.703	48.542	BARNES GROUP INC	B
18	1016	00418	MALE	620.000	952.468	0.000	334.430	GENUINE PARTS CO	GPC
19	10891	00474	MALE	410.004	400.000	0.000	1469.256	HORACE MANN EDUCATORS CORP	HMN
20	3296	00482	MALE	450.000	0.000	0.000	151.477	HUNT (JB) TRANSPRT SVCS INC	JBHT
21	3367	00522	MALE	750.000	0.000	46.067	740.528	KANSAS CITY SOUTHERN INDS	KSU
22	1329	00555	MALE	1185.577	3331.968	0.000	46497.711	LIMITED INC	LTD
23	4837	00563	MALE	1051.946	0.000	712.393	17308.998	LOEWS CORP	LTR
24	1459	00608	MALE	369.231	0.000	77.757	141.572	MCKESSON HBOC INC	MCK
25	1679	00705	MALE	730.769	0.000	159.776		OGDEN CORP	OG
26	5415	00731	MALE	800.000	10662.500	0.000	1185.531	PAINE WEBBER GROUP	PWJ
27	4903	00741	MALE	689.178	448.630	0.000	216.999	PPL CORP	PPL
28	9594	00749	MALE	990.000	698.465	128.496	1075.749	FORT JAMES CORP	FJ
29	586	00778	MALE	1000.000	8732.474	448.577	17500.746	CITIGROUP INC	C
30	10955	00791	MALE	773.085	975.000	0.000	4931.348	QUALCOMM INC	QCOM

5.1 5.2 5.3 5.4 5.5

want to view the extracted portion temporarily, or we may want to store it separately for further analysis. As an illustration, we use the Applicants database.

Question: In the Applicants database, what are the characteristics of the applicants from nonprofit organizations?

Suppose we want to view only the applicants who worked in nonprofit organizations. We first select the database and then choose Home▸Editing▸Sort & Filter▸Filter. This selection adds a list arrow to the title of each column. If we click on the Industry Description list arrow, we see a list of all the possible entries in this column, each one next to a checked box. The first step is to uncheck the box for `Select All`; this step removes all the checks. Then we can check `Nonprofit`, and we see the subset of the database that contains Nonprofit entries (Figure 5.8). Filtering does not actually *extract* any records: it merely *hides* rows that do not match the filter criteria. Thus, in Figure 5.8, we see that applicants from the Nonprofit sector appear in rows 11, 16, 87, 103, and so on. If we wish to extract the filtered records, we can copy and paste this subset to a different sheet.

FIGURE 5.6 Sorting by Three Fields

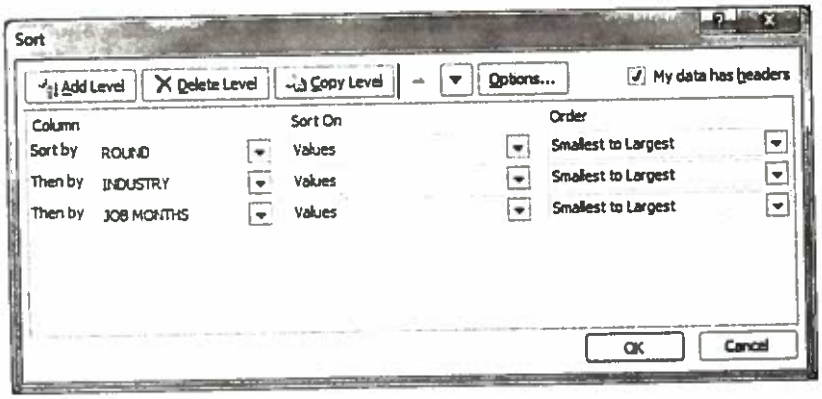


FIGURE 5.7 The Applicants Database Sorted by Round, Industry, and Job Months

	A	B	C	D	E	F	G	H	I
1	ID	ROUND	AGE	SEX	CITZ CODE	1ST CONTACT	JOB MONTHS	INDUSTRY	INDUSTRY DESC.
2	1637	1	25	M	U	Letter	22	10	Accounting
3	470	1	26	M	U	Letter	35	10	Accounting
4	91	1	26	F	U	Letter	36	10	Accounting
5	1694	1	27	M	U	Home Page	36	10	Accounting
6	1585	1	28	M	U	Phone call	37	10	Accounting
7	1638	1	27	M	U	Home Page	40	10	Accounting
8	392	1	24	F	U	Home Page	43	10	Accounting
9	1846	1	28	M	N	Phone call	44	10	Accounting
10	46	1	27	M	U	Phone call	45	10	Accounting
11	118	1	25	M	U	Phone call	48	10	Accounting
12	184	1	27	M	U	Test Score Tape (DNU)	48	10	Accounting
13	371	1	27	F	U	Phone call	48	10	Accounting
14	1793	1	27	F	U	Home Page	48	10	Accounting
15	1710	1	28	M	U	Phone call	49	10	Accounting
16	115	1	28	F	U	Phone call	56	10	Accounting
17	249	1	27	M	N	Home Page	56	10	Accounting
18	7	1	29	M	U	Phone call	60	10	Accounting
19	24	1	28	M	U	Home Page	60	10	Accounting
20	1906	1	29	M	U	Reapplicant	60	10	Accounting
21	1584	1	29	M	U	Phone call	66	10	Accounting
22	1626	1	29	M	U	Phone call	67	10	Accounting
23	504	1	29	M	U	Reapplicant	72	10	Accounting
24	1689	1	29	M	U	Home Page	75	10	Accounting
25	1518	1	29	M	N	Phone call	78	10	Accounting
26	1726	1	32	M	U	Email	82	10	Accounting
27	130	1	30	F	N	Phone call	100	10	Accounting
28	338	1	32	M	N	Email	102	10	Accounting
29	515	1	35	M	U	Reapplicant	120	10	Accounting
30	370	1	35	M	U	Home Page	144	10	Accounting

Alternatively, to restore the view of the entire database, we can either choose `Select All` using the list arrow again or choose Home▶Editing▶Sort & Filter▶Clear. While we are viewing a filtered subset of the database, the triangle corresponding to the list arrow we used displays the Filter symbol. This is a reminder that the information on the screen is filtered.

Question: In the Applicants database, what are the ages of the oldest applicants?

One of the options on the arrow list is `Number Filters`. This option allows us to use numerical criteria for filtering. (In a text field, the `Text Filters` option appears in the same place.) For example, to find the ten oldest applicants, we filter on Age, and from its arrow list, we select Number Filters▶Top 10 (see Figure 5.9). This window provides three

FIGURE 5.8 Filtering the Applicants Database to Highlight Nonprofit Backgrounds

	A	B	C	D	E	F	G	H	I
1	ID	ROUN	AGE	SEX	CITZ CO	1ST CONTACT	JOB MONTI	INDUSTI	INDUSTRY DESC.
11	10	1	31	M	U	Letter	78	420	Nonprofit
16	15	1	27	F	U	Phone call	48	420	Nonprofit
87	86	1	29	M	U	Test Score Tape (DNU)	4	420	Nonprofit
103	102	1	27	M	U	World Wide Web	30	420	Nonprofit
174	173	1	26	M	U	Home Page	37	420	Nonprofit
176	175	1	28	M	U	Phone call	60	420	Nonprofit
209	208	1	32	F	U	Home Page	108	420	Nonprofit
328	327	1	30	M	U	Phone call	62	420	Nonprofit
336	335	1	35	M	U	Phone call		420	Nonprofit
344	343	1	27	M	U	Phone call	44	420	Nonprofit
375	374	1	32	M	U	Home Page	88	420	Nonprofit
412	411	1	27	F	U	Phone call	48	420	Nonprofit
428	427	1	29	F	U	Phone call	62	420	Nonprofit
454	453	1	29	F	N	Test Score Tape (DNU)	56	420	Nonprofit
470	469	1	32	M	N	World Wide Web	61	420	Nonprofit
521	520	1	29	M	U	Home Page	72	420	Nonprofit
578	577	2	26	M	U	Test Score Tape (DNU)	25	420	Nonprofit
614	613	2	30	M	U	Home Page	6	420	Nonprofit
686	685	2	37	M	U	Letter	144	420	Nonprofit
748	747	2	28	F	U	Phone call	60	420	Nonprofit
825	824	2	29	M	N	Letter	58	420	Nonprofit
990	989	2	29	F	U	Phone call	67	420	Nonprofit
998	997	2	32	M	U	Reapplicant	98	420	Nonprofit
1120	1119	3	28	F	U	Phone call	35	420	Nonprofit
1172	1171	3	26	M	U	Home Page	36	420	Nonprofit
1184	1183	3	25	F	U	Phone call	38	420	Nonprofit
1202	1201	3	32	F	U	Email	84	420	Nonprofit
1229	1228	3	31	M	N	Home Page	15	420	Nonprofit
1263	1262	3	27	F	U	Phone call	68	420	Nonprofit

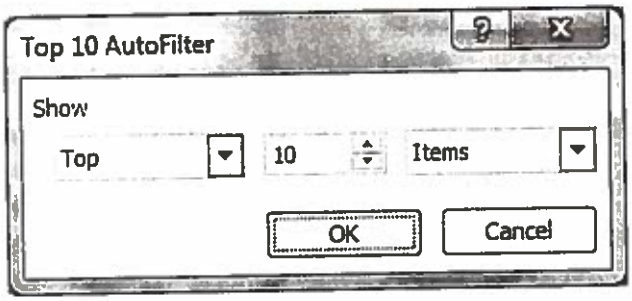


FIGURE 5.9 The Top 10 Autofilter Window

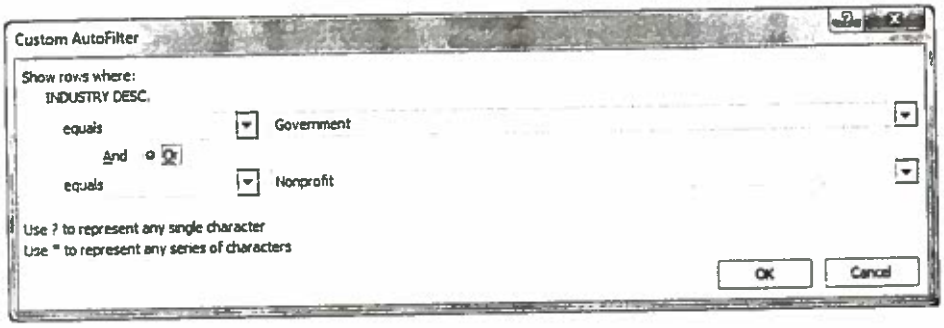


FIGURE 5.10 The Custom Autofilter Window

sets of options. The left-hand window allows us to choose from the `Top` or `Bottom` of the records; the right-hand window allows us to choose a fixed number of `Items` (records) or a `Percent` of all records, and the middle window allows us to select the actual number of records (e.g., 10 items or 10 percent).

Question: Isolate the applicants who worked in either the Nonprofit or Government sectors.

Another option under `Number Filters` or `Text Filters` is the Custom filter, which allows us to filter data using compound criteria. For example, to isolate the applicants who worked in *either* the Nonprofit *or* Government sectors, we select the Custom Filter option and click the Or button in the Custom AutoFilter window to set up the appropriate logical structure (see Figure 5.10).

Question: Isolate the applicants who worked in either the Nonprofit or Government sectors and had GMAT scores above 700.

For more complicated compound criteria, such as the one posed in this question, we use the Custom Filter on more than one field. We first filter the Industry Description field for Nonprofit and Government, and then we filter the GMAT field for scores above 700. The result of these filtering steps is shown in Figure 5.11.

FIGURE 5.11 The Applicants Database after Filtering

	A	B	C	D	E	F	G	H	I	J	K
1	ID	ROUN	AGE	SEX	CITZ CO	1ST CONTACT	JOB MONTI	INDUSTI	INDUSTRY DESC.	DECISION	GMAT
13	12	1	27	M	U	Phone call	51	370	Government	Cancel Enrollment	760
26	25	1	29	M	U	Phone call	30	370	Government	Deny	750
103	102	1	27	M	U	World Wide Web	30	420	Nonprofit	Deny	750
122	121	1	28	M	U	Letter		370	Government	Wait List Deny	720
209	208	1	32	F	U	Home Page	108	420	Nonprofit	Deny	740
263	262	1	32	M	U	Home Page	97	370	Government	Deny	770
323	323	1	32	M	U	Letter	76	370	Government	Wait List Deny	710
412	411	1	27	F	U	Phone call	48	420	Nonprofit	Wait List Deny	710
578	577	2	26	M	U	Test Score Tape (DNU)	26	420	Nonprofit	Deny	730
769	768	2	29	M	U	Email	62	370	Government	Deny	730
863	862	2	30	M	U	World Wide Web	84	370	Government	Deny	750
1128	1127	3	31	M	U	Phone call	78	370	Government	Deny	730
1184	1183	3	25	F	U	Phone call	38	420	Nonprofit	Deny	710
1229	1228	3	31	M	N	Home Page	15	420	Nonprofit	Deny	740
1251	1250	3	28	M	U	Home Page	61	370	Government	Deny	780
1259	1258	3	30	F	N	Letter	86	370	Government	Deny	720
1322	1321	4	28	M	N	World Wide Web	61	370	Government	Deny	730
1497	1496	1	30	F	U	Phone call	82	370	Government	Withdraw before Decision	770
1532	1531	1	26	F	U	Letter	50	370	Government	Withdraw after Decision	710
1561	1560	1	27	F	U	Phone call	48	420	Nonprofit	Deny	730
1574	1573	1	25	M	U	Email	38	420	Nonprofit	Deny	720
1809	1808	1	33	M	U	Phone call	113	420	Nonprofit	Withdraw after Decision	740
1654	1653	1	32	M	U	Home Page	98	370	Government	Enrolling	750
1809	1808	1	29	M	U	Phone call	24	370	Government	Deny	720
1834	1833	1	29	M	U	Home Page	63	370	Government	Wait List Deny	730
1844	1843	1	27	M	U	Home Page	46	420	Nonprofit	Withdraw after Decision	710
1846	1845	1	26	M	U	Home Page	38	420	Nonprofit	Deny	760
2101	2100	2	31	M	U	Phone call	48	420	Nonprofit	Wait List Deny	720
2113	2112	2	32	F	N	Test Score Tape (DNU)	89	370	Government	Deny	710

5.4.3 Examine Individual Variables and their Distribution

We now focus on the details of individual variables. For numerical variables, we typically want to know the range of the records from lowest to highest and the areas where most of the outcomes lie. For categorical variables with a small number of distinct values (such as Male/Female or High/Medium/Low), we typically want to know how many records fall into each category. Precise calculations require the use of Excel functions, which we cover in the next section. First, we use charts to generate visual summaries of the data.

Question: In the Applicants database, what are typical values for JOB MONTHS and what is the range from lowest to highest?

A common way to summarize the entire set of values for a numerical variable is to use a **histogram**. A histogram is created by sorting the values from low to high, grouping them into **bins** (for example, all values between 10 and 20), and displaying the number of values in each bin as a bar chart. To create a histogram in XLMiner we choose Explore▸Chart Wizard and the screen shown in Figure 5.12 appears.

This wizard offers us eight distinct types of charts. We choose Histogram, and in subsequent windows choose the Frequency option for the Y-axis and JOB MONTHS as the X-axis variable. When we select Finish, the chart in Figure 5.13 appears.

This chart shows that JOB MONTHS ranges from a low of 0 to a high around 280. The modal (most common) value appears to be about 40 months, and most records lie in a range from about 20 to 120 months. (Experiment with the Bins slider below the X-axis label to change the number of bins and the look of the chart.)

Question: Which are the most common decisions in the Applicants database?

DECISION is a categorical variable with values such as Admit, Deny, Cancel Enrollment, and Withdraw after Decision. To display the frequencies of these values, we use a Variable chart in the XLMiner Chart Wizard and select the DECISION variable. The result is shown in Figure 5.14.

This chart shows that the vast majority of these applicants received the Deny decision. Roughly 200 are Enrolling, while another 200 or so were given the decision Wait List Deny.

Histograms provide a useful visual overview of all the values of a variable. In many ways these charts represent a variable better than any single summary measure, such as the mean, median, minimum, or maximum. However, individual measures such as those are vital when we want to know specific facts about a variable.

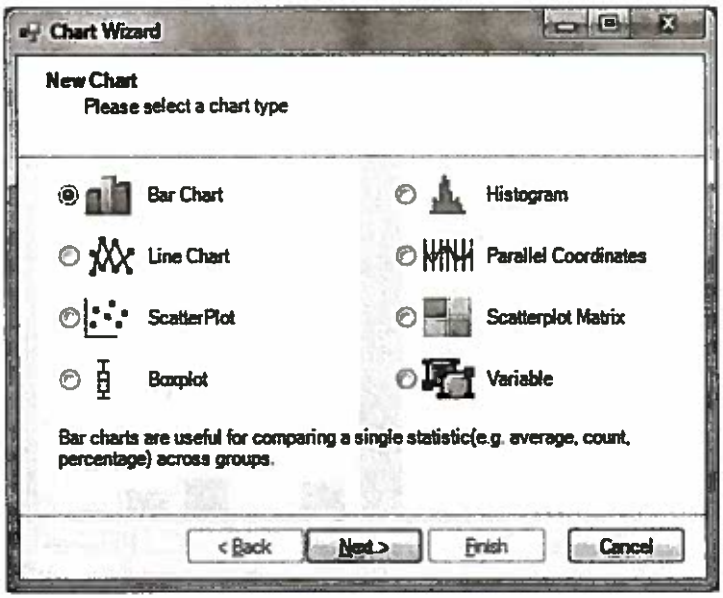
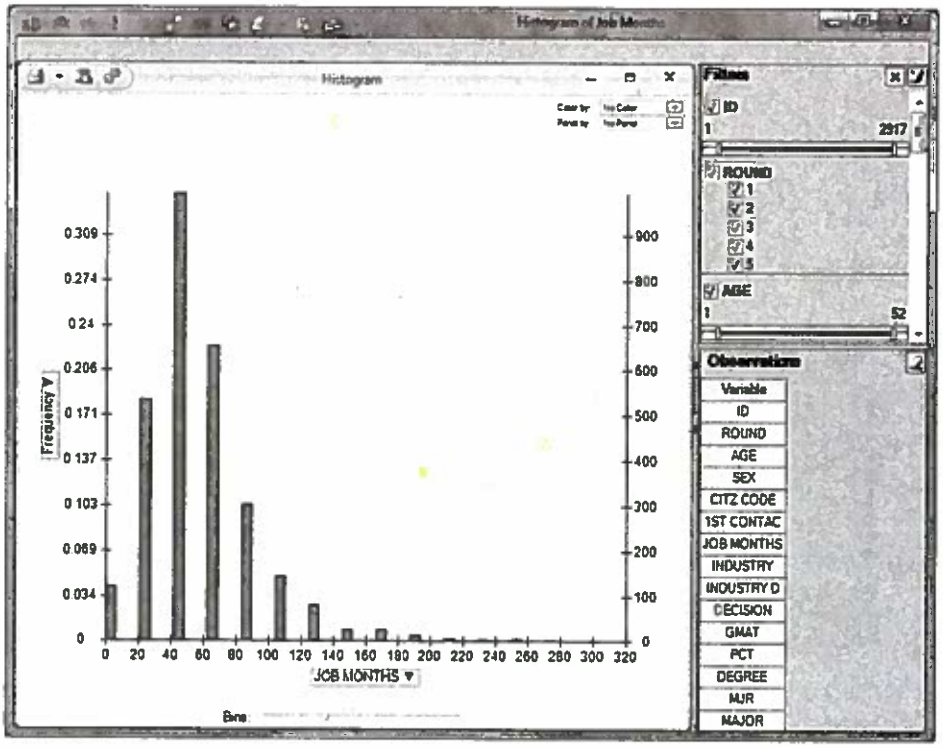


FIGURE 5.12 XLMiner Chart Wizard

FIGURE 5.13 Histogram of JOB MONTHS

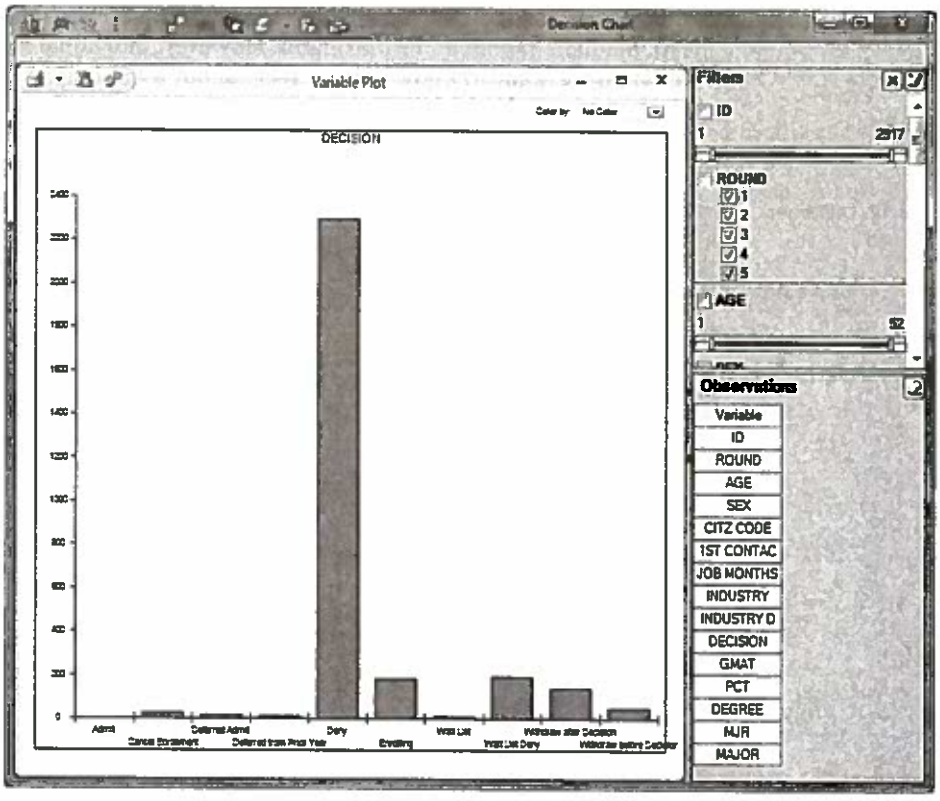


5.4.4 Calculate summary measures for individual variables

Excel provides a number of functions that are useful for investigating individual variables. Some functions can be used to summarize the values of a numerical variable; others can be used to identify or count specific values for both numerical and categorical variables.

Question: What is the average age in the Applicants database?

FIGURE 5.14 Histogram of DECISION



The most common summary measure of a numerical variable is its average or mean, which can be calculated with the AVERAGE function in Excel. In the Applicants database, we find that

$$\texttt{AVERAGE}(\texttt{C2:C2918}) = 28.97.$$

Other useful summary measures are the median, minimum, and maximum. Those measures for the AGE variable are calculated with Excel functions as follows:

$$\texttt{MEDIAN}(\texttt{C2:C2918}) = 28$$

$$\texttt{MIN}(\texttt{C2:C2918}) = 1$$

$$\texttt{MAX}(\texttt{C2:C2918}) = 52$$

The minimum age in this dataset is 1 year. Presumably this is an error. In fact, any ages below 20 for MBA applicants are highly suspect.

Question: How many applicants have ages less than 20 in the Applicants database?

Excel provides the COUNTIF function for such questions. The COUNTIF function calculates the number of cells in a range that satisfy a given criterion. Here we can use:

$$\texttt{COUNTIF}(\texttt{C2:C2918},\texttt{"<20"}) = 5$$

The quotation marks are necessary as part of the criterion.

Question: What percentage of the applicants were given the DENY decision?

DECISION is a categorical variable, so we cannot calculate values as if it were a numerical variable. But we can count the number of times DENY appears and divide that result by the number of non-blank cells, as follows:

$$\texttt{COUNTIF}(\texttt{\$J\$2:\$J\$2918},\texttt{"Deny"})/\texttt{COUNT}(\texttt{\$J\$2:\$J\$2918}) = 0.79$$

The COUNT function does not count blank cells or cells containing text, so this approach will work correctly if the range is known to hold only numbers.

Excel provides a number of related functions. For example, COUNTA records the number of nonempty cells in a range, and COUNTBLANK counts empty cells. (Cells containing zero are ignored.)

Question: What is the average age of the applicants from the nonprofit sector?

This question asks for an average in a subset of the database. Intuitively, we might think of first filtering the data and then using the AVERAGE function. However, Excel's AVERAGE function does not adjust for the fact that we have filtered the data: it makes the calculation for the hidden cells as well as the visible cells. To make this calculation for just the visible cells, we can use the function $\texttt{SUBTOTAL}(1,\texttt{C2:C2918})$ because it ignores the hidden cells in the specified range.

The SUBTOTAL function performs the calculations for one of Excel's more familiar functions according to the first argument inside the parentheses, while ignoring hidden cells. In our example, the argument 1 specifies the average value. The list below gives the functions that can be calculated with the SUBTOTAL function:

1. AVERAGE
2. COUNT
3. COUNTA
4. MAX
5. MIN
6. PRODUCT
7. STDEV
8. STDEVP

9. SUM
10. VAR
11. VARP

5.4.5 Examine Relationships among Variables

Up to this point we have examined variables one at a time, but in many cases the relationships among variables are more important to the analysis than the properties of one variable. For example, variables that are highly correlated with each other generally should not be used in data mining, so identifying those variables is a key step in data exploration. Graphical methods can track relationships among as many as four or five variables simultaneously.

Question: How long have older applicants held their current job?

To answer this question we use XLMiner to create a scatterplot between AGE and JOB MONTHS in the Applicants database. Select Explore▶Chart Wizard▶Scatterplot Matrix: select the variables AGE and JOB MONTHS; then click on Finish. The results appear in Figure 5.15.

On the main diagonal we find the histograms of the two variables AGE and JOB MONTHS. The scatterplots are shown in the off-diagonals. (These charts may be referred to as *X-Y charts* in Excel.) In the scatterplot on the lower left, each record is plotted with its AGE value on the X-axis and its JOB MONTHS value on the Y-axis. (The axes are reversed in the plot on the upper right.) As we might have guessed, older applicants tend to have spent more time in their current job, although the relationship is less well-defined at lower ages. (To generate the scatterplot between two variables without the histograms, use the Scatterplot chart option in XLMiner.)

Question: To what extent do executives with higher salaries work in firms with higher return on assets (ROA)?

In this case we follow the same steps to create a scatterplot for SALARY and ROA in the Executives database. The results are shown in Figure 5.16.

Again, we find the histograms in the diagonal cells and the scatterplots in the off-diagonal cells. If there were a strong (positive) relationship between salary and ROA, we would see the data points lying near a straight line with positive slope. But here we see something like a random cloud of points with very little apparent association.

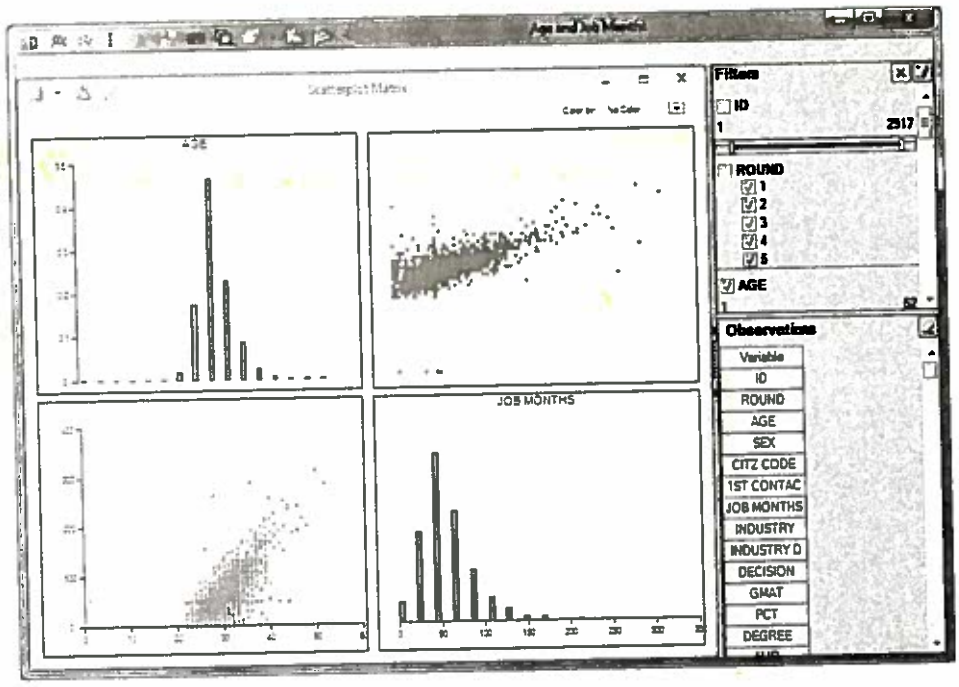


FIGURE 5.15 Scatterplot Matrix for AGE and JOB MONTHS

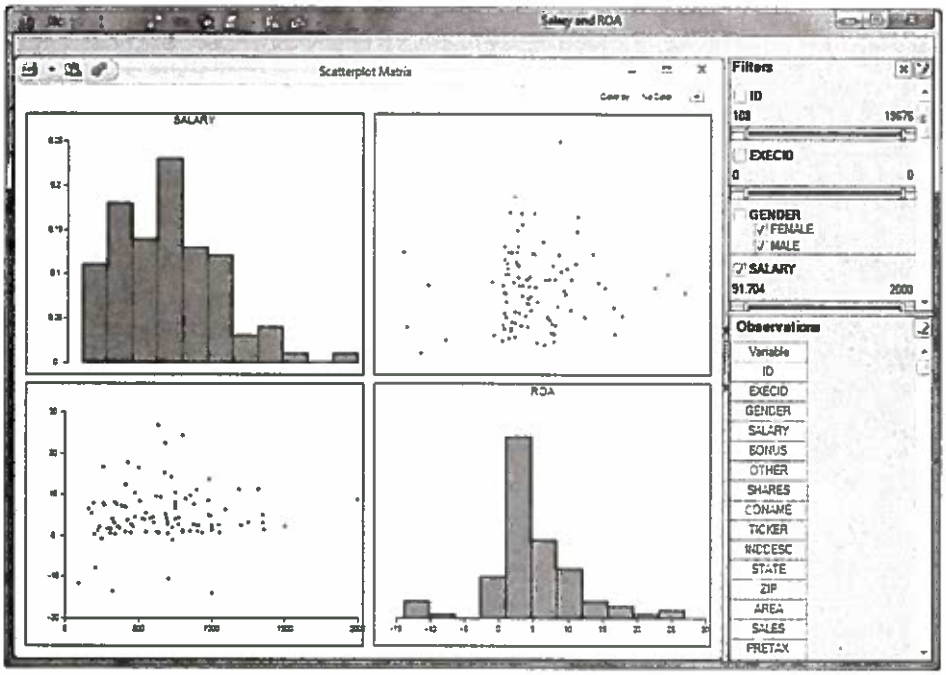


FIGURE 5.16 Scatterplot Matrix for SALARY and ROA

Scatterplots provide a strong visual indication of the degree to which two variables are correlated. However, they do not provide a precise numerical estimate of that correlation. For numerical precision we can use the CORREL function in Excel. For example, in the Applicants database, we find that

$$\texttt{CORREL(C2:C2918,G2:G2918)} = 0.72$$

which confirms the high degree of correlation we observe in Figure 5.15. On the other hand, in the Executives database, we find that

$$\texttt{CORREL(D2:D101,Q2:Q101)} = 0.07$$

which confirms the low degree of correlation we observe in Figure 5.16.

Question: How does the distribution of GMAT scores of applicants compare across the five application rounds?

This question asks us to compare five distributions, which is a difficult task because each distribution contains so much information. One solution is to summarize the distributions by various statistics, such as mean, median, and so on. We use the Boxplot option in XLMiner to generate this chart. (A boxplot is also known as a *box-and-whiskers plot*.) Select Explore▶Chart Wizard▶Boxplot; select the variables GMAT and ROUND; then click on Finish. The results appear in Figure 5.17.

Each of the five distributions for ROUNDS 1-5 is summarized in the vertical dimension by its mean, median, maximum, 75th percentile, 25th percentile, and minimum. The green rectangle highlights the range from the 75th to the 25th percentile. The mean is shown by a solid horizontal line and the median by a dashed horizontal line. The maximum and minimum values are shown by the horizontal lines connected to the green rectangle. (Place the cursor over one of the green rectangles to display the exact values of these statistics.) This chart shows that the mean (and most of the other statistics) of these distributions decline from Round 1 to Round 5. Also, the variability (as measured by the range from 75th to 25th percentile) appears to increase.

Question: How does the distribution of SALARY for executives compare among Males and Females?

We answer this question using a boxplot of SALARY and GENDER in the Executives database. The results are shown in Figure 5.18.

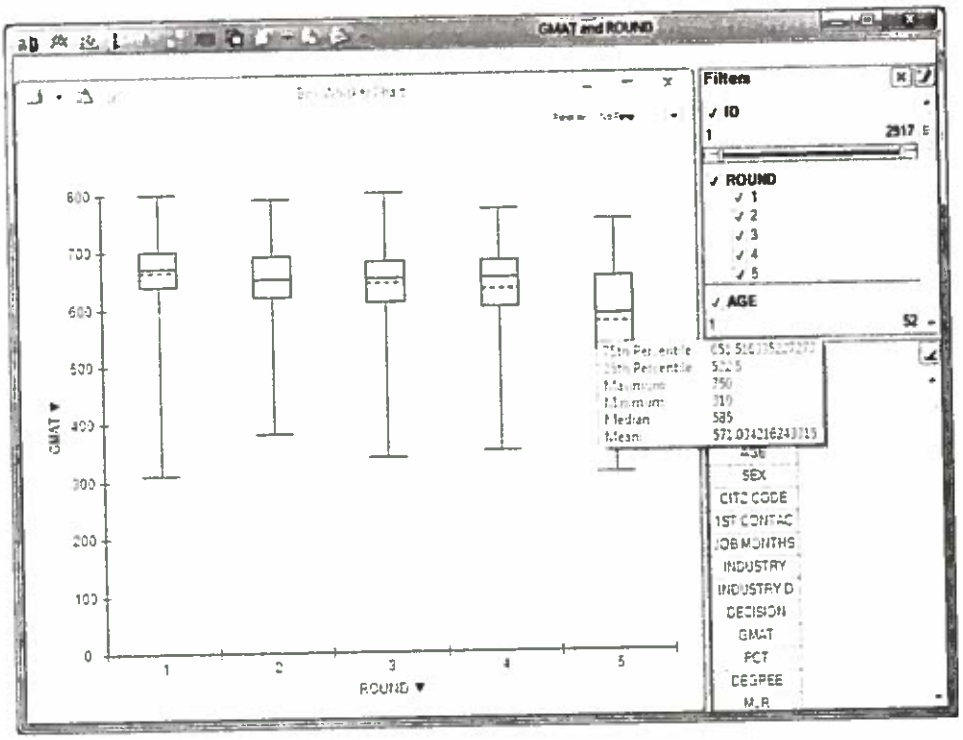


FIGURE 5.17 Boxplot of GMAT and ROUND

Among this sample of 100 executives, male salaries are generally higher than female salaries (with means, for example, of 685.62 versus 500.84), but the lowest salary for males is well below that for females. However, only two females appear in this database, so the results for this group may not be representative.

Question: How does the variability of GMAT scores compare across rounds?

We observed in Figure 5.17 that the variability of GMAT scores appeared to increase from Round 1 to Round 5. This conclusion was based on the range from the 25th to the 75th percentile. A more common measure of variability is the standard deviation. We use a

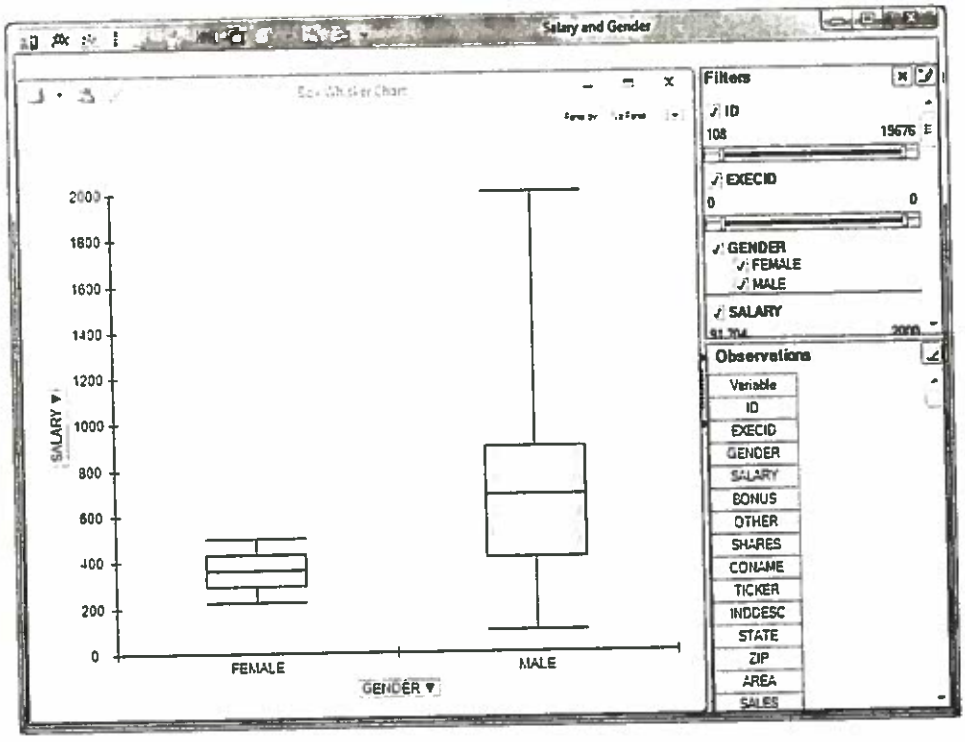


FIGURE 5.18 Boxplot of SALARY and GENDER

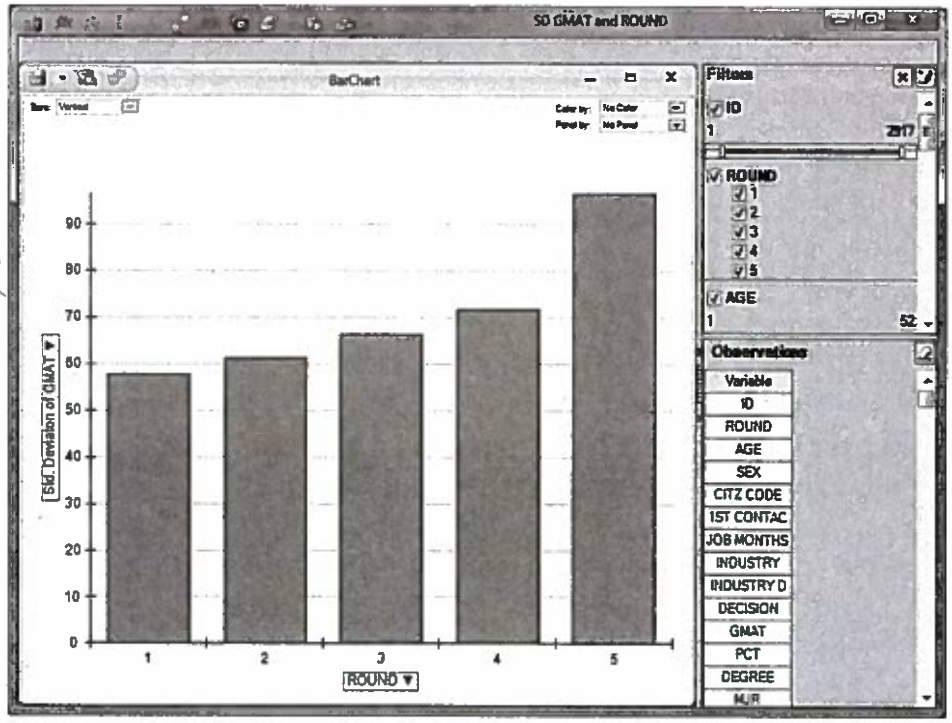


FIGURE 5.19 Bar Chart of the Standard Deviation of GMAT by ROUND

Bar Chart in XLMiner to show how the standard deviation varies across rounds. Select Explore▶Chart Wizard▶Bar Chart; select the variables GMAT and ROUND; then click on Finish. (The chart will first appear with COUNT of GMAT on the vertical axis. Double click on this box and a menu of statistics will appear. Then select `Std. Deviation`.) The results shown in Figure 5.19 demonstrate clearly that the standard deviation increases steadily across rounds.

Question: Is the salary of an individual executive related in a systematic way to the firm's sales, assets, or dividend yield?

We would often like to know whether a group of variables is tightly or loosely interconnected. This is a more complex question than simply asking whether pairs of variables are correlated. It requires us to somehow track the values of records on multiple variables and compare them as a group. One way to do this is with a Parallel Coordinates chart. Select Explore▶Chart Wizard▶Parallel Coordinates; select the variables SALARY, SALES, ASSETS, and DIVYIELD; then click on Finish. The results are shown in Figure 5.20

The four variables are shown on the horizontal axis, while the vertical axis depicts the range of values for each variable. For example, the SALARY variable ranges from 91.704 to 2000 in this sample. The values for each record are plotted on these four vertical axes, and a line is drawn (representing a given record) connecting each of the four points. (Highlight an individual record by drawing a rectangle with the cursor around any one line.) If, hypothetically, high salaries were associated with low sales, high assets, and low dividend yields, the chart would show bands of lines in a W-pattern. The results in this case are more complex. Salaries seem to be loosely correlated to sales, and sales to assets, but dividend yields vary widely among firms with similar assets.

Up to this point we have illustrated some of the most useful types of charts in XLMiner without going into detail about how charts can be specialized. Four options are available for most charts that allow us to address complex questions:

- Filtering
- Coloring
- Paneling
- Details-on-demand

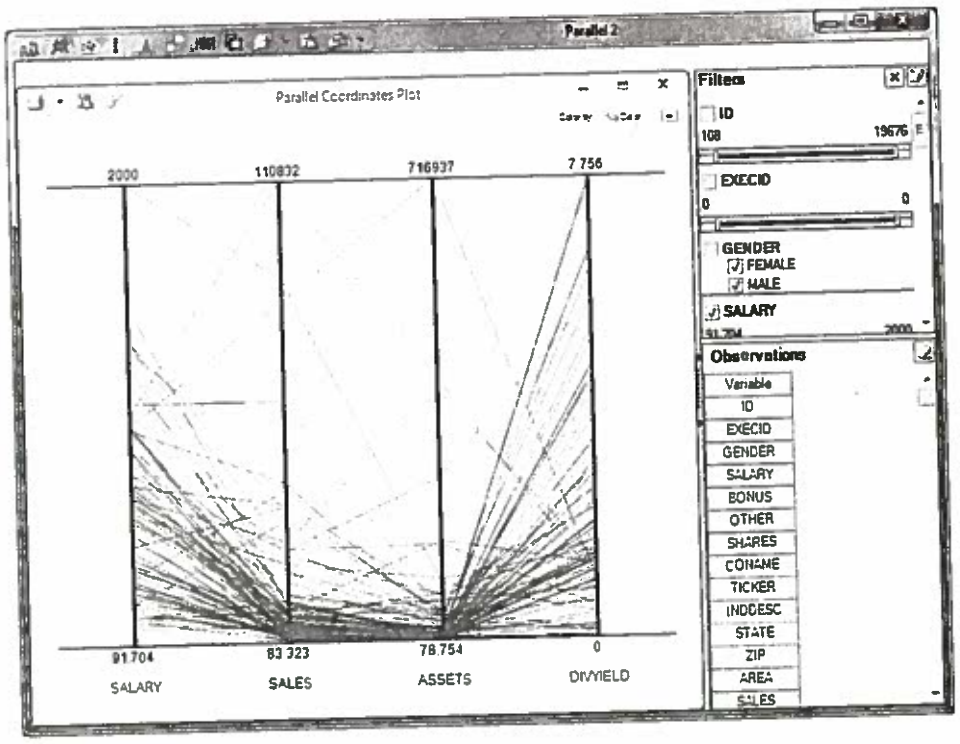


FIGURE 5.20 Parallel Coordinates Chart for SALARY, SALES, ASSETS, and DIVYIELD

Each chart is accompanied by a control panel on the right-hand side. (This panel can be displayed or hidden by clicking on the vertical space to the right of the chart itself.) The upper half of the control panel is devoted to **filters**. Each variable is given a filter, in the form of a slider for numerical variables or a checkbox for categorical variables. By using these sliders and checkboxes we can include or exclude any combination of values from the chart. For example, by checking Male but not Female, we can specialize the chart so it displays the data for Males only.

Coloring and Paneling are controlled by drop-down lists at the upper right of the chart. **Coloring** allows us to apply distinct colors to the different values of a categorical variable. **Paneling** creates a separate chart for each distinct value of a categorical variable. Figure 5.21 shows the mean for GMAT scores over the five rounds colored by sex and paneled by citizen code.

Although charts are generally an excellent way to capture an entire variable (or multiple variables) at one time, they are not as useful for identifying specific records. Yet there are times when we would like to understand more about a few specific points on a chart, such as outlying values. XLMiner provides an option called **details-on-demand** to make this form of exploration possible. In Figure 5.22 we have created a scatterplot of AGE and JOB MONTHS. Two records are of particular interest: the ones in the upper right of the chart with high values for both AGE and JOB MONTHS. To learn more about these records we draw a rectangle around them using the cursor. XLMiner then colors them differently from the other points and displays their values on all variables in the Observations pane of the control panel located on the lower right. Here we can see that these records both represent Females from the US.

Most of the methods we have discussed to this point involve charts. When more precision is needed, we turn to **cross-tabulation tables**, or cross-tabs. A cross-tabulation is a table that shows how one variable is distributed across another. For example, we might create a table showing how the average GMAT of applicants varies by application round. Another cross-tab would show analgesic sales by store and by brand.

Cross-tabs are created in Excel using the Pivot Table wizard, which is located on the Insert ribbon. To appreciate how pivot tables work, we return to the Analgesics database and specifically to an edited version where we have combined all of the item descriptions into single-word brand names. (The modified data are saved as a separate worksheet with a new database name, such as BrandData.) In what follows we describe the construction of three pivot tables, each elaborating on the one before.

Question: What are total sales of painkillers, and how do they break down by brand name and by store?

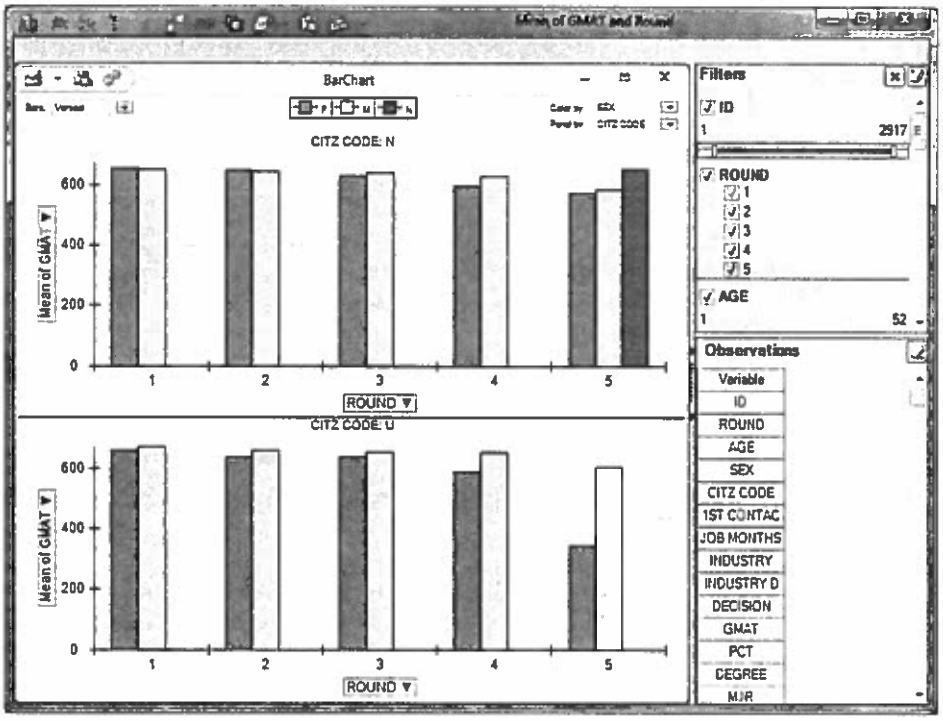


FIGURE 5.21 Bar Chart for Mean of GMAT Illustrating Paneling by CITZ CODE and Coloring by SEX

First select the database and then choose Insert▶Tables▶Pivot Table▶Pivot Table, which opens the Create Pivot Table window (Figure 5.23).

Place the Pivot Table report in a new worksheet and click OK. The new worksheet will have a template for the pivot table and a panel (Pivot Table Field List) on the right containing the field names (Figure 5.24).

The Pivot Table Field List contains a drop-down menu that offers different layouts; we have shown the "stacked" layout in Figure 5.24. Next, drag the list item SALES to the Values box. A simple pivot table appears with the title Sum of Sales in cell A3 and the value 9,432 in cell A4. (If a different title appears, such as Count of SALES, double-click

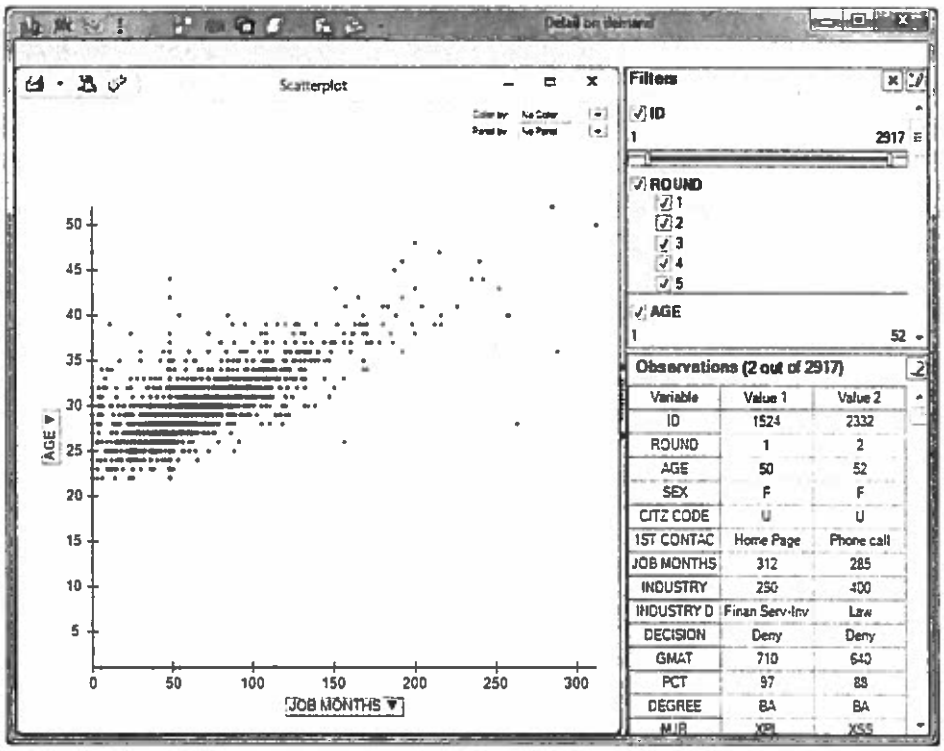


FIGURE 5.22 Scatterplot of AGE and JOB MONTHS Showing Detail on Demand

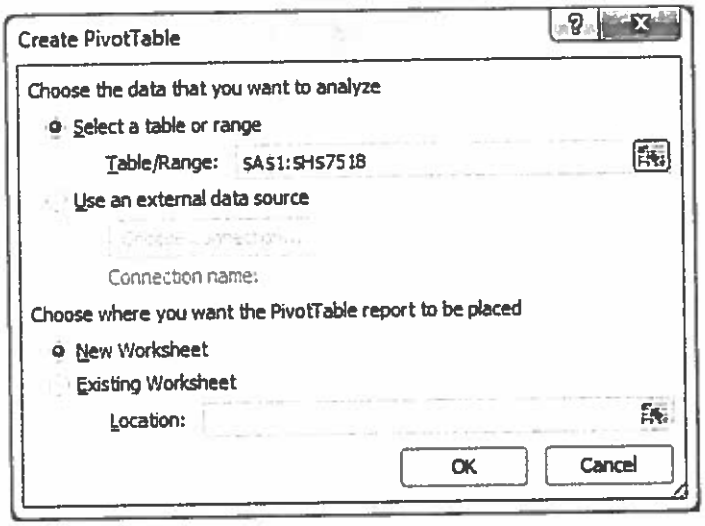


FIGURE 5.23 The Create Pivot Table Window

on that name, and the Value Field Settings window appears, similar to the one in Figure 5.25. Select Sum in the main window of this option and click OK.) When the cursor is located on the pivot table (cells A3:A4), the Field List remains visible; otherwise it disappears.

Extend the pivot table by dragging the list item DESCRIPTION to the Row Labels box. A more detailed pivot table appears in cells A3:B12, showing the breakdown of sales by brand name. Extend the pivot table again by dragging the list item STORE to the Column Labels box. The pivot table now shows a sales breakdown by both brand name and store, as shown in Figure 5.26.

Pivot tables can be modified easily after they are built. For example, we can use the filtering arrows in the row or column headings to limit our table to particular row and column entries. We can also edit the table using the Pivot Table Field List. (If the Pivot Table template and Field List are not visible, click anywhere in the pivot table to display them.) We can, for example, substitute WEEK for STORE in the Column Labels box to obtain a breakdown of sales by brand and by week.

Question: In the Applicants database, how does the average GMAT score vary according to the round in which the application is filed?

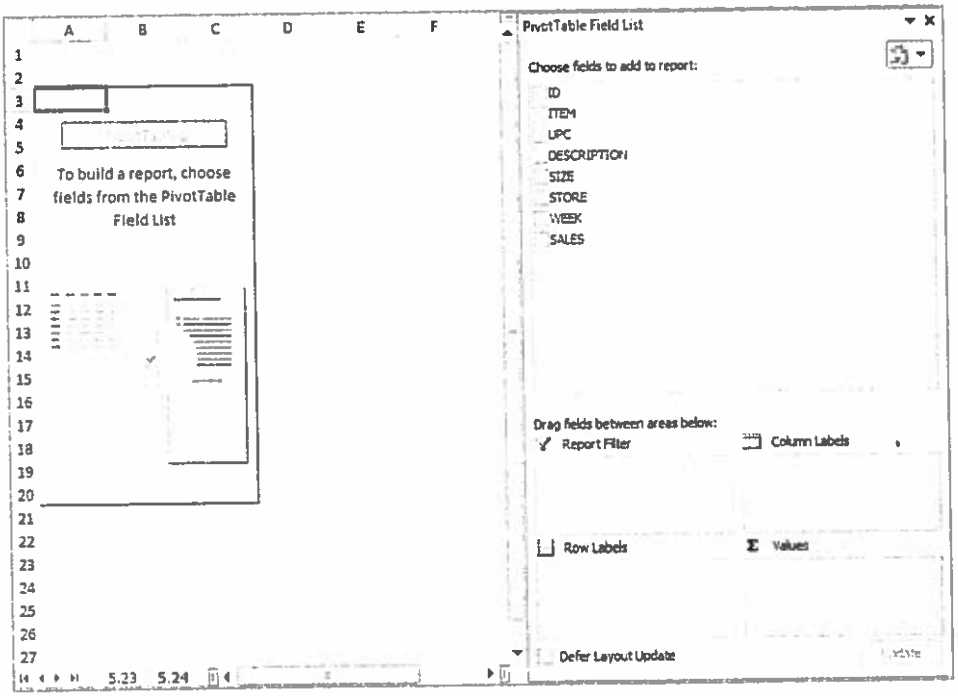


FIGURE 5.24 The Pivot Table Template

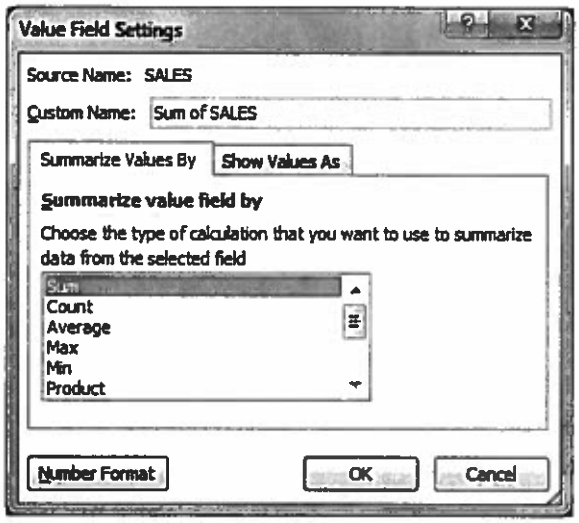


FIGURE 5.25 The Value Field Settings Window

In developing a strategy for round-by-round selections, it is helpful to know whether systematic differences exist among the rounds. To probe this topic, we return to the Applicants database and set up a pivot table. The Pivot Table shows average GMAT scores broken down by ROUND. Figure 5.27 displays the result. Evidently, there is a trend toward lower GMAT scores during the course of the overall admission process.

FIGURE 5.26 Pivot Table Showing Sales by Brand Name and by Store Number

	A	B	C	D	E	F	G	H	I	J
1										
2										
3	Sum of SALES	Column Labels								
4	Row Labels		100	101	102	103	104	105	Grand Total	
5	ADVIL		434	469	499	207	301	226	2136	
6	ALEVE		229	169	274	106	125	93	996	
7	ANACIN		105	64	93	29	11	63	365	
8	BAYER		213	223	312	65	111	95	1019	
9	BUFFERIN		32	32	46	8	26	12	156	
10	EXCEDRIN		211	235	444	132	188	115	1325	
11	MOTRIN		120	128	183	76	98	60	665	
12	TYLENOL		737	543	705	270	222	293	2770	
13	Grand Total		2081	1863	2556	893	1082	957	9432	
14										
15										
16										

5.23 5.24 5.25 5.26

FIGURE 5.27 Pivot Table Showing Average GMAT Scores by Round

	A	B	C	D	E	F	G	H
1								
2								
3	Row Labels	Average of GMAT						
4	1	665.1						
5	2	650.8						
6	3	642.8						
7	4	629.5						
8	5	537.5						
9	(blank)	675.0						
10	Grand Total	651.5						
11								
12								
13								
14								
15								
16								

5.23 5.24 5.25 5.26 5.27

5.5 SUMMARY

The ability to use data intelligently is a vital skill for business analysts. Although complex data is typically stored in sophisticated databases with specialized data management tools, analysts tend to perform most of their data analysis in Excel. In this chapter, we presented the tools needed to understand Excel databases, to explore data, and to prepare it for further analysis.

We organized the data exploration task into five steps:

- Understand the data.
- Organize and subset the database.
- Examine individual variables and their distributions.
- Calculate summary measures for individual variables.
- Examine relationships among variables.

The first step is the most important: before undertaking any analysis it is crucial to understand the database and how it was defined, collected, and scaled; whether it is numerical or categorical; whether it contains missing or erroneous values; and so on. Then we can organize and subset the data in various ways by sorting and filtering the data. The focus then shifts to individual variables and how are they are distributed. Distributions provide a useful graphical overview of individual variables; but when we want more detail, we calculate summary measures such as the mean, minimum, maximum, etc. Finally, we explore how variables are related to each other using scatterplots, boxplots, and other graphical aids provided by XLMiner.

Careful preparation of the raw data is often required before data mining can succeed. Missing values may have to be removed or replaced with average values for data mining algorithms to perform properly. Likewise, numerical variables may need to be converted to categorical variables, or categorical variables may need to be converted to numerical form. Sometimes a single variable must be transformed or multiple variables combined into a new variable in order to capture the information needed for data mining to succeed. Normalization of the data may also be required or recommended. We discuss the important details of data preparation in the chapter appendix.

SUGGESTED READINGS

A great deal has been written about the best ways to use charts in presentations, but much less has been written about how to explore data itself. The books listed below are some of our favorites.

Few, S. C. 2004. *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. Oakland, CA: Analytics Press.

Few, S. C. 2009. *Now You See It: Simple Visualization Techniques for Quantitative Analysis*. Oakland, CA: Analytics Press.

Koomey, J. G. 2008. *Turning Numbers into Knowledge: Mastering the Art of Problem Solving*. Oakland, CA: Analytics Press.

EXERCISES

1. The database *Dish.xlsx* contains a transaction history describing more than 4,000 purchases of detergent at a number of stores in a grocery chain over a period of several weeks.

a. How many records are in this database?

b. How many fields are in this database?

c. Is the field SALE nominal or numerical?

d. Is the variable PRICE measured on an interval or ratio scale?

e. Which field contains blank cells, and how many does it contain?

f. What is the highest price in the database?

g. How many records pertain to the description SUNLIGHT GEL 2.49?

h. Create a histogram of the PRICE variable and interpret it.

i. What is the average price?

j. Are higher priced products generally more profitable?

k. Does profit vary systematically across stores?

l. In week 387, which stores had average profit over 18?

2. The database *Tissue.xlsx* contains a transaction history describing more than 3,700 purchases of facial tissues at a number of stores in a grocery chain over a period of several weeks.

a. How many records are in this database?

b. How many fields are in this database?

c. Is the field STORE nominal or numerical?

d. How many records have a price exceeding 5?

e. Which field contains blank cells, and how many does it contain?

f. For what percentage of records does profit equal zero?

g. What is the second-highest value for profit?

h. Compare the average price across stores.

i. Compare the distribution of profit across the weeks.

j. Describe the relationship between price and profit for items on sale (SALE = S).

k. How much average profit did STORE 102 make in WEEK 390?

3. The database *Population.xlsx* contains data on the populations of the 50 states from 1990 to 1999.

a. How many records are in this database?

b. How many fields are in this database?

c. Are there any missing data?

d. Which state was ranked 25th in population in 1993?

e. Which state had the largest percentage increase in population from 1990 to 1999?

f. Which states had populations of more than 1 million and less than 2 million in 1995?

g. Create a histogram of the populations by state for 1999 and interpret it.

4. The database *Executives.xlsx* contains salary and company data on a number of executives.

a. How many records are in this database?

b. How many fields are in this database?

c. Are there any missing data?

d. Is the STATE field nominal or ordinal?

e. Among the top five executives in terms of Salary, how many work in the oil industry?

f. How many executives work for companies in California or New York and have sales less than 4,000?

g. Compare the salaries of male and female executives.

h. Does having a high return on assets predict a high bonus?

i. Which two states have the highest average salaries?

5. The database *Applicants.xlsx* contains a description of an MBA applicant pool.

a. How many records are in this database?

b. How many fields are in this database?

c. Are there any missing data?

d. Is the ROUND field nominal or ordinal?

e. What decision was made on the top four candidates by GMAT?

f. Among enrolling students, what is the average GMAT score? What is the average in the applicant pool as a whole?

g. How many in the applicant pool represented Financial Services in terms of job background? What was their average age?

h. Create a histogram for Job Months and interpret it.

6. The database *Boston Housing.xlsx** contains demographic, environmental, and housing data on a number of towns in the Boston area.

a. How many records are in this database?

b. How many fields are in this database?

c. Is the field RAD ordinal or numerical?

d. Is the field RM interval or ratio?

e. How many fields have missing data?

f. What is the lowest level of NOX among these towns?

g. How many towns have NOX below 4.0?

h. Does a histogram of MEDV reveal any unusual features of the data?

i. Is the median value of homes related to their proximity to the Charles River?

j. Does the distance of a town to Boston have an influence on the median values of homes?

7. The database *German Credit.xlsx*** contains data on prior applicants for credit, including whether they proved to have good or bad credit.

a. How many records are in this database?

b. How many fields are in this database?

c. Is the field HISTORY nominal or ordinal?

d. Is the field PRESENT_RESIDENT nominal or ordinal?

e. How many records are male, divorced, and unskilled resident?

f. Which record number has the highest credit amount?

g. What percentage of records has less than 100DM in a savings account?

h. How does the credit amount compare for those who own their residence versus those who do not?

i. How are the amount of credit and the duration of the loan related?

8. The database *Universal Bank.xlsx**** contains information on a large number of current bank customers.

a. How many records are in this database?

b. How many fields are in this database?

c. How many fields have missing data?

d. Are there any cells coded as text in the Education field?

e. What is the lowest value of Age?

f. How many records have Experience of 2 or less and a personal loan?

g. How is Income distributed?

h. Does the distribution of Income change with Education?

i. Is there a relationship between Income and Mortgage?

9. The database Book *Club.xlsx*† contains data on a number of current customers of a direct-mail book club.

a. How many records are in this database?

b. How many fields are in this database?

c. How many fields have missing data?

d. What is the highest value for the amount spent on all products?

e. How many records are females who first purchased 2 months ago?

f. What is the overall distribution of the amount spent on all products?

g. Does the distribution of the amount spent on all products change substantially with the number of Italian books purchased?

h. How are the amounts spent on all purchases and the number of purchases related?

10. The database *Income.xlsx*‡ contains average personal income data for a number of states over the years 1929–1999.

a. How many records are in this database?

b. How many fields are in this database?

c. What was the absolute increase in the personal income of Maine over this period?

d. What was the percentage increase in the personal income of California over this period?

e. In what years was the personal income of Illinois at, or under, 399 and that of Massachusetts at, or over, 613?

f. Plot the personal income for Connecticut over this time period.

g. Create a histogram of average personal income by state for 1999 and interpret it.

* Source: UCI Machine Learning Repository (http://archive.ics.uci.edu/ml).
** Source: UCI Machine Learning Repository (http://archive.ics.uci.edu/ml).
*** Source: Cytel, Inc. 2005
† Source: Shmueli, G., N. R. Patel, and P. C. Bruce. 2010. *Data Mining for Business Intelligence*. New Jersey: John Wiley & Sons, p. 367.
‡ Source: U.S. Department of Commerce, Bureau of Economic Analysis (http://www.bea.gov/bea/regional/spi/).

APPENDIX 5.1 DATA PREPARATION

Most raw data collected by analysts contains missing values, inconsistent data (for example, values in one field recorded as both numbers and text), or other types of problematic data. Some of these data problems will rule out the use of certain data mining tools; others will bias any analysis done on the data. For these reasons, careful cleaning of raw data is an important step in data analysis. Raw data, even when cleaned, is also not necessarily in the appropriate form for the intended analysis. For example, some data mining tools require that categorical variables be transformed into numerical variables. In this appendix we discuss a variety of approaches to cleaning and transforming data.

A.5.1 HANDLING MISSING DATA

Individual values can be missing from a database for many reasons. Missing values may be more common among variables that are difficult to collect or that subjects do not wish to divulge. For example, all the names and addresses may be present in a database, but a significant percentage of family incomes may be missing because respondents chose not to reveal that information. If there is no systematic pattern to the missing data, and only a small percentage of records are affected, the simple solution of removing all such records will suffice. But this procedure will introduce a bias if the missing data are not randomly distributed. If, for example, wealthier families are less likely to divulge their incomes, then removing records with missing income data may mean that wealthy families are under-represented in the database.

The first step in dealing with missing data is to locate it. The COUNTBLANK function can be used to count the number of blank cells in a column. If the count of blank cells is not zero, we can locate individual blank cells by inserting a new column of formulas parallel to the column of data that will highlight the blank cells. A formula such as

```
IF(ISBLANK(A2) = TRUE,"BLANK",0)
```

will show the highly-visible word BLANK if A2 is a blank cell and 0 if it is not. Having located the blank cells, we can detect whether patterns occur in the blank cells by sorting or filtering the database on an important variable and then checking whether the pattern of blank cells in any other variable appears to be concentrated.

Another common data quality problem involves a variable with both numerical and text entries. These cells can be counted and located using the ISNUMBER, and ISTEXT functions. For example, we can use the formula

```
IF(ISNUMBER(A2) = TRUE,1,0)
```

to locate individual cells with numerical entries, and then we can SUM the column of such formulas to count the number of numerical cells. In most cases, either the text or the numerical entries are the result of formatting errors and can easily be fixed by correcting them individually.

XLMiner provides a utility for dealing with missing data called Missing Data Handling (Data Analysis►Transform►Missing Data Handling). Figure 5A1 shows the Missing Data Handling window.

This utility provides a number of options for dealing with missing data in categorical and numerical data. For categorical variables, the choices are to ignore the missing data (`No treatment`), delete the record, replace the missing data with the modal value, or replace the missing data with a user-specified value. For numerical data the same choices are available. However, instead of replacing missing data with the modal value, the options include using the mean or median. Any one of these choices can bias the ultimate results, so the careful analyst will repeat the analysis using different approaches to missing data.

A.5.2 BINNING CONTINUOUS DATA

Some data mining methods require all-categorical data. To meet this requirement, we transform continuous (numerical) data into categorical form, a process known as **binning**.

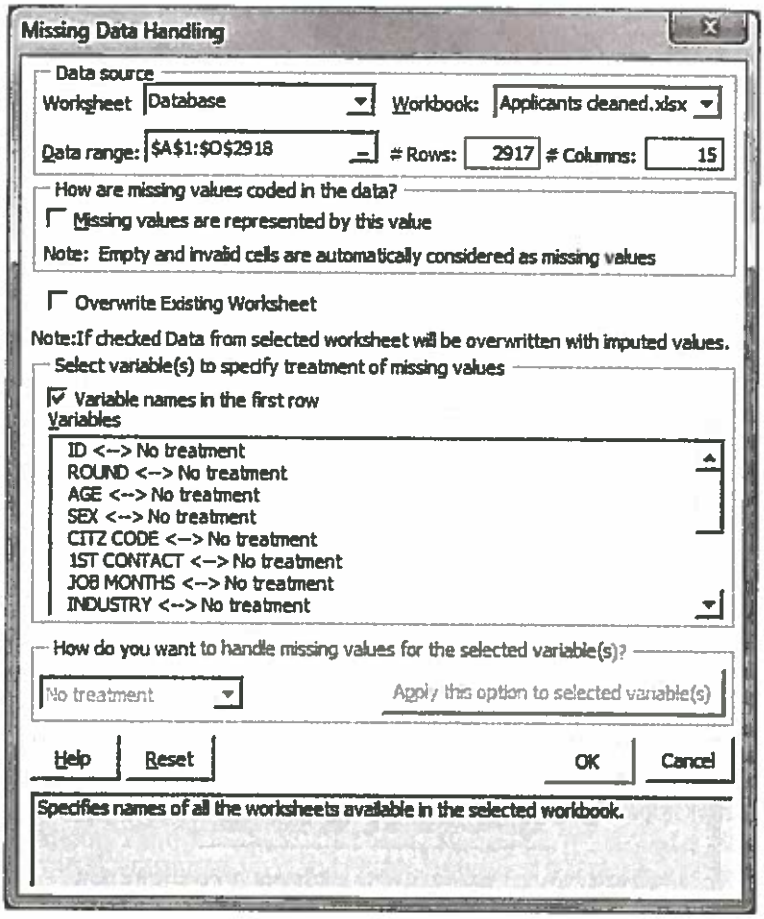


FIGURE 5A1 The Missing Data Handling Window

The basic idea is simple: define a number of intervals ("bins") that cover the range of the variable, and then assign a value to each interval.

XLMiner provides the Bin Continuous Data utility to automate this process (Data Analysis▶Transform▶Bin Continuous Data). Figure 5A2 shows the Bin Continuous Data window.

The main options provided here are to define the bins in terms of equal numbers of values (`Equal count`) or by equal length (`Equal interval`), and to assign values to bins using the rank, mean, or median of the bin.

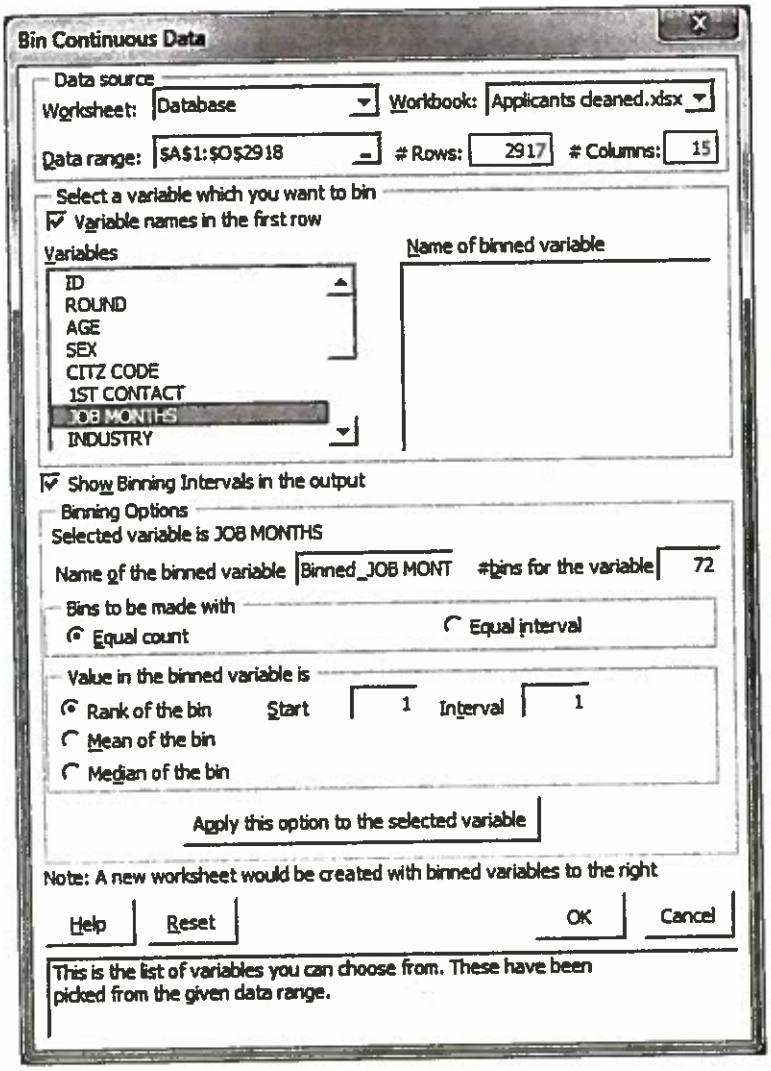
A.5.3 TRANSFORMING CATEGORICAL DATA

Just as numerical data sometimes need to be transformed into categorical data, the opposite transformation is often required. Multiple linear regression, for example, accepts only numerical variables. A categorical variable such as MONTHS, which takes on the 12 values January through December, can be transformed into 12 so-called **dummy variables**, each of which takes on only the values 0 and 1. The first dummy variable is 1 when the month is January and 0 otherwise; the second is 1 when the month is February and 0 otherwise; and so on. For any given value of the original categorical variable, only one of the dummy variables takes on the value 1. XLMiner provides the Create Dummies utility to automate this process (Data Analysis▶Transform▶Transform Categorical Data ▶Create Dummies).

A related transformation involves assigning sequential numbers to the values of an ordinal variable. Thus a variable that takes on the values Low, Medium, and High could be transformed into a numerical variable taking on the values 1, 2, and 3. XLMiner provides the Create Category Scores utility to automate this process (Data Analysis▶Transform▶Transform Categorical Data▶Create Category Scores).

Finally, some categorical variables have too large a number of distinct values to be useful in data mining. (XLMiner limits categorical variables to 30 distinct values.) The

FIGURE 5A2 The Bin Continuous Data Window



solution is to reduce the number of categories by combining some values into the same category. XLMiner provides the Reduce Categories utility to automate this process (Data Analysis▶Transform▶Transform Categorical Data▶Reduce Categories). Using this utility we can either define up to 29 categories for the most frequent values and then include all the remaining values in the 30th category, or manually assign values to categories (up to the limit of 30).

A.5.4 FUNCTIONAL TRANSFORMATIONS

We have emphasized that the raw data collected by analysts are generally not in a form suitable for analysis and require careful cleaning. But there is another important sense in which raw data are not suitable for analysis. The analyst's job is not only to acquire data but also to ensure that the *information* contained in the data is available for the analytic methods used to derive insights. One way to improve the information content of the data is to transform it. We can transform data in an unlimited number of ways, depending on the context. Some involve a single variable, such as taking the logarithm or square root of the original data; others involve combining more than one variable into a new variable. Unfortunately, analysts largely overlook these options. We give one example here as an illustration.

If we are trying to identify individuals who are likely to default on home loans, we may suspect that neither their income nor the size of their mortgage by itself is critical.

Rather, it is when income is relatively low *and* the size of the mortgage relatively high that individuals are most likely to default. To test this idea, we can create a new variable from the two original ones (INCOME and SIZE OF MORTGAGE), using a formula with logic such as this:

$$\text{If(INCOME} < 100,000 \text{ and SIZE OF MORTGAGE} > 1,000) \text{ then HIGH RISK;}$$
$$\text{else LOW RISK}$$

This creates a categorical variable with values High Risk and Low Risk from two existing numerical variables.

A.5.5 NORMALIZATIONS

Rescaling a variable, or **normalization**, is required for a number of data mining methods such as neural networks. It can also be used in other contexts to make variables more nearly comparable to each other.

The **linear normalization** is the most common. This approach rescales a variable so that its values all lie between 0 and 1. The original values of the variable are transformed using the following formula, where X_{new} is the new value, X_{old} is the old value, and Min and Max are the minimum and maximum values of this variable in the dataset:

$$X_{new} = (X_{old} - \text{Min})/(\text{Max} - \text{Min})$$

The **Z-score normalization** is used to adjust variables for differences in their variability as well as their scale. Without adjustment, a variable with a high variance relative to others is likely to have a larger impact. Imagine that we have two variables (Variable 1 and Variable 2) that both have a mean of 100, but standard deviations of 50 and 100, respectively. These could represent student test scores, for example. A value of 200 on Variable 1 lies two standard deviations above the mean, while an outcome of 200 on Variable 2 lies only one standard deviation above its mean. In a sense, the outcome of 200 on Variable 1 is more unusual than the same numerical outcome on Variable 2. To reflect this, we transform the original values using the following formula, where X_{new} is the new value, X_{old} is the old value, and the Mean and Standard Deviation are calculated from the values of this variable in the dataset:

$$X_{new} = (X_{old} - \text{Mean})/\text{Standard Deviation}$$

6 Classification and Prediction Methods

6.1 INTRODUCTION

In the previous chapter, we discussed three types of tasks that analysts engage in: descriptive, predictive, and prescriptive. This chapter is devoted to predictive methods, including both classification and numerical prediction. We use the term **classification** when the task is to predict which class an individual record will occupy; we use the term **prediction** when the task is to predict a numerical outcome for an individual record. For example, classification applies when we wish to predict whether a particular customer will buy; prediction applies when we wish to forecast how much they will spend.

Previously, we offered the admonition to maintain a skeptical attitude toward data. With a skeptical attitude, we are motivated to spend the time needed to carefully explore our data. Here, we offer another admonition: *do not expect magic* from the methods presented in this chapter. The success of any of these methods depends on carefully selected, cleaned, and prepared data. Even then there are pitfalls that only the skeptical analyst can avoid.

The methods we present here originated in a variety of fields. Some come from classical statistics, others from the more recently developed fields of machine learning, artificial intelligence, or decision analysis. As a broad generalization, we could say that in the world of classical statistics, computing was difficult and data were scarce. Moreover, the focus was more on building explanatory models, which suggest the factors that influence the outcome variable, than on predictive models, which aim for accurate predictions on new data. The methods of classical statistics were developed under those constraints. But the more recently developed approaches, which are often referred to collectively as **data mining**, have arisen in a world where large amounts of data are readily available and computing power is essentially unlimited. Moreover, the data mining approach focuses more on predictive modeling than on explanatory modeling. Within the data mining approach, a model is fit using one part of the database and tested using a different part of the database. Since a well-trained analyst needs some facility with a variety of methods, in this chapter we have included the most practical and reliable methods from both schools of thought.

We begin the chapter with a discussion of issues that are common to all classification and prediction approaches: over-fitting, partitioning, and performance measurement. Then we present six widely-used approaches to classification and prediction:

- *k*-Nearest Neighbor
- Naïve Bayes
- Classification and Prediction Trees
- Multiple Linear Regression
- Logistic Regression
- Neural Networks

6.2 PRELIMINARIES

6.2.1 The Problem of Over-fitting

Behind all data analysis is the notion that the data we observe result from two influences: **patterns** and **noise**. The patterns are the stable, underlying relationships that persist and that we can expect to observe both in the existing data and in new data generated by the

same process. Noise reflects the transient, random effects of factors that will not be observed in new data. The goal of data analysis is to fit models to the patterns and not to the noise. A model is **over-fit** when it matches the available data so well that it reflects not only the patterns but the noise as well. Over-fit models do not predict the future well because they reflect temporary effects too closely.

We can illustrate the problem of over-fitting using a simple example. The task is to predict sales given the data below:

Year	*Sales*
1	10
2	19
3	27
4	29
5	45
6	23
7	35
8	42
9	39
10	56

If we fit a model to this entire dataset, we have no additional data for testing it. Instead, we fit models to the first five data points and test how well they predict the remaining five data points. (This process, called *partitioning*, is described in more detail in the next section.)

Figure 6.1 shows one model for this data. Here we have fit a fourth-order polynomial function ($y = 1.208x^4 - 13.08x^3 + 47.79x^2 - 60.91x + 35$) to the five data points. This function goes precisely through each point, so the fit is perfect.

Figure 6.2 shows an alternative model: a linear fit ($y = 7.9x + 2.1$). In this case the model does not fit the data nearly as well.

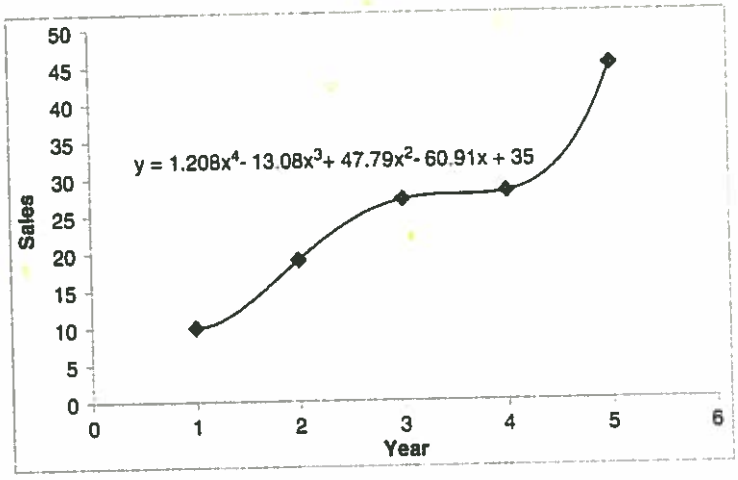


FIGURE 6.1 Polynomial Function Fit to Sales Data

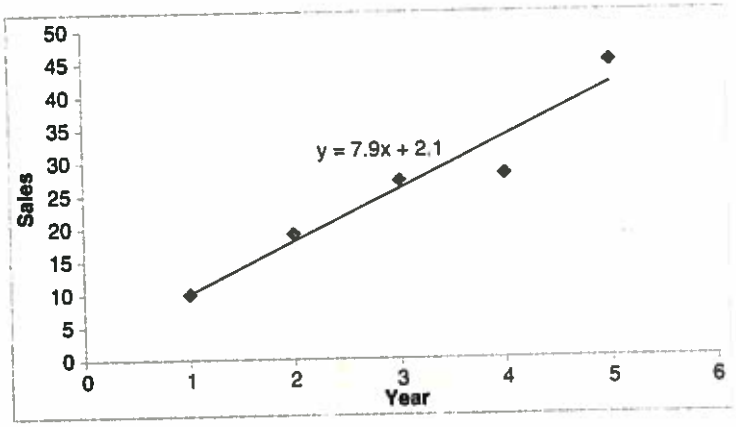


FIGURE 6.2 Linear Function Fit to Sales Data

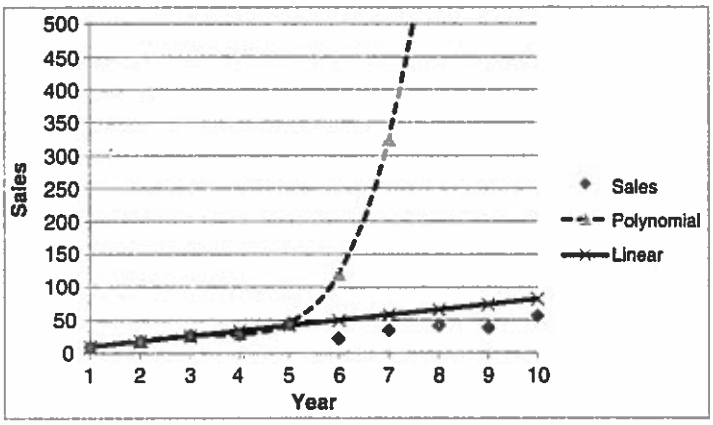


FIGURE 6.3 Sales Data and Linear Function

Which model does a better job of predicting the remaining data for years 6 through 10? As Figure 6.3 shows, the linear function does a reasonable job of predicting the values over this time period, even though sales appear to have dropped below the trend of the first five years. The polynomial function, on the other hand, predicts sales of 2,785 in Year 10, almost *fifty times* the actual value. As it happens, the underlying pattern in this data is closer to a linear trend than it is to the highly nonlinear shape of the polynomial. The polynomial fits both the pattern and the noise, so it diverges dramatically from the actual data in the later years. The linear function, despite its relatively poor fit to the early data, does not incorporate the noise to such an extent.

6.2.2 Partitioning the Database

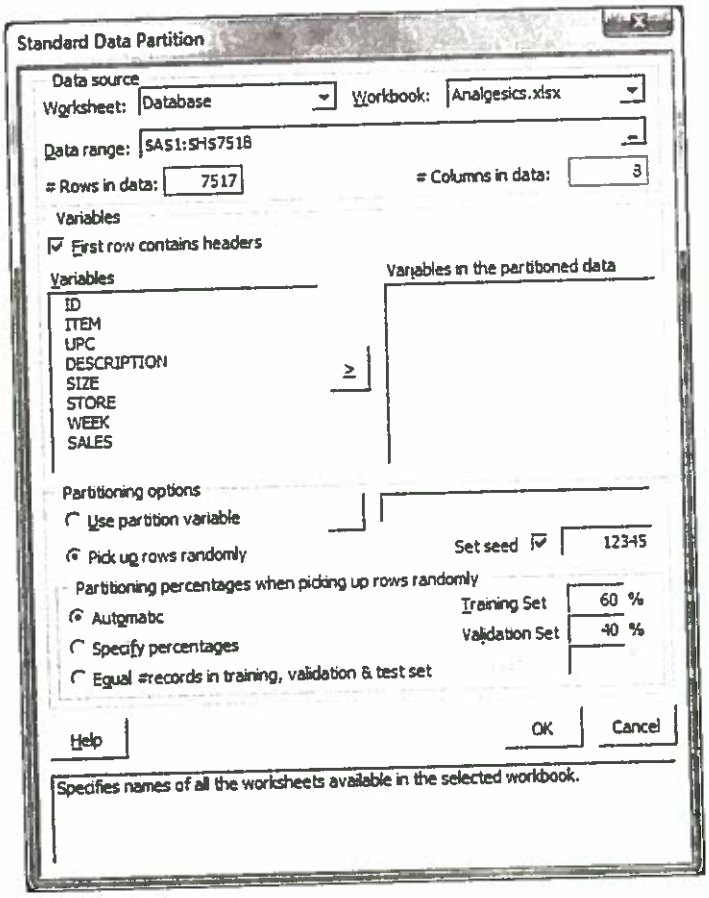
The fundamental strategy used in all classification and prediction methods to avoid the problem of over-fitting is to develop a model on one portion of data and test it on another. Without testing our models on a separate set of data, we will generally be overly optimistic as to how well they will predict. This process of creating subsets of the data for different purposes is called **partitioning**. The first partition, called the **training partition**, is used to develop the model. If different methods are to be tested and compared (such as logistic regression and neural networks), they are both developed on the training partition. The second partition is the **validation partition**, which is used to assess how well the model performs on new data. In some cases we also use the validation partition to fine-tune the model; in these cases we use a third partition called the **test partition** for a final, independent test of the performance of the model.

XLMiner provides several utilities for partitioning a database. The Standard Data Partition window is shown in Figure 6.4 (Data Mining▶Partition▶Standard Partition). All the variables in the database are listed so that some or all of them can be partitioned. Records can be selected for the various partitions in one of two ways: randomly or by using a partition variable. The random option is most commonly used as it ensures that the partitions are unbiased samples from the given data. Records will then be assigned randomly to each partition using either default percentages specified in the window or percentages specified by the user. A partition variable is created by the user and would typically be a categorical variable with codes for the training and validation partitions.

The Data Partition utility creates a new worksheet with information on how the partitioning was carried out and the records for the various partitions collected together. Figure 6.5 shows a typical Data Partition Sheet. The actual data records begin in row 19 with the training partition. Of the first ten records in the original database, records numbered 1, 4, 5, 6, 9, and 10 fall in the training partition; records 2, 3, 7 and 8 appear below in the validation partition.

XLMiner also provides two specialized utilities for partitioning: Time Series Partition and Partition with Oversampling. The Time Series Partition utility is used to divide data with a time element into early and late samples. Partition with Oversampling is used for classification tasks where the class of interest, say those who purchase an expensive car, is rare in the database. This utility allows the analyst to develop a training sample in which the rare attribute is much more common, thus leading to better models.

FIGURE 6.4 Standard Data Partition Window



6.2.3 Performance Measures

The ultimate goal of data analysis is to predict the future. When the task at hand is to classify new instances—for example, to predict whether a registered voter will actually vote on Election Day—we naturally measure predictive accuracy by the percentage of instances correctly classified. When the task is numerical prediction—say, predicting the

FIGURE 6.5 Data Partition Sheet

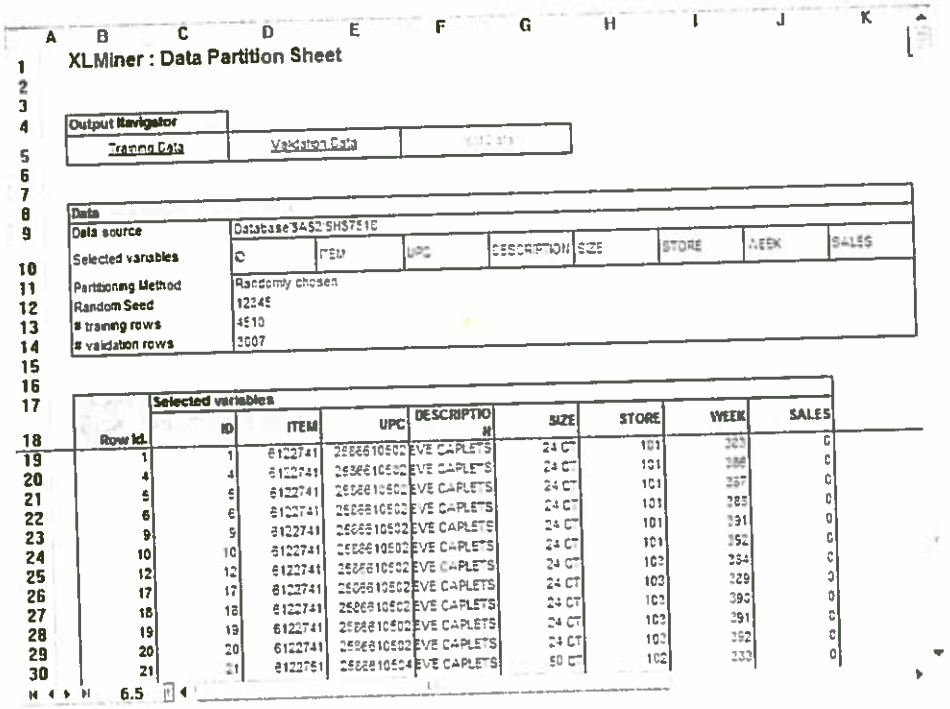


FIGURE 6.6 Classification Matrix

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	326	94
0	40	140

Error Report			
Class	# Cases	# Errors	% Error
1	420	94	22.38
0	180	40	22.22
Overall	600	134	22.33

number of votes received by the winner in each state—predictive accuracy will reflect the numerical differences between the actual outcomes and the predictions.

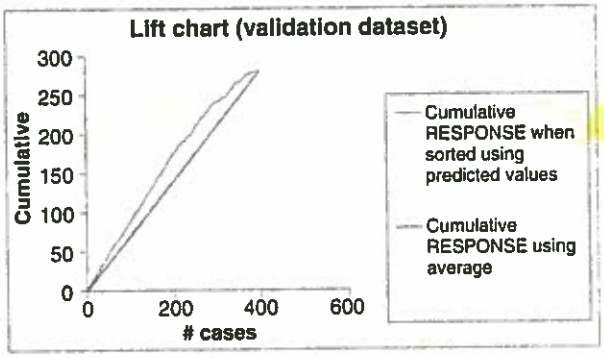
For classification, the predictive accuracy of a model on any given set of data is usually summarized in a **classification matrix** (sometimes called a confusion matrix). Figure 6.6 shows a typical classification matrix from XLMiner.

This chart shows that 600 records were in the dataset. The actual outcome variable (e.g., Vote/No Vote) for each record was either 0 or 1. Likewise, the model predicted the outcome as either 0 or 1. The four combinations of these outcomes are summarized in the upper panel of the chart. We see that 326 records were actually 1s and predicted 1s; 140 were actually 0s and predicted 0s. The errors are in the off-diagonal cells: 40 records were actually 0s but were predicted 1s; 94 were actually 1s but were predicted 0s. These outcomes are summarized in the lower panel, where the overall error rate is reported as 22.33 percent ((40 + 94)/600).

Another commonly-used summary of the predictive performance of a classification model is the **lift chart**. These charts are most helpful if we are interested in the accuracy of our classification on a particular *subset* of the records. Many classification methods generate a probability for each record that it is classified as a 1. Then a rule is used to assign a prediction (0 or 1) to this probability. The rule usually is: if the probability estimate exceeds 0.5, the prediction is 1; if not, the prediction is 0. The lift chart provides a comparison of the predictive performance of a model on those records with the highest probability of being a 1. The performance of the algorithm is compared to that of a naïve model, in which the classification is based solely on the average number of 1s in the data.

Figure 6.7 shows a typical lift chart. The lower line, which is always a straight line, shows the performance of the naïve model. The slope of this line is given by the percentage of 1s in the data. For example, if 1s represent 25 percent of the records, this line has a slope of 0.25, and the naïve model classifies records by assigning every fourth one as a 1. The upper curve represents the performance of the algorithm. It is based on the probabilities computed for each record, sorted from highest to lowest. This curve goes up one unit for every record in this sorted list that correctly predicts a 1, and it goes to the right one unit for every record incorrectly classified as a 0. The more accurate the model is in predicting 1s, the higher the predicted response curve is above the naive response curve.

FIGURE 6.7 Lift Chart



A closely-related summary of predictive performance is the **decile-wise lift chart**. This type of chart is also created by ranking all the records in terms of the probability of being classified as a 1, from highest to lowest probability. The records are then divided into deciles (groups of 10 percent), and the *factor* by which each decile outperforms a classification based on the naïve model is calculated. For example, if in the top 10 percent we have 100 1s, and the underlying percentages would predict 25, this factor is 4. Figure 6.8 shows a typical decile-wise lift chart. It

FIGURE 6.8 Decile-Wise Lift Chart

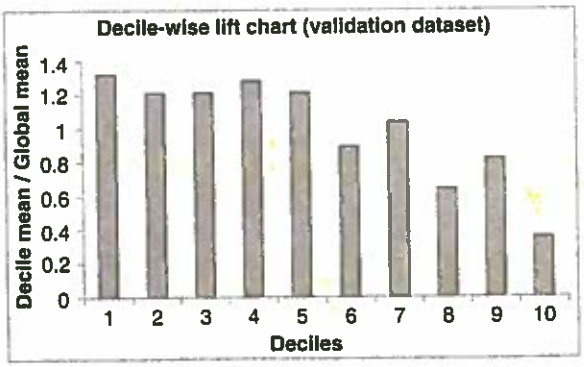


FIGURE 6.9 Performance Measures for Numerical Prediction

Total sum of squared errors	RMS Error	Average Error
94.73245727	0.729523691	0.181623702

shows that the top ten percent of records outperform the average assignment by a factor of about 1.3.

Classification matrices and lift charts focus on different aspects of the performance of an algorithm. The classification matrix treats each record and type of error equally, and gives a summary measure of the overall accuracy of the algorithm. Both types of lift charts, by contrast, focus on that subset of records for which the probability of being a 1 is highest. This is relevant when, for example, we have a limited budget for marketing to customers and want to identify that subset with the highest probability of purchasing.

When the task is prediction, model performance can be evaluated by a number of alternative measures. All of them make use of the difference (or error, e_i) between the actual value on each record and the predicted value. XLMiner routinely reports three summary measures: Total sum of squared errors, Root-mean-squared error (RMS Error), and Average error (Figure 6.9).

These are calculated as follows:

$$\text{Total sum of square errors} = \sum e_i^2$$

$$\text{RMS Error} = \text{SQRT} \sum [(1/n) e_i^2]$$

$$\text{Average Error} = \sum e_i$$

Each of these measures can be used to compare the predictive performance of alternative models since lower error is always better. Average error can be misleading, however, because positive and negative errors can cancel each other out, resulting in an unrealistically low overall error. Probably the most useful of these three error measures is the RMS error because it is an average error *in the same units* as the underlying variable, and by squaring the errors we penalize large errors relatively more than small ones.

6.3 THE *K*-NEAREST NEIGHBOR METHOD

The motivating idea behind the *k*-Nearest Neighbor method is to base the classification of a new case (or the numerical prediction for a new case) on those records in the database that are *most similar* to the new case. In the extreme, if there were one record in the database that was identical in every field to the new case, we could use the known classification of that record as our classification for the new case. But in a database with a large number of predictors, it would be unlikely that any record would have exactly the same values on all variables as the new case. This is why the *k*-Nearest Neighbor method is based on similar, not identical, records. Once similar records are identified, the method computes the most common class among those records and takes that as its classification for the new case.

An example will make this approach clearer. Consider the task of classifying customer orders as delivered on time or delivered late. The outcome variable of interest is DELIVERY, which takes on the values On Time and Late. We have a database of past customer orders with one field representing the delivery outcome and a number of other fields describing the order itself, the delivery mode, the customer, and so on. To predict

whether a new order will be delivered on time or late, we compare its values on all fields except DELIVERY (which is unknown) to similar records in the database. (Later, we elaborate on the meaning of "similar.") If we find five records that are similar, and four are On Time while one is Late, we predict that the new order will be On Time.

The *k*-Nearest Neighbor approach can be used both for classification and for numerical prediction. In most cases, the classification task is based on *majority voting* among similar records; the prediction task is based on *averaging* the known values of the outcome variable for similar records.

As simple as this idea is, it is embedded in some very familiar products. One example is Pandora, a music radio service on the Internet. Pandora allows its users to build customized lists of songs. Pandora's Music Genome Project uses an algorithm very similar to the *k*-Nearest Neighbor method to identify new songs that should appeal to the user and highlight them to be either included or excluded from the existing song list.

6.3.1 Overview of the *k*-Nearest Neighbor Algorithm

There are three major questions that must be addressed to implement the *k*-Nearest Neighbor algorithm. These are:

- How to define similarity between records?
- How many neighboring records to use?
- What classification or prediction rule to use?

We illustrate the implications of these choices using the situation of a college that wants to predict whether incoming students will successfully graduate. It has a database of past students that includes their high school grade-point average (GPA) and their standardized test scores (TEST SCORE), as well as whether they graduated (Y) or failed to graduate (N). Figure 6.10 shows a graph of the data, with GPA on the horizontal axis, TEST SCORE on the vertical axis, and the graduation result as Y or N. The solid dot represents a new student whose graduation outcome we are trying to predict.

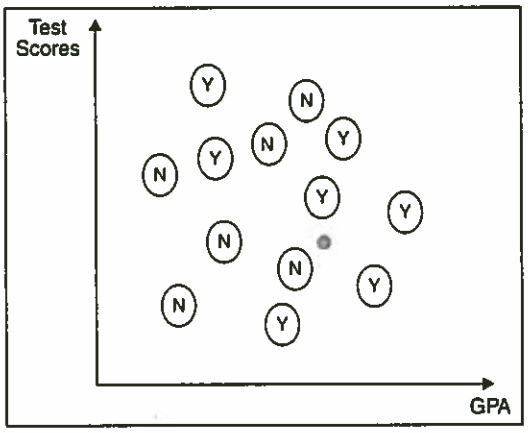


FIGURE 6.10 Data on Graduation Success

In this example we define similarity between records by the straight-line distance between the corresponding points in the graph. Figure 6.11 shows the four nearest neighbors to the new record, using this definition of similarity.

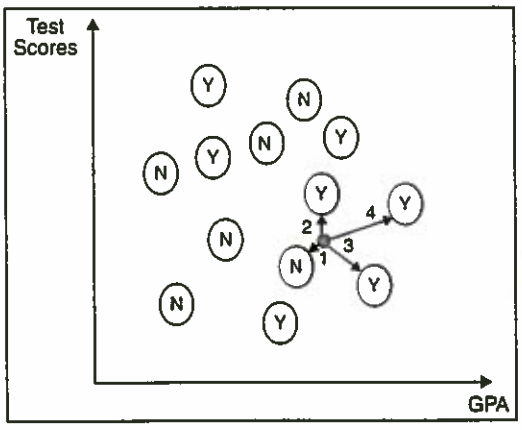


FIGURE 6.11 Four Nearest Neighbors to New Record

If we rely only on the closest record, which lies to the lower left of the new record and is N, our classification would be N. If we take into account the *two* closest records, including the second closest which is directly above the new record, we would have to break a tie between one Y and one N. If we use the three closest records, we would have two Ys and one N; our prediction by majority vote would be Y. And if we include the four closest records, we would have three Ys, one N, and a classification of Y. This example shows that the prediction we make generally depends on the number of nearest neighbors we take into account. The prediction also depends on the definition of distance and the classification rule.

In most applications of the *k*-Nearest Neighbor approach, the predictor variables are measured on different numerical scales. In our example,

the values for GPA are around 4.0 and those for Test Scores are around 700. A GPA of 3.0 is very different from one of 4.0, while a test score of 699 is almost identical to one of 700. To ensure that one variable does not dominate the others merely because of its scale, *k*-Nearest Neighbor algorithms typically call for normalizing the predictors. The usual normalization is by *z*-scores (see Chapter 5, Section A.5.5 for more details).

The distance between records can also be defined in different ways. The most common is the **Euclidean distance**, where the distance between two records X and Y (with values x_i and y_i on the i^{th} variable, respectively) is given by

$$Distance = [(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2]^{1/2} \quad (6.1)$$

As with all data mining methods, over-fitting is a major issue with the *k*-Nearest Neighbor approach. If we use just one neighbor ($k = 1$), our results are sure to be very sensitive to the particular data we are using; we are certainly over-fitting to the noise. At the other extreme, we could use all n records in the database ($k = n$). In this case, if the majority of the records were 1s on the outcome variable, we would predict a 1 for *every* new case. Here, we would be over-smoothing the data because all new cases would receive the same classification. Somewhere in between $k = 1$ and $k = n$ lies the best choice for k. It is common to choose the k that maximizes the classification accuracy on the validation sample.

6.3.2 An Application of the *k*-Nearest Neighbor Algorithm

We illustrate the application of the *k*-Nearest Neighbor approach using a database with 30 variables and 1,000 records on the credit rating assigned to previous applicants for credit (*German Credit.xlsx*).* The task is to use this dataset to create a method for rating the credit of new applicants. The outcome variable is Response (column AF), which takes on the value 0 when the credit rating was poor and 1 when it was good. The variables in the database are described in the following table:

Variable Name	Description	Type	Coding
OBS#	Observation number	Categorical	
CHK_ACCT	Checking account status	Categorical	0: <= 0 DM 1: 0< . . . <200 DM 2: => 200 DM 3: no account
DURATION	Duration of credit in months	Numerical	
HISTORY	Credit history	Categorical	0: no credits taken 1: all credits at this bank paid back duly 2: existing credits paid back duly till now 3: delay in paying off in the past 4: critical account
NEW_CAR	Purpose of credit	Binary	0: No 1: Yes
USED_CAR	Purpose of credit	Binary	0: No 1: Yes
FURNITURE	Purpose of credit	Binary	0: No 1: Yes
RADIO/TV	Purpose of credit	Binary	0: No 1: Yes
EDUCATION	Purpose of credit	Binary	0: No 1: Yes
RETRAINING	Purpose of credit	Binary	0: No 1: Yes
AMOUNT	Credit amount	Numerical	
SAV_ACCT		Categorical	0: < 100 DM

* Source: UCI Machine Learning Repository (http://archive.ics.uci.edu/ml).

	Average balance in savings account		1: 100< . . . <500 DM 2: 500< . . . <1000 DM 3: => 1000 DM 4: unknown/no savings account
EMPLOYMENT	Present employment Since	Categorical	0: unemployed 1: < 1 year 2: 1 <= . . . < 4 years 3: 4 < . . . < 7 years 4: >= 7 years
INSTALL_RATE	Installment rate as % disposable income	Numerical	
MALE_DIV	Applicant is male and divorced	Binary	0: No 1: Yes
MALE_SINGLE	Applicant is male and single	Binary	0: No 1: Yes
MALE_MAR_WID	Applicant is male and widowed	Binary	0: No 1: Yes
CO_APPLICANT	Application has a co-applicant	Binary	0: No 1: Yes
GUARANTOR	Application has a quarantor	Binary	0: No 1: Yes
PRESENT_RESIDENCE	Present residence since . . . years	Categorical	0: < 1 year 1: 1< . . . <= 2 years 2: 2< . . . <= 3 years 3: > 4 years
REAL_ESTATE	Applicant owns real estate	Binary	0: No 1: Yes
PROP_UNKNOWN	Applicant owns no property (or unknown)	Binary	0: No 1: Yes
AGE	Age in years	Numerical	
OTHER_INSTALL	Applicant has other installment plan credit	Binary	0: No 1: Yes
RENT	Applicant rents	Binary	0: No 1: Yes
OWN_RES	Applicant owns residence	Binary	0: No 1: Yes
NUM_CREDITS	Number of existing credits at this bank	Numerical	
JOB	Nature of job	Categorical	0: unemployed/unskilled – non-resident 1: unskilled – resident 2: skilled employee/official 3: management/self-employed/highly qualified
NUM_DEPENDENTS	Number of people for whom liable to provide maintenance	Numerical	
TELEPHONE	Applicant has phone in his or her name	Binary	0: No 1: Yes
FOREIGN	Foreign worker	Binary	0: No 1: Yes
RESPONSE	Credit rating is good	Binary	0: No 1: Yes

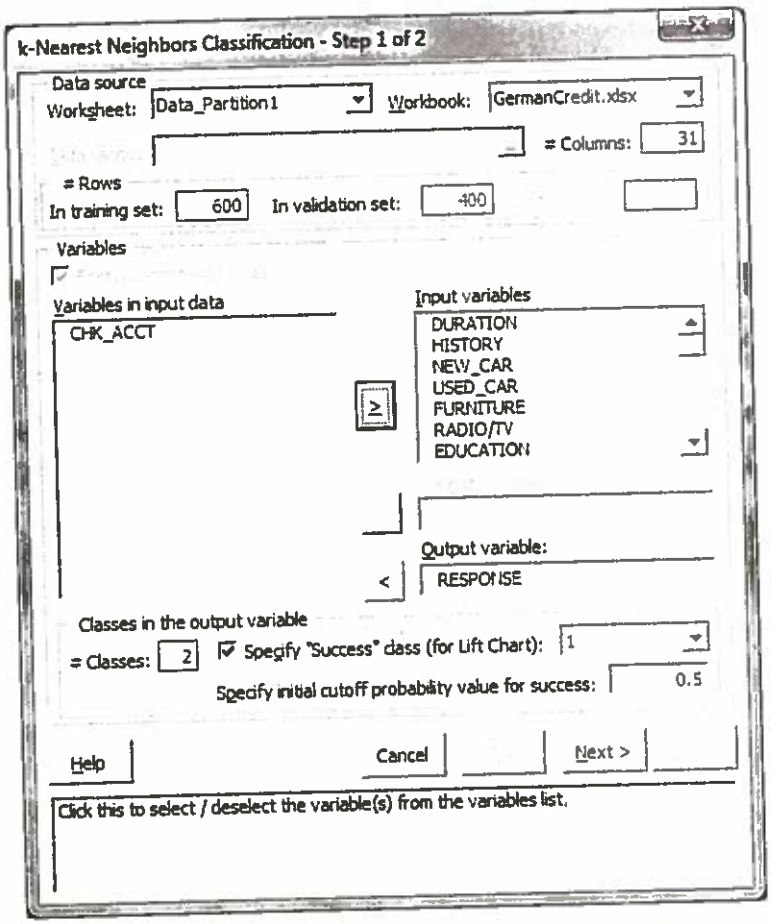


FIGURE 6.12 First *k*-Nearest Neighbor Classification Window

The first step is to partition the database into 600 training records and 400 validation records (using the Standard Partition utility in XLMiner). Then, select the *k*-Nearest Neighbor algorithm for classification: Data Mining▶Classify▶*k*-Nearest Neighbors. (If the task were numerical prediction rather than classification, we would select Data Mining▶Predict▶*k*-Nearest Neighbors.) This brings up a window similar to that shown in Figure 6.12.

In this window, we select RESPONSE as the Output variable and move all the other relevant variables into the Input variables category. We also confirm that there are two classes for the output variable: "Success" is a 1 (representing good credit), and the probability cutoff for the classification rule is 0.5.

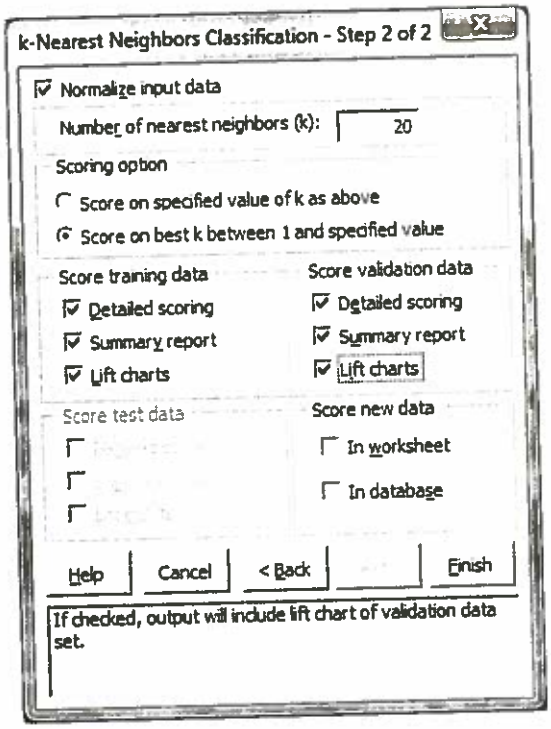


FIGURE 6.13 Second *k*-Nearest Neighbor Classification Window

When we click on OK, a second window appears, shown in Figure 6.13. Here, we chose to normalize the variables (using *z*-scores), we request up to 20 nearest neighbors (all cases from 1 to 20 will be computed), and we ask for the best alternative for *k* between 1 and 20 to be scored on both the training and validation partitions.

The Validation Error Log is shown in Figure 6.14. The first column shows the values of *k* that were tested from 1 to 20. The second and third columns show the error rates on the training and validation records, respectively. The error is zero on the training

FIGURE 6.14 Validation Error Log for *k*-Nearest Neighbor Algorithm

Validation error log for different k

Value of k	% Error Training	% Error Validation	
1	0.00	42.00	
2	20.17	31.75	
3	19.83	35.50	
4	24.83	33.00	
5	23.83	35.25	
6	25.17	33.25	
7	25.67	33.75	
8	25.83	33.00	
9	25.50	34.25	
10	27.33	33.50	
11	26.00	33.75	
12	27.83	32.00	
13	27.83	32.25	
14	27.33	32.00	
15	26.67	33.00	
16	27.17	30.75	<--- Best k
17	27.00	31.50	
18	27.00	30.75	
19	27.83	32.00	
20	28.17	31.00	

data for $k = 1$ because in this case the training data is matched against itself, so no errors can occur. The error on the training data starts out around 20 percent for low values of k and gradually increases as k increases. The error on the validation data shows a quite different pattern. In this case, the validation data is matched with the training data. The error rate for $k = 1$ is quite high, suggesting a high degree of noise in the data. The error rate then falls with k, reaching a minimum for $k = 16$ ($k = 18$ gives the same error rate). However, the error rate is between 30 and 35 percent for most values of k.

Figure 6.15 shows the classification matrix for both the training and the validation records. Since we specified that these matrices be displayed for the "best" value of k, they are computed for $k = 16$, which has the lowest error on the validation data. As we would expect, the error rate is somewhat higher on the validation data than on the training data. In

FIGURE 6.15 Classification Matrices for Training and Validation Partitions

Training Data scoring - Summary Report (for k=16)

Cut off Prob.Val. for Success (Updatable)	0.5

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	411	9
0	154	26

Error Report			
Class	# Cases	# Errors	% Error
1	420	9	2.14
0	180	154	85.56
Overall	600	163	27.17

Validation Data scoring - Summary Report (for k=16)

Cut off Prob.Val. for Success (Updatable)	0.5

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	270	10
0	113	7

Error Report			
Class	# Cases	# Errors	% Error
1	280	10	3.57
0	120	113	94.17
Overall	400	123	30.75

addition, the great majority of the errors (92 percent, 113/123) are cases in which the actual class was 0, but our model predicted 1.

It seems likely in this situation that the consequences of misclassifying someone who is a poor credit risk as a good credit risk (Actual = 0, Predicted = 1) far outweigh those of the opposite error. Yet this is the error our method makes most often. We can change these error rates by changing the cutoff probability in the classification rule, which was set at 0.5 initially. If we set the cutoff to 0.75, the error rate goes up somewhat, but the costly errors are far less common. In particular, the total number of errors on the 400 records in the validation sample increases from 123 to 146, but the percentage in the high-risk category drops from 92 to 23 percent.

6.3.3 Strengths and Weaknesses of the *k*-Nearest Neighbor Algorithm

The major advantage of the *k*-Nearest Neighbor algorithm is its simplicity. It requires essentially no assumptions as to the form of the model and few assumptions about the parameters. Only one parameter is estimated (k), and we can test all reasonable values for that parameter. This method has been found to perform well in situations where there is a large training database, and where there are many combinations of predictor variables that distinguish the class outcomes.

One drawback is that the method provides no information on which predictors are most effective in making a good classification. The only practical approach is to edit the database in advance, removing predictors that would intuitively appear to have little value. The method also can suffer from long computational times because the number of records required for a sufficiently large training database increases faster than linearly with the number of variables.

6.4 THE NAÏVE BAYES METHOD

The Naïve Bayes method is similar to the *k*-Nearest Neighbor approach in that it bases the classification of a new case on similar records in the database. However, it is restricted to situations in which all the predictor variables are categorical. (Numerical variables can be converted to categorical variables by binning; for details see Chapter 5, Section A.5.2.) With categorical variables, several records in the database are usually *identical* to the new case, and the more prevalent class among these records is taken as the classification for the new case.

Consider the task of predicting a school's admissions decision based on a single test score. The test scores range from 450 to 800 and are binned into three categories: Low (450–600), Medium (601–700), and High (701–800). The following data are available on 300 past decisions:

		Test Score		
		Low (450–600)	Medium (601–700)	High (701–800)
Admission	Y	10	30	60
	N	90	70	40

How would we categorize a new applicant with a test score of 650 using the logic of the Naïve Bayes method? This score falls in the Medium range, and the proportion of records that were admitted in that range is 30 percent, while 70 percent were not admitted. Since the proportion not admitted is higher, that is the category we choose for this applicant. The same logic applies when we have many more predictor variables: find the records that have the same values on all predictors and assign the category that is most prevalent among those records.

Spam filtering is an everyday application in which the Naïve Bayes algorithm is often used. The task is to classify individual emails as Spam or Not Spam. A very large database of actual emails is assembled in which a record is a single email, and the fields reflect whether each distinct word appears in that email. All the predictors are categorical

variables with two values: Word Appeared, and Word did not Appear. Experts also classify each record as Spam or Not Spam. New emails are then classified by breaking them down into individual words and identifying those records with identical words. The algorithm makes its classification based on whether Spam is more frequent than Not Spam in that subset of records.

6.4.1 Overview of the Naïve Bayes Algorithm

To understand how the Naïve Bayes approach works (and why it is "naïve") requires a bit of algebra. We assume we are attempting to classify an outcome variable Y that takes on the values 1 and 0. Our data consists of two (categorical) predictors X_1 and X_2. (For example, we might want to classify companies as to whether they will default on their debts, given data on prior defaults and company size.) In practice, we would have many more predictors, but the main points can be made with just two.

We use the notation $P(Y = 1|X_1,X_2)$ to represent the probability that the outcome is 1 given specific values for the two predictors X_1 and X_2 (e.g., the probability a given company defaults given that it had No Prior Defaults and was Small). Although we use the language of probability, we measure these probabilities using proportions of records in the database.

Bayes Rule, which is a standard probability formula, allows us to rewrite this probability in a more useful form:

$$P(Y = 1|X_1,X_2) = P(X_1,X_2|Y = 1) * P(Y = 1)/$$
$$[P(X_1,X_2|Y = 1) * P(Y = 1) + P(X_1,X_2|Y = 0) * P(Y = 0)] \quad (6.2)$$

In words, this says that we can rewrite the probability $P(Y = 1|X_1,X_2)$ as a ratio. In the numerator, we have the probability of getting the data given the outcome, $P(X_1,X_2|Y = 1)$, multiplied by the probability of getting the outcome, $P(Y = 1)$. If we think in terms of proportions of records in the database, the probability $P(X_1,X_2|Y = 1)$ is the proportion of records with the specific values for X_1 and X_2 among all those for which $Y = 1$. And the probability $P(Y = 1)$ is just the proportion of records in the database with $Y = 1$. The denominator consists of two terms, the numerator and its complement for $Y = 0$.

Using another probability formula, the chain rule, we can rewrite $P(X_1,X_2|Y = 1)$ in this form:

$$P(X_1,X_2|Y = 1) = P(X_1|Y = 1) * P(X_2|Y = 1,X_1) \quad (6.3)$$

In words, this says that the probability of getting the specific outcomes for X_1 and X_2, given $Y = 1$, is the product of the probability of getting X_1 given the outcome $Y = 1, P(X_1|Y = 1)$, and the probability of getting X_2 given *both* $Y = 1$ *and* X_1, $P(X_2|Y = 1,X_1)$.

Now, we can begin to see the problem that arises in estimating these probabilities using proportions of the database: the number of records satisfying the conditions gets smaller, and therefore the probability estimated from the proportion of records is more subject to noise. In fact, probabilities of the form $P(X_2|Y = 1,X_1)$ will get closer and closer to zero as the number of predictors increases. For example, if we had 30 predictors in our database, we would need to estimate $P(X_{30}|Y = 1,X_1,X_2,X_3,\ldots X_{29})$, and there might well be no records at all satisfying these conditions.

An example will make this clearer. Suppose we are attempting to predict the vote of a specific individual. We have information on the previous votes of other individuals, including data on race, gender, ethnic background, political affiliation, state of residence, whether they voted in the last two elections, marital status, and number of children. The individual in question is white, male, Hispanic, Democrat, from Ohio, voted in the last election, did not vote in the election before that, is divorced, and has one son. To predict his vote, we look in the database for similar voters, but we find no one with exactly his values on all predictors: what can we do? Surely the probability that he will vote is not zero, even though the formulas above would give that result.

One solution to this problem is to assume that knowing the specific value for X_1 does not change the probability for X_2. In other words,

$$P(X_2|Y = 1, X_1) = P(X_2|Y = 1) \tag{6.4}$$

This assumption is known as **class-conditional independence.** It makes it much less likely that the constituent probabilities will be zero because it is based on larger fractions of the database. Remarkably, this assumption provides the basis for a powerful classification algorithm even though it is almost never precisely true and is often grossly inaccurate. This is why the method is known as *Naïve* Bayes. The justification for making this assumption is simply that it works. But it works only if our goal is classification, not if we are trying to accurately estimate probabilities of class membership.

Here's an example with which we can compare the accurate approach using the full Bayes Rule, and the approximate approach using the Naïve Bayes approach assuming class-conditional independence. The task is to classify firms as to whether they will default on their debts (Default = Yes) or not (Default = No). The data we have is the prior default history and company size:

Prior Default	*Company Size*	*Default*
DYes	Small	Yes
DNo	Small	Yes
DNo	Large	Yes
DNo	Large	Yes
DNo	Small	Yes
DNo	Small	Yes
DYes	Small	No
DYes	Large	No
DNo	Large	No
DYes	Large	No

We take a new firm that has defaulted in the past (Prior Default = DYes) and is small (Company Size = Small) and estimate the probability it will not default in the future (Default = No). Using the full Bayes Rule we have:

$$\begin{aligned} P(\text{No}|\text{DYes}, \text{Small}) &= P(\text{No}) * P(\text{DYes}|\text{No}) * P(\text{Small}|\text{DYes}, \text{No}) / \\ &\quad [P(\text{No}) * P(\text{DYes}|\text{No}) * P(\text{Small}|\text{DYes}, \text{No}) + \\ &\quad P(\text{Yes}) * P(\text{DYes}|\text{Yes}) * P(\text{Small}|\text{DYes}, \text{Yes})] \\ &= (4/10 * 3/4 * 1/3) / \\ &\quad [4/10 * 3/4 * 1/3 + 6/10 * 1/6 * 1/1] \\ &= 0.50 \end{aligned}$$

Using the Naïve Bayes approach we have:

$$\begin{aligned} P(\text{No}|\text{DYes}, \text{Small}) &= P(\text{No}) * P(\text{DYes}|\text{No}) * P(\text{Small}|\text{No}) / [P(\text{No}) * P(\text{DYes}|\text{No}) \\ &\quad * P(\text{Small}|\text{No}) + P(\text{Yes}) * P(\text{DYes}|\text{Yes}) * P(\text{Small}|\text{Yes})] \\ &= (4/10 * 3/4 * 1/4) / \\ &\quad [4/10 * 3/4 * 1/4 + 6/10 * 1/6 * 4/6] \\ &= 0.53 \end{aligned}$$

It happens that the probabilities calculated in these two ways are close in this case, but that is not generally true. For a full picture, we show in the table below the estimated probabilities under both the full Bayes Rule and Naïve Bayes for all four combinations of predictors:

	Full Bayes	*Naïve Bayes*	*Classification (cutoff = 0.5)*
P(No|DYes, Small)	0.50	0.53	No
P(No|DYes, Large)	1.00	0.87	No
P(Yes|DYes, Small)	0.00	0.07	Yes
P(Yes|DYes, Small)	0.33	0.31	Yes

As the last column in the table shows, the classification we would make for each case is the same whether we use the the full Bayes Rule or the Naïve Bayes method. For example, if we use a cutoff probability of 0.5, then whenever the estimated probability is 0.5 or higher, we classify the case as No, otherwise as Yes. Both sets of probabilities therefore give the same classification outcomes because both are either above or below 0.5. With a larger number of predictors, we can expect that the estimated probabilities would differ more and more, but the classifications would continue to be similar.

6.4.2 An Application of the Naïve Bayes Algorithm

We illustrate the application of the Naïve Bayes approach using a database (*UniversalBank.xlsx*)* containing 14 variables and 5,000 records on bank customers who were made a personal loan offer. The outcome variable is Personal Loan, which takes on the value 0 when the offer was rejected and 1 when it was accepted. The task is to use this dataset to create a method for predicting whether new cases will respond positively to a loan offer. The variables in the database are described in the table below.

Variable	Description
ID	customer ID
Age	age in completed years
Experience	years of professional experience
Income	annual income ($000)
ZIPCODE	home zip code
Family	size of family
CCAvg	average monthly spending on credit cards ($000)
Education	1: undergraduate; 2: graduate; 3: advanced degree
Mortgage	value of house mortgage ($000)
Personal Loan	did customer accept personal loan offer? (1: Yes; 0: No)
Securities Account	did customer have securities account? (1: Yes; 0: No)
CD Account	did customer have CD account? (1: Yes; 0: No)
Online	did customer use online banking? (1: Yes; 0: No)
CreditCard	did customer use Universal credit card? (1: Yes; 0: No)

The first step is to examine the database for numerical variables because this method requires categorical variables. We find five: Age, Experience, Income, CCAvg, and Mortgage, which we convert to categorical variables (using 3 bins in each case) with the XLMiner utility Bin Continuous Variables. Next, we partition the data using the Standard Partition.

We then select the Naïve Bayes algorithm (Data Mining▶Classify▶Naïve Bayes). The first input window appears as in Figure 6.16. We specify Personal Loan as the outcome variable and all the other variables (except ID and zip code, which we exclude as irrelevant) as inputs (being careful to include only the binned versions of Age, Experience, Income, CCAvg, and Mortgage). There are two classes for the outcome variable and the cutoff for classification is 0.5.

The second input window (Figure 6.17) gives us a choice for calculating class probabilities: Choose `According to relative occurrences in training data`. The final input window allows us to choose the level of detail in the output. We specify complete results on both the training and the validation data (Figure 6.18).

The results of running the Naïve Bayes algorithm are summarized in the classification matrices shown in Figure 6.19. The overall classification error rate is 6.37 percent on the training data and 5.95 percent on the validation data. In both cases, there are more than twice as many Actual 1s classified as 0s than the reverse.

Figure 6.20 shows the actual and predicted classes for each record (in this case, for the training data). This report allows us to examine errors such as record number 19, for which the estimated probability of success was 0.434, so the prediction was 0, but the actual value was 1. Since the values of all the predictor variables are also shown here, this provides a convenient means for trying to understand the reasons behind classification errors.

* Source: Cytel, Inc. 2005.

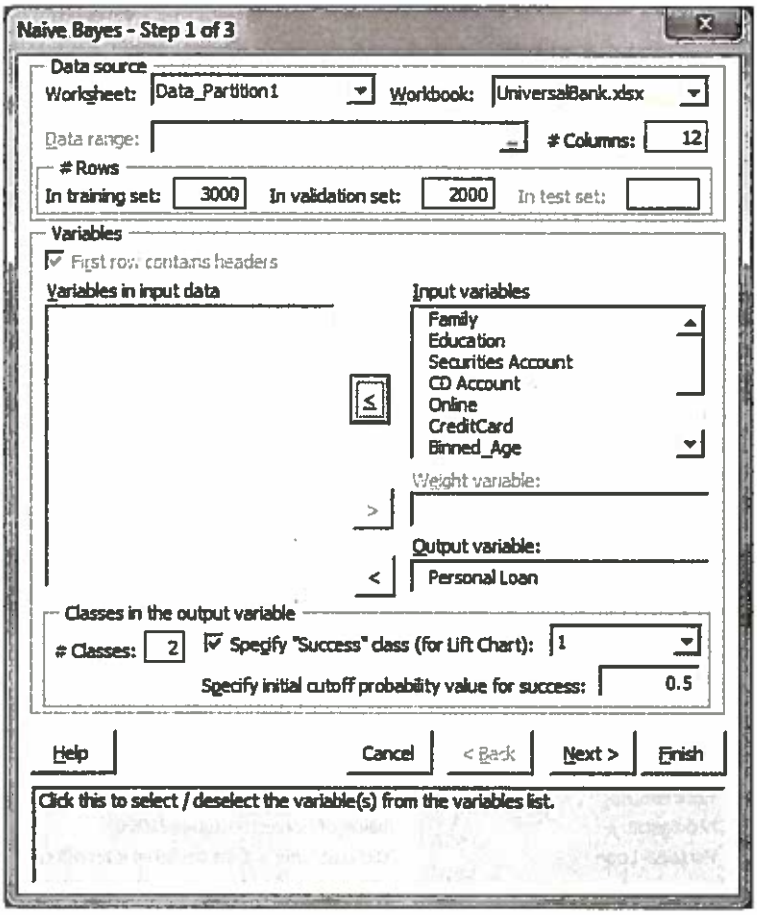


FIGURE 6.16 First Naïve Bayes Window

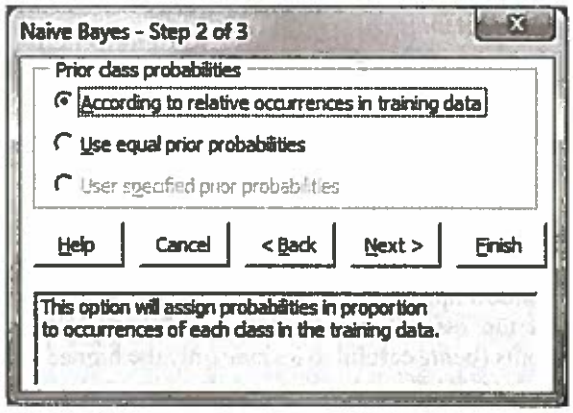


FIGURE 6.17 Second Naïve Bayes Window

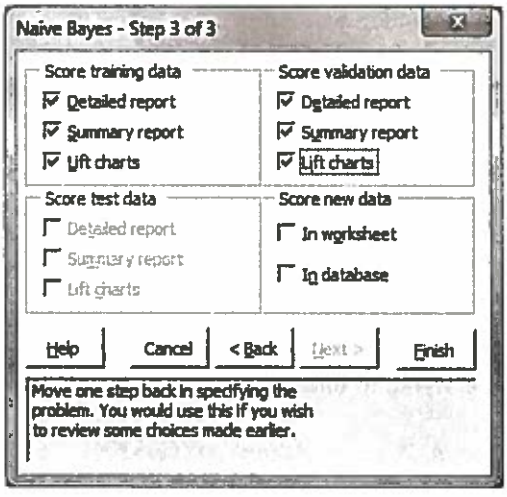


FIGURE 6.18 Third Naïve Bayes Window

Finally, the lift chart and decile-wise list chart provide visual summaries of the accuracy of this approach (Figure 6.21). In particular, the decile-wise chart shows that if we were to target the top 10 percent of the validation sample, we would achieve about a 700 percent improvement in classification accuracy over random classification.

6.4.3 Strengths and Weaknesses of the Naïve Bayes Algorithm

The Naïve Bayes algorithm is remarkably simple and often gives classification accuracy as good as or better than more sophisticated algorithms. It requires no assumptions other than class-conditional independence, and we have seen that the algorithm can perform well even when that assumption is violated.

Several downsides to this approach should be kept in mind. One is that a very large number of records may be needed to obtain

FIGURE 6.19 Classification Matrices for Training and Validation Data

Training Data scoring - Summary Report

Cut off Prob.Val. for Success (Updatable)	0.5

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	156	130
0	61	2653

Error Report			
Class	# Cases	# Errors	% Error
1	286	130	45.45
0	2714	61	2.25
Overall	3000	191	6.37

Validation Data scoring - Summary Report

Cut off Prob.Val. for Success (Updatable)	0.5

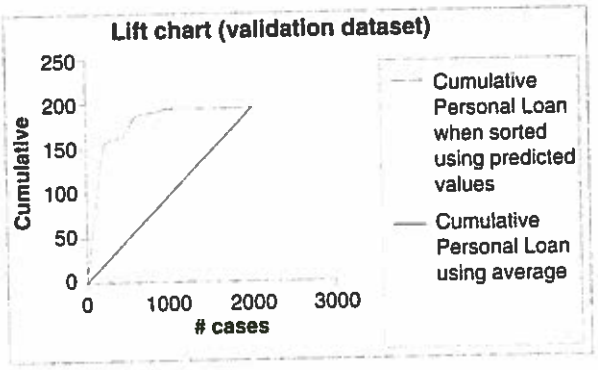
Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	108	86
0	33	1773

Error Report			
Class	# Cases	# Errors	% Error
1	194	86	44.33
0	1806	33	1.83
Overall	2000	119	5.95

good results. Another potential pitfall is that the method estimates a probability of zero for any new case with a predictor value missing from the training database. To use our earlier example, if the new voter whose vote we wish to predict has 15 children, and no record in the training data has 15 children, the result will be a probability of zero (regardless of the values of the other predictors). (This problem can be somewhat mitigated by careful binning.) Finally, this method is only suitable for classification, not for estimating class probabilities.

Row Id.	Predicted Class	Actual Class	Prob. for 1 (success)	Family	Education	Securities Account	CD Account	Online	CreditCard	Binned_Age	Binned_Experience	Binned_Income	Binned_CCAvg	Binned_Mortgage
1	0	0	0.001214055	4	1	1	0	0	0	1	1	2	2	1
4	0	0	0.454926457	1	2	0	0	0	0	1	1	3	3	1
5	0	0	0.003462409	4	2	0	0	0	1	1	1	2	2	1
6	0	0	0	4	2	0	0	1	0	1	1	1	1	3
9	0	0	0.004869015	3	2	0	0	1	0	1	1	2	1	3
10	0	1	0.46225673	1	3	0	0	0	0	1	1	3	3	1
12	0	0	0.003771046	3	2	0	0	1	0	1	1	2	1	1
17	1	1	0.60414012	4	3	0	0	0	0	1	1	3	3	3
18	0	0	0.006030049	4	1	0	0	0	0	2	2	2	3	1
19	0	1	0.434003183	2	3	0	0	0	0	2	2	3	3	1
20	0	0	0	1	2	1	0	0	1	3	3	1	1	1
21	0	0	0	4	2	0	0	1	0	3	3	1	1	3
23	0	0	0.000927473	1	1	0	0	1	0	1	1	2	2	3
26	0	0	0	3	1	0	0	1	0	2	2	1	1	3
27	0	0	0.002242151	4	3	0	0	0	0	2	2	2	1	1
29	0	0	0.02048296	1	3	0	0	1	1	3	3	2	3	1
30	1	1	0.910003999	1	2	0	1	1	1	1	1	3	3	1
31	0	0	0	1	3	0	0	1	0	3	3	1	2	3
32	0	0	0	1	2	0	0	1	0	2	2	1	2	1
35	0	0	0.003767977	4	3	0	0	1	0	1	1	2	2	1

FIGURE 6.20 Predicted and Actual Classes for Each Record



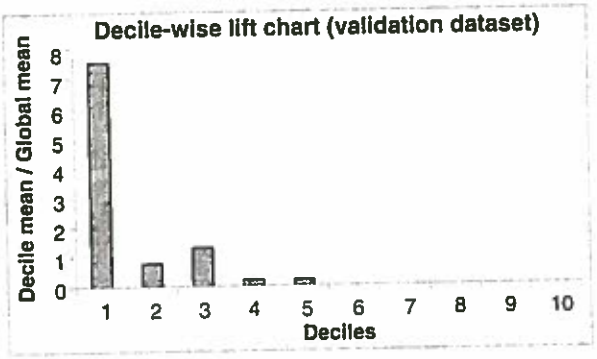


FIGURE 6.21 Lift Charts for Naïve Bayes Algorithm

6.5 CLASSIFICATION AND PREDICTION TREES

The tree approach to classification is based on the observation that there are subsets of the records in a database that contain mostly 1s or 0s. If we can identify these subsets, then we can classify a new record by determining the majority outcome in the subset it most resembles. The same approach can be used for numerical prediction although the prediction is made by *averaging* the values of the outcome variable in the subset, not by majority voting as in classification. (This class of algorithms is often referred to by the acronym CART, for Classification and Regression Tree. We prefer the term *prediction* to *regression* because the word *regression* often is taken to imply linear regression, an entirely different algorithm.)

When compared to other methods, classification and prediction trees are particularly simple to understand and explain to others. The process used to create them can be described in several equivalent ways: as recursive partitioning, as creating trees, or as defining a set of logical rules.

Imagine the task is to predict the purchase behavior ($Y = 1$ or $Y = 0$) of individuals about whom we know three facts: their age, income, and education. A typical classification tree could take the form of Figure 6.22. In this tree, circular nodes represent the partitioning of records, while rectangular nodes represent collections of records with similar values for the outcome variable. Starting at the top, the tree indicates that we first want to distinguish records for which INCOME exceeds 107.6 from those for which INCOME is less than this value. The records in the second group (INCOME $\leq$ 107.6) are immediately classified $Y = 0$. The ones in the first group (INCOME $>$ 107.6) are then further split by those for

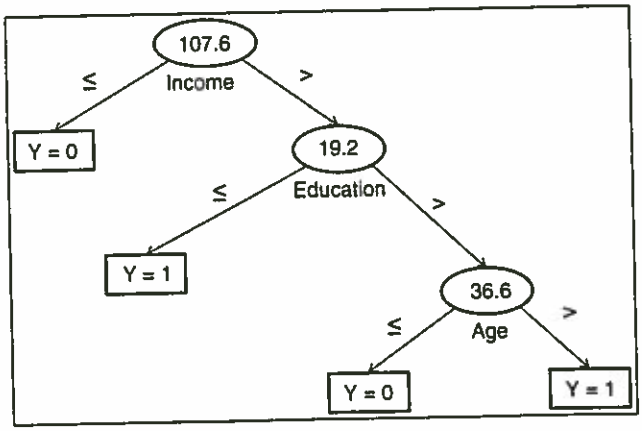


FIGURE 6.22 Classification Tree

whom EDUCATION exceeds 19.2 from those for whom EDUCATION does not exceed 19.2. The second group is classified $Y = 1$, while the first group is once again divided, this time by AGE, using an age of 36.6 as the split point. These two remaining groups can now be classified as either $Y = 0$ or $Y = 1$. Given this tree, we classify a new record by starting at the top and taking the branch at each circular node that corresponds to the new record's values until we reach a rectangular node, where the classification is made.

This example suggests qualitatively how a classification tree algorithm works. First, it must decide on which variable to make the initial split (INCOME in our example). Then, it must decide at what value to make the split (107.6 in our example). This initial split defines two subsets of records. These subsets are then assigned a classification or split further (usually with different variables and split points). This process is carried out recursively until the subsets of records are sufficiently homogeneous, by which we mean they contain either mostly 1s or mostly 0s.

6.5.1 Overview of Classification and Prediction Trees

The general outline of the classification and prediction tree algorithm is simple to describe. We first describe the approach used for classification with numerical predictor variables. Then, we describe how to use categorical predictor variables for classification, and finally how this algorithm is applied to numerical prediction.

The following steps describe the algorithm:

1. Pick a predictor variable.
2. Sort its values from low to high.
3. Define a set of **split points** as the midpoints between each pair of values.
4. For each split point, divide the records into two sets: those having values above and those below the split point.
5. Evaluate the **homogeneity** of the records in each of these subsets. (Homogeneity measures the extent to which the records are mostly 1s or 0s.)
6. Repeat this process for all split points for this variable.
7. Choose the split point for this variable that gives the most homogeneous subsets.
8. Repeat this process for all variables.
9. Split on the variable with the highest homogeneity.
10. Repeat the entire process for each subset of records created by this split.

This basic outline is used for all tree algorithms. However, categorical variables are treated somewhat differently. Instead of sorting them as we do with numerical variables, all possible subsets are created and evaluated for homogeneity. For a categorical variable with two outcomes, Y and N, there is only one possible split point: between Y and N. For a categorical variable with three outcomes, say H, M, and L, there are three possible splits: {H} and {M, L}; {M} and {H, L}; and {L} and {H, M}.

When the task at hand is numerical prediction, not classification, essentially the same algorithm can be used. The major difference is that the "homogeneity" of a subset of records is usually measured by the variance (sum of squared deviations from the mean) of the outcome variable for those records.

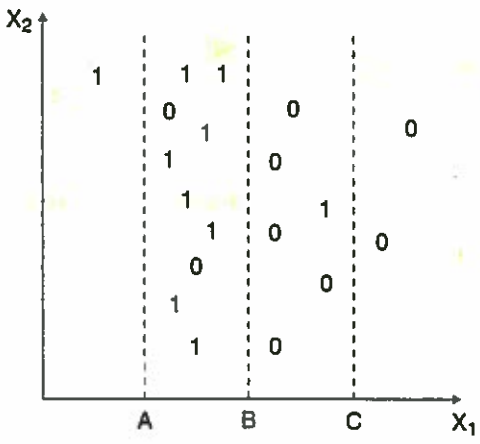


FIGURE 6.23 Illustration of Homogeneity

Measuring the degree of homogeneity of groups of records is an essential element of this approach. Homogeneity, again, reflects the extent to which a subset is predominantly either 1s or 0s. The goal of the algorithm is to create subsets of records with a high degree of homogeneity.

An example will make this concept clearer. Consider the case where we have two predictor variables, X_1 and X_2, and the outcomes are Y and N. Figure 6.23 shows a plot of 19 records. We consider three alternative split points for X_1: A, B, or C. (The algorithm considers all possible split points; here we illustrate just these three.)

Each time we split, we form two subsets of records: the group to the right and the group to the left of the split point. If we split at A, the group to the left has one record with a Y; the group to the right has 18 records with 9 Ys and 9 Ns. Splitting at B we have 9 Ys and 2 Ns to the left; 1 Y and 7 Ns to the right. Finally, splitting at C we have 10 Ys and 7 Ns to the left; 0 Ys and 2 Ns to the right. These results are displayed in the following table:

	Left Group		Right Group	
Split Point	Ys	Ns	Ys	Ns
A	1	0	9	9
B	9	2	1	7
C	10	7	0	2

If we had to choose among these three split points, which would create the most homogeneous subsets? The group on the right for split point A is problematic because it is evenly divided between Ys and Ns, and is therefore not homogeneous. Likewise, split point C creates a highly nonhomogeneous group on the left. Only point B creates two relatively homogeneous subsets, and it is the one we would choose among these three.

In XLMiner, homogeneity is computed using a variant of the **Gini index**. If we define p_0 as the proportion of records in the subset with $Y = 0$ and p_1 as the proportion of records with $Y = 1$, then the Gini Index is

$$\text{Gini Index} = 1 - p_0^2 - p_1^2 \tag{6.5}$$

This formula reaches its minimum value of zero when either p_0 or p_1 is 1. These are the extremes of perfect homogeneity. Its maximum occurs when $p_0 = p_1 = 0.5$, which is the least homogeneous possibility. Because the most homogeneous cases are ones with the lowest values of the Gini index, the algorithm seeks to *minimize* the Gini score.

The Gini Index for each of these subsets is shown in the table below. The overall index is the weighted average of the indices for the left and right groups, where the weights are the number of records in each group. These numerical results confirm our previous observation that B is the split point for which *both* subsets are relatively homogenous.

Split Point	Left Group	Right Group	Weighted Average
A	0.00	0.50	0.47
B	0.33	0.22	0.26
C	0.48	0.00	0.43

Over-fitting is a particular problem with classification and prediction trees since we can achieve a *perfect* fit on any dataset simply by continuing to divide records until the final subsets contain only one record each. Such a tree will perfectly classify training data, but it is likely to perform poorly on validation data. This is because smaller subsets contain fewer records, which are less representative of the underlying patterns and more influenced by noise. The solution to this problem is to **prune** the tree—to remove nodes at the bottom that divide small subsets into even smaller subsets, until the resulting tree no longer over-fits the data. This requires partitioning the database into three partitions: a training partition, a validation partition, and a test partition. The procedure is this: fit a full tree to the training data (for a perfect fit), prune this tree using validation data to minimize classification error, test the resulting tree on the test data.

6.5.2 An Application of Classification and Prediction Trees

We illustrate the application of the classification and prediction tree approach using a database with 25 variables and 3,120 records on the donation histories of members of a nonprofit organization (*Fundraising.xlsx*)*. The outcome variable is TARGET-B (column W), which takes on the value 1 when the individual has donated and 0 otherwise. The task is to use this data to create a method for predicting the donations from new members. The variables in the database are described in the following table:

* Source: Shmueli, G., N. R. Patel, and P. C. Bruce. 2010. *Data Mining for Business Intelligence*. New Jersey: John Wiley & Sons, page 387.

Variable	Description
ZIP	Zip code group
HOMEOWNER	0: not homeowner, 1 homeowner
NUMCHLD	number of children
INCOME	household income
GENDER	0: Male, 1: Female
WEALTH	wealth rating from 0-9 lowest to highest
HV	average home value in neighborhood
ICmed	median family income in neighborhood
ICavg	average family income in neighborhood
IC15	percent of population earning less than $15,000 in neighborhood
NUMPROM	number of promotions received to date
RAMNTALL	lifetime gifts to date
MAXRAMNT	largest gift to date
LASTGIFT	most recent gift
TOTALMONTHS	months from last donation
TIMELAG	months between first and second donation
AVGGIFT	average size of donation
TARGET_B	0: nondonor, 1: donor

The first step is to examine the predictor variables in the database. We note that ZIPCODE has been converted from a 5-digit code to four dummy variables by region. The WEALTH variable is an index variable, ranging from 1 to 9, depending on median family wealth and area population. This is technically an ordinal categorical variable, but we will interpret it as numerical for the purposes of developing a tree model. The remaining variables are either numerical or binary, and therefore in an appropriate format for the classification tree algorithm.

We select the Classification Tree algorithm (Data Mining▶Classify▶Classification Tree). The first input window appears as in Figure 6.24. We specify TARGET-B as the

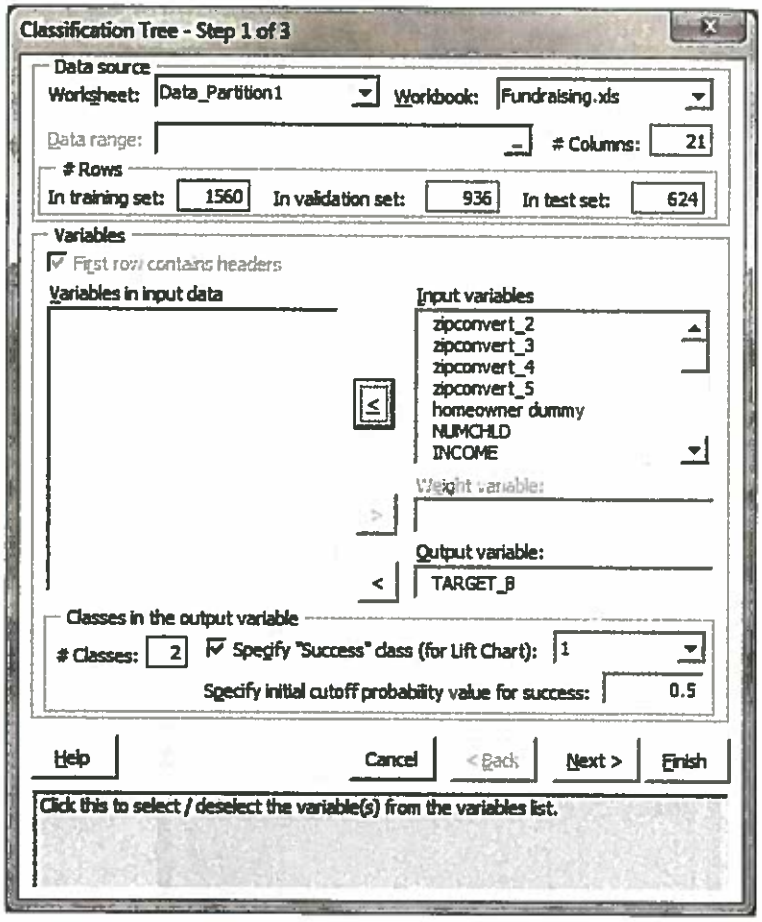


FIGURE 6.24 First Classification Tree Window

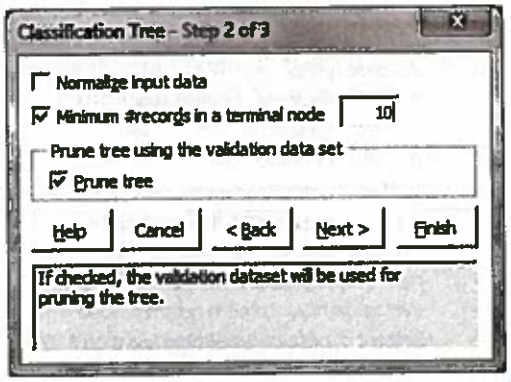


FIGURE 6.25 Second Classification Tree Window

outcome variable and all the other variables as inputs. There are two classes for the outcome variable and the cutoff for classification is 0.5.

On the second input window (Figure 6.25) we specify 10 for `Minimum #records in a terminal node`. This choice controls the depth of the tree that is fit to the training data. With 1,560 records in the training partition we might expect that it will take a very complex tree to separate the records into perfectly homogeneous subsets, and doing so will lead to extreme over-fitting. Thus we allow the algorithm to stop when the subsets contain 10 records. We also check the box `Prune tree` so the trees constructed on the training data will be pruned on the validation data.

Finally, on the third input window (Figure 6.26) we choose options in the top panel (`Trees`) for how the trees will be displayed in the output. In most cases we choose a small number (in this case 5) for `Maximum # levels to be displayed` so that the displayed trees are not too complex. We also choose to report all the available detail for all three partitions: training, validation, and test.

The full classification tree developed on the training partition is shown in Figure 6.27. The first split is on the variable MAXRAMNT, which measures the maximum donation given to date. The split value is 14.5; 727 records are less than or equal to this value and go down the left branch, while 833 records are greater than this value and go down the right branch ($727 + 833 = 1,560$, which is the total number of records in the partition). The left branch then splits on INCOME, at a value of 4.5. with 521 records going to the left and 206 going to the right ($521 + 206 = 727$, which is the number of records splitting left above this node). The right branch splits on RAMNTALL, which is the total of gifts lifetime. The split value is 102.5, with 451 records going to the left and 382 going to the right.

The training log (Figure 6.28) shows how the error rate declines as we add nodes to the tree using the training data. With zero nodes the error rate is 50 percent; after 119 nodes are added the error rate declines to 24.8 percent. However, we are more interested

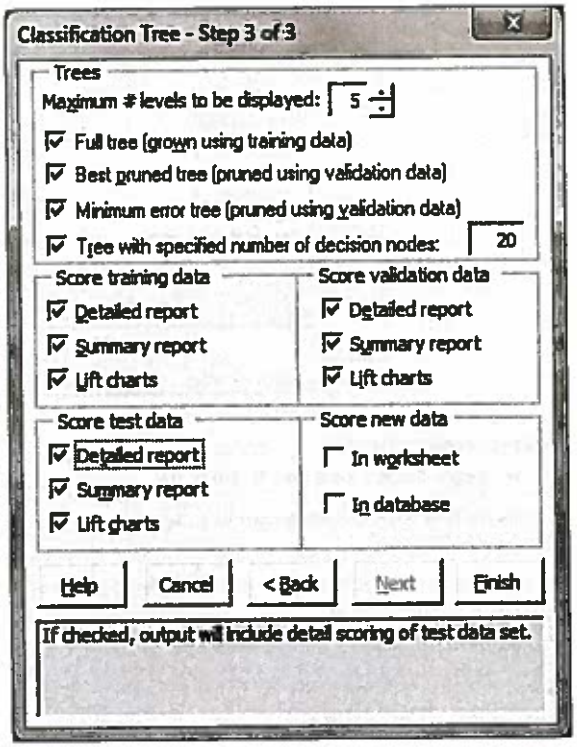


FIGURE 6.26 Third Classification Tree Window

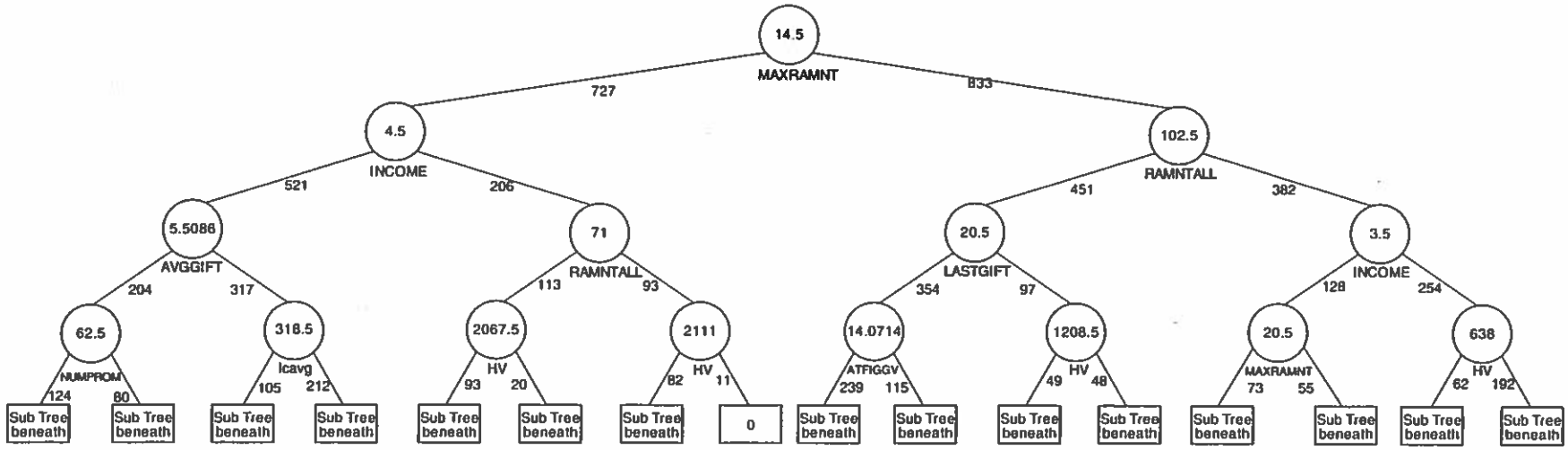


FIGURE 6.27 Full Classification Tree Fit to Training Data

Training Log (Growing the full tree using training data)

# Decision Nodes	% Error
0	49.62
1	43.78
2	43.78
3	43.53
4	43.08
5	43.08
6	43.08
7	42.05
8	42.05
9	40.9
10	40.9
11	40.71
12	40.13
13	40.13
14	39.81
15	39.17
16	39.04
17	39.04
18	38.72
19	38.08
20	37.89
21	37.69
22	37.69
23	37.37
24	36.86
25	36.86
26	36.79
27	36.28
28	36.28
29	36.03
30	35.96
31	35.96
32	35.45
33	35.45
34	35.13
35	34.87
36	34.87
37	34.36
38	34.29
39	34.23
40	33.97
41	33.85
42	33.85
43	33.27
44	33.27
45	32.88
46	32.88
47	32.88
48	32.56
49	32.31
50	31.99
51	31.79
52	31.54
53	31.47
54	31.47
55	30.9
56	30.9
57	30.84
58	30.64
59	30.64
60	30.64
61	30.45
62	30.45
63	30.45
64	30.06
65	29.94
66	29.94
67	29.66
68	29.62
69	29.62
70	29.62
71	29.62
72	29.36
73	29.36
74	28.97
75	28.91
76	28.91
77	28.65
78	28.59
79	28.59
80	28.59
81	28.53
82	28.59
83	28.59
84	28.4
85	28.08
86	28.08
87	27.82
88	27.69
89	27.63
90	27.63
91	27.37
92	27.37
93	26.99
94	26.73
95	26.73
96	26.73
97	26.73
98	26.73
99	26.73
100	26.6
101	26.6
102	26.15
103	26.15
104	26.15
105	26.03
106	25.9
107	25.9
108	25.83
109	25.58
110	25.58
111	25.32
112	25.32
113	25.32
114	25.32
115	25.06
116	25.06
117	25.06
118	25.06
119	24.81

FIGURE 6.28 Training Log

FIGURE 6.29 Prune Log Using Validation Data

# Decision Nodes	% Error			
119	51.282051			
118	51.282051			
117	51.282051			
116	51.282051			
115	51.282051			
114	51.282051			
113	51.282051			
112	51.282051			
111	51.282051			
110	51.282051			
109	51.282051			
108	51.282051			
107	51.282051			
106	51.282051			
105	51.282051			
104	51.282051			
103	51.282051			
102	51.282051			
101	51.282051			
100	51.282051			
20	43.376068			
19	43.376068			
18	43.376068			
17	44.230769			
16	44.551282			
15	43.376068			
14	43.376068			
13	43.376068			
12	43.376068			
11	43.376068			
10	42.948718			
9	42.948718			
8	42.948718			
7	42.094017			
6	42.094017			
5	42.094017			
4	42.094017	<-- Min. Err. Tree	Std. Err.	0.016137415
3	43.696581			
2	43.696581			
1	43.696581	<-- Best Pruned Tree		
0	49.679487			

in the error rates on the validation partition, which are displayed in Figure 6.29. As we prune away branches from the 119 used in the full tree, the error rate on the validation data declines. It reaches its minimum with four nodes; this is the Minimum Error Tree. The Best Pruned Tree, however, has only one node. (The Best Pruned Tree generally has fewer nodes than the Minimum Error Tree. It accounts for the variability in the estimates of the error rate.)

These results suggest that the full tree is extremely over-fit to the training data, and that the best trees for classifying new records have a small number of branches (but do not perform terribly well). This conclusion is supported by the classification matrices (Figure 6.30), which show that the misclassification is lowest on the training data but increases for both the validation and test data.

Finally, we examine the lift charts for the test data (Figure 6.31). As expected, the lift chart shows that the classification tree performs only slightly better than the average, and the decile-wise lift chart shows that this approach is about 10 percent better than a naïve model on the first decile of the records.

FIGURE 6.30 Classification Matrices for all Three Partitions

Training Data scoring - Summary Report (Using Full Tree)

Cut off Prob.Val. for Success (Updatable)	0.5

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	604	182
0	205	569

Error Report			
Class	# Cases	# Errors	% Error
1	786	182	23.16
0	774	205	26.49
Overall	1560	387	24.81

Validation Data scoring - Summary Report (Using Best Pruned Tree)

Cut off Prob.Val. for Success (Updatable)	0.5

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	255	216
0	193	272

Error Report			
Class	# Cases	# Errors	% Error
1	471	216	45.86
0	465	193	41.51
Overall	936	409	43.70

Test Data scoring - Summary Report (Using Best Pruned Tree)

Cut off Prob.Val. for Success (Updatable)	0.5

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	171	132
0	146	175

Error Report			
Class	# Cases	# Errors	% Error
1	303	132	43.56
0	321	146	45.48
Overall	624	278	44.55

6.5.3 Strengths and Weaknesses of Classification and Prediction Trees

The classification and prediction tree approach has the great advantage of being easy to understand and explain to others. Its results are transparent and can be interpreted as explicit If-Then rules. These rules can be directly programmed for automatic classification or prediction of new cases.

In addition to these advantages, this approach is based on few assumptions and works well even with missing data and outliers. Finally, it identifies the most important predictor variables because the early splits are based on those variables that increase the homogeneity of the subsets of the records the most.

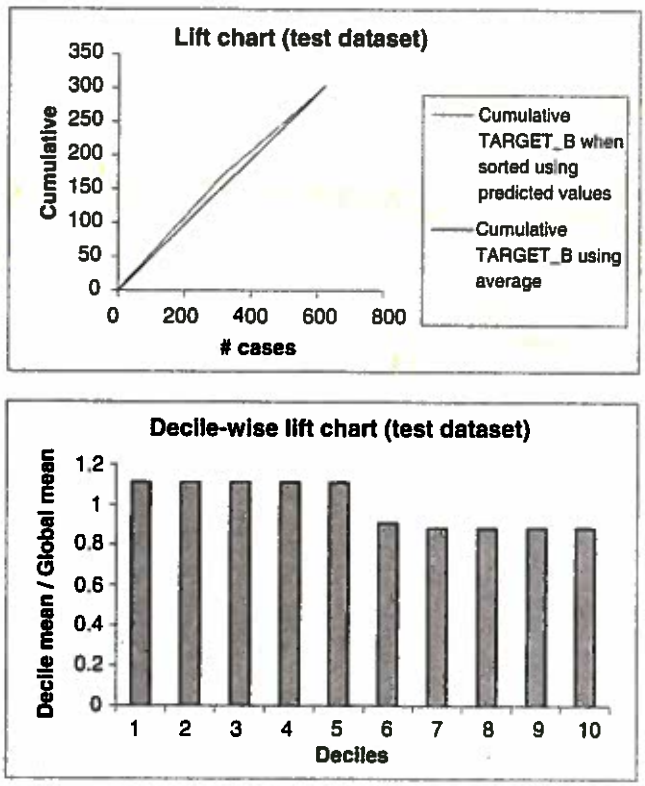


FIGURE 6.31 Lift Charts for Test Data

Among the few disadvantages of trees is this: accurate results can require very large databases. In addition, this approach allows partitioning of only individual variables, not pairs or groups of variables. This approach will miss relationships between variables; the solution is to define new variables by combining existing ones.

A limitation specific to XLMiner is that only binary categorical variables are allowed. Thus categorical variables with more than two values must be replaced with dummy variables.

6.6 MULTIPLE LINEAR REGRESSION

Linear regression is one of the most widely-used tools from classical statistics, and when used appropriately, it is an important tool in the business analyst's toolkit. This method is ubiquitous in the natural and social sciences, where it is more often used for *explanatory modeling* than for *predictive modeling*. The goal of explanatory modeling is to determine whether specific variables influence the outcome variable. (It is also called inferential modeling because the goal is to *infer* from the data the existence and strength of relationships between variables.) Here are some typical questions of this type that linear regression has been used to answer:

- Do the data support the claim that women are paid less than men in comparable jobs?
- Is there evidence in the data that price discounts and rebates lead to higher long-term sales?
- Do data support the idea that firms that outsource their manufacturing overseas have higher profits?
- Do the available epidemiological data support the notion that smoking is associated with lung cancer?

Linear regression was originally developed to answer questions of scientific interest. For example, agricultural experimenters wanted to know which treatments lead to higher crop yields, medical experts wanted to know whether certain drugs lead to improvements in health, and economists wanted to identify the key determinants of firms' profitability. In most of these cases, the models were not used for prediction. The focus instead was on determining which specific factors could reliably be said to influence the outcome variable.

When linear regression was developed, data were scarce, so all the available data were typically used to fit the model. This left no unused data with which to test the predictive accuracy of the model. Indirect methods, therefore, had to be developed that would suggest which predictor variables had the strongest relationship to the outcome variable. These methods, known as inferential statistics, have been used for decades to help select the variables to include in the best fitting model.

Our focus in this chapter, by contrast, is on *predicting* the outcome variable, not primarily on *explaining* it. This difference in emphasis causes us to use linear regression in a somewhat different way than it is used for explanatory modeling. As with other methods, we are less interested in how well the model fits the data than in how well it predicts with new data. This is why we create a model using a training dataset and test its predictive power on a validation dataset. A model that fits the training data very well but has poor predictive accuracy on the validation data would hardly be considered adequate.

6.6.1 Overview of Multiple Linear Regression

Multiple linear regression is a method for making numerical predictions using numerical predictor variables. It cannot be used for classification, and it cannot employ categorical predictor variables. Categorical data can, however, often be transformed into numerical data by using dummy variables (see Chapter 5, Section A.5.3.) (In the next section, we discuss a closely related method, logistic regression, which can be used for classification.)

A linear regression model takes the form

$$Y = b_0 + b_1X_1 + b_2X_2 + \ldots + b_qX_q + e \tag{6.6}$$

where the outcome variable is Y, the explanatory variables are $X_1, X_2, \ldots, X_q$; the parameters to be estimated from the data are $b_0, b_1, b_2, \ldots, b_q$, and e represents a random error term that accounts for all the unobserved factors that influence the outcome Y.

This model is based on two strong assumptions. One is that the effect of each predictor variable X_i is *additive* to those of other predictors. The other assumption is that the effect of each predictor is *proportional* to its value, with the proportionality factor given by the coefficient b_i.

These assumptions may seem excessively restrictive, especially when compared to algorithms such as Classification and Prediction Trees or Naïve Bayes that assume very little about the form of the model or its parameters. However, by suitably transforming the original data we can adapt the linear regression model to a wide variety of situations. A simple example illustrates this point.

The relationship between the speed of a vehicle and the distance it takes to bring that vehicle to a stop is crucial to traffic safety. Figure 6.32 shows a set of experimental data on this relationship. It is clear from these data that stopping distance increases with speed at an increasing rate: the relationship is not linear. A straight line fit to these data, as shown in the figure, consistently under- or over-predicts stopping distance.

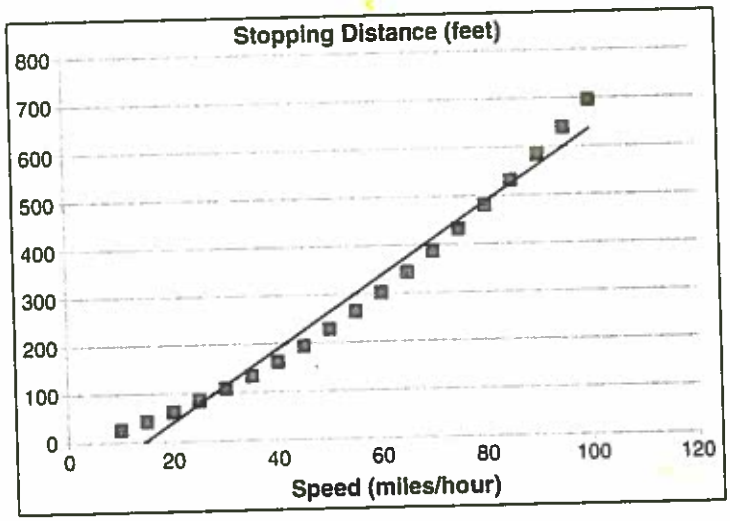


FIGURE 6.32 Stopping Distance as a Function of Speed

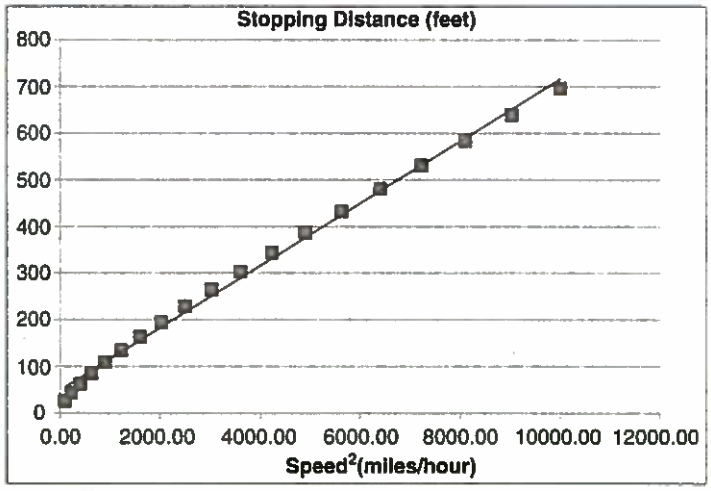


FIGURE 6.33 Stopping Distance as a Function of Speed Squared

However, the figure does suggest that the relationship between stopping distance and speed might be quadratic, so that if we plotted stopping distance against the *square* of speed it would appear linear. Figure 6.33 confirms this hunch and shows that a straight line fits this transformed data very well.

The lesson to draw from this example is that we should avoid using the raw data at our disposal in a linear regression model until we have confirmed that a linear relationship is plausible. Creating a scatterplot of each predictor against the outcome variable is an important step in this process. If the relationship does not appear to be linear, either from the data or from background knowledge, we should transform the predictor (as we did in the example above by squaring it) to more closely satisfy the linearity assumption.

In multiple linear regression, we assume the data are given and the task is to determine the "best" values for the regression coefficients b_i. This becomes a form of optimization. The objective function to be minimized is the sum (over all the records) of the squared errors between the data and the regression function. The decision variables are the coefficients b_i. In effect, the algorithm adjusts the values for the b_i until it finds the smallest possible total error. Fortunately, under the assumption of linearity, this process is straightforward: unless the predictor variables are almost perfectly correlated, all that is required is solving a set of simple algebraic equations. Unlike more complex search algorithms, multiple linear regression always finds an answer quickly and reliably.

The practical challenge in using multiple linear regression is not in solving for the parameters but in determining which predictor variables to use, particularly when a large database with scores of potential predictor variables is available. One approach would be to use *all* the available predictors; after all, why throw away perfectly good data? But this approach is likely to lead to over-fitting and eventually to poor predictions. To see why this is so, consider that every predictor X_i is correlated to the outcome Y to some degree: either weakly, strongly, or somewhere in between. If we add dozens of predictors to our model, some of which are weakly correlated, we are fitting the model to the noise represented by the weakly correlated variables. As always with over-fitting, the result will be poor predictions.

There are three complementary approaches to selecting predictors for a regression model:

- pre-processing the data
- use of inferential statistics
- search to select the best subset of predictors

In pre-processing the data, we ask a series of questions to help us eliminate potentially problematic predictor variables. If, for example, a predictor X on its face does not have any influence on the outcome Y, then we eliminate it from our data at the outset. For example, sunspots surely do not influence store sales. In the same manner, we would eliminate from further consideration variables in our dataset that contain numerous missing values, or that are expensive to collect, or that are likely to be recorded inaccurately, or that will not be available at the time predictions must be made. Finally, we eliminate variables that are highly correlated with other predictor variables because including them will add little to the overall predictive accuracy of the model.

The second approach is to use inferential statistics to determine predictor variables whose apparent influence on the outcome variable may be due to chance. This approach requires running the multiple regression algorithm and examining the p-value of each regression coefficient. These p-values can be used to answer this question: Could this value for b_i have arisen due to random sampling if the actual parameter value was zero? (If $b_i = 0$ then the predictor variable has no influence on the outcome variable.) The p-value measures the probability in repeated sampling from the data that an observed value at least as large as b_i could have arisen even though the true value is zero. A p-value above 0.05 is customarily taken to indicate that the variable in question should be eliminated from the model.

This approach to variable selection cannot be followed mechanically; p-values provide only a rule of thumb. Certainly it is a mistake to believe that a very low p-value indicates that the variable in question is important in the model or that the regression coefficient is accurate. The five percent cutoff is nothing more than a convention, and there are many instances where we would choose to keep a variable in a model even though it failed this test. On the other hand, a high p-value can be taken as a warning that this coefficient deserves careful scrutiny and might not belong in the model. The final arbiter, of course, is predictive accuracy on the validation data, so we can always compare the accuracy of models with and without a particular variable.

The third approach to selecting predictors is to systematically search over different subsets of variables from all of those in the dataset and attempt to determine the best collection. If we had to do this by hand, it would be impractical because the number of subsets of variables is huge even for a modest number of predictors. But most software for multiple linear regression offers one or more automated approaches to this problem. We will describe the approaches available in XLMiner in more detail in the following section. Here we give a brief overview.

The two most common approaches to automated subset selection are *forward selection* and *backward elimination*. In forward selection, variables are added into the regression model one a time, starting with the one that most improves the fit of the model to the data. Under backward elimination, a model is first run with *all* the variables, and then they are eliminated one by one, starting with the one that makes the smallest contribution to the fit.

As with the use of p-values, these search methods should be used only to provide suggestions for variables to include or exclude. They should never be relied on as providing the final word. Do not expect any of these search methods to serve up the best model automatically. We should examine the contribution each predictor variable makes to the model, and we should always test predictive accuracy on validation data.

6.6.2 An Application of Multiple Linear Regression

We illustrate the application of the multiple linear regression approach using a database with 9 variables and 398 records on the mileage and other characteristics of automobile models (*MPG.xlsx*)*. The outcome variable is MPG, which measures fuel efficiency in miles per gallon. The potential predictor variables include a number of measurements of the physical features of the vehicle, such as horsepower and weight, as well as the year of the model and the region of origin (1 = US, 2 = Europe, 3 = Asia). The task is to use the data to predict the fuel efficiency of a new vehicle. The variables in the database are described in the following table:

Variable	Description
MPG	fuel efficiency in miles per gallon
CYLINDERS	number of cylinders
DISPLACEMENT	engine displacement in cubic inches
HORSEPOWER	engine horsepower
WEIGHT	vehicle weight in pounds
ACCELERATION	acceleration in feet per second squared
YEAR	model year
ORIGIN	country of origin (1 = US, 2 = Europe, 3 = Asia)
NAME	model name

* Source: UCI Machine Learning Repository (http://archive.ics.uci.edu/ml)

We first examine the predictor variables and consider the effect we would expect each to have on mileage. CYLINDERS, DISPLACEMENT, HORSEPOWER, WEIGHT, and ACCELERATION are all numerical variables, and we intuitively expect each of them to have a negative impact on mileage. By a negative impact, we mean that the associated regression coefficient should be negative, so MPG should decrease as each of these variables increases. YEAR is also numerical, and we might expect fuel efficiency to improve with the model year (under the influence of technological change and government regulations) so its coefficient would be positive. ORIGIN is a categorical variable although the values it takes on are numerical, so we must be careful about how we use it in a regression model. To see why this is so, consider that the ORIGIN variable takes on the values 1 for US, 2 for Europe, and 3 for Asia. If we include it in the regression, then the estimated coefficient will measure the effect on MPG of a 1-unit increase in ORIGIN, that is, going from US to Europe, and from Europe to Asia. But we have no reason to believe that these effects are equal. A better approach is to create dummy variables (see Chapter 5, Section A.5.3) for all three possible outcomes. We use only two of the dummy variables in the regression (those for Europe and Asia) because the information contained in the third variable is redundant and adds no information. The coefficients of these dummy variables in the regression will measure the impact on MPG of a vehicle coming from Europe (relative to being from the US), and coming from Asia (also relative to coming from the US).

We partition the resulting dataset and select the multiple regression procedure for the prediction task in XLMiner (Data Mining▶Predict▶Multiple Linear Regression). The first input screen appears as in Figure 6.34.

We select MPG as the Output variable, and all of the remaining variables as inputs other than NAME and ORIGIN_1 (US), which is redundant with the dummies for Europe and Asia.

The second input window (Figure 6.35) gives us the option to force the constant term in the regression to be zero. The constant term reflects the value of the outcome variable when all the predictor variables are zero. Because we have no reason to believe this value is

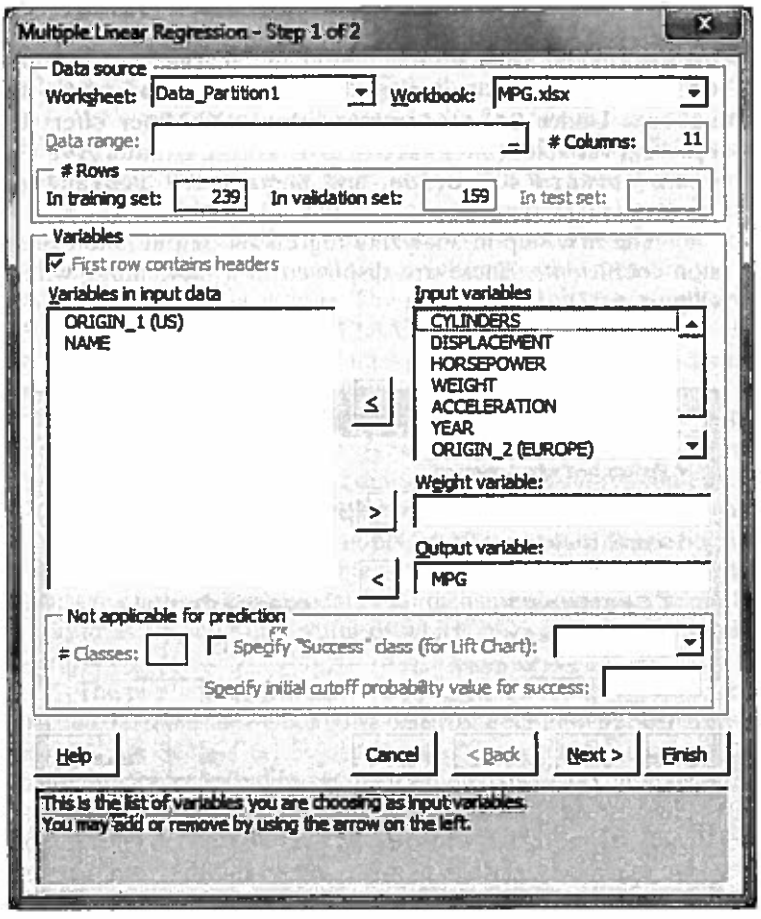


FIGURE 6.34 First Multiple Linear Regression Window

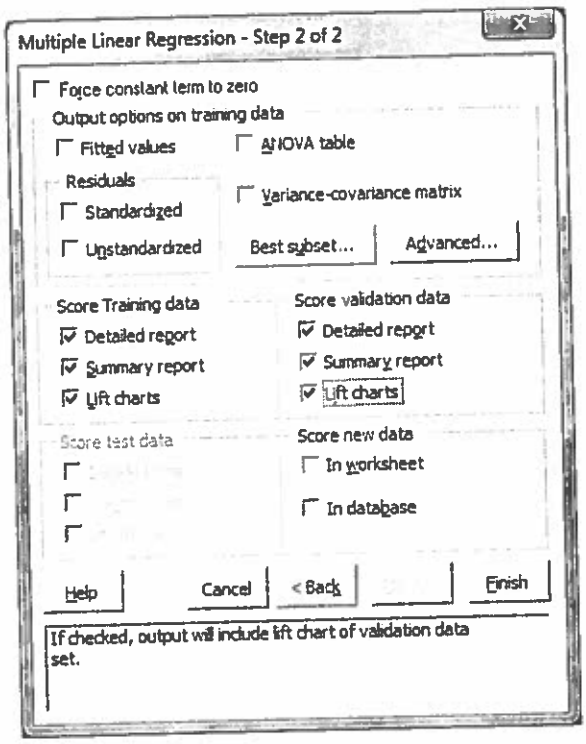


FIGURE 6.35 Second Multiple Linear Regression Window

zero, we leave this box unchecked. This window also gives us options for how the results on the training data will be displayed. We choose to display all the scoring results for both the training and validation datasets.

The `Best subset` option in this window allows us to run the regression algorithm on subsets of the predictor variables. In each case the algorithm will attempt to choose those variables that create the best-fitting model. This is a good procedure to use when we have a large number of predictor variables and we suspect that many of them will not be useful in the final model. Click on this button and another window opens, shown in Figure 6.36. Check `Perform best subset selection` and choose 8 for the `Maximum size of best subset`. Under `Selection procedure`, XLMiner offers five different methods for choosing variables (`Backward elimination`, `Exhaustive search`, `Stepwise selection`, `Forward selection`, and `Sequential replacement`). Accept the default, `Backward elimination`.

The first step in analyzing regression output is to examine the estimated regression coefficients. These are displayed in a table along with their associated p-values (Figure 6.37).

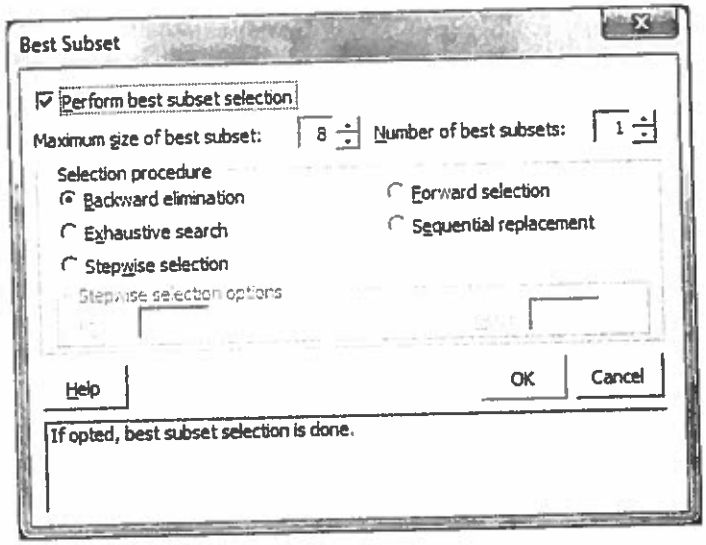


FIGURE 6.36 Best Subset Window for Linear Regression

FIGURE 6.37 Estimated Regression Coefficients and *p*-Values

The Regression Model

Input variables	Coefficient	Std. Error	p-value	SS
Constant term	-10.63756657	5.63866806	0.06048214	135269.7031
CYLINDERS	-0.87675035	0.38984695	0.02546228	9097.974609
DISPLACEMENT	0.02932786	0.01001434	0.00374738	615.5471191
HORSEPOWER	-0.03307875	0.01755075	0.06072474	239.9298706
WEIGHT	-0.00602707	0.00088837	0	479.6585083
ACCELERATION	0.02440948	0.12455896	0.84480882	2.32501674
YEAR	0.70097178	0.06264555	0	1316.237549
ORIGIN_2 (EUROPE)	2.70879459	0.69464511	0.00012651	24.82397842
ORIGIN_3 (ASIA)	3.59014463	0.6924535	0.00000047	280.2293396

The regression equation takes the following form when we substitute in the estimated parameters:

$$\begin{aligned}\text{MPG} = &-10.64 - 0.88\,\text{CYLINDERS} + 0.03\,\text{DISPLACEMENT} - 0.03\\ &\text{HORSEPOWER} - 0.01\,\text{WEIGHT} + 0.02\,\text{ACCELERATION} + 0.70\,\text{YEAR}\\ &+ 2.71\,\text{ORIGIN\_2(EUROPE)} + 3.59\,\text{ORIGIN\_3(ASIA)}\end{aligned}$$

Recall that a regression parameter measures the effect on the outcome variable of increasing the predictor variable by *one unit*. So the coefficient of –0.88 on CYLINDERS means that fuel efficiency decreases by 0.88 miles per gallon for every additional cylinder added to a vehicle. Likewise, the coefficient of –0.01 on WEIGHT means that fuel efficiency decreases by 0.01 miles per gallon for every increase of one pound of WEIGHT. To interpret these coefficients properly, it is important to keep in mind the units of the predictor variable. For example, we might be surprised at how small the effect of WEIGHT seems to be until we realize that 0.01 here means 0.01 miles *per pound*, so increasing the weight of a car by 100 pounds decreases fuel efficiency by 1 mile per gallon. Most of the regression coefficients are relatively small except for CYLINDERS and YEAR; increasing the model year by one year appears to increase fuel efficiency by 0.70 miles per gallon, or 7 miles per gallon each decade.

We expected the coefficients on the first five variables to be negative, but this is not the case for DISPLACEMENT and ACCELERATION. It is possible that our intuition is wrong about the effects of these factors, but we should also seriously question keeping predictor variables in the model when the regression coefficients have the opposite sign from what we expected. In this context it is useful to examine the *p*-values in the fourth column of Figure 6.37. Recall that *p*-values roughly measure the probability that the true regression coefficient is zero. The *p*-values in this regression are quite low (below 5 or 10 percent) except for ACCELERATION, whose *p*-value is 0.84. This is another indication, in addition to the unexpected sign of the coefficient, that we might want to consider eliminating this variable from the model.

The regression algorithm provides an estimate of how well the model fits the data in the form of the R^2 statistic. This measure reflects the percentage of the variability in the output variable that is accounted for by the regression equation (R^2 lies between 0 and 1). The higher the R^2 the better the model fits the data. The R^2 in this case is 0.83, indicating that the model accounts for 83 percent of the variability in MPG. There is no magic level for R^2 that indicates a good enough fit. After all, it is really the prediction accuracy on the validation data that measures how well the model performs. But R^2 can be used as a rough indication of which of two regression models is likely to perform better.

The predictive accuracy of the regression model is summarized in terms of the RMS error in the table shown in Figure 6.38 (for details on performance measures for prediction models, see Section 6.1.3). As we would expect, the error is somewhat higher on the validation data (3.50) than on the training data (3.17), but the difference is not dramatic. It is helpful to observe in this context that the average value of MPG in this database is 23.8, so this level of error is about 15 percent of the average value. For another perspective, we can use the information given in the Scoring worksheet, where the difference between the actual and predicted values is listed for every record, to calculate that the average

FIGURE 6.38 Regression Accuracy on Training and Validation Data

Training Data scoring - Summary Report

Total sum of squared errors	RMS Error	Average Error
2397.721649	3.167381909	-3.83544E-06

Validation Data scoring - Summary Report

Total sum of squared errors	RMS Error	Average Error
1946.325514	3.498719905	-0.597126923

absolute error is 2.40. The RMS error is higher than the average absolute error because it penalizes large errors relatively more than small ones.

The results of the best subset selection process are shown in Figure 6.39. Recall that we specified in setting up the algorithm that XLMiner should search for the best subsets of predictor variables, up to a limit of 8. The rows of this table report the results of those searches for 1 through 8 predictors (a constant is also included in each case). The ninth row of this table summarizes the results when the best 8 variables are included. When we compare this row to the one above, we see that the first variable to be excluded is ACCELERATION, which is not surprising given our previous conclusion that the sign of the coefficient for this variable is the opposite of what we would expect. The next variable to be eliminated as we scan the rows of this table is CYLINDERS, followed by DISPLACEMENT. At each stage we track the performance of the model by examining the R^2 value (in the fourth column of the table), which decreases as each variable is eliminated.

As we look at the R^2 values, we can see that the value associated with the best subset drops off very little from eight predictors ($R^2 = 0.83$) to two predictors ($R^2 = 0.81$), but drops off much more when only one predictor remains. Thus, the two-predictor subset, with YEAR and WEIGHT in the model, may capture most of the predictive capability we can provide based on this dataset.

6.6.3 Strengths and Weaknesses of Multiple Linear Regression

Multiple linear regression is a very well-known and accepted model for prediction. It is easy to implement and the resulting regression equation is easy to interpret. Under the assumptions of this approach, each of the regression coefficients measures the (constant) per unit impact of the corresponding predictor variable. Some analysts consider it a strength of the regression approach that inferential statistics (p-values and R^2) are available. However, within the context of a data mining approach, these measures provide only rough guidance. Certainly a regression model with a high R^2 but low predictive accuracy on validation data is not worthy of serious consideration.

A number of alternative approaches to prediction should be considered in addition to multiple linear regression. These include k-Nearest Neighbor, Prediction Trees, and

Best subset selection

	#Coefs	RSS	Cp	R-Squared	Adj-R Squared	Probability	Model(Constant present in all models) 1	2	3	4	5	6	7	8	9
Choose Subset	2						Constant								
Choose Subset	3						Constant								
Choose Subset	4						Constant								
Choose Subset	5						Constant								
Choose Subset	6						Constant								
Choose Subset	7						Constant								
Choose Subset	8						Constant								
Choose Subset	9						Constant								

FIGURE 6.39 Best Subset Selection for Multiple Linear Regression

Neural Networks. Many data mining analysts build competing models using two or more distinct approaches and implement the one that is most effective.

6.7 LOGISTIC REGRESSION

Logistic regression, which is a statistical approach to the classification of categorical outcome variables, is very widely used in data mining. As the name implies, it is similar in many ways to multiple linear regression: it assumes a certain functional form for the relationship between the outcome and the predictor variables, and it uses the data to estimate the parameters of this relationship. Usually the outcome variable is binary, but logistic regression can also be used when the outcome has more than two values.

When the outcome variable is binary, it takes on two values that we can denote as Success/Failure, Yes/No, or 1/0, depending on the context. For example, we might wish to classify flights into those that will be delayed and those that will not, or classify companies into those that will default on their bonds and those that will not, or classify employees into those who will be promoted and those who will not.

The logistic regression model uses the data to produce a *probability* that a given case falls into one of these two classes. If we denote this probability by q, then the formula estimates

$$q = \text{Prob}(Y = 1) \tag{6.7}$$

where Y is the outcome variable. We then classify records as 1s when $q \geq 0.5$, and 0s when $q < 0.5$.

Probabilities like q, of course, are required to be between 0 and 1. This is why multiple linear regression cannot be used for the classification task: it can produce values that fall outside the acceptable range for probabilities. Logistic regression solves this problem, while preserving many of the benefits of multiple linear regression, by transforming the probability q into a form that can be estimated using a linear model.

6.7.1 Overview of Logistic Regression

To explain how logistic regression works, we begin with an example using one predictor variable. Assume we are attempting to predict which customers will default on their credit cards based on the size of their balances. The outcome variable is Y, which takes on the values 1 for default and 0 for no default. The predictor variable is X, which measures the size of their balance.

Our approach will be to transform the predictor value X for a specific customer into a probability q that the customer will default. Then, given q, we will classify that customer as $Y = 1$ or $Y = 0$ by comparing q to a cut-off probability that we choose. (The cut-off probability is usually 0.5; we will explain later why we sometimes use a higher or lower value.) The classification rule then is

$$\text{If } q \geq 0.5, Y = 1;$$

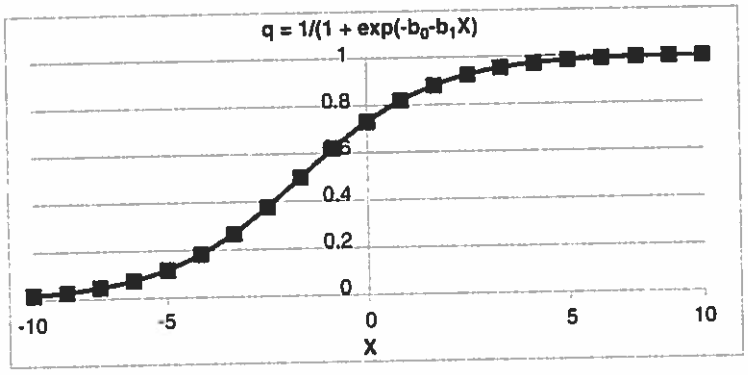
$$\text{if } q < 0.5, Y = 0.$$

As is usual in data mining, we fit our model to a training dataset and then test its classification accuracy on a validation dataset.

We pointed out above that we cannot use linear regression directly to calculate q, because there is no way to guarantee that a linear function of X such as $b_0 + b_1X$ will produce a result between 0 and 1. But we can ensure this condition if we relate X to q in a nonlinear fashion. Logistic regression gets its name from the logistic curve, which is an S-shaped curve often used to model the growth of populations or the growth of market share for a new product. The logistic curve takes the form

$$q = 1/[1 + e^{(-b_0 - b_1 X)}] \tag{6.8}$$

FIGURE 6.40 Logistic Function of One Variable



where the coefficients b_0 and b_1 are estimated from the data. Figure 6.40 shows a plot of this relationship for the parameters $b_0 = 1.0$ and $b_1 = 0.5$. Note that q falls between 0 and 1 regardless of the value of X.

Although the logistic function does accomplish our goal of generating an acceptable probability, the logistic regression algorithm actually uses an equivalent form based on *odds*, not probability. The odds of an event that occurs with probability q are simply $q/(1-q)$. (This quantity is called the *odds in favor*, as opposed to the *odds against*, which are often quoted in gambling.) So, for example, if $q = 0.5$, the corresponding odds are

$$q/(1-q) = 0.5/(1-0.5) = 0.5/0.5 = 1.0$$

Odds of 1.0 are usually stated as 1:1. Likewise, when $q = 0.1$, the odds of the event are $0.1/(1-0.1) = 0.1/0.9 = 1/9 = 1:9$. And when $q = 0.9$, the odds of the event are $0.9/(1-0.9) = 0.9/0.1 = 9/1 = 9:1$. Figure 6.41 shows how odds and probability are related.

An algebraically equivalent form of the logistic function is

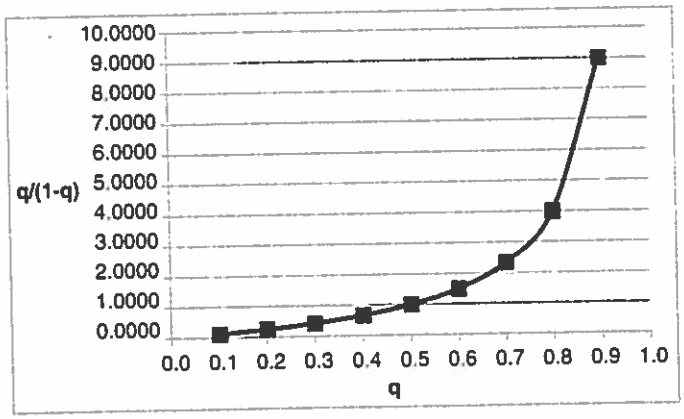
$$\ln[q/(1-q)] = -b_0 - b_1 X \tag{6.9}$$

In this form, the natural logarithm of the odds is a linear function of the predictor X. This form of the relationship is the one used in the logistic regression algorithm. When we have multiple predictors $X_1, X_2, \ldots X_q$, the model takes the form

$$\ln[q/(1-q)] = -b_0 - b_1 X_1 - b_2 X_2 - \ldots - b_q X_q \tag{6.10}$$

As in multiple linear regression, we estimate the values of the regression coefficients $b_1, b_2, \ldots, b_q$ using training data, and we assess the classification accuracy of the model on validation data.

FIGURE 6.41 Odds and Probability



6.7.2 An Application of Logistic Regression

We illustrate the application of the logistic regression approach using a database with 27 variables and 132 records on the financial condition of banks, some of which have declared bankruptcy (*Bankruptcy.xlsx*)*. The outcome variable is DECLARED, which takes on the value 1 when the bank has declared bankruptcy and 0 otherwise. The task is to use this database to create a method for predicting bankruptcy among banks. The variables in the database are described in the following table:

Variable	Description
NO	ID number
DECLARED	1: declared bankruptcy; 0 otherwise
YR	year of bankruptcy
R1	CASH/CURDEBT
R2	CASH/SALES
R3	CASH/ASSETS
R4	CASH/DEBTS
R5	CFFO/SALES
R6	CFFO/ASSETS
R7	CFFO/DEBTS
R8	COGS/INV
R9	CURASS/CURBEDT
R10	CURASS/SALES
R11	CURASS/ASSETS
R12	CURDEBT/DEBTS
R13	INC/SALES
R14	INC/ASSETS
R15	INC/DEBTS
R16	UBCDEP/SALES
R17	INCDEP/ASSETS
R18	INCDEP/DEBTS
R19	SALES/REC
R20	SALES/ASSETS
R21	ASSETS/DEBTS
R22	WCFO/SALES
R23	WCFO/ASSETS
R24	WCFO/DEBTS

The first step is to partition the database into 79 training records and 53 validation records (using the Standard Partition utility in XLMiner). We then select the logistic regression algorithm for classification: Data Mining▶Classify▶Logistic Regression. This brings up the window shown in Figure 6.42.

We identify the `Output variable` as Declared, and move all the other relevant variables into the `Input variables` category (excluding the irrelevant variables ID and YR). We also confirm that there are two classes for the output variable, "Success" is a 1 (representing good credit), and the probability cutoff for the classification rule is 0.5.

When we click on OK, a second window appears, shown in Figure 6.43.

We ignore the options in this window except for `Best subset`. We click on this button and another window opens, shown in Figure 6.44. We check `Perform best subset selection` and choose 10 for the `Maximum size of best subset`. Under this option the logistic regression algorithm runs on subsets of the predictor variables. This is a good procedure to use when we have a large number of predictor variables and we suspect that many of them will not be useful in the final model. Under `Selection procedure` XLMiner offers five different methods for choosing variables; we accept the default `Backward elimination`. Under this method, a logistic regression model will be created with a constant and the best 10 predictors. Then predictors are eliminated from

* Source: University of Virginia Darden School Foundation, 1988. Cited in Shmueli, G., N. R. Patel, and P. C. Bruce. 2010. *Data Mining for Business Intelligence*. New Jersey: John Wiley & Sons, page 390.

FIGURE 6.42 First Logistic Regression Window

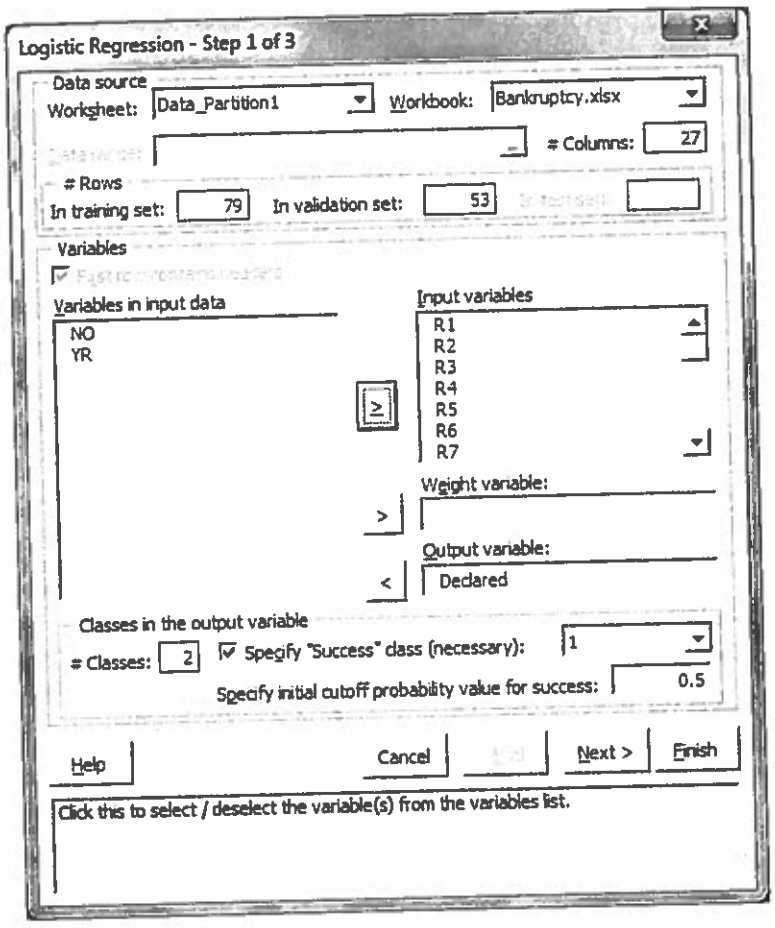


FIGURE 6.43 Second Logistic Regression Window

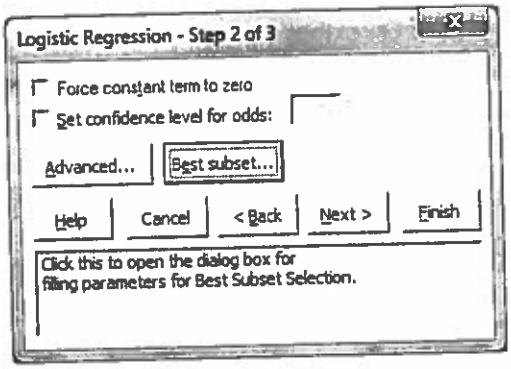


FIGURE 6.44 Best Subset Window for Logistic Regression

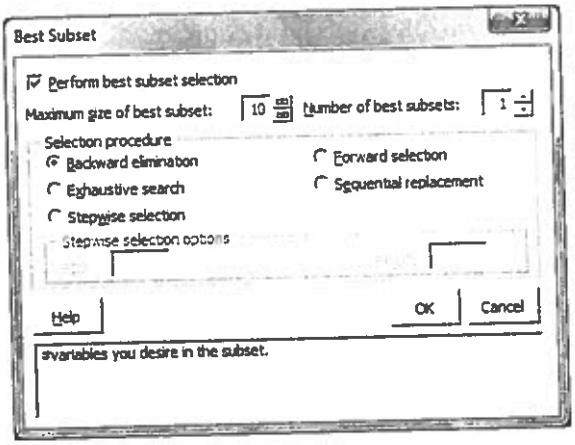
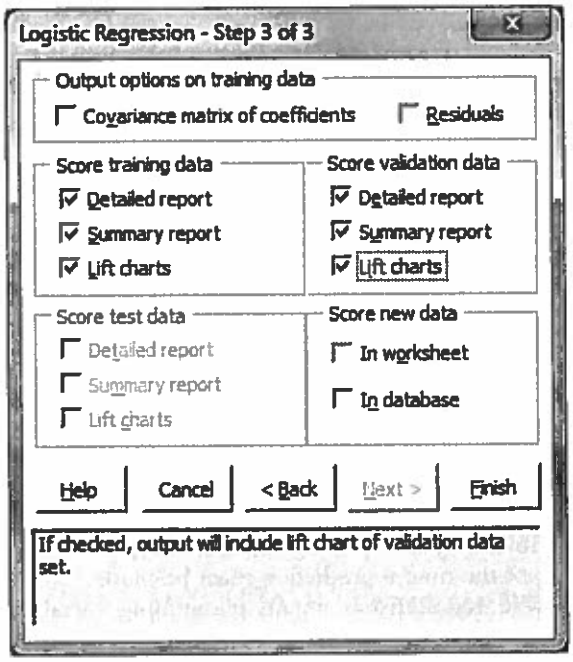


FIGURE 6.45 Third Logistic Regression Window



the model one by one. The eliminated predictors are those that contribute the least to the fit (R^2) of the model to the data.

We click OK and the final window opens, allowing us to select the level of detail in the output for both the training and validation partitions (Figure 6.45).

To assess the results of this procedure, we first examine the estimated regression coefficients b_i. These are displayed in a table along with their standard errors, p-values, and odds (Figure 6.46).

We begin by evaluating the sign and magnitude of each of these coefficients and testing them against our intuition and knowledge of the application area. For *continuous* predictor variables, a positive coefficient means that a higher value for the variable

FIGURE 6.46 Logistic Regression Output

Input variables	Coefficient	Std. Error	p-value	Odds
Constant term	0.94022244	11.45125103	0.93456203	*
R1	-72.5627213	43.50226974	0.09531145	0
R2	239.3921661	147.3618164	0.10426495	*
R3	-248.003448	140.2663422	0.07704595	0
R4	152.2921753	84.22820282	0.07059248	*
R5	62.8237648	38.32469559	0.10116117	1.92316E+27
R6	-239.699112	132.5966187	0.07064826	0
R7	97.11029053	72.98288727	0.18332401	*
R8	0.20126833	0.11925387	0.0914631	1.22295284
R9	3.16049457	12.9766655	0.80757773	23.58225441
R10	-52.4358826	39.23118591	0.18135756	0
R11	31.32620811	38.54645538	0.41639748	4.02531E+13
R12	-42.6215744	35.76930618	0.2334305	0
R13	184.1763458	194.9337006	0.34475318	*
R14	-1117.86804	617.1432495	0.07008529	0
R15	596.4025269	339.3278198	0.07881561	*
R16	-219.07251	158.6295929	0.1672692	0
R17	60.28596497	204.8086243	0.76848841	1.52007E+26
R18	211.1898956	200.7417908	0.29277778	*
R19	-0.05616639	0.04893949	0.2511048	0.94538182
R20	0.23155816	2.3573463	0.92175102	1.26056266
R21	6.15925217	13.79048634	0.65514183	473.0740967
R22	88.58720398	113.0458527	0.43325165	*
R23	565.6481934	309.3605652	0.06748305	*
R24	-425.693054	287.3888245	0.13854149	0

indicates a higher probability that $Y = 1$, and vice versa; a negative coefficient means that a higher value for the variable indicates a lower probability that $Y = 1$, and vice versa. For *binary categorical* predictor variables, a positive coefficient means a higher probability when $X = 1$ versus $X = 0$, and vice versa. In this case, we might question the positive coefficient on R8: COGS/INV: why would a higher Cost of Good Sold have a *positive* impact on good credit?

As we explained in more detail in Section 6.5, there are three complementary approaches to selecting predictors for a regression model, whether multiple linear regression or logistic regression:

- pre-processing the data
- use of inferential statistics
- search to select the best subset of predictors

Pre-processing the data means considering which predictor variables to use in the model *before* running the regression algorithm and eliminating any that do not prove acceptable. Variables that do not have a plausible connection to the output or ones that are highly correlated with other predictors should certainly be excluded. We also consider eliminating variables that are measured inaccurately, contain many missing values, or will not be available at the time a prediction must be made.

Use of inferential statistics means eliminating variables whose p-values exceed a cutoff, usually 5 or 10 percent. The p-values are listed in the fourth column of Figure 6.46. Note, for example, that the estimated coefficient for R9: CURASS/CURDEBT is 3.16, with a p-value of 0.81. This value indicates that the probability in repeated sampling is 81 percent that the true coefficient for this variable is zero. At the very least we can conclude that the true coefficient could be either positive or negative, and the case for including it in a final model is weak.

The third approach to selecting predictors, searching for the best subset, takes us to another portion of the XLMiner output, shown in Figure 6.47.

When we set up the algorithm, we specified that XLMiner should search for the best subsets of predictor variables, up to a limit of 10. The nine rows of this table report the results of those searches for 1 through 10 predictors (a constant is included in each case). To interpret this table we can start in the last row: this row shows the 10 variables that together fit the data best. As we move up a row, we see which variable among these 10 was eliminated as contributing the least to a good fit (R14). Four variables consistently appear among these alternative models: R5, R6, R11, and R12. These four probably belong in any model with a small number of predictors.

In multiple linear regression, the regression coefficients b_i have a convenient interpretation as the (constant) incremental impact of the corresponding predictor variable on the outcome variable. The same is *not* true in logistic regression, because the model is not linear. In logistic regression, the *odds* of the outcome $Y = 1$ change by the constant *factor* e^{bi} when the predictor X_i changes by 1. (For categorical variables, when X_i changes from 0 to 1 the odds of $Y = 1$ go up by the factor e^{bi}.) This factor is reported in the last column of Figure 6.46 (under the confusing heading "Odds"). So, in our example, the value of e^{bi} for the R8 variable is 1.22, which indicates that the odds increase by 22 percent for every increase of one unit in the ratio COGS/INV. The value of e^{bi} for R19 is 0.95, indicating that the odds decrease by 5 percent as the ratio SALES/REC increases by one unit.

Best subset selection

	#Coeffs	RSS	Cp	Probability	Model (Constant present in all models)										
					1	2	3	4	5	6	7	8	9	10	11
Choose Subset	2	61.88358307	-11.948802	0.98305781	Constant	R2									
Choose Subset	3	61.65716171	-10.1794958	0.99092835	Constant	R2	R5								
Choose Subset	4	61.31307983	-8.53006935	0.98914178	Constant	R5	R6	R11							
Choose Subset	5	60.57032776	-7.28683567	0.99044251	Constant	R5	R6	R11	R12						
Choose Subset	6	60.46712112	-5.39196971	0.98611212	Constant	R5	R6	R11	R12	R23					
Choose Subset	7	60.0524025	-3.81453323	0.98415732	Constant	R5	R6	R11	R12	R16	R23				
Choose Subset	8	59.25681087	-2.62533975	0.98678395	Constant	R5	R8	R11	R12	R14	R16	R23			
Choose Subset	9	59.07365799	-0.61174457	0.98163474	Constant	R2	R5	R6	R11	R12	R14	R16	R23		
Choose Subset	10	58.80073547	0.91018343	0.97835508	Constant	R1	R2	R5	R6	R11	R12	R14	R16	R23	
Choose Subset	11	58.54005051	2.64457989	0.98920838	Constant	R1	R2	R5	R6	R8	R11	R12	R14	R16	R23

FIGURE 6.47 Best Subset Selection for Logistic Regression

FIGURE 6.48 Classification Matrices for Logistic Regression

Training Data scoring - Summary Report

Cut off Prob.Val. for Success (Updatable)	0.5

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	36	1
0	1	41

Error Report			
Class	# Cases	# Errors	% Error
1	37	1	2.70
0	42	1	2.38
Overall	79	2	2.53

Validation Data scoring - Summary Report

Cut off Prob.Val. for Success (Updatable)	0.5

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	20	9
0	9	15

Error Report			
Class	# Cases	# Errors	% Error
1	29	9	31.03
0	24	9	37.50
Overall	53	18	33.96

The classification matrices for the logistic regression model are shown in Figure 6.48 for both the training and validation partitions. Among the 79 records in the training data set this model makes 2 classification errors for an overall error rate of 2.5 percent. Among the 53 records in the validation dataset it misclassifies 18 records, for an error rate of 34.0 percent. As usual, the model performs less well on the validation data than on the training data, but in this case the difference is dramatic. This is a model that is extremely over-fit to the training data and performs relatively poorly on the validation data.

The lift charts for the validation partition are shown in Figure 6.49. The decile-wise lift chart shows that the logistic regression model improves predictive accuracy by a factor of about 1.7 over the average accuracy on the top 10 percent of records.

An additional sign of an over-fit model is the large number of coefficients with high p-values (none of the coefficients have p-values below the traditional cut-off of 0.05). We should not be surprised by this because many of the variables in the database involve identical accounting measures (such as Sales, Assets, and Debts). This suggests that some of the variables may be highly correlated; for example, the correlation between CURASS/DEBT and ASSETS/DEBTS is 76 percent. An alternative model with only nine predictor variables (chosen to be relatively uncorrelated with each other) has a much higher error rate on the training data (17.7 percent) but a much lower error rate on the validation data (17.0 percent). This is likely to be a much better model for making predictions.

6.7.3 Strengths and Weaknesses of Logistic Regression

Logistic regression is a well-known and widely used technique, especially in marketing. It is easy to implement and fairly straightforward to interpret, although a facility with the concept of odds is necessary. When the data include a large number of predictor variables,

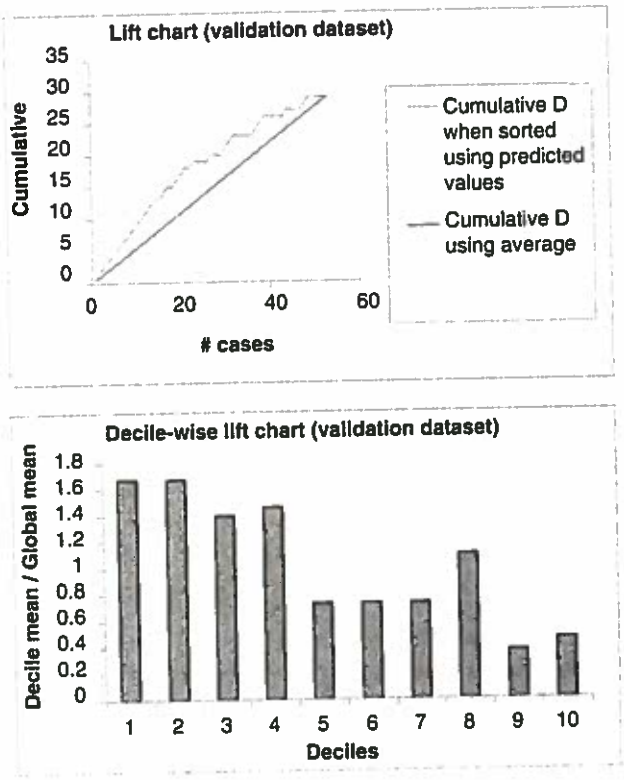


FIGURE 6.49 Lift Charts for Logistic Regression

it is necessary to reduce them to the most important ones using a combination of preprocessing, inferential statistics, and best subset selection.

Several approaches to classification should be considered in addition to logistic regression. These include *k*-Nearest Neighbor, Naïve Bayes, Classification Trees, and Neural Networks. Many data mining analysts build competing models using two or more distinct approaches and implement the most effective one.

6.8 NEURAL NETWORKS

Neural networks are an outgrowth of research within artificial intelligence into how the brain works. The brain is composed of complex networks of interconnected neurons. Each neuron is connected to a number of other neurons that send electrical impulses (or signals) into it, and it in turn sends electrical impulses out to a number of other neurons. Just how a neuron transforms the input signals into an output signal is not fully understood, but a type of learning called *reinforcement learning* seems to be involved. Reinforcement learning occurs over time when the learner takes in information and alters its internal state based on the difference between its actual output and its desired output. One can imagine learning to ride a bicycle as a process of reinforcement learning. After all, we don't learn to ride by listening to lectures or reading a book. We learn to ride by getting on a bicycle, straying too far to the left and trying various corrective actions until we regain the proper heading, then straying too far to the right and repeating the process. Over time, through this reinforcement process, our minds and bodies learn to ride. This is more or less how a neural network learns to perform its task.

Neural networks can be used for classification and for prediction. They accept as inputs both continuous and ordinal variables. Both multiple linear regression models and logistic regression models have the structure of simple neural networks, although the optimization algorithms used to estimate the parameters in those models are quite different from the learning algorithms used in neural networks.

Neural networks have been applied in an extremely wide variety of areas, ranging from financial applications to controlling robots. Within finance, they have been used to predict the bankruptcy of firms; to trade on currency, stock or bond markets; and to predict credit card fraud. Neural network models have a reputation for being complex and difficult to understand intuitively, but also for providing high predictive accuracy.

6.8.1 Overview of Neural Networks

A neural network has three components: a network structure, a set of input-output parameters, and a learning algorithm. The network structure is equivalent to the model in other data mining approaches. The network consists of **nodes**, which are analogous to neurons. At each node, a set of parameters governs how the node translates input data into an output to the next nodes in the network. Finally, a variety of learning algorithms can be used in neural networks. These algorithms use the data on the outcome variable and the predictor variables to determine the input-output parameters through a process similar to reinforcement learning. The only choice the modeler has to make is the network structure; once that is determined, the algorithm does the rest.

An example will make this clearer. Consider a college that is trying to determine which of its prospective incoming students will ultimately graduate. It has data on the high school GPA and Test Scores of a number of students, and on whether they graduated. Figure 6.50 shows a typical neural network structure for this situation. Nodes 1 and 2 are *input nodes*: they take in the data on the two predictor variables GPA and Test Scores for a particular student. Node 6 is the *output node*: it produces the output probability from which we will make the classification of the student as Graduating or Not Graduating. Nodes 3, 4, and 5 are *hidden layer nodes* that stand between the input and output nodes and transform the input values into the output value.

All neural networks have input, hidden, and output nodes. There must be one input node for each predictor variable and one output node for classification of a variable with two values. In between we can have any number of hidden layers, each of which can contain any number of nodes. However, simple networks with one hidden layer work quite well in many applications. (XLMiner includes an option to test models with varying numbers of layers and nodes.)

As we have said, each node in a neural network takes in input numbers from all upstream nodes that are connected to it, transforms that data into an output value, and sends the output to every downstream node it is connected to. For example, node 4 in Figure 6.50 takes in data from nodes 1 and 2 and sends its output to node 6. Each node has two sets of associated parameters: a *bias* (θ) *and weights* ($w_1, w_2, \ldots$). The bias is a single number, whereas there is one weight for each input node (see Figure 6.51). The output is computed using the logistic function

$$y = 1/[1 + e^{[-\theta - w_1X_1 - w_2X_2 - \ldots - w_qX_q]}] \tag{6.11}$$

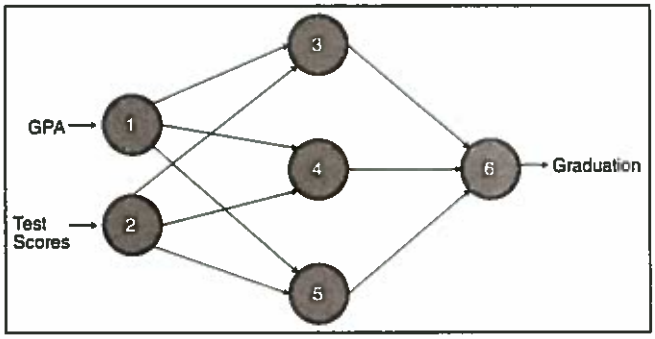


FIGURE 6.50 Simple Neural Network

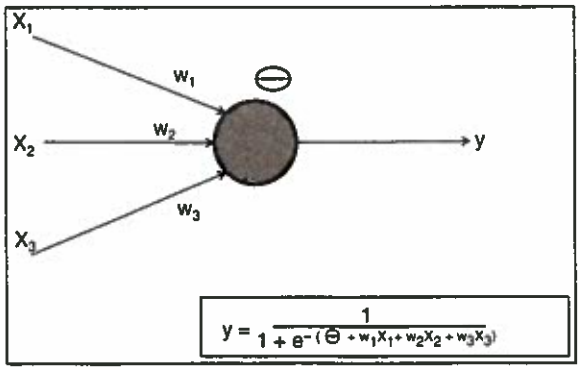


FIGURE 6.51 Computation of Output at a Node

In this formula, the X_i are the inputs from upstream nodes, the parameters w_i and θ are determined from the data, and the output y is the value sent to downstream nodes. In most neural networks, the raw predictor variables are normalized to the range from 0 to 1. The logistic function then ensures that the output y of each node in the network remains between 0 and 1.

Many different types of learning algorithms can be used in neural networks to determine the parameters θ and w_i at each node. We describe here the method of *back propagation* used in XLMiner. The first step in determining the parameters is to pick initial values. These are randomly selected between -0.5 and $+0.5$. Next, each record is run through the network to determine the output value for these initial parameter values. When the output is in error, that is, when the network predicts $Y = 1$ for a record for which $Y = 0$ (or vice versa), the values of the parameters are adjusted to better match the correct output. This process is repeated for all of the records in the training dataset, which is referred to as one *epoch*. The training process then continues over a number of epochs chosen by the user. Training stops when the weights are no longer changing significantly, or when the misclassification rate is acceptable, or when the maximum number of epochs has been reached. Once the training has been completed, the final network is used on the validation data to assess its classification accuracy.

6.8.2 An Application of Neural Networks

We illustrate the application of neural networks using a database with 16 variables and 4,985 records on the effectiveness of a direct mail campaign to sell wireless phone service (*EastWest.xlsx*)*. The outcome variable is PHONESALE, which takes on the value 1 when a customer agrees to buy a phone contract, and 0 otherwise. The task is to use this database to develop a method for classifying new customers by whether they will purchase or not. The variables in the database are described in the following table:

Variable	Description
ID#	unique identifier
Topflight	indicates whether flyer is Topflight (1) or not (0)
Balance	miles eligible for award travel
Qual_miles	miles qualifying for Topflight
cc1_miles	has member used frequent flyer card (1 = Yes, 0 = No)
cc2_miles	has member used Rewards card (1 = Yes, 0 = No)
cc3_miles	has member used Business card (1 = Yes, 0 = No)
Bonus_miles	miles earned from bonus
Bonus_trans	number of bonus transactions
Flight_miles_12mo	number of flight miles last 12 months
Flight_trans_12	number of flight transactions last 12 months
Online_12	online purchases last 12 months
Email	email address on file (1 = Yes, 0 = No)
Club_member	airline club member (1 = Yes, 0 = No)
Any_cc_miles_12mo	added miles on any credit card (1 = Yes, 0 = No)
PHONESALE	purchased in response to direct mail campaign

The first step is to partition the database into 60 percent (2,991) training records and 40 percent (1,994) validation records (using the Standard Partition utility in XLMiner). We then select the neural network algorithm for classification: Data Mining▶Classify▶Neural Network. This brings up a window similar to that shown in Figure 6.52.

In this window we have identified the `Output variable` as PHONE SALE, and moved all the other relevant variables into the `Input variables` category. We also confirm that there are two classes for the output variable, "Success" is a 1 (representing a phone sale), and the probability cutoff for the classification rule is 0.5.

When we click on OK a second window appears, shown in Figure 6.53.

In this window we check the box for `Normalize input data`, so all the predictor variables are normalized to fall between 0 and 1 (for more details see Chapter 5, Section A.5.5). We also select the `Manual` option under `Network architecture`, and choose a

* Source: Resampling Stats, Inc. 2005.

FIGURE 6.52 First Neural Network Window

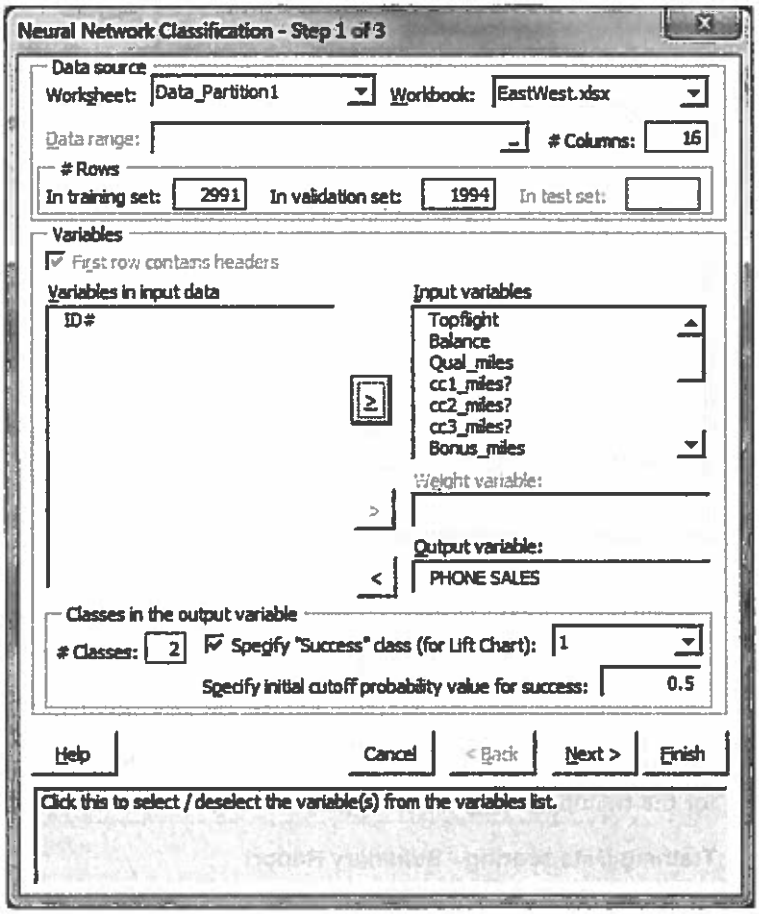


FIGURE 6.53 Second Neural Network Window

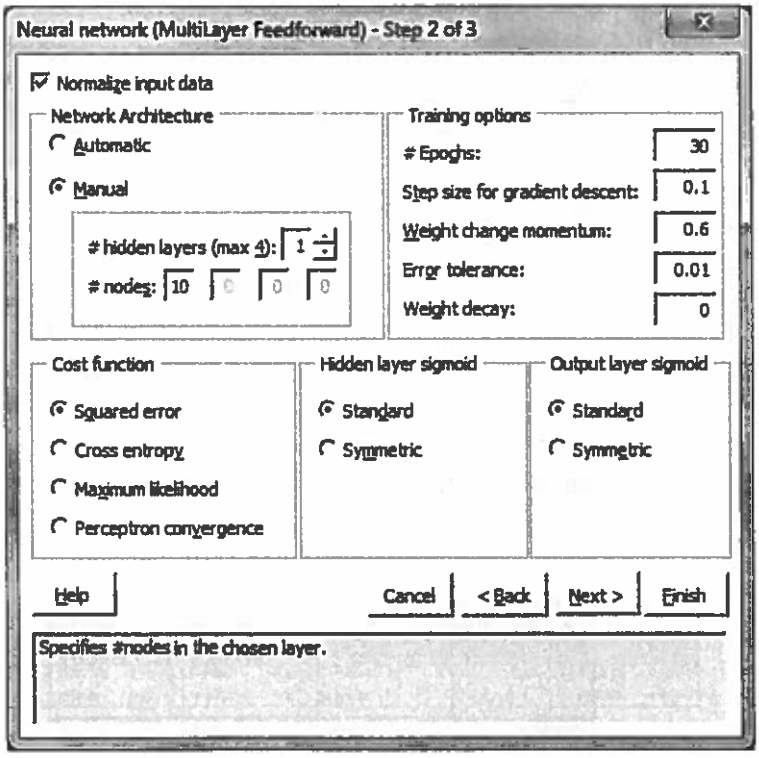
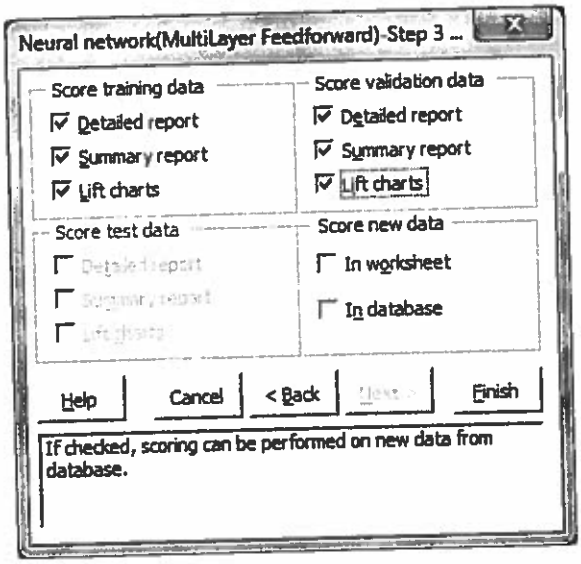


FIGURE 6.54 Third Neural Network Window



single hidden layer for the network with 10 nodes. Note that the default number of epochs is 30, so the input-output parameters will be updated by passing through the training data 30 times. (Under the `Automatic` option, XLMiner searches over the number of hidden layers and the number of nodes for models that perform well.)

The final input window (Figure 6.54) offers the usual options for displaying details for both training and validation data.

The most important outputs for a neural network are the classification matrices for the training and validation partitions. Figure 6.55 shows that the overall error rate is

FIGURE 6.55 Classification Matrices for Neural Network

Training Data scoring - Summary Report

Cut off Prob.Val. for Success (Updatable)	0.5

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	16	391
0	10	2574

Error Report			
Class	# Cases	# Errors	% Error
1	407	391	96.07
0	2584	10	0.39
Overall	2991	401	13.41

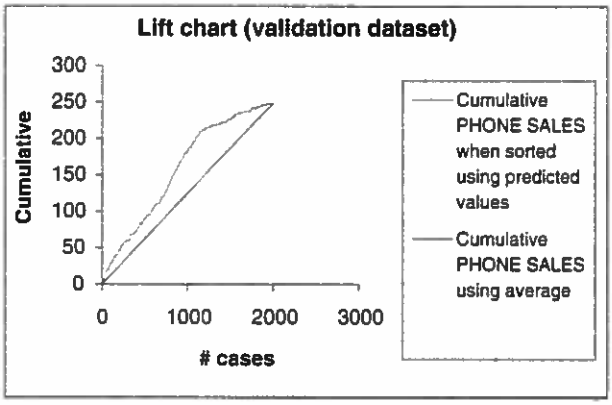
Validation Data scoring - Summary Report

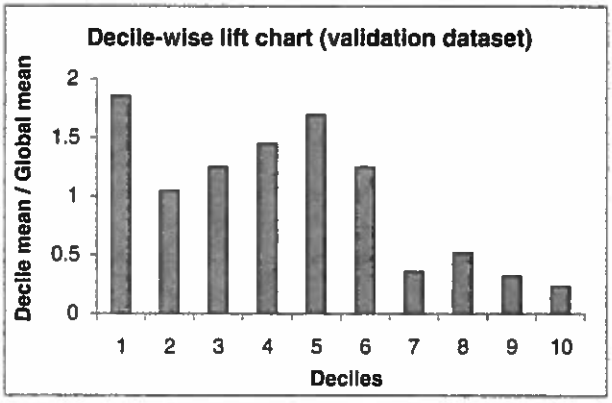
Cut off Prob.Val. for Success (Updatable)	0.5

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	5	243
0	11	1735

Error Report			
Class	# Cases	# Errors	% Error
1	248	243	97.98
0	1746	11	0.63
Overall	1994	254	12.74

FIGURE 6.56 Lift Charts for Neural Network





13.4 percent on the training data and 12.7 percent on the validation data. The great majority of errors in both cases are actual 1s classified as 0s. This may indicate that the neural network is setting too high a cutoff for classifying cases, and that lowering the cutoff might improve the performance of the model.

The lift charts for the validation dataset are shown in Figure 6.56. The decile-wise lift chart shows that the neural network performs about 70 percent better than the average classification on the top 10 percent of the cases.

We commented earlier that neural networks can appear to be complex and hard to understand intuitively. One contributing factor is that it is difficult to relate the network structure to any specific features of the problem itself, and the input-output parameters are not easy to interpret. These parameters are reported in the model output, as shown in Figure 6.57. Here the top table shows the weights at the hidden layer nodes. So the value of –1.27 in the first cell represents the weight on the first input variable (Topflight) at Node 1 (in the hidden layer). The values in the bottom table represent the weights on the output

Inter-layer connections weights

	Input Layer														
Hidden Layer # 1	Topflight	Balance	Qual_miles	cc1_miles?	cc2_miles?	cc3_miles?	Bonus_miles	Bonus_trans	Flight_miles_12mo	Flight_trans_12	Online_12	Email	Club_member	Any_cc_mile_12mo	Bias Node
Node # 1	-1.26888	-0.521952	-0.500775	0.0504865	-0.960643	0.19123	0.471247	0.230036	-0.605145	-0.00310082	-0.381642	0.898314	-1.02632	-0.557609	-0.71998
Node # 2	-0.415826	0.281331	-0.132404	0.224407	-0.274082	0.175251	0.257998	-0.245295	-0.207111	-0.0428865	-0.570688	0.115005	-0.798496	-0.182563	-0.599523
Node # 3	-0.606482	-0.346143	-0.428211	0.239136	-0.341649	0.0391274	0.433343	-0.248349	-0.0533675	-0.0131345	-0.0706507	-0.291621	-0.761374	-0.19301	-0.752636
Node # 4	2.83627	2.18589	0.386613	-0.57445	0.660958	-0.184517	-1.90795	-1.59632	-0.16358	-0.162792	-1.24851	0.048337	0.0220563	-1.01026	-0.373526
Node # 5	-0.216231	0.221634	-0.0598451	0.046953	-0.0505927	-0.216557	-0.0534339	-0.271672	-0.283615	0.130407	0.261381	-0.281946	-0.252978	-0.227955	-0.643668
Node # 6	-0.474873	0.322736	0.259249	0.38886	-0.00959488	0.14296	-0.807199	-0.0847936	0.35581	0.0452288	-0.206198	-0.673137	0.555938	-0.268357	-0.935843
Node # 7	-0.338404	0.597929	0.341455	0.139753	0.00729393	0.148224	-1.04514	-0.0212707	-0.13685	0.170048	-0.33268	-0.503064	0.796175	-0.22694	-0.965298
Node # 8	0.0229727	0.594877	0.066968	0.266679	0.0582912	-0.322123	-0.282686	-0.453696	0.0578073	-0.107503	0.293959	-0.23851	-0.207362	-0.00656528	-0.64518
Node # 9	-0.531628	-0.18146	-0.215265	0.222575	-0.223252	0.166299	0.235117	-0.0971159	-0.0282929	-0.105447	-0.380705	-0.0727286	-0.451626	-0.328063	-0.864976
Node # 10	-0.131846	0.215764	-0.0758814	0.115726	-0.000707	-0.0278689	-0.0853493	-0.165909	-0.174273	0.00847592	-0.0447191	-0.347188	-0.180851	-0.200236	-0.630423

	Hidden Layer # 1										
Output Layer	Node # 1	Node # 2	Node # 3	Node # 4	Node # 5	Node # 6	Node # 7	Node # 8	Node # 9	Node # 10	Bias Node
1	-0.911189	-0.117039	-0.280877	-1.54116	-0.168917	-0.321988	-0.442434	-0.145818	-0.124954	-0.0566348	0.345543
0	0.902308	0.136452	0.266758	1.54274	0.170604	0.325626	0.450494	0.162422	0.16118	0.0052299	-0.348467

FIGURE 6.57 Input-output Parameters for Neural Network

node for each hidden layer node. Thus the first entry of –0.91 represents the weight of Node 1 in the output.

Because neural networks are difficult to interpret, their value lies entirely in their predictive power.

6.8.3 Strengths and Weaknesses of Neural Networks

Neural networks have been highly successful in a number of applications and have been unsuccessful in many more. Their advantage is that they are very flexible because the fundamental structure of the model (the number of hidden layers and nodes) is chosen by the user. Thus they can capture complex relationships between the inputs and the outputs (much more so than linear regression, for example). However, this approach has several disadvantages. The models are difficult to interpret, and therefore hard to justify to skeptics. They also provide limited insight into the underlying relationships. Further, there is no mechanism within this algorithm for choosing the best predictor variables, so the modeler must carefully pre-process predictor variables and in many cases experiment with different sets of predictors to see which offers the most effective model.

For the classification task, alternative algorithms to neural networks include *k*-Nearest Neighbor, Naïve Bayes, Classification Trees, and Logistic Regression. For prediction, we can consider Prediction Trees or Multiple Linear Regression.

6.9 SUMMARY

In this chapter we presented a variety of methods for building models from large databases for classifying cases or making numerical predictions. Classification methods apply when the task is to predict which class an individual record will occupy, for example, *whether* a customer will buy a certain product. Prediction methods apply when the task is to predict a numerical outcome for an individual record, for example, *how much* a customer will buy.

We presented the following six methods in this chapter:

- *k*-Nearest Neighbor
- Naïve Bayes
- Classification and Prediction Trees
- Multiple Linear Regression
- Logistic Regression
- Neural Networks

For each of these methods we provided an intuitive overview of the approach, followed by an example of how the method is applied using XLMiner. Many of these methods can be applied to the same situation, and it is quite common for analysts to construct models using competing methods and then select the most effective one.

SUGGESTED READINGS

Many of the topics introduced in this chapter are covered in more detail and at a higher level of mathematical sophistication in the following book:

Shmueli, G., N. R. Patel, and P. C. Bruce. 2010. *Data Mining for Business Intelligence*. New Jersey: John Wiley & Sons.

The following book is one of the bibles in this area, with extensive coverage of a wide variety of methods and descriptions of numerous applications:

Nisbet, R., J. Elder, and G. Miner. 2009. *Handbook of Statistical Analysis and Data Mining Applications*. London: Elsevier.

WEKA is an open-source software platform on the Web for data mining. As such, it does not require that the analyst have access to any particular software product. The following book is a useful guide to data mining in general and WEKA in particular:

Witten, I. H., E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Massachusetts: Elsevier.

EXERCISES

In each of the following exercises, the goal is to develop a classification or prediction model for the given situation and data. Follow these steps:

1. Examine the data descriptions and explore the data.
2. Clean the data as needed.
3. Transform the data as needed (e.g., create dummy variables for categorical variables).
4. Partition the data.
5. Run the specified algorithm.
6. Interpret the results and choose the best model (e.g., choose the best *k* in the *k*-Nearest Neighbor method).

1. *K*-NEAREST NEIGHBORS—CLASSIFICATION

The database *Low Birth Weight.xlsx** consists of data on 189 women, 59 of whom had low-birth-weight babies. The data provided include basic demographic information (e.g., age of mother) as well as detailed information on the mother's behavior prior to the birth (e.g., smoking status). The goal is to create a *k*-Nearest Neighbor model to classify births as low risk.

*Source: University of Massachusetts Statistical Software Information (http://www.umass.edu/statdata/statdata/index.html).

2. *K*-NEAREST NEIGHBORS—PREDICTION

The database *Forest Fires.xlsx** contains information on the weather and ground conditions before the start of a number of forest fires. The goal is to create a *k*-Nearest Neighbor model to predict the size of a forest fire.

3. NAÏVE BAYES—CLASSIFICATION

The database *Bank Marketing.xlsx*** contains information on the results of a direct marketing campaign by a Portuguese bank. The purpose of the campaign was to enroll customers in a bank term deposit account. The data include demographic information (e.g., marital status) as well as financial information (e.g., size of balance). The goal is to create a Naïve Bayes model to determine which customers will enroll.

4. NAÏVE BAYES—CLASSIFICATION

The database *Spambase.xlsx*** contains detailed information on the words used in a large number of emails. The emails are classified as Spam or Not Spam. The goal is to create a Naïve Bayes model to determine whether individual emails are spam or not.

5. CLASSIFICATION AND PREDICTION TREES—CLASSIFICATION

The database *Boston Housing.xlsx*** contains information on 506 census tracts around the city of Boston. The data include housing-related information (e.g., average rooms per dwelling) as well as demographic information (e.g., per capita crime rate). The goal is to create a Classification Tree model to predict which census tracts have a median home price above $30,000.

6. CLASSIFICATION AND PREDICTION TREES—PREDICTION

The database *Automobile Losses.xlsx*** contains information on the average loss payment per insured year for a number of makes and models of cars. It also contains data on the physical characteristics of the vehicles (e.g., weight) and the price. The goal is to create a Prediction Tree model to predict the loss payment on a vehicle.

7. MULTIPLE LINEAR REGRESSION—PREDICTION

The database *Wine Quality.xlsx*** contains data on the chemical properties of nearly 5,000 wines. It also has a quality score for each wine that is based on the opinions of experts. The goal is to create a Multiple Linear Regression model which predicts the quality score.

8. MULTIPLE LINEAR REGRESSION—PREDICTION

The database *Airfares.xlsx**** contains information on the airfare on 638 routes in the United States. In addition, it contains data on the length of the route, the average income in the starting and ending cities, whether Southwest Airlines flies on the route, and other variables. The goal is to create a Multiple Linear Regression model which predicts the airfare along a given route.

9. LOGISTIC REGRESSION—CLASSIFICATION

The database *ICU Data.xlsx*† contains information on 200 subjects from a much larger study of the survival of patients admitted to an intensive care unit (ICU) of a hospital. The data provided include basic demographic information (e.g., age and sex) as well as detailed medical data (e.g., blood gas measurements). The goal is to create a Logistic Regression model to determine which patients admitted to the ICU will live.

10. LOGISTIC REGRESSION—CLASSIFICATION

The database *German Credit.xlsx*** contains information on the credit risk of 1,000 customers. The data include demographic information (e.g., gender) and financial information (e.g., savings account balance). The goal is to create a Logistic Regression model to classify customers by credit risk.

11. NEURAL NETWORKS—CLASSIFICATION

The database *Credit Approval.xlsx*** contains information on a large number of individuals who have applied for credit cards, including whether they were approved. The data include demographic data (e.g., age) and financial data (e.g., income). The goal is to create a Neural Network model to classify applicants by credit worthiness.

12. NEURAL NETWORKS—CLASSIFICATION

The database *Voting Records.xlsx*** contains data on the voting records for 435 members of the U.S. House of Representatives on 16 important pieces of legislation. The goal is to create a Neural Network model to classify members by party (Republican or Democrat).

*Source: UCI Machine Learning Repository (http://archive.ics.uci.edu/ml).
Citation: P. Cortez and A. Morais. "A Data Mining Approach to Predict Forest Fires using Meteorological Data." In J. Neves, M. F. Santos and J. Machado Eds., New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007—Portuguese Conference on Artificial Intelligence, December, Guimaraes, Portugal, pp. 512-523, 2007. APPIA, ISBN-13 978-989-95618-0-9. Available at: http://www.dsi.uminho.pt/~pcortez/fires.pdf.

**Source: UCI Machine Learning Repository (http://archive.ics.uci.edu/ml).

***Source: Shmueli, G., N. R. Patel, and P. C. Bruce. 2010. Data Mining for Business Intelligence. New Jersey: John Wiley & Sons, page 134.

†Source: University of Massachusetts Statistical Software Information (http://www.umass.edu/statdata/statdata/index.html).