

**Wiley Series on Methods and
Applications in Data Mining**

Daniel T. Larose, Series Editor

Second Edition

DISCOVERING KNOWLEDGE IN DATA

An Introduction to Data Mining

Daniel T. Larose • Chantal D. Larose

*DISCOVERING
KNOWLEDGE IN DATA*

WILEY SERIES ON METHODS AND APPLICATIONS IN DATA MINING

Series Editor: **Daniel T. Larose**

Discovering Knowledge in Data: An Introduction to Data Mining, Second Edition •
Daniel T. Larose and Chantal D. Larose

*Data Mining for Genomics and Proteomics: Analysis of Gene and Protein Expression
Data* • Darius M. Dziuda

Knowledge Discovery with Support Vector Machines • Lutz Hamel

Data-Mining on the Web: Uncovering Patterns in Web Content, Structure, and Usage •
Zdravko Markov and Daniel Larose

Data Mining Methods and Models • Daniel Larose

Practical Text Mining with Perl • Roger Bilisoly

SECOND EDITION

DISCOVERING KNOWLEDGE IN DATA

An Introduction to Data Mining

DANIEL T. LAROSE
CHANTAL D. LAROSE

IEEE
 computer
society

WILEY

Copyright © 2014 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permission>.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our website at www.wiley.com.

Library of Congress Cataloging-in-Publication Data:

Larose, Daniel T.

Discovering knowledge in data : an introduction to data mining / Daniel T. Larose and Chantal D. Larose. – Second edition.

pages cm

Includes index.

ISBN 978-0-470-90874-7 (hardback)

1. Data mining. I. Larose, Chantal D. II. Title.

QA76.9.D343L38 2014

006.3'12–dc23

2013046021

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

CONTENTS

PREFACE

xi

CHAPTER 1 AN INTRODUCTION TO DATA MINING

1

- 1.1 What is Data Mining? 1
- 1.2 Wanted: Data Miners 2
- 1.3 The Need for Human Direction of Data Mining 3
- 1.4 The Cross-Industry Standard Practice for Data Mining 4
 - 1.4.1 Crisp-DM: The Six Phases 5
- 1.5 Fallacies of Data Mining 6
- 1.6 What Tasks Can Data Mining Accomplish? 8
 - 1.6.1 Description 8
 - 1.6.2 Estimation 8
 - 1.6.3 Prediction 10
 - 1.6.4 Classification 10
 - 1.6.5 Clustering 12
 - 1.6.6 Association 14
- References 14
- Exercises 15

CHAPTER 2 DATA PREPROCESSING

16

- 2.1 Why do We Need to Preprocess the Data? 17
- 2.2 Data Cleaning 17
- 2.3 Handling Missing Data 19
- 2.4 Identifying Misclassifications 22
- 2.5 Graphical Methods for Identifying Outliers 22
- 2.6 Measures of Center and Spread 23
- 2.7 Data Transformation 26
- 2.8 Min-Max Normalization 26
- 2.9 Z-Score Standardization 27
- 2.10 Decimal Scaling 28
- 2.11 Transformations to Achieve Normality 28
- 2.12 Numerical Methods for Identifying Outliers 35
- 2.13 Flag Variables 36
- 2.14 Transforming Categorical Variables into Numerical Variables 37
- 2.15 Binning Numerical Variables 38
- 2.16 Reclassifying Categorical Variables 39
- 2.17 Adding an Index Field 39
- 2.18 Removing Variables that are Not Useful 39
- 2.19 Variables that Should Probably Not Be Removed 40
- 2.20 Removal of Duplicate Records 41

2.21	A Word About ID Fields	41
	The R Zone	42
	References	48
	Exercises	48
	Hands-On Analysis	50

CHAPTER 3 *EXPLORATORY DATA ANALYSIS*

51

3.1	Hypothesis Testing Versus Exploratory Data Analysis	51
3.2	Getting to Know the Data Set	52
3.3	Exploring Categorical Variables	55
3.4	Exploring Numeric Variables	62
3.5	Exploring Multivariate Relationships	69
3.6	Selecting Interesting Subsets of the Data for Further Investigation	71
3.7	Using EDA to Uncover Anomalous Fields	71
3.8	Binning Based on Predictive Value	72
3.9	Deriving New Variables: Flag Variables	74
3.10	Deriving New Variables: Numerical Variables	77
3.11	Using EDA to Investigate Correlated Predictor Variables	77
3.12	Summary	80
	The R Zone	82
	Reference	88
	Exercises	88
	Hands-On Analysis	89

CHAPTER 4 *UNIVARIATE STATISTICAL ANALYSIS*

91

4.1	Data Mining Tasks in <i>Discovering Knowledge in Data</i>	91
4.2	Statistical Approaches to Estimation and Prediction	92
4.3	Statistical Inference	93
4.4	How Confident are We in Our Estimates?	94
4.5	Confidence Interval Estimation of the Mean	95
4.6	How to Reduce the Margin of Error	97
4.7	Confidence Interval Estimation of the Proportion	98
4.8	Hypothesis Testing for the Mean	99
4.9	Assessing the Strength of Evidence Against the Null Hypothesis	101
4.10	Using Confidence Intervals to Perform Hypothesis Tests	102
4.11	Hypothesis Testing for the Proportion	104
	The R Zone	105
	Reference	106
	Exercises	106

CHAPTER 5 *MULTIVARIATE STATISTICS*

109

5.1	Two-Sample <i>t</i> -Test for Difference in Means	110
5.2	Two-Sample Z-Test for Difference in Proportions	111
5.3	Test for Homogeneity of Proportions	112
5.4	Chi-Square Test for Goodness of Fit of Multinomial Data	114
5.5	Analysis of Variance	115
5.6	Regression Analysis	118

5.7	Hypothesis Testing in Regression	122
5.8	Measuring the Quality of a Regression Model	123
5.9	Dangers of Extrapolation	123
5.10	Confidence Intervals for the Mean Value of y Given x	125
5.11	Prediction Intervals for a Randomly Chosen Value of y Given x	125
5.12	Multiple Regression	126
5.13	Verifying Model Assumptions	127
	The R Zone	131
	Reference	135
	Exercises	135
	Hands-On Analysis	136

CHAPTER 6 *PREPARING TO MODEL THE DATA*
138

6.1	Supervised Versus Unsupervised Methods	138
6.2	Statistical Methodology and Data Mining Methodology	139
6.3	Cross-Validation	139
6.4	Overfitting	141
6.5	BIAS–Variance Trade-Off	142
6.6	Balancing the Training Data Set	144
6.7	Establishing Baseline Performance	145
	The R Zone	146
	Reference	147
	Exercises	147

CHAPTER 7 *k-NEAREST NEIGHBOR ALGORITHM*
149

7.1	Classification Task	149
7.2	k -Nearest Neighbor Algorithm	150
7.3	Distance Function	153
7.4	Combination Function	156
	7.4.1 Simple Unweighted Voting	156
	7.4.2 Weighted Voting	156
7.5	Quantifying Attribute Relevance: Stretching the Axes	158
7.6	Database Considerations	158
7.7	k -Nearest Neighbor Algorithm for Estimation and Prediction	159
7.8	Choosing k	160
7.9	Application of k -Nearest Neighbor Algorithm Using IBM/SPSS Modeler	160
	The R Zone	162
	Exercises	163
	Hands-On Analysis	164

CHAPTER 8 *DECISION TREES*
165

8.1	What is a Decision Tree?	165
8.2	Requirements for Using Decision Trees	167
8.3	Classification and Regression Trees	168
8.4	C4.5 Algorithm	174
8.5	Decision Rules	179

8.6	Comparison of the C5.0 and Cart Algorithms Applied to Real Data	180
	The R Zone	183
	References	184
	Exercises	185
	Hands-On Analysis	185

CHAPTER 9 *NEURAL NETWORKS*

187

9.1	Input and Output Encoding	188
9.2	Neural Networks for Estimation and Prediction	190
9.3	Simple Example of a Neural Network	191
9.4	Sigmoid Activation Function	193
9.5	Back-Propagation	194
	9.5.1 Gradient Descent Method	194
	9.5.2 Back-Propagation Rules	195
	9.5.3 Example of Back-Propagation	196
9.6	Termination Criteria	198
9.7	Learning Rate	198
9.8	Momentum Term	199
9.9	Sensitivity Analysis	201
9.10	Application of Neural Network Modeling	202
	The R Zone	204
	References	207
	Exercises	207
	Hands-On Analysis	207

CHAPTER 10 *HIERARCHICAL AND k -MEANS CLUSTERING*

209

10.1	The Clustering Task	209
10.2	Hierarchical Clustering Methods	212
10.3	Single-Linkage Clustering	213
10.4	Complete-Linkage Clustering	214
10.5	k -Means Clustering	215
10.6	Example of k -Means Clustering at Work	216
10.7	Behavior of MSB, MSE, and PSEUDO- F as the k -Means Algorithm Proceeds	219
10.8	Application of k -Means Clustering Using SAS Enterprise Miner	220
10.9	Using Cluster Membership to Predict Churn	223
	The R Zone	224
	References	226
	Exercises	226
	Hands-On Analysis	226

CHAPTER 11 *KOHONEN NETWORKS*

228

11.1	Self-Organizing Maps	228
11.2	Kohonen Networks	230
	11.2.1 Kohonen Networks Algorithm	231
11.3	Example of a Kohonen Network Study	231
11.4	Cluster Validity	235
11.5	Application of Clustering Using Kohonen Networks	235

11.6	Interpreting the Clusters	237
11.6.1	Cluster Profiles	240
11.7	Using Cluster Membership as Input to Downstream Data Mining Models	242
	The R Zone	243
	References	245
	Exercises	245
	Hands-On Analysis	245

CHAPTER 12 ASSOCIATION RULES

247

12.1	Affinity Analysis and Market Basket Analysis	247
12.1.1	Data Representation for Market Basket Analysis	248
12.2	Support, Confidence, Frequent Itemsets, and the a Priori Property	249
12.3	How Does the a Priori Algorithm Work?	251
12.3.1	Generating Frequent Itemsets	251
12.3.2	Generating Association Rules	253
12.4	Extension from Flag Data to General Categorical Data	255
12.5	Information-Theoretic Approach: Generalized Rule Induction Method	256
12.5.1	J-Measure	257
12.6	Association Rules are Easy to do Badly	258
12.7	How Can We Measure the Usefulness of Association Rules?	259
12.8	Do Association Rules Represent Supervised or Unsupervised Learning?	260
12.9	Local Patterns Versus Global Models	261
	The R Zone	262
	References	263
	Exercises	263
	Hands-On Analysis	264

CHAPTER 13 IMPUTATION OF MISSING DATA

266

13.1	Need for Imputation of Missing Data	266
13.2	Imputation of Missing Data: Continuous Variables	267
13.3	Standard Error of the Imputation	270
13.4	Imputation of Missing Data: Categorical Variables	271
13.5	Handling Patterns in Missingness	272
	The R Zone	273
	Reference	276
	Exercises	276
	Hands-On Analysis	276

CHAPTER 14 MODEL EVALUATION TECHNIQUES

277

14.1	Model Evaluation Techniques for the Description Task	278
14.2	Model Evaluation Techniques for the Estimation and Prediction Tasks	278
14.3	Model Evaluation Techniques for the Classification Task	280
14.4	Error Rate, False Positives, and False Negatives	280
14.5	Sensitivity and Specificity	283
14.6	Misclassification Cost Adjustment to Reflect Real-World Concerns	284
14.7	Decision Cost/Benefit Analysis	285
14.8	Lift Charts and Gains Charts	286

X CONTENTS

14.9	Interweaving Model Evaluation with Model Building	289
14.10	Confluence of Results: Applying a Suite of Models	290
	The R Zone	291
	Reference	291
	Exercises	291
	Hands-On Analysis	291
 <i>APPENDIX: DATA SUMMARIZATION AND VISUALIZATION</i>		294
<i>INDEX</i>		309

PREFACE

WHAT IS DATA MINING?

According to the Gartner Group,

Data mining is the process of discovering meaningful new correlations, patterns and trends by sifting through large amounts of data stored in repositories, using pattern recognition technologies as well as statistical and mathematical techniques.

Today, there are a variety of terms used to describe this process, including *analytics*, *predictive analytics*, *big data*, *machine learning*, and *knowledge discovery in databases*. But these terms all share in common the objective of mining actionable nuggets of knowledge from large data sets. We shall therefore use the term *data mining* to represent this process throughout this text.

WHY IS THIS BOOK NEEDED?

Humans are inundated with data in most fields. Unfortunately, these valuable data, which cost firms millions to collect and collate, are languishing in warehouses and repositories. *The problem is that there are not enough trained human analysts available who are skilled at translating all of these data into knowledge*, and thence up the taxonomy tree into wisdom. This is why this book is needed.

The McKinsey Global Institute reports:¹

There will be a shortage of talent necessary for organizations to take advantage of big data. A significant constraint on realizing value from big data will be a shortage of talent, particularly of people with deep expertise in statistics and machine learning, and the managers and analysts who know how to operate companies by using insights from big data We project that demand for deep analytical positions in a big data world could exceed the supply being produced on current trends by 140,000 to 190,000 positions. . . . In addition, we project a need for 1.5 million additional managers and analysts in the United States who can ask the right questions and consume the results of the analysis of big data effectively.

This book is an attempt to help alleviate this critical shortage of data analysts. *Discovering Knowledge in Data: An Introduction to Data Mining* provides readers with:

- The models and techniques to uncover hidden nuggets of information,

¹*Big data: The next frontier for innovation, competition, and productivity*, by James Manyika et al., McKinsey Global Institute, www.mckinsey.com, May, 2011. Last accessed March 16, 2014.

- The insight into how the data mining algorithms really work, and
- The experience of actually performing data mining on large data sets.

Data mining is becoming more widespread everyday, because it empowers companies to uncover profitable patterns and trends from their existing databases. Companies and institutions have spent millions of dollars to collect megabytes and terabytes of data, but are not taking advantage of the valuable and actionable information hidden deep within their data repositories. However, as the practice of data mining becomes more widespread, companies which do not apply these techniques are in danger of falling behind, and losing market share, because their competitors are applying data mining, and thereby gaining the competitive edge.

In *Discovering Knowledge in Data*, the step-by-step, hands-on solutions of real-world business problems, using widely available data mining techniques applied to real-world data sets, will appeal to managers, CIOs, CEOs, CFOs, and others who need to keep abreast of the latest methods for enhancing return-on-investment.

WHAT'S NEW FOR THE SECOND EDITION?

The second edition of *Discovery Knowledge in Data* is enhanced with an abundance of new material and useful features, including:

- Nearly 100 pages of new material.
- Three new chapters:
 - Chapter 5: *Multivariate Statistical Analysis* covers the hypothesis tests used for verifying whether data partitions are valid, along with analysis of variance, multiple regression, and other topics.
 - Chapter 6: *Preparing to Model the Data* introduces a new formula for balancing the training data set, and examines the importance of establishing baseline performance, among other topics.
 - Chapter 13: *Imputation of Missing Data* addresses one of the most overlooked issues in data analysis, and shows how to impute missing values for continuous variables and for categorical variables, as well as how to handle patterns in missingness.
- *The R Zone*. In most chapters of this book, the reader will find *The R Zone*, which provides the actual R code needed to obtain the results shown in the chapter, along with screen shots of some of the output, using *R Studio*.
- A host of new topics not covered in the first edition. Here is a sample of these new topics, chapter by chapter:
 - Chapter 2: *Data Preprocessing*. Decimal scaling; Transformations to achieve normality; Flag variables; Transforming categorical variables into numerical variables; Binning numerical variables; Reclassifying categorical variables; Adding an index field; Removal of duplicate records.

- Chapter 3: *Exploratory Data Analysis*. Binning based on predictive value; Deriving new variables: Flag variables; Deriving new variables: Numerical variables; Using EDA to investigate correlated predictor variables.
- Chapter 4: *Univariate Statistical Analysis*. How to reduce the margin of error; Confidence interval estimation of the proportion; Hypothesis testing for the mean; Assessing the strength of evidence against the null hypothesis; Using confidence intervals to perform hypothesis tests; Hypothesis testing for the proportion.
- Chapter 5: *Multivariate Statistics*. Two-sample test for difference in means; Two-sample test for difference in proportions; Test for homogeneity of proportions; Chi-square test for goodness of fit of multinomial data; Analysis of variance; Hypothesis testing in regression; Measuring the quality of a regression model.
- Chapter 6: *Preparing to Model the Data*. Balancing the training data set; Establishing baseline performance.
- Chapter 7: *k-Nearest Neighbor Algorithm*. Application of k -nearest neighbor algorithm using IBM/SPSS Modeler.
- Chapter 10: *Hierarchical and k-Means Clustering*. Behavior of MSB, MSE, and pseudo- F as the k -means algorithm proceeds.
- Chapter 12: *Association Rules*. How can we measure the usefulness of association rules?
- Chapter 13: *Imputation of Missing Data*. Need for imputation of missing data; Imputation of missing data for continuous variables; Imputation of missing data for categorical variables; Handling patterns in missingness.
- Chapter 14: *Model Evaluation Techniques*. Sensitivity and Specificity.
- An *Appendix on Data Summarization and Visualization*. Readers who may be a bit rusty on introductory statistics may find this new feature helpful. Definitions and illustrative examples of introductory statistical concepts are provided here, along with many graphs and tables, as follows:
 - Part 1: *Summarization 1: Building Blocks of Data Analysis*
 - Part 2: *Visualization: Graphs and Tables for Summarizing and Organizing Data*
 - Part 3: *Summarization 2: Measures of Center, Variability, and Position*
 - Part 4: *Summarization and Visualization of Bivariate Relationships*
- New Exercises. There are over 100 new chapter exercises in the second edition.

DANGER! DATA MINING IS EASY TO DO BADLY

The plethora of new off-the-shelf software platforms for performing data mining has kindled a new kind of danger. The ease with which these graphical user interface (GUI)-based applications can manipulate data, combined with the power of the

formidable data mining algorithms embedded in the black box software currently available, makes their misuse proportionally more hazardous.

Just as with any new information technology, *data mining is easy to do badly*. A little knowledge is especially dangerous when it comes to applying powerful models based on large data sets. For example, analyses carried out on unpreprocessed data can lead to erroneous conclusions, or inappropriate analysis may be applied to data sets that call for a completely different approach, or models may be derived that are built upon wholly specious assumptions. These errors in analysis can lead to very expensive failures, if deployed.

“WHITE BOX” APPROACH: UNDERSTANDING THE UNDERLYING ALGORITHMIC AND MODEL STRUCTURES

The best way to avoid these costly errors, which stem from a blind black-box approach to data mining, is to instead apply a “white-box” methodology, which emphasizes an understanding of the algorithmic and statistical model structures underlying the software.

***Discovering Knowledge in Data* applies this white-box approach by:**

- Walking the reader through the various algorithms;
- Providing examples of the operation of the algorithm on actual large data sets;
- Testing the reader’s level of understanding of the concepts and algorithms;
- Providing an opportunity for the reader to do some real data mining on large data sets; and
- Supplying the reader with the actual R code used to achieve these data mining results, in *The R Zone*.

Algorithm Walk-Throughs

Discovering Knowledge in Data walks the reader through the operations and nuances of the various algorithms, using small sample data sets, so that the reader gets a true appreciation of what is really going on inside the algorithm. For example, in Chapter 10, *Hierarchical and K-Means Clustering*, we see the updated cluster centers being updated, moving toward the center of their respective clusters. Also, in Chapter 11, *Kohonen Networks*, we see just which kind of network weights will result in a particular network node “winning” a particular record.

Applications of the Algorithms to Large Data Sets

Discovering Knowledge in Data provides examples of the application of the various algorithms on actual large data sets. For example, in Chapter 9, *Neural Networks*, a classification problem is attacked using a neural network model on a real-world data set. The resulting neural network topology is examined, along with the network connection weights, as reported by the software. These data sets are included on the

data disk, so that the reader may follow the analytical steps on their own, using data mining software of their choice.

Chapter Exercises: Check Your Understanding

Discovering Knowledge in Data includes over 260 chapter exercises, which allow readers to assess their depth of understanding of the material, as well as have a little fun playing with numbers and data. These include conceptual exercises, which help to clarify some of the more challenging concepts in data mining, and “Tiny data set” exercises, which challenge the reader to apply the particular data mining algorithm to a small data set, and, step-by-step, to arrive at a computationally sound solution. For example, in Chapter 8, *Decision Trees*, readers are provided with a small data set and asked to construct—by hand, using the methods shown in the chapter—a *C4.5* decision tree model, as well as a *classification and regression tree* model, and to compare the benefits and drawbacks of each.

Hands-On Analysis: Learn Data Mining by Doing Data Mining

Most chapters provide *hands-on analysis problems*, representing an opportunity for the reader to apply newly-acquired data mining expertise to solving real problems using large data sets. Many people learn by doing. This book provides a framework where the reader can learn data mining by doing data mining.

The intention is to mirror the real-world data mining scenario. In the real world, dirty data sets need to be cleaned; raw data needs to be normalized; outliers need to be checked. So it is with *Discovering Knowledge in Data*, where about 100 hands-on analysis problems are provided. The reader can “ramp up” quickly, and be “up and running” data mining analyses in a short time.

For example, in Chapter 12, *Association Rules*, readers are challenged to uncover high confidence, high support rules for predicting which customer will be leaving the company’s service. In Chapter 14, *Model Evaluation Techniques*, readers are asked to produce lift charts and gains charts for a set of classification models using a large data set, so that the best model may be identified.

The R Zone

R is a powerful, open-source language for exploring and analyzing data sets (www.r-project.org). Analysts using *R* can take advantage of many freely available packages, routines, and GUIs, to tackle most data analysis problems. In most chapters of this book, the reader will find *The R Zone*, which provides the actual *R* code needed to obtain the results shown in the chapter, along with screen shots of some of the output. *The R Zone* is written by Chantal D. Larose (Ph.D. candidate in Statistics, University of Connecticut, Storrs), daughter of the author, and *R* expert, who uses *R* extensively in her research, including research on multiple imputation of missing data, with her dissertation advisors, Dr. Dipak Dey and Dr. Ofer Harel.

DATA MINING AS A PROCESS

One of the fallacies associated with data mining implementations is that data mining somehow represents an isolated set of tools, to be applied by an aloof analysis department, and marginally related to the mainstream business or research endeavor. Organizations which attempt to implement data mining in this way will see their chances of success much reduced. Data mining should be viewed as a *process*.

Discovering Knowledge in Data presents data mining as a well-structured *standard process*, intimately connected with managers, decision makers, and those involved in deploying the results. Thus, this book is not only for analysts, but for managers as well, who need to communicate in the language of data mining.

The standard process used is the *CRISP-DM* framework: the *Cross-Industry Standard Process for Data Mining*. *CRISP-DM* demands that data mining be seen as an entire process, from communication of the business problem, through data collection and management, data preprocessing, model building, model evaluation, and, finally, model deployment. Therefore, this book is not only for analysts and managers, but also for data management professionals, database analysts, and decision makers.

GRAPHICAL APPROACH, EMPHASIZING EXPLORATORY DATA ANALYSIS

Discovering Knowledge in Data emphasizes a graphical approach to data analysis. There are more than 170 screen shots of computer output throughout the text, and 40 other figures. Exploratory data analysis (EDA) represents an interesting and fun way to “feel your way” through large data sets. Using graphical and numerical summaries, the analyst gradually sheds light on the complex relationships hidden within the data. *Discovering Knowledge in Data* emphasizes an EDA approach to data mining, which goes hand-in-hand with the overall graphical approach.

HOW THE BOOK IS STRUCTURED

Discovering Knowledge in Data: An Introduction to Data Mining provides a comprehensive introduction to the field. Common myths about data mining are debunked, and common pitfalls are flagged, so that new data miners do not have to learn these lessons themselves. The first three chapters introduce and follow the *CRISP-DM* standard process, especially the data preparation phase and data understanding phase. The next nine chapters represent the heart of the book, and are associated with the *CRISP-DM* modeling phase. Each chapter presents data mining methods and techniques for a specific data mining task.

- Chapters 4 and 5 examine univariate and multivariate statistical analyses, respectively, and exemplify the *estimation* and *prediction* tasks, for example, using multiple regression.

- Chapters 7–9 relate to the *classification* task, examining *k*-nearest neighbor (Chapter 7), decision trees (Chapter 8), and neural network (Chapter 9) algorithms.
- Chapters 10 and 11 investigate the *clustering* task, with hierarchical and *k*-means clustering (Chapter 10) and Kohonen networks (Chapter 11) algorithms.
- Chapter 12 handles the *association* task, examining association rules through the *a priori* and *GRI* algorithms.
- Finally, Chapter 14 considers model evaluation techniques, which belong to the *CRISP-DM* evaluation phase.

Discovering Knowledge in Data as a Textbook

Discovering Knowledge in Data: An Introduction to Data Mining naturally fits the role of textbook for an introductory course in data mining. Instructors may appreciate:

- The presentation of data mining as a *process*
- The “White-box” approach, emphasizing an understanding of the underlying algorithmic structures
 - Algorithm walk-throughs
 - Application of the algorithms to large data sets
 - Chapter exercises
 - Hands-on analysis, and
 - *The R Zone*
- The graphical approach, emphasizing exploratory data analysis, and
- The logical presentation, flowing naturally from the *CRISP-DM* standard process and the set of data mining tasks.

Discovering Knowledge in Data is appropriate for advanced undergraduate or graduate-level courses. Except for one section in the neural networks chapter, no calculus is required. An introductory statistics course would be nice, but is not required. No computer programming or database expertise is required.

ACKNOWLEDGMENTS

I first wish to thank my mentor Dr. Dipak K. Dey, Distinguished Professor of Statistics, and Associate Dean of the College of Liberal Arts and Sciences at the University of Connecticut, as well as Dr. John Judge, Professor of Statistics in the Department of Mathematics at Westfield State College. My debt to the two of you is boundless, and now extends beyond one lifetime. Also, I wish to thank my colleagues in the data mining programs at Central Connecticut State University: Dr. Chun Jin, Dr. Daniel S. Miller, Dr. Roger Bilisoly, Dr. Dariusz Dziuda, and Dr. Krishna Saha. Thanks to my daughter Chantal Danielle Larose, for her important contribution to this book, as well as for her cheerful affection and gentle insanity. Thanks to my twin children

Tristan Spring and Ravel Renaissance for providing perspective on what life is really about. Finally, I would like to thank my wonderful wife, Debra J. Larose, for our life together.

DANIEL T. LAROSE, Ph.D.

Professor of Statistics and Data Mining
Director, Data Mining@CCSU
www.math.ccsu.edu/larose

I would first like to thank my PhD advisors, Dr. Dipak Dey, Distinguished Professor and Associate Dean, and Dr. Ofer Harel, Associate Professor, both from the Department of Statistics at the University of Connecticut. Their insight and understanding have framed and sculpted our exciting research program, including my PhD dissertation, *Model-Based Clustering of Incomplete Data*. Thanks also to my father Daniel for kindling my enduring love of data analysis, and to my mother Debra for her care and patience through many statistics-filled conversations. Finally thanks to my siblings, Ravel and Tristan, for perspective, music, and friendship.

CHANTAL D. LAROSE, MS

Department of Statistics
University of Connecticut

Let us settle ourselves, and work, and wedge our feet downwards through the mud and slush of opinion and tradition and prejudice and appearance and delusion, . . . till we come to a hard bottom with rocks in place which we can call *reality* and say, “This is, and no mistake.”

HENRY DAVID THOREAU

AN INTRODUCTION TO DATA MINING

1.1	WHAT IS DATA MINING?	1
1.2	WANTED: DATA MINERS	2
1.3	THE NEED FOR HUMAN DIRECTION OF DATA MINING	3
1.4	THE CROSS-INDUSTRY STANDARD PRACTICE FOR DATA MINING	4
1.5	FALLACIES OF DATA MINING	6
1.6	WHAT TASKS CAN DATA MINING ACCOMPLISH?	8
	REFERENCES	14
	EXERCISES	15

1.1 WHAT IS DATA MINING?

The McKinsey Global Institute (MGI) reports [1] that most American companies with more than 1000 employees had an average of at least 200 terabytes of stored data. MGI projects that the amount of data generated worldwide will increase by 40% annually, creating profitable opportunities for companies to leverage their data to reduce costs and increase their bottom line. For example, retailers harnessing this “big data” to best advantage could expect to realize an increase in their operating margin of more than 60%, according to the MGI report. And healthcare providers and health maintenance organizations (HMOs) that properly leverage their data storehouses could achieve \$300 in cost savings annually, through improved efficiency and quality.

The MIT Technology Review reports [2] that it was the Obama campaign’s effective use of data mining that helped President Obama win the 2012 presidential election over Mitt Romney. They first identified likely Obama voters using a data mining model, and then made sure that these voters actually got to the polls. The campaign also used a separate data mining model to predict the polling outcomes

county-by-county. In the important swing county of Hamilton County, Ohio, the model predicted that Obama would receive 56.4% of the vote; the Obama share of the actual vote was 56.6%, so that the prediction was off by only 0.02%. Such precise predictive power allowed the campaign staff to allocate scarce resources more efficiently.

About 13 million customers per month contact the West Coast customer service call center of the Bank of America, as reported by *CIO Magazine* [3]. In the past, each caller would have listened to the same marketing advertisement, whether or not it was relevant to the caller's interests. However, "rather than pitch the product of the week, we want to be as relevant as possible to each customer," states Chris Kelly, vice president and director of database marketing at Bank of America in San Francisco. Thus Bank of America's customer service representatives have access to individual customer profiles, so that the customer can be informed of new products or services that may be of greatest interest to him or her. This is an example of mining customer data to help identify the type of marketing approach for a particular customer, based on customer's individual profile.

So, what is data mining?

Data mining is the process of discovering useful patterns and trends in large data sets.

While waiting in line at a large supermarket, have you ever just closed your eyes and listened? You might hear the beep, beep, beep, of the supermarket scanners, reading the bar codes on the grocery items, ringing up on the register, and storing the data on company servers. Each beep indicates a new row in the database, a new "observation" in the information being collected about the shopping habits of your family, and the other families who are checking out.

Clearly, a lot of data is being collected. However, what is being learned from all this data? What knowledge are we gaining from all this information? Probably not as much as you might think, because there is a serious shortage of skilled data analysts.

1.2 WANTED: DATA MINERS

As early as 1984, in his book *Megatrends* [4], John Naisbitt observed that "We are drowning in information but starved for knowledge." The problem today is not that there is not enough data and information streaming in. We are in fact inundated with data in most fields. Rather, the problem is that there are not enough trained *human* analysts available who are skilled at translating all of these data into knowledge, and thence up the taxonomy tree into wisdom.

The ongoing remarkable growth in the field of data mining and knowledge discovery has been fueled by a fortunate confluence of a variety of factors:

- The explosive growth in data collection, as exemplified by the supermarket scanners above,

- The storing of the data in data warehouses, so that the entire enterprise has access to a reliable, current database,
- The availability of increased access to data from web navigation and intranets,
- The competitive pressure to increase market share in a globalized economy,
- The development of “off-the-shelf” commercial data mining software suites,
- The tremendous growth in computing power and storage capacity.

Unfortunately, according to the McKinsey report [1],

There will be a shortage of talent necessary for organizations to take advantage of big data. A significant constraint on realizing value from big data will be a shortage of talent, particularly of people with deep expertise in statistics and machine learning, and the managers and analysts who know how to operate companies by using insights from big data We project that demand for deep analytical positions in a big data world could exceed the supply being produced on current trends by 140,000 to 190,000 positions. . . . In addition, we project a need for 1.5 million additional managers and analysts in the United States who can ask the right questions and consume the results of the analysis of big data effectively.

This book is an attempt to help alleviate this critical shortage of data analysts.

1.3 THE NEED FOR HUMAN DIRECTION OF DATA MINING

Many software vendors market their analytical software as being a plug-and-play, out-of-the-box application that will provide solutions to otherwise intractable problems, without the need for human supervision or interaction. Some early definitions of data mining followed this focus on automation. For example, Berry and Linoff, in their book *Data Mining Techniques for Marketing, Sales and Customer Support* [5] gave the following definition for data mining: “Data mining is the process of exploration and analysis, *by automatic or semi-automatic means*, of large quantities of data in order to discover meaningful patterns and rules” [emphasis added]. Three years later, in their sequel *Mastering Data Mining* [6], the authors revisit their definition of data mining, and mention that, “If there is anything we regret, it is the phrase ‘by automatic or semi-automatic means’ . . . because we feel there has come to be too much focus on the automatic techniques and not enough on the exploration and analysis. This has misled many people into believing that data mining is a product that can be bought rather than a discipline that must be mastered.”

Very well stated! Automation is no substitute for human input. Humans need to be actively involved at every phase of the data mining process. Rather than asking where humans fit into data mining, we should instead inquire about how we may design data mining into the very human process of problem solving.

Further, the very power of the formidable data mining algorithms embedded in the black box software currently available makes their misuse proportionally more dangerous. Just as with any new information technology, *data mining is easy to do badly*. Researchers may apply inappropriate analysis to data sets that call for a

completely different approach, for example, or models may be derived that are built upon wholly specious assumptions. Therefore, an understanding of the statistical and mathematical model structures underlying the software is required.

1.4 THE CROSS-INDUSTRY STANDARD PRACTICE FOR DATA MINING

There is a temptation in some companies, due to departmental inertia and compartmentalization, to approach data mining haphazardly, to re-invent the wheel and duplicate effort. A cross-industry standard was clearly required, that is industry-neutral, tool-neutral, and application-neutral. The Cross-Industry Standard Process for Data Mining (CRISP-DM) [7] was developed by analysts representing Daimler-Chrysler, SPSS, and NCR. CRISP provides a nonproprietary and freely available standard process for fitting data mining into the general problem solving strategy of a business or research unit.

According to CRISP-DM, a given data mining project has a life cycle consisting of six phases, as illustrated in Figure 1.1. Note that the phase-sequence is *adaptive*.

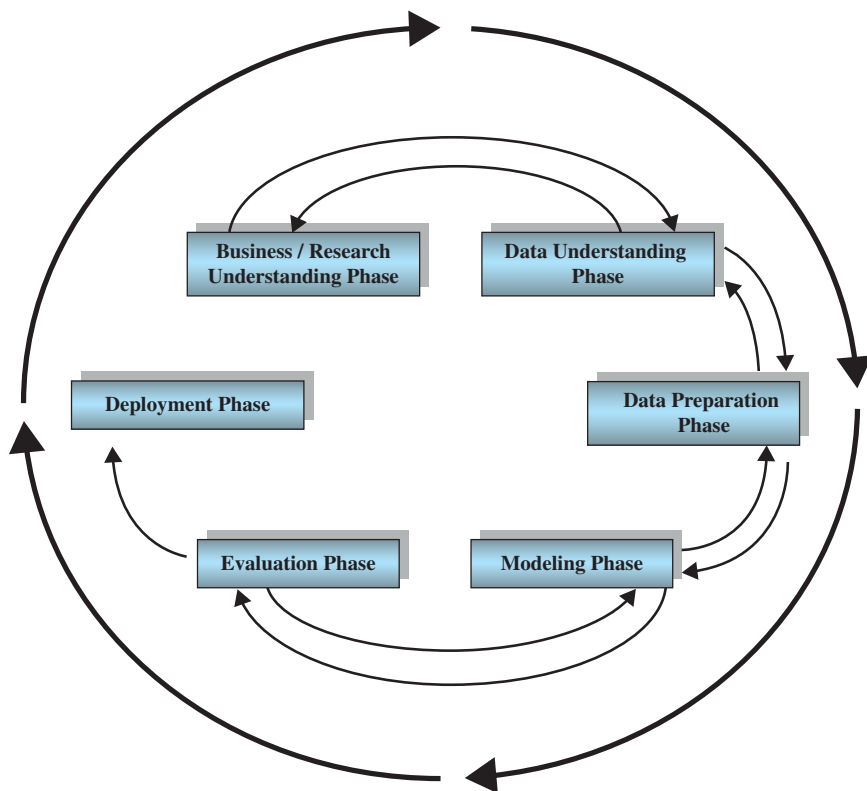


Figure 1.1 CRISP-DM is an iterative, adaptive process.

That is, the next phase in the sequence often depends on the outcomes associated with the previous phase. The most significant dependencies between phases are indicated by the arrows. For example, suppose we are in the modeling phase. Depending on the behavior and characteristics of the model, we may have to return to the data preparation phase for further refinement before moving forward to the model evaluation phase.

The iterative nature of CRISP is symbolized by the outer circle in Figure 1.1. Often, the solution to a particular business or research problem leads to further questions of interest, which may then be attacked using the same general process as before. Lessons learned from past projects should always be brought to bear as input into new projects. Here is an outline of each phase.

Issues encountered during the evaluation phase can conceivably send the analyst back to any of the previous phases for amelioration.

1.4.1 Crisp-DM: The Six Phases

1. Business/Research Understanding Phase

- a. First, clearly enunciate the project objectives and requirements in terms of the business or research unit as a whole.
- b. Then, translate these goals and restrictions into the formulation of a data mining problem definition.
- c. Finally, prepare a preliminary strategy for achieving these objectives.

2. Data Understanding Phase

- a. First, collect the data.
- b. Then, use exploratory data analysis to familiarize yourself with the data, and discover initial insights.
- c. Evaluate the quality of the data.
- d. Finally, if desired, select interesting subsets that may contain actionable patterns.

3. Data Preparation Phase

- a. This labor-intensive phase covers all aspects of preparing the final data set, which shall be used for subsequent phases, from the initial, raw, dirty data.
- b. Select the cases and variables you want to analyze, and that are appropriate for your analysis.
- c. Perform transformations on certain variables, if needed.
- d. Clean the raw data so that it is ready for the modeling tools.

4. Modeling Phase

- a. Select and apply appropriate modeling techniques.
- b. Calibrate model settings to optimize results.

- c. Often, several different techniques may be applied for the same data mining problem.
- d. May require looping back to data preparation phase, in order to bring the form of the data into line with the specific requirements of a particular data mining technique.

5. Evaluation Phase

- a. The modeling phase has delivered one or more models. These models must be evaluated for quality and effectiveness, before we deploy them for use in the field.
- b. Also, determine whether the model in fact achieves the objectives set for it in Phase 1.
- c. Establish whether some important facet of the business or research problem has not been sufficiently accounted for.
- d. Finally, come to a decision regarding the use of the data mining results.

6. Deployment Phase

- a. Model creation does not signify the completion of the project. Need to make use of created models.
- b. Example of a simple deployment: Generate a report.
- c. Example of a more complex deployment: Implement a parallel data mining process in another department.
- d. For businesses, the customer often carries out the deployment based on your model.

This book broadly follows CRISP-DM, with some modifications. For example, we prefer to clean the data (Chapter 2) before performing exploratory data analysis (Chapter 3).

1.5 FALLACIES OF DATA MINING

Speaking before the US House of Representatives SubCommittee on Technology, Information Policy, Intergovernmental Relations, and Census, Jen Que Louie, President of Nautilus Systems, Inc. described four fallacies of data mining [8]. Two of these fallacies parallel the warnings we have described above.

- **Fallacy 1.** There are data mining tools that we can turn loose on our data repositories, and find answers to our problems.
 - *Reality.* There are no automatic data mining tools, which will mechanically solve your problems “while you wait.” Rather data mining is a process. CRISP-DM is one method for fitting the data mining process into the overall business or research plan of action.

- **Fallacy 2.** The data mining process is autonomous, requiring little or no human oversight.
 - *Reality.* Data mining is not magic. Without skilled human supervision, blind use of data mining software will only provide you with the wrong answer to the wrong question applied to the wrong type of data. Further, the wrong analysis is worse than no analysis, since it leads to policy recommendations that will probably turn out to be expensive failures. Even after the model is deployed, the introduction of new data often requires an updating of the model. Continuous quality monitoring and other evaluative measures must be assessed, by human analysts.
- **Fallacy 3.** Data mining pays for itself quite quickly.
 - *Reality.* The return rates vary, depending on the start-up costs, analysis personnel costs, data warehousing preparation costs, and so on.
- **Fallacy 4.** Data mining software packages are intuitive and easy to use.
 - *Reality.* Again, ease of use varies. However, regardless of what some software vendor advertisements may claim, you cannot just purchase some data mining software, install it, sit back, and watch it solve all your problems. For example, the algorithms require specific data formats, which may require substantial preprocessing. Data analysts must combine subject matter knowledge with an analytical mind, and a familiarity with the overall business or research model.

To the above list, we add three further common fallacies:

- **Fallacy 5.** Data mining will identify the causes of our business or research problems.
 - *Reality.* The knowledge discovery process will help you to uncover patterns of behavior. Again, it is up to the humans to identify the causes.
- **Fallacy 6.** Data mining will automatically clean up our messy database.
 - *Reality.* Well, not automatically. As a preliminary phase in the data mining process, data preparation often deals with data that has not been examined or used in years. Therefore, organizations beginning a new data mining operation will often be confronted with the problem of data that has been lying around for years, is stale, and needs considerable updating.
- **Fallacy 7.** Data mining always provides positive results.
 - *Reality.* There is no guarantee of positive results when mining data for actionable knowledge. Data mining is not a panacea for solving business problems. But, used properly, by people who understand the models involved, the data requirements, and the overall project objectives, data mining can indeed provide actionable and highly profitable results.

The above discussion may have been termed, *what data mining cannot or should not do*. Next we turn to a discussion of what data mining can do.

1.6 WHAT TASKS CAN DATA MINING ACCOMPLISH?

The following list shows the most common data mining tasks.

Data Mining Tasks

- Description
- Estimation
- Prediction
- Classification
- Clustering
- Association

1.6.1 Description

Sometimes researchers and analysts are simply trying to find ways to *describe* patterns and trends lying within the data. For example, a pollster may uncover evidence that those who have been laid off are less likely to support the present incumbent in the presidential election. Descriptions of patterns and trends often suggest possible explanations for such patterns and trends. For example, those who are laid off are now less well off financially than before the incumbent was elected, and so would tend to prefer an alternative.

Data mining models should be as *transparent* as possible. That is, the results of the data mining model should describe clear patterns that are amenable to intuitive interpretation and explanation. Some data mining methods are more suited to transparent interpretation than others. For example, decision trees provide an intuitive and human-friendly explanation of their results. On the other hand, neural networks are comparatively opaque to nonspecialists, due to the nonlinearity and complexity of the model.

High quality description can often be accomplished with *exploratory data analysis*, a graphical method of exploring the data in search of patterns and trends. We look at exploratory data analysis in Chapter 3.

1.6.2 Estimation

In estimation, we approximate the value of a numeric target variable using a set of numeric and/or categorical predictor variables. Models are built using “complete” records, which provide the value of the target variable, as well as the predictors. Then, for new observations, estimates of the value of the target variable are made, based on the values of the predictors.

For example, we might be interested in estimating the systolic blood pressure reading of a hospital patient, based on the patient’s age, gender, body-mass index, and blood sodium levels. The relationship between systolic blood pressure and the predictor variables in the training set would provide us with an estimation model. We can then apply that model to new cases.

Examples of estimation tasks in business and research include

- Estimating the amount of money a randomly chosen family of four will spend for back-to-school shopping this fall
- Estimating the percentage decrease in rotary movement sustained by a football running back with a knee injury
- Estimating the number of points per game, LeBron James will score when double-teamed in the playoffs
- Estimating the grade point average (GPA) of a graduate student, based on that student's undergraduate GPA.

Consider Figure 1.2, where we have a scatter plot of the graduate GPAs against the undergraduate GPAs for 1000 students. Simple linear regression allows us to find the line that best approximates the relationship between these two variables, according to the least squares criterion. The regression line, indicated as a straight line increasing from left to right in Figure 1.2 may then be used to estimate the graduate GPA of a student, given that student's undergraduate GPA.

Here, the equation of the regression line (as produced by the statistical package *Minitab*, which also produced the graph) is $\hat{y} = 1.24 + 0.67x$. This tells us that the estimated graduate GPA \hat{y} equals 1.24 plus 0.67 times the student's undergrad GPA. For example, if your undergrad GPA is 3.0, then your estimated graduate GPA is $\hat{y} = 1.24 + 0.67(3) = 3.25$. Note that this point ($x = 3.0$, $\hat{y} = 3.25$) lies precisely on the regression line, as do all of the linear regression predictions.

The field of statistical analysis supplies several venerable and widely used estimation methods. These include, point estimation and confidence interval estimations, simple linear regression and correlation, and multiple regression. We examine these

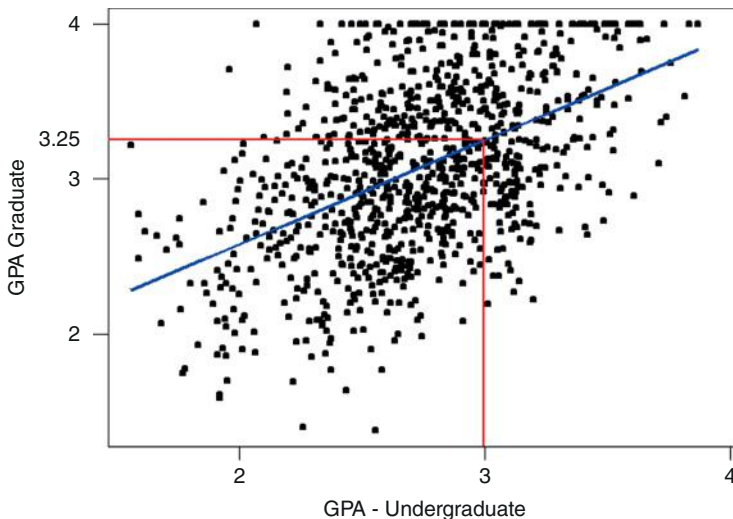


Figure 1.2 Regression estimates lie on the regression line.

methods and more in Chapters 4 and 5. Neural networks (Chapter 9) may also be used for estimation.

1.6.3 Prediction

Prediction is similar to classification and estimation, except that for prediction, the results lie in the future. Examples of prediction tasks in business and research include

- Predicting the price of a stock 3 months into the future.
- Predicting the percentage increase in traffic deaths next year if the speed limit is increased.
- Predicting the winner of this fall’s World Series, based on a comparison of the team statistics.
- Predicting whether a particular molecule in drug discovery will lead to a profitable new drug for a pharmaceutical company.

Any of the methods and techniques used for classification and estimation may also be used, under appropriate circumstances, for prediction. These include the traditional statistical methods of point estimation and confidence interval estimations, simple linear regression and correlation, and multiple regression, investigated in Chapters 4 and 5, as well as data mining and knowledge discovery methods like *k*-nearest neighbor methods (Chapter 7), decision trees (Chapter 8), and neural networks (Chapter 9).

1.6.4 Classification

Classification is similar to estimation, except that the target variable is categorical rather than numeric. In classification, there is a target categorical variable, such as *income bracket*, which, for example, could be partitioned into three classes or categories: high income, middle income, and low income. The data mining model examines a large set of records, each record containing information on the target variable as well as a set of input or predictor variables. For example, consider the excerpt from a data set shown in Table 1.1.

Suppose the researcher would like to be able to *classify* the income bracket of new individuals, not currently in the above database, based on the other characteristics associated with that individual, such as age, gender, and occupation. This task is a classification task, very nicely suited to data mining methods and techniques.

TABLE 1.1 Excerpt from data set for classifying income

Subject	Age	Gender	Occupation	Income Bracket
001	47	F	Software Engineer	High
002	28	M	Marketing Consultant	Middle
003	35	M	Unemployed	Low
...

The algorithm would proceed roughly as follows. First, examine the data set containing both the predictor variables and the (already classified) target variable, *income bracket*. In this way, the algorithm (software) “learns about” which combinations of variables are associated with which income brackets. For example, older females may be associated with the high income bracket. This data set is called the *training set*.

Then the algorithm would look at new records, for which no information about income bracket is available. Based on the classifications in the training set, the algorithm would assign classifications to the new records. For example, a 63-year-old female professor might be classified in the high income bracket.

Examples of classification tasks in business and research include

- Determining whether a particular credit card transaction is fraudulent;
- Placing a new student into a particular track with regard to special needs;
- Assessing whether a mortgage application is a good or bad credit risk;
- Diagnosing whether a particular disease is present;
- Determining whether a will was written by the actual deceased, or fraudulently by someone else;
- Identifying whether or not certain financial or personal behavior indicates possible criminal behavior.

For example, in the medical field, suppose we are interested in classifying the type of drug a patient should be prescribed, based on certain patient characteristics, such as the age of the patient and the patient’s sodium/potassium (Na/K) ratio. For a sample of 200 patients, Figure 1.3 presents a scatter plot of the patients’ Na/K ratio against the patients’ age. The particular drug prescribed is symbolized by the shade of the points. Light gray points indicate Drug Y; medium gray points indicate Drugs A or X; dark gray points indicate Drugs B or C. In this scatter plot, Na/K ratio is plotted on the Y (vertical) axis and age is plotted on the X (horizontal) axis.

Suppose that we will base our prescription recommendation based on this data set.

1. Which drug should be prescribed for a young patient with high Na/K ratio?
2. Young patients are on the left in the graph, and high Na/K ratios are in the upper half, which indicates that previous young patients with high Na/K ratios were prescribed Drug Y (light gray points). The recommended prediction classification for such patients is Drug Y.
3. Which drug should be prescribed for older patients with low Na/K ratios?
4. Patients in the lower right of the graph have been taking different prescriptions, indicated by either dark gray (Drugs B or C) or medium gray (Drugs A or X). Without more specific information, a definitive classification cannot be made here. For example, perhaps these drugs have varying interactions with beta-blockers, estrogens, or other medications, or are contraindicated for conditions such as asthma or heart disease.

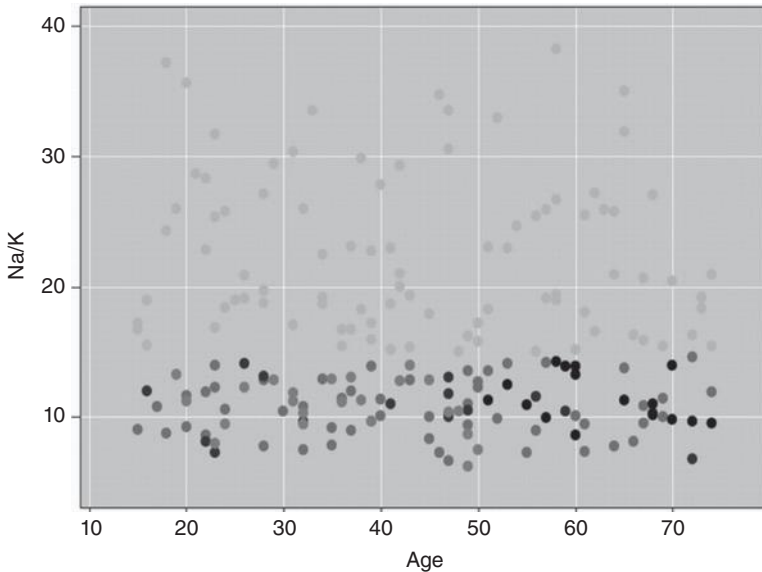


Figure 1.3 Which drug should be prescribed for which type of patient?

Graphs and plots are helpful for understanding two and three dimensional relationships in data. But sometimes classifications need to be based on many different predictors, requiring a many-dimensional plot. Therefore, we need to turn to more sophisticated models to perform our classification tasks. Common data mining methods used for classification are *k*-nearest neighbor (Chapter 7), decision trees (Chapter 8), and neural networks (Chapter 9).

1.6.5 Clustering

Clustering refers to the grouping of records, observations, or cases into classes of similar objects. A *cluster* is a collection of records that are similar to one another, and dissimilar to records in other clusters. Clustering differs from classification in that there is no target variable for clustering. The clustering task does not try to classify, estimate, or predict the value of a target variable. Instead, clustering algorithms seek to segment the whole data set into relatively homogeneous subgroups or clusters, where the similarity of the records within the cluster is maximized, and the similarity to records outside of this cluster is minimized.

Nielsen Claritas is in the clustering business. Among the services they provide is a demographic profile of each of the geographic areas in the country, as defined by zip code. One of the clustering mechanisms they use is the *PRIZM* segmentation system, which describes every American zip code area in terms of distinct lifestyle types. The 66 distinct clusters are shown in Table 1.2.

For illustration, the clusters for zip code 90210, Beverly Hills, California are

- Cluster # 01: Upper Crust Estates
- Cluster # 03: Movers and Shakers

TABLE 1.2 The 66 clusters used by the *PRIZM* segmentation system

01 Upper Crust	02 Blue Blood Estates	03 Movers and Shakers
04 Young Digerati	05 Country Squires	06 Winner's Circle
07 Money and Brains	08 Executive Suites	09 Big Fish, Small Pond
10 Second City Elite	11 God's Country	12 Brite Lites, Little City
13 Upward Bound	14 New Empty Nests	15 Pools and Patios
16 Bohemian Mix	17 Beltway Boomers	18 Kids and Cul-de-sacs
19 Home Sweet Home	20 Fast-Track Families	21 Gray Power
22 Young Influentials	23 Greenbelt Sports	24 Up-and-Comers
25 Country Casuals	26 The Cosmopolitans	27 Middleburg Managers
28 Traditional Times	29 American Dreams	30 Suburban Sprawl
31 Urban Achievers	32 New Homesteaders	33 Big Sky Families
34 White Picket Fences	35 Boomtown Singles	36 Blue-Chip Blues
37 Mayberry-ville	38 Simple Pleasures	39 Domestic Duos
40 Close-in Couples	41 Sunset City Blues	42 Red, White and Blues
43 Heartlanders	44 New Beginnings	45 Blue Highways
46 Old Glories	47 City Startups	48 Young and Rustic
49 American Classics	50 Kid Country, USA	51 Shotguns and Pickups
52 Suburban Pioneers	53 Mobility Blues	54 Multi-Culti Mosaic
55 Golden Ponds	56 Crossroads Villagers	57 Old Milltowns
58 Back Country Folks	59 Urban Elders	60 Park Bench Seniors
61 City Roots	62 Hometown Retired	63 Family Thrifts
64 Bedrock America	65 Big City Blues	66 Low-Rise Living

- Cluster # 04: Young Digerati
- Cluster # 07: Money and Brains
- Cluster # 16: Bohemian Mix

The description for Cluster # 01: Upper Crust is “The nation’s most exclusive address, Upper Crust is the wealthiest lifestyle in America, a Haven for empty-nesting couples between the ages of 45 and 64. No segment has a higher concentration of residents earning over \$100,000 a year and possessing a postgraduate degree. And none has a more opulent standard of living.”

Examples of clustering tasks in business and research include

- Target marketing of a niche product for a small-cap business which does not have a large marketing budget,
- For accounting auditing purposes, to segmentize financial behavior into benign and suspicious categories,
- As a dimension-reduction tool when the data set has hundreds of attributes,
- For gene expression clustering, where very large quantities of genes may exhibit similar behavior.

Clustering is often performed as a preliminary step in a data mining process, with the resulting clusters being used as further inputs into a different technique

downstream, such as neural networks.¹ We discuss hierarchical and k -means clustering in Chapter 10 and Kohonen networks in Chapter 11.

1.6.6 Association

The association task for data mining is the job of finding which attributes “go together.” Most prevalent in the business world, where it is known as affinity analysis or market basket analysis, the task of association seeks to uncover rules for quantifying the relationship between two or more attributes. Association rules are of the form “If *antecedent* then *consequent*,” together with a measure of the support and confidence associated with the rule. For example, a particular supermarket may find that, of the 1000 customers shopping on a Thursday night, 200 bought diapers, and of those 200 who bought diapers, 50 bought beer. Thus, the association rule would be “If buy diapers, then buy beer,” with a support of $200/1000 = 20\%$ and a confidence of $50/200 = 25\%$.

Examples of association tasks in business and research include

- Investigating the proportion of subscribers to your company’s cell phone plan that respond positively to an offer of a service upgrade,
- Examining the proportion of children whose parents read to them who are themselves good readers,
- Predicting degradation in telecommunications networks,
- Finding out which items in a supermarket are purchased together, and which items are never purchased together,
- Determining the proportion of cases in which a new drug will exhibit dangerous side effects.

We discuss two algorithms for generating association rules, the *a priori* algorithm, and the generalized rule induction (GRI) algorithm, in Chapter 12.

REFERENCES

1. James Manyika, by James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, Angela Hung Byers, *Big data: The next frontier for innovation, competition, and productivity*, McKinsey Global Institute, 2011, www.mckinsey.com, last accessed March 8, 2014.
2. Sasha Issenberg, *How President Obama’s campaign used big data to rally individual voters*, MIT Technology Review, December 19, 2012.
3. Peter Fabris, *Advanced Navigation*, in CIO Magazine, May 15, 1998 http://www.cio.com/archive/051598_mining.html.
4. John Naisbitt, *Megatrends*, 6th edn, Warner Books, 1986.
5. Michael Berry and Gordon Linoff, *Data Mining Techniques for Marketing, Sales and Customer Support*, John Wiley and Sons, New York, 1997.
6. Michael Berry and Gordon Linoff, *Mastering Data Mining*, John Wiley and Sons, New York, 2000.

¹For the use of clustering in market segmentation, see the forthcoming *Data Mining and Predictive Analytics*, by Daniel Larose and Chantal Larose, John Wiley and Sons, Inc., 2015.

7. Peter Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinart, Colin Shearer, and Rudiger Wirth, *CRISP-DM Step-by-Step Data Mining Guide*, 2000, www.the-modeling-agency.com/crisp-dm.pdf, last accessed March 8, 2014.
8. Jen Que Louie, President of Nautilus Systems, Inc. (www.nautilus-systems.com), Testimony before the US House of Representatives SubCommittee on Technology, Information Policy, Intergovernmental Relations, and Census, Federal Document Clearing House, Congressional Testimony, March 25, 2003.

EXERCISES

1. Refer to the Bank of America example early in the chapter. Which data mining task or tasks are implied in identifying “the type of marketing approach for a particular customer, based on customer’s individual profile”? Which tasks are not explicitly relevant?
2. For each of the following, identify the relevant data mining task(s):
 - a. The Boston Celtics would like to approximate how many points their next opponent will score against them.
 - b. A military intelligence officer is interested in learning about the respective proportions of Sunnis and Shias in a particular strategic region.
 - c. A NORAD defense computer must decide immediately whether a blip on the radar is a flock of geese or an incoming nuclear missile.
 - d. A political strategist is seeking the best groups to canvass for donations in a particular county.
 - e. A Homeland Security official would like to determine whether a certain sequence of financial and residence moves implies a tendency to terrorist acts.
 - f. A Wall Street analyst has been asked to find out the expected change in stock price for a set of companies with similar price/earnings ratios.
3. For each of the following meetings, explain which phase in the CRISP-DM process is represented:
 - a. Managers want to know by next week whether deployment will take place. Therefore, analysts meet to discuss how useful and accurate their model is.
 - b. The data mining project manager meets with the data warehousing manager to discuss how the data will be collected.
 - c. The data mining consultant meets with the Vice President for Marketing, who says that he would like to move forward with customer relationship management.
 - d. The data mining project manager meets with the production line supervisor, to discuss implementation of changes and improvements.
 - e. The analysts meet to discuss whether the neural network or decision tree models should be applied.
4. Discuss the need for human direction of data mining. Describe the possible consequences of relying on completely automatic data analysis tools.
5. CRISP-DM is not the only standard process for data mining. Research an alternative methodology (Hint: SEMMA, from the SAS Institute). Discuss the similarities and differences with CRISP-DM. ■

DATA PREPROCESSING

- 2.1 WHY DO WE NEED TO PREPROCESS THE DATA? 17**
- 2.2 DATA CLEANING 17**
- 2.3 HANDLING MISSING DATA 19**
- 2.4 IDENTIFYING MISCLASSIFICATIONS 22**
- 2.5 GRAPHICAL METHODS FOR IDENTIFYING OUTLIERS 22**
- 2.6 MEASURES OF CENTER AND SPREAD 23**
- 2.7 DATA TRANSFORMATION 26**
- 2.8 MIN-MAX NORMALIZATION 26**
- 2.9 Z-SCORE STANDARDIZATION 27**
- 2.10 DECIMAL SCALING 28**
- 2.11 TRANSFORMATIONS TO ACHIEVE NORMALITY 28**
- 2.12 NUMERICAL METHODS FOR IDENTIFYING OUTLIERS 35**
- 2.13 FLAG VARIABLES 36**
- 2.14 TRANSFORMING CATEGORICAL VARIABLES INTO NUMERICAL VARIABLES 37**
- 2.15 BINNING NUMERICAL VARIABLES 38**
- 2.16 RECLASSIFYING CATEGORICAL VARIABLES 39**
- 2.17 ADDING AN INDEX FIELD 39**
- 2.18 REMOVING VARIABLES THAT ARE NOT USEFUL 39**
- 2.19 VARIABLES THAT SHOULD PROBABLY NOT BE REMOVED 40**
- 2.20 REMOVAL OF DUPLICATE RECORDS 41**
- 2.21 A WORD ABOUT ID FIELDS 41**
 - THE R ZONE 42**
 - REFERENCES 48**
 - EXERCISES 48**
 - HANDS-ON ANALYSIS 50**

Discovering Knowledge in Data: An Introduction to Data Mining, Second Edition.
By Daniel T. Larose and Chantal D. Larose.
© 2014 John Wiley & Sons, Inc. Published 2014 by John Wiley & Sons, Inc.

Chapter 1 introduced us to data mining, and the CRISP-DM standard process for data mining model development. In Phase 1 of the data mining process, **business understanding** or **research understanding**, businesses and researchers first enunciate project objectives, then translate these objectives into the formulation of a data mining problem definition, and finally prepare a preliminary strategy for achieving these objectives.

Here in this chapter, we examine the next two phases of the CRISP-DM standard process, **data understanding** and **data preparation**. We will show how to evaluate the quality of the data, clean the raw data, deal with missing data, and perform transformations on certain variables. All of Chapter 3, *Exploratory Data Analysis*, is devoted to this very important aspect of the **data understanding** phase. The heart of any data mining project is the **modeling** phase, which we begin examining in Chapter 4.

2.1 WHY DO WE NEED TO PREPROCESS THE DATA?

Much of the raw data contained in databases is unpreprocessed, incomplete, and noisy. For example, the databases may contain

- Fields that are obsolete or redundant,
- Missing values,
- Outliers,
- Data in a form not suitable for the data mining models,
- Values not consistent with policy or common sense.

In order to be useful for data mining purposes, the databases need to undergo preprocessing, in the form of **data cleaning** and **data transformation**. Data mining often deals with data that have not been looked at for years, so that much of the data contain field values that have expired, are no longer relevant, or are simply missing. The overriding objective is to *minimize GIGO*, to minimize the Garbage that gets Into our model, so that we can minimize the amount of Garbage that our models give Ot.

Depending on the data set, data preprocessing alone can account for 10–60% of all the time and effort for the entire data mining process. In this chapter, we shall examine several ways to preprocess the data for further analysis downstream.

2.2 DATA CLEANING

To illustrate the need for cleaning up the data, let us take a look at some of the kinds of errors that could creep into even a tiny data set, such as that in Table 2.1.

Let us discuss, attribute by attribute, some of the problems that have found their way into the data set in Table 2.1. The *customer ID* variable seems to be fine. What about *zip*?

Let us assume that we are expecting all of the customers in the database to have the usual five-numeral American zip code. Now, Customer 1002 has this unusual (to American eyes) zip code of J2S7K7. If we were not careful, we might be tempted to

TABLE 2.1 Can you find any problems in this tiny data set?

Customer ID	Zip	Gender	Income	Age	Marital Status	Transaction Amount
1001	10048	M	78,000	C	M	5000
1002	J2S7K7	F	−40,000	40	W	4000
1003	90210		10,000,000	45	S	7000
1004	6269	M	50,000	0	S	1000
1005	55101	F	99,999	30	D	3000

classify this unusual value as an error, and toss it out, until we stop to think that not all countries use the same zip code format. Actually, this is the zip code of St. Hyacinthe, Quebec, Canada, and so probably represents real data from a real customer. What has evidently occurred is that a French-Canadian customer has made a purchase, and put their home zip code down in the required field. In the era of globalization, we must be ready to expect unusual values in fields such as zip codes, which vary from country to country.

What about the zip code for Customer 1004? We are unaware of any countries that have four digit zip codes, such as the 6269 indicated here, so this must be an error, right? Probably not. Zip codes for the New England states begin with the numeral 0. Unless the zip code field is defined to be *character* (text) and not *numeric*, the software will most likely chop off the leading zero, which is apparently what happened here. The zip code may well be 06269, which refers to Storrs, Connecticut, home of the University of Connecticut.

The next field, *gender*, contains a missing value for customer 1003. We shall detail methods for dealing with missing values later in this chapter.

The income field has three potentially anomalous values. First, Customer 1003 is shown as having an income of \$10,000,000 per year. While entirely possible, especially when considering the customer’s zip code (90210, Beverly Hills), this value of income is nevertheless an *outlier*, an extreme data value. Certain statistical and data mining modeling techniques do not function smoothly in the presence of outliers; therefore, we shall examine methods of handling outliers later in the chapter.

Poverty is one thing, but it is rare to find an income that is negative, as our poor Customer 1002 has. Unlike Customer 1003’s income, Customer 1002’s reported income of −\$40,000 lies beyond the field bounds for income, and therefore must be an error. It is unclear how this error crept in, with perhaps the most likely explanation being that the negative sign is a stray data entry error. However, we cannot be sure, and should approach this value cautiously, and attempt to communicate with the database manager most familiar with the database history.

So what is wrong with Customer 1005’s income of \$99,999? Perhaps nothing; it may in fact be valid. But, if all the other incomes are rounded to the nearest \$5000, why the precision with Customer 1005? Often, in legacy databases, certain specified values are meant to be codes for anomalous entries, such as missing values. Perhaps 99999 was coded in an old database to mean *missing*. Again, we cannot be sure and should again refer to the database administrator.

Finally, are we clear regarding which unit of measure the income variable is measured in? Databases often get merged, sometimes without bothering to check whether such merges are entirely appropriate for all fields. For example, it is quite possible that customer 1002, with the Canadian zip code, has an income measured in Canadian dollars, not U.S. dollars.

The *age* field has a couple of problems. Though all the other customers have numeric values for *age*, Customer 1001's "age" of *C* probably reflects an earlier categorization of this man's age into a bin labeled *C*. The data mining software will definitely not like this categorical value in an otherwise numeric field, and we will have to resolve this problem somehow. How about Customer 1004's age of 0? Perhaps there is a *newborn* male living in Storrs, Connecticut who has made a transaction of \$1000. More likely, the age of this person is probably missing and was coded as 0 to indicate this or some other anomalous condition (e.g., refused to provide the age information).

Of course, keeping an *age* field in a database is a minefield in itself, since the passage of time will quickly make the field values obsolete and misleading. It is better to keep *date*-type fields (such as birthdate) in a database, since these are constant, and may be transformed into ages when needed.

The *marital status* field seems fine, right? Maybe not. The problem lies in the meaning behind these symbols. We all think we know what these symbols mean, but are sometimes surprised. For example, if you are in search of cold water in a restroom in Montreal, and turn on the faucet marked *C*, you may be in for a surprise, since the *C* stands for *chaude*, which is French for *hot*. There is also the problem of ambiguity. In Table 2.1, for example, does the *S* for Customers 1003 and 1004 stand for *single* or *separated*?

The *transaction amount* field seems satisfactory, as long as we are confident that we know what unit of measure is being used, and that all records are transacted in this unit.

2.3 HANDLING MISSING DATA

Missing data are a problem that continues to plague data analysis methods. Even as our analysis methods gain sophistication, we nevertheless continue to encounter missing values in fields, especially in databases with a large number of fields. The absence of information is rarely beneficial. All things being equal, more information is almost always better. Therefore, we should think carefully about how we handle the thorny issue of missing data.

To help us tackle this problem, we will introduce ourselves to a new data set, the *cars* data set, originally compiled by Barry Becker and Ronny Kohavi of Silicon Graphics, and available for download at the book series website www.dataminingconsultant.com. The data set consists of information about 261 automobiles manufactured in the 1970s and 1980s, including gas mileage, number of cylinders, cubic inches, horsepower, and so on.

Suppose, however, that some of the field values were missing for certain records. Figure 2.1 provides a peek at the first 10 records in the data set, with two of the field values missing.

	mpg	cubicinches	hp	brand
1	14.000	350	165	US
2	31.900		71	Europe
3	17.000	302	140	US
4	15.000	400	150	
5	37.700	89	62	Japan

Figure 2.1 Some of our field values are missing.

A common method of “handling” missing values is simply to omit the records or fields with missing values from the analysis. However, this may be dangerous, since the pattern of missing values may in fact be systematic, and simply deleting the records with missing values would lead to a biased subset of the data. Further, it seems like a waste to omit the information in all the other fields, just because one field value is missing. In fact, Schmueli, *et al.* [1] state that if only 5% of data values are missing from a data set of 30 variables, and the missing values are spread evenly throughout the data, almost 80% of the records would have at least one missing value. Therefore, data analysts have turned to methods that would replace the missing value with a value substituted according to various criteria.

Some common criteria for choosing replacement values for missing data are as follows:

1. Replace the missing value with some constant, specified by the analyst.
2. Replace the missing value with the field mean¹ (for numeric variables) or the mode (for categorical variables).
3. Replace the missing values with a value generated at random from the observed distribution of the variable.
4. Replace the missing values with *imputed* values based on the other characteristics of the record.

Let us examine each of the first three methods, none of which is entirely satisfactory, as we shall see. Figure 2.2 shows the result of replacing the missing values with the constant 0 for the numerical variable *cubicinches* and the label *missing* for the categorical variable *brand*.

Figure 2.3 illustrates how the missing values may be replaced with the respective field means and modes.

	mpg	cubicinches	hp	brand
1	14.000	350	165	US
2	31.900	0	71	Europe
3	17.000	302	140	US
4	15.000	400	150	Missing
5	37.700	89	62	Japan

Figure 2.2 Replacing missing field values with user-defined constants.

¹ See the Appendix for the definition of *mean* and *mode*.

	mpg	cubicinches	hp	brand
1	14.000	350	165	US
2	31.900	200.65	71	Europe
3	17.000	302	140	US
4	15.000	400	150	US
5	37.700	89	62	Japan

Figure 2.3 Replacing missing field values with means or modes.

The variable *brand* is categorical, with mode *US*, so the software replaces the missing *brand* value with *brand* = *US*. *Cubicinches*, on the other hand, is continuous (numeric), so that the software replaces the missing *cubicinches* values with *cubicinches* = 200.65, which is the mean of all 258 non-missing values of that variable.

Isn't it nice to have the software take care of your missing data problems like this? In a way, certainly. However, do not lose sight of the fact that the software is creating information on the spot, actually fabricating data to fill in the holes in our data set. Choosing the field mean as a substitute for whatever value would have been there may sometimes work out well. However, the end-user needs to be informed that this process has taken place.

Further, the mean may not always be the best choice for what constitutes a "typical" value. For example, Larose [2] examines a data set where the mean is greater than the 81st percentile. Also, if many missing values are replaced with the mean, the resulting confidence levels for statistical inference will be overoptimistic, since measures of spread will be artificially reduced. It must be stressed that replacing missing values is a gamble, and the benefits must be weighed against the possible invalidity of the results.

Finally, Figure 2.4 demonstrates how missing values can be replaced with values generated at random from the observed distribution of the variable.

One benefit of this method is that the measures of center and spread should remain closer to the original, when compared to the mean replacement method. However, there is no guarantee that the resulting records would make sense. For example, the random values drawn in Figure 2.4 has led to at least one car that does not in fact exist! There is no Japanese-made car in the database which has an engine size of 400 cubic inches.

	mpg	cubicinches	hp	brand
1	14.000	350	165	US
2	31.900	450	71	Europe
3	17.000	302	140	US
4	15.000	400	150	Japan
5	37.700	89	62	Japan

Figure 2.4 Replacing missing field values with random draws from the distribution of the variable.

We therefore need *data imputation methods* that take advantage of the knowledge that the car is Japanese when calculating its missing cubic inches. In data imputation, we ask “What would be the most likely value for this missing value, given all the other attributes for a particular record?” For instance, an American car with 300 cubic inches and 150 horsepower would probably be expected to have more cylinders than a Japanese car with 100 cubic inches and 90 horsepower. This is called *imputation of missing data*. Before we can profitably discuss data imputation, however, we need to learn the tools needed to do so, such as multiple regression or classification and regression trees. Therefore, to learn about the imputation of missing data, please see Chapter 13.

2.4 IDENTIFYING MISCLASSIFICATIONS

Let us look at an example of checking the classification labels on the categorical variables, to make sure that they are all valid and consistent. Suppose that a frequency distribution of the variable *brand* was as shown in Table 2.2.

The frequency distribution shows five classes, USA, France, US, Europe, and Japan. However, two of the classes, USA and France, have a count of only one automobile each. What is clearly happening here is that two of the records have been inconsistently classified with respect to the origin of manufacture. To maintain consistency with the remainder of the data set, the record with origin *USA* should have been labeled *US*, and the record with origin *France* should have been labeled *Europe*.

2.5 GRAPHICAL METHODS FOR IDENTIFYING OUTLIERS

Outliers are extreme values that go against the trend of the remaining data. Identifying outliers is important because they may represent errors in data entry. Also, even if an outlier is a valid data point and not an error, certain statistical methods are sensitive to the presence of outliers and may deliver unreliable results.

One graphical method for identifying outliers for numeric variables is to examine a *histogram*² of the variable. Figure 2.5 shows a histogram of the vehicle weights from the (slightly amended) *cars* data set. (Note: This slightly amended data set is available as *cars2* from the series website.)

TABLE 2.2 Notice anything strange about this frequency distribution?

Brand	Frequency
USA	1
France	1
US	156
Europe	46
Japan	51

²See the Appendix for more on histograms, including a caution on their interpretation.

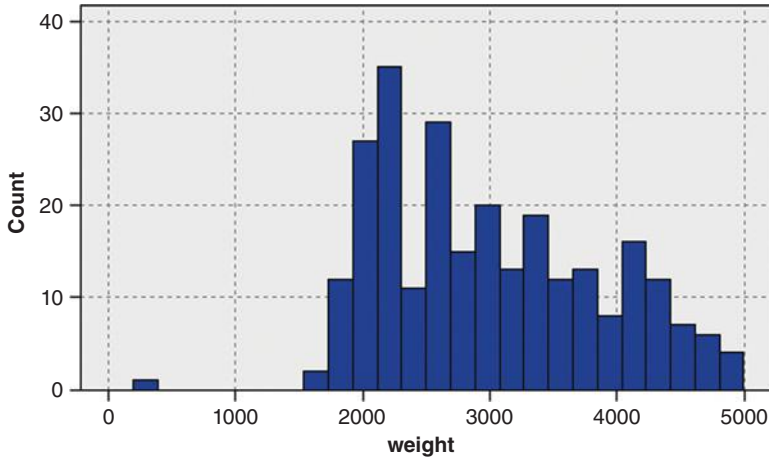


Figure 2.5 Histogram of vehicle weights: can you find the outlier?

There appears to be one lonely vehicle in the extreme left tail of the distribution, with a vehicle weight in the hundreds of pounds rather than in the thousands. Further investigation (not shown) tells us that the minimum weight is 192.5 pounds, which is undoubtedly our little outlier in the lower tail. As 192.5 pounds is rather light for an automobile, we would tend to doubt the validity of this information.

We can surmise that perhaps the weight was originally 1925 pounds, with the decimal inserted somewhere along the line. We cannot be certain, however, and further investigation into the data sources is called for.

Sometimes two-dimensional scatter plots³ can help to reveal outliers in more than one variable. Figure 2.6, a scatter plot of *mpg* against *weightlbs*, seems to have netted two outliers.

Most of the data points cluster together along the horizontal axis, except for two outliers. The one on the left is the same vehicle we identified in Figure 2.6, weighing only 192.5 pounds. The outlier near the top is something new: a car that gets over 500 miles per gallon! Clearly, unless this vehicle runs on dilithium crystals, we are looking at a data entry error.

Note that the 192.5 pound vehicle is an outlier with respect to weight but not with respect to mileage. Similarly, the 500-mpg car is an outlier with respect to mileage but not with respect to weight. Thus, a record may be an outlier in a particular dimension but not in another. We shall examine numeric methods for identifying outliers, but we need to pick up a few tools first.

2.6 MEASURES OF CENTER AND SPREAD

Suppose that we are interested in estimating where the center of a particular variable lies, as measured by one of the numerical *measures of center*, the most common

³See the Appendix for more on scatter plots.

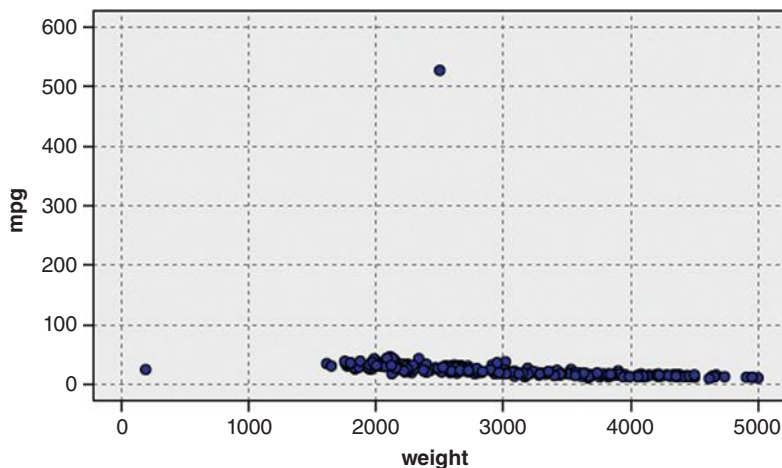


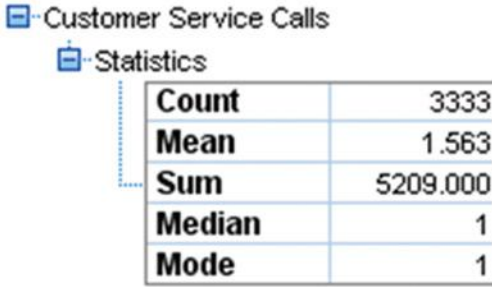
Figure 2.6 Scatter plot of *mpg* against *Weightlbs* shows two outliers.

of which are the mean, median, and mode. Measures of center are a special case of *measures of location*, numerical summaries that indicate *where* on a number line a certain characteristic of the variable lies. Examples of measures of location are percentiles and quantiles.

The *mean* of a variable is simply the average of the valid values taken by the variable. To find the mean, simply add up all the field values and divide by the sample size. Here we introduce a bit of notation. The sample mean is denoted as \bar{x} (“*x*-bar”) and is computed as $\bar{x} = \sum x/n$, where \sum (capital sigma, the Greek letter “S,” for “summation”) represents “sum all the values” and n represents the sample size. For example, suppose that we are interested in estimating where the center of the *customer service calls* variable lies from the *churn* data set that we will explore in Chapter 3. IBM/SPSS Modeler supplies us with the statistical summaries shown in Figure 2.7. The mean number of customer service calls for this sample of $n = 3333$ customers is given as $\bar{x} = 1.563$. Using the *sum* and the *count* statistics, we can verify that

$$\bar{x} = \frac{\sum x}{n} = \frac{5209}{3333} = 1.563$$

For variables that are not extremely skewed, the mean is usually not too far from the variable center. However, for extremely skewed data sets, the mean becomes less representative of the variable center. Also, the mean is sensitive to the presence of outliers. For this reason, analysts sometimes prefer to work with alternative measures of center, such as the *median*, defined as the field value in the middle when the field values are sorted into ascending order. The median is resistant to the presence of outliers. Other analysts may prefer to use the *mode*, which represents the field value occurring with the greatest frequency. The mode may be used with either numerical or categorical data, but is not always associated with the variable center.



Customer Service Calls	
Statistics	
Count	3333
Mean	1.563
Sum	5209.000
Median	1
Mode	1

Figure 2.7 Statistical summary of *customer service calls*.

Note that measures of center do not always concur as to where the center of the data set lies. In Figure 2.7, the median is 1, which means that half of the customers made at least one customer service call; the mode is also 1, which means that the most frequent number of customer service calls was 1. The median and mode agree. However, the mean is 1.563, which is 56.3% higher than the other measures. This is due to the mean's sensitivity to the right-skewness of the data.

Measures of location are not sufficient to summarize a variable effectively. In fact, two variables may have the very same values for the mean, median, and mode, and yet have different natures. For example, suppose that stock portfolio A and stock portfolio B contained five stocks each, with the price/earnings (P/E) ratios as shown in Table 2.3. The portfolios are distinctly different in terms of P/E ratios. Portfolio A includes one stock that has a very small P/E ratio and another with a rather large P/E ratio. On the other hand, portfolio B's P/E ratios are more tightly clustered around the mean. But despite these differences, the mean, median, and mode of the portfolios, P/E ratios are precisely the same: The mean P/E ratio is 10, the median is 11, and the mode is 11 for each portfolio.

Clearly, these measures of center do not provide us with a complete picture. What is missing are *measures of spread* or *measures of variability*, which will describe how spread out the data values are. Portfolio A's P/E ratios are more spread out than those of portfolio B, so the measures of variability for portfolio A should be larger than those of B.

Typical measures of variability include the *range* (maximum – minimum), the standard deviation, the mean absolute deviation, and the interquartile range. The

TABLE 2.3 The two portfolios have the same mean, median, and mode, but are clearly different

Stock Portfolio A	Stock Portfolio B
1	7
11	8
11	11
11	11
16	13

sample *standard deviation* is perhaps the most widespread measure of variability and is defined by

$$s = \sqrt{\frac{\sum (x - \bar{x})^2}{n - 1}}$$

Because of the squaring involved, the standard deviation is sensitive to the presence of outliers, leading analysts to prefer other measures of spread, such as the *mean absolute deviation*, in situations involving extreme values.

The standard deviation can be interpreted as the “typical” distance between a field value and the mean, and most field values lie within two standard deviations of the mean. From Figure 2.7 we can state that the number of customer service calls made by most customers lies within $2(1.315) = 2.63$ of the mean of 1.563 calls. In other words, most of the number of customer service calls lie within the interval $(-1.067, 4.193)$, that is, $(0, 4)$. (This can be verified by examining the histogram of customer service calls in Figure 3.14 in Chapter 3.)

More information about these statistics may be found in the Appendix. A more complete discussion of measures of location and variability can be found in any introductory statistics textbook, such as Larose [2].

2.7 DATA TRANSFORMATION

Variables tend to have ranges that vary greatly from each other. For example, if we are interested in major league baseball, players’ batting averages will range from zero to less than 0.400, while the number of home runs hit in a season will range from zero to around 70. For some data mining algorithms, such differences in the ranges will lead to a tendency for the variable with greater range to have undue influence on the results. That is, the greater variability in home runs will dominate the lesser variability in batting averages.

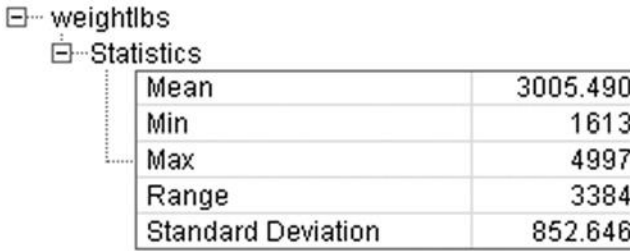
Therefore, data miners should *normalize* their numeric variables, in order to standardize the scale of effect each variable has on the results. Neural networks benefit from *normalization*, as do algorithms that make use of distance measures, such as the *k*-nearest neighbor algorithm. There are several techniques for normalization, and we shall examine three of the more prevalent methods. Let X refer to our original field value, and X^* refer to the normalized field value.

2.8 MIN-MAX NORMALIZATION

Min-max normalization works by seeing how much greater the field value is than the minimum value $\min(X)$, and scaling this difference by the range. That is

$$X_{\text{mm}}^* = \frac{X - \min(X)}{\text{range}(X)} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

The summary statistics for *weight* are shown in Figure 2.8.



Mean	3005.490
Min	1613
Max	4997
Range	3384
Standard Deviation	852.646

Figure 2.8 Summary statistics for *weight*.

The minimum weight is 1613 pounds, and the range = $\max(X) - \min(X) = 4997 - 1613 = 3384$ pounds. Let us find the min-max normalization for three automobiles weighing 1613 pounds, 3384 pounds, and 4997, respectively.

- For an ultra-light vehicle, weighing only 1613 pounds (the field minimum), the min-max normalization is

$$X_{\text{mm}}^* = \frac{X - \min(X)}{\text{range}(X)} = \frac{1613 - 1613}{3384} = 0$$

Thus, data values that represent the minimum for the variable will have a min-max normalization value of zero.

- The *midrange* equals the average of the maximum and minimum values in a data set. That is,

$$\text{midrange}(X) = \frac{\max(X) + \min(X)}{2} = \frac{4997 + 1613}{2} = 3305 \text{ pounds}$$

For a “midrange” vehicle (if any), which weighs exactly halfway between the minimum weight and the maximum weight, the min-max normalization is

$$X_{\text{mm}}^* = \frac{X - \min(X)}{\text{range}(X)} = \frac{3305 - 1613}{3384} = 0.5$$

So the midrange data value has a min-max normalization value of 0.5.

- The heaviest vehicle has a min-max normalization value of

$$X_{\text{mm}}^* = \frac{X - \min(X)}{\text{range}(X)} = \frac{4997 - 1613}{3384} = 1$$

That is, data values representing the field maximum will have a min-max normalization of 1. To summarize, min-max normalization values will range from 0 to 1.

2.9 Z-SCORE STANDARDIZATION

Z-score standardization, which is very widespread in the world of statistical analysis, works by taking the difference between the field value and the field mean value, and scaling this difference by the standard deviation of the field values. That is

$$\text{Z-score} = \frac{X - \text{mean}(X)}{\text{SD}(X)}$$

Figure 2.8 tells us that $\text{mean}(\text{weight}) = 3005.49$ and $\text{SD}(\text{weight}) = 852.49$.

- For the vehicle weighing only 1613 pounds, the Z-score standardization is

$$\text{Z-score} = \frac{X - \text{mean}(X)}{\text{SD}(X)} = \frac{1613 - 3005.49}{852.49} \approx -1.63$$

Thus, data values that lie below the mean will have a negative Z-score standardization.

- For an “average” vehicle (if any), with a weight equal to $\text{mean}(X) = 3005.49$ pounds, the Z-score standardization is

$$\text{Z-score} = \frac{X - \text{mean}(X)}{\text{SD}(X)} = \frac{3005.49 - 3005.49}{852.49} = 0.$$

That is, values falling exactly on the mean will have a Z-score standardization of zero.

- For the heaviest car, the Z-score standardization is

$$\text{Z-score} = \frac{X - \text{mean}(X)}{\text{SD}(X)} = \frac{4997 - 3005.49}{852.49} \approx 2.34.$$

That is, data values that lie above the mean will have a positive Z-score standardization.⁴

2.10 DECIMAL SCALING

Decimal scaling ensures that every normalized value lies between -1 and 1 .

$$X_{\text{decimal}}^* = \frac{X}{10^d}$$

where d represents the number of digits in the data value with the largest absolute value. For the weight data, the largest absolute value is $|4997| = 4997$, which has $d = 4$ digits. The decimal scaling for the minimum and maximum weights is

$$\text{Min: } X_{\text{decimal}}^* = \frac{1613}{10^4} = 0.1613 \quad \text{Max: } X_{\text{decimal}}^* = \frac{4997}{10^4} = 0.4997$$

2.11 TRANSFORMATIONS TO ACHIEVE NORMALITY

Some data mining algorithms and statistical methods require that the variables be *normally distributed*. The normal distribution is a continuous probability distribution commonly known as the bell curve, which is symmetric. It is centered at mean μ (“myu”) and has its spread determined by standard deviation σ (sigma). Figure 2.9 shows the normal distribution that has mean $\mu = 0$ and standard deviation $\sigma = 1$, known as the *standard normal distribution Z*.

⁴Also, for a given Z-score, we may find its associated data value. See the Appendix.

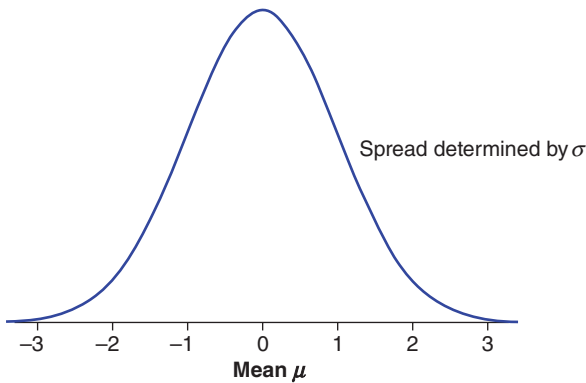


Figure 2.9 Standard normal Z distribution.

It is a common misconception that variables that have had the Z-score standardization applied to them follow the standard normal Z distribution. This is not correct! It is true that the Z-standardized data will have mean 0 and standard deviation 1 (see Figure 2.14), but the distribution may still be skewed. Compare the histogram of the original *weight* data in Figure 2.10 with the Z-standardized data in Figure 2.11. Both histograms are right-skewed; in particular, Figure 2.10 is not symmetric, and so cannot be normally distributed.

We use the following statistic to measure the *skewness* of a distribution⁵:

$$\text{Skewness} = \frac{3(\text{mean} - \text{median})}{\text{standard deviation}}$$

For right-skewed data, the mean is greater than the median, and thus the skewness will be positive (Figure 2.12), while for left-skewed data, the mean is smaller

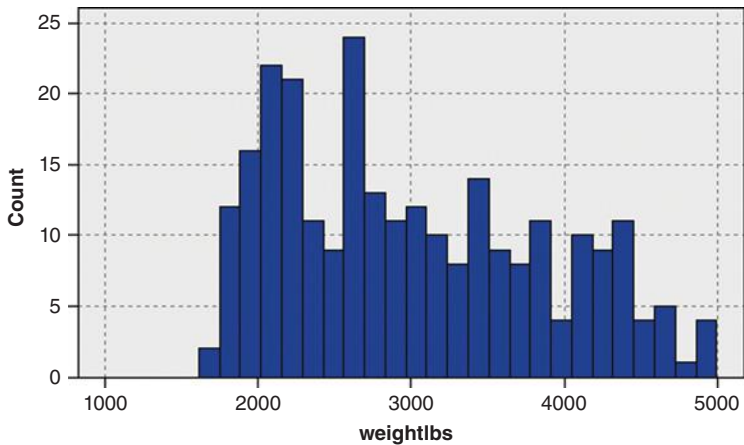


Figure 2.10 Original data.

⁵Find more about standard deviations in the Appendix.

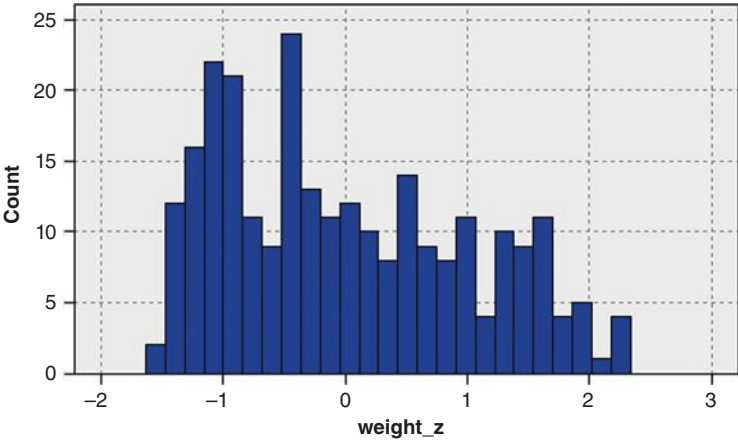


Figure 2.11 Z-Standardized data are still right-skewed, not normally distributed.

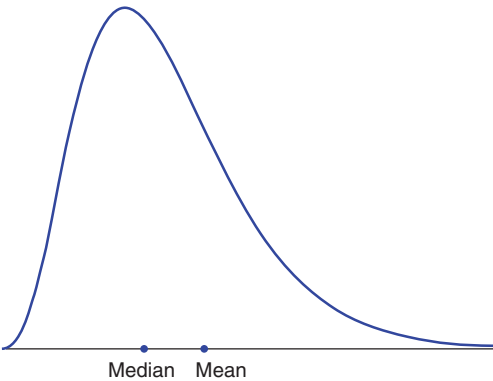


Figure 2.12 Right-skewed data have positive skewness.

than the median, generating negative values for skewness (Figure 2.13). For perfectly symmetric data (such as in Figure 2.9) of course, the mean, median, and mode are all equal, and so the skewness equals zero.

Much real-world data are right-skewed, including most financial data. Left-skewed data are not as common, but often occurs when the data are right-censored, such as test scores on an easy test, which can get no higher than 100. We use the statistics for *weight* and *weight_Z* shown in Figure 2.14 to calculate the skewness for these variables.

For *weight* we have

$$\text{Skewness} = \frac{3(\text{mean} - \text{median})}{\text{standard deviation}} = \frac{3(3005.490 - 2835)}{852.646} = 0.6$$

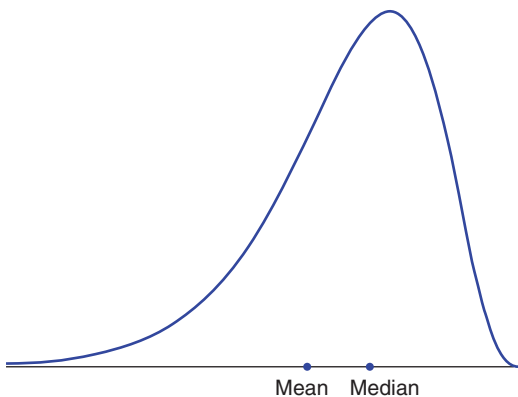


Figure 2.13 Left-skewed data have negative skewness.

For *weight_Z* we have

$$\text{Skewness} = \frac{3(\text{mean} - \text{median})}{\text{standard deviation}} = \frac{3(0 - (-0.2))}{1} = 0.6$$

Thus, Z-score standardization has *no effect* on skewness.

To make our data “more normally distributed,” we must first make it symmetric, which means eliminating the skewness. To eliminate skewness, we apply a *transformation* to the data. Common transformations are the natural log transformation $\ln(\text{weight})$, the square root transformation $\sqrt{\text{weight}}$, and the inverse square root transformation $1/\sqrt{\text{weight}}$. Application of the square root transformation (Figure 2.15) somewhat reduces the skewness, while applying the \ln transformation (Figure 2.16) reduces skewness even further.

weightlbs	
Statistics	
Mean	3005.490
Standard Deviation	852.646
Median	2835

weight_Z	
Statistics	
Mean	0.000
Standard Deviation	1.000
Median	-0.200

Figure 2.14 Statistics for calculating skewness.

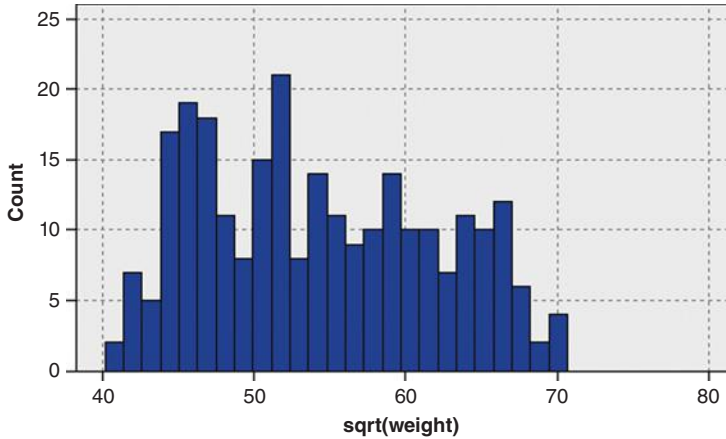


Figure 2.15 Square root transformation somewhat reduces skewness.

The statistics in Figure 2.17 is used to calculate the reduction in skewness:

$$\text{Skewness}(\text{sqrt}(\text{weight})) = \frac{3(54.280 - 53.245)}{7.709} \approx 0.40$$

$$\text{Skewness}(\ln(\text{weight})) = \frac{3(7.968 - 7.950)}{0.284} \approx 0.19$$

Finally, we try the inverse square root transformation $1/\sqrt{\text{weight}}$, which gives us the distribution in Figure 2.18. The statistics in Figure 2.19 gives us

$$\text{Skewness}(\text{inverse_sqrt}(\text{weight})) = \frac{3(0.019 - 0.019)}{0.003} = 0$$

which indicates that we have eliminated the skewness and achieved a symmetric distribution.

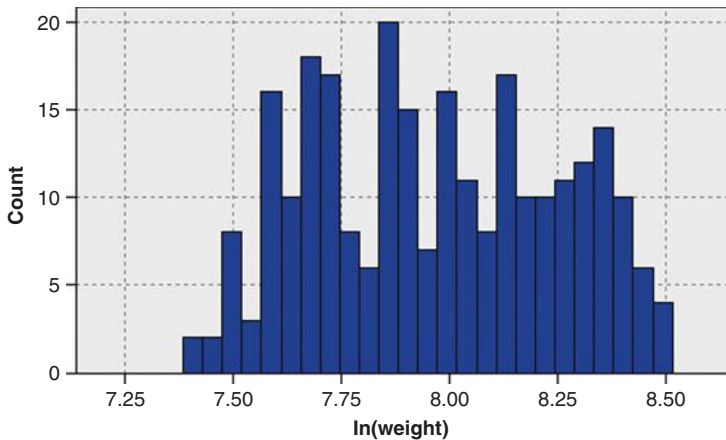


Figure 2.16 Natural log transformation reduces skewness even further.

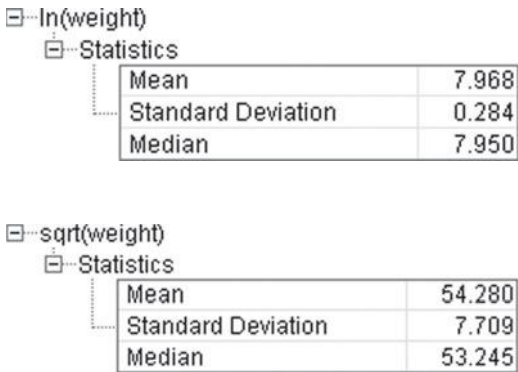


Figure 2.17 Statistics for calculating skewness.

Now, there is nothing magical about the inverse square root transformation; it just happened to work for the amount of skewness present in this variable.

Though we have achieved symmetry, we still have not arrived at normality. To check for normality, we construct a *normal probability plot*, which plots the quantiles of a particular distribution against the quantiles of the standard normal distribution. Similar to a *percentile*, the p^{th} quantile of a distribution is the value x_p such that $p\%$ of the distribution values are less than or equal to x_p .

In a normal probability plot, if the distribution is normal, the bulk of the points in the plot should fall on a straight line; systematic deviations from linearity in this plot indicate nonnormality. Note from Figure 2.18 that the distribution is not a good fit for the normal distribution curve shown. Thus, we would not expect our normal probability plot to exhibit normality. As expected, the normal probability plot of

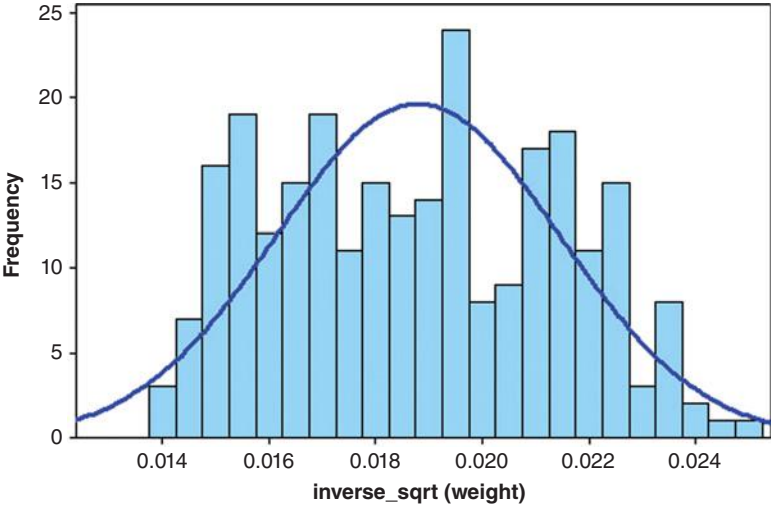


Figure 2.18 The transformation *inverse_sqrt (weight)* has eliminated the skewness, but is still not normal.

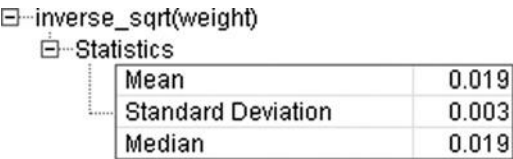


Figure 2.19 Statistics for *inverse_sqrt(weight)*.

inverse_sqrt(weight) in Figure 2.20 shows systematic deviations from linearity, indicating nonnormality. For contrast, a normal probability plot of normally distributed data is shown in Figure 2.21; this graph shows no systematic deviations from linearity.

Experimentation with further transformations (not shown) did not yield acceptable normality for *inverse_sqrt(weight)*. Fortunately, algorithms requiring normality usually do fine when supplied with data that is symmetric and unimodal.

Finally, when the algorithm is done with its analysis, *do not forget to “de-transform” the data.* Let *x* represent the original variable, and *y* represent the transformed variable. Then, for the inverse square root transformation, we have

$$y = \frac{1}{\sqrt{x}}$$

“De-transforming,” we obtain: $x = \frac{1}{y^2}$. Results that your algorithm provided on the transformed scale would have to be de-transformed using this formula.⁶

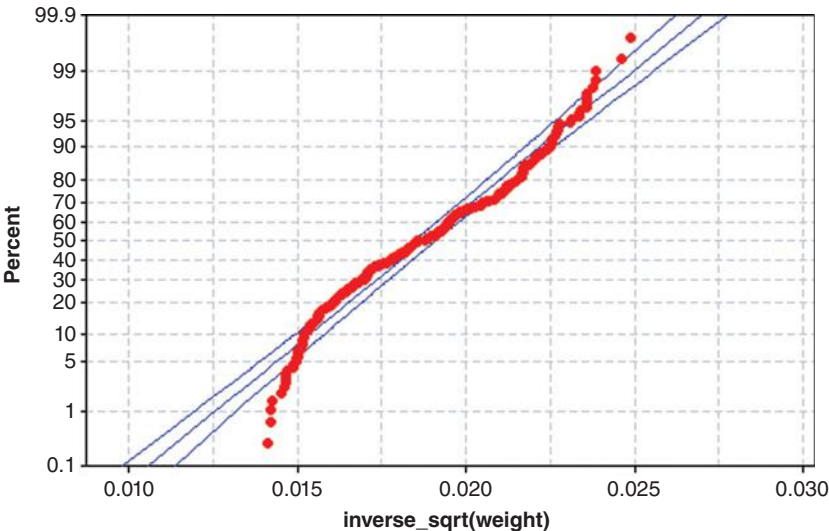


Figure 2.20 Normal probability plot of *inverse_sqrt(weight)* indicates nonnormality.

⁶For more on data transformations, see *Data Mining Methods and Models*, by Daniel Larose (Wiley, 2006) or *Data Mining and Predictive Analytics*, by Daniel Larose and Chantal Larose (Wiley, 2015, to appear).

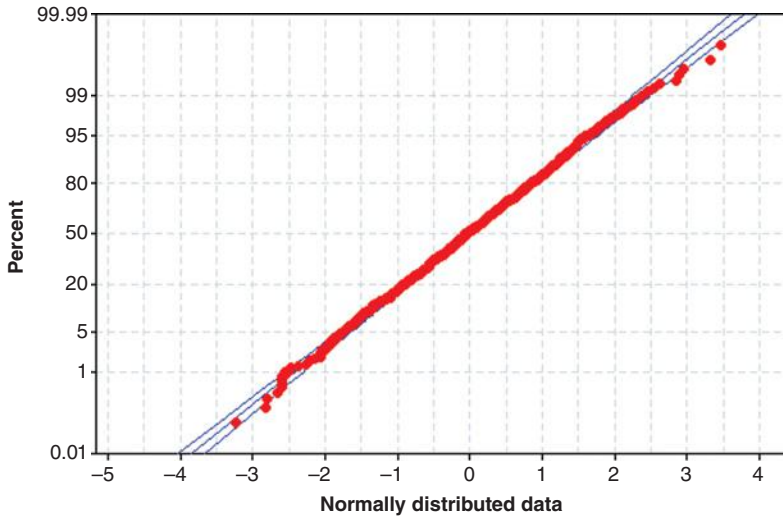


Figure 2.21 Normal probability plot of normally distributed data.

2.12 NUMERICAL METHODS FOR IDENTIFYING OUTLIERS

The *Z-score method for identifying outliers* states that a data value is an outlier if it has a *Z*-score that is either less than -3 or greater than 3 . Variable values with *Z*-scores much beyond this range may bear further investigation, in order to verify that they do not represent data entry errors or other issues. On the other hand, one should not automatically omit outliers from analysis.

We saw that the minimum *Z*-score was for the vehicle weighing only 1613 pounds, and having a *Z*-score of -1.63 , while the maximum *Z*-score was for the 4997-pound vehicle, with a *Z*-score of 2.34 . Since neither *Z*-score is either less than -3 or greater than 3 , we conclude that there are no outliers among the vehicle weights.

Unfortunately, the mean and standard deviation, which are both part of the formula for the *Z*-score standardization, are both rather *sensitive* to the presence of outliers. That is, if an outlier is added to (or deleted from) a data set, the values of mean and standard deviation will both be unduly affected by the presence (or absence) of this new data value. Therefore, when choosing a method for evaluating outliers, it may not seem appropriate to use measures which are themselves sensitive to their presence.

Therefore, data analysts have developed more *robust* statistical methods for outlier detection, which are less sensitive to the presence of the outliers themselves. One elementary robust method is to use the *interquartile range* (IQR). The *quartiles* of a data set divide the data set into four parts, each containing 25% of the data.

- The *first quartile* (Q_1) is the 25th percentile.
- The *second quartile* (Q_2) is the 50th percentile, that is, the median.
- The *third quartile* (Q_3) is the 75th percentile.

Then, the *interquartile range* (*IQR*) is a measure of variability, much more robust than the standard deviation. The *IQR* is calculated as $IQR = Q3 - Q1$, and may be interpreted to represent the spread of the middle 50% of the data.

A robust measure of outlier detection is therefore defined as follows. A data value is an outlier if

- a. It is located $1.5(IQR)$ or more below $Q1$ or
- b. It is located $1.5(IQR)$ or more above $Q3$.

For example, suppose for a set of test scores, the 25th percentile was $Q1 = 70$ and the 75th percentile was $Q3 = 80$, so that half of all the test scores fell between 70 and 80. Then the *interquartile range*, or the difference between these quartiles was $IQR = 80 - 70 = 10$.

A test score would be robustly identified as an outlier if

- a. It is lower than $Q1 - 1.5(IQR) = 70 - 1.5(10) = 55$ or
- b. It is higher than $Q3 + 1.5(IQR) = 80 + 1.5(10) = 95$.

2.13 FLAG VARIABLES

Some analytical methods, such as regression, require predictors to be numeric. Thus, analysts wishing to use categorical predictors in regression need to recode the categorical variable into one or more *flag variables*. A **flag variable** (or **dummy variable** or **indicator variable**) is a categorical variable taking only two values, 0 and 1. For example, the categorical predictor *sex*, taking values *female* and *male*, could be recoded into the flag variable *sex_flag* as follows:

If $sex = female$ then $sex_flag = 0$; if $sex = male$ then $sex_flag = 1$.

When a categorical predictor takes $k \geq 3$ possible values, then define $k - 1$ dummy variables, and use the unassigned category as the *reference category*. For example, if a categorical predictor *region* has $k = 4$ possible categories, $\{north, east, south, west\}$, then the analyst could define the following $k - 1 = 3$ flag variables.

north_flag : If $region = north$ then $north_flag = 1$; otherwise $north_flag = 0$.
east_flag : If $region = east$ then $east_flag = 1$; otherwise $east_flag = 0$.
south_flag : If $region = south$ then $south_flag = 1$; otherwise $south_flag = 0$.

The flag variable for the west is not needed, since, $region = west$ is already uniquely identified by zero values for each of the three existing flag variables. (Further, inclusion of the fourth flag variable will cause some algorithms to fail, because of the singularity of the $(X'X)^{-1}$ matrix in regression, for instance.) Instead, the unassigned category becomes the reference category, meaning that, the interpretation of the value of *north_flag* is $region = north$ compared to $region = west$. For example, if we are running a regression analysis with income as the target variable, and the regression coefficient (see Chapter 5) for *north_flag* equals \$1000, then the estimated income

for *region = north* is \$1000 greater than for *region = west*, when all other predictors are held constant.

2.14 TRANSFORMING CATEGORICAL VARIABLES INTO NUMERICAL VARIABLES

Would not it be easier to simply transform the categorical variable *region* into a single numerical variable rather than using several different flag variables? For example, suppose we defined the quantitative variable *region_num* as follows:

Region	Region_num
North	1
East	2
South	3
West	4

Unfortunately, this is a *common and hazardous error*. The algorithm now erroneously thinks the following:

- The four regions are ordered,
- West > South > East > North,
- West is three times closer to South compared to North, and so on.

So, in most instances, the data analyst should avoid transforming categorical variables to numerical variables. The exception is for categorical variables that are clearly ordered, such as the variable *survey_response*, taking values *always*, *usually*, *sometimes*, *never*. In this case, one could assign numerical values to the responses, though one may bicker with the actual values assigned, such as:

Survey response	Survey Response_num
Always	4
Usually	3
Sometimes	2
Never	1

Should *never* be “0” rather than “1”? Is *always* closer to *usually* than *usually* is to *sometimes*? Careful assignment of the numerical values is important.

2.15 BINNING NUMERICAL VARIABLES

Some algorithms prefer categorical rather than continuous predictors⁷, in which case we would need to partition any numerical predictors into *bins* or *bands*. For example, we may wish to partition the numerical predictor *house value* into *low*, *medium*, and *high*. There are four common methods for binning numerical predictors:

1. **Equal width binning** divides the numerical predictor into k categories of equal width, where k is chosen by the client or analyst.
2. **Equal frequency binning** divides the numerical predictor into k categories, each having k/n records, where n is the total number of records.
3. **Binning by clustering** uses a clustering algorithm, such as *k-means clustering* (Chapter 10) to automatically calculate the “optimal” partitioning.
4. **Binning based on predictive value.** Methods (1)–(3) ignore the target variable; binning based on predictive value partitions the numerical predictor based on the effect each partition has on the value of the target variable. Chapter 3 contains an example of this.

Equal width binning is not recommended for most data mining applications, since the width of the categories can be greatly affected by the presence of outliers. Equal frequency distribution assumes that each category is equally likely, an assumption which is usually not warranted. Therefore, methods (3) and (4) are preferred.

Suppose we have the following tiny data set, which we would like to discretize into $k = 3$ categories: $X = \{1, 1, 1, 1, 1, 2, 2, 11, 11, 12, 12, 44\}$.

1. Using equal width binning, we partition X into the following categories of equal width, illustrated in Figure 2.22a:
 - *Low*: $0 \leq X < 15$, which contains all the data values except one.
 - *Medium*: $15 \leq X < 30$, which contains no data values at all.
 - *High*: $30 \leq X < 45$, which contains a single outlier.
2. Using equal frequency binning, we have $n = 12$, $k = 3$, and $n/k = 4$. The partition is illustrated in Figure 2.22b.
 - *Low*: Contains the first four data values, all $X = 1$.
 - *Medium*: Contains the next four data values, $X = \{1, 2, 2, 11\}$.
 - *High*: Contains the last four data values, $X = \{11, 12, 12, 44\}$.

Note that one of the *medium* data values equals a data value in the *low* category, and another equals a data value in the *high* category. This violates what should be a self-evident heuristic: Equal data values should belong to the same category.

3. Finally, *k-means clustering* identifies what seems to be the intuitively correct partition, as shown in Figure 2.22c.

⁷For further information about discrete and continuous variables, as well as other ways of classifying variables, see the Appendix.

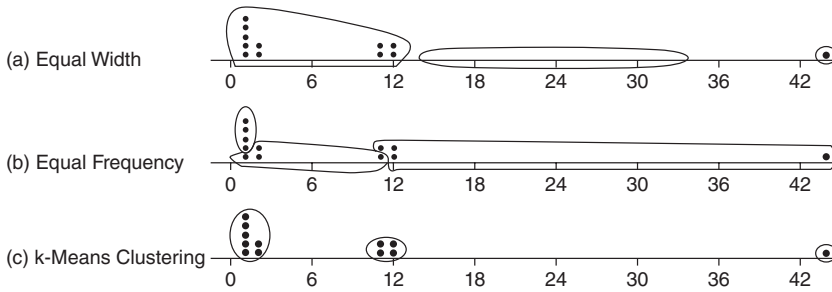


Figure 2.22 Illustration of binning methods.

We provide two examples of binning based on predictive value in Chapter 3.

2.16 RECLASSIFYING CATEGORICAL VARIABLES

Reclassifying categorical variables is the categorical equivalent of binning numerical variables. Often, a categorical variable will contain too many field values to be easily analyzable. For example, the predictor *state* could contain 50 different field values. Data mining methods such as logistic regression and the C4.5 decision tree algorithm perform suboptimally when confronted with predictors containing too many field values. In such a case, the data analyst should reclassify the field values. For example, the 50 states could each be reclassified as the variable *region*, containing field values *Northeast*, *Southeast*, *North Central*, *Southwest*, and *West*. Thus, instead of 50 different field values, the analyst (and algorithm) is faced with only 5. Alternatively, the 50 states could be reclassified as the variable *economic_level*, with three field values containing the richer states, the midrange states, and the poorer states. The data analyst should choose a reclassification that supports the objectives of the business problem or research question.

2.17 ADDING AN INDEX FIELD

It is recommended that the data analyst create an index field, which tracks the sort order of the records in the database. Data mining data gets partitioned at least once (and sometimes several times). It is helpful to have an index field so that the original sort order may be recreated. For example, using IBM/SPSS Modeler, you can use the *@Index* function in the *Derive* node to create an index field.

2.18 REMOVING VARIABLES THAT ARE NOT USEFUL

The data analyst may wish to remove variables that will not help the analysis, regardless of the proposed data mining task or algorithm. Such variables include

- Unary variables
- Variables which are very nearly unary

Unary variables take on only a single value, so a unary variable is not so much a variable as a *constant*. For example, data collection on a sample of students at an all-girls private school would find that the *sex* variable would be unary, since every subject would be female. Since *sex* is constant across all observations, it cannot have any effect on any data mining algorithm or statistical tool. The variable should be removed.

Sometimes a variable can be very nearly unary. For example, suppose that 99.95% of the players in a field hockey league are female, with the remaining 0.05% male. The variable *sex* is therefore very nearly, but not quite, unary. While it may be useful to investigate the male players, some algorithms will tend to treat the variable as essentially unary. For example, a classification algorithm can be better than 99.9% confident that a given player is female. So, the data analyst needs to weigh how close to unary a given variable is, and whether such a variable should be retained or removed.

2.19 VARIABLES THAT SHOULD PROBABLY NOT BE REMOVED

It is (unfortunately) a common—though questionable—practice to remove from analysis the following types of variables:

- Variables for which 90% or more of the values are missing.
- Variables which are strongly correlated.

Before you remove a variable because it has 90% or more missing values, consider that there may be a pattern in the missingness, and therefore useful information, that you may be jettisoning. Variables which contain 90% missing values present a challenge to any strategy for imputation of missing data (see Chapter 13). For example, are the remaining 10% of the cases are truly representative of the missing data, or are the missing values occurring due to some systematic but unobserved phenomenon? For example, suppose we have a field called *donation_dollars* in a self-reported survey database. Conceivably, those who donate a lot would be inclined to report their donations, while those who do not donate much may be inclined to skip this survey question. Thus, the 10% who report are not representative of the whole. In this case, it may be preferable to construct a flag variable, *donation_flag*, since there is a pattern in the missingness which may turn out to have predictive power.

However, if the data analyst has reason to believe that the 10% are representative, then he or she may choose to proceed with the imputation of the missing 90%. It is strongly recommended that the imputation be based on the regression or decision tree methods shown in Chapter 13. Regardless of whether the 10% are representative of the whole or not, the data analyst may decide that it is wise to construct a flag variable for the non-missing values, since they may very well be useful for prediction or classification. Also, there is nothing special about the 90% figure; the data analyst

may use any large proportion he or she considers warranted. Bottom line: one should avoid removing variables just because they have lots of missing values.

An example of correlated variables may be precipitation and attendance at a state beach. As precipitation increases, attendance at the beach tends to decrease, so that the variables are negatively correlated⁸. Inclusion of correlated variables may at best double-count a particular aspect of the analysis, and at worst lead to instability of the model results. When confronted with two strongly correlated variables, therefore, some data analysts may decide to simply remove one of the variables. We advise against doing so, since important information may thereby be discarded. Instead, it is suggested that principal component analysis be applied, where the common variability in correlated predictors may be translated into a set of uncorrelated principal components⁹.

2.20 REMOVAL OF DUPLICATE RECORDS

During a database's history, records may have been inadvertently copied, thus creating duplicate records. Duplicate records lead to an overweighting of the data values in those records, so, if the records are truly duplicate, only one set of them should be retained. For example, if the ID field is duplicated, then definitely remove the duplicate records. However, the data analyst should apply common sense. To take an extreme case, suppose a data set contains three nominal fields, and each field takes only three values. Then there are only $3 \times 3 \times 3 = 27$ possible different sets of observations. In other words, if there are more than 27 records, at least one of them has to be a duplicate. So, the data analyst should weigh the likelihood that the duplicates represent truly different records against the likelihood that the duplicates are indeed just duplicated records.

2.21 A WORD ABOUT ID FIELDS

Because ID fields have a different value for each record, they will not be helpful for your downstream data mining algorithms. They may even be hurtful, with the algorithm finding some spurious relationship between ID field and your target. Thus it is recommended that ID fields should be filtered out from the data mining algorithms, but should not be removed from the data, so that the data analyst can differentiate between similar records.

In Chapter 3, *Exploratory Data Analysis*, we apply some basic graphical and statistical tools to help us begin to uncover simple patterns and trends in the data structure.

⁸For more on correlation, see the Appendix.

⁹For more on principal component analysis, see *Data Mining Methods and Models*, by Daniel Larose (Wiley, 2006) or *Data Mining and Predictive Analytics*, by Daniel Larose and Chantal Larose (Wiley, 2015, to appear).

THE R ZONE

Getting Started with R

R is a powerful, open-source language for exploring and analyzing data sets. Analysts using R can take advantage of many freely available packages, routines, and graphical user interfaces, to tackle most data analysis problems. Go to www.r-project.org, select “download R,” choose a CRAN mirror close to your location, click the download link that applies to your operating system, and follow the instructions to install R for the first time.

The format we will follow in much of The R Zone is to present the R code in the left column, and the associated output in the right column. Sometimes, the R code takes up both the columns.

How to Handle Missing Data: Example Using the Cars Data Set

Note: Lines beginning with “#” are comment lines explaining what we are doing. The R compiler skips these lines. Lines indented (e.g., Create a Histogram, below) are meant to be on the same line as the one above. A semicolon tells R to separate one line into two lines, one before and one after the semicolon.

Input data set Cars into Data Frame “cars”

```
cars <- read.csv(file = "C:/.../cars.txt", > cars
               stringsAsFactors = FALSE)
# Show the new Data Frame "cars"
cars
# (only the first seven records shown
# in output)
```

	mpg	cylinders	cubicinches	hp	weight	time.to.60	year	brand
1	14.0	8	350	165	4209	12	1972	US
2	31.9	4	89	71	1925	14	1980	Europe
3	17.0	8	302	140	3449	11	1971	US
4	15.0	8	400	150	3761	10	1971	US
5	30.5	4	98	63	2051	17	1978	US
6	23.0	8	350	125	3900	17	1980	US
7	13.0	8	351	158	4363	13	1974	US

Create subset of "cars"

```
# Use records 1 to 5, variables 1, 3, 4, and 8.
cars_tiny <- cars[1:5,c(1, 3, 4, 8)]
cars_tiny
```

```
> cars_tiny
   mpg cubicinches  hp  brand
1 14.0         350 165    US
2 31.9          89  71 Europe
3 17.0         302 140    US
4 15.0         400 150    US
5 30.5          98  63    US
```

Replace the missing value with some constant

```
cars_tiny[2,2] <- cars_tiny[4,4] <- NA
cars_tiny[2,2] <- 0
cars_tiny[4,4] <- "Missing"
cars_tiny
```

```
> cars_tiny
   mpg cubicinches  hp  brand
1 14.0         350 165    US
2 31.9           0  71 Europe
3 17.0         302 140    US
4 15.0         400 150 Missing
5 30.5          98  63    US
```

Replace the missing value with the field mean or mode

```
# Recreate the missing value table
cars_tiny[2,2] <- cars_tiny[4,4] <- NA
# Replace cars_tiny[2,2] with cubicinches mean
cars_tiny[2,2] <-
  mean(na.omit(cars_tiny$cubicinches))
# Replace cars_tiny[4,4] with brand mode
our_table <- table(cars_tiny$brand)
our_mode <- names(our_table)[our_table ==
  max(our_table)]
cars_tiny[4,4] <- our_mode
cars_tiny
```

```
> cars_tiny
  mpg cubicinches  hp  brand
1 14.0      350.0 165    US
2 31.9      287.5  71 Europe
3 17.0      302.0 140    US
4 15.0      400.0 150    US
5 30.5       98.0  63    US
```

Replace missing values with a value generated at random from the observed distribution

```
# Recreate the missing value table
cars_tiny[2,2] <- cars_tiny[4,4] <- NA
# Generate random observation from
# observed distribution; results will vary
obs_brand <-
  sample(na.omit(cars_tiny$brand), 1)
obs_cubicinches <-
  sample(na.omit(cars_tiny$cubicinches), 1)
# Replace the missing values
cars_tiny[2,2] <- obs_cubicinches
cars_tiny[4,4] <- obs_brand
cars_tiny
```

```
> cars_tiny
  mpg cubicinches  hp  brand
1 14.0      350 165    US
2 31.9      400  71 Europe
3 17.0      302 140    US
4 15.0      400 150    US
5 30.5       98  63    US
```

Five Number Summary with Mean

```
summary(cars$weight)
```

Count

```
length(cars$weight)
```

Min-Max Normalization

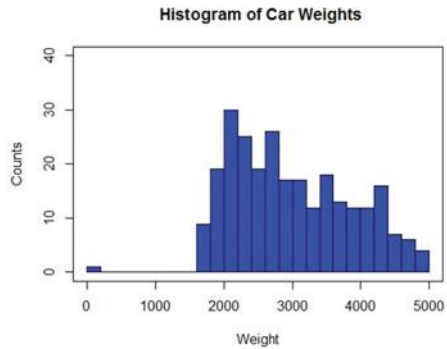
```
mmnorm.weight <- (cars$weight -
  min(cars$weight))/(max(cars$weight) -
  min(cars$weight))
mmnorm.weight
```

Z-score

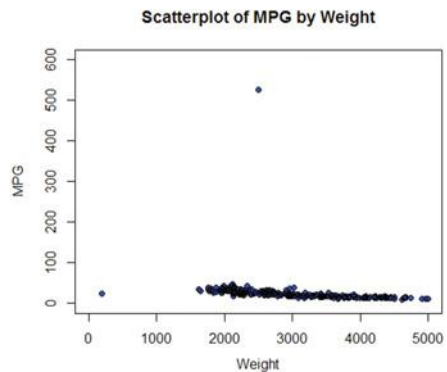
```
zscore.weight <- (cars$weight -
  mean(cars$weight))/sd(cars$weight)
zscore.weight
```

Create a histogram

```
# Input the cars2 dataset
cars2 <- read.csv(file = "C:/.../cars2.txt",
  stringsAsFactors = TRUE)
# Set up the plot area
par(mfrow = c(1,1))
# Create the histogram bars
hist(cars2$weight,
  breaks = 30,
  xlim = c(0, 5000),
  col = "blue",
  border = "black",
  ylim = c(0, 40),
  xlab = "Weight",
  ylab = "Counts",
  main = "Histogram of Car Weights")
# Make a box around the plot
box(which = "plot",
  lty = "solid",
  col="black")
```

**# Create a scatterplot**

```
plot(cars2$weight, cars2$mpg,
  xlim = c(0, 5000),
  ylim = c(0, 600),
  xlab = "Weight",
  ylab = "MPG",
  main = "Scatterplot of MPG by Weight",
  type = "p",
  pch = 16,
  col = "blue")
#Add open black circles
points(cars2$weight,
  cars2$mpg,
  type = "p",
  col = "black")
```

**# Natural Log transformation**

```
natlog_weight <- log(cars$weight)
natlog_weight
```

Inverse Square Root transformation

```
invsqrt_weight <- 1 / sqrt(cars$weight)
invsqrt_weight
```

Calculate skewness

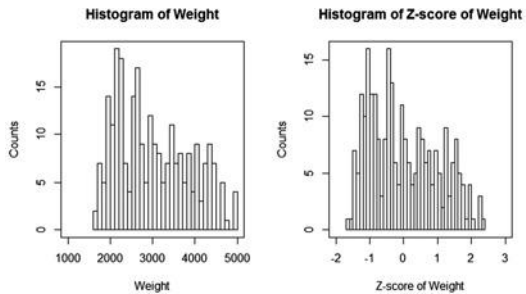
```
weight_skew <- (3*(mean(cars$weight) -
  median(cars$weight))) / sd(cars$weight)
zscore.weight_skew <-
  (3*(mean(zscore.weight) -
    median(zscore.weight))) /
    sd(zscore.weight)
weight_skew; zscore.weight_skew
```

Find the skewness

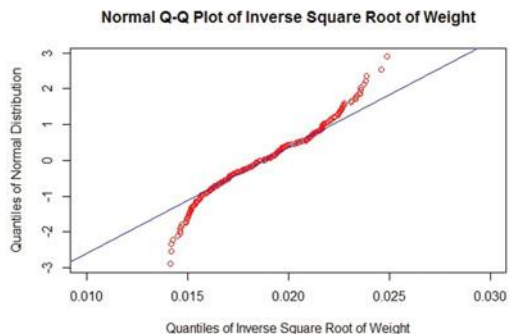
```
lnweight_skew <- (3*(mean(natlog_weight) -
  median(natlog_weight))) /
  sd(natlog_weight)
lnweight_skew
```

Side-by-Side Histograms of Weight and Z-score of Weight

```
par(mfrow = c(1,2))
hist(cars$weight, breaks = 30,
  xlim = c(1000, 5000),
  main = "Histogram of Weight",
  xlab = "Weight",
  ylab = "Counts")
box(which = "plot",
  lty = "solid",
  col = "black")
hist(zscore.weight,
  breaks = 30,
  xlim = c(-2, 3),
  main = "Histogram of Z-score
    of Weight",
  xlab = "Z-score of Weight",
  ylab = "Counts")
box(which = "plot",
  lty = "solid",
  col = "black")
```

**# Normal probability plot**

```
par(mfrow = c(1,1))
qqnorm(invsqrt_weight,
  datax = TRUE,
  col = "red",
  ylim = c(0.01, 0.03),
  main = "Normal Q-Q Plot of Inverse
    Square Root of Weight")
qqline(invsqrt_weight,
  col = "blue",
  datax = TRUE)
```

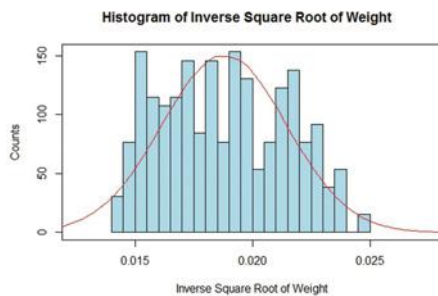


Inverse Square Root skewness

```
invsqrt_weight_skew <- (3*(mean(invsqrt_weight) - median(invsqrt_weight))/sd(invsqrt_weight))
invsqrt_weight_skew
```

Create histogram with fitted Normal distribution

```
# Simulate from a Normal distribution
x <- rnorm(1000000,
  mean = mean(invsqrt_weight),
  sd = sd(invsqrt_weight))
par(mfrow = c(1,1))
hist(invsqrt_weight,
  breaks = 30,
  xlim = c(0.0125, 0.0275),
  col = "lightblue",
  prob = TRUE,
  border = "black",
  xlab = "Inverse Square Root of Weight",
  ylab = "Counts",
  main = "Histogram of Inverse Square Root of
  Weight")
box(which = "plot",
  lty = "solid",
  col = "black")
lines(density(x),
  col = "red")
```

**# Create three flag variables for four categories**

```
# Ten observations, with 999 as our placeholder for unassigned values
north_flag <- east_flag <- south_flag <- c(rep(999, 10))
# Create the variable region
region <- c(rep(c("north", "south", "east",
  "west"), 2), "north", "south")
# Change the flag variable values to 0 or 1
for (i in 1:length(region)) {
  if(region[i] == "north") north_flag[i] = 1
  else north_flag[i] = 0
  if(region[i] == "east") east_flag[i] = 1 else
  east_flag[i] = 0
  if(region[i] == "south") south_flag[i] = 1
  else south_flag[i] = 0
}
north_flag; east_flag; south_flag
```

Transforming the data

```
x <- cars$weight[1]; x
# Transform x using  $y = 1 / \sqrt{x}$ 
y <- 1 / sqrt(x); y
# Detransform x using  $x = 1 / (y)^2$ 
detransformedx <- 1 / y^2; detransformedx
```

Find duplicate records in a data frame

```
anyDuplicated(cars)
# To examine each record, use Duplicated
duplicated(cars)
# 'True' indicates a record which is a duplicate of previous records,
# 'False' indicates a record which is not a duplicate of previous records
# Duplicate the first record in 'data' to make new dataset
new.cars <- rbind(cars, cars[1,])
# Check for duplicates
anyDuplicated(new.cars)
# The 262nd record is a duplicate
duplicated(new.cars)
# 'True' indicates a duplicate of a previous record
```

Create an index field

```
# For data frames
# Data frames already have an index field;
# the left-most column
# The index of a record will stay with that
# record, even if the records are reordered.
cars
cars[order(cars$mpg),]
```

```
# For vectors or matrices
# Add a column to act as an index field
x <- c(1,1,3:1,1:4,3); y <- c(9,9:1)
z <- c(2,1:9)
matrix <- t(rbind(x,y,z)); matrix
indexed_m <- cbind(c(1:length(x)), matrix);
indexed_m
indexed_m[order(z),]
```

Binning

```
# Enter the dataset, call it xdata
xdata <- c(1,1,1,1,1,2,2,11,11,12,12,44)
# Get the sample size of the variable
n <- length(xdata)
#Declare number of bins and bin indicator
nbins <- 3
whichbin <- c(rep(0, n))
# Equal frequency
freq <- n/nbins
# Sort the data
xsorted <- sort(xdata)
for (i in 1:nbins) {
  for (j in 1:n) {
    if((i-1)*freq < j && j <=i*freq)
      whichbin[j] <- i
  }
}
whichbin
```

```
# K-means
kmeansclustering <- kmeans(xdata,
  centers = nbins)
whichbin <- kmeansclustering$cluster;
whichbin

# Equal width
range_xdata <- max(xdata) - min(xdata) + 1
binwidth <- range_xdata/nbins
for (i in 1:nbins) {
  for (j in 1:n) {
    if((i-1)*binwidth < xdata[j] &&
      xdata[j] <= (i)*binwidth)
      whichbin[j] <- i
  }
}
whichbin
```

REFERENCES

1. Gallit Shmueli, Nitin Patel, and Peter Bruce, *Data Mining for Business Intelligence*, 2nd edn, John Wiley and Sons, 2010.
2. Daniel Larose, *Discovering Statistics*, second edition, W.H. Freeman Publishers, New York, 2013.

EXERCISES

1. Describe the possible negative effects of proceeding directly to mine data that has not been preprocessed.
2. Refer to the income attribute of the five customers in Table 2.1, before preprocessing.
 - a. Find the mean income before preprocessing.
 - b. What does this number actually mean?
 - c. Now, calculate the mean income for the three values left after preprocessing. Does this value have a meaning?
3. Explain why zip codes should be considered text variables rather than numeric.
4. What is an outlier? Why do we need to treat outliers carefully?
5. Explain why a birthdate variable would be preferred to an age variable in a database.
6. True or false: All things being equal, more information is almost always better.
7. Explain why it is not recommended, as a strategy for dealing with missing data, to simply omit the records or fields with missing values from the analysis.

- 8. Which of the four methods for handling missing data would tend to lead to an underestimate of the spread (e.g., standard deviation) of the variable? What are some benefits to this method?
- 9. What are some of the benefits and drawbacks for the method for handling missing data that chooses values at random from the variable distribution?
- 10. Of the four methods for handling missing data, which method is preferred?
- 11. Make up a classification scheme which is inherently flawed, and would lead to misclassification, as we find in Table 2.2. For example, classes of items bought in a grocery store.
- 12. Make up a data set, consisting of the heights and weights of six children, in which one of the children is an outlier with respect to one of the variables, but not the other. Then alter this data set so that the child is an outlier with respect to both variables.

Use the following stock price data (in dollars) for Exercises 13–18.

10	7	20	12	75	15	9	18	4	12	8	14
----	---	----	----	----	----	---	----	---	----	---	----

- 13. Calculate the mean, median, and mode stock price.
- 14. Compute the standard deviation of the stock price. Interpret what this number means.
- 15. Find the min-max normalized stock price for the stock worth \$20.
- 16. Calculate the midrange stock price.
- 17. Compute the Z-score standardized stock price for the stock worth \$20.
- 18. Find the decimal scaling stock price for the stock worth \$20.
- 19. Calculate the skewness of the stock price data.
- 20. Explain why data analysts need to normalize their numeric variables.
- 21. Describe three characteristics of the standard normal distribution.
- 22. If a distribution is symmetric, does it follow that it is normal? Give a counterexample.
- 23. What do we look for in a normal probability plot to indicate nonnormality?
Use the stock price data for Exercises 24–26.
- 24. Do the following.
 - a. Identify the outlier.
 - b. Verify that this value is an outlier, using the Z-score method.
 - c. Verify that this value is an outlier, using the IQR method.
- 25. Identify all possible stock prices that would be outliers, using:
 - a. The Z-score method.
 - b. The IQR method.
- 26. Investigate how the outlier affects the mean and median by doing the following.
 - a. Find the mean score and the median score, with and without the outlier.
 - b. State which measure, the mean or the median, the presence of the outlier affects more, and why.

27. What are the four common methods for binning numerical predictors? Which of these are preferred?

Use the following data set for Exercises 28–30: 1 1 1 3 3 7

28. Bin the data into three bins of equal width (width = 3).
 29. Bin the data into three bins of two records each.
 30. Clarify why each of the binning solutions above are not optimal.
 31. Explain why we might not want to remove a variable that had 90% or more missing values.
 32. Explain why we might not want to remove a variable just because it is highly correlated with another variable.

HANDS-ON ANALYSIS

Use the *churn* data set on the book series website for the following exercises.

33. Explore whether there are missing values for any of the variables.
 34. Compare the area code and state fields. Discuss any apparent abnormalities.
 35. Use a graph to visually determine whether there are any outliers among the number of calls to customer service.
 36. Identify the range of customer service calls that should be considered outliers, using:
 a. The Z-score method, and
 b. The IQR method.
 37. Transform the *day minutes* attribute using Z-score standardization.
 38. Work with skewness as follows.
 a. Calculate the skewness of *day minutes*.
 b. Then calculate the skewness of the Z-score standardized *day minutes*. Comment.
 c. Based on the skewness value, would you consider *day minutes* to be skewed or nearly perfectly symmetric?
 39. Construct a normal probability plot of *day minutes*. Comment on the normality of the data.
 40. Work with *international minutes* as follows.
 a. Construct a normal probability plot of *international minutes*.
 b. What is stopping this variable from being normally distributed.
 c. Construct a flag variable to deal with the situation in (b).
 d. Construct a normal probability plot of the derived variable *nonzero international minutes*. Comment on the normality.
 41. Transform the *night minutes* attribute using Z-score standardization. Using a graph, describe the range of the standardized values. ■

EXPLORATORY DATA ANALYSIS

3.1	HYPOTHESIS TESTING VERSUS EXPLORATORY DATA ANALYSIS	51
3.2	GETTING TO KNOW THE DATA SET	52
3.3	EXPLORING CATEGORICAL VARIABLES	55
3.4	EXPLORING NUMERIC VARIABLES	62
3.5	EXPLORING MULTIVARIATE RELATIONSHIPS	69
3.6	SELECTING INTERESTING SUBSETS OF THE DATA FOR FURTHER INVESTIGATION	71
3.7	USING EDA TO UNCOVER ANOMALOUS FIELDS	71
3.8	BINNING BASED ON PREDICTIVE VALUE	72
3.9	DERIVING NEW VARIABLES: FLAG VARIABLES	74
3.10	DERIVING NEW VARIABLES: NUMERICAL VARIABLES	77
3.11	USING EDA TO INVESTIGATE CORRELATED PREDICTOR VARIABLES	77
3.12	SUMMARY	80
	THE R ZONE	82
	REFERENCE	88
	EXERCISES	88
	HANDS-ON ANALYSIS	89

3.1 HYPOTHESIS TESTING VERSUS EXPLORATORY DATA ANALYSIS

When approaching a data mining problem, a data mining analyst may already have some *a priori* hypotheses that he or she would like to test regarding the relationships between the variables. For example, suppose that cell phone executives are interested in whether a recent increase in the fee structure has led to a decrease in market share. In this case, the analyst would *test* the *hypothesis* that market share has decreased, and would therefore use *hypothesis testing* procedures.

A myriad of statistical hypothesis testing procedures are available through the traditional statistical analysis literature. We cover many of these in Chapters 4 and 5. However, analysts do not always have *a priori* notions of the expected relationships among the variables. Especially when confronted with unknown, large databases, analysts often prefer to use *exploratory data analysis* (EDA), or *graphical data analysis*. EDA allows the analyst to

- Delve into the data set;
- Examine the inter-relationships among the attributes;
- Identify interesting subsets of the observations; and
- Develop an initial idea of possible associations among the predictors, as well as between the predictors and the target variable.

3.2 GETTING TO KNOW THE DATA SET

Graphs, plots, and tables often uncover important relationships that could indicate important areas for further investigation. In this chapter, we use exploratory methods to delve into the **Churn** data set [1] from the UCI Repository of Machine Learning Databases at the University of California, Irvine. The data set is also available on the book series website, www.dataminingconsultant.com. *Churn*, also called attrition, is a term used to indicate a customer leaving the service of one company in favor of another company. The data set contains 20 predictors worth of information about 3333 customers, along with the target variable, *churn*, an indication of whether that customer churned (left the company) or not.

The variables are as follows:

- *State*: Categorical, for the 50 states and the District of Columbia
- *Account Length*: Integer-valued, how long account has been active
- *Area code*: Categorical
- *Phone Number*: Essentially a surrogate for customer ID
- *International Plan*: Dichotomous categorical, yes or no
- *Voice Mail Plan*: Dichotomous categorical, yes or no
- *Number of Voice Mail Messages*: Integer-valued
- *Total Day Minutes*: Continuous, minutes customer used service during the day
- *Total Day Calls*: Integer-valued
- *Total Day Charge*: Continuous, perhaps based on above two variables
- *Total Eve Minutes*: Continuous, minutes customer used service during the evening
- *Total Eve Calls*: Integer-valued
- *Total Eve Charge*: Continuous, perhaps based on above two variables
- *Total Night Minutes*: Continuous, minutes customer used service during the night
- *Total Night Calls*: Integer-valued

- *Total Night Charge*: Continuous, perhaps based on above two variables
- *Total International Minutes*: Continuous, minutes customer used service to make international calls
- *Total International Calls*: Integer-valued
- *Total International Charge*: Continuous, perhaps based on above two variables
- *Number of Calls to Customer Service*: Integer-valued
- *Churn*: Target. Indicator of whether the customer has left the company (True or False)

To begin, it is often best to simply take a look at the field values for some of the records. Figure 3.1 shows the variable values for the first 10 records of the *churn* data set.

We can begin to get a feel for the data by looking at Figure 3.1. We note for example:

- The variable *Phone* uses only seven digits
- There are two flag variables
- Most of our variables are continuous
- The response variable *Churn* is a flag variable having two values, *True* and *False*

Next, we turn to summarization and visualization (see Appendix). Figure 3.2 shows graphs (either histograms or bar charts) and summary statistics for each variable in the data set, except *Phone*, which is an identification field. The variable types for this software (Modeler, by IBM/SPSS) are shown (*set* for categorical, *flag* for flag, and *range* for continuous). We may note that *Vmail messages* has a spike on the length, and that most quantitative variables seem to be normally distributed, except for *Intl Calls* and *CustServ Calls*, which are right-skewed (note that the Skewness

	State	Account Length	Area Code	Phone	Intl Plan	VMail Plan	VMail Message	Day Mins	Day Calls	Day Charge	Eve Mins
1	KS	128	415 382	4657	no	yes	25	265.100	110	45.070	197.400
2	OH	107	415 371	7191	no	yes	26	161.600	123	27.470	195.500
3	NJ	137	415 358	1921	no	no	0	243.400	114	41.380	121.200
4	OH	84	408 375	9999	yes	no	0	299.400	71	50.900	61.900
5	OK	75	415 330	6626	yes	no	0	166.700	113	28.340	148.300
6	AL	118	510 391	8027	yes	no	0	223.400	98	37.980	220.600
7	MA	121	510 355	9993	no	yes	24	218.200	88	37.090	348.500
8	MO	147	415 329	9001	yes	no	0	157.000	79	26.690	103.100
9	LA	117	408 335	4719	no	no	0	184.500	97	31.370	351.600
10	VW	141	415 330	8173	yes	yes	37	258.800	84	43.960	222.000

(a)

	Eve Calls	Eve Charge	Night Mins	Night Calls	Night Charge	Intl Mins	Intl Calls	Intl Charge	CustServ Calls	Churn
1	99	16.780	244.700	91	11.010	10.000	3	2.700		1 False
2	103	16.620	254.400	103	11.450	13.700	3	3.700		1 False
3	110	10.300	162.600	104	7.320	12.200	5	3.290		0 False
4	88	5.260	196.900	89	8.860	6.600	7	1.780		2 False
5	122	12.610	186.900	121	8.410	10.100	3	2.730		3 False
6	101	18.750	203.900	118	9.180	6.300	6	1.700		0 False
7	108	29.620	212.600	118	9.570	7.500	7	2.030		3 False
8	94	8.760	211.800	96	9.530	7.100	6	1.920		0 False
9	80	29.890	215.800	90	9.710	8.700	4	2.350		1 False
10	111	18.870	326.400	97	14.690	11.200	5	3.020		0 False

(b)

Figure 3.1 Field values of the first 10 records in the *churn* data set.

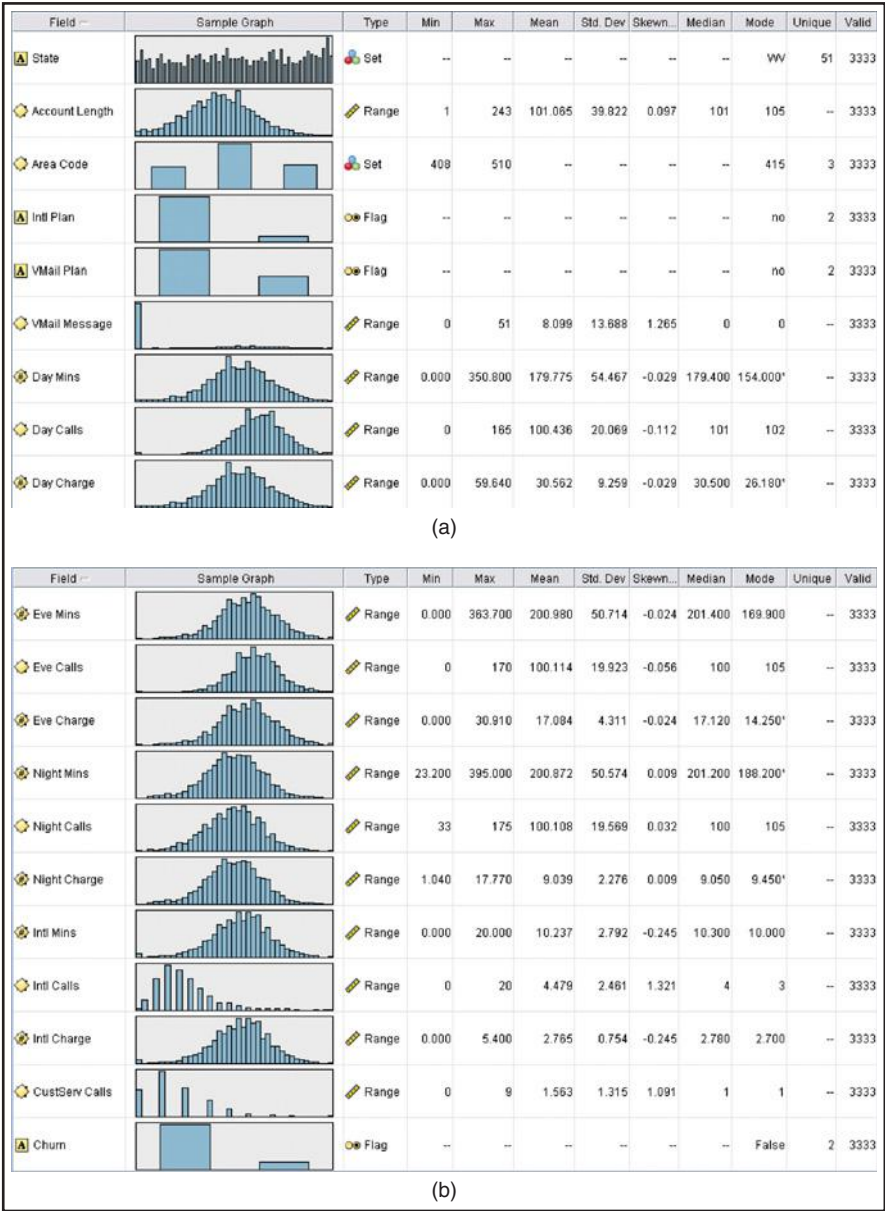


Figure 3.2 Summarization and visualization of the churn data set.

statistic is larger for these variables). *Unique* represents the number of distinct field values. We wonder how it can be that there are 51 distinct values for *State*, but only 3 distinct values for *Area Code*. Also, the mode of *State* being West Virginia may have us scratching our heads a bit. More on this later. We are still just getting to know the data set.

3.3 EXPLORING CATEGORICAL VARIABLES

The bar graph in Figure 3.3 shows the counts and percentages of customers who churned (True) and did not churn (False). Fortunately, only a minority (14.49%) of our customers have left our service. *Our task is to identify patterns in the data that will help to reduce the proportion of churners.*

One of the primary reasons for performing exploratory data analysis is to investigate the variables, examine the distributions of the categorical variables, look at histograms of the numeric variables, and explore the relationships among sets of variables. On the other hand, our overall objective for the data mining project as a whole (not just the EDA phase) is to develop a model of the type of customer likely to churn (jump from your company's service to another company's service). Today's software packages allow us to become familiar with the variables, while at the same time beginning to see which variables are associated with churn. In this way, we can explore the data while keeping an eye on our overall goal. We begin by considering the categorical variables, and their relationship to *churn*.

The first categorical variable we investigate is *International Plan*. Figure 3.4 shows a bar chart of International Plan, with an *overlay* of Churn, and represents a comparison of the proportion of churners and non-churners, among customers who either had selected the International Plan (yes, 9.69% of customers) or had not selected it (no, 90.31% of customers). The graphic appears to indicate that a greater proportion of International Plan holders are churning, but it is difficult to be sure.

In order to “increase the contrast” and better discern whether the proportions differ, we can ask the software (in this case, *IBM/SPSS Modeler*) to provide the same size bars for each category. Thus, in Figure 3.5, we see a graph of the very same information as in Figure 3.4, except that the bar for the *yes* category has been “stretched” out to be the same length as for the *no* category. This allows us to better discern whether the churn proportions differ among the categories. Clearly, those who have selected the International Plan have a greater chance of leaving the company's service than do those who do not have the International Plan.

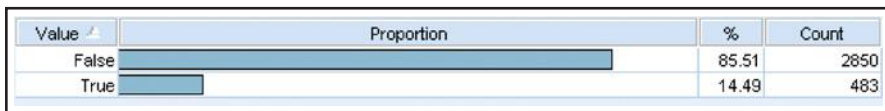


Figure 3.3 Churners and non-churners.



Figure 3.4 Comparison bar chart of churn proportions, by International Plan participation.



Figure 3.5 Comparison bar chart of churn proportions, by International Plan participation, with equal bar length.

The graphics above tell us that International Plan holders tend to churn more frequently, but they do not *quantify* the relationship. In order to quantify the relationship between International Plan holding and churning, we may use a contingency table (Table 3.1), since both variables are categorical.

Note that the counts in the first column add up to the total number of nonselectors of the International Plan from Figure 3.4: $2664 + 346 = 3010$. Similarly for the second column. The first row in Table 3.1 shows the counts of those who did not churn, while the second row shows the counts of those that did churn.

The total column contains the *marginal distribution* for churn, that is, the frequency distribution for this variable alone. Similarly the total row represents the marginal distribution for International Plan. Note that the marginal distribution for International Plan concurs with the counts in Figure 3.5.

We may enhance Table 3.1 with percentages, depending on our question of interest. For example, Table 3.2 adds *column percentages*, which indicate, for each cell, the percentage of the column total.

We calculate the column percentages whenever we are interested in comparing the percentages of the row variable for each value of the column variable. For example, here we are interested in comparing the proportions of churners (row variable) for those who belong or do not belong to the International Plan (column variable). Note that $137/(137 + 186) = 42.4\%$ of the International Plan holders churned, as compared to only $346/(346 + 2664) = 11.5\%$ of those without the International Plan. Customers selecting the International Plan are more than three times as likely to leave the company’s service and those without the plan. Thus, we have now quantified the relationship that we uncovered graphically earlier.

The graphical counterpart of the contingency table is the *clustered bar chart*. Figure 3.6 shows a Minitab bar chart of churn, *clustered by* International Plan. The

TABLE 3.1 Contingency table of International Plan with Churn

		International Plan		
		No	Yes	Total
Churn	False	2664	186	2850
	True	346	137	483
	Total	3010	323	3333

TABLE 3.2 Contingency table with column percentages

		International Plan		
		No	Yes	Total
Churn	False	Count 2664	Count 186	Count 2850
		Col% 88.5%	Col% 57.6%	Col% 85.5%
	True	Count 346	Count 137	Count 483
		Col% 11.5%	Col% 42.4%	Col% 14.5%
	Total	3010	323	3333

first set of two bars represents those who do not belong to the plan, and is associated with the “No” column in Table 3.2. The second set of two bars represents those who do belong to the International Plan, and is associated with the “Yes” column in Table 3.2. Clearly, the proportion of churners is greater among those belonging to the plan.

Another useful graphic for comparing two categorical variables is the *comparative pie chart*. Figure 3.7 shows a comparative pie chart of churn, for those who do not (“no”) and those who do (“yes”) belong to the International Plan. The clustered bar chart is usually preferred, because it conveys counts as well as proportions, while the comparative pie chart conveys only proportions.

Contrast Table 3.2 with Table 3.3, the contingency table with *row percentages*, which indicate, for each cell, the percentage of the row total. We calculate the row percentages whenever we are interested in comparing the percentages of the column



Figure 3.6 The clustered bar chart is the graphical counterpart of the contingency table.

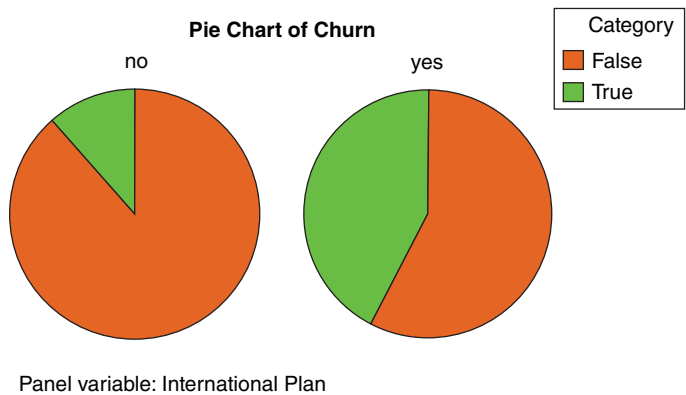


Figure 3.7 Comparative pie chart associated with Table 3.2.

variable for each value of the row variable. Table 3.3 indicates, for example, that 28.4% of churners belong to the International Plan, compared to 6.5% of non-churners.

Figure 3.8 contains the bar chart of International Plan, clustered by Churn, and represents the graphical counterpart of the contingency table with row percentages in Table 3.3. The first set of bars represents non-churners, and is associated with the “False” row in Table 3.3. The second set of bars represents churners, and is associated with the “True” row in Table 3.3. Clearly, the proportion of International Plan holders is greater among the churners. Similarly for Figure 3.9, which shows the comparative bar chart of International Plan holders, by whether or not they have churned (“True” or “False”).

To summarize, this EDA on the International Plan has indicated that

- 1. Perhaps we should investigate what it is about our International Plan that is inducing our customers to leave!
- 2. We should expect that, whatever data mining algorithms we use to predict churn, the model will probably include whether or not the customer selected the International Plan.

TABLE 3.3 Contingency table with row percentages

		International Plan		
		No	Yes	Total
Churn	False	Count 2664	Count 186	2850
		Row % 93.5%	Row % 6.5%	
	True	Count 346	Count 137	483
		Row % 71.6%	Row % 28.4%	
	Total	Count 3010	Count 323	3333
		Row % 90.3%	Row % 9.7%	

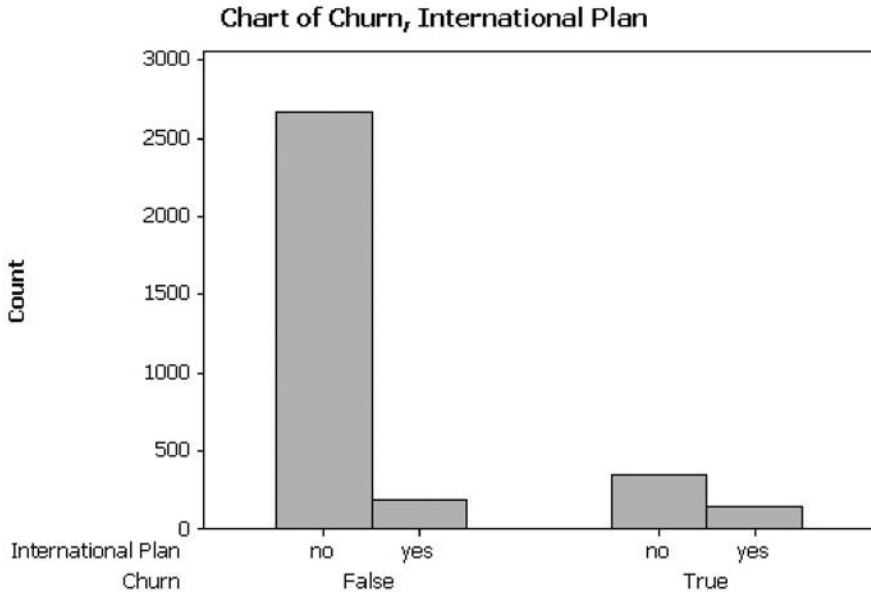
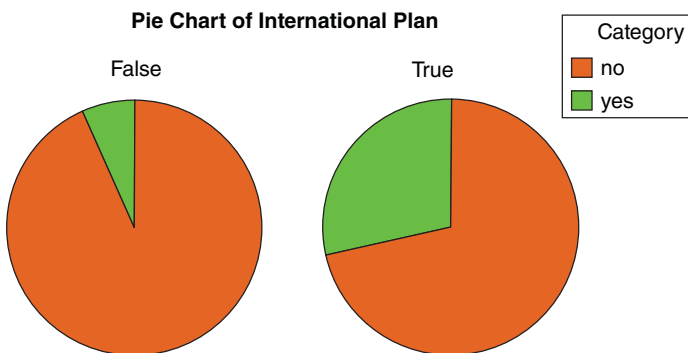


Figure 3.8 Clustered bar chart associated with Table 3.3.

Let us now turn to the Voice Mail Plan. Figure 3.10 shows, using a bar graph with equalized lengths, that those who do not have the Voice Mail Plan are more likely to churn than those who do have the Plan. (The numbers in the graph indicate proportions and counts of those who do and do not have the Voice Mail Plan, without reference to churning.)

Again, we may quantify this finding by using a contingency table. Because we are interested in comparing the percentages of the row variable (Churn) for each value of the column variable (Voice Mail Plan), we choose a contingency table with column percentages, shown in Table 3.4.



Panel variable: Churn

Figure 3.9 Comparative pie chart associated with Table 3.3.

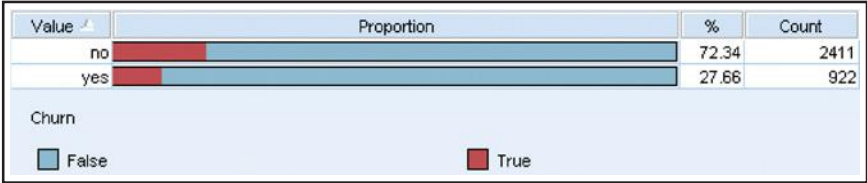


Figure 3.10 Those without the Voice Mail Plan are more likely to churn.

The marginal distribution for Voice Mail Plan (row total) indicates that $842 + 80 = 922$ customers have the Voice Mail Plan, while $2008 + 403 = 2411$ do not. We then find that $403/2411 = 16.7\%$ of those without the Voice Mail Plan are churners, as compared to $80/922 = 8.7\%$ of customers who do have the Voice Mail Plan. Thus, customers without the Voice Mail Plan are nearly twice as likely to churn as customers with the plan.

To summarize, this EDA on the Voice Mail Plan has indicated that

1. Perhaps we should enhance our Voice Mail Plan still further, or make it easier for customers to join it, as an instrument for increasing customer loyalty.
2. We should expect that, whatever data mining algorithms we use to predict churn, the model will probably include whether or not the customer selected the Voice Mail Plan. Our confidence in this expectation is perhaps not quite as high as for the International Plan.

We may also explore the *two-way interactions* among categorical variables with respect to churn. For example, Figure 3.11 shows a multilayer clustered bar chart of churn, clustered by *both* International Plan and Voice Mail Plan.

The statistics associated with Figure 3.11 are shown in Figure 3.12. Note that there are many more customers who have neither plan ($1878 + 302 = 2180$) than have the International Plan only ($130 + 101 = 231$). More importantly, among customers with no Voice Mail Plan, the proportion of churners is greater for those who do have an International Plan ($101/231 = 44\%$) than for those who do not ($302/2180 = 14\%$). There are many more customers who have the Voice Mail Plan only ($786 + 44 = 830$) than have both plans ($56 + 36 = 92$). Again, however, among customers with the Voice Mail Plan, the proportion of churners is much greater for those who also

TABLE 3.4 Contingency table with column percentages for the Voice Mail Plan

		Voice Mail Plan		
		No	Yes	Total
Churn	False	Count 2008	Count 842	Count 2850
		Col% 83.3%	Col% 91.3%	Col% 85.5%
	True	Count 403	Count 80	Count 483
		Col% 16.7%	Col% 8.7%	Col% 14.5%
	Total	2411	922	3333

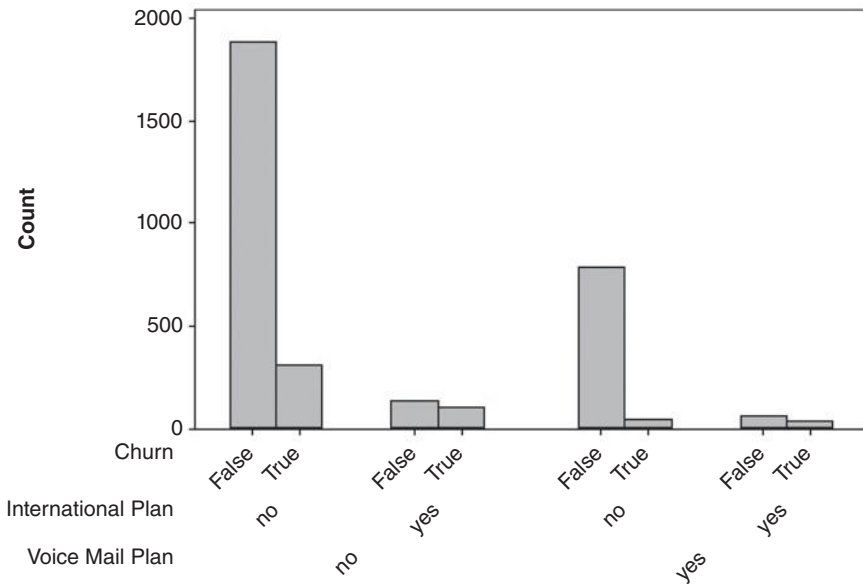


Figure 3.11 Multilayer clustered bar chart.

Results for Voice Mail Plan = no

Rows: Churn Columns: International Plan

	no	yes	All
False	1878	130	2008
True	302	101	403
All	2180	231	2411

Results for Voice Mail Plan = yes

Rows: Churn Columns: International Plan

	no	yes	All
False	786	56	842
True	44	36	80
All	830	92	922

Figure 3.12 Statistics for multilayer clustered bar chart.

select the International Plan ($36/92 = 39\%$) than for those who do not ($44/830 = 5\%$). Note also that there is no interaction among the categorical variables. That is, International Plan holders have greater churn regardless of whether they are Voice Mail Plan adopters or not.

Finally, Figure 3.13 shows a *directed web graph* of the relationships between International Plan holders, VoiceMail Plan holders, and churners. Note that three lines lead to the Churn = False node, which is good. However, note that one faint line leads to the Churn = True node, that of the International Plan holders, indicating that a greater proportion of International Plan holders choose to churn. This supports our earlier findings.

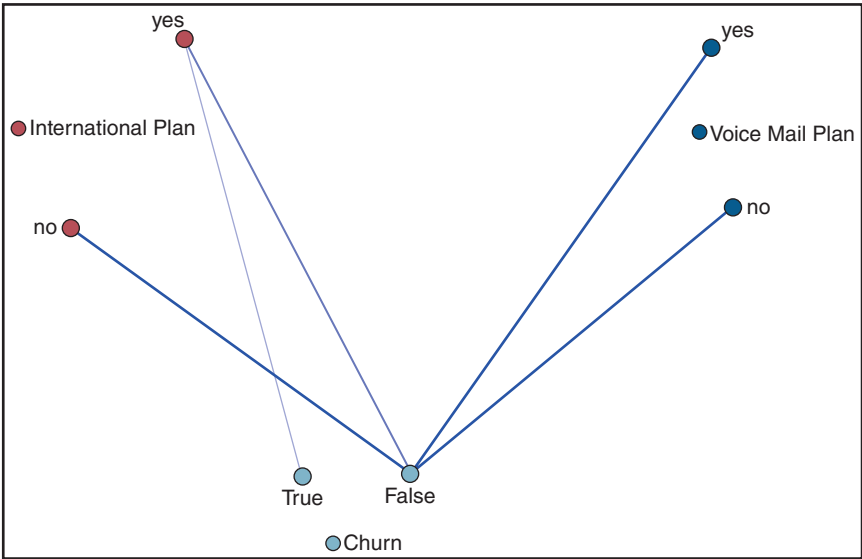


Figure 3.13 Directed web graph supports earlier findings.

3.4 EXPLORING NUMERIC VARIABLES

Next, we turn to an exploration of the numeric predictive variables. Refer back to Figure 3.2 and for histograms and summary statistics of the various predictors. Note that many fields show evidence of symmetry, such as *account length* and *all the minutes*, *charge*, and *call* fields. Fields not showing evidence of symmetry include *voice mail messages* and *customer service calls*. The median for *voice mail messages* is zero, indicating that at least half of all customers had no voice mail messages. This results of course from fewer than half of the customers selecting the Voice Mail Plan, as we saw above. The mean of *customer service calls* (1.563) is greater than the median (1.0), indicating some right-skewness, as also indicated by the maximum number of customer service calls being nine.

Unfortunately, the usual type of histograms (such as those in Figure 3.2) do not help us determine whether the predictor variables are associated with the target variable. To explore whether a predictor is useful for predicting the target variable,

we should use an *overlay histogram*, which is a histogram where the rectangles are colored according to the values of the target variable. For example, Figure 3.14 shows a histogram of the predictor variable *customer service calls*, with no overlay. We can see that the distribution is right skewed with a mode of one call, but we have no information on whether this variable is useful for predicting churn. Next, Figure 3.15 shows a histogram of customer service calls, with an overlay of the target variable *churn*.

Figure 3.15 hints that the proportion of churn may be greater for higher numbers of customer service calls, but it is difficult to discern this result unequivocally. We therefore turn to the “normalized” histogram, where every rectangle has the same height and width, as shown in Figure 3.16. Note that the *proportions* of churners versus non-churners in Figure 3.16 is exactly the same as in Figure 3.15; it is just that “stretching out” the rectangles that have low counts enables better definition and contrast.

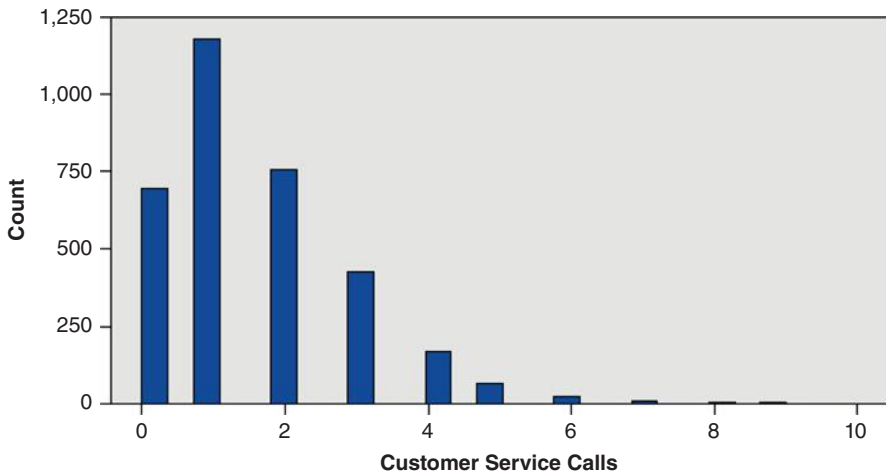


Figure 3.14 Histogram of customer service calls with no overlay.

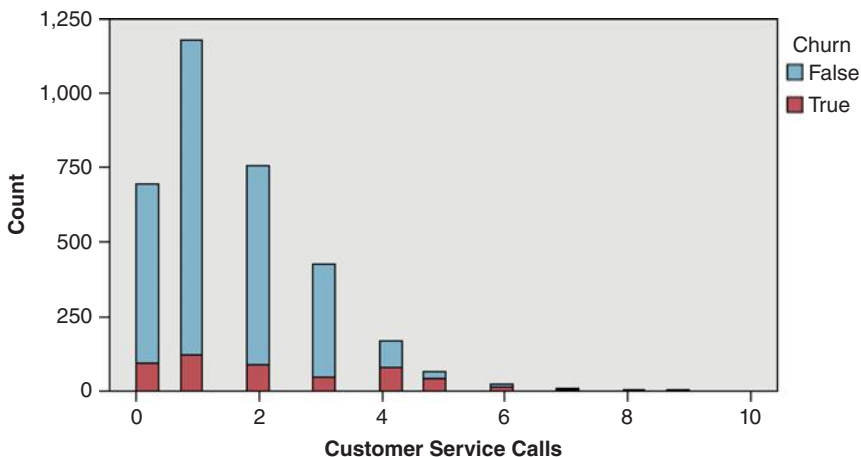


Figure 3.15 Histogram of customer service calls, with churn overlay.

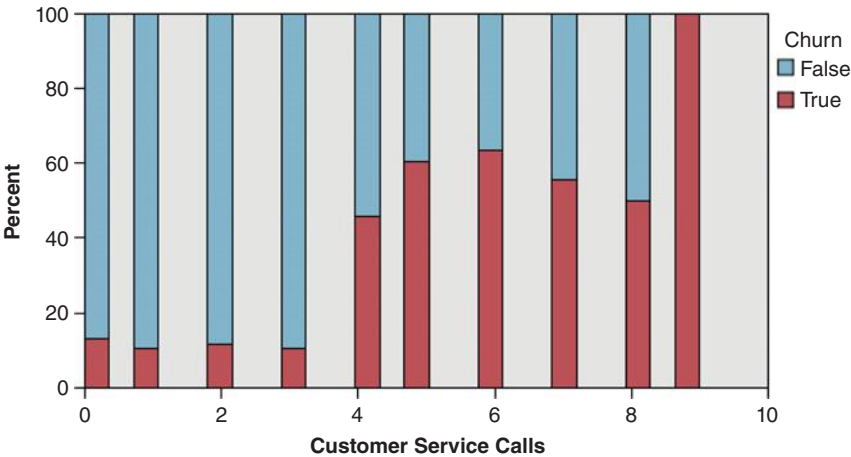


Figure 3.16 “Normalized” histogram of customer service calls, with churn overlay.

The pattern now becomes crystal clear. Customers who have called customer service three times or less have a markedly lower churn rate (darker part of the rectangle) than customers who have called customer service four or more times.

This EDA on the customer service calls has indicated that

1. We should carefully track the number of customer service calls made by each customer. By the third call, specialized incentives should be offered to retain customer loyalty, because, by the fourth call, the probability of churn increases greatly.
2. We should expect that, whatever data mining algorithms we use to predict churn, the model will probably include the number of customer service calls made by the customer.

Important note: Normalized histograms are useful for teasing out the relationship between a numerical predictor and the target. However, data analysts should always provide the companion nonnormalized histogram along with the normalized histogram, because the normalized histogram does not provide any information on the frequency distribution of the variable. For example, Figure 3.16 indicates that the churn rate for customers logging nine service calls is 100%; but Figure 3.15 shows that there are only two customers with this number of calls.

Let us now turn to the remaining numerical predictors. The normalized histogram of *Day Minutes* in Figure 3.17b shows that high day-users tend to churn at a higher rate. Therefore

1. We should carefully track the number of day minutes used by each customer. As the number of day minutes passes 200, we should consider special incentives.
2. We should investigate why heavy day-users are tempted to leave.
3. We should expect that our eventual data mining model will include *day minutes* as a predictor of churn.

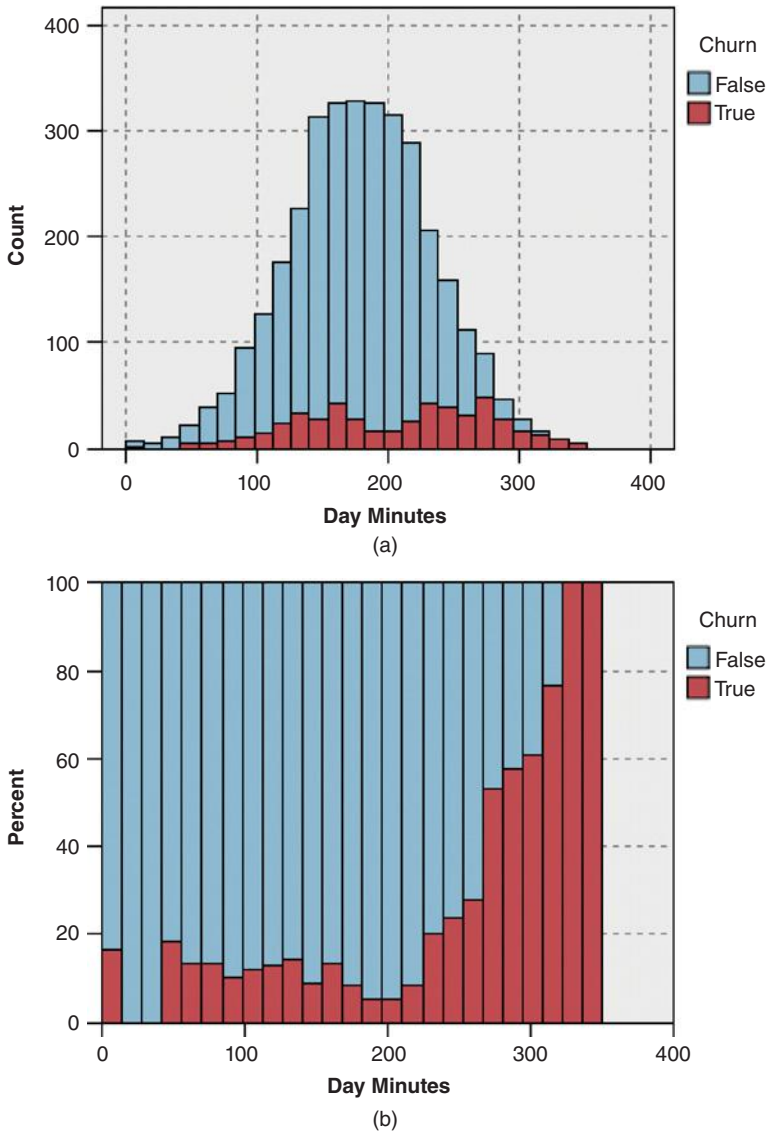


Figure 3.17 (a) Nonnormalized histogram of day minutes; (b) normalized histogram of day minutes.

Figure 3.18b shows a slight tendency for customers with higher *evening minutes* to churn. Based solely on the graphical evidence, however, we cannot conclude beyond a reasonable doubt that such an effect exists. Therefore, we shall hold off on formulating policy recommendations on evening cell-phone use until our data mining models offer firmer evidence that the putative effect is in fact present.

Figures 3.19b indicates that there is no obvious association between churn and *night minutes*, since the pattern is relatively flat. In fact, EDA would indicate no

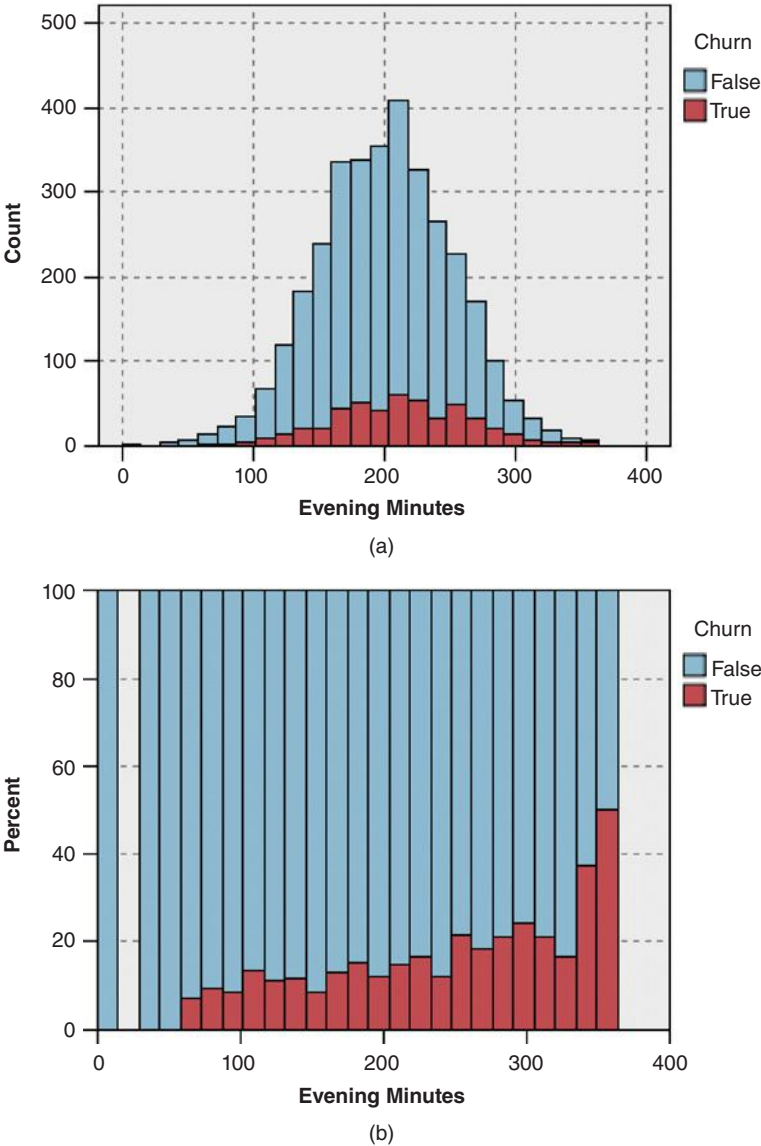


Figure 3.18 (a) Nonnormalized histogram of evening minutes; (b) normalized histogram of evening minutes.

obvious association with the target for any of the remaining numeric variables in the data set (except one), though showing this is left as an exercise.

Note: The lack of obvious association at the EDA stage between a predictor and a target variable is not sufficient reason to omit that predictor from the model. For example, based on the lack of evident association between churn and night minutes, we will not necessarily expect the data mining models to uncover valuable predictive

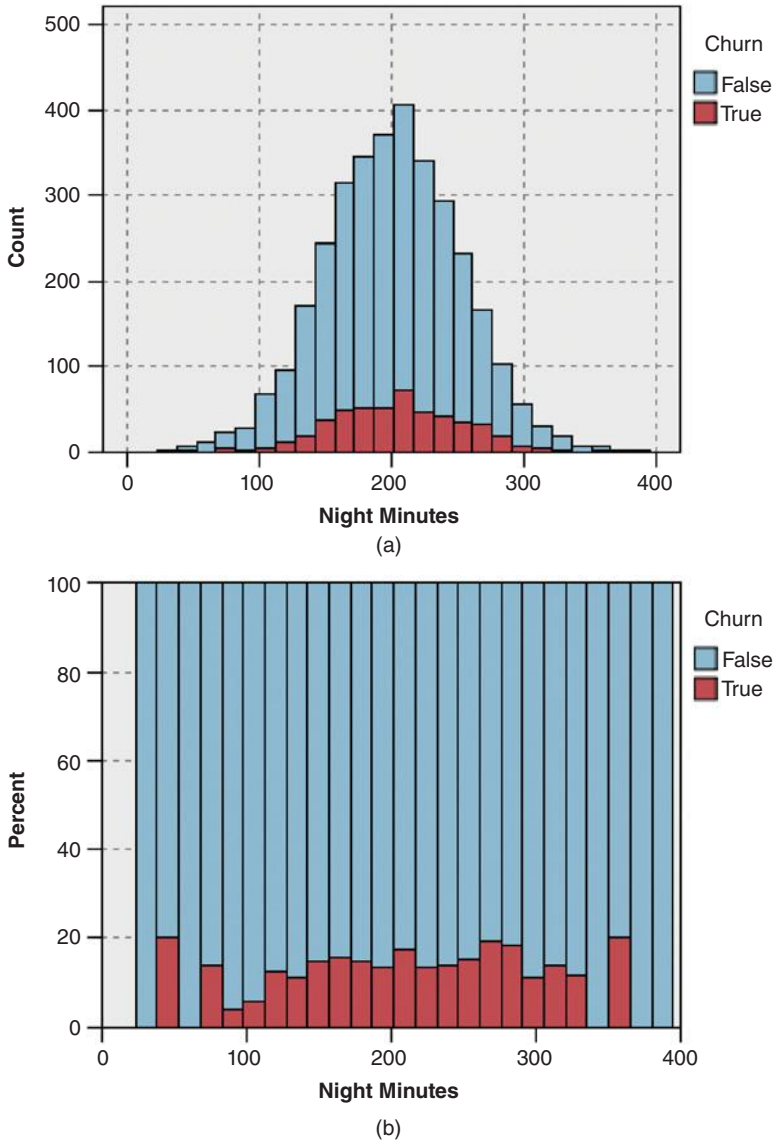


Figure 3.19 (a) Nonnormalized histogram of night minutes; (b) normalized histogram of night minutes.

information using this predictor. However, we should nevertheless retain the predictor as an input variable for the data mining models, because actionable associations may still exist for identifiable subsets of the records, and they may be involved in higher-dimension associations and interactions. In any case, unless there is a good reason for eliminating the variable prior to modeling, then we should probably allow the modeling process to identify which variables are predictive and which are not.

For example, Figures 3.20a and 3.20b, of the predictor *International Calls* with *churn* overlay, do not indicate strong graphical evidence of the predictive importance of *International Calls*. However, a *t*-test (see Chapter 4) for the difference in mean number of international calls for churners and non-churners is statistically significant (Figure 3.21, p -value = 0.003; p -values larger than, say, 0.10 are not considered significant; see Chapter 4), meaning that this variable is indeed useful for predicting churn: Churners tend to place a lower mean number of international calls. Thus, had we omitted *International Calls* from the analysis based on the seeming lack of graphical evidence, we would have committed a mistake, and our predictive model would not perform as well.

A hypothesis test, such as this *t*-test, represents statistical inference and model building, and as such lies beyond the scope of exploratory data analysis. We mention it here merely to underscore the importance of not omitting predictors merely because their relationship with the target is nonobvious using EDA.

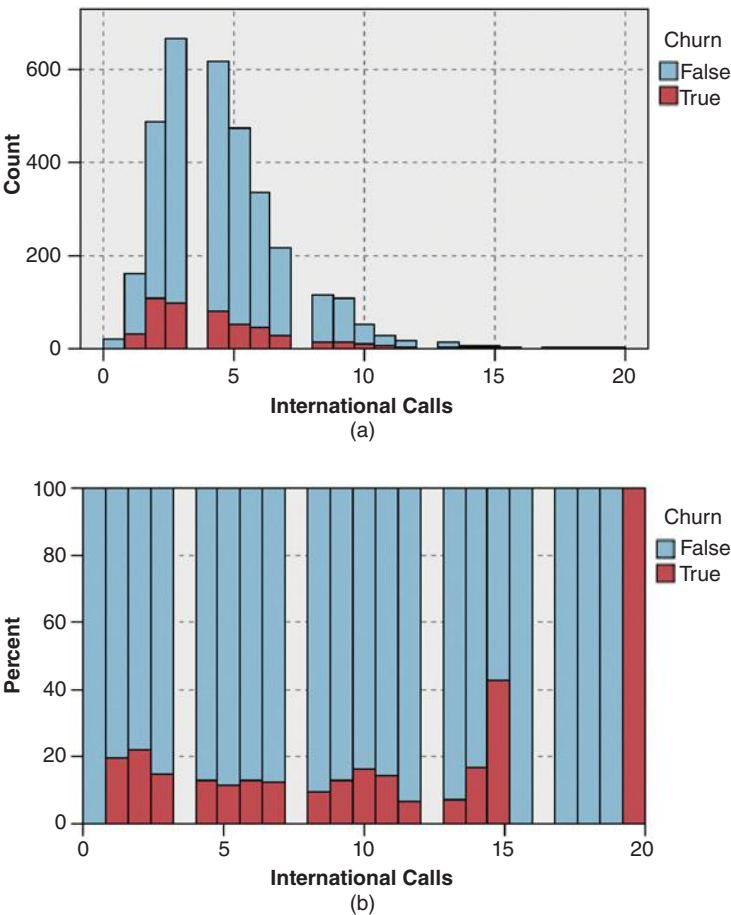


Figure 3.20 (a) Nonnormalized histogram of *International Calls*; (b) normalized histogram of *International Calls*.

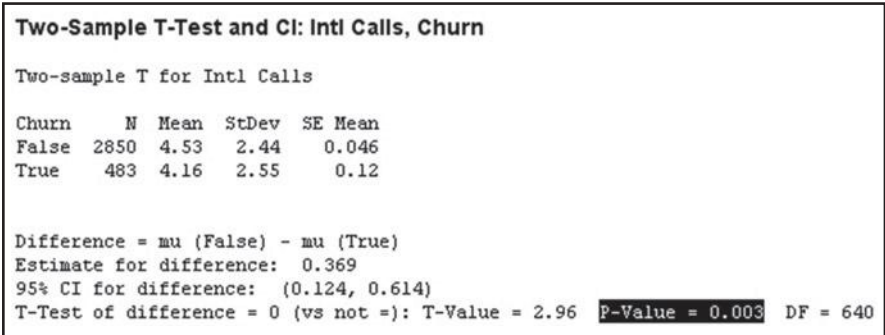


Figure 3.21 t -test is significant for difference in mean international calls for churners and non-churners.

3.5 EXPLORING MULTIVARIATE RELATIONSHIPS

We next turn to an examination of possible multivariate associations of numeric variables with churn, using scatter plots. Multivariate graphics can uncover new interaction effects which our univariate exploration missed.

Figure 3.22 shows a scatter plot of *day minutes* versus *evening minutes*, with churners indicated by the darker circles. Note the straight-line partitioning off the

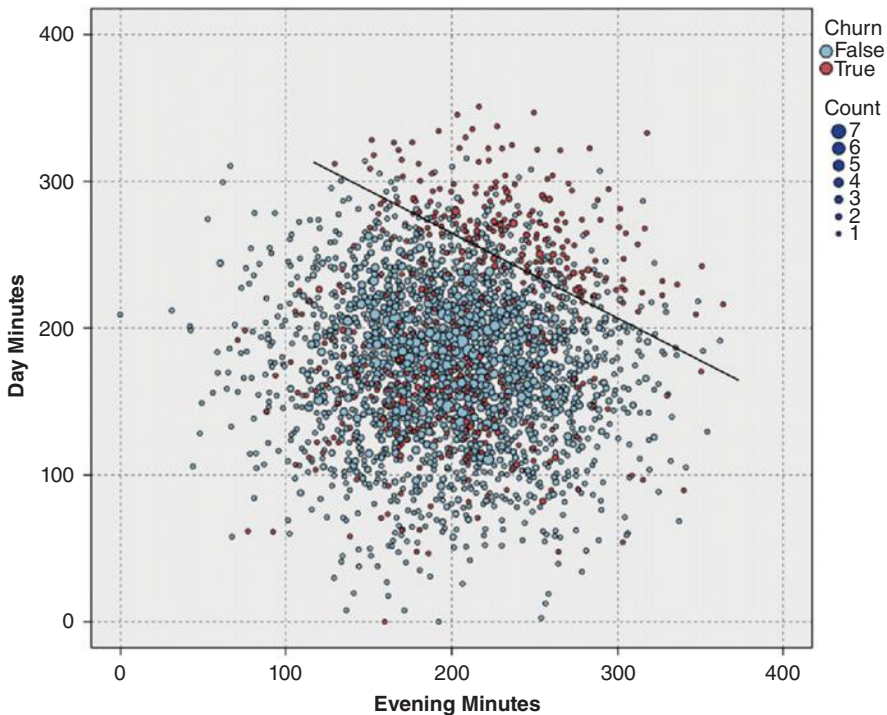


Figure 3.22 Customers with both high day minutes and high evening minutes are at greater risk of churning.

upper right section of the graph. Records above this diagonal line, representing customers with both high day minutes and high evening minutes, appear to have a higher proportion of churners than records below the line. The univariate evidence for a high churn rate for high evening minutes was not conclusive (Figure 3.18b), so it is nice to have a multivariate graph that indicates that supports the association, at least for customers with high day minutes.

Figure 3.23 shows a scatter plot of *customer service calls* versus *day minutes*. Churners and non-churners are indicated with large and small circles, respectively. Consider the records inside the rectangle partition shown in the scatter plot, which indicates a high churn area in the upper left section of the graph. These records represent customers who have a combination of a high number of customer service calls and a low number of day minutes used. Note that this group of customers could not have been identified had we restricted ourselves to univariate exploration (exploring variable by single variable). This is because of the *interaction* between the variables.

In general, customers with higher numbers of customer service calls tend to churn at a higher rate, as we learned earlier in the univariate analysis. However, Figure 3.23 shows that, of these customers with high numbers of customer service calls, those who also have high day minutes are somewhat “protected” from this high churn rate. The customers in the upper right of the scatter plot exhibit a lower churn rate than those in the upper left. But how do we quantify these graphical findings?

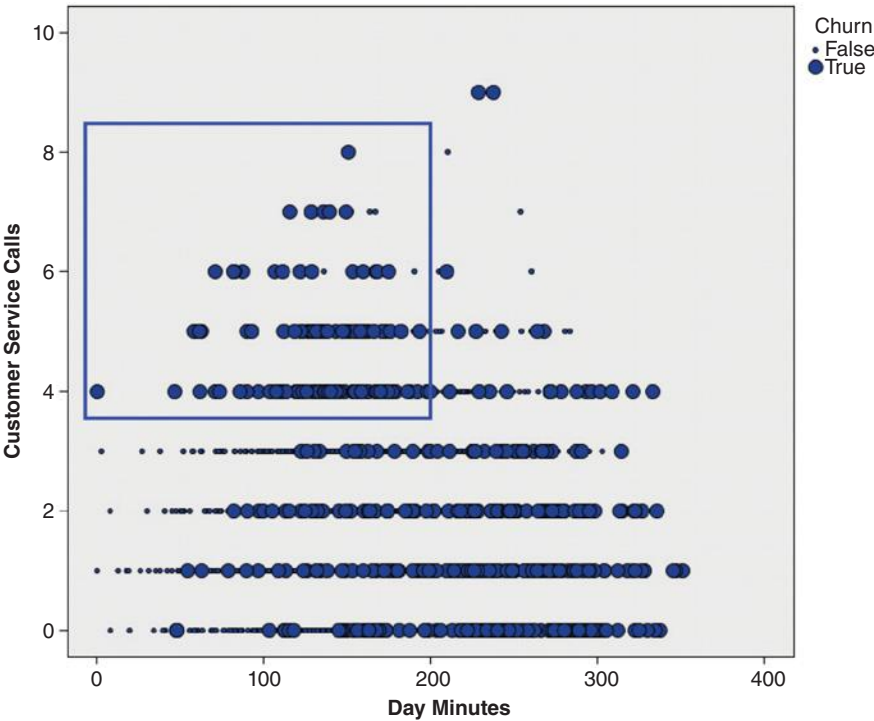


Figure 3.23 There is an interaction effect between *customer service calls* and *day minutes*, with respect to churn.

3.6 SELECTING INTERESTING SUBSETS OF THE DATA FOR FURTHER INVESTIGATION

Graphical EDA can uncover subsets of records that call for further investigation, as the rectangle in Figure 3.23 illustrates. Let us examine the records in the rectangle more closely. *IBM/SPSS Modeler* allows the user to click and drag a box around data points of interest, and select them for further investigation. Here we select the records within the rectangular box in the upper left. Figure 3.24 shows that about 65% (115 of 177) of the selected records are churners. That is, those with high customer service calls and low day minutes have a 65% probability of churning. Compare this to the records with high customer service calls and high day minutes (essentially the data points to the right of the rectangle). Figure 3.25 shows that only about 26% of customers with high customer service calls and high day minutes are churners. Thus, it is recommended that we red-flag customers with low day minutes who have a high number of customer service calls, as they are at much higher risk of leaving the company’s service than customers with the same number of customer service calls, but higher day minutes.

To summarize the strategy we implemented here.

- 1. Generate multivariate graphical EDA, such as scatter plots with a flag overlay.
- 2. Use these plots to uncover subsets of interesting records.
- 3. Quantify the differences by analyzing the subsets of records.

Value	Proportion	%	Count
False	<div></div>	35.03	62
True	<div></div>	64.97	115

Figure 3.24 Very high proportion of churners for high customer service calls and low day minutes.

Value	Proportion	%	Count
False	<div></div>	74.44	67
True	<div></div>	25.56	23

Figure 3.25 Much lower proportion of churners for high customer service calls and high day minutes.

3.7 USING EDA TO UNCOVER ANOMALOUS FIELDS

Exploratory data analysis will sometimes uncover strange or anomalous records or fields which the earlier data cleaning phase may have missed. Consider, for example, the *area code* field in the present data set. Though the area codes contain numerals, they can also be used as categorical variables, since they can classify customers according to geographic location. We are intrigued by the fact that the area code field

contains only three different values for all the records, *408*, *415*, and *510* (which all happen to be California area codes), as shown by Figure 3.26.

Now, this would not be anomalous if the records indicated that the customers all lived in California. However, as shown in the contingency table in Figure 3.27 (shown only up to Georgia, to save space), the three area codes seem to be distributed more or less evenly across all the states and the District of Columbia. Also, the chi-square test (see Chapter 5) has a *p*-value of 0.608, supporting the suspicion that the area codes are distributed randomly across all the states. Now, it is possible that domain experts might be able to explain this type of behavior, but it is also possible that the field just contains bad data.

We should therefore be wary of this *area code* field, and should not include it as input to the data mining models in the next phase. Further, the *state* field may be in error as well. Either way, further communication with someone familiar with the data history, or a domain expert, is called for before inclusion of these variables in the data mining models.

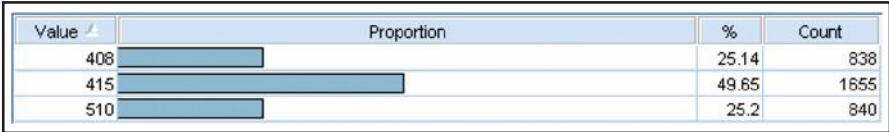


Figure 3.26 Only three area codes for all records.

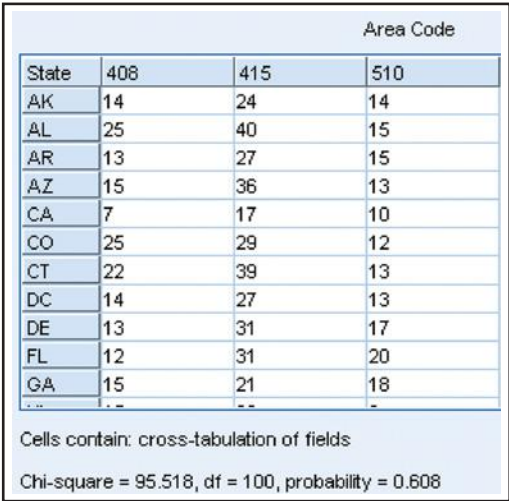


Figure 3.27 Anomaly: three area codes distributed randomly across all 50 states.

3.8 BINNING BASED ON PREDICTIVE VALUE

Chapter 2 discussed four methods for binning numerical variables. Here, we provide two examples of the fourth method: Binning based on predictive value. Recall

Figure 3.16, where we saw that customers with less than four calls to customer service had a lower churn rate than customers who had four or more calls to customer service. We may therefore decide to bin the *customer service calls* variable into two classes, *low* (fewer than four) and *high* (four or more). Table 3.5 shows that the churn rate for customers with a low number of calls to customer service is 11.3%, while the churn rate for customers with a high number of calls to customer service is 51.7%, more than four times higher.

This binning of customer service calls created a flag variable with two values, high and low. Our next example of binning creates an ordinal categorical variable with three values, low, medium, and high. Recall that we are trying to determine whether there is a relationship between *evening minutes* and *churn*. Figure 3.18b hinted at a relationship, but inconclusively. Can we use binning to help tease out a signal from this noise? We reproduce Figure 3.18b here as Figure 3.28, somewhat enlarged, and with the boundaries between the bins indicated.

Binning is an art, requiring judgment. *Where can I insert boundaries between the bins that will maximize the difference in churn proportions?* The first boundary

TABLE 3.5 Binning customer service calls shows difference in churn rates

		CustServPlan_Bin	
		Low	High
Churn	False	Count 2721 Col% 88.7%	Count 129 Col% 48.3%
	True	Count 345 Col% 11.3%	Count 138 Col% 51.7%

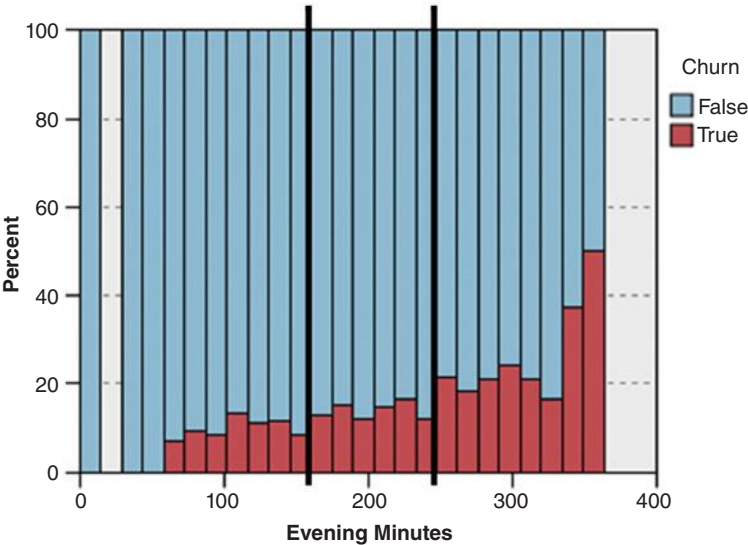


Figure 3.28 Binning *evening minutes* helps to tease out a signal from the noise.

is inserted at *evening minutes* = 160, since the group of rectangles to the right of this boundary seem higher than the group of rectangles to the left. And the second boundary is inserted at *evening minutes* = 240 for the same reason. (Analysts may fine tune these boundaries for maximum contrast, but for now these boundary values will do just fine; remember that we need to explain our results to the client, and that nice round numbers are more easily explained.) These boundaries thus define three bins, or categories, shown in Table 3.6.

Did the binning manage to tease out a signal? We can answer this by constructing a contingency table of *EveningMinutes_Bin* with *Churn*, shown in Table 3.7.

About half of the customers have medium amounts of evening minutes (1626/3333 = 48.8%), with about one-quarter each having low and high evening minutes. Recall that the *baseline churn rate* for all customers is 14.49% (Figure 3.3). The *medium* group comes in very close to this baseline rate, 14.1%. However, the *high* evening minutes group has nearly double the churn proportion compared to the *low* evening minutes group, 19.5% to 10%. The chi-square test (Chapter 4) is significant, meaning that these results are most likely real and not due to chance alone. In other words, we have succeeded in teasing out a signal from the *evening minutes* versus *churn* relationship.

TABLE 3.6 Bin values for *Evening Minutes*

Bin for Categorical Variable <i>EveningMinutes_Bin</i>	Values of Numerical Variable <i>Evening Minutes</i>
Low	<i>Evening Minutes</i> ≤ 160
Medium	160 < <i>Evening Minutes</i> ≤ 240
High	<i>Evening Minutes</i> > 240

TABLE 3.7 We have uncovered significant differences in churn rates among the three categories

		EveningMinutes_Bin		
		Low	Medium	High
Churn	False	Count 618	Count 1626	Count 606
		Col% 90.0%	Col% 85.9%	Col% 80.5%
	True	Count 69	Count 138	Count 138
		Col% 10.0%	Col% 14.1%	Col% 19.5%

3.9 DERIVING NEW VARIABLES: FLAG VARIABLES

Strictly speaking, deriving new variables is a data preparation activity. However, we cover it here in the EDA chapter to illustrate how the usefulness of the new derived variables in predicting the target variable may be assessed. We begin with an example

of a derived variable which is not particularly useful. Figure 3.2 shows a spike in the distribution of the variable *voice mail messages*, which makes its analysis problematic. We therefore derive a flag variable (see Chapter 2), *VoiceMailMessages_Flag*, to address this problem, as follows:

```
If Voice Mail Messages > 0 then VoiceMailMessages_Flag = 1;
otherwise VoiceMailMessages_Flag = 0.
```

The resulting contingency table is shown in Table 3.8. Compare the results with those from Table 3.4, the contingency table for the Voice Mail Plan. The results are exactly the same, which is not surprising, since those without the plan can have no voice mail messages. Thus, since *VoiceMailMessages_Flag* has identical values as the flag variable *Voice Mail Plan*, it is not deemed to be a useful derived variable.

Recall Figure 3.22 (reproduced here as Figure 3.29), showing a scatter plot of *day minutes* versus *evening minutes*, with a straight line separating a group in the upper right (with both high day minutes and high evening minutes) that apparently churns at a greater rate. It would be nice to quantify this claim. We do so by selecting the records in the upper right, and compare their churn rate to that of the other records. One way to do this in IBM/SPSS Modeler is to draw an oval around the desired records, which the software then selects (not shown). However, this method is *ad hoc*, and not portable to a different data set (say the validation set). A better idea is to

1. Estimate the equation of the straight line and
2. Use the equation to separate the records, via a flag variable.

This method is portable to a validation set or other related data set.

We estimate the equation of the line in Figure 3.22 to be

$$\hat{y} = 400 - 0.6x$$

That is, for each customer, the estimated day minutes equals 400 minutes minus 0.6 times the evening minutes. We may then create a flag variable *HighDayEveMins_Flag* as follows:

```
If Day Minutes > 400 - 0.6 Evening Minutes then
HighDayEveMins_Flag = 1; otherwise HighDayEveMins_Flag = 0.
```

Then each data point above the line will have *HighDayEveMins_Flag* = 1 while the data points below the line will have *HighDayEveMins_Flag* = 0. The resulting contingency table (Table 3.9) shows the highest churn proportion of any variable

TABLE 3.8 Contingency table for *VoiceMailMessages_Flag*

		VoiceMailMessages_Flag	
		0	1
Churn	False	Count 2008	Count 842
		Col% 83.3%	Col% 91.3%
	True	Count 403	Count 80
		Col% 16.7%	Col% 8.7%

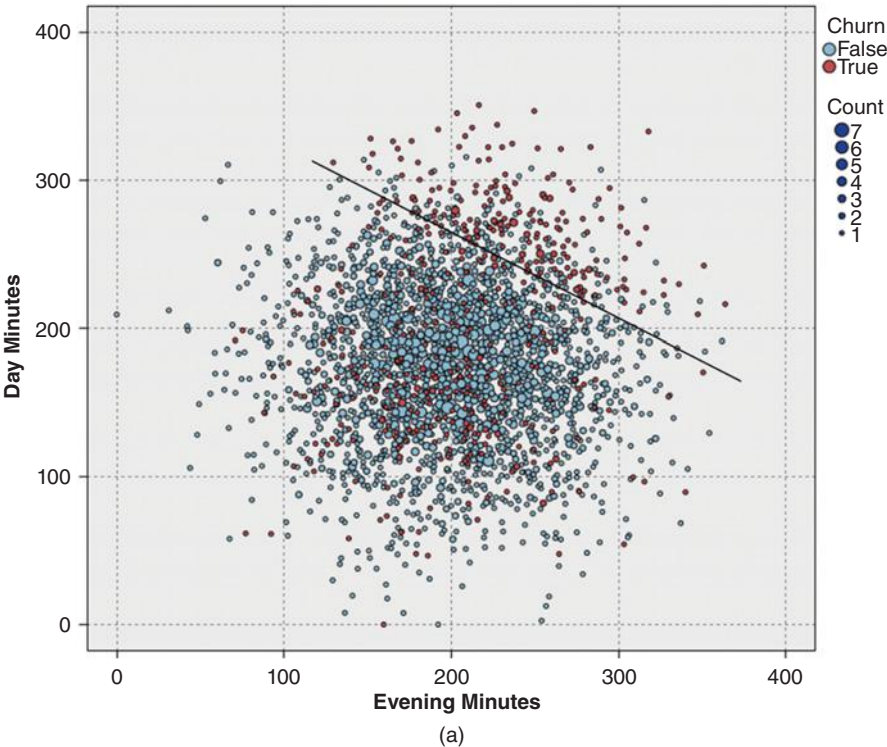


Figure 3.29 Use the equation of the line to separate the records, via a flag variable.

TABLE 3.9 Contingency table for *HighDayEveMins_Flag*

		HighDayEveMins_Flag	
		0	1
Churn	False	Count 2792 Col% 89.0%	Count 58 Col% 29.6%
	True	Count 345 Col% 11.0%	Count 138 Col% 70.4%

we have studied thus far, 70.4% versus 11%, a more than sixfold difference. On the other hand, this 70.4% churn rate is restricted to a subset of fewer than 200 records, fortunately for the company.

**A NOTE ABOUT CRISP-DM FOR DATA MINERS:
BE STRUCTURED BUT FLEXIBLE**

For *EveningMinutes_Bin*, we referred to the chi-square significance test (Chapter 5), which really belongs to the modeling phase of data analysis. Also, our derived variable really

belongs to the data preparation phase. These examples illustrate the flexibility of the CRISP-DM standard practice (or indeed any well-structured standard practice) of data mining. The assorted phases are interdependent, and should not be viewed as isolated from each other. For example, deriving variables is a data preparation activity, but derived variables need to be explored using EDA and (sometimes) significance tests. The data miner needs to be as flexible as CRISP-DM.

However, some data analysts fall victim to the opposite problem, interminably iterating back and forth between data preparation and EDA, getting lost in the details, and never advancing toward the research objectives. When this happens, CRISP-DM can serve as a useful road map, a structure to keep the data miner organized and moving toward the fulfillment of the research goals.

3.10 DERIVING NEW VARIABLES: NUMERICAL VARIABLES

Suppose we would like to derive a new numerical variable which combines *Customer Service Calls* and *International Calls*, and whose values will be the mean of the two fields. Now, since *International Calls* has a larger mean and standard deviation than *Customer Service Calls*, it would be unwise to take the mean of the raw field values, since *International Calls* would thereby be more heavily weighted. Instead, when combining numerical variables, we first need to standardize. The new derived variable therefore takes the form:

$$CSCInternational_Z = \frac{(CSC_Z + International_Z)}{2}$$

where *CSC_Z* represents the z-score standardization of *Customer Service Calls* and *International_Z* represents the z-score standardization of *International Calls*. The resulting normalized histogram of *CSCInternational_Z* indicates that it will be useful for predicting churn, as shown in Figure 3.30b.

3.11 USING EDA TO INVESTIGATE CORRELATED PREDICTOR VARIABLES

Two variables x and y are linearly correlated if an increase in x is associated with either an increase in y or a decrease in y . The *correlation coefficient* r quantifies the strength and direction of the linear relationship between x and y . The threshold for significance of the correlation coefficient r depends on the sample size, but in data mining, where there are a large number of records (over 1000), even small values of r , such as $-0.1 \leq r \leq 0.1$ may be statistically significant.

One should take care to avoid feeding correlated variables to one's data mining and statistical models. At best, using correlated variables will overemphasize one data component; at worst, using correlated variables will cause the model to become unstable and deliver unreliable results. However, *just because two variables are correlated does not mean that we should omit one of them*. Instead, while in the EDA stage, we should apply the following strategy:

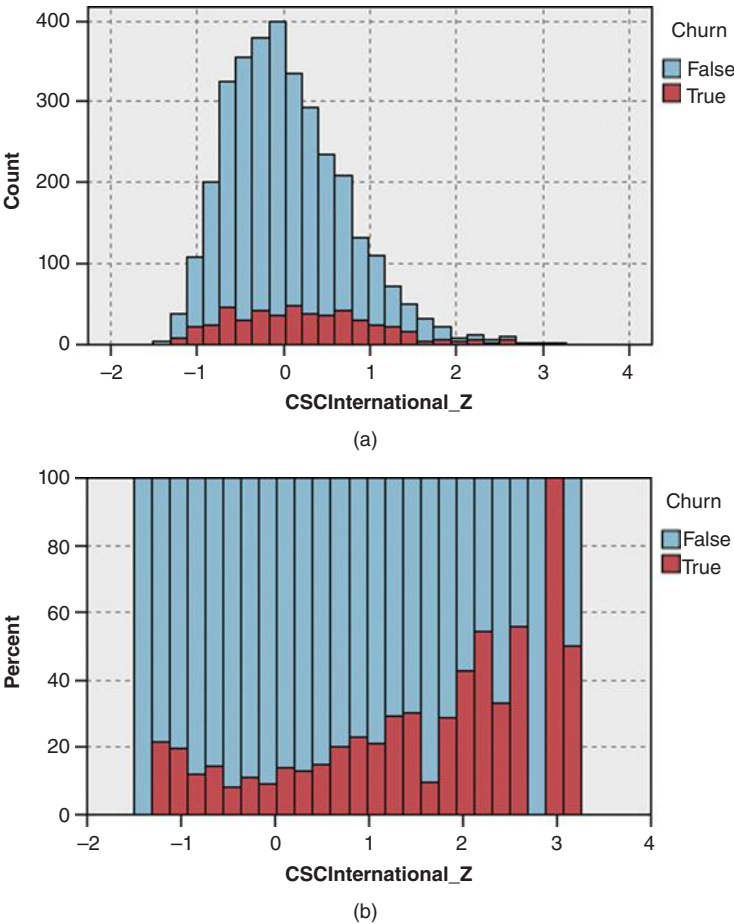


Figure 3.30 (a) Nonnormalized histogram of *CSCInternational_Z*; (b) normalized histogram of *CSCInternational_Z*.

STRATEGY FOR HANDLING CORRELATED PREDICTOR VARIABLES AT THE EDA STAGE

1. Identify any variables that are perfectly correlated (i.e., $r = 1.0$ or $r = -1.0$). Do not retain both variables in the model, but rather omit one.
2. Identify groups of variables that are correlated with each other. Then, later, during the modeling phase, apply *dimension reduction methods*, such as *principal components analysis*¹, to these variables.

¹For more on dimension reductions and principal component analysis, see *Data Mining Methods and Models*, by Daniel Larose (Wiley, 2006) or *Data Mining and Predictive Analytics*, by Daniel Larose and Chantal Larose (Wiley, 2015, to appear).

Note that this strategy applies to uncovering correlation among the predictors alone, not between a given predictor and the target variable.

Turning to our data set, for each of *day*, *evening*, *night*, and *international*, the data set contains three variables, *minutes*, *calls*, and *charge*. The data description indicates that the *charge* variable may be a function of *minutes* and *calls*, with the result that the variables would be correlated. We investigate using a *matrix plot* (Figure 3.31), which is a matrix of scatter plots for a set of numeric variables, in this case for *day minutes*, *day calls*, and *day charge*. Figure 3.32 contains the correlation coefficient values and the *p*-values for each pairwise set of variables.

There does not seem to be any relationship between *day minutes* and *day calls*, nor between *day calls* and *day charge*. This we find to be rather odd, as one may have expected that, as the number of calls increased, the number of minutes would tend to increase (and similarly for charge), resulting in a positive correlation between these fields. However, the graphical evidence in Figure 3.31 does not support this, nor do the correlations in Figure 3.32, which are $r = 0.07$ for both relationships, with a large *p*-value of 0.697.

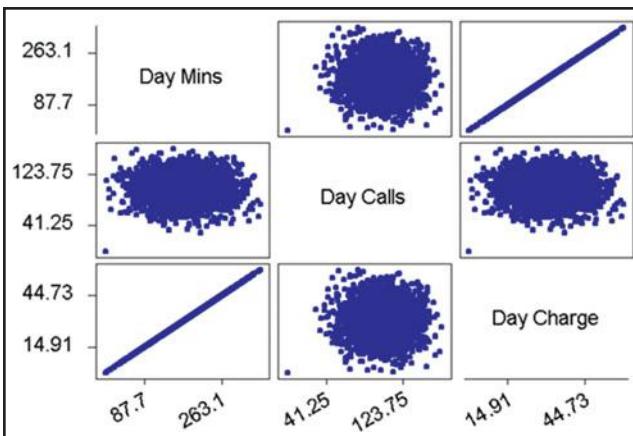


Figure 3.31 Matrix plot of *Day Minutes*, *Day Calls*, and *Day Charge*.

Correlations: Day Mins, Day Calls, Day Charge		
	Day Mins	Day Calls
Day Calls	0.007 0.697	
Day Charge	1.000 0.000	0.007 0.697
Cell Contents: Pearson correlation P-Value		

Figure 3.32 Correlations and *p*-values.

On the other hand, there is a perfect linear relationship between *day minutes* and *day charge*, indicating that *day charge* is a simple linear function of *day minutes* only. Using *Minitab*’s regression tool (see Figure 3.33), we find that we may express this function as the estimated regression equation: “*Day charge* equals 0.000613 plus 0.17 times *Day minutes*.” This is essentially a flat rate model, billing 17 cents per minute for day use. Note from Figure 3.33 that the *R*-squared statistic is precisely 100%, indicating a perfect linear relationship.

Since *day charge* is perfectly correlated with *day minutes*, then we should eliminate one of the two variables. We do so, arbitrarily choosing to eliminate *day charge* and retain *day minutes*. Investigation of the *evening*, *night*, and *international* components reflected similar findings, and we thus also eliminate *evening charge*, *night charge*, and *international charge*. Note that had we proceeded to the modeling phase without first uncovering these correlations, our data mining and statistical models may have returned incoherent results, due, for example, to multicollinearity in multiple regression. We have therefore reduced the number of predictors from 20 to 16 by eliminating one of each pair of perfectly correlated predictors. A further benefit of doing so is that the dimensionality of the solution space is reduced, so that certain data mining algorithms may more efficiently find the globally optimal solution.

After dealing with the perfectly correlated predictors, the data analyst should turn to Step 2 of the strategy, and *identify any other correlated predictors*, for later handling with principal component analysis. The correlation of each numerical predictor with every other numerical predictor should be checked, if feasible. Correlations with small *p*-values should be identified. A subset of this procedure is shown here in Figure 3.34. Note that the correlation coefficient 0.038 between *account length* and *day calls* has a small *p*-value of 0.026, telling us that account length and day calls are positively correlated. The data analyst should note this, and prepare to apply principal component analysis during the modeling phase.

3.12 SUMMARY

Let us consider some of the insights we have gained into the *churn* data set through the use of exploratory data analysis. We have examined each of the variables (here and in the exercises), and have taken a preliminary look at their relationship with *churn*.

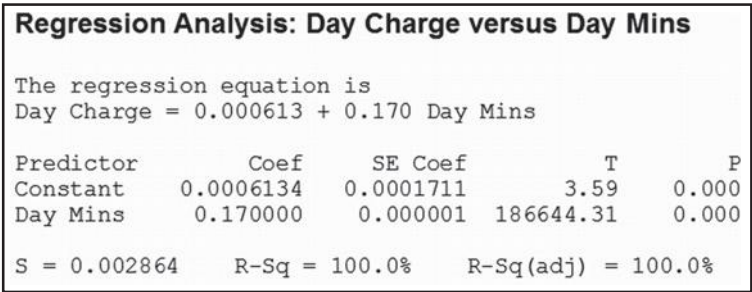


Figure 3.33 *Minitab* regression output for *Day Charge* vs. *Day Minutes*.

Correlations: Account Leng, VMail Messag, Day Mins, Day Calls, CustServ Cal				
	Account Length	VMail Message	Day Mins	Day Calls
VMail Message	-0.005 0.789			
Day Mins	0.006 0.720	0.001 0.964		
Day Calls	0.038 0.026	-0.010 0.582	0.007 0.697	
CustServ Calls	-0.004 0.827	-0.013 0.444	-0.013 0.439	-0.019 0.274
Cell Contents: Pearson correlation P-Value				

Figure 3.34 Account length is positively correlated with day calls.

- The four *charge* fields are linear functions of the *minute* fields, and should be omitted.
- The *area code* field and/or the *state* field are anomalous, and should be omitted until further clarification is obtained.

Insights with respect to *churn*:

- Customers with the *International Plan* tend to churn more frequently.
- Customers with the *VoiceMail Plan* tend to churn less frequently.
- Customers with four or more *Customer Service Calls* churn more than four times as often as the other customers.
- Customers with high *Day Minutes* and *Evening Minutes* tend to churn at a higher rate than the other customers.
- Customers with both high *Day Minutes* and high *Evening Minutes* churn at a rate about six times greater than the other customers.
- Customers with low *Day Minutes* and high *Customer Service Calls* churn at a higher rate than the other customers.
- Customers with lower numbers of *International Calls* churn at a higher rate than do customers with more international calls.
- For the remaining predictors, EDA uncovers no obvious association of *churn*. However, these variables are still retained for input to downstream data mining models and techniques.

Note the power of exploratory data analysis. We have not applied any high powered data mining algorithms yet on this data set, such as decision trees or neural network algorithms. Yet, we have still gained considerable insight into the attributes that are associated with the customers leaving the company, simply by careful application of exploratory data analysis. These insights can be easily formulated into actionable recommendations, so that the company can take action to lower the churn rate among its customer base.

THE R ZONE

Input data set Churn into Data Frame "Churn"

```
churn <- read.csv(file = "C:/... /churn.txt",
  stringsAsFactors=TRUE)
# Show the first ten records
churn[1:10,]
```

```
> churn[1:10,]
  State Account.Length Area.Code Phone Int.l.Plan
1    KS             128     415 382-4657      no
2    OH             107     415 371-7191      no
3    NJ             137     415 358-1921      no
4    OH              84     408 375-9999     yes
5    OK              75     415 330-6626     yes
6    AL             118     510 391-8027     yes
7    MA             121     510 355-9993      no
8    MO             147     415 329-9001     yes
9    LA             117     408 335-4719      no
10   WV             141     415 330-8173     yes
```

Summarize the Churn variable

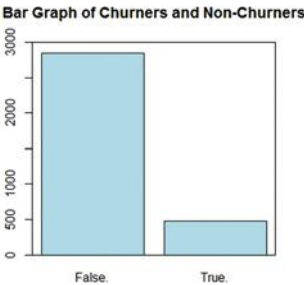
```
sum.churn <- summary(churn$Churn)
sum.churn
```

Calculate proportion of churners

```
prop.churn <- sum(churn$Churn == "True") /
  length(churn$Churn)
prop.churn
```

Bar chart of variable Churn

```
barplot(sum.churn,
  ylim = c(0, 3000),
  main = "Bar Graph of Churners and Non-Churners",
  col = "lightblue")
box(which = "plot",
  lty = "solid",
  col="black")
```



Make a table for counts of Churn and International Plan

```
counts <- table(churn$Churn, churn$Int.l.Plan,
  dnn=c("Churn", "International Plan"))
counts
```

```
> counts
      International Plan
Churn   no  yes
False. 2664 186
True.   346  137
```

Create a table with sums for both variables

```
sumtable <- addmargins(counts, FUN = sum)
sumtable
```

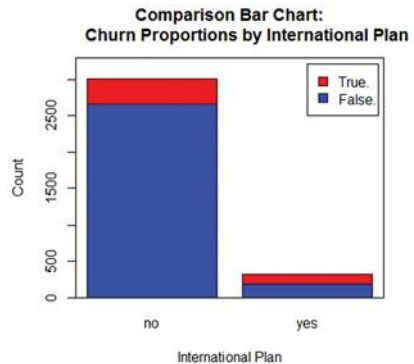
```
> sumtable
      International Plan
Churn   no  yes  sum
False. 2664 186 2850
True.   346  137  483
sum     3010 323 3333
```

Overlaid bar chart

```

barplot(counts,
  legend = rownames(counts),
  col = c("blue", "red"),
  ylim = c(0, 3300),
  ylab = "Count",
  xlab = "International Plan",
  main = "Comparison Bar Chart:
  Churn Proportions by International Plan")
box(which = "plot",
  lty = "solid",
  col = "black")

```

**# Create a table of proportions over rows**

```

row.margin <- round(prop.table(counts,
  margin = 1),
  4)*100
row.margin

```

```

> row.margin
      International Plan
Churn   no   yes
False. 93.47  6.53
True.  71.64 28.36

```

Create a table of proportions over columns

```

col.margin <- round(prop.table(counts,
  margin = 2),
  4)*100
col.margin

```

```

> col.margin
      International Plan
Churn   no   yes
False. 88.50 57.59
True.  11.50 42.41

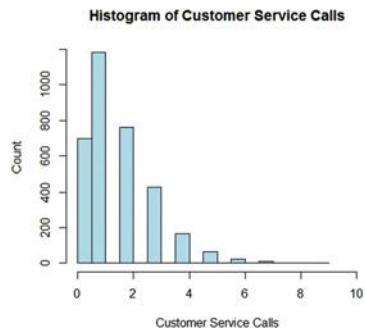
```

Histogram of non-overlaid Customer Service Calls

```

hist(churn$CustServ.Calls,
  xlim = c(0,10),
  col = "lightblue",
  ylab = "Count",
  xlab = "Customer Service Calls",
  main = "Histogram of Customer Service Calls")

```

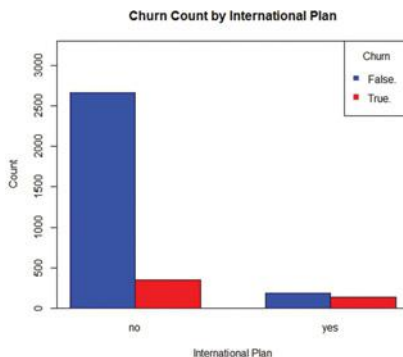


Clustered Bar Chart, with legend

```

barplot(counts,
  col = c("blue", "red"),
  ylim = c(0, 3300),
  ylab = "Count",
  xlab = "International Plan",
  main = "Churn Count by International Plan",
  beside = TRUE)
legend("topright",
  c(rownames(counts)),
  col = c("blue", "red"),
  pch = 15,
  title = "Churn")
box(which = "plot",
  lty = "solid",
  col = "black")

```

**# Download and install the R Package ggplot2**

```

install.packages("ggplot2")
# Pick any CRAN mirror
# (see example image)
# Open the new package
library(ggplot2)

```

```

70: Turkey
73: UK (St Andrews)
76: USA (IA)
79: USA (MD)
82: USA (OH)
85: USA (PA 2)
88: USA (WA 1)
91: Vietnam
Selection: 74

```

```

71: UK (Bristol)
74: USA (CA 1)
77: USA (IN)
80: USA (MI)
83: USA (OR)
86: USA (TN)
89: USA (WA 2)

```

```

72: UK (London)
75: USA (CA 2)
78: USA (KS)
81: USA (MO)
84: USA (PA 1)
87: USA (TX 1)
90: Venezuela

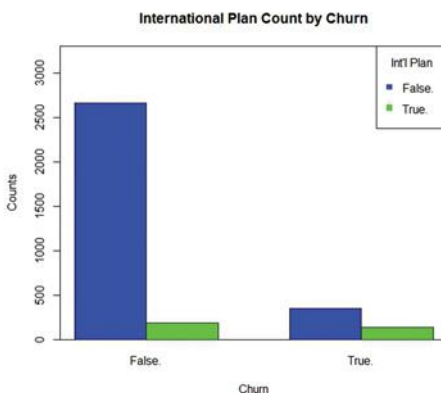
```

Clustered Bar Chart of Churn and Int'l Plan with legend

```

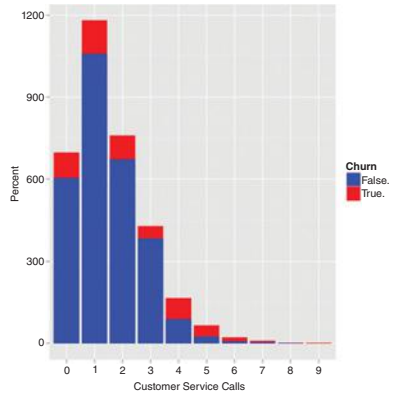
barplot(t(counts),
  col = c("blue", "green"),
  ylim = c(0, 3300),
  ylab = "Counts",
  xlab = "Churn",
  main = "International Plan Count by Churn",
  beside = TRUE)
legend("topright",
  c(rownames(counts)),
  col = c("blue", "green"),
  pch = 15,
  title = "Int'l Plan")
box(which = "plot",
  lty = "solid",
  col = "black")

```

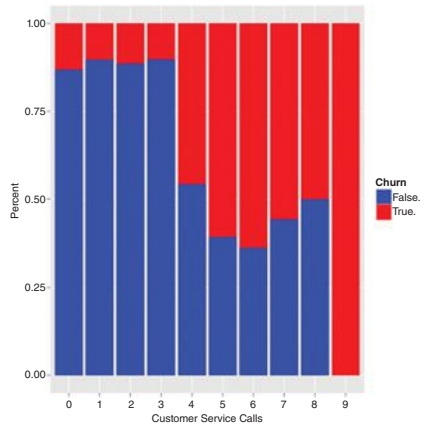


Overlaid bar charts

```
ggplot() +
  geom_bar(data = churn,
    aes(x = factor(churn$CustServ.Calls),
      fill = factor(churn$Churn.)),
    position = "stack") +
  scale_x_discrete("Customer Service Calls") +
  scale_y_continuous("Percent") +
  guides(fill=guide_legend(title="Churn")) +
  scale_fill_manual(values=c("blue", "red"))
```



```
ggplot() +
  geom_bar(data=churn,
    aes(x = factor(churn$CustServ.Calls),
      fill = factor(churn$Churn.)),
    position = "fill") +
  scale_x_discrete("Customer Service Calls") +
  scale_y_continuous("Percent") +
  guides(fill=guide_legend(title="Churn")) +
  scale_fill_manual(values=c("blue", "red"))
```



Two-sample T-Test for Int'l Calls

```
# Partition data
churn.false <- subset(churn,
  churn$Churn == "False")
churn.true <- subset(churn,
  churn$Churn == "True")
# Run the test
t.test(churn.false$Intl.Calls,
  churn.true$Intl.Calls)
```

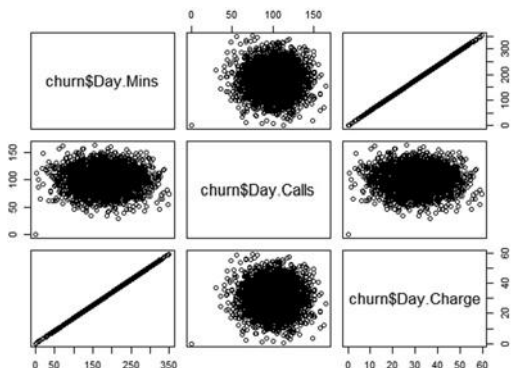
```
> t.test(churn.false$Intl.Calls,
+        churn.true$Intl.Calls)

welch Two Sample t-test

data:  churn.false$Intl.Calls and churn.true$Intl.Calls
t = 2.9604, df = 640.643, p-value = 0.003186
alternative hypothesis: true difference in means is not
equal to 0
95 percent confidence interval:
 0.1243807 0.6144620
sample estimates:
mean of x mean of y
 4.532982  4.163561
```

Scatterplot matrix

```
pairs(~churn$Day.Mins+
      churn$Day.Calls+
      churn$Day.Charge)
```



Regression of Day Charge vs Day Minutes

```
fit <- lm(churn$Day.Charge ~
          churn$Day.Mins)
summary(fit)
```

```
> summary(fit)

Call:
lm(formula = churn$Day.Charge ~ churn$Day.Mins)

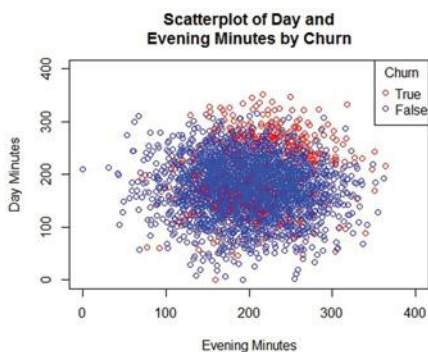
Residuals:
    Min       1Q   Median       3Q      Max
-0.0045935 -0.0025391  0.0004326  0.0024587  0.0045224

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.134e-04  1.711e-04  3.585e+00  0.000341 ***
churn$Day.Mins 1.700e-01  9.108e-07  1.866e+05  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.002864 on 3331 degrees of freedom
Multiple R-squared:  1, Adjusted R-squared:  1
F-statistic: 3.484e+10 on 1 and 3331 DF, p-value: < 2.2e-16
```

Scatterplot of Evening Minutes and Day Minutes, colored by Churn

```
plot(churn$Eve.Mins,
     churn$Day.Mins,
     xlim = c(0, 400),
     ylim = c(0, 400),
     xlab = "Evening Minutes",
     ylab = "Day Minutes",
     main = "Scatterplot of Day and Evening Minutes by Churn",
     col = ifelse(churn$Churn=="True",
                  "red",
                  "blue"))
legend("topright",
      c("True",
        "False"),
      col = c("red",
              "blue"),
      pch = 1,
      title = "Churn")
```

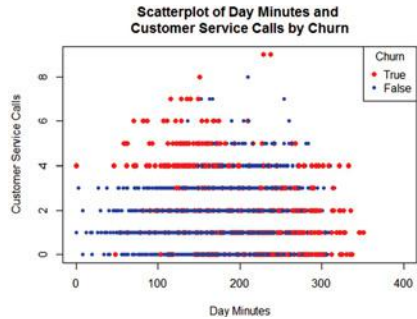


Scatterplot of Day Minutes and Customer Service Calls, colored by Churn

```

plot(churn$Day.Mins,
     churn$CustServ.Calls,
     xlim = c(0, 400),
     xlab = "Day Minutes",
     ylab = "Customer Service Calls",
     main = "Scatterplot of Day Minutes and
           Customer Service Calls by Churn",
     col = ifelse(churn$Churn=="True",
                  "red",
                  "blue"),
     pch = ifelse(churn$Churn=="True",
                  16, 20))
legend("topright",
      c("True",
        "False"),
      col = c("red",
              "blue"),
      pch = c(16, 20),
      title = "Churn")

```

**# Correlation values, with p-values**

```

days <- cbind(churn$Day.Mins,
               churn$Day.Calls,
               churn$Day.Charge)
MinsCallsTest <- cor.test(churn$Day.Mins,
                          churn$Day.Calls)
MinsChargeTest <- cor.test(churn$Day.Mins,
                          churn$Day.Charge)
CallsChargeTest <- cor.test(churn$Day.Calls,
                          churn$Day.Charge)
round(cor(days),
      4)
MinsCallsTest$p.value
MinsChargeTest$p.value
CallsChargeTest$p.value

```

```

> round(cor(days), 4)
      [,1] [,2] [,3]
[1,] 1.0000 0.0068 1.0000
[2,] 0.0068 1.0000 0.0068
[3,] 1.0000 0.0068 1.0000
> MinsCallsTest$p.value
[1] 0.6968515
> MinsChargeTest$p.value
[1] 0
> CallsChargeTest$p.value
[1] 0.6967428

```

Correlation values and p-values in matrix form

```
# Collect variables of interest
```

```
corrdata <- cbind(churn$Account.Length,
  churn$VMail.Message,
  churn$Day.Mins,
  churn$Day.Calls,
  churn$CustServ.Calls)
```

```
# Declare the matrix
```

```
corrvalues <- matrix(rep(0, 25),
  ncol = 5)
```

```
# Fill the matrix with correlations
```

```
for (i in 1:4) {
  for (j in (i+1):5) {
    corrvalues[i,j] <- corrvalues[j,i] <-
      round(cor.test(corrdata[,i],
        corrdata[,j])$p.value,
        4)
  }
}
```

```
round(cor(corrdata), 4)
```

```
corrvalues
```

```
> round(cor(corrdata), 4)
      [,1] [,2] [,3] [,4] [,5]
[1,] 1.0000 -0.0046 0.0062 0.0385 -0.0038
[2,] -0.0046 1.0000 0.0008 -0.0095 -0.0133
[3,] 0.0062 0.0008 1.0000 0.0068 -0.0134
[4,] 0.0385 -0.0095 0.0068 1.0000 -0.0189
[5,] -0.0038 -0.0133 -0.0134 -0.0189 1.0000
> corrvalues
      [,1] [,2] [,3] [,4] [,5]
[1,] 0.0000 0.7894 0.7198 0.0264 0.8266
[2,] 0.7894 0.0000 0.9642 0.5816 0.4440
[3,] 0.7198 0.9642 0.0000 0.6969 0.4385
[4,] 0.0264 0.5816 0.6969 0.0000 0.2743
[5,] 0.8266 0.4440 0.4385 0.2743 0.0000
```

REFERENCE

1. Blake, C.L. and Merz, C.J., *Churn Data Set*, UCI Repository of Machine Learning Databases, University of California, Department of Information and Computer Science, Irvine, CA, 1998, kdd.ics.uci.edu/. Accessed March 17, 2014.

EXERCISES

1. Explain the difference between EDA and hypothesis testing, and why analysts may prefer EDA when doing data mining.
2. Why do we need to perform exploratory data analysis? Why should not we simply proceed directly to the modeling phase and start applying our high powered data mining software?
3. Why do we use contingency tables, instead of just presenting the graphical results?
4. How can we find the marginal distribution of each variable in a contingency table?
5. What is the difference between taking row percentages and taking column percentages in a contingency table?
6. What is the graphical counterpart of a contingency table?
7. Describe what it would mean for interaction to take place between two categorical variables, using an example.

8. What type of histogram is useful for examining the relationship between a numerical predictor and the target?
9. Explain one benefit and one drawback of using a normalized histogram. Should we ever present a normalized histogram without showing its nonnormalized counterpart?
10. Explain whether we should omit a predictor from the modeling stage if it does not show any relationship with the target variable in the EDA stage, and why.
11. Describe how scatter plots can uncover patterns in two dimensions that would be invisible from one-dimensional EDA.
12. Make up a fictional data set (attributes with no records is fine) with a pair of anomalous attributes. Describe how EDA would help to uncover the anomaly.
13. Explain the objective and the method of binning based on predictive value.
14. Why is binning based on predictive value considered to be somewhat of an art?
15. What step should precede the deriving of a new numerical variable representing the mean of two other numerical variables?
16. What does it mean to say that two variables are correlated?
17. Describe the possible consequences of allowing correlated variables to remain in the model.
18. A common practice among some analysts when they encounter two correlated predictors is to omit one of them from the analysis. Is this practice recommended?
19. Describe the strategy for handling correlated predictor variables at the EDA stage.
20. For each of the following descriptive methods, state whether it may be applied to categorical data, continuous numerical data, or both.
 - a. Bar charts
 - b. Histograms
 - c. Summary statistics
 - d. Crosstabulations
 - e. Correlation analysis
 - f. Scatter plots
 - g. Web graphs
 - h. Binning

HANDS-ON ANALYSIS

21. Using the *churn* data set, develop EDA which shows that the remaining numeric variables in the data set (apart from those covered in the text above) indicate no obvious association with the target variable.
 Use the *Adult* data set from the book series website for the following exercises. The target variable is *income*, and the goal is to classify income based on the other variables.
22. Which variables are categorical and which are continuous?
23. Using software, construct a table of the first 10 records of the data set, in order to get a feel for the data.

24. Investigate whether we have any correlated variables.
25. For each of the categorical variables, construct a bar chart of the variable, with an overlay of the target variable. Normalize if necessary.
 - a. Discuss the relationship, if any, each of these variables has with the target variables.
 - b. Which variables would you expect to make a significant appearance in any data mining classification model we work with?
26. For each pair of categorical variables, construct a crosstabulation. Discuss your salient results.
27. (If your software supports this.) Construct a web graph of the categorical variables. Fine tune the graph so that interesting results emerge. Discuss your findings.
28. Report on whether anomalous fields exist in this data set, based on your EDA, which fields these are, and what we should do about it.
29. Report the mean, median, minimum, maximum, and standard deviation for each of the numerical variables.
30. Construct a histogram of each numerical variables, with an overlay of the target variable *income*. Normalize if necessary.
 - a. Discuss the relationship, if any, each of these variables has with the target variables.
 - b. Which variables would you expect to make a significant appearance in any data mining classification model we work with?
31. For each pair of numerical variables, construct a scatter plot of the variables. Discuss your salient results.
32. Based on your EDA so far, identify interesting sub-groups of records within the data set that would be worth further investigation.
33. Apply binning to one of the numerical variables. Do it in such a way as to maximize the effect of the classes thus created (following the suggestions in the text). Now do it in such a way as to minimize the effect of the classes, so that the difference between the classes is diminished. Comment.
34. Refer to the previous exercise. Apply the other two binning methods (equal width, and equal number of records) to this same variable. Compare the results and discuss the differences. Which method do you prefer?
35. Summarize your salient EDA findings from the above exercises, just as if you were writing a report. ■

UNIVARIATE STATISTICAL ANALYSIS

4.1	DATA MINING TASKS IN <i>DISCOVERING KNOWLEDGE IN DATA</i>	91
4.2	STATISTICAL APPROACHES TO ESTIMATION AND PREDICTION	92
4.3	STATISTICAL INFERENCE	93
4.4	HOW CONFIDENT ARE WE IN OUR ESTIMATES?	94
4.5	CONFIDENCE INTERVAL ESTIMATION OF THE MEAN	95
4.6	HOW TO REDUCE THE MARGIN OF ERROR	97
4.7	CONFIDENCE INTERVAL ESTIMATION OF THE PROPORTION	98
4.8	HYPOTHESIS TESTING FOR THE MEAN	99
4.9	ASSESSING THE STRENGTH OF EVIDENCE AGAINST THE NULL HYPOTHESIS	101
4.10	USING CONFIDENCE INTERVALS TO PERFORM HYPOTHESIS TESTS	102
4.11	HYPOTHESIS TESTING FOR THE PROPORTION	104
	THE R ZONE	105
	REFERENCE	106
	EXERCISES	106

4.1 DATA MINING TASKS IN *DISCOVERING KNOWLEDGE IN DATA*

In Chapter 1 we were introduced to the six data mining tasks:

- Description
- Estimation
- Prediction
- Classification

Discovering Knowledge in Data: An Introduction to Data Mining, Second Edition.
By Daniel T. Larose and Chantal D. Larose.
© 2014 John Wiley & Sons, Inc. Published 2014 by John Wiley & Sons, Inc.

- Clustering
- Association

In the description task, analysts try to find ways to describe patterns and trends lying within the data. Descriptions of patterns and trends often suggest possible explanations for such patterns and trends, as well as possible recommendations for policy changes. This description task can be accomplished capably with exploratory data analysis (EDA), as we saw in Chapter 3. The description task may also be performed using descriptive statistics, such as the sample proportion or the regression equation, which we learn about in Chapters 4 and 5. Table 4.1 provides an outline of where in this book we learn about each of the data mining tasks.

Of course, the data mining methods are not restricted to only one task each, which results in a fair amount of overlap among data mining methods and tasks. For example, decision trees may be used for classification, estimation, or prediction. Therefore, Table 4.1 should not be considered as a definitive partition of the tasks, but rather as a general outline of how we are introduced to the tasks and the methods used to accomplish them.

TABLE 4.1 Data mining tasks in *Discovering Knowledge in Data*

Task	We Learn about This Task in
Description	Chapter 3: Exploratory data analysis Chapter 4: Univariate statistical analysis Chapter 5: Multivariate statistical analysis
Estimation	Chapter 4: Univariate statistical analysis Chapter 5: Multivariate statistical analysis
Prediction	Chapter 4: Univariate statistical analysis Chapter 5: Multivariate statistical analysis
Classification	Chapter 7: k -Nearest neighbor algorithm Chapter 8: Decision trees Chapter 9: Neural networks
Clustering	Chapter 10: Hierarchical and k -means clustering Chapter 11: Kohonen networks
Association	Chapter 12: Association rules

4.2 STATISTICAL APPROACHES TO ESTIMATION AND PREDICTION

If estimation and prediction are considered to be data mining tasks, statistical analysts have been performing data mining for over a century. In Chapters 4 and 5 we examine widespread and traditional methods of estimation and prediction, drawn from the world of statistical analysis. Here in Chapter 4 we examine univariate

methods, statistical estimation, and prediction methods that analyze one variable at a time. These methods include point estimation and confidence interval estimation for population means and proportions. We discuss ways of reducing the margin of error of a confidence interval estimate. Then we turn to hypothesis testing, examining hypothesis tests for population means and proportions. Then, in Chapter 5 we consider multivariate methods for statistical estimation and prediction.

4.3 STATISTICAL INFERENCE

Consider our roles as data miners. We have been presented with a data set with which we are presumably unfamiliar. We have completed the data understanding and data preparation phases and have gathered some descriptive information using EDA. Next, we would like to perform univariate estimation and prediction. A widespread tool for performing estimation and prediction is *statistical inference*.

Statistical inference consists of methods for estimating and testing hypotheses about population characteristics based on the information contained in the sample. A *population* is the collection of *all* elements (persons, items, or data) of interest in a particular study.

For example, presumably, the cell phone company does not want to restrict its actionable results to the sample of 3333 customers from which it gathered the data. Rather, it would prefer to deploy its churn model to *all* of its present and future cell phone customers, which would therefore represent the population. A *parameter* is a characteristic of a population, such as the mean number of customer service calls of all cell phone customers.

A *sample* is simply a subset of the population, preferably a representative subset. If the sample is not representative of the population, that is, if the sample characteristics deviate systematically from the population characteristics, *statistical inference should not be applied*. A *statistic* is a characteristic of a sample, such as the mean number of customer service calls of the 3333 customers in the sample (1.563).

Note that the values of population parameters are *unknown* for most interesting problems. Specifically, the value of the population mean is usually unknown. For example, we do not know the true mean number of customer service calls to be made by all of the company's cell phone customers. To represent their unknown nature, population parameters are often denoted with Greek letters. For example, the population mean is symbolized using the Greek lowercase letter μ (pronounced "myu"), which is the Greek letter for "m" ("mean").

The value of the population mean number of customer service calls μ is unknown for a variety of reasons, including the fact that the data may not yet have been collected or warehoused. Instead, data analysts would use *estimation*. For example, they would estimate the unknown value of the population mean μ by obtaining a sample and computing the sample mean \bar{x} , which would be used to estimate μ . Thus, we would estimate the mean number of customer service calls for all customers to be 1.563, since this is the value of our observed sample mean.

An important *caveat* is that estimation is valid only as long as the sample is truly representative of the population. For example, suppose for a moment that the

churn data set represents a sample of 3333 disgruntled customers. Then this sample would not be representative (one hopes!) of the population of all the company’s customers, and none of the EDA that we performed in Chapter 3 would be actionable with respect to the population of all customers.

Analysts may also be interested in proportions, such as the proportion of customers who churn. The sample proportion p is the statistic used to measure the unknown value of the population proportion, π . For example, in Chapter 3 we found that the proportion of churners in the data set was $p = 0.145$, which could be used to estimate the true proportion of churners for the population of all customers, keeping in mind the caveats above.

Point estimation refers to the use of a single known value of a statistic to estimate the associated population parameter. The observed value of the statistic is called the *point estimate*. We may summarize estimation of the population mean, standard deviation, and proportion using Table 4.2.

Estimation need not be restricted to the parameters in Table 4.2. Any statistic observed from sample data may be used to estimate the analogous parameter in the population. For example, we may use the sample maximum to estimate the population maximum, or we could use the sample 27th percentile to estimate the population 27th percentile. *Any* sample characteristic is a statistic, which, under the appropriate circumstances, can be used to estimate its respective parameter.

More specifically, for example, we could use the sample churn proportion of customers who did select the VoiceMail Plan, but did not select the International Plan, and who made three customer service calls to estimate the population churn proportion of all such customers. Or, we could use the sample 99th percentile of day minutes used for customers without the VoiceMail Plan to estimate the population 99th percentile of day minutes used for all customers without the VoiceMail Plan.

TABLE 4.2 Use observed sample statistics to estimate unknown population parameters

	Sample Statistic	... Estimates ...	Population Parameter
Mean	\bar{x}	→	μ
Standard deviation	s	→	σ
Proportion	p	→	π

4.4 HOW CONFIDENT ARE WE IN OUR ESTIMATES?

Let us face it: anyone can make estimates. Crystal ball gazers will be happy (for a price) to provide you with an estimate of the parameter in which you are interested. The question is: *How confident can we be in the accuracy of the estimate?*

Do you think that the population mean number of customer service calls made by all of the company’s customers is exactly the same as the sample mean $\bar{x} = 1.563$? Probably not. In general, since the sample is a subset of the population, inevitably the population contains more information than the sample about any given characteristic. Hence, unfortunately, our point estimates will nearly always “miss”

the target parameter by a certain amount, and thus be in error by this amount, which is probably, though not necessarily, small.

This distance between the observed value of the point estimate and the unknown value of its target parameter is called *sampling error*, defined as $|\text{statistic} - \text{parameter}|$. For example, the sampling error for the mean is $|\bar{x} - \mu|$, the distance (always positive) between the observed sample mean and the unknown population mean. Since the true values of the parameter are usually unknown, the value of the sampling error is usually unknown in real-world problems. In fact, for continuous variables, the probability that the observed value of a point estimate exactly equals its target parameter is precisely zero. This is because probability represents area above an interval for continuous variables, and there is no area above a point.

Point estimates have no measure of confidence in their accuracy; there is no probability statement associated with the estimate. All we know is that the estimate is probably close to the value of the target parameter (small sampling error) but that possibly it may be far away (large sampling error). In fact, point estimation has been likened to a dart thrower, throwing darts with infinitesimally small tips (the point estimates) toward a vanishingly small bull's-eye (the target parameter). Worse, the bull's-eye is hidden, and the thrower will never know for sure how close the darts are coming to the target.

The dart thrower could perhaps be forgiven for tossing a beer mug in frustration rather than a dart. But wait! Since the beer mug has width, there does indeed exist a positive probability that some portion of the mug has hit the hidden bull's-eye. We still do not know for sure, but we can have a certain degree of confidence that the target has been hit. Very roughly, the beer mug represents our next estimation method, *confidence intervals*.

4.5 CONFIDENCE INTERVAL ESTIMATION OF THE MEAN

A *confidence interval estimate* of a population parameter consists of an interval of numbers produced by a point estimate, together with an associated *confidence level* specifying the probability that the interval contains the parameter. Most confidence intervals take the general form

$$\text{point estimate} \pm \text{margin of error}$$

where the margin of error is a measure of the precision of the interval estimate. Smaller margins of error indicate greater precision. For example, the t -interval for the population mean is given by

$$\bar{x} \pm t_{\alpha/2}(s/\sqrt{n})$$

where the sample mean \bar{x} is the point estimate and the quantity $t_{\alpha/2}(s/\sqrt{n})$ represents the margin of error. The t -interval for the mean may be used when either the population is normal or the sample size is large.

Under what conditions will this confidence interval provide precise estimation? That is, when will the margin of error $t_{\alpha/2}(s/\sqrt{n})$ be small? The quantity s/\sqrt{n} represents the standard error of the sample mean (the standard deviation of the sampling distribution of \bar{x}) and is small whenever the sample size is large or the sample variability is small. The multiplier $t_{\alpha/2}$ is associated with the sample size and the confidence level (usually 90–99%) specified by the analyst, and is smaller for lower confidence levels. Since we cannot influence the sample variability directly and we hesitate to lower our confidence level, we must turn to increasing the sample size should we seek to provide more precise confidence interval estimation.

Usually, finding a large sample size is not a problem for many data mining scenarios. For example, using the statistics in Figure 4.1, we can find the 95% t -interval for the mean number of customer service calls for all customers as follows:

$$\begin{aligned} \bar{x} \pm t_{\alpha/2}(s/\sqrt{n}) \\ 1.563 \pm 1.96(1.315/\sqrt{3333}) \\ 1.563 \pm 0.045 \\ (1.518, 1.608) \end{aligned}$$

We are 95% confident that the population mean number of customer service calls for all customers falls between 1.518 and 1.608 calls. Here, the margin of error is 0.045 customer service calls, which is fairly precise for most applications.

However, data miners are often called upon to perform *subgroup analyses*, that is, to estimate the behavior of specific subsets of customers instead of the entire customer base, as in the example above. For example, suppose that we are interested in estimating the mean number of customer service calls for customers who have both the International Plan and the VoiceMail Plan and who have more than 220 day minutes. This reduces the sample size to 28 (Figure 4.2), which however is still large enough to construct the confidence interval.

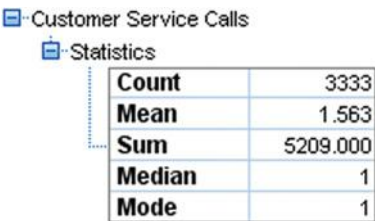


Figure 4.1 Summary statistics of customer service calls.

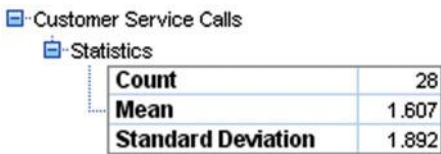


Figure 4.2 Summary statistics of customer service calls for those with both the International Plan and VoiceMail Plan and with more than 200 day minutes.

There are only 28 customers in the sample who have both plans and who logged more than 220 minutes of day use. The point estimate for the population mean number of customer service calls for all such customers is the sample mean 1.607. We may find the 95% t -confidence interval estimate as follows:

$$\begin{aligned}\bar{x} \pm t_{\alpha/2}(s/\sqrt{n}) \\ 1.607 \pm 2.048(1.892/\sqrt{28}) \\ 1.607 \pm 0.732 \\ (0.875, 2.339)\end{aligned}$$

We are 95% confident that the population mean number of customer service calls for all customers who have both plans and who have more than 220 minutes of day use falls between 0.875 and 2.339 calls. Here, 0.875 is called the *lower bound* and 2.339 is called the *upper bound* of the confidence interval. The margin of error for this specific subset of customers is 0.732, which indicates that our estimate of the mean number of customer service calls for this subset of customers is much less precise than for the customer base as a whole.

Confidence interval estimation can be applied to any desired target parameter. The most widespread interval estimates are for the population mean and the population proportion.

4.6 HOW TO REDUCE THE MARGIN OF ERROR

The margin of error E for a 95% confidence interval for the population mean μ is $E = t_{\alpha/2}(s/\sqrt{n})$ and may be interpreted as follows:

We can estimate μ to within E units with 95% confidence.

For example, the margin of error above of the number of customer service calls for all customers equals 0.045 service calls, which may be interpreted as, “We can estimate the mean number of customer service calls for all customers to within 0.045 calls with 95% confidence.”

Now, the smaller the margin of error, the more precise our estimation. So the question arises, *how can we reduce our margin of error?* Now the margin of error E contains three quantities:

- $t_{\alpha/2}$, which depends on the confidence level and the sample size,
- the sample standard deviation s , which is a characteristic of the data, and may not be changed, and
- n , the sample size.

Thus, we may decrease our margin of error in two ways

- By decreasing the confidence level, which reduces the value of $t_{\alpha/2}$, and therefore reduces E . **Not recommended.**

- By increasing the sample size. **Recommended.** Increasing the sample size is the only way to decrease the margin of error while maintaining a constant level of confidence.

For example, had we procured a new sample of 5000 customers, with the same standard deviation $s = 1.315$, then the margin of error for a 95% confidence interval would be:

$$E = t_{\alpha/2}(s/\sqrt{n}) = 1.96(1.315/\sqrt{5000}) = 0.036$$

Due to the \sqrt{n} in the formula for E , an increase of n in the sample size leads to a reduction in margin of error of \sqrt{a} .

4.7 CONFIDENCE INTERVAL ESTIMATION OF THE PROPORTION

Figure 3.3 showed that 483 of 3333 customers had churned, so that an estimate of the *population proportion* π of all of the company's customers who churn is

$$p = \frac{\text{number who churn}}{\text{sample size}} = \frac{x}{n} = \frac{483}{3333} = 0.1449$$

Unfortunately, with respect to the population of our entire customer base, we have no measure of our confidence in the accuracy of this estimate. In fact, it is nearly impossible that this value exactly equals π . Thus, we would prefer a *confidence interval for the population proportion* π , given as follows.

$$p \pm Z_{\alpha/2} \sqrt{\frac{p \cdot (1-p)}{n}}$$

where the sample proportion p is the point estimate of π and the quantity $Z_{\alpha/2} \sqrt{\frac{p \cdot (1-p)}{n}}$ represents the margin of error. The quantity $Z_{\alpha/2}$ depends on the confidence level: for 90% confidence $Z_{\alpha/2} = 1.645$, for 95% confidence $Z_{\alpha/2} = 1.96$, and for 99% confidence $Z_{\alpha/2} = 2.576$. This Z-interval for π may be used whenever both $np \geq 5$ and $n(1-p) \geq 5$.

For example, a 95% confidence interval for the proportion π of churners among the entire population of the company's customers is given by:

$$\begin{aligned} p \pm Z_{\alpha/2} \sqrt{\frac{p \cdot (1-p)}{n}} &= 0.1449 \pm 1.96 \sqrt{\frac{(0.1449)(0.8551)}{3333}} \\ &= 0.1449 \pm 0.012 \\ &= (0.1329, 0.1569) \end{aligned}$$

We are 95% confident that this interval captures the population proportion π . Note that the confidence interval for π takes the form

$$p \pm E = 0.1449 \pm 0.012$$

where the margin of error E for a 95% confidence interval for the population mean π is $E = Z_{\alpha/2} \sqrt{\frac{p \cdot (1-p)}{n}}$. The margin of error may be interpreted as follows:

We can estimate π to within E with 95% confidence.

In this case, we can estimate the population proportion of churners to within 0.012 (or 1.2%) with 95% confidence. For a given confidence level, the margin of error can be reduced only by taking a larger sample size.

4.8 HYPOTHESIS TESTING FOR THE MEAN

Hypothesis testing is a procedure where claims about the value of a population parameter (such as μ or π) may be considered using the evidence from the sample. Two competing statements, or *hypotheses*, are crafted about the parameter value:

- The *null hypothesis* H_0 is the status quo hypothesis, representing what has been assumed about the value of the parameter, and
- The *alternative hypothesis* or *research hypothesis* H_a represents an alternative claim about the value of the parameter.

The two possible conclusions are (a) Reject H_0 , and (b) Do not reject H_0 . A criminal trial is a form of a hypothesis test, with the following hypotheses:

$$H_0 : \text{Defendant is innocent} \quad H_a : \text{Defendant is guilty}$$

Table 4.3 illustrates the four possible outcomes of the criminal trial with respect to the jury’s decision, and what is true in reality.

- *Type I error*: Reject H_0 when H_0 is true. The jury convicts an innocent person.
- *Type II error*: Do not reject H_0 when H_0 is false. The jury acquits a guilty person.
- Correct decisions:
 - Reject H_0 when H_0 is false. The jury convicts a guilty person.
 - Do not reject H_0 when H_0 is true. The jury acquits an innocent person.

TABLE 4.3 Four possible outcomes of the criminal trial hypothesis test

		Reality	
		H_0 true: defendant did not commit crime	H_0 false: defendant did commit crime
Jury’s decision	Reject H_0 : find defendant guilty	Type I error	Correct decision
	Do not Reject H_0 : find defendant not guilty	Correct decision	Type II error

The probability of a Type I error is denoted α , while the probability of a Type II error is denoted β . For a constant sample size, a decrease in α is associated with an increase in β , and vice versa. In statistical analysis, α is usually fixed at some small value, such as 0.05, and called the *level of significance*.

A common treatment of hypothesis testing for the mean is to restrict the hypotheses to the following three forms.

- Left-tailed test. $H_0 : \mu \geq \mu_0$ vs. $H_a : \mu < \mu_0$
- Right-tailed test. $H_0 : \mu \leq \mu_0$ vs. $H_a : \mu > \mu_0$
- Two-tailed test. $H_0 : \mu = \mu_0$ vs. $H_a : \mu \neq \mu_0$

where μ_0 represents a hypothesized value of μ .

When the sample size is large or the population is normally distributed, the test statistic

$$t_{\text{data}} = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}$$

follows a t distribution, with $n - 1$ degrees of freedom. The value of t_{data} is interpreted as the number of standard errors above or below the hypothesized mean μ , that the sample mean \bar{x} resides, where the standard error equals s/\sqrt{n} . (Roughly, the *standard error* represents a measure of spread of the distribution of a statistic.) When the value of t_{data} is extreme, this indicates a conflict between the null hypothesis (with the hypothesized value μ_0) and the observed data. Since the data represent empirical evidence whereas the null hypothesis represents merely a claim, such conflicts are resolved in favor of the data, so that, when t_{data} is extreme, the null hypothesis H_0 is rejected. How extreme is extreme? This is measured using the p -value.

The p -value is the probability of observing a sample statistic (such as \bar{x} or t_{data}) at least as extreme as the statistic actually observed, if we assume that the null hypothesis is true. Since the p -value (“probability value”) represents a probability, its value must always fall between zero and one. Table 4.4 indicates how to calculate the p -value for each form of the hypothesis test.

The names of the forms of the hypothesis test indicate in which tail or tails of the t distribution the p -value will be found.

TABLE 4.4 How to calculate p -value

Form of Hypothesis Test	p -Value
Left-tailed test $H_0 : \mu \geq \mu_0$ vs. $H_a : \mu < \mu_0$	$P(t < t_{\text{data}})$
Right-tailed test $H_0 : \mu \leq \mu_0$ vs. $H_a : \mu > \mu_0$	$P(t > t_{\text{data}})$
Two-tailed test $H_0 : \mu = \mu_0$ vs. $H_a : \mu \neq \mu_0$	If $t_{\text{data}} < 0$ then $p\text{-value} = 2 \cdot P(t < t_{\text{data}})$ If $t_{\text{data}} > 0$ then $p\text{-value} = 2 \cdot P(t > t_{\text{data}})$

A small p -value will indicate conflict between the data and the null hypothesis. Thus we will reject H_0 if the p -value is small. How small is small? Since researchers set the level of significance α at some small value (such as 0.05), we consider the p -value to be small if it is less than α . This leads us to the rejection rule:

Reject H_0 if the p -value is $< \alpha$.

For example, recall our subgroup of customers who have both the International Plan and the VoiceMail Plan and who have more than 220 day minutes. Suppose we would like to test whether the mean number of customer service calls of all such customers differs from 2.4, and we set the level of significance α to be 0.05. We would have a two-tailed hypothesis test:

$$H_0 : \mu = 2.4 \text{ vs. } H_a : \mu \neq 2.4$$

The null hypothesis will be rejected if the p -value is < 0.05 . Here, we have $\mu_0 = 2.4$, and earlier we saw that $\bar{x} = 1.607$, $s = 1.892$, and $n = 28$. Thus,

$$t_{\text{data}} = \frac{\bar{x} - \mu_0}{s/\sqrt{n}} = \frac{1.607 - 2.4}{1.892/\sqrt{28}} = -2.2178$$

Since $t_{\text{data}} < 0$, we have

$$p\text{-value} = 2 \cdot P(t < t_{\text{data}}) = 2 \cdot P(t < -2.2178) = 2 \cdot 0.01758 = 0.035$$

Since the p -value of 0.035 is less than the level of significance $\alpha = 0.05$, we reject H_0 . The interpretation of this conclusion is that there is evidence at level of significance $\alpha = 0.05$ that the population mean number of customer service calls of all such customers differs from 2.4. Had we not rejected H_0 , we could simply insert the word “insufficient” before “evidence” in the previous sentence.

4.9 ASSESSING THE STRENGTH OF EVIDENCE AGAINST THE NULL HYPOTHESIS

However, there is nothing written in stone saying that the level of significance α must be 0.05. What if we had chosen $\alpha = 0.01$ in this example? Then the p -value 0.035 would not have been less than $\alpha = 0.01$, and we would not have rejected H_0 . Note that *the hypotheses have not changed and the data have not changed, but the conclusion has been reversed simply by changing the value of α .*

Further, consider that hypothesis testing restricts us to a simple “yes-or-no” decision: either to reject H_0 or not reject H_0 . But this dichotomous conclusion provides no indication of the strength of evidence against the null hypothesis residing in the data. For example, for level of significance $\alpha = 0.05$, one set of data may return a p -value of 0.06 while another set of data provides a p -value of 0.96. Both p -values lead to the same conclusion—do not reject H_0 . However, the first data set came close to rejecting H_0 , and shows a fair amount of evidence against the null hypothesis, while the second data set shows no evidence at all against the null hypothesis. A simple “yes-or-no” decision misses the distinction between these two scenarios. The p -value provides extra information that a dichotomous conclusion does not take advantage of.

TABLE 4.5 Strength of evidence against H_0 for various p -values

p -Value	Strength of Evidence Against H_0
$p\text{-value} \leq 0.001$	Extremely strong evidence
$0.001 < p\text{-value} \leq 0.01$	Very strong evidence
$0.01 < p\text{-value} \leq 0.05$	Solid evidence
$0.05 < p\text{-value} \leq 0.10$	Mild evidence
$0.10 < p\text{-value} \leq 0.15$	Slight evidence
$0.15 < p\text{-value}$	No evidence

Some data analysts do not think in terms of whether or not to reject the null hypothesis so much as to *assess the strength of evidence against the null hypothesis*. Table 4.5 provides a thumbnail interpretation of the strength of evidence against H_0 for various p -values. For certain data domains, such as physics and chemistry, the interpretations may differ.

Thus, for the hypothesis test $H_0 : \mu = 2.4$ vs. $H_a : \mu \neq 2.4$, where the p -value equals 0.035, we would not provide a conclusion as to whether or not to reject H_0 . Instead, we would simply state that there is *solid evidence against the null hypothesis*.

4.10 USING CONFIDENCE INTERVALS TO PERFORM HYPOTHESIS TESTS

Did you know that one confidence interval is worth 1000 hypothesis tests? Because the t confidence interval and the t hypothesis test are both based on the same distribution with the same assumptions, we may state the following:

A $100(1 - \alpha)\%$ confidence interval for μ is equivalent to
a two-tailed hypothesis test for μ , with level of significance α .

Table 4.6 shows the equivalent confidence levels and levels of significance. The equivalency is stated as follows (see Figure 4.3):

- If a certain hypothesized value for μ_0 falls *outside* the confidence interval with confidence level $100(1 - \alpha)\%$, then the two-tailed hypothesis test with level of significance α will reject H_0 for that value of μ_0 .

TABLE 4.6 Confidence levels and levels of significance for equivalent confidence intervals and hypothesis tests

Confidence Level $100(1 - \alpha)\%$	Level of Significance α
90%	0.10
95%	0.05
99%	0.01

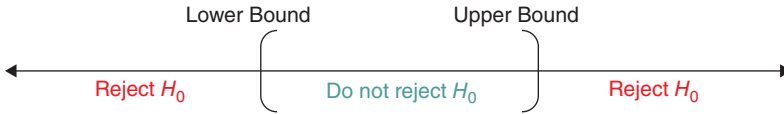


Figure 4.3 Reject values of μ_0 that would fall outside the equivalent confidence interval.

- If the hypothesized value for μ_0 falls *inside* the confidence interval with confidence level $100(1 - \alpha)\%$, then the two-tailed hypothesis test with level of significance α will not reject H_0 for that value of μ_0 .

For example, recall that our 95% confidence interval for the population mean number of customer service calls for all customers who have the International Plan and the VoiceMail plan and who have more than 220 minutes of day use is:

$$(\text{lower bound, upper bound}) = (0.875, 2.339)$$

We may use this confidence interval to test any number of possible values of μ_0 , as long as the test is two-tailed with level of significance $\alpha = 0.05$. For example, use level of significance $\alpha = 0.05$ to test whether the mean number of customer service calls for such customers differs from the following values:

- 0.5
- 1.0
- 2.4

The solution is as follows. We have the following hypothesis tests:

- $H_0 : \mu = 0.5$ vs. $H_a : \mu \neq 0.5$
- $H_0 : \mu = 1.0$ vs. $H_a : \mu \neq 1.0$
- $H_0 : \mu = 2.4$ vs. $H_a : \mu \neq 2.4$

We construct the 95% confidence interval, and place the hypothesized values of μ_0 on the number line, as shown in Figure 4.4.

Their placement in relation to the confidence interval allows us to immediately state the conclusion of the two-tailed hypothesis test with level of significance $\alpha = 0.05$, as shown in Table 4.7.

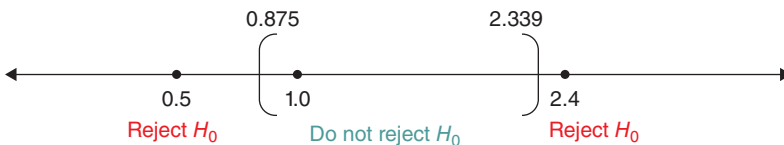


Figure 4.4 Placing the hypothesized values of μ_0 on the number line in relation to the confidence interval informs us immediately of the conclusion.

TABLE 4.7 Conclusions for three hypothesis tests using the confidence interval

μ_0	Hypotheses With $\alpha = 0.05$	Position in Relation to 95% Confidence Interval	Conclusion
0.5	$H_0 : \mu = 0.5$ vs. $H_a : \mu \neq 0.5$	Outside	Reject H_0
1.0	$H_0 : \mu = 1.0$ vs. $H_a : \mu \neq 1.0$	Inside	Do not reject H_0
2.4	$H_0 : \mu = 2.4$ vs. $H_a : \mu \neq 2.4$	Outside	Reject H_0

4.11 HYPOTHESIS TESTING FOR THE PROPORTION

Hypothesis tests may also be performed about the population proportion π . The test statistic is:

$$Z_{\text{data}} = \frac{p - \pi_0}{\sqrt{\frac{\pi_0(1 - \pi_0)}{n}}}$$

where π_0 is the hypothesized value of π , and p is the sample proportion

$$p = \frac{\text{number of successes}}{n}$$

The hypotheses and p -values are shown in Table 4.8.

For example, recall that 483 of 3333 customers in our sample had churned, so that an estimate of the population proportion π of all of the company’s customers who churn is

$$p = \frac{\text{number who churn}}{\text{sample size}} = \frac{x}{n} = \frac{483}{3333} = 0.1449$$

Suppose we would like to test using level of significance $\alpha = 0.10$ whether π differs from 0.15. The hypotheses are

$$H_0 : \pi = 0.15 \text{ vs. } H_a : \pi \neq 0.15$$

TABLE 4.8 Hypotheses and p -values for hypothesis tests about π

Hypotheses	p -Value
Left-tailed test $H_0 : \pi \geq \pi_0$ vs. $H_a : \pi < \pi_0$	$P(Z < Z_{\text{data}})$
Right-tailed test $H_0 : \pi \leq \pi_0$ vs. $H_a : \pi > \pi_0$	$P(Z > Z_{\text{data}})$
Two-tailed test $H_0 : \pi = \pi_0$ vs. $H_a : \pi \neq \pi_0$	If $Z_{\text{data}} < 0$ then $p\text{-value} = 2 \cdot P(Z < Z_{\text{data}})$ If $Z_{\text{data}} > 0$ then $p\text{-value} = 2 \cdot P(Z > Z_{\text{data}})$

The test statistic is

$$Z_{\text{data}} = \frac{p - \pi_0}{\sqrt{\frac{\pi_0(1 - \pi_0)}{n}}} = \frac{0.1449 - 0.15}{\sqrt{\frac{0.15(0.85)}{3333}}} = -0.8246$$

Since $Z_{\text{data}} < 0$ the p -value $= 2 \cdot P(Z < Z_{\text{data}}) = 2 \cdot P(Z < -0.8246) = 2 \cdot 0.2048 = 0.4096$.

Since the p -value is not less than $\alpha = 0.10$ we would not reject H_0 . There is insufficient evidence that the proportion of all our customers who churn differs from 15%. Further, assessing the strength of evidence against the null hypothesis using Table 4.5 would lead us to state that there is no evidence against H_0 . Also, given a confidence interval, we may perform two-tailed hypothesis tests for π , just as we did for μ .

THE R ZONE

Input the data set Churn

```
churn <- read.csv(file = "C:/.../churn.txt",
  stringsAsFactors=TRUE)
# Show the variables in Churn
names(churn)
# Show the first ten values of Int'l.Plan
churn$Int'l.Plan[1:10]
# Show the first ten values of VMail Plan
churn$VMail.Plan[1:10]
```

```
> names(churn)
 [1] "State"          "Account.Length" "Area.Code"
 [4] "Phone"          "Int'l.Plan"     "VMail.Plan"
 [7] "vmail.Message"  "Day.Mins"       "Day.Calls"
[10] "Day.charge"     "Eve.Mins"       "Eve.Calls"
[13] "Eve.charge"     "Night.Mins"     "Night.Calls"
[16] "Night.charge"   "Intl.Mins"      "Intl.Calls"
[19] "Intl.charge"    "CustServ.Calls" "churn"
> churn$Int'l.Plan[1:10]
 [1] no no no yes yes yes no yes no yes
Levels: no yes
> churn$VMail.Plan[1:10]
 [1] yes yes no no no no yes no no yes
Levels: no yes
```

Find a subgroup

```
# Customers who have Int'l and VMail
# plans, and more than 220 day minutes
subchurn <- subset(churn,
  churn$Int'l.Plan == "yes" &
  churn$VMail.Plan == "yes" &
  churn$Day.Mins > 220)
subchurn
summary(subchurn$CustServ.Calls)
length(subchurn$CustServ.Calls)
```

```
> summary(subchurn$CustServ.Calls)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.000  0.750   1.000   1.607  2.000   9.000
> length(subchurn$CustServ.Calls)
 [1] 28
```

One Sample T-test and Confidence Interval for Mean

```
mean.test <- t.test(x= subchurn$CustServ.Calls,
  mu=2.4,
  conf.level= 0.95)
mean.test$statistic
mean.test$p.value
mean.test$conf.int
```

```
> mean.test$statistic
      t
-2.217128
> mean.test$p.value
[1] 0.03522289
> mean.test$conf.int
[1] 0.8733969 2.3408888
attr(,"conf.level")
[1] 0.95
```

One sample Proportion Test and Confidence Interval

```
# Show possible levels of Churn variable
levels(churn$Churn)
# Find how many customers churned
num.churn <- sum(churn$Churn == "True")
# Find sample size, Calculate p, Z_data
sample.size <- dim(churn)[1]
p <- num.churn/sample.size
Z_data <- (p - 0.15) / sqrt((0.15*(1-0.15))/sample.size)
# Find confidence interval, p-value of Z_data
error <- qnorm(0.975, mean = 0, sd = 1)*
  sqrt((p*(1-p))/sample.size)
lower.bound <- p - error
upper.bound <- p + error
p.value <- 2*pnorm(Z_data, mean = 0, sd = 1)
Z_data; p.value
lower.bound; upper.bound
```

```
> Z_data; p.value
[1] -0.8222369
[1] 0.4109421
> lower.bound; upper.bound
[1] 0.1329639
[1] 0.1568651
```

REFERENCE

1. Much more information regarding the topics covered in this chapter may be found in any introductory statistics textbook, such as *Discovering Statistics*, by Daniel T. Larose, second edition, published by W. H. Freeman, New York, 2013.

EXERCISES

1. Explain what is meant by statistical inference. Give an example of statistical inference from everyday life, say, a political poll.
2. What is the difference between a population and a sample?
3. Describe the difference between a parameter and a statistic.
4. When should statistical inference not be applied?
5. What is the difference between point estimation and confidence interval estimation?

6. Discuss the relationship between the width of a confidence interval and the confidence level associated with it.
7. Discuss the relationship between the sample size and the width of a confidence interval. Which is better, a wide interval or a tight interval? Why?
8. Explain what we mean by sampling error.
9. What is the meaning of the term *margin of error*?
10. What are the two ways to reduce margin of error, and what is the recommended way?
11. A political poll has a margin of error of 3%. How do we interpret this number?
12. What is hypothesis testing?
13. Describe the two ways a correct conclusion can be made, and the two ways an incorrect conclusion can be made.
14. Explain clearly why a small p -value leads to rejection of the null hypothesis.
15. Explain why it may not always be desirable to draw a black-and-white, up-or-down conclusion in a hypothesis test. What can we do instead?
16. How can we use a confidence interval to conduct hypothesis tests?
17. The duration customer service calls to an insurance company is normally distributed, with mean 20 minutes, and standard deviation 5 minutes. For the following sample sizes, construct a 95% confidence interval for the population mean duration of customer service calls.
 - a. $n = 25$
 - b. $n = 100$
 - c. $n = 400$
18. For each of the confidence intervals in the previous exercise, calculate and interpret the margin of error.
19. Refer to the previous exercise. Describe the relationship between margin of error and sample size.
20. Of 1000 customers who received promotional materials for a marketing campaign, 100 responded to the promotion. For the following confidence levels, construct a confidence interval for the population proportion who would respond to the promotion.
 - a. 90%
 - b. 95%
 - c. 99%
21. For each of the confidence intervals in the previous exercise, calculate and interpret the margin of error.
22. Refer to the previous exercise. Describe the relationship between margin of error and confidence level.
23. A sample of 100 donors to a charity has a mean donation amount of \$55 with a sample standard deviation of \$25. Test using $\alpha = 0.05$ whether the population mean donation amount exceeds \$50.
 - a. Provide the hypotheses. State the meaning of μ .
 - b. What is the rejection rule?

- c. What is the meaning of the test statistic t_{data} ?
 - d. Is the value of the test statistic t_{data} extreme? How can we tell?
What is the meaning of the p -value in this example?
 - e. What is our conclusion?
 - f. Interpret our conclusion so that a nonspecialist could understand it.
24. Refer to the hypothesis test in the previous exercise. Suppose we now set $\alpha = 0.01$.
- a. What would our conclusion now be? Interpret this conclusion.
 - b. Note that the conclusion has been reversed simply because we have changed the value of α . But have the data changed? No, simply our level of what we consider to be significance. Instead go ahead and assess the strength of evidence against the null hypothesis.
25. Refer to the first confidence interval you calculated for the population mean duration of customer service calls. Use this confidence interval to test whether this population mean differs from the following values, using level of significance $\alpha = 0.05$.
- a. 15 minutes
 - b. 20 minutes
 - c. 25 minutes
26. In a sample of 100 customers, 240 churned when the company raised rates. Test whether the population proportion of churners is less than 25%, using level of significance $\alpha = 0.01$. ■

MULTIVARIATE STATISTICS

- 5.1** TWO-SAMPLE t -TEST FOR DIFFERENCE IN MEANS **110**
- 5.2** TWO-SAMPLE Z-TEST FOR DIFFERENCE IN PROPORTIONS **111**
- 5.3** TEST FOR HOMOGENEITY OF PROPORTIONS **112**
- 5.4** CHI-SQUARE TEST FOR GOODNESS OF FIT OF MULTINOMIAL DATA **114**
- 5.5** ANALYSIS OF VARIANCE **115**
- 5.6** REGRESSION ANALYSIS **118**
- 5.7** HYPOTHESIS TESTING IN REGRESSION **122**
- 5.8** MEASURING THE QUALITY OF A REGRESSION MODEL **123**
- 5.9** DANGERS OF EXTRAPOLATION **123**
- 5.10** CONFIDENCE INTERVALS FOR THE MEAN VALUE OF y GIVEN x **125**
- 5.11** PREDICTION INTERVALS FOR A RANDOMLY CHOSEN VALUE OF y GIVEN x **125**
- 5.12** MULTIPLE REGRESSION **126**
- 5.13** VERIFYING MODEL ASSUMPTIONS **127**
 - THE R ZONE **131**
 - REFERENCE **135**
 - EXERCISES **135**
 - HANDS-ON ANALYSIS **136**

So far we have discussed inference methods for one variable at a time. Data analysts are also interested in multivariate inferential methods, where the relationships between two variables, or between one target variable and a set of predictor variables, are analyzed.

We begin with bivariate analysis, where we have two independent samples and wish to test for significant differences in the means or proportions of the two samples. When would data miners be interested in using bivariate analysis? In Chapter 6, we

illustrate how the data is partitioned into a training data set and a test data set for cross-validation purposes. Data miners can use the hypothesis tests shown here to determine whether significant differences exist between the means of various variables in the training and test data sets. If such differences exist, then the cross-validation is invalid, because the training data set is nonrepresentative of the test data set.

- For a continuous variable, use the *two-sample t-test for the difference in means*.
- For a flag variable, use the *two-sample Z-test for the difference in proportions*.
- For a multinomial variable, use the *test for the homogeneity of proportions*.

Of course, there are presumably many variables in each of the training set and test set. However, spot-checking of a few randomly chosen variables is usually sufficient.

5.1 TWO-SAMPLE *t*-TEST FOR DIFFERENCE IN MEANS

To test for the difference in population means, we use the following test statistic,

$$t_{\text{data}} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

which follows an approximate *t* distribution with degrees of freedom the smaller of $n_1 - 1$ and $n_2 - 1$, whenever either both populations are normally distributed or both samples are large.

For example, we partitioned the churn data set into a training set of 2529 records and a test set of 804 records (the reader’s partition will differ). We would like to assess the validity of the partition by testing whether the population mean number of customer service calls differs between the two data sets. The summary statistics are given in Table 5.1.

Now, the sample means do not look very different, but we would like to have the results of the hypothesis test just to make sure. The hypotheses are:

$$H_0 : \mu_1 = \mu_2 \quad \text{vs.} \quad H_a : \mu_1 \neq \mu_2$$

The test statistic is:

$$t_{\text{data}} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} = \frac{1.5714 - 1.5361}{\sqrt{\frac{1.3126^2}{2529} + \frac{1.3251^2}{804}}} = 0.6595$$

TABLE 5.1 Summary statistics for customer service calls, training data set and test data set

Data Set	Sample Mean	Sample Standard Deviation	Sample Size
Training set	$\bar{x}_1 = 1.5714$	$s_1 = 1.3126$	$n_1 = 2529$
Test set	$\bar{x}_2 = 1.5361$	$s_2 = 1.3251$	$n_2 = 804$

The two-tailed p -value for $t_{\text{data}} = 0.6594$ is:

$$p\text{-value} = 2 \cdot P(t > 0.6595) = 0.5098$$

Since the p -value is large, there is no evidence that the mean number of customer service calls differs between the training data set and the test data set. For this variable at least, the partition seems valid.

5.2 TWO-SAMPLE Z-TEST FOR DIFFERENCE IN PROPORTIONS

Of course not all variables are numeric, like customer service calls. What if we have a 0/1 flag variable—such as membership in the VoiceMail Plan—and wish to test whether the proportions of records with value 1 differ between the training data set and test data set? We could turn to the two-sample Z-test for the difference in proportions. The test statistic is

$$Z_{\text{data}} = \frac{p_1 - p_2}{\sqrt{p_{\text{pooled}} \cdot (1 - p_{\text{pooled}}) \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}}$$

where $p_{\text{pooled}} = \frac{x_1 + x_2}{n_1 + n_2}$, and x_i and p_i represents the number of and proportion of records with value 1 for sample i , respectively.

For example, our partition resulted in $x_1 = 707$ of $n_1 = 2529$ customers in the training set belonging to the VoiceMail Plan, while $x_2 = 215$ of $n_2 = 804$ customers in the test set belonging, so that $p_1 = \frac{x_1}{n_1} = \frac{707}{2529} = 0.2796$, $p_2 = \frac{x_2}{n_2} = \frac{215}{804} = 0.2674$,

and $p_{\text{pooled}} = \frac{x_1 + x_2}{n_1 + n_2} = \frac{707 + 215}{2529 + 804} = 0.2766$.

The hypotheses are

$$H_0 : \pi_1 = \pi_2 \quad \text{vs.} \quad H_a : \pi_1 \neq \pi_2$$

The test statistic is

$$\begin{aligned} Z_{\text{data}} &= \frac{p_1 - p_2}{\sqrt{p_{\text{pooled}} \cdot (1 - p_{\text{pooled}}) \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}} \\ &= \frac{0.2796 - 0.2674}{\sqrt{0.2766 \cdot (0.7234) \left(\frac{1}{2529} + \frac{1}{804} \right)}} = 0.6736 \end{aligned}$$

The p -value is

$$p\text{-value} = 2 \cdot P(Z > 0.6736) = 0.5006$$

Thus there is no evidence that the proportion of VoiceMail Plan members differs between the training and test data sets. For this variable, the partition is valid.

5.3 TEST FOR HOMOGENEITY OF PROPORTIONS

Multinomial data is an extension of binomial data to $k > 2$ categories. For example, suppose a multinomial variable *marital status* takes the values *married*, *single*, and *other*. Suppose we have a training set of 1000 people and a test set of 250 people, with the frequencies shown in Table 5.2.

To determine whether significant differences exist between the multinomial proportions of the two data sets, we could turn to the test for the homogeneity of proportions.¹ The hypotheses are

$$\begin{aligned} H_0 : & p_{\text{married,training}} = p_{\text{married,test}}, \\ & p_{\text{single,training}} = p_{\text{single,test}}, \\ & p_{\text{other,training}} = p_{\text{other,test}} \\ H_a : & \text{At least one of the claims in } H_0 \text{ is wrong.} \end{aligned}$$

To determine whether these *observed frequencies* represent proportions that are significantly different for the training and test data sets, we compare these observed frequencies with the *expected frequencies* that we would expect if H_0 were true. For example, to find the expected frequency for the number of married people in the training set, we (a) find the overall proportion of married people in both the training and test sets, $\frac{505}{1250}$, and (b) we multiply this overall proportion by the number of people in the training set, 1000, giving us the expected proportion of married people in the training set to be

$$\text{Expected frequency}_{\text{married,training}} = \frac{(1000)(505)}{1250} = 404$$

We use the overall proportion in (a) because H_0 states that the training and test proportions are equal. Generalizing, for each cell in the table, the expected frequencies are calculated as follows:

$$\text{Expected frequency} = \frac{(\text{row total})(\text{column total})}{\text{grand total}}$$

Applying this formula to each cell in the table gives us the table of expected frequencies in Table 5.3.

TABLE 5.2 Observed frequencies

Data Set	Married	Single	Other	Total
Training set	410	340	250	1000
Test set	95	85	70	250
Total	505	425	320	1250

¹Thanks to Dr. Daniel S. Miller for helpful discussions on this topic.

TABLE 5.3 Expected frequencies

Data Set	Married	Single	Other	Total
Training set	404	340	256	1000
Test set	101	85	64	250
Total	505	425	320	1250

The observed frequencies (O) and the expected frequencies (E) are compared using a test statistic from the χ^2 (chi-square) distribution:

$$\chi_{\text{data}}^2 = \sum \frac{(O - E)^2}{E}$$

Large differences between the observed and expected frequencies, and thus a large value for χ_{data}^2 , will lead to a small p -value, and a rejection of the null hypothesis. Table 5.4 illustrates how the test statistic is calculated.

The p -value is the area to the right of χ_{data}^2 under the χ^2 curve with degrees of freedom equal to $(\text{number of rows} - 1)(\text{number of columns} - 1) = (1)(2) = 2$:

$$p\text{-value} = P(\chi^2 > \chi_{\text{data}}^2) = P(\chi^2 > 1.15) = 0.5627$$

Because this p -value is large, there is no evidence that the observed frequencies represent proportions that are significantly different for the training and test data sets. In other words, for this variable, the partition is valid.

This concludes our coverage of the tests to apply when checking the validity of a partition.

TABLE 5.4 Calculating the test statistic χ_{data}^2

Cell	Observed Frequency	Expected Frequency	$\frac{(\text{Obs} - \text{Exp})^2}{\text{Exp}}$
Married, training	410	404	$\frac{(410 - 404)^2}{404} = 0.09$
Married, test	95	101	$\frac{(95 - 101)^2}{101} = 0.36$
Single, training	340	340	$\frac{(340 - 340)^2}{340} = 0$
Single, test	85	85	$\frac{(85 - 85)^2}{85} = 0$
Other, training	250	256	$\frac{(250 - 256)^2}{256} = 0.14$
Other, test	70	64	$\frac{(70 - 64)^2}{64} = 0.56$
			$\chi_{\text{data}}^2 = 1.15$

5.4 CHI-SQUARE TEST FOR GOODNESS OF FIT OF MULTINOMIAL DATA

Next, suppose a multinomial variable *marital status* takes the values *married*, *single*, and *other*, and suppose that we know that 40% of the individuals in the *population* are married, 35% are single, and 25% report another marital status. We are taking a sample and would like to determine whether the sample is representative of the population. We could turn to the χ^2 (chi-square) goodness of fit test.

The hypotheses for this χ^2 goodness of fit test would be as follows:

$$H_0 : p_{\text{married}} = 0.40, \quad p_{\text{single}} = 0.35, \quad p_{\text{other}} = 0.25$$

$$H_a : \text{At least one of the proportions in } H_0 \text{ is wrong.}$$

Our sample of size $n = 100$ yields the following *observed frequencies* (represented by the letter “O”):

$$O_{\text{married}} = 36, O_{\text{single}} = 35, O_{\text{other}} = 29$$

To determine whether these counts represent proportions that are significantly different from those expressed in H_0 , we compare these observed frequencies with the *expected frequencies* that we would expect if H_0 were true. If H_0 were true, then we would expect 40% of our sample of 100 individuals to be married, that is, the expected frequency for *married* is

$$E_{\text{married}} = n \cdot p_{\text{married}} = 100 \cdot 0.40 = 40$$

Similarly,

$$E_{\text{single}} = n \cdot p_{\text{single}} = 100 \cdot 0.35 = 35$$

$$E_{\text{other}} = n \cdot p_{\text{other}} = 100 \cdot 0.25 = 25$$

These frequencies are compared using the test statistic:

$$\chi^2_{\text{data}} = \sum \frac{(O - E)^2}{E}$$

Again, large differences between the observed and expected frequencies, and thus a large value for χ^2_{data} , will lead to a small p -value, and a rejection of the null hypothesis. Table 5.5 illustrates how the test statistic is calculated.

The p -value is the area to the right of χ^2_{data} under the χ^2 curve with $k - 1$ degrees of freedom, where k = the number of categories (here $k = 3$):

$$p\text{-value} = P(\chi^2 > \chi^2_{\text{data}}) = P(\chi^2 > 1.04) = 0.5945$$

Thus, there is no evidence that the observed frequencies represent proportions that differ significantly from those in the null hypothesis. In other words, our sample is representative of the population.

TABLE 5.5 Calculating the test statistic χ^2_{data}

Marital Status	Observed Frequency	Expected Frequency	$\frac{(\text{Obs} - \text{Exp})^2}{\text{Exp}}$
Married	36	40	$\frac{(36 - 40)^2}{40} = 0.4$
Single	35	35	$\frac{(35 - 35)^2}{35} = 0$
Other	29	25	$\frac{(29 - 25)^2}{25} = 0.64$
			$\chi^2_{\text{data}} = 1.04$

5.5 ANALYSIS OF VARIANCE

In an extension of the situation for the two-sample t test, suppose that we have a threefold partition of the data set, and wish to test whether the mean value of a continuous variable is the same across all three subsets. We could turn to one-way analysis of variance (ANOVA). To understand how ANOVA works, consider the following small example. We have samples from Groups A, B, and C, of four observations each, for the continuous variable *age*, shown in Table 5.6.

The hypotheses are

$$H_0 : \mu_A = \mu_B = \mu_c$$

$$H_a : \text{Not all the population means are equal.}$$

The sample mean ages are $\bar{x}_A = 45$, $\bar{x}_B = 40$, and $\bar{x}_C = 35$. A comparison dotplot of the data (Figure 5.1) shows that there is a considerable amount of overlap among the three data sets. So, despite the difference in sample means, the dotplot

TABLE 5.6 Sample ages for Groups A, B, and C

Group A	Group B	Group C
30	25	25
40	30	30
50	50	40
60	55	45

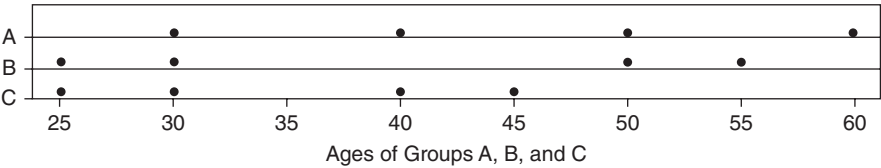


Figure 5.1 Dotplot of groups A, B, and C shows considerable overlap.

offers little or no evidence to reject the null hypothesis that the population means are all equal.

Next, consider the following samples from Groups D, E, and F, for the continuous variable *age*, shown in Table 5.7.

Once again, the sample mean ages are $\bar{x}_D = 45$, $\bar{x}_E = 40$, and $\bar{x}_F = 35$. A comparison dotplot of these data (Figure 5.2) illustrates that there is very little overlap among the three data sets. Thus, Figure 5.2 offers good evidence to reject the null hypothesis that the population means are all equal.

To recapitulate, Figure 5.1 shows no evidence of difference in group means, while Figure 5.2 shows good evidence of difference in group means, *even though the respective sample means are the same in both cases*. The distinction stems from the overlap among the groups, which itself is a result of the *spread* within each group. Note that the spread is large for each group in Figure 5.1, and small for each group in Figure 5.2. When the spread within each sample is large (Figure 5.1), the difference in sample means seems small. When the spread within each sample is small (Figure 5.2), the difference in sample means seems large.

ANOVA works by performing the following comparison. Compare

- 1. The *between-sample variability*, that is, the variability in the sample means, such as $\bar{x}_A = 45$, $\bar{x}_B = 40$, and $\bar{x}_C = 35$, with
- 2. The *within-sample variability*, that is, the variability within each sample, measured for example by the sample standard deviations.

When (1) is much larger than (2), this represents evidence that the population means are not equal. Thus, the analysis depends on measuring variability, hence the term *analysis of variance*.

Let \bar{x} represent the overall sample mean, that is, the mean of all observations from all groups. We measure the between-sample variability by finding the variance

TABLE 5.7 Sample ages for Groups D, E, and F

Group D	Group E	Group F
43	37	34
45	40	35
45	40	35
47	43	36

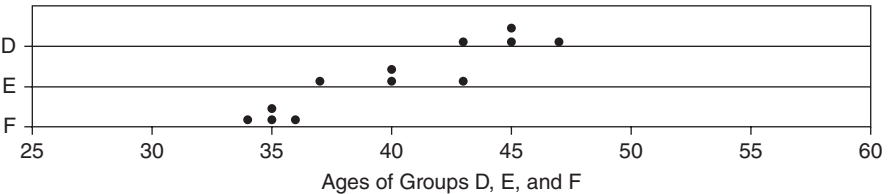


Figure 5.2 Dotplot of Groups D, E, and F shows little overlap.

of the k sample means, weighted by sample size, and expressed as the *mean square treatment* (MSTR):

$$\text{MSTR} = \frac{\sum n_i(\bar{x}_i - \bar{\bar{x}})^2}{k - 1}$$

We measure the within-sample variability by finding the weighted mean of the sample variances, expressed as the *mean square error* (MSE):

$$\text{MSE} = \frac{\sum (n_i - 1)s_i^2}{n_t - k}$$

We compare these two quantities by taking their ratio:

$$F_{\text{data}} = \frac{\text{MSTR}}{\text{MSE}}$$

which follows an F distribution, with degrees of freedom $df_1 = k - 1$ and $df_2 = n_t - k$. The numerator of MSTR is the *sum of squares treatment*, SSTR, and the numerator of MSE is the *sum of squares error*, SSE. The total sum of squares (SST) is the sum of SSTR and SSE. A convenient way to display the above quantities is in the ANOVA table, shown in Table 5.8.

The test statistic F_{data} will be large when the between-sample variability is much greater than the within-sample variability, which is indicative of a situation calling for rejection of the null hypothesis. The p -value is $P(F > F_{\text{data}})$; reject the null hypothesis when the p -value is small, which happens when F_{data} is large.

For example, let us verify our claim that Figure 5.1 showed little or no evidence that the population means were not equal. Figure 5.3 shows the Minitab ANOVA results.

The p -value of 0.548 indicates that there is no evidence against the null hypothesis that all population means are equal. This bears out our earlier claim. Next let us verify our claim that Figure 5.2 showed evidence that the population means were not equal. Figure 5.4 shows the Minitab ANOVA results.

TABLE 5.8 ANOVA table

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F
Treatment	SSTR	$df_1 = k - 1$	$\text{MSTR} = \frac{\text{SSTR}}{df_1}$	$F_{\text{data}} = \frac{\text{MSTR}}{\text{MSE}}$
Error	SSE	$df_2 = n_t - k$	$\text{MSE} = \frac{\text{MSE}}{df_2}$	
Total	SST			

Source	DF	SS	MS	F	P
Factor	2	200	100	0.64	0.548
Error	9	1400	156		
Total	11	1600			

Figure 5.3 ANOVA results for $H_0 : \mu_A = \mu_B = \mu_C$.

Source	DF	SS	MS	F	P
Factor	2	200.00	100.00	32.14	0.000
Error	9	28.00	3.11		
Total	11	228.00			

Figure 5.4 ANOVA results for $H_0 : \mu_D = \mu_E = \mu_F$.

The p -value of approximately zero indicates that there is strong evidence that not all the population mean ages are equal, thus supporting our earlier claim.

5.6 REGRESSION ANALYSIS

To help us learn about regression methods for estimation and prediction, let us get acquainted with a new data set, *cereals*. The *cereals* data set, included at the book series website courtesy of the *Data and Story Library* [1], contains nutrition information for 77 breakfast cereals and includes the following variables:

- Cereal name
- Cereal manufacturer
- Type (hot or cold)
- Calories per serving
- Grams of protein
- Grams of fat
- Milligrams of sodium
- Grams of fiber
- Grams of carbohydrates
- Grams of sugars
- Milligrams of potassium
- Percentage of recommended daily allowance of vitamins (0%, 25%, or 100%)
- Weight of one serving
- Number of cups per serving
- Shelf location (1 = bottom, 2 = middle, 3 = top)
- Nutritional rating, calculated by *Consumer Reports*

Table 5.9 provides a peek at eight of these fields for the first six cereals. We are interested in estimating the nutritional *rating* of a cereal given its *sugar* content.

It is important to note that this data set contains some missing data. The following four field values are missing:

- Potassium content of Almond Delight
- Potassium content of Cream of Wheat
- Carbohydrates and sugars content of Quaker Oatmeal

TABLE 5.9 Excerpt from *cereals* data set: eight fields, first six cereals

Cereal Name	Manuf.	Sugars	Calories	Protein	Fat	Sodium	Rating
100% Bran	N	6	70	4	1	130	68.4030
100% Natural Bran	Q	8	120	3	5	15	33.9837
All-Bran	K	5	70	4	1	260	59.4255
All-Bran Extra Fiber	K	0	50	4	0	140	93.7049
Almond Delight	R	8	110	2	2	200	34.3848
Apple Cinnamon Cheerios	G	10	110	2	2	180	29.5095
:	:	:	:	:	:	:	:

We shall therefore not be able to use the sugar content of Quaker Oatmeal to help build the regression model for predicting nutrition rating. In Chapter 13, *Imputation of Missing Data*, we shall learn how to apply multiple regression to impute these four missing values.

Figure 5.5 shows a scatter plot of the nutritional rating versus the sugar content for the cereals, along with the least-squares regression line.

The regression line is written in the form $\hat{y} = b_0 + b_1x$, called the *regression equation*, where:

- \hat{y} is the estimated value of the response variable
- b_0 is the *y-intercept* of the regression line
- b_1 is the *slope* of the regression line
- b_0 and b_1 , together, are called the *regression coefficients*

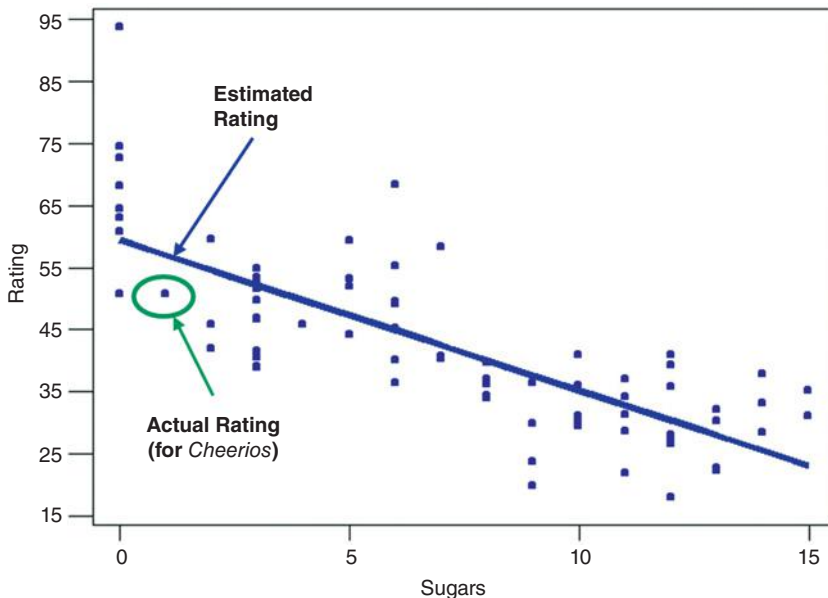


Figure 5.5 Scatter plot of nutritional rating versus sugar content for 76 cereals.

The regression results are shown in Figure 5.6. At the top we see, “The regression equation is: Rating = 59.9 – 2.46 sugars”. These are rounded values for the regression coefficients. Next comes a note that only 76 cereals were used (because the sugar information for Quaker Oats is missing). Then, below “Coef”, are presented more accurate values for the regression coefficients. We see that b_0 (the coefficient for constant) is 59.853, and b_1 (the coefficient for sugars) is –2.4614. Thus regression

The regression equation is
Rating = 59.9 - 2.46 Sugars

76 cases used, 1 cases contain missing values

Predictor	Coef	SE Coef	T	P
Constant	59.853	1.998	29.96	0.000
Sugars	-2.4614	0.2417	-10.18	0.000

S = 9.16616 R-Sq = 58.4% R-Sq(adj) = 57.8%

Analysis of Variance

Source	DF	SS	MS	F	P
Regression	1	8711.9	8711.9	103.69	0.000
Residual Error	74	6217.4	84.0		
Total	75	14929.3			

Unusual Observations

Obs	Sugars	Rating	Fit	SE Fit	Residual	St Resid
1	6.0	68.40	45.08	1.08	23.32	2.56R
4	0.0	93.70	59.85	2.00	33.85	3.78R

R denotes an observation with a large standardized residual.

Predicted Values for New Observations

New Obs	Fit	SE Fit	95% CI	95% PI
1	57.39	1.80	(53.81, 60.97)	(38.78, 76.00)

Values of Predictors for New Observations

New Obs	Sugars
1	1.00

Figure 5.6 Regression results for using *sugars* to estimate *rating*.

equation is given as $\hat{y} = 59.853 - 2.4614(\text{sugars})$. This estimated regression equation can then be interpreted as: “The estimated cereal rating equals 59.853 minus 2.4614 times the sugar content in grams.” The regression line and the regression equation are used as a *linear approximation* of the relationship between the x (predictor) and y (response) variables, that is, between sugar content and nutritional rating. We can then use the regression equation to make estimates or predictions.

For example, suppose that we are interested in estimating the nutritional rating for a new cereal (not in the original data) that contains $x = 1$ gram of sugar. Using the regression equation, we find the estimated nutritional rating for a cereal with 1 gram of sugar to be $\hat{y} = 59.853 - 2.4614(1) = 57.3916$. Note that this estimated value for the nutritional rating lies directly on the regression line, at the location $(x = 1, \hat{y} = 57.3916)$, as shown in Figure 5.5. In fact, for any given value of x (sugar content), the estimated value for y (nutritional rating) lies precisely on the regression line.

Now, there is one cereal in our data set that does have a sugar content of 1 gram, Cheerios. Its nutrition rating, however, is 50.765, not 57.3916 as we estimated above for the new cereal with 1 gram of sugar. Cheerios’ point in the scatter plot is located at $(x = 1, y = 50.765)$, within the oval in Figure 5.5. Now, the upper arrow in Figure 5.5 is pointing to a location on the regression line directly above the Cheerios point. This is where the regression equation predicted the nutrition rating to be for a cereal with a sugar content of 1 gram. The prediction was too low by $50.765 - 57.3916 = -6.6266$ rating points, which represents the vertical distance from the Cheerios data point to the regression line. This vertical distance, in general $(y - \hat{y})$, is known variously as the *prediction error*, *estimation error*, or *residual*. In Figure 5.6 we see that Minitab identifies two *unusual observations*, cereal 1 (100% Bran) and cereal 4 (All-Bran with Extra Fiber), which have large positive residuals, indicating that the nutrition rating was unexpectedly high, given their sugar level.

We, of course, seek to minimize the overall size of our prediction errors. *Least-squares regression* works by choosing the unique regression line that minimizes the sum of squared errors (SSE) over all the data points. There are alternative methods of choosing the line that best approximates the linear relationship between the variables, such as median regression, although least squares remain the most common method.

The y -intercept b_0 is the location on the y -axis where the regression line intercepts the y -axis, that is, the estimated value for the response variable when the predictor variable equals zero. Now, in many regression situations, a value of zero for the predictor variable would not make sense. For example, suppose that we were trying to predict elementary school student weight (y) based on student height (x). The meaning of *height* = 0 is unclear, so that the denotative meaning of the y -intercept would not make interpretive sense in this case. However, for our data set, a value of zero for the sugar content does make sense, as several cereals contain zero grams of sugar. Therefore, for our data set, the y -intercept $b_0 = 59.853$ simply represents the estimated nutritional rating for cereals with zero sugar content.

The slope of the regression line indicates the estimated change in y per unit increase in x . We interpret $b_1 = -2.4614$ to mean the following: “For each increase of 1 gram in sugar content, the estimated nutritional rating *decreases* by 2.4614 rating points.” For example, cereal A with 5 more grams of sugar than cereal B would have an estimated nutritional rating $5(2.4614) = 12.307$ rating points lower than cereal B.

5.7 HYPOTHESIS TESTING IN REGRESSION

Just as in univariate analysis we used the known value of the sample mean \bar{x} to perform inference for the unknown value of the population mean μ , so here in regression analysis, we will use the known value for the slope b_1 of the regression equation to perform inference for the unknown value of the slope β_1 of the population regression equation. The population regression equation represents a linear approximation of the relationship between, say, nutritional ratings and sugar content for the entire population of cereals, not just the cereals in our sample. The population regression equation looks like this:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

where ε represents a random variable for modeling the errors. Note that, when $\beta_1 = 0$, the population regression equation becomes:

$$y = \beta_0 + (0)x + \varepsilon = \beta_0 + \varepsilon$$

In other words, when $\beta_1 = 0$, there is no relationship between the predictor x and the response y . For any other value of β_1 , there is a linear relationship between x and y . Thus, if we wish to test for the existence of a linear relationship between x and y , we may simply perform the following hypothesis test:

$$H_0 : \beta_1 = 0 \text{ No relationship between } x \text{ and } y$$

vs.

$$H_a : \beta_1 \neq 0 \text{ Linear relationship between } x \text{ and } y$$

The test statistic for this hypothesis test is:

$$t = \frac{b_1}{s_{b_1}}$$

where s_{b_1} represents the standard error of the coefficient b_1 . For example, in Figure 5.6, one may find $s_{b_1} = 0.2417$ under *SE coef*. This is a measure of the variability in the slope of the regression line, observed from sample to sample. Large values for s_{b_1} indicate that there may be too much wiggle-factor (that is, variability) in our slope estimate b_1 , making precise inference regarding the slope difficult. This is reflected in the presence of s_{b_1} in the denominator of the t -statistic, so that large values of s_{b_1} tend to reduce the size of the t -statistic.

Figure 5.6 provides the value of the t -statistic for *sugars* as $t = -10.18$. This is obtained as the ratio of the slope b_1 and the standard error s_{b_1} :

$$t = \frac{b_1}{s_{b_1}} = \frac{-2.4614}{0.2417} = -10.18$$

The p -value for this test statistic is found in Figure 5.6 under P . The p -value indicates the probability of observing this value for t , if there really is no relationship between x and y . A small p -value (usually < 0.05) indicates that β_1 differs significantly from zero. Here we have p -value ≈ 0 , leading us to conclude that $\beta_1 \neq 0$, and that there exists a linear relationship between sugar content and nutritional rating.

5.8 MEASURING THE QUALITY OF A REGRESSION MODEL

How do we measure how helpful our regression model is? Clearly, if we cannot reject the null hypothesis in the hypothesis test above, then the regression model is no help at all. But if we do find that $\beta_1 \neq 0$, then there are two statistics that indicate the quality of our regression model. A very useful statistic is s , the *standard error of the estimate* (not to be confused with s , the sample standard deviation for univariate statistics, or s_{b_1} , the standard error of the slope coefficient).

$$s = \sqrt{\text{MSE}} = \sqrt{\text{SSE}/n - 2}$$

where SSE is the sum of squared errors. The statistic s is useful for assessing the quality of a regression model because its value indicates a measure of the size of the “typical” prediction error. For example, in Figure 5.6 we see that $s = 9.16616 \approx 9.2$. Thus, if we are given the sugar content (in grams) of a new cereal, our typical error in predicting nutritional rating will be about 9.2 points. Roughly (using the empirical rule), our estimate of the new cereal’s rating will be within 9.2 points about two-thirds of the time. This may be good enough for some applications. But it is not likely to be good enough for precise estimation of the nutritional rating. Thus, to reduce the value of s , we would need to add more predictors to the model (such as *sodium*, and so on), and use multiple regression.

Another measure of the quality of a regression model is the r^2 (“ r -squared”) statistic. r^2 measures how closely the linear regression fits the data, with values closer to 100% indicating a more perfect fit. The formula for r^2 is:

$$r^2 = \frac{\text{SSR}}{\text{SST}}$$

where SST represents the variability in the y -variable, and SSR represents the improvement in estimation from using the regression model as compared to just using \bar{y} to estimate y . Thus, r^2 may be interpreted as the ratio of the total variability in y that is accounted for by the linear relationship between x and y .

- r^2 is a measure of how closely the linear regression model fits the data, with values closer to 90–100% indicating a very nice fit.

The correlation coefficient r for *rating* and *sugars* is -0.764 (not shown), indicating that the nutritional rating and the sugar content are negatively correlated. It is not a coincidence that both r and b_1 are both negative. In fact, the correlation coefficient r and the regression slope b_1 always have the same sign. In Figure 5.6, we have $r^2 = 58.4\%$. Thus, 58.4% of the variability in nutritional rating is accounted for by the linear relationship between rating and sugars alone, without looking at other variables such as *sodium*.

5.9 DANGERS OF EXTRAPOLATION

Suppose that a new cereal (say, the Chocolate Frosted Sugar Bombs loved by Calvin, the comic strip character written by Bill Watterson) arrives on the market with a

very high sugar content of 30 grams per serving. Let us use our estimated regression equation to estimate the nutritional rating for Chocolate Frosted Sugar Bombs:

$$\hat{y} = 59.853 - 2.4614 (\text{sugars}) = 59.4 - 2.4614(30) = -13.989.$$

In other words, Calvin's cereal has so much sugar that its nutritional rating is actually a negative number, unlike any of the other cereals in the data set (minimum = 18) and analogous to a student receiving a negative grade on an exam. What is going on here?

The negative estimated nutritional rating for Chocolate Frosted Sugar Bombs is an example of the dangers of *extrapolation*. Analysts should confine the estimates and predictions made using the regression equation to values of the predictor variable contained within the range of the values of x in the data set. For example, in the *cereals* data set, the lowest sugar content is zero grams and the highest is 15 grams, so that predictions of nutritional rating for any value of x (sugar content) between zero and 15 grams would be appropriate. However, *extrapolation*, making predictions for x -values lying outside this range, can be dangerous, since we do not know the nature of the relationship between the response and predictor variables outside this range.

Extrapolation should be avoided if possible. If predictions outside the given range of x must be performed, the end user of the prediction needs to be informed that no x -data is available to support such a prediction. The danger lies in the possibility that the relationship between x and y , which may be linear within the range of x in the data set, may no longer be linear outside these bounds.

Consider Figure 5.7. Suppose that our data set consisted only of the data points in black but that the true relationship between x and y consisted of both the black (observed) and the gray (unobserved) points. Then, a regression line based solely

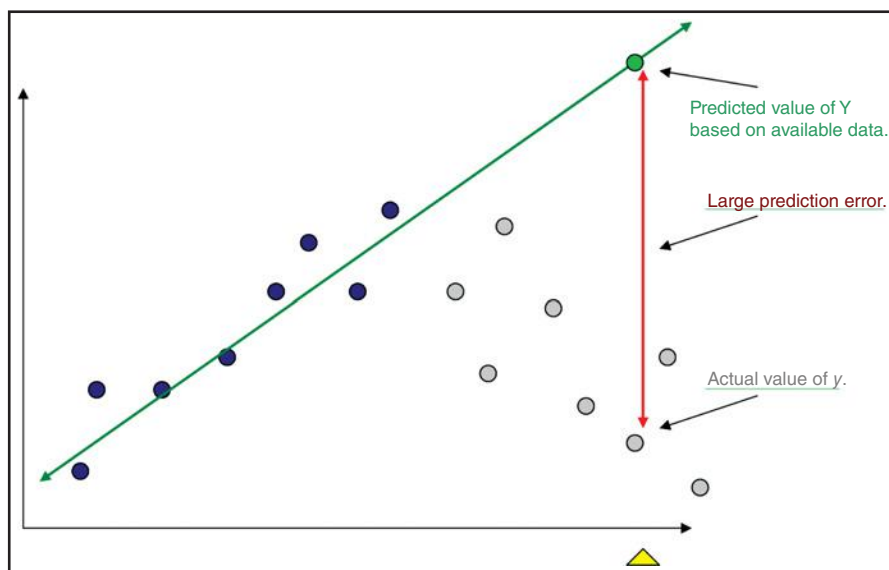


Figure 5.7 Dangers of extrapolation.

on the available (black dot) data would look approximately similar to the regression line indicated. Suppose that we were interested in predicting the value of y for an x -value located at the triangle. The prediction based on the available data would then be represented by the dot on the regression line indicated by the upper arrow. Clearly, this prediction has failed spectacularly, as shown by the vertical line indicating the huge prediction error. Of course, since the analyst would be completely unaware of the hidden data, he or she would hence be oblivious to the massive scope of the error in prediction. Policy recommendations based on such erroneous predictions could certainly have costly results.

5.10 CONFIDENCE INTERVALS FOR THE MEAN VALUE OF y GIVEN x

Thus far, we have discussed point estimates for values of the response variable for a given value of the predictor variable. Of course, point estimates in this context suffer the same drawbacks as point estimates in the univariate case, notably the lack of a probability statement associated with their accuracy. We may therefore turn to confidence intervals for the mean value of y for a given value of x .

The confidence interval for the mean value of y for a given value of x is as follows:

$$\text{point estimate} \pm \text{margin of error} = \hat{y}_p \pm t_{\alpha/2}(s) \sqrt{\frac{1}{n} + \frac{(x_p - \bar{x})^2}{\sum (x_i - \bar{x})^2}}$$

where

- x_p = the particular value of x for which the prediction is being made
- \hat{y}_p = the point estimate of y for a particular value of x
- $t_{\alpha/2}$ = a multiplier associated with the sample size and confidence level
- $s = \sqrt{\text{MSE}} = \sqrt{\text{SSE}/n - 1}$ = the standard error of the estimate
- SSE = the sum of squared residuals

We look at an example of this type of confidence interval below, but first we are introduced to a new type of interval, the prediction interval.

5.11 PREDICTION INTERVALS FOR A RANDOMLY CHOSEN VALUE OF y GIVEN x

Have you ever considered that it is “easier” to predict the mean value of a variable than it is to predict a randomly chosen value of that variable? For example, baseball buffs perusing the weekly batting average statistics will find that the team batting averages (which are the means of all the team’s players) are more closely bunched together than are the batting averages of the individual players. An estimate of the team batting average will therefore be more precise than an estimate of a randomly chosen member of that team for the same level of confidence.

Exam scores provide another example. It is not unusual for a randomly selected student's score to exceed 95, say, but it is quite unusual for the class average to be that high. This anecdotal evidence reflects the smaller variability associated with the mean (class average) of a variable rather than a randomly selected value (individual score) of that variable. Therefore, it is "easier" to predict the class average on an exam than it is to predict a randomly chosen student's score.

In many situations, data miners are more interested in predicting an individual value rather than the mean of all the values, given x . For example, an analyst may be more interested in predicting the credit score for a particular credit applicant rather than predicting the mean credit score of all similar applicants. Or, a geneticist may be interested in the expression of a particular gene rather than the mean expression of all similar genes.

Prediction intervals are used to estimate the value of a randomly chosen value of y , given x . Clearly, this is a more difficult task than estimating the mean, resulting in intervals of greater width (lower precision) than confidence intervals for the mean with the same confidence level. The prediction interval for a randomly chosen value of y for a given value of x is as follows:

$$\text{point estimate} \pm \text{margin of error} = \hat{y}_p \pm t_{\alpha/2}(s) \sqrt{1 + \frac{1}{n} + \frac{(x_p - \bar{x})^2}{\sum (x_i - \bar{x})^2}}$$

Note that this formula is precisely the same as the formula for the confidence interval for the mean value of y , given x , except for the presence of the "1+" inside the square root. This ensures that the prediction interval is always wider than the analogous confidence interval.

The last half-dozen lines of Figure 5.6 indicate the results for our confidence intervals and prediction intervals for a new cereal containing 1 gram of sugar.

- *Fit* is nothing but the point estimate of the nutritional rating for a cereal with 1 gram of sugar: $\hat{y} = 59.853 - 2.4614(1) = 57.3916$.
- *SE fit* is a measure of the variability of the point estimate.
- The 95% confidence interval for the mean nutritional rating of all cereals containing 1 gram of sugar is (53.81, 60.97).
- The 95% prediction interval for the nutritional rating of a randomly chosen cereal containing 1 gram of sugar is (38.78, 76.00).

Note that as expected, the prediction interval is wider than the confidence interval, reflecting the greater challenge of estimating a particular y value rather than the mean y value for a given value of x .

5.12 MULTIPLE REGRESSION

Most data mining applications enjoy a wealth (indeed, a superfluity) of data, with some data sets including hundreds of variables, many of which may have a linear

relationship with the target (response) variable. *Multiple regression modeling* provides an elegant method of describing such relationships. Multiple regression models provide improved precision for estimation and prediction, analogous to the improved precision of regression estimates over univariate estimates.

To illustrate the use of multiple regression modeling using the *cereals* data set, we shall add the predictor *sodium* to the model, and observe whether the quality of the model has improved or not. The multiple regression equation for two predictors looks like this:

$$\hat{y} = b_0 + b_1x_1 + b_2x_2$$

Figure 5.8 shows the multiple regression results for predicting nutritional rating based on sugars and sodium.

From Figure 5.8 we have the multiple regression equation:

$$\hat{y} = 69.180 - 2.3944 (\text{sugars}) - 0.06057 (\text{sodium})$$

We have $b_2 = -0.06057$, which is interpreted as follows. For each additional milligram (mg) of sodium, the estimated decrease in nutritional rating is 0.06057 points, *when sugars are held constant*. The point estimate of nutritional rating for a cereal, like Cheerios, that has 1 gram of sugars and 290 mg of sodium, is

$$\hat{y} = 69.180 - 2.3944(1) - 0.06057(290) = 49.22$$

The prediction error for Cheerios is the difference between its actual rating y and the predicted rating \hat{y} : $(y - \hat{y}) = 50.765 - 49.22 = 1.545$. Note that the prediction error for this multiple regression model is smaller than the prediction error from the previous model, $(y - \hat{y}) = -6.62266$, because our new model uses double the data (two predictors rather than one) compared to the earlier regression model.

The standard error of the estimate has been reduced from $s \approx 9.2$ points to $s \approx 7.7$ points. The addition of the sodium information to the model has reduced our typical prediction errors to 7.7 points. Finally, the value of $r^2 = 58.4\%$ has increased from 58.4% to 70.8%, so that the proportion of the variability in nutritional rating that is explained by our regression model is now over 70%.

5.13 VERIFYING MODEL ASSUMPTIONS

Before a model can be implemented, the requisite model assumptions must be verified. Using a model whose assumptions are not verified is like building a house whose foundation may be cracked. Making predictions using a model where the assumptions are violated may lead to erroneous and overoptimistic results, with costly consequences when deployed.

These assumptions—linearity, independence, normality, and constant variance—may be checked using a normality plot of the residuals (Figure 5.9, upper left), and a plot of the standardized residuals against the fitted (predicted) values (Figure 5.9, upper right). One evaluates a normality plot by judging whether systematic deviations from linearity exist in the plot, in which case one concludes that the data values plotted (the residuals in this case) are not drawn from the particular distribution

The regression equation is
Rating = 69.2 - 2.39 Sugars - 0.0606 Sodium

76 cases used, 1 cases contain missing values

Predictor	Coef	SE Coef	T	P
Constant	69.180	2.373	29.15	0.000
Sugars	-2.3944	0.2041	-11.73	0.000
Sodium	-0.06057	0.01086	-5.58	0.000

S = 7.72769 R-Sq = 70.8% R-Sq(adj) = 70.0%

Analysis of Variance

Source	DF	SS	MS	F	P
Regression	2	10569.9	5285.0	88.50	0.000
Residual Error	73	4359.4	59.7		
Total	75	14929.3			

Source	DF	Seq SS
Sugars	1	8711.9
Sodium	1	1858.0

Unusual Observations

Obs	Sugars	Rating	Fit	SE Fit	Residual	St Resid
1	6.0	68.403	46.940	0.970	21.463	2.80R
2	8.0	33.984	49.116	1.845	-15.133	-2.02R
3	5.0	59.426	41.461	1.465	17.964	2.37R
4	0.0	93.705	60.701	1.691	33.004	4.38R

R denotes an observation with a large standardized residual.

Predicted Values for New Observations

New Obs	Fit	SE Fit	95% CI	95% PI
1	49.222	2.107	(45.023, 53.421)	(33.258, 65.185)

Values of Predictors for New Observations

New Obs	Sugars	Sodium
1	1.00	290

Figure 5.8 Multiple regression results.

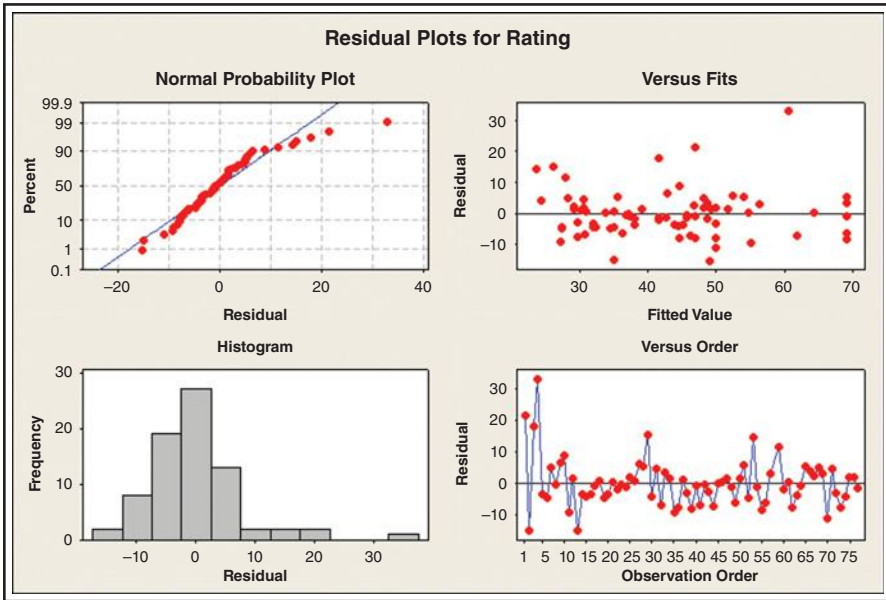


Figure 5.9 Plots for verifying regression model assumptions. Note the outlier.

(the normal distribution in this case). The bulk of the points in the normal probability plot do line up on a straight line, so our normality assumption is essentially intact. But there is one extreme value (cereal 4: All-Bran with Extra Fiber) which may be making mischief with respect to normality. We will return to this in a moment.

The plot of the residuals versus the fits (Figure 5.9, upper right) is examined for discernible patterns. If obvious curvature exists in the scatter plot, the linearity assumption is violated. If the vertical spread of the points in the plot is systematically nonuniform, the constant variance assumption is violated. We detect no such patterns in the plot of the residuals versus fits and therefore conclude that the linearity and constant variance assumptions are intact for this example.

The independence assumption makes sense for this data set, since we would not expect that the rating for one particular cereal would depend on the rating for another cereal. Time-dependent data can be examined for order independence using a runs test or a plot of the residuals versus ordering.

Now, humans are pattern recognition experts. For example, Rorschach tests would indicate that humans see patterns everywhere, even where no such pattern actually exists! Thus, we need to exercise care when reading these plots not to see patterns that do not exist. For example, the departures from linearity must be systematic and significant. Secondly, the huge data sets involved in data mining usually are not well behaved mathematically, conforming perfectly with normality and so on. Thus, the analyst should give the model assumptions the benefit of the doubt, unless there is good evidence to the contrary.

Because least squares regression is based on the *squared* prediction errors, outliers can have an outsized influence on the results. Thus care should be taken to

identify extreme outliers, and, *if necessary and appropriate*, omit them. Figure 5.8 identified 4 outliers, including the most extreme outlier, cereal 4: All-Bran with Extra Fiber. Note that this cereal is extreme in each of the four plots in Figure 5.9, and is skewing our normality a bit. Suppose that we omit cereal 4 from our analysis. Then our assumption plots look better behaved (Figure 5.10), especially the plots for normality (upper left and lower left).

Is our omission of cereal 4 warranted? This depends on the scope and purpose of the project. If we intend to include further predictors, then we should definitely *not* omit cereal 4 at this stage, since an observation that is an outlier in 3-dimensional space may not be an outlier in 10-dimensional space. On the other hand, if our project required us to ignore all the other predictors in the data set, and restrict ourselves to sugars and sodium, then omission of cereal 4 may perhaps be considered.

In Chapter 13, *Imputation of Missing Data*, we illustrate how to use multiple regression to impute missing predictor values.

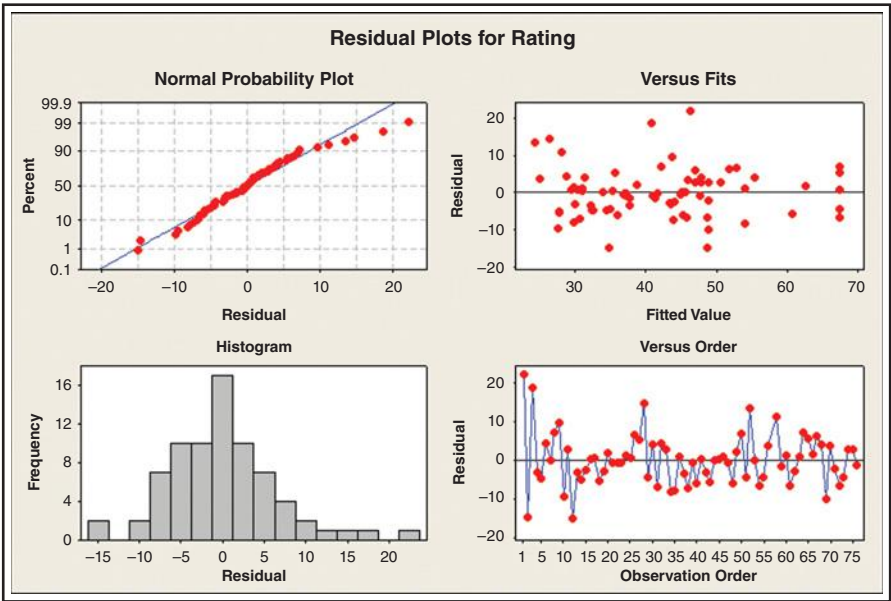


Figure 5.10 Plots for verifying regression model assumptions, after outlier omitted.

THE R ZONE

Two-Sample T-Test for difference in means

```
# Input the summary statistics from Table 5.1
xbar1 <- 1.5714
xbar2 <- 1.5361
s1 <- 1.3126
s2 <- 1.3251
n1 <- 2529
n2 <- 804
# Make the degrees of freedom the smaller of the two sample sizes
dfs <- min(n1-1, n2-1)
# Calculate test statistic
tdata <- (xbar1 - xbar2) / sqrt((s1^2/n1)+(s2^2/n2))
# Find and display the p-value
pvalue <- 2*pt(tdata, df = dfs, lower.tail=FALSE)
tdata; pvalue
```

```
> tdata; pvalue
[1] 0.6594724
[1] 0.5097815
```

Two-Sample Z-Test for Difference in Proportions

```
# Input the summary statistics
# Some of these will override the values
# from the previous example
x1 <- 707
x2 <- 215
n1 <- 2529
n2 <- 804
p1 <- x1 / n1
p2 <- x2 / n2
ppooled <- (x1+x2) / (n1+n2)
# Calculate test statistic
zdata <- (p1-p2) / sqrt(ppooled*(1-ppooled)*((1/n1)+(1/n2)))
# Find the p-value
pvalue <- 2*pnorm(abs(zdata), lower.tail = FALSE)
pvalue
```

```
> pvalue
[1] 0.5025133
```

Chi-Square Goodness of Fit of Multinomial Data

```
# Population proportions
p_status <- c(0.40, 0.35, 0.25)
# Observed frequencies
o_status <- c(36, 35, 29)
chisq.test(o_status, p = p_status)
```

```
> chisq.test(o_status, p = p_status)

Chi-squared test for given probabilities

data:  o_status
X-squared = 1.04, df = 2, p-value = 0.5945
```

Chi-Square Test for Homogeneity of Proportions

```
# Recreate Table 5.2
table5.2 <- as.table(rbind(c(410, 340, 250),
  c(95, 85, 70)))
dimnames(table5.2) <- list(Data.Set =
  c("Training Set", "Test Set"),
  Status = c("Married", "Single", "Other"))
Xsq_data <- chisq.test(table5.2)
# Show the test statistic,
# p-value, expected frequencies
Xsq_data$statistic
Xsq_data$p.value
Xsq_data$expected
```

```
> Xsq_data$statistic
X-squared
1.14867
> Xsq_data$p.value
[1] 0.5630793
> Xsq_data$expected
      Status
Data.Set Married Single other
Training Set    404    340   256
Test Set       101     85    64
```

ANOVA

```
a <- c(30, 40, 50, 60)
b <- c(25, 30, 50, 55)
c <- c(25, 30, 40, 45)
ab <- append(a,b)
datavalues <- append(ab, c)
datalabels <- factor(c(rep("a", length(a)),
  rep("b", length(b)),
  rep("c", length(c))))
anova.results <- aov(datavalues ~ datalabels)
summary(anova.results)
```

```
> summary(anova.results)
      Df Sum Sq Mean Sq F value Pr(>F)
datalabels    2    200   100.0   0.643   0.548
Residuals    9   1400   155.6
```

Recreate Table 5.9

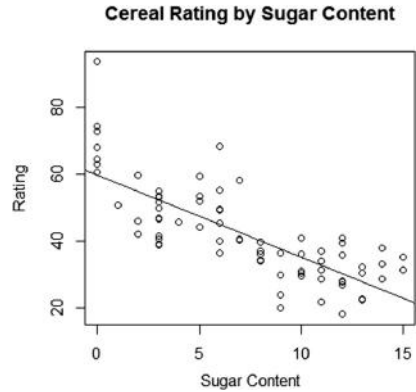
```
# Input the dataset Cereals
cereal <- read.csv(file =
  "C:/.../cereals.txt",
  stringsAsFactors=TRUE,
  header=TRUE,
  sep="\t")
# Display the names of all the columns
names(cereal)
cereal[1:6,c("Name", "Manuf", "Type",
  "Sugars", "Calories", "Protein",
  "Fat", "Sodium", "Rating")]
```

```
> names(cereal)
[1] "Name"      "Manuf"     "Type"
[4] "Calories"  "Protein"   "Fat"
[7] "Sodium"    "Fiber"     "Carbo"
[10] "Sugars"    "Potass"    "vitamins"
[13] "Shelf"     "weight"    "Cups"
[16] "Rating"
```

	Name	Manuf	Type	Sugars	Calories	Protein	Fat	Sodium	Rating
1	100%_Bran	N	C	6	70	4	1	130	68.40297
2	100%_Natural_Bran	Q	C	8	120	3	5	15	33.98368
3	All-Bran	K	C	5	70	4	1	260	59.42551
4	All-Bran_with_Extra_Fiber	K	C	0	50	4	0	140	93.70491
5	Almond_Delight	R	C	8	110	2	2	200	34.38484
6	Apple_Cinnamon_Cheerios	G	C	10	110	2	2	180	29.50954

Create regression line of rating by sugar content

```
# Save Rating and Sugar as new variables
sugars <- cereal$Sugars
rating <- cereal$Rating
# Create regression line
lm.out <- lm(rating~sugars)
# Create scatterplot and overlay regression line
plot(sugars,
     rating,
     main = "Cereal Rating by Sugar Content",
     xlab = "Sugar Content",
     ylab = "Rating")
abline(lm.out)
```

**# Show the full regression output**

```
summary(lm.out)      > summary(lm.out)

Call:
lm(formula = rating ~ sugars)

Residuals:
    Min       1Q   Median       3Q      Max
-17.877  -5.612  -1.285   4.689  33.852

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  59.8530     1.9975   29.96 < 2e-16 ***
sugars       -2.4614     0.2417  -10.18 1.01e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.166 on 74 degrees of freedom
(1 observation deleted due to missingness)
Multiple R-squared:  0.5835, Adjusted R-squared:  0.5779
F-statistic: 103.7 on 1 and 74 DF,  p-value: 1.006e-15
```

Confidence interval and Prediction interval using Regression output

```
pred.confidence <- predict(lm.out,
                           data.frame(sugars = 1),
                           interval = "confidence")
pred.prediction <- predict(lm.out,
                           data.frame(sugars = 1),
                           interval = "prediction")
pred.confidence
pred.prediction
```

```
> pred.confidence
      fit      lwr      upr
1 57.3916 53.81197 60.97123
> pred.prediction
      fit      lwr      upr
1 57.3916 38.78015 76.00305
```

Multiple Regression

```
# Use Sodium and Sugars
# to predict Rating
sodium <- cereal$Sodium
mreg.out <- lm(rating ~ sugars
               + sodium)
summary(mreg.out)
```

```
> summary(mreg.out)

call:
lm(formula = rating ~ sugars + sodium)

Residuals:
    Min       1Q   Median       3Q      Max
-15.133  -4.288  -0.722   2.922  33.004

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  69.18005   2.37320   29.151 < 2e-16 ***
sugars       -2.39439   0.20414  -11.729 < 2e-16 ***
sodium       -0.06057   0.01086   -5.578 3.9e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.728 on 73 degrees of freedom
(1 observation deleted due to missingness)
Multiple R-squared:  0.708, Adjusted R-squared:  0.7
F-statistic: 88.5 on 2 and 73 DF, p-value: < 2.2e-16
```

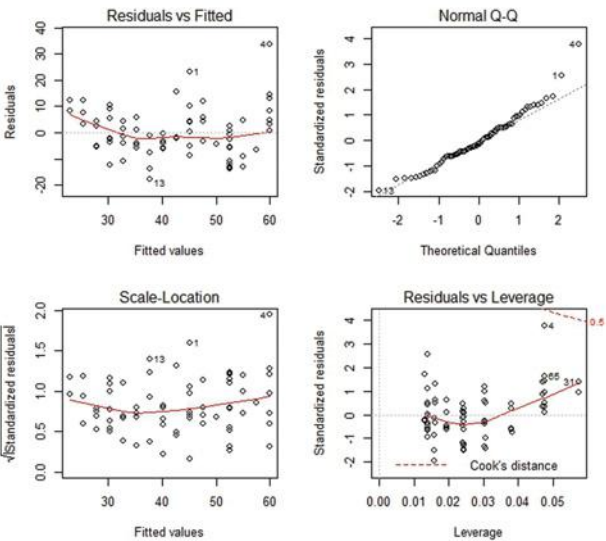
Prediction interval using Multiple Regression output

```
mreg.int <- predict(mreg.out,
                    data.frame(sugars = 1,
                               sodium = 290),
                    interval = "confidence")
mreg.int
```

```
> mreg.int
      fit      lwr      upr
1 49.22176 45.02262 53.4209
```

Plot the four default diagnostic plots

```
# The first two are the top
# two plots in Figure 5.9
par(mfrow=c(2,2))
plot(lm.out)
```



REFERENCE

1. *Data and Story Library*, Carnegie Mellon University, Pittsburgh, PA, www.lib.stat.cmu.edu/DASL, accessed March 17, 2014.

Much more information regarding the topics covered in this chapter may be found in any introductory statistics textbook, such as *Discovering Statistics*, by Daniel T. Larose, second edition, published by W. H. Freeman, New York, 2013.

EXERCISES

- 1. In Chapter 6, we will learn to split the data set into a training data set and a test data set. To test whether there exist unwanted differences between the training and test set, which hypothesis test do we perform, for the following types of variables:
 - a. Flag variable
 - b. Multinomial variable
 - c. Continuous variable
- 2. Table 5.10 contains information on the mean duration of customer service calls between a training and a test data set. Test whether the partition is valid for this variable, using $\alpha = 0.10$.

TABLE 5.10 Summary statistics for duration of customer service calls

Data set	Sample Mean	Sample Standard Deviation	Sample Size
Training set	$\bar{x}_1 = 20.5$	$s_1 = 5.2$	$n_1 = 2000$
Test set	$\bar{x}_2 = 20.4$	$s_2 = 4.9$	$n_2 = 600$

- 3. Our partition shows that 800 of the 2000 customers in our test set own a tablet, while 230 of the 600 customers in our training set own a tablet. Test whether the partition is valid for this variable, using $\alpha = 0.10$.
- 4. Table 5.11 contains the counts for the marital status variable for the training and test set data. Test whether the partition is valid for this variable, using $\alpha = 0.10$.

TABLE 5.11 Observed frequencies for marital status

Data set	Married	Single	Other	Total
Training set	800	750	450	2000
Test set	240	250	110	600
Total	1040	1000	560	2600

- 5. The multinomial variable *payment preference* takes the values *credit card*, *debit card*, and *check*. Now, suppose we know that 50% of the customers in our population prefer to pay by credit card, 20% prefer debit card, and 30% prefer to pay by check. We have taken a sample from our population, and would like to determine whether it is representative of the population. The sample of size 200 shows 125 customers preferring to pay by credit card, 25 by debit card, and 50 by check. Test whether the sample is representative of the population, using $\alpha = 0.05$.

6. Suppose we wish to test for difference in population means among three groups.
 - a. Explain why it is not sufficient to simply look at the differences among the sample means, without taking into account the variability within each group.
 - b. Describe what we mean by between-sample variability and within-sample variability.
 - c. Which statistics measure the concepts in (b).
 - d. Explain how analysis of variance would work in this situation.
7. Table 5.12 contains the amount spent (in dollars) in a random sample of purchases where the payment was made by credit card, debit card, and check, respectively. Test whether the population mean amount spent differs among the three groups, using $\alpha = 0.05$.

TABLE 5.12 Purchase amounts for three payment methods

Credit Card	Debit Card	Check
100	80	50
110	120	70
90	90	80
100	110	80

8. Refer to the previous exercise. Now test whether the population mean amount spent differs among the three groups, using $\alpha = 0.01$. Describe any conflict between your two conclusions. Suggest at least two courses of action to ameliorate the situation.
9. Explain why we use regression analysis and for which type of variables it is appropriate.
10. Suppose that we are interested in predicting weight of students based on height. We have run a regression analysis with the resulting estimated regression equation as follows: "The estimated weight equals (-180 pounds) plus (5 pounds times the height in inches)."

 - a. Suppose that one student is 3 inches taller than another student. What is the estimated difference in weight?
 - b. Suppose that a given student is 65 inches tall. What is the estimated weight?
 - c. Suppose that the regression equation above was based on a sample of students ranging in height from 60 to 75 inches. Now estimate the height of a 48-inch-tall student. Comment.
 - d. Explain clearly the meaning of the 5 in the equation above.
 - e. Explain clearly the meaning of the -180 in the equation above.

HANDS-ON ANALYSIS

Use the *cereals* data set included, at the book series website, for the following exercises. Use regression to estimate *rating* based on *fiber* alone.

11. What is the estimated regression equation?
12. Explain clearly the value of the slope coefficient you obtained in the regression.
13. What does the value of the y-intercept mean for the regression equation you obtained? Does it make sense in this example?

14. What would be a typical prediction error obtained from using this model to predict *rating*? Which statistic are you using to measure this? What could we do to lower this estimated prediction error?
15. How closely does our model fit the data? Which statistic are you using to measure this?
16. Find a point estimate for the rating for a cereal with a fiber content of 3 grams.
17. Find a 95% confidence interval for the true mean rating for all cereals with a fiber content of 3 grams.
18. Find a 95% prediction interval for a randomly chosen cereal with a fiber content of 3 grams.
19. Based on the regression results, what would we expect a scatter plot of *rating* versus *fiber* to look like? Why?

For the following exercises, use multiple regression to estimate *rating* based on *fiber* and *sugars*.

20. What is the estimated regression equation?
21. Explain clearly and completely the value of the coefficient for fiber you obtained in the regression.
22. Compare the r^2 values from the multiple regression and the regression done earlier in the exercises. What is going on? Will this always happen?
23. Compare the s values from the multiple regression and the regression done earlier in the exercises. Which value is preferable, and why? ■

PREPARING TO MODEL THE DATA

- 6.1 SUPERVISED VERSUS UNSUPERVISED METHODS 138
- 6.2 STATISTICAL METHODOLOGY AND DATA MINING METHODOLOGY 139
- 6.3 CROSS-VALIDATION 139
- 6.4 OVERFITTING 141
- 6.5 BIAS-VARIANCE TRADE-OFF 142
- 6.6 BALANCING THE TRAINING DATA SET 144
- 6.7 ESTABLISHING BASELINE PERFORMANCE 145
 - THE R ZONE 146
 - REFERENCE 147
 - EXERCISES 147

6.1 SUPERVISED VERSUS UNSUPERVISED METHODS

Data mining methods may be categorized as either supervised or unsupervised. In *unsupervised methods*, no target variable is identified as such. Instead, the data mining algorithm searches for patterns and structure among all the variables. The most common unsupervised data mining method is clustering, our topic for Chapters 10 and 11. For example, political consultants may analyze congressional districts using clustering methods, to uncover the locations of voter clusters that may be responsive to a particular candidate's message. In this case, all appropriate variables (e.g., income, race, gender) would be input to the clustering algorithm, with no target variable specified, in order to develop accurate voter profiles for fund-raising and advertising purposes.

Another data mining method, which may be supervised or unsupervised, is association rule mining. In market basket analysis, for example, one may simply be interested in "which items are purchased together," in which case no target variable

would be identified. The problem here, of course, is that there are so many items for sale, that searching for all possible associations may present a daunting task, due to the resulting combinatorial explosion. Nevertheless, certain algorithms, such as the *a priori* algorithm, attack this problem cleverly, as we shall see when we cover association rule mining in Chapter 12.

Most data mining methods are *supervised methods*, however, meaning that (1) there is a particular prespecified target variable, and (2) the algorithm is given many examples where the value of the target variable is provided, so that the algorithm may learn which values of the target variable are associated with which values of the predictor variables. For example, the regression methods of Chapter 5 are supervised methods, since the observed values of the response variable y are provided to the least-squares algorithm, which seeks to minimize the squared distance between these y values and the y values predicted given the x -vector. All of the classification methods we examine in Chapters 7–9 are supervised methods, including decision trees, neural networks, and k -nearest neighbors.

Note: The terms *supervised* and *unsupervised* are widespread in the literature, and hence used here. However, we do not mean to imply that unsupervised methods require no human involvement. To the contrary, effective cluster analysis and association rule mining both require substantial human judgment and skill.

6.2 STATISTICAL METHODOLOGY AND DATA MINING METHODOLOGY

In Chapters 4 and 5 we were introduced to a wealth of statistical methods for performing inference, that is, for estimating or testing the unknown parameters of a population of interest. Statistical methodology and data mining methodology differ in two ways.

1. Applying statistical inference using the huge sample sizes encountered in data mining tends to result in statistical significance, even when the results are not of practical significance.
2. In statistical methodology, the data analyst has an *a priori* hypothesis in mind. Data mining procedures usually do not have an *a priori* hypothesis, instead freely trolling through the data for actionable results.

6.3 CROSS-VALIDATION

Unless properly conducted, data mining can become data dredging, whereby the analyst “uncovers” phantom spurious results, due to random variation rather than real effects. It is therefore crucial that data miners avoid data dredging. This is accomplished through *cross-validation*.

Cross-validation is a technique for insuring that the results uncovered in an analysis are generalizable to an independent, unseen, data set. In data mining, the most common methods are *twofold cross-validation* and *k-fold cross-validation*. In twofold cross-validation, the data are partitioned, using random assignment, into a

training data set and a *test data set*. The test data set should then have the target variable omitted. Thus, the only systematic difference between the training data set and the test data set is that the training data includes the target variable and the test data does not. For example, if we are interested in classifying *income bracket*, based on *age*, *gender*, and *occupation*, our classification algorithm would need a large pool of records, containing complete (as complete as possible) information about every field, including the target field, *income bracket*. In other words, the records in the *training set* need to be *preclassified*. A provisional data mining model is then constructed using the training samples provided in the training data set.

However, the training set is necessarily incomplete; that is, it does not include the “new” or future data that the data modelers are really interested in classifying. Therefore, the algorithm needs to guard against “memorizing” the training set and blindly applying all patterns found in the training set to the future data. For example, it may happen that all customers named “David” in a training set may be in the high income bracket. We would presumably not want our final model, to be applied to new data, to include the pattern “If the customer’s first name is David, the customer has a high income.” Such a pattern is a spurious artifact of the training set and needs to be verified before deployment.

Therefore, the next step in supervised data mining methodology is to examine how the provisional data mining model performs on a *test set* of data. In the test set, a holdout data set, the values of the target variable are hidden temporarily from the provisional model, which then performs classification according to the patterns and structure it learned from the training set. The efficacy of the classifications is then evaluated by comparing them against the true values of the target variable. The provisional data mining model is then adjusted to minimize the error rate on the test set.

Estimates of model performance for future, unseen data can then be computed by observing various evaluative measures applied to the test data set. Such model evaluation techniques are covered in Chapter 14. The bottom line is that cross-validation guards against spurious results, since it is highly unlikely that the same random variation would be found to be significant in both training set and the test set. For example, a spurious signal with probability 0.05 of being observed, if in fact no real signal existed, would have only $0.05^2 = 0.0025$ probability of being observed in both the training and test sets, because these data sets are independent. In other words, the data analyst could report on average 400 results before one would expect a spurious result be reported.

But the data analyst must insure that the training and test data sets are indeed independent, by *validating the partition*. We validate the partition into training and test data sets by performing graphical and statistical comparisons between the two sets. For example, we may find that, even though the assignment of records was made randomly, a significantly higher proportion of positive values of an important flag variable were assigned to the training set, compared to the test set. This would bias our results, and hurt our prediction or classification accuracy on the test data set. It is especially important that the characteristics of the target variable be as similar as possible between the training and test data sets. Table 6.1 shows the suggested hypothesis test for validating the target variable, based on the type of target variable.

In *k*-fold cross-validation, the original data are partitioned into *k* independent and similar subsets. The model is then built using the data from *k* – 1 subsets, using

TABLE 6.1 Suggested hypothesis tests for validating different types of target variables

Type of Target Variable	Test From Chapter 5
Continuous	Two-sample <i>t</i> -test for difference in means
Flag	Two-sample <i>Z</i> -test for difference in proportions
Multinomial	Test for homogeneity of proportions

the *k*th subset as the test set. This is done iteratively until we have *k* different models. The results from the *k* models are then combined using averaging or voting. A popular choice for *k* is 10. A benefit of using *k*-fold cross-validation is that each record appears in the test set exactly once; a drawback is that the requisite validation task is made more difficult.

To summarize, most supervised data mining methods apply the following methodology for building and evaluating a model.

METHODOLOGY FOR BUILDING AND EVALUATING A DATA MODEL

1. Partition the available data into a *training set* and a *test set*. Validate the partition.
2. Build a data mining model using the training set data.
3. Evaluate the data mining model using the test set data.

6.4 OVERFITTING

Usually, the accuracy of the provisional model is not as high on the test set as it is on the training set, often because the provisional model is *overfitting* on the training set. Overfitting results when the provisional model tries to account for every possible trend or structure in the training set, even idiosyncratic ones such as the “David” example above. There is an eternal tension in model building between model complexity (resulting in high accuracy on the training set) and generalizability to the test and validation sets. Increasing the complexity of the model in order to increase the accuracy on the training set eventually and inevitably leads to a degradation in the generalizability of the provisional model to the test set, as shown in Figure 6.1.

Figure 6.1 shows that as the provisional model begins to grow in complexity from the null model (with little or no complexity), the error rates on both the training set and the test set fall. As the model complexity increases, the error rate on the training set continues to fall in a monotone fashion. However, as the model complexity increases, the test set error rate soon begins to flatten out and increase because the provisional model has memorized the training set rather than leaving room for generalizing to unseen data. The point where the minimal error rate on the test set is encountered is the optimal level of model complexity, as indicated in Figure 6.1. Complexity greater than this is considered to be overfitting; complexity less than this is considered to be underfitting.

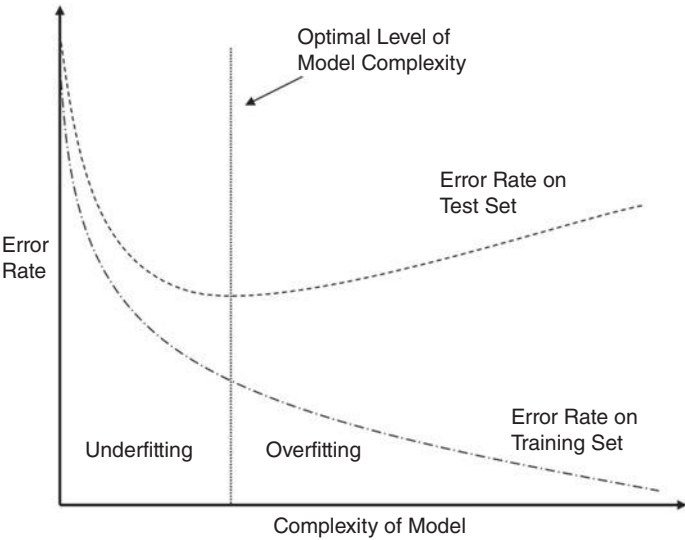


Figure 6.1 The optimal level of model complexity is at the minimum error rate on the test set.

6.5 BIAS–VARIANCE TRADE-OFF

Suppose that we have the scatter plot in Figure 6.2 and are interested in constructing the optimal curve (or straight line) that will separate the dark gray points from the light gray points. The straight line has the benefit of low complexity but suffers from some classification errors (points ending up on the wrong side of the line).

In Figure 6.3 we have reduced the classification error to zero but at the cost of a much more complex separation function (the curvy line). One might be tempted to adopt the greater complexity in order to reduce the error rate. However, one should be careful not to depend on the idiosyncrasies of the training set. For example,

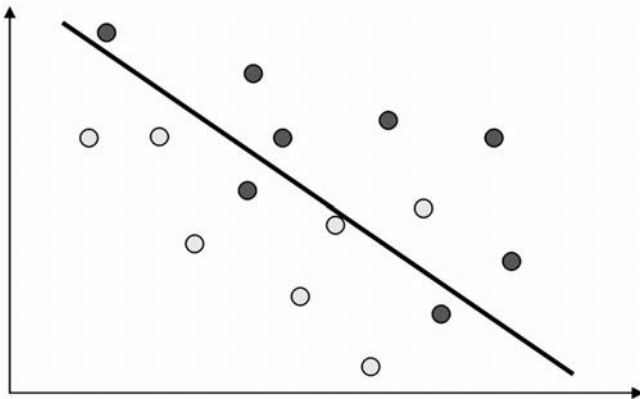


Figure 6.2 Low complexity separator with high error rate.

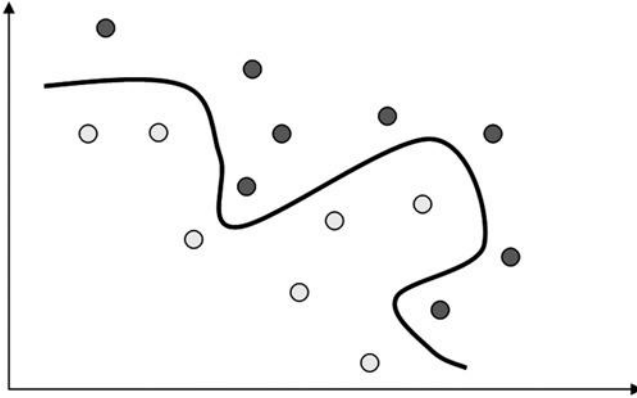


Figure 6.3 High complexity separator with low error rate.

suppose that we now add more data points to the scatter plot, giving us the graph in Figure 6.4.

Note that the low complexity separator (the straight line) need not change very much to accommodate the new data points. This means that this low complexity separator has *low variance*. However, the high complexity separator, the curvy line, must alter considerably if it is to maintain its pristine error rate. This high degree of change indicates that the high complexity separator has a *high variance*.

Even though the high complexity model has a low *bias* (in terms of the error rate on the training set), it has a high *variance*; and even though the low complexity model has a high *bias*, it has a low *variance*. This is what is known as the *bias-variance trade-off*. The bias-variance trade-off is another way of describing the overfitting/underfitting dilemma shown in Figure 6.1. As model complexity increases, the bias on the training set decreases but the variance increases. The goal is to construct a model in which neither the bias nor the variance is too high, but usually, minimizing one tends to increase the other.

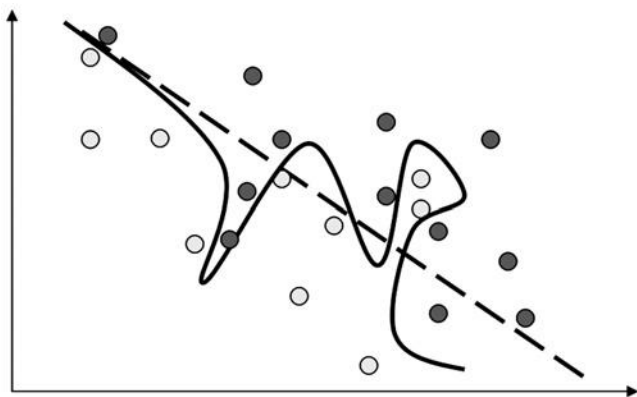


Figure 6.4 With more data: low complexity separator need not change much; high complexity separator needs much revision.

For example, a common method of evaluating how accurate model estimation is proceeding for a continuous target variable is to use the *mean-squared error* (MSE). Between two competing models, one may select the better model as that model with the lower MSE. Why is MSE such a good evaluative measure? Because it combines both bias and variance. The MSE is a function of the estimation error (SSE) and the model complexity (e.g., degrees of freedom). It can be shown (e.g., Hand et al. [1]) that the MSE can be partitioned using the following equation, which clearly indicates the complementary relationship between bias and variance:

$$\text{MSE} = \text{variance} + \text{bias}^2$$

6.6 BALANCING THE TRAINING DATA SET

For classification models in which one of the target variable classes has much lower relative frequency than the other classes, balancing is recommended. A benefit of balancing the data is to provide the classification algorithms with a rich balance of records for each classification outcome, so that the algorithms have a chance to learn about all types of records, not just those with high target frequency. For example, suppose we are running a fraud classification model and our training data set consists of 100,000 transactions, only 1000 of which are fraudulent. Then our classification model could simply predict “non-fraudulent” for all transactions, and achieve 99% classification accuracy. However, clearly this model is useless.

Instead the analyst should balance the training data set so that the relative frequency of fraudulent transactions is increased. There are two ways to accomplish this.

1. Resample a number of fraudulent records.
2. Set aside a number of non-fraudulent records.

Resampling refers to the process of sampling at random and with replacement from a data set. Suppose we wished our 1000 fraudulent records to represent 25% of the balanced training set, rather than the 1% represented by these records in the raw training data set. Then we could add 32,000 resampled fraudulent records so that we had 33,000 fraudulent records, out of a total of $100,000 + 32,000 = 132,000$ records in all. This represents $\frac{33,000}{132,000} = 0.25$ or the desired 25%.

How did we arrive at the number of 32,000 additional fraudulent records? By using the equation

$$1000 + x = 0.25 (100,000 + x)$$

and solving for x , the required number of additional records to resample. In general, this equation is:

$$\text{rare} + x = p(\text{records} + x)$$

and solving for x gives us:

$$x = \frac{p(\text{records}) - \text{rare}}{1 - p}$$

where x is the required number of resampled records, p represents the desired proportion of rare values in the balanced data set, *records* represents the number of

records in the unbalanced data set, and *rare* represents the current number of rare target values.

Some data miners have a philosophical aversion to resampling records to achieve balance, since they feel this amounts to fabricating data. In this case, a sufficient number of non-fraudulent transactions would instead be set aside, thereby increasing the proportion of fraudulent transactions. To achieve a 25% balance proportion, we would retain only 3000 non-fraudulent records. We would then need to discard from the analysis 96,000 of the 99,000 non-fraudulent records, using random selection. It would not be surprising if our data mining models would suffer as a result of starving them of data in this way. Instead, the data analyst would probably be well-advised either to decrease the desired balance proportion to something like 10%, or to use resampling.

When choosing a desired balancing proportion, recall the rationale for doing so: in order to allow the model a sufficiently rich variety of records to learn how to classify the rarer value of the target variable across a range of situations. The balancing proportion can be relatively low (e.g., 10%) if the analyst is confident that the rare target value is exposed to a sufficiently rich variety of records. The balancing proportion should be higher (e.g., 25%) if the analyst is not so confident of this.

The test data set should never be balanced. The test data set represents new data that the models have not seen yet. Certainly the real world will not balance tomorrow's data for the convenience of our classification models; therefore, the test data set itself should not be balanced. Note that all model evaluation will take place using the test data set, so that the evaluative measures will all be applied to unbalanced (real-world-like) data.

6.7 ESTABLISHING BASELINE PERFORMANCE

In *Star Trek IV: The Voyage Home*, Captain Kirk travels back in time to the twentieth century, finds himself in need of cash, and pawns his eyeglasses. The buyer offers him \$100, to which Captain Kirk responds, "Is that a lot?" Unfortunately, the Captain had no frame of reference to compare the \$100 to, and so was unable to determine whether the \$100 was a satisfactory offer or not. As data analysts we should do our best to avoid putting our clients into Captain Kirk's situation, by reporting results with no comparison to a baseline. Without comparison to a baseline, a client cannot determine whether our results are any good.

For example, suppose we naively report that "only" 28.4% of customers adopting our International Plan (see Table 3.3) will churn. That does not sound too bad, until we recall that, among all of our customers, the overall churn rate is only 14.49% (Figure 3.3). This overall churn rate may be considered our *baseline*, against which any further results can be calibrated. Thus, belonging to the International Plan actually nearly doubles the churn rate, which is clearly not good.

The type of baseline one should use depends on the way the results are reported. For the churn example, we are interested in decreasing the overall churn rate, which is expressed as a percentage. So, our objective would be to report a decrease in the overall churn rate. Note the difference between an absolute difference in churn rate versus a relative difference in churn rate. Suppose our data mining model resulted

in a predicted churn rate of 9.99%. This represents only a $14.49\% - 9.99\% = 4.5\%$ absolute decrease in the churn rate, but a $4.5\%/14.49\% = 31\%$ relative decrease in the churn rate. The analyst should make it clear for the client which comparison method is being used.

Suppose our task is estimation, and we are using a regression model. Then our baseline model may take the form of a “ \bar{y} model”, that is, a model which simply finds the mean of the response variable, and predicts that value for every record. Clearly this is quite naïve, so no data mining model worth its salt should have a problem beating this \bar{y} model. By the same token, if your data mining model cannot outperform the \bar{y} model, then something is clearly wrong. (Recall that we measure the goodness of a regression model using the standard error of the estimate s along with r^2 .)

A more challenging yardstick against which to calibrate your model is to use existing research or results already existing in the field. For example, suppose the algorithm your analytics company currently uses succeeds in identifying 90% of all fraudulent online transactions. Then your company will probably expect your new data mining model to outperform this 90% baseline.

THE R ZONE

Input the data

```
dat <- read.csv(file = "C:/... /adult.txt",
  stringsAsFactors=TRUE)
```

Partition data into 75% training data, 25% testing data

```
dat$part <- runif(length(dat$income),
  min = 0,
  max = 1)
dat[1:5, c(1,2,3,16)]
training <- dat[dat$part <= 0.75,]
testing <- dat[dat$part > 0.75,]
training[1:5, c(1,2,3,16)]
testing[1:5, c(1,2,3,16)]
```

	age	workclass	demogweight	part
1	39	State-gov	77516	0.07055998
2	50	Self-emp-not-inc	83311	0.96249655
3	38	Private	215646	0.25632883
4	53	Private	234721	0.51016661
5	28	Private	338409	0.11353234

	age	workclass	demogweight	part
1	39	State-gov	77516	0.07055998
3	38	Private	215646	0.25632883
4	53	Private	234721	0.51016661
5	28	Private	338409	0.11353234
6	37	Private	284582	0.34662341

	age	workclass	demogweight	part
2	50	Self-emp-not-inc	83311	0.9624965
11	37	Private	280464	0.9193413
13	23	Private	122272	0.7865689
14	32	Private	205019	0.7539208
17	25	Self-emp-not-inc	176756	0.8244207

Remove the target variable, Income, from the testing data

```

names(testing)
# The target variable is column 15
testing <- testing[,-15]
names(testing)
# Target variable is no longer in
# the testing data

```

```

> names(testing)
[1] "age"           "workclass"      "demogweight"
[4] "education"     "education.num"  "marital.status"
[7] "occupation"    "relationship"   "race"
[10] "sex"           "capital.gain"   "capital.loss"
[13] "hours.per.week" "native.country" "income"
[16] "part"
> testing <- testing[,-15]
> names(testing)
[1] "age"           "workclass"      "demogweight"
[4] "education"     "education.num"  "marital.status"
[7] "occupation"    "relationship"   "race"
[10] "sex"           "capital.gain"   "capital.loss"
[13] "hours.per.week" "native.country" "part"

```

Remove the partitioning variable, Part, from both data sets

```

# Part is now the 15th variable
testing <- testing[,-15]
names(testing)
names(training)
# Part is the 16th variable
# in the training data set
training <- training[,-16]
names(training)

```

```

> names(testing)
[1] "age"           "workclass"      "demogweight"
[4] "education"     "education.num"  "marital.status"
[7] "occupation"    "relationship"   "race"
[10] "sex"           "capital.gain"   "capital.loss"
[13] "hours.per.week" "native.country"
[16] "part"
> names(training)
[1] "age"           "workclass"      "demogweight"
[4] "education"     "education.num"  "marital.status"
[7] "occupation"    "relationship"   "race"
[10] "sex"           "capital.gain"   "capital.loss"
[13] "hours.per.week" "native.country" "income"
[16] "part"
> training <- training[,-16]
> names(training)
[1] "age"           "workclass"      "demogweight"
[4] "education"     "education.num"  "marital.status"
[7] "occupation"    "relationship"   "race"
[10] "sex"           "capital.gain"   "capital.loss"
[13] "hours.per.week" "native.country" "income"

```

REFERENCE

1. David Hand, Heikki Mannila, and Padhraic Smyth, *Principles of Data Mining*, MIT Press, Cambridge, MA, 2001.

EXERCISES

1. Explain the difference between supervised and unsupervised methods. Which data mining tasks are associated with unsupervised methods? Supervised? Both?
2. Describe the differences between the training set, test set, and validation set.
3. Should we strive for the highest possible accuracy with the training set? Why or why not? How about the validation set?
4. How is the bias–variance trade-off related to the issue of overfitting and underfitting? Is high bias associated with overfitting and underfitting, and why? High variance?
5. Explain why we sometimes need to balance the data.

6. Suppose we are running a fraud classification model, with a training set of 10,000 records of which only 400 are fraudulent. How many fraudulent records need to be resampled if we would like the proportion of fraudulent records in the balanced data set to be 20%?
7. When should the test data set be balanced?
8. Explain why we should always report a baseline performance, rather than merely citing the uncalibrated results from our model.
9. Explain the distinction between reporting an absolute difference versus a relative difference.
10. If we are using a regression model, what form may our baseline model take? ■

k-NEAREST NEIGHBOR ALGORITHM

7.1	CLASSIFICATION TASK	149
7.2	<i>k</i> -NEAREST NEIGHBOR ALGORITHM	150
7.3	DISTANCE FUNCTION	153
7.4	COMBINATION FUNCTION	156
7.5	QUANTIFYING ATTRIBUTE RELEVANCE: STRETCHING THE AXES	158
7.6	DATABASE CONSIDERATIONS	158
7.7	<i>k</i> -NEAREST NEIGHBOR ALGORITHM FOR ESTIMATION AND PREDICTION	159
7.8	CHOOSING <i>k</i>	160
7.9	APPLICATION OF <i>k</i> -NEAREST NEIGHBOR ALGORITHM USING IBM/SPSS MODELER	160
	THE R ZONE	162
	EXERCISES	163
	HANDS-ON ANALYSIS	164

7.1 CLASSIFICATION TASK

Perhaps the most common data mining task is that of *classification*. Examples of classification tasks may be found in nearly every field of endeavor:

- *Banking*: determining whether a mortgage application is a good or bad credit risk, or whether a particular credit card transaction is fraudulent
- *Education*: placing a new student into a particular track with regard to special needs
- *Medicine*: diagnosing whether a particular disease is present

- *Law*: determining whether a will was written by the actual person deceased or fraudulently by someone else
- *Homeland security*: identifying whether or not certain financial or personal behavior indicates a possible terrorist threat

In classification, there is a target categorical variable, (e.g., *income bracket*), which is partitioned into predetermined classes or categories, such as high income, middle income, and low income. The data mining model examines a large set of records, each record containing information on the target variable as well as a set of input or predictor variables. For example, consider the excerpt from a data set shown in Table 7.1. Suppose that the researcher would like to be able to *classify* the income bracket of persons not currently in the database, based on the other characteristics associated with that person, such as age, gender, and occupation. This task is a classification task, very nicely suited to data mining methods and techniques.

The algorithm would proceed roughly as follows. First, examine the data set containing both the predictor variables and the (already classified) target variable, *income bracket*. In this way, the algorithm (software) “learns about” which combinations of variables are associated with which income brackets. For example, older females may be associated with the high income bracket. This data set is called the *training set*. Then the algorithm would look at new records for which no information about income bracket is available. Based on the classifications in the training set, the algorithm would assign classifications to the new records. For example, a 63-year-old female professor might be classified in the high income bracket.

7.2 *k*-NEAREST NEIGHBOR ALGORITHM

The first algorithm we shall investigate is the *k-nearest neighbor* algorithm, which is most often used for classification, although it can also be used for estimation and prediction. *k*-Nearest neighbor is an example of *instance-based learning*, in which the training data set is stored, so that a classification for a new unclassified record may be found simply by comparing it to the most similar records in the training set. Let us consider an example.

Recall the example from Chapter 1 where we were interested in classifying the type of drug a patient should be prescribed, based on certain patient characteristics, such as the age of the patient and the patient’s sodium/potassium (Na/K) ratio. For a sample of 200 patients, Figure 7.1 presents a scatter plot of the patients’ Na/K ratio against the patients’ age. The particular drug prescribed is symbolized by the shade

TABLE 7.1 Excerpt from data set for classifying income

Subject	Age	Gender	Occupation	Income Bracket
001	47	F	Software engineer	High
002	28	M	Marketing consultant	Middle
003	35	M	Unemployed	Low
:				

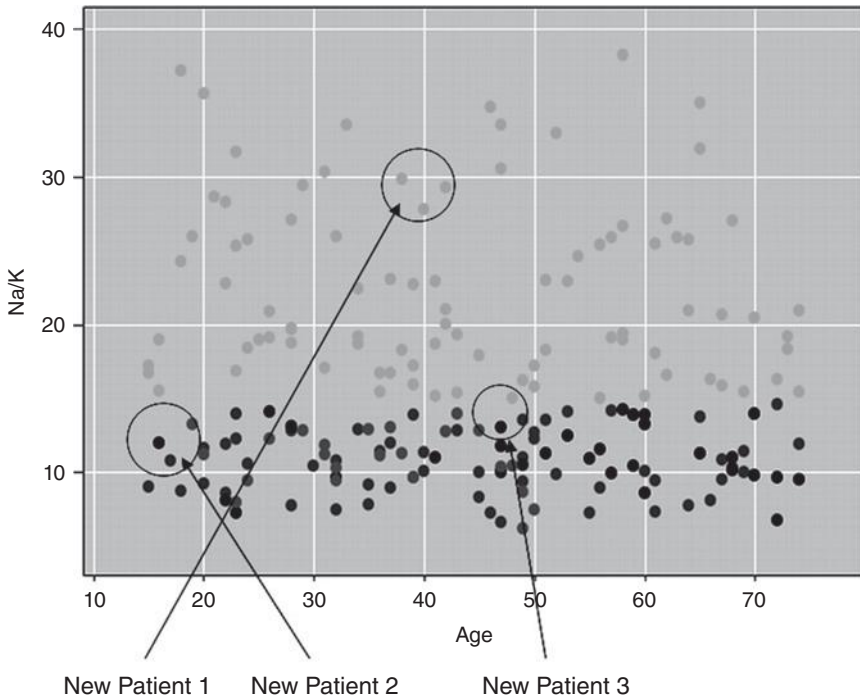


Figure 7.1 Scatter plot of sodium/potassium ratio against age, with drug overlay.

of the points. Light gray points indicate drug Y; medium gray points indicate drug A or X; dark gray points indicate drug B or C.

Now suppose that we have a new patient record, without a drug classification, and would like to classify which drug should be prescribed for the patient based on which drug was prescribed for other patients with similar attributes. Identified as “new patient 1,” this patient is 40 years old and has a Na/K ratio of 29, placing her at the center of the circle indicated for new patient 1 in Figure 7.1. Which drug classification should be made for new patient 1? Since her patient profile places her deep into a section of the scatter plot where all patients are prescribed drug Y, we would thereby classify new patient 1 as drug Y. All of the points nearest to this point, that is, all of the patients with a similar profile (with respect to age and Na/K ratio) have been prescribed the same drug, making this an easy classification.

Next, we move to new patient 2, who is 17 years old with a Na/K ratio of 12.5. Figure 7.2 provides a close-up view of the training data points in the local neighborhood of and centered at new patient 2. Suppose we let $k = 1$ for our k -nearest neighbor algorithm, so that new patient 2 would be classified according to whichever *single* (one) observation it was closest to. In this case, new patient 2 would be classified for drugs B and C (dark gray), since that is the classification of the point closest to the point on the scatter plot for new patient 2.

However, suppose that we now let $k = 2$ for our k -nearest neighbor algorithm, so that new patient 2 would be classified according to the classification of the $k = 2$

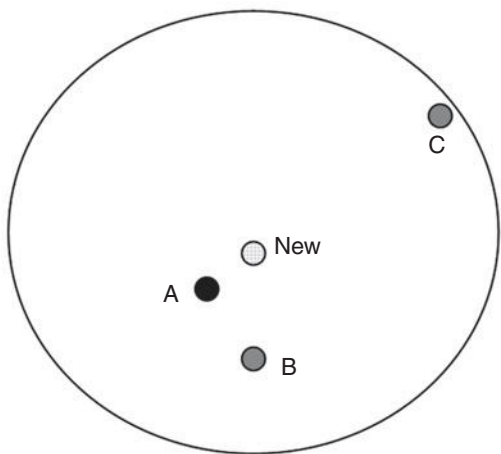


Figure 7.2 Close-up of three nearest neighbors to new patient 2.

points closest to it. One of these points is dark gray, and one is medium gray, so that our classifier would be faced with a decision between classifying new patient 2 for drugs B and C (dark gray) or drugs A and X (medium gray). How would the classifier decide between these two classifications? Voting would not help, since there is one vote for each of two classifications.

Voting would help, however, if we let $k = 3$ for the algorithm, so that new patient 2 would be classified based on the three points closest to it. Since two of the three closest points are medium gray, a classification based on voting would therefore choose drugs A and X (medium gray) as the classification for new patient 2. Note that the classification assigned for new patient 2 differed based on which value we chose for k .

Finally, consider new patient 3, who is 47 years old and has a Na/K ratio of 13.5. Figure 7.3 presents a close-up of the three nearest neighbors to new

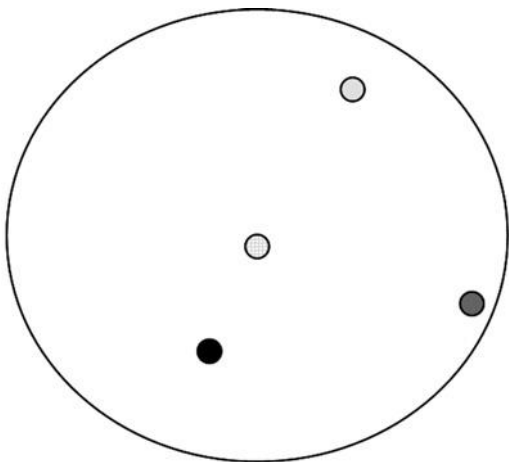


Figure 7.3 Close-up of three nearest neighbors to new patient 3.

patient 3. For $k = 1$, the k -nearest neighbor algorithm would choose the dark gray (drugs B and C) classification for new patient 3, based on a distance measure. For $k = 2$, however, voting would not help. But voting would not help for $k = 3$ in this case either, since the three nearest neighbors to new patient 3 are of three different classifications.

This example has shown us some of the issues involved in building a classifier using the k -nearest neighbor algorithm. These issues include:

- How many neighbors should we consider? That is, what is k ?
- How do we measure distance?
- How do we combine the information from more than one observation?

Later we consider other questions, such as:

- Should all points be weighted equally, or should some points have more influence than others?

7.3 DISTANCE FUNCTION

We have seen above how, for a new record, the k -nearest neighbor algorithm assigns the classification of the most similar record or records. But just how do we define *similar*? For example, suppose that we have a new patient who is a 50-year-old male. Which patient is more similar, a 20-year-old male or a 50-year-old female?

Data analysts define distance metrics to measure similarity. A *distance metric* or *distance function* is a real-valued function d , such that for any coordinates x, y , and z :

1. $d(x, y) \geq 0$, and $d(x, y) = 0$ if and only if $x = y$
2. $d(x, y) = d(y, x)$
3. $d(x, z) \leq d(x, y) + d(y, z)$

Property 1 assures us that distance is always nonnegative, and the only way for distance to be zero is for the coordinates (e.g., in the scatter plot) to be the same. Property 2 indicates commutativity, so that, for example, the distance from New York to Los Angeles is the same as the distance from Los Angeles to New York. Finally, property 3 is the *triangle inequality*, which states that introducing a third point can never shorten the distance between two other points.

The most common distance function is *Euclidean distance*, which represents the usual manner in which humans think of distance in the real world:

$$d_{\text{Euclidean}}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

where $\mathbf{x} = x_1, x_2, \dots, x_m$, and $\mathbf{y} = y_1, y_2, \dots, y_m$ represent the m attribute values of two records. For example, suppose that patient A is $x_1 = 20$ years old and has a Na/K

ratio of $x_2 = 12$, while patient B is $y_1 = 30$ years old and has a Na/K ratio of $y_2 = 8$. Then the Euclidean distance between these points, as shown in Figure 7.4, is

$$\begin{aligned} d_{\text{Euclidean}}(\mathbf{x}, \mathbf{y}) &= \sqrt{\sum_i (x_i - y_i)^2} = \sqrt{(20 - 30)^2 + (12 - 8)^2} \\ &= \sqrt{100 + 16} = 10.77 \end{aligned}$$

When measuring distance, however, certain attributes that have large values, such as income, can overwhelm the influence of other attributes which are measured on a smaller scale, such as years of service. To avoid this, the data analyst should make sure to *normalize* the attribute values.

For continuous variables, the *min-max normalization* or *Z-score standardization*, discussed in Chapter 2, may be used:

Min-max normalization:

$$X^* = \frac{X - \min(X)}{\text{range}(X)} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

Z-score standardization:

$$X^* = \frac{X - \text{mean}(X)}{\text{SD}(X)}$$

For categorical variables, the Euclidean distance metric is not appropriate. Instead, we may define a function, “different from,” used to compare the i th attribute values of a pair of records, as follows:

$$\text{Different}(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{otherwise} \end{cases}$$

where x_i and y_i are categorical values. We may then substitute $\text{different}(x_i, y_i)$ for the i th term in the Euclidean distance metric above.

For example, let us find an answer to our earlier question: which patient is more similar to a 50-year-old male: a 20-year-old male or a 50-year-old female?

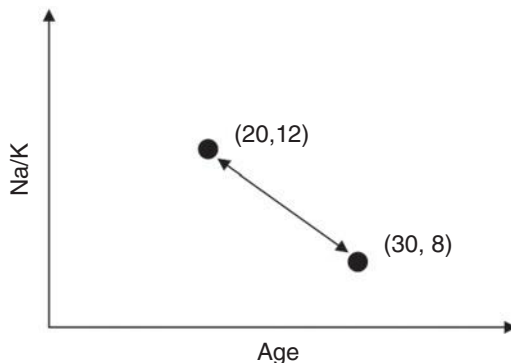


Figure 7.4 Euclidean distance.

Suppose that for the *age* variable, the range is 50, the minimum is 10, the mean is 45, and the standard deviation is 15. Let patient A be our 50-year-old male, patient B the 20-year-old male, and patient C the 50-year-old female. The original variable values, along with the min-max normalization (age_{MMN}) and Z-score standardization ($\text{age}_{\text{Z-score}}$), are listed in Table 7.2.

We have one continuous variable (age, x_1) and one categorical variable (gender, x_2). When comparing patients A and B, we have different $(x_2, y_2) = 0$, with different $(x_2, y_2) = 1$ for the other combinations of patients. First, let us see what happens when we forget to normalize the age variable. Then the distance between patients A and B is $d(A, B) = \sqrt{(50 - 20)^2 + 0^2} = 30$, and the distance between patients A and C is $d(A, C) = \sqrt{(20 - 20)^2 + 1^2} = 1$. We would thus conclude that the 20-year-old male is 30 times more “distant” from the 50-year-old male than the 50-year-old female is. In other words, the 50-year-old female is 30 times more “similar” to the 50-year-old male than the 20-year-old male is. Does this seem justified to you? Well, in certain circumstances, it may be justified, as in certain age-related illnesses. But, in general, one may judge that the two men are just as similar as are the two 50-year olds. The problem is that the *age* variable is measured on a larger scale than the different (x_2, y_2) variable. Therefore, we proceed to account for this discrepancy by normalizing and standardizing the age values, as shown in Table 7.2.

Next, we use the min-max normalization values to find which patient is more similar to patient A. We have $d_{\text{MMN}}(A, B) = \sqrt{(0.8 - 0.2)^2 + 0^2} = 0.6$ and $d_{\text{MMN}}(A, C) = \sqrt{(0.8 - 0.8)^2 + 1^2} = 1.0$, which means that patient B is now considered to be more similar to patient A.

Finally, we use the Z-score standardization values to determine which patient is more similar to patient A. We have $d_{\text{Z-score}}(A, B) = \sqrt{[0.33 - (-1.67)]^2 + 0^2} = 2.0$ and $d_{\text{Z-score}}(A, C) = \sqrt{(0.33 - 0.33)^2 + 1^2} = 1.0$, which means that patient C is again closer. Using the Z-score standardization rather than the min-max standardization has reversed our conclusion about which patient is considered to be more similar to patient A. This underscores the importance of understanding which type of normalization one is using. The min-max normalization will almost always lie between zero and 1 just like the “identical” function. The Z-score standardization, however, usually takes values $-3 < z < 3$, representing a wider scale than that of the min-max normalization. Therefore, perhaps, when mixing categorical and continuous variables, the min-max normalization may be preferred.

TABLE 7.2 Variable values for age and gender

Patient	Age	Age_{MMN}	$\text{Age}_{\text{Z-score}}$	Gender
A	50	$\frac{50 - 10}{50} = 0.8$	$\frac{50 - 45}{15} = 0.33$	Male
B	20	$\frac{20 - 10}{50} = 0.2$	$\frac{20 - 45}{15} = -1.67$	Male
C	50	$\frac{50 - 10}{50} = 0.8$	$\frac{50 - 45}{15} = 0.33$	Female

7.4 COMBINATION FUNCTION

Now that we have a method of determining which records are most similar to the new, unclassified record, we need to establish how these similar records will combine to provide a classification decision for the new record. That is, we need a *combination function*. The most basic combination function is simple unweighted voting.

7.4.1 Simple Unweighted Voting

1. Before running the algorithm, decide on the value of k , that is, how many records will have a voice in classifying the new record.
2. Then, compare the new record to the k nearest neighbors, that is, to the k records that are of minimum distance from the new record in terms of the Euclidean distance or whichever metric the user prefers.
3. Once the k records have been chosen, then for simple unweighted voting, their distance from the new record no longer matters. It is simple one record, one vote.

We observed simple unweighted voting in the examples for Figures 7.2 and 7.3. In Figure 7.2, for $k = 3$, a classification based on simple voting would choose drugs A and X (medium gray) as the classification for new patient 2, since two of the three closest points are medium gray. The classification would then be made for drugs A and X, with *confidence* 66.67%, where the confidence level represents the count of records, with the winning classification divided by k .

On the other hand, in Figure 7.3, for $k = 3$, simple voting would fail to choose a clear winner since each of the three categories receives one vote. There would be a tie among the three classifications represented by the records in Figure 7.3, and a tie may not be a preferred result.

7.4.2 Weighted Voting

One may feel that neighbors that are closer or more similar to the new record should be weighted more heavily than more distant neighbors. For example, in Figure 7.3, does it seem fair that the light gray record farther away gets the same vote as the dark gray vote that is closer to the new record? Perhaps not. Instead, the analyst may choose to apply *weighted voting*, where closer neighbors have a larger voice in the classification decision than do more distant neighbors. Weighted voting also makes it much less likely for ties to arise.

In weighted voting, the influence of a particular record is inversely proportional to the distance of the record from the new record to be classified. Let us look at an example. Consider Figure 7.2, where we are interested in finding the drug classification for a new record, using the $k = 3$ nearest neighbors. Earlier, when using simple unweighted voting, we saw that there were two votes for the medium gray classification, and one vote for the dark gray. However, the dark gray record is closer

than the other two records. Will this greater proximity be enough for the influence of the dark gray record to overcome that of the more numerous medium gray records?

Assume that the records in question have the values for age and Na/K ratio given in Table 7.3, which also shows the min-max normalizations for these values. Then the distances of records A, B, and C from the new record are as follows:

$$d(\text{new}, A) = \sqrt{(0.05 - 0.0467)^2 + (0.25 - 0.2471)^2} = 0.004393$$

$$d(\text{new}, B) = \sqrt{(0.05 - 0.0533)^2 + (0.25 - 0.1912)^2} = 0.58893$$

$$d(\text{new}, C) = \sqrt{(0.05 - 0.0917)^2 + (0.25 - 0.2794)^2} = 0.051022$$

The votes of these records are then weighted according to the inverse square of their distances.

One record (A) votes to classify the new record as dark gray (drugs B and C), so the weighted vote for this classification is

$$\text{votes (dark gray)} = \frac{1}{d(\text{new}, A)^2} = \frac{1}{0.004393^2} \cong 51,818$$

Two records (B and C) vote to classify the new record as medium gray (drugs A and X), so the weighted vote for this classification is

$$\begin{aligned} \text{votes (medium gray)} &= \frac{1}{d(\text{new}, B)^2} + \frac{1}{d(\text{new}, C)^2} \\ &= \frac{1}{0.58893^2} + \frac{1}{0.051022^2} \cong 672 \end{aligned}$$

Therefore, by the convincing total of 51,818 to 672, the weighted voting procedure would choose dark gray (drugs B and C) as the classification for a new 17-year-old patient with a Na/K ratio of 12.5. Note that this conclusion reverses the earlier classification for the unweighted $k = 3$ case, which chose the medium gray classification.

When the distance is zero, the inverse would be undefined. In this case the algorithm should choose the majority classification of all records whose distance is zero from the new record.

Consider for a moment that once we begin weighting the records, there is no theoretical reason why we could not increase k arbitrarily so that all existing records are included in the weighting. However, this runs up against the practical consideration of very slow computation times for calculating the weights of all of the records every time a new record needs to be classified.

TABLE 7.3 Age and Na/K ratios for records from Figure 5.4

Record	Age	Na/K	Age _{MMN}	Na/K _{MMN}
New	17	12.5	0.05	0.25
A (dark gray)	16.8	12.4	0.0467	0.2471
B (medium gray)	17.2	10.5	0.0533	0.1912
C (medium gray)	19.5	13.5	0.0917	0.2794

7.5 QUANTIFYING ATTRIBUTE RELEVANCE: STRETCHING THE AXES

Consider that not all attributes may be relevant to the classification. In decision trees (Chapter 8), for example, only those attributes that are helpful to the classification are considered. In the *k*-nearest neighbor algorithm, the distances are by default calculated on all the attributes. It is possible, therefore, for relevant records that are proximate to the new record in all the important variables, but are distant from the new record in unimportant ways, to have a moderately large distance from the new record, and therefore not be considered for the classification decision. Analysts may therefore consider restricting the algorithm to fields known to be important for classifying new records, or at least to blind the algorithm to known irrelevant fields.

Alternatively, rather than restricting fields a priori, the data analyst may prefer to indicate which fields are of more or less importance for classifying the target variable. This can be accomplished using a *cross-validation approach* or one based on domain expert knowledge. First, note that the problem of determining which fields are more or less important is equivalent to finding a coefficient z_j by which to multiply the j th axis, with larger values of z_j associated with more important variable axes. This process is therefore termed *stretching the axes*.

The cross-validation approach then selects a random subset of the data to be used as a training set and finds the set of values z_1, z_2, \dots, z_m that minimize the classification error on the test data set. Repeating the process will lead to a more accurate set of values z_1, z_2, \dots, z_m . Otherwise, domain experts may be called upon to recommend a set of values for z_1, z_2, \dots, z_m . In this way, the *k*-nearest neighbor algorithm may be made more precise.

For example, suppose that either through cross-validation or expert knowledge, the Na/K ratio was determined to be three times as important as age for drug classification. Then we would have $z_{\text{Na/K}} = 3$ and $z_{\text{age}} = 1$. For the example above, the new distances of records A, B, and C from the new record would be as follows:

$$d(\text{new}, A) = \sqrt{(0.05 - 0.0467)^2 + [3(0.25 - 0.2471)]^2} = 0.009305$$

$$d(\text{new}, B) = \sqrt{(0.05 - 0.0533)^2 + [3(0.25 - 0.1912)]^2} = 0.17643$$

$$d(\text{new}, C) = \sqrt{(0.05 - 0.0917)^2 + [3(0.25 - 0.2794)]^2} = 0.09756$$

In this case, the classification would not change with the stretched axis for Na/K, remaining dark gray. In real-world problems, however, axis stretching can lead to more accurate classifications, since it represents a method for quantifying the relevance of each variable in the classification decision.

7.6 DATABASE CONSIDERATIONS

For instance-based learning methods such as the *k*-nearest neighbor algorithm, it is vitally important to have access to a rich database full of as many different combinations of attribute values as possible. It is especially important that rare classifications

be represented sufficiently, so that the algorithm does not only predict common classifications. Therefore, the data set would need to be *balanced*, with a sufficiently large percentage of the less common classifications. One method to perform balancing is to reduce the proportion of records with more common classifications.

Maintaining this rich database for easy access may become problematic if there are restrictions on main memory space. Main memory may fill up, and access to auxiliary storage is slow. Therefore, if the database is to be used for *k*-nearest neighbor methods only, it may be helpful to retain only those data points that are near a classification “boundary.” For example, in Figure 7.1, all records with Na/K ratio value greater than, say, 19 could be omitted from the database without loss of classification accuracy, since all records in this region are classified as light gray. New records with Na/K ratio >19 would therefore be classified similarly.

7.7 *k*-NEAREST NEIGHBOR ALGORITHM FOR ESTIMATION AND PREDICTION

So far we have considered how to use the *k*-nearest neighbor algorithm for classification. However, it may be used for estimation and prediction as well as for continuous-valued target variables. One method for accomplishing this is called *locally weighted averaging*. Assume that we have the same data set as the example above, but this time rather than attempting to classify the drug prescription, we are trying to estimate the systolic blood pressure reading (BP, the target variable) of the patient, based on that patient’s *age* and *Na/K ratio* (the predictor variables). Assume that BP has a range of 80 with a minimum of 90 in the patient records.

In this example we are interested in estimating the systolic blood pressure reading for a 17-year-old patient with a Na/K ratio of 12.5, the same new patient record for which we earlier performed drug classification. If we let $k = 3$, we have the same three nearest neighbors as earlier, shown here in Table 7.4. Assume that we are using the $z_{\text{Na/K}}$ = three-axis stretching to reflect the greater importance of the Na/K ratio.

Locally weighted averaging would then estimate BP as the weighted average of BP for the $k = 3$ nearest neighbors, using the same inverse square of the distances for the weights that we used earlier. That is, the estimated target value \hat{y} is calculated as

$$\hat{y}_{\text{new}} = \frac{\sum_i w_i y_i}{\sum_i w_i}$$

TABLE 7.4 $k = 3$ nearest neighbors of the new record

Record	Age	Na/K	BP	Age _{MMN}	Na/K _{MMN}	Distance
New	17	12.5	?	0.05	0.25	–
A	16.8	12.4	120	0.0467	0.2471	0.009305
B	17.2	10.5	122	0.0533	0.1912	0.17643
C	19.5	13.5	130	0.0917	0.2794	0.26737

where $w_i = 1/d(\text{new}, x_i)^2$ for existing records x_1, x_2, \dots, x_k . Thus, in this example, the estimated systolic blood pressure reading for the new record would be

$$\hat{y}_{\text{new}} = \frac{\sum_i w_i y_i}{\sum_i w_i} = \frac{\frac{120}{0.009305^2} + \frac{122}{0.17643^2} + \frac{130}{0.09756^2}}{\frac{1}{0.009305^2} + \frac{1}{0.17643^2} + \frac{1}{0.09756^2}} = 120.0954.$$

As expected, the estimated BP value is quite close to the BP value in the present data set that is much closer (in the stretched attribute space) to the new record. In other words, since record A is closer to the new record, its BP value of 120 contributes greatly to the estimation of the BP reading for the new record.

7.8 CHOOSING k

How should one go about choosing the value of k ? In fact, there may not be an obvious best solution. Consider choosing a small value for k . Then it is possible that the classification or estimation may be unduly affected by outliers or unusual observations (“noise”). With small k (e.g., $k = 1$), the algorithm will simply return the target value of the nearest observation, a process that may lead the algorithm toward overfitting, tending to memorize the training data set at the expense of generalizability.

On the other hand, choosing a value of k that is not too small will tend to smooth out any idiosyncratic behavior learned from the training set. However, if we take this too far and choose a value of k that is too large, locally interesting behavior will be overlooked. The data analyst needs to balance these considerations when choosing the value of k .

It is possible to allow the data itself to help resolve this problem, by following a cross-validation procedure similar to the earlier method for finding the optimal values z_1, z_2, \dots, z_m for axis stretching. Here we would try various values of k with different randomly selected training sets and choose the value of k that minimizes the classification or estimation error.

7.9 APPLICATION OF *k*-NEAREST NEIGHBOR ALGORITHM USING IBM/SPSS MODELER

Table 7.5 contains a small data set of 10 records excerpted from the *ClassifyRisk* data set, with predictors *age*, *marital status*, and *income*, and target variable *risk*. We seek the k -nearest neighbor for record 10, using $k = 2$. Modeler’s results are shown in Figure 7.5. (Note that Modeler automatically normalizes the data.) Records 8 and 9 are the two nearest neighbors to record 10, with the same marital status, and somewhat similar ages. Since both records 8 and 9 are classified as good risk, the prediction for record 10 would be good risk as well.

TABLE 7.5 Find the *k*-nearest neighbor for record #10

Record	Age	Marital	Income	Risk
1	22	Single	\$46,156.98	Bad loss
2	33	Married	\$24,188.10	Bad loss
3	28	Other	\$28,787.34	Bad loss
4	51	Other	\$23,886.72	Bad loss
5	25	Single	\$47,281.44	Bad loss
6	39	Single	\$33,994.90	Good risk
7	54	Single	\$28,716.50	Good risk
8	55	Married	\$49,186.75	Good risk
9	50	Married	\$46,726.50	Good risk
10	66	Married	\$36,120.34	Good risk

THE R ZONE

Install the package needed to run KNN

```
install.packages("class")
library(class)
```

Create the data set using Table 7.3

```
new <- c(0.05,0.25)
A <- c(0.0467, 0.2471)
B <- c(0.0533, 0.1912)
C <- c(0.0917, 0.2794)
data <- rbind(A, B, C)
dimnames(data) <- list(c("Dark", "Medium", "Medium"),
  c("Age (MMN)", "Na/K (MMN)"))
# Declare true classifications of A, B, and C
trueclass <- c("Dark", "Medium", "Medium")
```

Run KNN

```
knn(data,
  new,
  cl = trueclass,
  k = 3,
  prob = TRUE)
> knn(data,
+   new,
+   cl = trueclass,
+   k = 3,
+   prob = TRUE)
[1] Medium
attr(,"prob")
[1] 0.6666667
Levels: Dark Medium
```

Calculate the Euclidean distance

```
install.packages("fields")
library(fields)
together <- rbind(new, data)
# The top row of rdist are the
# distance values from New
rdist(together)
```

```
> rdist(together)
      [,1]      [,2]      [,3]      [,4]
[1,] 0.0000000001 0.0043931765 0.0588925292 0.0510220541
[2,] 0.0043931765 0.0000000001 0.0562882759 0.0553921475
[3,] 0.0588925292 0.0562882759 0.0000000001 0.0961966735
[4,] 0.0510220541 0.0553921475 0.0961966735 0.0000000001
```

Stretch the axes

```
ds_newA <- sqrt((new[1] -A[1])^2 + (3*(new[2]-A[2]))^2)
ds_newB <- sqrt((new[1] -B[1])^2 + (3*(new[2]-B[2]))^2)
ds_newC <- sqrt((new[1] -C[1])^2 + (3*(new[2]-C[2]))^2)
```

Table 7.4

```
distance <- c(ds_newA,
  ds_newB,
  ds_newC)
BP <- c(120, 122, 130)
data <- cbind(BP, data, distance)
data
```

```
> data
      BP Age (MMN) Na/k (MMN) distance
Dark  120    0.0467    0.2471 0.009304837
Medium 122    0.0533    0.1912 0.176430865
Medium 130    0.0917    0.2794 0.097560904
```

Weights = inverse of distance

```
weights <- (1/(distance^2))
sum_wi <- sum(weights)
sum_wiyi <- sum(weights*data[,1])
yhat_new <- sum_wiyi/sum_wi
yhat_new
```

```
> yhat_new
[1] 120.0954
```

EXERCISES

- 1. Clearly describe what is meant by classification.
- 2. What is meant by the term *instance-based learning*?
- 3. Make up a set of three records, each with two numeric predictor variables and one categorical target variable, so that the classification would not change regardless of the value of *k*.
- 4. Refer to Exercise 3. Alter your data set so that the classification changes for different values of *k*.

5. Refer to Exercise 4. Find the Euclidean distance between each pair of points. Using these points, verify that Euclidean distance is a true distance metric.
6. Compare the advantages and drawbacks of unweighted versus weighted voting.
7. Why does the database need to be balanced?
8. The example in the text regarding using the *k*-nearest neighbor algorithm for estimation has the closest record, overwhelming the other records in influencing the estimation. Suggest two creative ways that we could dilute this strong influence of the closest record.
9. Discuss the advantages and drawbacks of using a small value versus a large value for *k*.
10. Why would one consider stretching the axes?
11. What is locally weighted averaging, and how does it help in estimation?

HANDS-ON ANALYSIS

12. Using the data in Table 7.5, find the *k*-nearest neighbor for record #10, using $k = 3$.
13. Using the *ClassifyRisk* data set with predictors *age*, *marital status*, and *income*, and target variable *risk*, find the *k*-nearest neighbor for record #1, using $k = 2$ and Euclidean distance.
14. Using the *ClassifyRisk* data set with predictors *age*, *marital status*, and *income*, and target variable *risk*, find the *k*-nearest neighbor for record #1, using $k = 2$ and Minkowski (city-block) distance (see Chapter 10). ■

DECISION TREES

- 8.1 WHAT IS A DECISION TREE? 165
- 8.2 REQUIREMENTS FOR USING DECISION TREES 167
- 8.3 CLASSIFICATION AND REGRESSION TREES 168
- 8.4 C4.5 ALGORITHM 174
- 8.5 DECISION RULES 179
- 8.6 COMPARISON OF THE C5.0 AND CART ALGORITHMS APPLIED TO REAL DATA 180
 - THE R ZONE 183
 - REFERENCES 184
 - EXERCISES 185
 - HANDS-ON ANALYSIS 185

8.1 WHAT IS A DECISION TREE?

In this chapter we continue our examination of classification methods for data mining. One attractive classification method involves the construction of a *decision tree*, a collection of *decision nodes*, connected by *branches*, extending downward from the *root node* until terminating in *leaf nodes*. Beginning at the root node, which by convention is placed at the top of the decision tree diagram, attributes are tested at the decision nodes, with each possible outcome resulting in a branch. Each branch then leads either to another decision node or to a terminating leaf node. Figure 8.1 provides an example of a simple decision tree.

The target variable for the decision tree in Figure 8.1 is *credit risk*, with potential customers being classified as either good or bad credit risks. The predictor variables are *savings* (low, medium, and high), *assets* (low or not low), and *income* ($\leq \$50,000$ or $> \$50,000$). Here, the root node represents a decision node, testing whether each record has a low, medium, or high savings level (as defined by the analyst or domain expert). The data set is partitioned, or *split*, according to the values of this attribute.

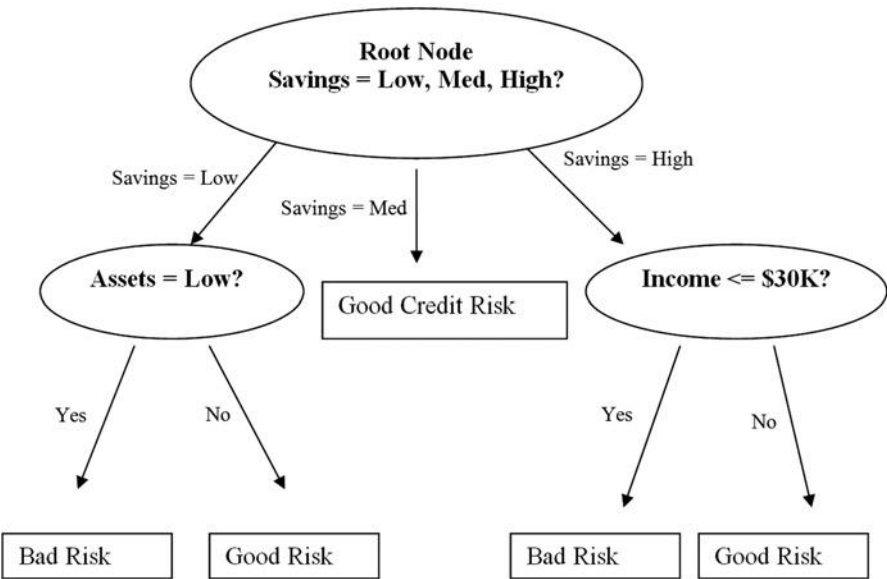


Figure 8.1 Simple decision tree.

Those records with low savings are sent via the leftmost branch (*savings = low*) to another decision node. The records with high savings are sent via the rightmost branch to a different decision node.

The records with medium savings are sent via the middle branch directly to a leaf node, indicating the termination of this branch. Why a leaf node and not another decision node? Because, in the data set (not shown), all of the records with medium savings levels have been classified as good credit risks. There is no need for another decision node, because our knowledge that the customer has medium savings predicts good credit with 100% accuracy in the data set.

For customers with low savings, the next decision node tests whether the customer has low assets. Those with low assets are then classified as bad credit risks; the others are classified as good credit risks. For customers with high savings, the next decision node tests whether the customer has an income of at most \$30,000. Customers with incomes of \$30,000 or less are then classified as bad credit risks, with the others classified as good credit risks.

When no further splits can be made, the decision tree algorithm stops growing new nodes. For example, suppose that all of the branches terminate in “pure” leaf nodes, where the target variable is unary for the records in that node (e.g., each record in the leaf node is a good credit risk). Then no further splits are necessary, so no further nodes are grown.

However, there are instances when a particular node contains “diverse” attributes (with non-unary values for the target attribute), and yet the decision tree cannot make a split. For example, suppose that we consider the records from Figure 8.1 with high savings and low income ($\leq \$30,000$). Suppose that there are five records with these values, all of which also have low assets. Finally, suppose that three of

these five customers have been classified as bad credit risks and two as good credit risks, as shown in Table 8.1. In the real world, one often encounters situations such as this, with varied values for the response variable, even for exactly the same values for the predictor variables.

Here, since all customers have the same predictor values, there is no possible way to split the records according to the predictor variables that will lead to a pure leaf node. Therefore, such nodes become diverse leaf nodes, with mixed values for the target attribute. In this case, the decision tree may report that the classification for such customers is “bad,” with 60% confidence, as determined by the three-fifths of customers in this node who are bad credit risks. Note that not all attributes are tested for all records. Customers with low savings and low assets, for example, are not tested with regard to income in this example.

8.2 REQUIREMENTS FOR USING DECISION TREES

Certain requirements must be met before decision tree algorithms may be applied:

1. Decision tree algorithms represent supervised learning, and as such require pre-classified target variables. A training data set must be supplied which provides the algorithm with the values of the target variable.
2. This training data set should be rich and varied, providing the algorithm with a healthy cross section of the types of records for which classification may be needed in the future. Decision trees learn by example, and if examples are systematically lacking for a definable subset of records, classification and prediction for this subset will be problematic or impossible.
3. The target attribute classes must be discrete. That is, one cannot apply decision tree analysis to a continuous target variable. Rather, the target variable must take on values that are clearly demarcated as either belonging to a particular class or not belonging.

Why in the example above, did the decision tree choose the *savings* attribute for the root node split? Why did it not choose *assets* or *income* instead? Decision trees seek to create a set of leaf nodes that are as “pure” as possible, that is, where each of the records in a particular leaf node has the same classification. In this way, the decision tree may provide classification assignments with the highest measure of confidence available.

TABLE 8.1 Sample of records that cannot lead to pure leaf node

Customer	Savings	Assets	Income	Credit Risk
004	High	Low	≤\$30,000	Good
009	High	Low	≤\$30,000	Good
027	High	Low	≤\$30,000	Bad
031	High	Low	≤\$30,000	Bad
104	High	Low	≤\$30,000	Bad

However, how does one measure uniformity, or conversely, how does one measure heterogeneity? We shall examine two of the many methods for measuring leaf node purity, which lead to the two leading algorithms for constructing decision trees:

- Classification and regression trees (CART) algorithm
- C4.5 algorithm

8.3 CLASSIFICATION AND REGRESSION TREES

The *classification and regression trees* (CART) method was suggested by Breiman *et al.* [1] in 1984. The decision trees produced by CART are strictly binary, containing exactly two branches for each decision node. CART recursively partitions the records in the training data set into subsets of records with similar values for the target attribute. The CART algorithm grows the tree by conducting for each decision node, an exhaustive search of all available variables and all possible splitting values, selecting the optimal split according to the following criteria (from Kennedy *et al.* [2]).

Let $\Phi(s|t)$ be a measure of the “goodness” of a candidate split s at node t , where

$$\Phi(s|t) = 2P_L P_R \sum_{j=1}^{\text{\#classes}} |P(j|t_L) - P(j|t_R)| \quad (8.1)$$

and where

$$\begin{aligned} t_L &= \text{left child node of node } t \\ t_R &= \text{right child node of node } t \\ P_L &= \frac{\text{number of records at } t_L}{\text{number of records in training set}} \\ P_R &= \frac{\text{number of records at } t_R}{\text{number of records in training set}} \\ P(j|t_L) &= \frac{\text{number of class } j \text{ records at } t_L}{\text{number of records at } t} \\ P(j|t_R) &= \frac{\text{number of class } j \text{ records at } t_R}{\text{number of records at } t} \end{aligned}$$

Then the optimal split is whichever split maximizes this measure $\Phi(s|t)$ over all possible splits at node t .

Let us look at an example. Suppose that we have the training data set shown in Table 8.2 and are interested in using CART to build a decision tree for predicting whether a particular customer should be classified as being a good or a bad credit risk. In this small example, all eight training records enter into the root node. Since CART is restricted to binary splits, the candidate splits that the CART algorithm would evaluate for the initial partition at the root node are shown in Table 8.3. Although *income* is a continuous variable, CART may still identify a finite list of possible splits

TABLE 8.2 Training set of records for classifying credit risk

Customer	Savings	Assets	Income (\$1000s)	Credit Risk
1	Medium	High	75	Good
2	Low	Low	50	Bad
3	High	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
6	High	High	25	Good
7	Low	Low	25	Bad
8	Medium	Medium	75	Good

based on the number of different values that the variable actually takes in the data set. Alternatively, the analyst may choose to categorize the continuous variable into a smaller number of classes.

For each candidate split, let us examine the values of the various components of the optimality measure $\Phi(s|t)$ in Table 8.4. Using these observed values, we may investigate the behavior of the optimality measure under various conditions. For example, when is $\Phi(s|t)$ large? We see that $\Phi(s|t)$ is large when both of its main components are large: $2P_L P_R$ and $\sum_{j=1}^{\# \text{ classes}} |P(j|t_L) - P(j|t_R)|$.

Let $Q(s|t) = \sum_{j=1}^{\# \text{ classes}} |P(j|t_L) - P(j|t_R)|$. When is the component $Q(s|t)$ large? $Q(s|t)$ is large when the distance between $P(j|t_L)$ and $P(j|t_R)$ is maximized across each class (value of the target variable). In other words, this component is maximized when the proportions of records in the child nodes for each particular value of the target variable are as different as possible. The maximum value would therefore occur when for each class the child nodes are completely uniform (pure). The theoretical maximum value for $Q(s|t)$ is k , where k is the number of classes for the target variable. Since our output variable *credit risk* takes two values, good and bad, $k = 2$ is the maximum for this component.

The component $2P_L P_R$ is maximized when P_L and P_R are large, which occurs when the proportions of records in the left and right child nodes are equal. Therefore,

TABLE 8.3 Candidate splits for $t = \text{root node}$

Candidate Split	Left Child Node, t_L	Right Child Node, t_R
1	<i>Savings</i> = low	<i>Savings</i> \in {medium, high}
2	<i>Savings</i> = medium	<i>Savings</i> \in {low, high}
3	<i>Savings</i> = high	<i>Savings</i> \in {low, medium}
4	<i>Assets</i> = low	<i>Assets</i> \in {medium, high}
5	<i>Assets</i> = medium	<i>Assets</i> \in {low, high}
6	<i>Assets</i> = high	<i>Assets</i> \in {low, medium}
7	<i>Income</i> \leq \$25,000	<i>Income</i> $>$ \$25,000
8	<i>Income</i> \leq \$50,000	<i>Income</i> $>$ \$50,000
9	<i>Income</i> \leq \$75,000	<i>Income</i> $>$ \$75,000

TABLE 8.4 Values of the components of the optimality measure $\Phi(s|t)$ for each candidate split, for the root node

Split	P_L	P_R	$P(j t_L)$	$P(j t_R)$	$2P_LP_R$	$Q(s t)$	$\Phi(s t)$
1	0.375	0.625	G: .333 B: .667	G: .8 B: .2	0.46875	0.934	0.4378
2	0.375	0.625	G: 1 B: 0	G: 0.4 B: 0.6	0.46875	1.2	0.5625
3	0.25	0.75	G: 0.5 B: 0.5	G: 0.0667 B: 0.333	0.375	0.334	0.1253
4	0.25	0.75	G: 0 B: 1	G: 0.833 B: 0.167	0.375	1.667	0.6248
5	0.5	0.5	G: 0.75 B: 0.25	G: 0.5 B: 0.5	0.5	0.5	0.25
6	0.25	0.75	G: 1 B: 0	G: 0.5 B: 0.5	0.375	1	0.375
7	0.375	0.625	G: 0.333 B: 0.667	G: 0.8 B: 0.2	0.46875	0.934	0.4378
8	0.625	0.375	G: 0.4 B: 0.6	G: 1 B: 0	0.46875	1.2	0.5625
9	0.875	0.125	G: 0.571 B: 0.429	G: 1 B: 0	0.21875	0.858	0.1877

$\Phi(s|t)$ will tend to favor balanced splits that partition the data into child nodes containing roughly equal numbers of records. Hence, the optimality measure $\Phi(s|t)$ prefers splits that will provide child nodes that are homogeneous for all classes and have roughly equal numbers of records. The theoretical maximum for $2P_LP_R$ is $2(0.5)(0.5) = 0.5$.

In this example, only candidate split 5 has an observed value for $2P_LP_R$ that reaches the theoretical maximum for this statistic, 0.5, since the records are partitioned equally into two groups of four. The theoretical maximum for $Q(s|t)$ is obtained only when each resulting child node is pure, and thus is not achieved for this data set.

The maximum observed value for $\Phi(s|t)$ among the candidate splits is therefore attained by split 4, with $\Phi(s|t) = 0.6248$. CART therefore chooses to make the initial partition of the data set using candidate split 4, assets = low versus assets $\in \{\text{medium, high}\}$, as shown in Figure 8.2.

The left child node turns out to be a terminal leaf node, since both of the records that were passed to this node had *bad* credit risk. The right child node, however, is diverse and calls for further partitioning.

We again compile a table of the candidate splits (all are available except split 4), along with the values for the optimality measure (Table 8.5). Here two candidate splits (3 and 7) share the highest value for $\Phi(s|t)$, 0.4444. We arbitrarily select the first split encountered, split 3, savings = high versus savings $\in \{\text{low, medium}\}$, for decision node A, with the resulting tree shown in Figure 8.3.

Since decision node B is diverse, we again need to seek the optimal split. Only two records remain in this decision node, each with the same value for savings

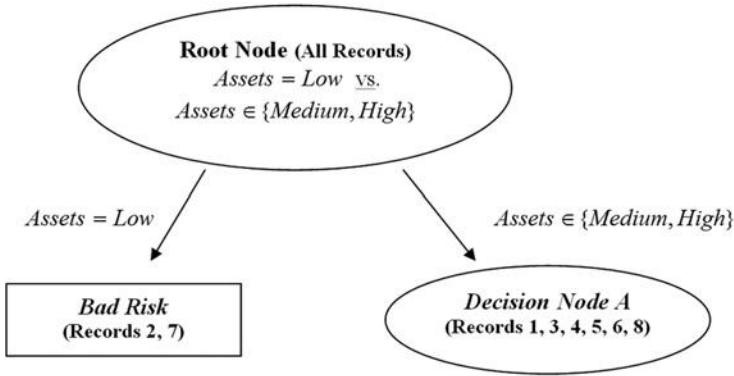


Figure 8.2 CART decision tree after initial split.

(high) and income (25). Therefore, the only possible split is assets = high versus assets = medium, providing us with the final form of the CART decision tree for this example, in Figure 8.4. Compare Figure 8.4 with Figure 8.5, the decision tree generated by *Modeler's* CART algorithm.

Let us leave aside this example now, and consider how CART would operate on an arbitrary data set. In general, CART would recursively proceed to visit each remaining decision node and apply the procedure above to find the optimal split at each node. Eventually, no decision nodes remain, and the “full tree” has been grown. However, as we have seen in Table 8.1, not all leaf nodes are necessarily homogeneous, which leads to a certain level of *classification error*.

TABLE 8.5 Values of the components of the optimality measure $\Phi(s|t)$ for each candidate split, for decision node A

Split	P_L	P_R	$P(j t_L)$	$P(j t_R)$	$2P_LP_R$	$Q(s t)$	$\Phi(s t)$
1	0.167	0.833	G: 1 B: 0	G: .8 B: .2	0.2782	0.4	0.1113
2	0.5	0.5	G: 1 B: 0	G: 0.667 B: 0.333	0.5	0.6666	0.3333
3	0.333	0.667	G: 0.5 B: 0.5	G: 1 B: 0	0.4444	1	0.4444
5	0.667	0.333	G: 0.75 B: 0.25	G: 1 B: 0	0.4444	0.5	0.2222
6	0.333	0.667	G: 1 B: 0	G: 0.75 B: 0.25	0.4444	0.5	0.2222
7	0.333	0.667	G: 0.5 B: 0.5	G: 1 B: 0	0.4444	1	0.4444
8	0.5	0.5	G: 0.667 B: 0.333	G: 1 B: 0	0.5	0.6666	0.3333
9	0.833	0.167	G: 0.8 B: 0.2	G: 1 B: 0	0.2782	0.4	0.1112

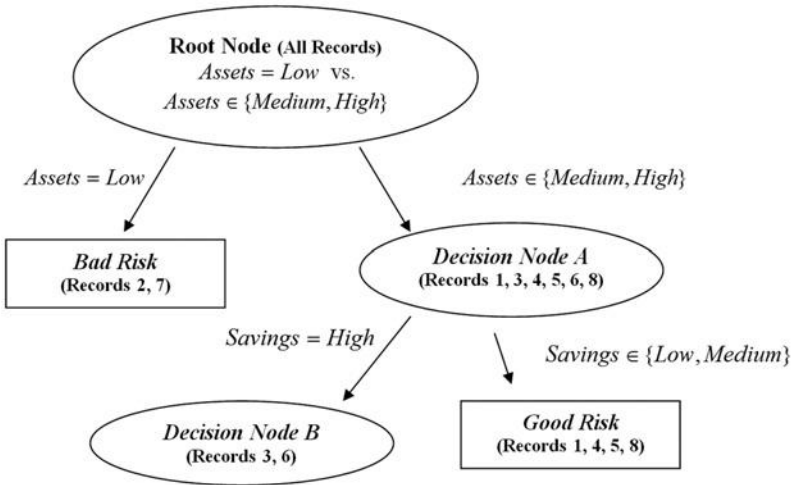


Figure 8.3 CART decision tree after decision node A split.

For example, suppose that since we cannot further partition the records in Table 8.1, we classify the records contained in this leaf node as *bad credit risk*. Then the probability that a randomly chosen record from this leaf node would be classified correctly is 0.6, since three of the five records (60%) are actually classified as bad credit risks. Hence, our *classification error rate* for this particular leaf would be 0.4 or 40%, since two of the five records are actually classified as good credit risks. CART would then calculate the error rate for the entire decision tree to be the weighted average of the individual leaf error rates, with the weights equal to the proportion of records in each leaf.

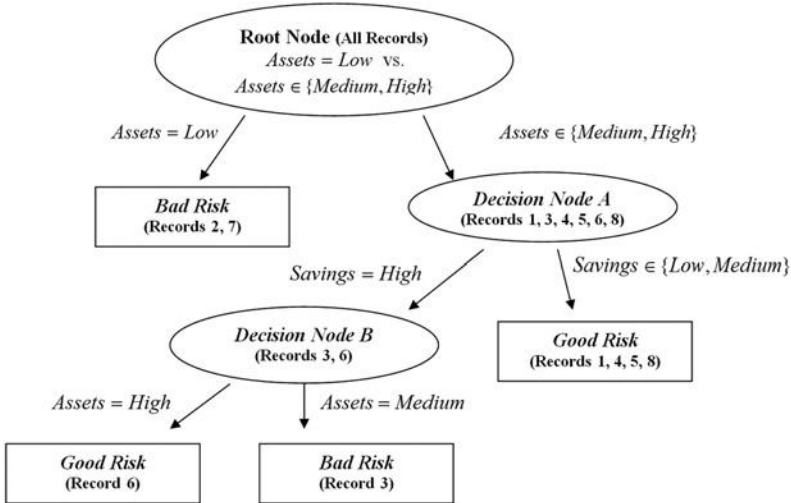


Figure 8.4 CART decision tree, fully grown form.



Figure 8.5 Modeler’s CART decision tree.

To avoid memorizing the training set, the CART algorithm needs to begin pruning nodes and branches that would otherwise reduce the generalizability of the classification results. Even though the fully grown tree has the lowest error rate on the training set, the resulting model may be too complex, resulting in overfitting. As each decision node is grown, the subset of records available for analysis becomes smaller and less representative of the overall population. Pruning the tree will increase the generalizability of the results. How the CART algorithm performs tree pruning is explained in Breiman *et al.* [1]. Essentially, an adjusted overall error rate is found that penalizes the decision tree for having too many leaf nodes and thus too much complexity.

8.4 C4.5 ALGORITHM

The C4.5 *algorithm* is Quinlan's extension of his own ID3 algorithm for generating decision trees [3]. Just as with CART, the C4.5 algorithm recursively visits each decision node, selecting the optimal split, until no further splits are possible. However, there are interesting differences between CART and C4.5:

- Unlike CART, the C4.5 algorithm is not restricted to binary splits. Whereas CART always produces a binary tree, C4.5 produces a tree of more variable shape.
- For categorical attributes, C4.5 by default produces a separate branch for each value of the categorical attribute. This may result in more “bushiness” than desired, since some values may have low frequency or may naturally be associated with other values.
- The C4.5 method for measuring node homogeneity is quite different from the CART method and is examined in detail below.

The C4.5 algorithm uses the concept of *information gain* or *entropy reduction* to select the optimal split. Suppose that we have a variable X whose k possible values have probabilities p_1, p_2, \dots, p_k . What is the smallest number of bits, on average per symbol, needed to transmit a stream of symbols representing the values of X observed? The answer is called the *entropy of X* and is defined as

$$H(X) = - \sum_j p_j \log_2(p_j)$$

Where does this formula for entropy come from? For an event with probability p , the average amount of information in bits required to transmit the result is $-\log_2(p)$. For example, the result of a fair coin toss, with probability 0.5, can be transmitted using $-\log_2(0.5) = 1$ bit, which is a zero or 1, depending on the result of the toss. For variables with several outcomes, we simply use a weighted sum of the $\log_2(p_j)$'s, with weights equal to the outcome probabilities, resulting in the formula

$$H(X) = - \sum_j p_j \log_2(p_j)$$

C4.5 uses this concept of entropy as follows. Suppose that we have a candidate split S , which partitions the training data set T into several subsets, T_1, T_2, \dots, T_k . The mean information requirement can then be calculated as the weighted sum of the entropies for the individual subsets, as follows:

$$H_S(T) = - \sum_{i=1}^k p_i H_S(T_i) \quad (8.2)$$

where p_i represents the proportion of records in subset i . We may then define our *information gain* to be $\text{gain}(S) = H(T) - H_S(T)$, that is, the increase in information produced by partitioning the training data T according to this candidate split S . At each decision node, C4.5 chooses the optimal split to be the split that has the greatest information gain, $\text{gain}(S)$.

To illustrate the C4.5 algorithm at work, let us return to the data set in Table 8.2 and apply the C4.5 algorithm to build a decision tree for classifying credit risk, just as we did earlier using CART. Once again, we are at the root node and are considering all possible splits using all the data (Table 8.6).

Now, because five of the eight records are classified as *good credit risk*, with the remaining three records classified as *bad credit risk*, the entropy before splitting is

$$H(T) = - \sum_j p_j \log_2(p_j) = -\frac{5}{8} \log_2 \left(\frac{5}{8} \right) - \frac{3}{8} \log_2 \left(\frac{3}{8} \right) = 0.9544$$

We shall compare the entropy of each candidate split against this $H(T) = 0.9544$, to see which split results in the greatest reduction in entropy (or gain in information).

For candidate split 1 (*savings*), two of the records have *high* savings, three of the records have *medium* savings, and three of the records have *low* savings, so we have: $P_{\text{high}} = \frac{2}{8}$, $P_{\text{medium}} = \frac{3}{8}$, $P_{\text{low}} = \frac{3}{8}$. Of the records with *high* savings, one is a good credit risk and one is bad, giving a probability of 0.5 of choosing the record with a good credit risk. Thus, the entropy for *high* savings is $-\frac{1}{2} \log_2 \left(\frac{1}{2} \right) - \frac{1}{2} \log_2 \left(\frac{1}{2} \right) = 1$, which is similar to the flip of a fair coin. All three of the records with *medium* savings are good credit risks, so that the entropy for *medium* is $-\frac{3}{3} \log_2 \left(\frac{3}{3} \right) - \frac{0}{3} \log_2 \left(\frac{0}{3} \right) = 0$, where by convention we define $\log_2(0) = 0$.

In engineering applications, *information* is analogous to *signal*, and *entropy* is analogous to *noise*, so it makes sense that the entropy for medium savings is zero, since the signal is crystal clear and there is no noise. If the customer has medium

TABLE 8.6 Candidate splits at root node for C4.5 algorithm

Candidate Split		Child Nodes	
1	<i>Savings</i> = low	<i>Savings</i> = medium	<i>Savings</i> = high
2	<i>Assets</i> = low	<i>Assets</i> = medium	<i>Assets</i> = high
3	<i>Income</i> ≤ \$25,000		<i>Income</i> > \$25,000
4	<i>Income</i> ≤ \$50,000		<i>Income</i> > \$50,000
5	<i>Income</i> ≤ \$75,000		<i>Income</i> > \$75,000

savings, he or she is a good credit risk, with 100% confidence. The amount of information required to transmit the credit rating of these customers is zero, as long as we know that they have medium savings.

One of the records with *low* savings is a good credit risk, and two records with *low* savings are bad credit risks, giving us our entropy for *low* credit risk as $-\frac{1}{3} \log_2 (\frac{1}{3}) - \frac{2}{3} \log_2 (\frac{2}{3}) = 0.9183$. We combine the entropies of these three subsets, using Equation (8.2) and the proportions of the subsets p_i , so that $H_{\text{savings}}(T) = \frac{2}{8}(1) + \frac{3}{8}(0) + \frac{3}{8}(0.9183) = 0.5944$. Then the information gain represented by the split on the *savings* attribute is calculated as $H(T) - H_{\text{savings}}(T) = 0.9544 - 0.5944 = 0.36$ bits.

How are we to interpret these measures? First, $H(T) = 0.9544$ means that, on average, one would need 0.9544 bit (0's or 1's) to transmit the credit risk of the eight customers in the data set. Now, $H_{\text{savings}}(T) = 0.5944$ means that the partitioning of the customers into three subsets has lowered the average bit requirement for transmitting the credit risk status of the customers to 0.5944 bits. Lower entropy is good. This *entropy reduction* can be viewed as *information gain*, so that we have gained on average $H(T) - H_{\text{savings}}(T) = 0.9544 - 0.5944 = 0.36$ bits of information by using the *savings* partition. We will compare this to the information gained by the other candidate splits, and choose the split with the largest information gain as the optimal split for the root node.

For candidate split 2 (*assets*), two of the records have *high* assets, four of the records have *medium* assets, and two of the records have *low* assets, so we have $P_{\text{high}} = \frac{2}{8}$, $P_{\text{medium}} = \frac{4}{8}$, $P_{\text{low}} = \frac{2}{8}$. Both of the records with *high* assets are classified as good credit risks, which means that the entropy for *high* assets will be zero, just as it was for *medium savings* above.

Three of the records with *medium* assets are good credit risks and one is a bad credit risk, giving us entropy $-\frac{3}{4} \log_2 (\frac{3}{4}) - \frac{1}{4} \log_2 (\frac{1}{4}) = 0.8113$. And both of the records with *low* assets are bad credit risks, which results in the entropy for *low* assets equaling zero. Combining the entropies of these three subsets, using Equation (8.2) and the proportions of the subsets p_i , we have $H_{\text{assets}}(T) = \frac{2}{8}(0) + \frac{4}{8}(0.8113) + \frac{2}{8}(0) = 0.4057$. The entropy for the *assets* split is lower than the entropy (0.5944) for the *savings* split, which indicates that the *assets* split contains less noise and is to be preferred over the *savings* split. This is measured directly using the information gain, as follows: $H(T) - H_{\text{assets}}(T) = 0.9544 - 0.4057 = 0.5487$ bits. This information gain of 0.5487 bits is larger than that for the *savings* split of 0.36 bits, verifying that the *assets* split is preferable.

While C4.5 partitions the categorical variables differently from CART, the partitions for the numerical variables are similar. Here we have four observed values for *income*: 25,000, 50,000, 75,000, and 100,000, which provide us with three thresholds for partitions, as shown in Table 8.6. For candidate split 3 from Table 8.6, *income* \leq \$25,000 versus *income* $>$ \$25,000, three of the records have *income* \leq \$25,000, with the other five records having *income* $>$ \$25,000 giving us $P_{\text{income} \leq 25,000} = \frac{3}{8}$, $P_{\text{income} > 25,000} = \frac{5}{8}$. Of the records with *income* \leq \$25,000, one is a good credit risk and two are bad, giving us the entropy for *income* \leq \$25,000 as $-\frac{1}{3} \log_2 (\frac{1}{3}) - \frac{2}{3} \log_2 (\frac{2}{3}) = 0.9183$. Four of the five records with *income* $>$ \$25,000 are good credit

risks, so that the entropy for $income > \$25,000$ is $-\frac{4}{5} \log_2(\frac{4}{5}) - \frac{1}{5} \log_2(\frac{1}{5}) = 0.7219$. Combining, we find the entropy for candidate split 3 to be $H_{income \leq \$25,000}(T) = \frac{3}{8}(0.9183) + \frac{5}{8}(0.7219) = 0.7956$. Then the information gain for this split is $H(T) - H_{income \leq \$25,000}(T) = 0.9544 - 0.7956 = 0.1588$ bits, which is our poorest choice yet.

For candidate split 4, $income \leq \$50,000$ versus $income > \$50,000$, two of the five records with $income \leq \$50,000$ are good credit risks, and three are bad, while all three of the records with $income > \$50,000$ are good credit risks. This gives us the entropy for candidate split 4 as

$$\begin{aligned} H_{income \leq \$50,000}(T) &= \frac{5}{8} \left(-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} \right) \\ &\quad + \frac{3}{8} \left(-\frac{3}{3} \log_2 \frac{3}{3} - \frac{0}{3} \log_2 \frac{0}{3} \right) = 0.6069 \end{aligned}$$

The information gain for this split is thus $H(T) - H_{income \leq \$50,000}(T) = 0.9544 - 0.6069 = 0.3475$, which is not as good as for *assets*. Finally, for candidate split 5, $income \leq \$75,000$ versus $income > \$75,000$, four of the seven records with $income \leq \$75,000$ are good credit risks, and three are bad, while the single record with $income > \$75,000$ is a good credit risk. Thus, the entropy for candidate split 4 is

$$\begin{aligned} H_{income \leq \$75,000}(T) &= \frac{7}{8} \left(-\frac{4}{7} \log_2 \frac{4}{7} - \frac{3}{7} \log_2 \frac{3}{7} \right) + \frac{1}{8} \left(-\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1} \right) \\ &= 0.8621 \end{aligned}$$

The information gain for this split is $H(T) - H_{income \leq \$75,000}(T) = 0.9544 - 0.8621 = 0.0923$, making this split the poorest of the five candidate splits.

Table 8.7 summarizes the information gain for each candidate split at the root node. Candidate split 2, *assets*, has the largest information gain, and so is chosen for the initial split by the C4.5 algorithm. Note that this choice for an optimal split concurs with the partition preferred by CART, which split on *assets = low* versus

TABLE 8.7 Information gain for each candidate split at the root node

Candidate Split	Child Nodes	Information Gain (Entropy Reduction)
1	<i>Savings = low</i> <i>Savings = medium</i> <i>Savings = high</i>	0.36 bits
2	<i>Assets = low</i> <i>Assets = medium</i> <i>Assets = high</i>	0.5487 bits
3	<i>Income ≤ \$25,000</i> <i>Income > \$25,000</i>	0.1588 bits
4	<i>Income ≤ \$50,000</i> <i>Income > \$50,000</i>	0.3475 bits
5	<i>Income ≤ \$75,000</i> <i>Income > \$75,000</i>	0.0923 bits

assets = {*medium*, *high*}. The partial decision tree resulting from C4.5’s initial split is shown in Figure 8.6.

The initial split has resulted in the creation of two terminal leaf nodes and one new decision node. Since both records with *low assets* have bad credit risk, this classification has 100% confidence, and no further splits are required. Similarly for the two records with *high assets*. However, the four records at decision node A (*assets* = *medium*) contain both *good* and *bad* credit risks, so that further splitting is called for.

We proceed to determine the optimal split for decision node A, containing records 3, 4, 5, and 8, as indicated in Table 8.8. Because three of the four records are classified as *good credit risks*, with the remaining record classified as a *bad credit risk*, the entropy before splitting is

$$H(A) = - \sum_j p_j \log_2(p_j) = -\frac{3}{4} \log_2 \left(\frac{3}{4} \right) - \frac{1}{4} \log_2 \left(\frac{1}{4} \right) = 0.8113$$

The candidate splits for decision node A are shown in Table 8.9.

For candidate split 1, *savings*, the single record with *low savings* is a good credit risk, along with the two records with *medium savings*. Perhaps counterintuitively, the single record with *high savings* is a *bad* credit risk. So the entropy for each of these three classes equals zero, since the level of savings determines the credit risk completely. This also results in a combined entropy of zero for the *assets* split, $H_{\text{assets}}(A) = 0$, which is optimal for decision node A. The information gain for this split is thus $H(A) - H_{\text{assets}}(A) = 0.8113 - 0.0 = 0.8113$. This is, of course, the maximum information gain possible for decision node A. We therefore need not continue our calculations, since no other split can result in a greater information gain.

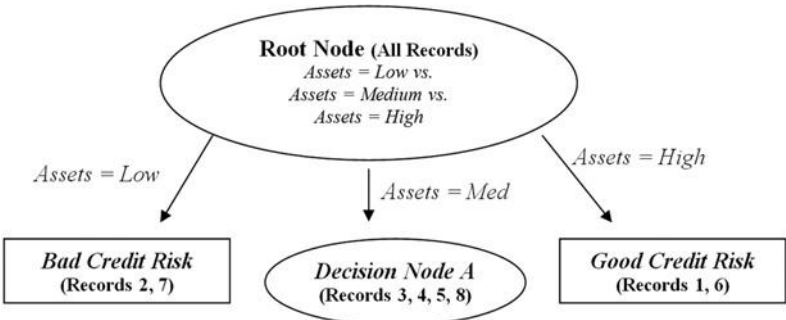


Figure 8.6 C4.5 concurs with CART in choosing *assets* for the initial partition.

TABLE 8.8 Records available at decision node A for classifying credit risk

Customer	Savings	Assets	Income (\$1000s)	Credit Risk
3	High	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
8	Medium	Medium	75	Good

TABLE 8.9 Candidate splits at decision node A

Candidate Split		Child Nodes	
1	<i>Savings = low</i>	<i>Savings = medium</i>	<i>Savings = high</i>
3	<i>Income</i> ≤ \$25,000		<i>Income</i> > \$25,000
4	<i>Income</i> ≤ \$50,000		<i>Income</i> > \$50,000
5	<i>Income</i> ≤ \$75,000		<i>Income</i> > \$75,000

As it happens, candidate split 3, *income* ≤ \$25,000 versus *income* > \$25,000, also results in the maximal information gain, but again we arbitrarily select the first such split encountered, the *savings* split.

Figure 8.7 shows the form of the decision tree after the *savings* split. Note that this is the fully grown form, since all nodes are now leaf nodes, and C4.5 will grow no further nodes. Comparing the C4.5 tree in Figure 8.7 with the CART tree in Figure 8.4, we see that the C4.5 tree is “bushier,” providing a greater breadth, while the CART tree is one level deeper. Both algorithms concur that *assets* is the most important variable (the root split) and that *savings* is also important. Finally, once the decision tree is fully grown, C4.5 engages in *pessimistic postpruning*. Interested readers may consult Kantardzic [4].

8.5 DECISION RULES

One of the most attractive aspects of decision trees lies in their interpretability, especially with respect to the construction of *decision rules*. Decision rules can be constructed from a decision tree simply by traversing any given path from the root node to any leaf. The complete set of decision rules generated by a decision tree is equivalent (for classification purposes) to the decision tree itself. For example, from the decision tree in Figure 8.7, we may construct the decision rules given in Table 8.10.

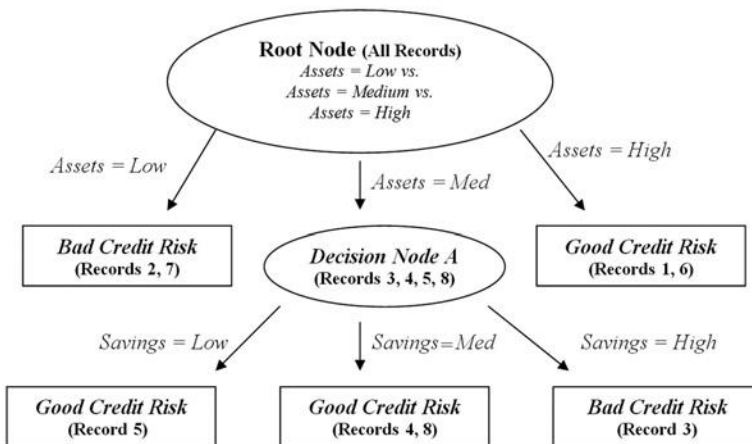


Figure 8.7 C4.5 Decision tree: fully grown form.

TABLE 8.10 Decision rules generated from decision tree in Figure 8.7

Antecedent	Consequent	Support	Confidence
If <i>assets</i> = <i>low</i>	Then bad credit risk	$\frac{2}{8}$	1.00
If <i>assets</i> = <i>high</i>	Then good credit risk	$\frac{2}{8}$	1.00
If <i>assets</i> = <i>medium</i> and <i>savings</i> = <i>low</i>	Then good credit risk	$\frac{1}{8}$	1.00
If <i>assets</i> = <i>medium</i> and <i>savings</i> = <i>medium</i>	Then good credit risk	$\frac{2}{8}$	1.00
If <i>assets</i> = <i>medium</i> and <i>savings</i> = <i>high</i>	Then bad credit risk	$\frac{1}{8}$	1.00

Decision rules come in the form *if antecedent, then consequent*, as shown in Table 8.10. For decision rules, the antecedent consists of the attribute values from the branches taken by the particular path through the tree, while the consequent consists of the classification value for the target variable given by the particular leaf node.

The *support* of the decision rule refers to the proportion of records in the data set that rest in that particular terminal leaf node. The *confidence* of the rule refers to the proportion of records in the leaf node for which the decision rule is true. In this small example, all of our leaf nodes are pure, resulting in perfect confidence levels of $100\% = 1.00$. In real-world examples, such as in the next section, one cannot expect such high confidence levels.

8.6 COMPARISON OF THE C5.0 AND CART ALGORITHMS APPLIED TO REAL DATA

Next, we apply decision tree analysis using IBM/SPSS Modeler on a real-world data set. We use a subset of the data set *adult*, which was drawn from US census data by Kohavi [5]. You may download the data set used here from the book series website, www.dataminingconsultant.com. Here we are interested in classifying whether or not a person’s income is less than \$50,000, based on the following set of predictor fields.

- Numerical variables
 - Age
 - Years of education
 - Capital gains
 - Capital losses
 - Hours worked per week
- Categorical variables
 - Race
 - Gender
 - Work class
 - Marital status

The numerical variables were normalized so that all values ranged between zero and 1. Some collapsing of low frequency classes was carried out on the *work class* and *marital status* categories. Modeler was used to compare the C5.0 algorithm (an update of the C4.5 algorithm) with CART. The decision tree produced by the CART algorithm is shown in Figure 8.8.

Here, the tree structure is displayed horizontally, with the root node at the left and the leaf nodes on the right. For the CART algorithm, the root node split is on *marital status*, with a binary split separating married persons from all others (*Marital_Status* in ["Divorced" "Never-married" "Separated" "Widowed"]). That is, this particular split on *marital status* maximized the CART split selection criterion [Equation (8.1)]:

$$\Phi(s|t) = 2P_L P_R \sum_{j=1}^{\# \text{ classes}} |P(j|t_L) - P(j|t_R)|$$

Note that the mode classification for each branch is $\leq \$50,000$. The married branch leads to a decision node, with several further splits downstream. However, the nonmarried branch is a leaf node, with a classification of $\leq \$50,000$ for the 9228 such records, with 93.7% confidence. In other words, of the 9228 persons in the data set who are not presently married, 93.7% of them have incomes of at most \$50,000.

The root node split is considered to indicate the most important single variable for classifying income. Note that the split on the *Marital_Status* attribute is binary, as are all CART splits on categorical variables. All the other splits in the full CART decision tree shown in Figure 8.8 are on numerical variables. The next decision node is *Years of education_mm*, representing the min-max normalized number of years of education. The split occurs at *Years of education_mm* ≤ 0.700 (mode $\leq \$50,000$) versus *Years of education_mm* > 0.700 (mode $> \$50,000$). However, your client may not understand what the normalized value of 0.700 represents. So, when reporting results, the analyst should always *denormalize*, to identify the original field values. The min-max normalization was of the form $X^* = \frac{X - \min(X)}{\text{range}(X)} = \frac{X - \min(X)}{\max(X) - \min(X)}$. Years of education ranged from 16 (maximum) to 1 (minimum), for a range of 15. Therefore, denormalizing, we have $X = X^* \times \text{range}(X) + \min(X) = 0.700 \times 15 + 1 = 11.5$. Thus, the split occurs at 11.5 years of education. It seems that those who have graduated high school tend to have higher incomes than those who have not.

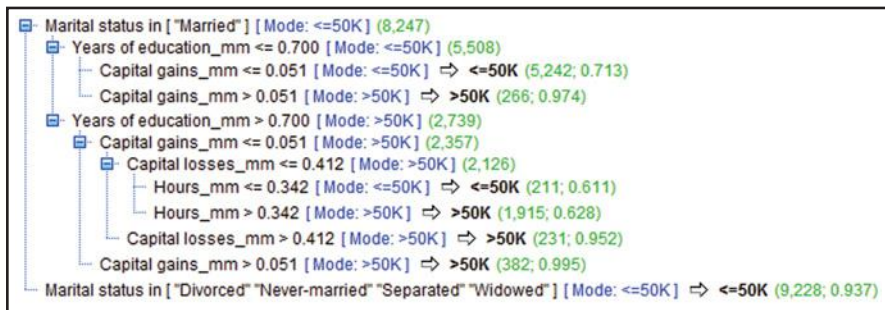


Figure 8.8 CART decision tree for the adult data set.

Interestingly, for both education groups, *Capital gains* represents the next most important decision node. For the branch with more years of education, there are two further splits, on *Capital loss*, and then *Hours* (worked per week).

Now, will the information-gain splitting criterion and the other characteristics of the C5.0 algorithm lead to a decision tree that is substantially different from or largely similar to the tree derived using CART's splitting criteria? Compare the CART decision tree above with Modeler's C5.0 decision tree of the same data displayed in Figure 8.9. (We needed to specify only three levels of splits. *Modeler* gave us eight levels of splits, which would not have fit on the page.)

Differences emerge immediately at the root node. Here, the root split is on the *Capital gains_mm* attribute, with the split occurring at the relatively low normalized level of 0.068. Since the range of capital gains in this data set is \$99,999 (maximum = 99,999, minimum = 0), this is denormalized as $X = X^* \times \text{range}(X) + \min(X) = 0.0685 \times 99,999 + 0 = \6850 . More than half of those with capital gains greater than \$6850 have incomes above \$50,000, whereas more than half of those with capital gains of less than \$6850 have incomes below \$50,000. This is the split that was chosen by the information-gain criterion as the optimal split among all possible splits over all fields. Note, however, that there are 23 times more records in the low capital gains category than in the high capital gains category (23,935 vs. 1065 records).

For records with lower capital gains, the second split occurs on *capital loss*, with a pattern similar to the earlier split on capital gains. Most people (23,179 records) had low capital loss, and most of these have incomes below \$50,000. Most of the few (756 records) who had higher capital loss had incomes above \$50,000.

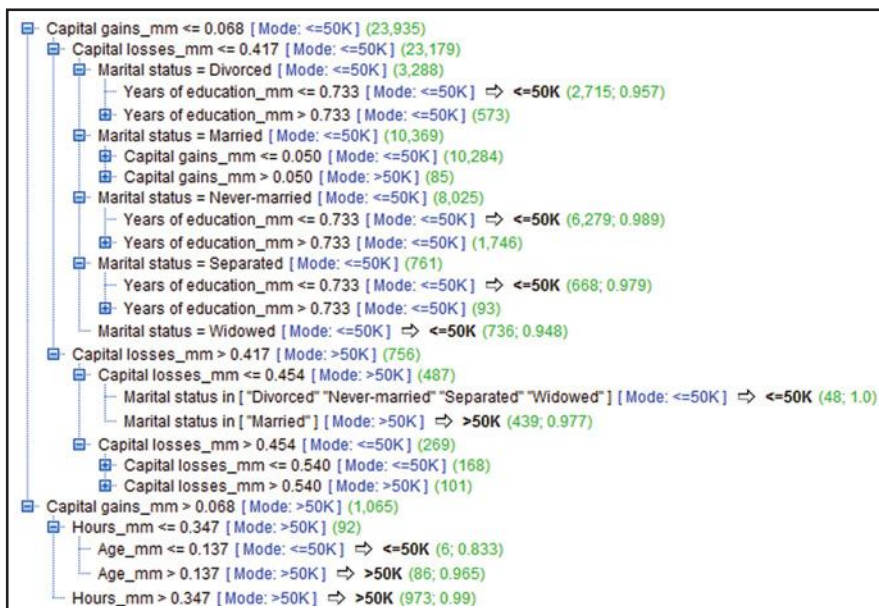


Figure 8.9 C5.0 decision tree for the adult data set.

For records with low capital gains and low capital loss, consider the next split, which is made on *marital status*. Note that C5.0 provides a separate branch for each categorical field value, whereas CART was restricted to binary splits. A possible drawback of C5.0's strategy for splitting categorical variables is that it may lead to an overly bushy tree, with many leaf nodes containing few records.

Although the CART and C5.0 decision trees do not agree in the details, we may nevertheless glean useful information from the broad areas of agreement between them. For example, the most important variables are clearly marital status, education, capital gains, capital loss, and perhaps hours per week. Both models agree that these fields are important, but disagree as to the ordering of their importance. Much more modeling analysis may be called for.

THE R ZONE

Read in the data, install and load required packages for this chapter

```
dat <- read.csv(file = "C:/.../adult.txt",
  stringsAsFactors=TRUE)
install.packages(c("rpart", "rpart.plot", "C50"))
library("rpart"); library("rpart.plot"); library("C50")
```

Collapse some of the categories by giving them the same factor label

levels(dat\$marital.status)	> levels(dat\$marital.status)
levels(dat\$workclass)	[1] "Divorced" "Married-AF-spouse"
levels(dat\$marital.status)[2:4] <-	[3] "Married-civ-spouse" "Married-spouse-absent"
"Married"	[5] "Never-married" "Separated"
levels(dat\$workclass)[c(2,3,8)] <-	[7] "widowed"
"Gov"	> levels(dat\$workclass)
levels(dat\$workclass)[c(5, 6)] <-	[1] "?" "Federal-gov" "Local-gov"
"Self"	[4] "Never-worked" "Private" "Self-emp-inc"
levels(dat\$marital.status)	[7] "Self-emp-not-inc" "State-gov" "without-pay"
levels(dat\$workclass)	> levels(dat\$marital.status)[2:4] <- "Married"
	> levels(dat\$workclass)[c(2,3,8)] <- "Gov"
	> levels(dat\$workclass)[c(5, 6)] <- "Self"
	> levels(dat\$marital.status)
	[1] "Divorced" "Married" "Never-married"
	[4] "Separated" "widowed"
	> levels(dat\$workclass)
	[1] "?" "Gov" "Never-worked"
	[4] "Private" "Self" "without-pay"

Standardize the numeric variables

```
dat$age.z <- (dat$age - mean(dat$age))/sd(dat$age)
dat$education.num.z <- (dat$education.num - mean(dat$education.num))/sd(dat$education.num)
dat$capital.gain.z <- (dat$capital.gain - mean(dat$capital.gain))/sd(dat$capital.gain)
dat$capital.loss.z <- (dat$capital.loss - mean(dat$capital.loss))/sd(dat$capital.loss)
dat$hours.per.week.z <- (dat$hours.per.week -
  mean(dat$hours.per.week))/sd(dat$hours.per.week)
```

Use predictors to classify whether or not a person's income is less than \$50K

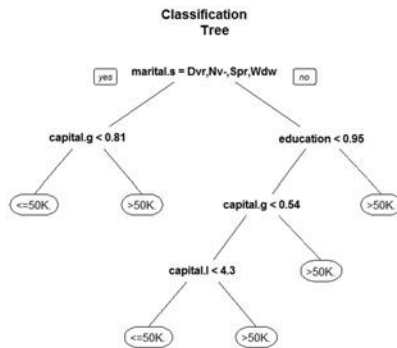
```

cartfit <- rpart(income ~ age.z + education.num.z + capital.gain.z + capital.loss.z +
  hours.per.week.z + race + sex + workclass + marital.status,
  data = dat,
  method = "class")
print(cartfit)

```

Plot the decision tree

```
rpart.plot(cartfit)
```

**# C5.0**

```

# Put the predictors into 'x', the response into 'y'
names(dat)
x <- dat[,c(2,6, 9, 10, 16, 17, 18, 19, 20)]
y <- dat$income
c50fit <- C5.0(x, y)
summary(c50fit)

```

REFERENCES

1. Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone, *Classification and Regression Trees*, Chapman & Hall/CRC Press, Boca Raton, FL, 1984.
2. Ruby L. Kennedy, Yuchun Lee, Benjamin Van Roy, Christopher D. Reed, and Richard P. Lippman, *Solving Data Mining Problems through Pattern Recognition*, Pearson Education, Upper Saddle River, NJ, 1995.
3. J. Ross Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, CA, 1992.
4. Mehmed Kantardzic, *Data Mining: Concepts, Models, Methods, and Algorithms*, 2nd edn, Wiley-Interscience, Hoboken, NJ, 2011.
5. Ronny Kohavi, Scaling up the accuracy of naive Bayes classifiers: A decision tree hybrid, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, OR, 1996.

EXERCISES

- 1. Describe the possible situations when no further splits can be made at a decision node.
- 2. Suppose that our target variable is continuous numeric. Can we apply decision trees directly to classify it? How can we work around this?
- 3. True or false: Decision trees seek to form leaf nodes to maximize heterogeneity in each node.
- 4. Discuss the benefits and drawbacks of a binary tree versus a bushier tree.

Consider the data in Table 8.11. The target variable is salary. Start by discretizing salary as follows:

- Less than \$35,000 Level 1
- \$35,000 to less than \$45,000 Level 2
- \$45,000 to less than \$55,000 Level 3
- Above \$55,000 Level 4

TABLE 8.11 Decision tree data

Occupation	Gender	Age	Salary
Service	Female	45	\$48,000
	Male	25	\$25,000
	Male	33	\$35,000
Management	Male	25	\$45,000
	Female	35	\$65,000
	Male	26	\$45,000
	Female	45	\$70,000
Sales	Female	40	\$50,000
	Male	30	\$40,000
Staff	Female	50	\$40,000
	Male	25	\$25,000

- 5. Construct a classification and regression tree to classify *salary* based on the other variables. Do as much as you can by hand, before turning to the software.
- 6. Construct a C4.5 decision tree to classify *salary* based on the other variables. Do as much as you can by hand, before turning to the software.
- 7. Compare the two decision trees and discuss the benefits and drawbacks of each.
- 8. Generate the full set of decision rules for the CART decision tree.
- 9. Generate the full set of decision rules for the C4.5 decision tree.
- 10. Compare the two sets of decision rules and discuss the benefits and drawbacks of each.

HANDS-ON ANALYSIS

For the following exercises, use the *churn* data set available at the book series website. Normalize the numerical data and deal with the correlated variables.

- 11.** Generate a CART decision tree.
- 12.** Generate a C4.5-type decision tree.
- 13.** Compare the two decision trees and discuss the benefits and drawbacks of each.
- 14.** Generate the full set of decision rules for the CART decision tree.
- 15.** Generate the full set of decision rules for the C4.5 decision tree.
- 16.** Compare the two sets of decision rules and discuss the benefits and drawbacks of each. ■

NEURAL NETWORKS

9.1	INPUT AND OUTPUT ENCODING	188
9.2	NEURAL NETWORKS FOR ESTIMATION AND PREDICTION	190
9.3	SIMPLE EXAMPLE OF A NEURAL NETWORK	191
9.4	SIGMOID ACTIVATION FUNCTION	193
9.5	BACK-PROPAGATION	194
9.6	TERMINATION CRITERIA	198
9.7	LEARNING RATE	198
9.8	MOMENTUM TERM	199
9.9	SENSITIVITY ANALYSIS	201
9.10	APPLICATION OF NEURAL NETWORK MODELING	202
	THE R ZONE	204
	REFERENCES	207
	EXERCISES	207
	HANDS-ON ANALYSIS	207

The inspiration for neural networks was the recognition that complex learning systems in animal brains consisted of closely interconnected sets of neurons. Although a particular neuron may be relatively simple in structure, dense networks of interconnected neurons could perform complex learning tasks such as classification and pattern recognition. The human brain, for example, contains approximately 10^{11} neurons, each connected on average to 10,000 other neurons, making a total of $1,000,000,000,000,000 = 10^{15}$ synaptic connections. *Artificial neural networks* (hereafter, *neural networks*) represent an attempt at a very basic level to imitate the type of nonlinear learning that occurs in the networks of neurons found in nature.

As shown in Figure 9.1, a real neuron uses dendrites to gather inputs from other neurons and combines the input information, generating a nonlinear response (“firing”) when some threshold is reached, which it sends to other neurons using the

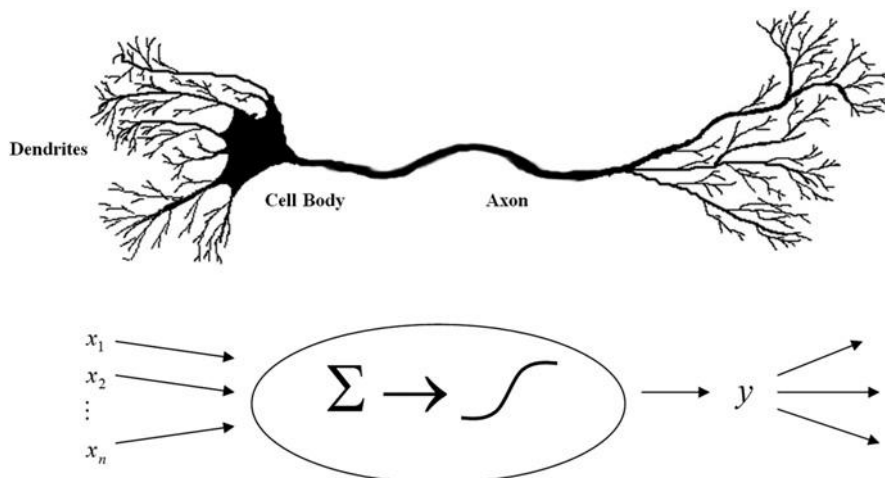


Figure 9.1 Real neuron and artificial neuron model.

axon. Figure 9.1 also shows an artificial neuron model used in most neural networks. The inputs (x_i) are collected from upstream neurons (or the data set) and combined through a combination function such as summation (Σ), which is then input into a (usually nonlinear) activation function to produce an output response (y), which is then channeled downstream to other neurons.

What types of problems are appropriate for neural networks? One of the advantages of using neural networks is that they are quite robust with respect to noisy data. Because the network contains many nodes (artificial neurons), with weights assigned to each connection, the network can learn to work around these uninformative (or even erroneous) examples in the data set. However, unlike decision trees, which produce intuitive rules that are understandable to nonspecialists, neural networks are relatively opaque to human interpretation, as we shall see. Also, neural networks usually require longer training times than decision trees, often extending into several hours.

9.1 INPUT AND OUTPUT ENCODING

One possible drawback of neural networks is that all attribute values must be encoded in a standardized manner, taking values between zero and 1, even for categorical variables. Later, when we examine the details of the back-propagation algorithm, we shall understand why this is necessary. For now, however, how does one go about standardizing all the attribute values?

For continuous variables, this is not a problem, as we discussed in Chapter 2. We may simply apply the *min-max normalization*:

$$X^* = \frac{X - \min(X)}{\text{range}(X)} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

This works well as long as the minimum and maximum values are known and all potential new data are bounded between them. Neural networks are somewhat robust to minor violations of these boundaries. If more serious violations are expected, certain ad hoc solutions may be adopted, such as rejecting values that are outside the boundaries, or assigning such values to either the minimum or maximum value.

Categorical variables are more problematical, as might be expected. If the number of possible categories is not too large, one may use *indicator (flag) variables*. For example, many data sets contain a *gender attribute*, containing values *female*, *male*, and *unknown*. Since the neural network could not handle these attribute values in their present form, we could, instead, create indicator variables for *female* and *male*. Each record would contain values for each of these two indicator variables. Records for females would have a value of 1 for *female* and 0 for *male*, while records for males would have a value of 0 for *female* and 1 for *male*. Records for persons of unknown gender would have values of 0 for *female* and 0 for *male*. In general, categorical variables with k classes may be translated into $k - 1$ indicator variables, as long as the definition of the indicators is clearly defined.

Be wary of recoding unordered categorical variables into a single variable with a range between zero and 1. For example, suppose that the data set contains information on a *marital status* attribute. Suppose that we code the attribute values *divorced*, *married*, *separated*, *single*, *widowed*, and *unknown*, as 0.0, 0.2, 0.4, 0.6, 0.8, and 1.0, respectively. Then this coding implies, for example, that *divorced* is “closer” to *married* than it is to *separated*, and so on. The neural network would be aware only of the numerical values in the *marital status* field, not of their pre-encoded meanings, and would thus be naive of their true meaning. Spurious and meaningless findings may result.

With respect to output, we shall see that neural network output nodes always return a continuous value between zero and 1 as output. How can we use such continuous output for classification?

Many classification problems have a dichotomous result, an up-or-down decision, with only two possible outcomes. For example, “Is this customer about to leave our company’s service?” For dichotomous classification problems, one option is to use a single output node (such as in Figure 9.2), with a threshold value set a priori which would separate the classes, such as “leave” or “stay.” For example, with the

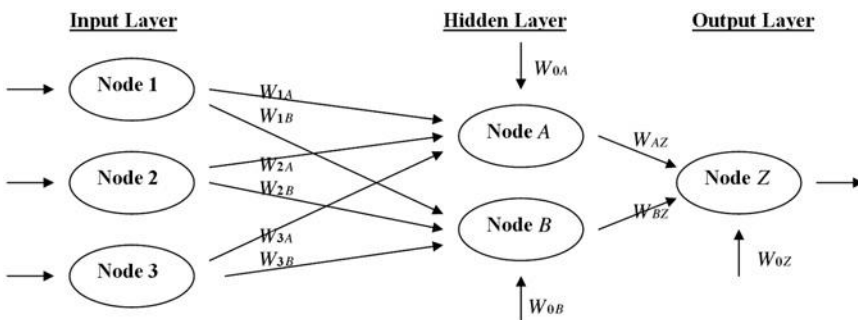


Figure 9.2 Simple neural network.

threshold of “leave if $output \geq 0.67$,” an output of 0.72 from the output node would classify that record as likely to leave the company’s service.

Single output nodes may also be used when the classes are clearly ordered. For example, suppose that we would like to classify elementary school reading prowess based on a certain set of student attributes. Then we may be able to define the thresholds as follow:

- If $0 \leq output < 0.25$, classify first-grade reading level.
- If $0.25 \leq output < 0.50$, classify second-grade reading level.
- If $0.50 \leq output < 0.75$, classify third-grade reading level.
- If $output \geq 0.75$, classify fourth-grade reading level.

Fine-tuning of the thresholds may be required, tempered by experience and the judgment of domain experts.

Not all classification problems, however, are soluble using a single output node only. For instance, suppose that we have several unordered categories in our target variable, as, for example, with the *marital status* variable above. In this case, we would choose to adopt *1-of-n output encoding*, where one output node is used for each possible category of the target variable. For example, if *marital status* was our target variable, the network would have six output nodes in the output layer, one for each of the six classes *divorced*, *married*, *separated*, *single*, *widowed*, and *unknown*. The output node with the highest value is then chosen as the classification for that particular record.

One benefit of using 1-of- n output encoding is that it provides a measure of confidence in the classification, in the form of the difference between the highest value output node and the second highest value output node. Classifications with low confidence (small difference in node output values) can be flagged for further clarification.

9.2 NEURAL NETWORKS FOR ESTIMATION AND PREDICTION

Clearly, since neural networks produce continuous output, they may quite naturally be used for estimation and prediction. Suppose, for example, that we are interested in predicting the price of a particular stock 3 months in the future. Presumably, we would have encoded price information using the min-max normalization above. However, the neural network would output a value between zero and 1, which (one would hope) does not represent the predicted price of the stock.

Rather, the min-max normalization needs to be inverted, so that the neural network output can be understood on the scale of the stock prices. In general, this denormalization is as follows:

$$\text{prediction} = \text{output}(\text{data range}) + \text{minimum}$$

where *output* represents the neural network output in the (0,1) range, *data range* represents the range of the original attribute values on the nonnormalized scale, and

minimum represents the smallest attribute value on the nonnormalized scale. For example, suppose that the stock prices ranged from \$20 to \$30 and that the network output was 0.69. Then the predicted stock price in 3 months is

$$\text{prediction} = \text{output}(\text{data range}) + \text{minimum} = 0.69(\$10) + \$20 = \$26.90$$

9.3 SIMPLE EXAMPLE OF A NEURAL NETWORK

Let us examine the simple neural network shown in Figure 9.2. A neural network consists of a *layered, feedforward, completely connected* network of artificial neurons, or *nodes*. The *feedforward* nature of the network restricts the network to a single direction of flow and does not allow looping or cycling. The neural network is composed of two or more layers, although most networks consist of three layers: an *input layer*, a *hidden layer*, and an *output layer*. There may be more than one hidden layer, although most networks contain only one, which is sufficient for most purposes. The neural network is *completely connected*, meaning that every node in a given layer is connected to every node in the next layer, although not to other nodes in the same layer. Each connection between nodes has a weight (e.g., W_{1A}) associated with it. At initialization, these weights are randomly assigned to values between zero and 1.

The number of input nodes usually depends on the number and type of attributes in the data set. The number of hidden layers and the number of nodes in each hidden layer are both configurable by the user. One may have more than one node in the output layer, depending on the particular classification task at hand.

How many nodes should one have in the hidden layer? Since more nodes in the hidden layer increases the power and flexibility of the network for identifying complex patterns, one might be tempted to have a large number of nodes in the hidden layer. On the other hand, an overly large hidden layer leads to overfitting, memorizing the training set at the expense of generalizability to the validation set. If overfitting is occurring, one may consider reducing the number of nodes in the hidden layer; conversely, if the training accuracy is unacceptably low, one may consider increasing the number of nodes in the hidden layer.

The input layer accepts inputs from the data set, such as attribute values, and simply passes these values along to the hidden layer without further processing. Thus, the nodes in the input layer do not share the detailed node structure that the hidden layer nodes and the output layer nodes share.

We will investigate the structure of hidden layer nodes and output layer nodes using the sample data provided in Table 9.1. First, a *combination function* (usually

TABLE 9.1 Data inputs and initial values for neural network weights

$x_0 = 1.0$	$W_{0A} = 0.5$	$W_{0B} = 0.7$	$W_{0Z} = 0.5$
$x_1 = 0.4$	$W_{1A} = 0.6$	$W_{1B} = 0.9$	$W_{1Z} = 0.9$
$x_2 = 0.2$	$W_{2A} = 0.8$	$W_{2B} = 0.8$	$W_{2Z} = 0.9$
$x_3 = 0.7$	$W_{3A} = 0.6$	$W_{3B} = 0.4$	

summation, Σ) produces a linear combination of the node inputs and the connection weights into a single scalar value, which we will term *net*. Thus, for a given node j ,

$$\text{net}_j = \sum_i W_{ij}x_{ij} = W_{0j}x_{0j} + W_{1j}x_{1j} + \cdots + W_{Ij}x_{Ij}$$

where x_{ij} represents the i th input to node j , W_{ij} represents the weight associated with the i th input to node j , and there are $I + 1$ inputs to node j . Note that x_1, x_2, \dots, x_I represent inputs from upstream nodes, while x_0 represents a *constant* input, analogous to the constant factor in regression models, which by convention uniquely takes the value $x_{0j} = 1$. Thus, each hidden layer or output layer node j contains an “extra” input equal to a particular weight $W_{0j}x_{0j} = W_{0j}$, such as W_{0B} for node B .

For example, for node A in the hidden layer, we have

$$\begin{aligned}\text{net}_A &= \sum_i W_{iA}x_{iA} = W_{0A}(1) + W_{1A}x_{1A} + W_{2A}x_{2A} + W_{3A}x_{3A} \\ &= 0.5 + 0.6(0.4) + 0.80(0.2) + 0.6(0.7) = 1.32\end{aligned}$$

Within node A , this combination function $\text{net}_A = 1.32$ is then used as an input to an activation function. In biological neurons, signals are sent between neurons when the combination of inputs to a particular neuron cross a certain threshold, and the neuron “fires.” This is nonlinear behavior, since the firing response is not necessarily linearly related to the increment in input stimulation. Artificial neural networks model this behavior through a nonlinear activation function.

The most common activation function is the sigmoid function:

$$y = \frac{1}{1 + e^{-x}}$$

where e is base of natural logarithms, equal to about 2.718281828. Thus, within node A , the activation would take $\text{net}_A = 1.32$ as input to the sigmoid activation function, and produce an output value of $y = 1/(1 + e^{-1.32}) = 0.7892$. Node A ’s work is done (for the moment), and this output value would then be passed along the connection to the output node Z , where it would form (via another linear combination) a component of net_Z .

But before we can compute net_Z , we need to find the contribution of node B . From the values in Table 9.1, we have

$$\begin{aligned}\text{net}_B &= \sum_i W_{iB}x_{iB} = W_{0B}(1) + W_{1B}x_{1B} + W_{2B}x_{2B} + W_{3B}x_{3B} \\ &= 0.7 + 0.9(0.4) + 0.80(0.2) + 0.4(0.7) = 1.5\end{aligned}$$

Then

$$f(\text{net}_B) = \frac{1}{1 + e^{-1.5}} = 0.8176$$

Node Z then combines these outputs from nodes A and B , through net_Z , a weighted sum, using the weights associated with the connections between these

nodes. Note that the inputs x_i to node Z are not data attribute values but the outputs from the sigmoid functions from upstream nodes:

$$\begin{aligned}\text{net}_Z &= \sum_i W_{iZ}x_{iZ} = W_{0Z}(1) + W_{AZ}x_{AZ} + W_{BZ}x_{BZ} \\ &= 0.5 + 0.9(0.7892) + 0.9(0.8176) = 1.9461\end{aligned}$$

Finally, net_Z is input into the sigmoid activation function in node Z , resulting in

$$f(\text{net}_Z) = \frac{1}{1 + e^{-1.9461}} = 0.8750$$

This value of 0.8750 is the *output* from the neural network for this first pass through the network, and represents the value predicted for the target variable for the first observation.

9.4 SIGMOID ACTIVATION FUNCTION

Why use the sigmoid function? Because it combines nearly linear behavior, curvilinear behavior, and nearly constant behavior, depending on the value of the input. Figure 9.3 shows the graph of the sigmoid function $y = f(x) = 1/(1 + e^{-x})$, for $-5 < x < 5$ (although $f(x)$ may theoretically take any real-valued input). Through much of the center of the domain of the input x (e.g., $-1 < x < 1$), the behavior of $f(x)$ is nearly linear. As the input moves away from the center, $f(x)$ becomes curvilinear. By the time the input reaches extreme values, $f(x)$ becomes nearly constant.

Moderate increments in the value of x produce varying increments in the value of $f(x)$, depending, on the location of x . Near the center, moderate increments in the value of x produce moderate increments in the value of $f(x)$; however, near the extremes, moderate increments in the value of x produce tiny increments in the value of $f(x)$. The sigmoid function is sometimes called a *squashing function*, since it takes any real-valued input and returns an output bounded between zero and 1.

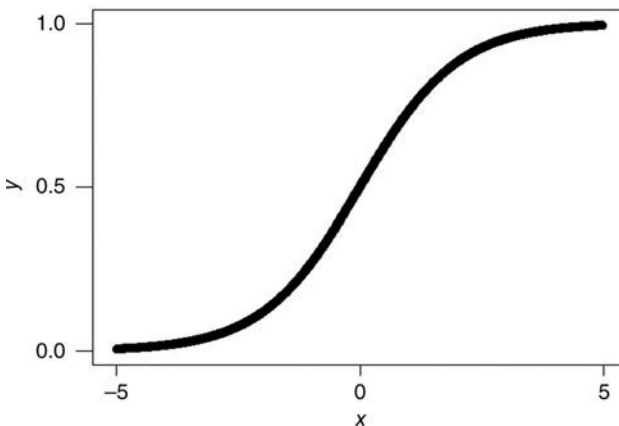


Figure 9.3 Graph of the sigmoid function $y = f(x) = 1/(1 + e^{-x})$.

9.5 BACK-PROPAGATION

How does the neural network learn? Neural networks represent a supervised learning method, requiring a large training set of complete records, including the target variable. As each observation from the training set is processed through the network, an output value is produced from the output node (assuming that we have only one output node, as in Figure 9.2). This output value is then compared to the actual value of the target variable for this training set observation, and the error (actual – output) is calculated. This prediction error is analogous to the residuals in regression models. To measure how well the output predictions fit the actual target values, most neural network models use the sum of squared errors (SSEs):

$$\text{SSE} = \sum_{\text{records}} \sum_{\text{output nodes}} (\text{actual} - \text{output})^2$$

where the squared prediction errors are summed over all the output nodes and over all the records in the training set.

The problem is therefore to construct a set of model weights that will minimize the SSE. In this way, the weights are analogous to the parameters of a regression model. The “true” values for the weights that will minimize SSE are unknown, and our task is to estimate them, given the data. However, due to the nonlinear nature of the sigmoid functions permeating the network, there exists no closed-form solution for minimizing SSE as exists for least-squares regression.

9.5.1 Gradient Descent Method

We must therefore turn to optimization methods, specifically gradient-descent methods, to help us find the set of weights that will minimize SSE. Suppose that we have a set (vector) of m weights $\mathbf{w} = w_0, w_1, w_2, \dots, w_m$ in our neural network model and we wish to find the values for each of these weights that, together, minimize SSE. We can use the gradient descent method, which *gives us the direction that we should adjust the weights* in order to decrease SSE. The gradient of SSE with respect to the vector of weights \mathbf{w} is the vector derivative:

$$\nabla \text{SSE}(\mathbf{w}) = \left[\frac{\partial \text{SSE}}{\partial w_0}, \frac{\partial \text{SSE}}{\partial w_1}, \dots, \frac{\partial \text{SSE}}{\partial w_m} \right]$$

that is, the vector of partial derivatives of SSE with respect to each of the weights.

To illustrate how gradient descent works, let us consider the case where there is only a single weight w_1 . Consider Figure 9.4, which plots the error SSE against the range of values for w_1 . We would prefer values of w_1 that would minimize the SSE. The optimal value for the weight w_1 is indicated as w_1^* . We would like to develop a rule that would help us move our current value of w_1 closer to the optimal value w_1^* as follows: $w_{\text{new}} = w_{\text{current}} + \Delta w_{\text{current}}$, where $\Delta w_{\text{current}}$ is the “change in the current location of w .”

Now, suppose that our current weight value w_{current} is near w_{1L} . Then we would like to *increase* our current weight value to bring it closer to the optimal value w_1^* . On the other hand, if our current weight value w_{current} were near w_{1R} , we would

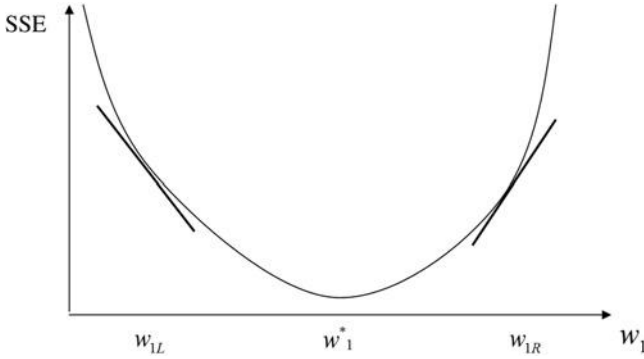


Figure 9.4 Using the slope of SSE with respect to w_1 to find weight adjustment direction.

instead prefer to *decrease* its value, to bring it closer to the optimal value w_1^* . Now the derivative $\partial \text{SSE} / \partial w_1$ is simply the slope of the SSE curve at w_1 . For values of w_1 close to w_{1L} , this slope is negative, and for values of w_1 close to w_{1R} , this slope is positive. Hence, the direction for adjusting w_{current} is the negative of the sign of the derivative of SSE at w_{current} , that is, $-\text{sign}(\partial \text{SSE} / \partial w_{\text{current}})$.

Now, how far should w_{current} be adjusted in the direction of $-\text{sign}(\partial \text{SSE} / \partial w_{\text{current}})$? Suppose that we use the magnitude of the derivative of SSE at w_{current} . When the curve is steep, the adjustment will be large, since the slope is greater in magnitude at those points. When the curve is nearly flat, the adjustment will be smaller, due to less slope. Finally, the derivative is multiplied by a positive constant η (Greek lowercase eta), called the *learning rate*, with values ranging between zero and 1. (We discuss the role of η in more detail below.) The resulting form of $\Delta w_{\text{current}}$ is as follows: $\Delta w_{\text{current}} = -\eta (\partial \text{SSE} / \partial w_{\text{current}})$, meaning that the change in the current weight value equals negative a small constant times the slope of the error function at w_{current} .

9.5.2 Back-Propagation Rules

The back-propagation algorithm takes the prediction error (actual – output) for a particular record and percolates the error back through the network, assigning partitioned responsibility for the error to the various connections. The weights on these connections are then adjusted to decrease the error, using gradient descent.

Using the sigmoid activation function and gradient descent, Mitchell [1] derives the back-propagation rules as follows:

$$w_{ij, \text{ new}} = w_{ij, \text{ current}} + \Delta w_{ij} \quad \text{where} \quad \Delta w_{ij} = \eta \delta_j x_{ij}$$

Now we know that η represents the learning rate and x_{ij} signifies the i th input to node j , but what does δ_j represent? The component δ_j represents the *responsibility* for a particular error belonging to node j . The error responsibility is computed using the partial derivative of the sigmoid function with respect to net_j and takes the following

forms, depending on whether the node in question lies in the output layer or the hidden layer:

$$\delta_j = \begin{cases} \text{output}_j(1 - \text{output}_j)(\text{actual}_j - \text{output}_j) & \text{for output layer node} \\ \text{output}_j(1 - \text{output}_j) \sum_{\text{downstream}} W_{jk} \delta_j & \text{for hidden layer nodes} \end{cases}$$

where $\sum_{\text{downstream}} W_{jk} \delta_j$ refers to the weighted sum of the error responsibilities for the nodes downstream from the particular hidden layer node. (For the full derivation, see Mitchell [1].)

Also, note that the back-propagation rules illustrate why the attribute values need to be normalized to between zero and 1. For example, if income data, with values ranging into six figures, were not normalized, the weight adjustment $\Delta w_{ij} = \eta \delta_j x_{ij}$ would be dominated by the data value x_{ij} . Hence the error propagation (in the form of δ_j) through the network would be overwhelmed, and learning (weight adjustment) would be stifled.

9.5.3 Example of Back-Propagation

Recall from our introductory example that the output from the first pass through the network was $\text{output} = 0.8750$. Assume that the actual value of the target attribute is $\text{actual} = 0.8$ and that we will use a learning rate of $\eta = 0.01$. Then the *prediction error* equals $0.8 - 0.8750 = -0.075$, and we may apply the foregoing rules to illustrate how the back-propagation algorithm works to adjust the weights by portioning out responsibility for this error to the various nodes. Although it is possible to update the weights only after all records have been read, neural networks use *stochastic* (or *online*) back-propagation, which updates the weights after each record.

First, the error responsibility δ_Z for node Z is found. Since node Z is an output node, we have

$$\begin{aligned} \delta_Z &= \text{output}_Z(1 - \text{output}_Z)(\text{actual}_Z - \text{output}_Z) \\ &= 0.8751(1 - 0.875)(0.8 - 0.875) = -0.0082 \end{aligned}$$

We may now adjust the “constant” weight W_{0Z} (which transmits an “input” of 1) using the back-propagation rules as follows:

$$\begin{aligned} \Delta W_{0Z} &= \eta \delta_Z(1) = 0.1(-0.0082)(1) = -0.00082 \\ w_{0Z, \text{new}} &= w_{0Z, \text{current}} + \Delta w_{0Z} = 0.5 - 0.00082 = 0.49918 \end{aligned}$$

Next, we move upstream to node A . Since node A is a hidden layer node, its error responsibility is

$$\delta_A = \text{output}_A(1 - \text{output}_A) \sum_{\text{downstream}} W_{jk} \delta_j$$

The only node downstream from node A is node Z . The weight associated with this connection is $W_{AZ} = 0.9$, and the error responsibility at node Z is -0.0082 , so that $\delta_A = 0.7892(1 - 0.7892)(0.9)(-0.0082) = -0.00123$.

We may now update weight W_{AZ} using the back-propagation rules as follows:

$$\begin{aligned}\Delta W_{AZ} &= \eta \delta_Z \cdot \text{output}_A = 0.1(-0.0082)(0.7892) = -0.000647 \\ w_{AZ,\text{new}} &= w_{AZ,\text{current}} + \Delta w_{AZ} = 0.9 - 0.000647 = 0.899353\end{aligned}$$

The weight for the connection between hidden layer node A and output layer node Z has been adjusted from its initial value of 0.9 to its new value of 0.899353.

Next, we turn to node B , a hidden layer node, with error responsibility

$$\delta_B = \text{output}_B(1 - \text{output}_B) \sum_{\text{downstream}} W_{jk} \delta_j$$

Again, the only node downstream from node B is node Z , giving us $\delta_B = 0.8176(1 - 0.8176)(0.9)(-0.0082) = -0.0011$.

Weight W_{BZ} may then be adjusted using the back-propagation rules as follows:

$$\begin{aligned}\Delta W_{BZ} &= \eta \delta_Z \cdot \text{output}_B = 0.1(-0.0082)(0.8176) = -0.00067 \\ w_{BZ,\text{new}} &= w_{BZ,\text{current}} + \Delta w_{BZ} = 0.9 - 0.00067 = 0.89933\end{aligned}$$

We move upstream to the connections being used as inputs to node A . For weight W_{1A} we have

$$\begin{aligned}\Delta W_{1A} &= \eta \delta_A x_1 = 0.1(-0.00123)(0.4) = -0.0000492 \\ w_{1A,\text{new}} &= w_{1A,\text{current}} + \Delta w_{1A} = 0.6 - 0.000492 = 0.5999508\end{aligned}$$

For weight W_{2A} we have

$$\begin{aligned}\Delta W_{2A} &= \eta \delta_A x_2 = 0.1(-0.00123)(0.2) = -0.0000246 \\ w_{2A,\text{new}} &= w_{2A,\text{current}} + \Delta w_{2A} = 0.8 - 0.0000246 = 0.7999754\end{aligned}$$

For weight W_{3A} we have

$$\begin{aligned}\Delta W_{3A} &= \eta \delta_A x_3 = 0.1(-0.00123)(0.7) = -0.0000861 \\ w_{3A,\text{new}} &= w_{3A,\text{current}} + \Delta w_{3A} = 0.6 - 0.0000861 = 0.5999139.\end{aligned}$$

Finally, for weight W_{0A} we have

$$\begin{aligned}\Delta W_{0A} &= \eta \delta_A (1) = 0.1(-0.00123) = -0.000123 \\ w_{0A,\text{new}} &= w_{0A,\text{current}} + \Delta w_{0A} = 0.5 - 0.000123 = 0.499877\end{aligned}$$

Adjusting weights W_{0B} , W_{1B} , W_{2B} , and W_{3B} is left as an exercise.

Note that the weight adjustments have been made based on only a single perusal of a single record. The network calculated a predicted value for the target variable, compared this output value to the actual target value, and then percolated the error in prediction throughout the network, adjusting the weights to provide a smaller prediction error. Showing that the adjusted weights result in a smaller prediction error is left as an exercise.

9.6 TERMINATION CRITERIA

The neural network algorithm would then proceed to work through the training data set, record by record, adjusting the weights constantly to reduce the prediction error. It may take many passes through the data set before the algorithm's termination criterion is met. What, then, serves as the termination criterion, or stopping criterion? If training time is an issue, one may simply set the number of passes through the data, or the amount of real time the algorithm may consume, as termination criteria. However, what one gains in short training time is probably bought with degradation in model efficacy.

Alternatively, one may be tempted to use a termination criterion that assesses when the SSE on the training data has been reduced to some low threshold level. Unfortunately, because of their flexibility, neural networks are prone to overfitting, memorizing the idiosyncratic patterns in the training set instead of retaining generalizability to unseen data.

Therefore, most neural network implementations adopt the following cross-validation termination procedure:

1. Retain part of the original data set as a holdout validation set.
2. Proceed to train the neural network as above on the remaining training data.
3. Apply the weights learned from the training data on the validation data.
4. Monitor *two sets of weights*, one “current” set of weights produced by the training data, and one “best” set of weights, as measured by the lowest SSE so far on the validation data.
5. When the current set of weights has significantly greater SSE than the best set of weights, then terminate the algorithm.

Regardless of the stopping criterion used, the neural network is not guaranteed to arrive at the optimal solution, known as the *global minimum* for the SSE. Rather, the algorithm may become stuck in a local minimum, which represents a good, if not optimal solution. In practice, this has not presented an insuperable problem.

- For example, multiple networks may be trained using different initialized weights, with the best-performing model being chosen as the “final” model.
- Second, the *online* or *stochastic* back-propagation method itself acts as a guard against getting stuck in a local minimum, since it introduces a random element to the gradient descent (see Reed and Marks [2]).
- Alternatively, a *momentum* term may be added to the back-propagation algorithm, with effects discussed below.

9.7 LEARNING RATE

Recall that the learning rate η , $0 < \eta < 1$, is a constant chosen to help us move the network weights toward a global minimum for SSE. However, what value should η take? How large should the weight adjustments be?

When the learning rate is very small, the weight adjustments tend to be very small. Thus, if η is small when the algorithm is initialized, the network will probably take an unacceptably long time to converge. Is the solution therefore to use large values for η ? Not necessarily. Suppose that the algorithm is close to the optimal solution and we have a large value for η . This large η will tend to make the algorithm overshoot the optimal solution.

Consider Figure 9.5, where W^* is the optimum value for weight W , which has current value W_{current} . According to the gradient descent rule, $\Delta w_{\text{current}} = -\eta(\partial \text{SSE} / \partial w_{\text{current}})$, W_{current} will be adjusted in the direction of W^* . But if the learning rate η , which acts as a multiplier in the formula for $\Delta w_{\text{current}}$, is too large, the new weight value W_{new} will jump right past the optimal value W^* , and may in fact end up farther away from W^* than W_{current} .

In fact, since the new weight value will then be on the opposite side of W^* , the next adjustment will again overshoot W^* , leading to an unfortunate oscillation between the two “slopes” of the valley and never settling down in the ravine (the minimum). One solution is to allow the learning rate η to change values as the training moves forward. At the start of training, η should be initialized to a relatively large value to allow the network to quickly approach the general neighborhood of the optimal solution. Then, when the network is beginning to approach convergence, the learning rate should gradually be reduced, thereby avoiding overshooting the minimum.

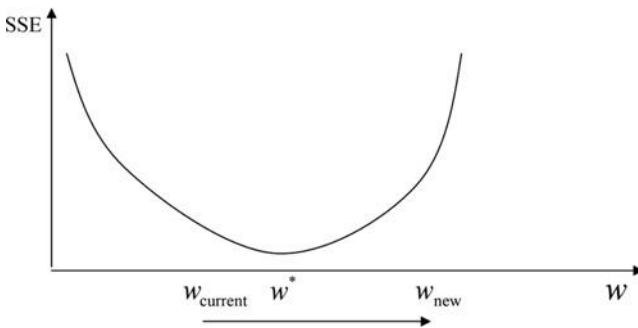


Figure 9.5 Large η may cause algorithm to overshoot global minimum.

9.8 MOMENTUM TERM

The back-propagation algorithm is made more powerful through the addition of a *momentum term* α , as follows:

$$\Delta w_{\text{current}} = -\eta \frac{\partial \text{SSE}}{\partial w_{\text{current}}} + \alpha \Delta w_{\text{previous}}$$

where $\Delta w_{\text{previous}}$ represents the previous weight adjustment, and $0 \leq \alpha < 1$. Thus, the new component $\alpha \Delta w_{\text{previous}}$ represents a fraction of the previous weight adjustment for a given weight.

Essentially, the momentum term represents *inertia*. Large values of α will influence the adjustment in the current weight, $\Delta w_{\text{current}}$, to move in the same direction as previous adjustments. It has been shown (e.g., Reed and Marks [2]) that including momentum in the back-propagation algorithm results in the adjustment becoming an exponential average of *all* previous adjustments:

$$\Delta w_{\text{current}} = -\eta \sum_{k=0}^{\infty} \alpha^k \frac{\partial \text{SSE}}{\partial w_{\text{current}-k}}$$

The α^k term indicates that the more recent adjustments exert a larger influence. Large values of α allow the algorithm to “remember” more terms in the adjustment history. Small values of α reduce the inertial effects as well as the influence of previous adjustments, until, with $\alpha = 0$, the component disappears entirely.

Clearly, a momentum component will help to dampen the oscillations around optimality mentioned earlier, by encouraging the adjustments to stay in the same direction. But momentum also helps the algorithm in the early stages of the algorithm, by increasing the rate at which the weights approach the neighborhood of optimality. This is because these early adjustments will probably all be in the same direction, so that the exponential average of the adjustments will also be in that direction. Momentum is also helpful when the gradient of SSE with respect to \mathbf{w} is flat. If the momentum term α is too large, however, the weight adjustments may again overshoot the minimum, due to the cumulative influences of many previous adjustments.

For an informal appreciation of momentum, consider Figures 9.6 and 9.7. In both figures, the weight is initialized at location I, local minima exist at locations A and C, with the optimal global minimum at B. In Figure 9.6, suppose that we have a small value for the momentum term α , symbolized by the small mass of the “ball” on the curve. If we roll this small ball down the curve, it may never make it over the first hill, and remain stuck in the first valley. That is, the small value for α enables the algorithm to easily find the first trough at location A, representing a local minimum, but does not allow it to find the global minimum at B.

Next, in Figure 9.7, suppose that we have a large value for the momentum term α , symbolized by the large mass of the “ball” on the curve. If we roll this large ball

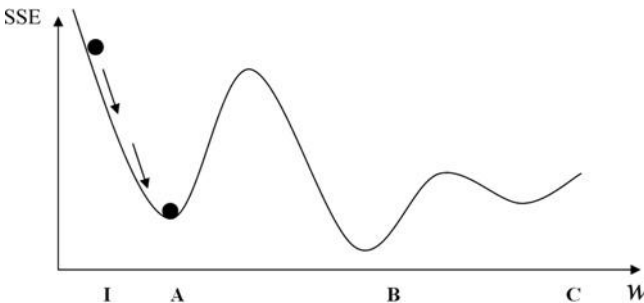


Figure 9.6 Small momentum α may cause algorithm to undershoot global minimum.

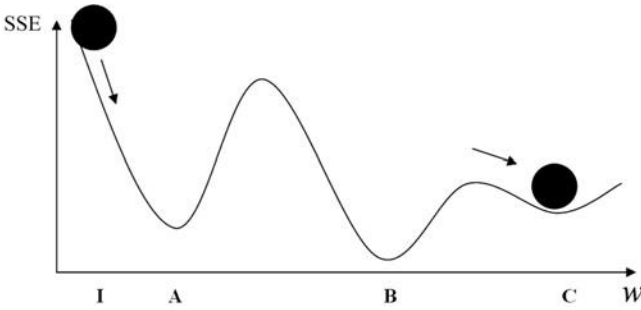


Figure 9.7 Large momentum α may cause algorithm to overshoot global minimum.

down the curve, it may well make it over the first hill but may then have so much momentum that it overshoots the global minimum at location B and settles for the local minimum at location C.

Thus, one needs to consider carefully what values to set for both the learning rate η and the momentum term α . Experimentation with various values of η and α may be necessary before the best results are obtained.

9.9 SENSITIVITY ANALYSIS

One of the drawbacks of neural networks is their opacity. The same wonderful flexibility that allows neural networks to model a wide range of nonlinear behavior also limits our ability to interpret the results using easily formulated rules. Unlike decision trees, no straightforward procedure exists for translating the weights of a neural network into a compact set of decision rules.

However, a procedure is available, called *sensitivity analysis*, which does allow us to measure the relative influence each attribute has on the output result. Using the test data set mentioned above, the sensitivity analysis proceeds as follows:

1. Generate a new observation x_{mean} , with each attribute value in x_{mean} equal to the mean of the various attribute values for all records in the test set.
2. Find the network output for input x_{mean} . Call it $\text{output}_{\text{mean}}$.
3. Attribute by attribute, vary x_{mean} to reflect the attribute minimum and maximum. Find the network output for each variation and compare it to $\text{output}_{\text{mean}}$.

The sensitivity analysis will find that varying certain attributes from their minimum to their maximum will have a greater effect on the resulting network output than it has for other attributes. For example, suppose that we are interested in predicting stock price based on *price-earnings ratio*, *dividend yield*, and other attributes. Also, suppose that varying *price-earnings ratio* from its minimum to its maximum results in an increase of 0.20 in the network output, while varying *dividend yield* from its minimum to its maximum results in an increase of 0.30 in the network output when the other attributes are held constant at their mean value. We

conclude that the network is more *sensitive* to variations in dividend yield and that therefore dividend yield is a more important factor for predicting stock prices than is price–earnings ratio.

9.10 APPLICATION OF NEURAL NETWORK MODELING

Next, we apply a neural network model using Insightful Miner on the same *adult* data set [3] from the UC Irvine Machine Learning Repository that we analyzed in Chapter 8. The Insightful Miner neural network software was applied to a training set of 25,000 cases, using a single hidden layer with eight hidden nodes. The algorithm iterated 47 epochs (runs through the data set) before termination. The resulting neural network is shown in Figure 9.8. The squares on the left represent the input nodes. For the categorical variables, there is one input node per class. The eight dark circles represent the hidden layer. The light gray circles represent the constant inputs. There is only a single output node, indicating whether or not the record is classified as having income less than or equal to \$50,000.

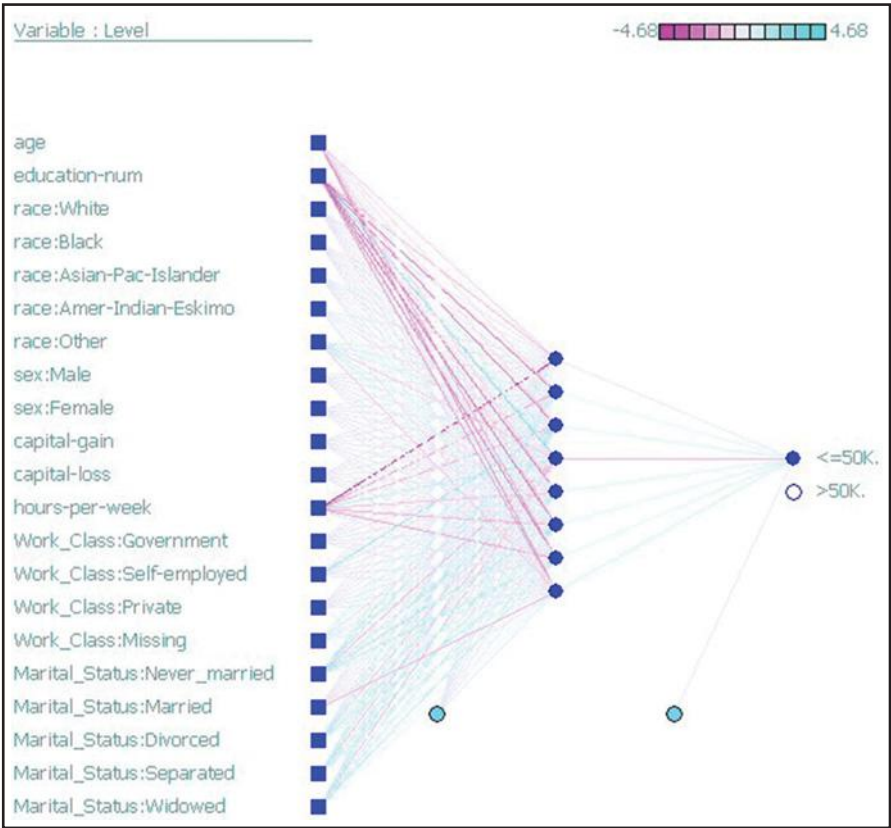


Figure 9.8 Neural network for the adult data set generated by Insightful Miner.

In this algorithm, the weights are centered at zero. An excerpt of the computer output showing the weight values is provided in Figure 9.9. The columns in the first table represent the input nodes: 1 = *age*, 2 = *education-num*, and so on, while the rows represent the hidden layer nodes: 22 = first (top) hidden node, 23 = second hidden node, and so on. For example, the weight on the connection from *age* to the topmost hidden node is -0.97 , while the weight on the connection from *Race: American Indian/Eskimo* (the sixth input node) to the last (bottom) hidden node is -0.75 . The lower section of Figure 9.9 displays the weights from the hidden nodes to the output node.


The estimated prediction accuracy using this very basic model is 82%, which is in the ballpark of the accuracies reported by Kohavi [4]. Since over 75% of the subjects have incomes at or below \$50,000, simply predicted “less than or equal to \$50,000” for every person would provide a baseline accuracy of about 75%.

However, we would like to know which variables are most important for predicting (classifying) income. We therefore perform a sensitivity analysis using Modeler, with results shown in Figure 9.10. Clearly, the amount of capital gains is the best predictor of whether a person has income less than or equal to \$50,000, followed by the number of years of education. Other important variables include the number of hours worked per week and marital status. A person’s gender does not seem to be highly predictive of income.

Weights									
To/From	1	2	3	4	5	6	7	8	9
22	-0.97	-1.32	-0.18	-0.51	0.69	0.13	-0.25	-0.33	0.30
23	-0.70	-2.97	-0.12	0.34	0.43	0.50	1.03	-0.29	-0.10
24	-0.70	-2.96	-0.24	0.05	0.16	0.46	1.15	-0.16	-0.07
25	0.74	2.86	0.22	0.41	-0.03	-0.59	-1.05	0.18	0.14
26	-0.84	-2.82	-0.23	0.02	-0.16	0.62	1.06	-0.22	-0.20
27	-0.68	-2.89	-0.18	-0.03	-0.03	0.50	1.07	-0.24	-0.12
28	-1.68	-2.54	-0.43	-0.09	0.04	0.54	0.88	-0.18	-0.26
29	-2.11	-1.95	0.01	0.34	0.04	-0.75	-1.16	-0.03	0.38

Weights									
To/From	22	23	24	25	26	27	28	29	0
30	0.18	0.59	0.69	-1.40	0.77	0.76	0.74	1.06	-0.08

Figure 9.9 Some of the neural network weights for the income example.



Relative Importance of Inputs

capital-gain	0.719519
education-num	0.486229
hours-per-week	0.289301
Marital_Status	0.27691
age	0.237282
capital-loss	0.228844
race	0.183006
Work_Class	0.119079
sex	0.0641384

Figure 9.10 Most important variables: results from sensitivity analysis.

Of course, there is much more involved with developing a neural network classification model. For example, further data preprocessing may be called for; the model would need to be validated using a holdout validation data set, and so on.

THE R ZONE

Install package needed for this chapter

```
install.packages("neuralnet")
library(neuralnet)
```

Read in the data, shrink the data set to 500 records for a brief example

```
dat <- read.csv(file = "C:/... /adult.txt",
  stringsAsFactors=TRUE)
newdat <- dat[1:500,]
# Combine factor labels as in Chapter 8
levels(newdat$marital.status)[2:4] <- "Married"
levels(newdat$workclass)[c(2,3,8)] <- "Gov"
levels(newdat$workclass)[c(5, 6)] <- "Self"
```

Determine how many Indicator variables are needed

```
unique(newdat$income)    # One variable for income
unique(newdat$sex)        # One variable for sex
unique(newdat$race)       # Four variables for race
unique(newdat$workclass)  # Three variables for workclass
unique(newdat$marital.status) # Four variables for marital.status
```

Declare, assign indicator variables

```

newdat$race_white <- newdat$race_black <- newdat$race_as.pac.is <-
  newdat$race_am.in.esk <- newdat$wc_gov <- newdat$wc_self <- newdat$wc_priv <-
  newdat$ms_marr <- newdat$ms_div <- newdat$ms_sep <- newdat$ms_wid <-
  newdat$income2 <- newdat$sex2 <- c(rep(0, length(newdat$income)))
for (i in 1:length(newdat$income)) {
  if(newdat$income[i]!="<=50K.")
    newdat$income2[i]<-1
  if(newdat$sex[i] == "Male")
    newdat$sex2[i] <- 1
  if(newdat$race[i] == "White") newdat$race_white[i] <- 1
  if(newdat$race[i] == "Amer-Indian-Eskimo") newdat$race_am.in.esk[i] <- 1
  if(newdat$race[i] == "Asian-Pac-Islander") newdat$race_as.pac.is[i] <- 1
  if(newdat$race[i] == "Black") newdat$race_black[i] <- 1
  if(newdat$workclass[i] == "Gov") newdat$wc_gov[i] <- 1
  if(newdat$workclass[i] == "Self") newdat$wc_self[i] <- 1
  if(newdat$workclass[i] == "Private" ) newdat$wc_priv[i] <- 1
  if(newdat$marital.status[i] == "Married") newdat$ms_marr[i] <- 1
  if(newdat$marital.status[i] == "Divorced" ) newdat$ms_div[i] <- 1
  if(newdat$marital.status[i] == "Separated" ) newdat$ms_sep[i] <- 1
  if(newdat$marital.status[i] == "Widowed" ) newdat$ms_wid[i] <- 1
}

```

To use the R package neuralnet, all variables must be of type Numeric or Complex

Check the variable types.

```

class(newdat$age); class(newdat$education.num); class(newdat$capital.gain);
  class(newdat$capital.loss); class(newdat$hours.per.week)

```

Change these Integer types to Numeric types

```

newdat$age <- as.numeric(newdat$age)
newdat$education.num <- as.numeric(newdat$education.num)
newdat$capital.gain <- as.numeric(newdat$capital.gain)
newdat$capital.loss <- as.numeric(newdat$capital.loss)
newdat$hours.per.week <- as.numeric(newdat$hours.per.week)

```

Check the rest of the variables

```

class(newdat$income2); class(newdat$sex2)
class(newdat$wc_priv); class(newdat$wc_gov); class(newdat$wc_self)
class(newdat$race_am.in.esk); class(newdat$race_as.pac.is); class(newdat$race_black);
  class(newdat$race_white)
class(newdat$ms_div); class(newdat$ms_sep); class(newdat$ms_wid); class(newdat$ms_marr)

```

These do not need to be changed

Delete the variables we don't need

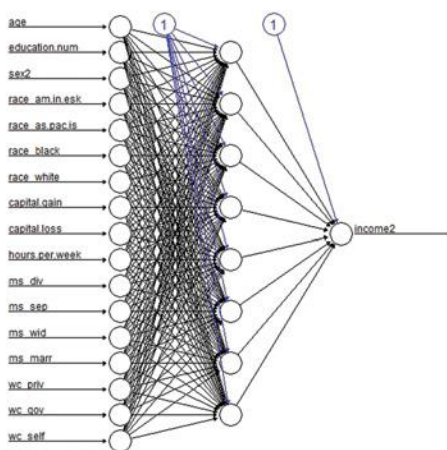
```

newdat <- newdat[, -c(2, 3, 4, 6, 7, 8, 9, 10, 14, 15)]

```

Run the neural net package

```
# Code may take some minutes to run
print(net.dat <- neuralnet(formula =
  income2 ~ age + education.num + sex2 +
  race_am.in.esk + race_as.pac.is +
  race_black + race_white + capital.gain +
  capital.loss + hours.per.week +
  ms_div + ms_sep + ms_wid + ms_marr +
  wc_priv + wc_gov + wc_self,
  data = newdat,
  rep = 10,
  hidden = 8,
  linear.output=FALSE))
# Note: If the package fails to converge,
# run the same lines again.
plot(net.dat, show.weights = FALSE)
```

**# Show the weights**

```
net.dat$weights
# In the first matrix,
# rows represent the 17
# input variables and one
# constant, while columns
# represent 8 hidden nodes.
```

```
> net.dat$weights
[[1]]
[[1]] [[1]]
      [,1] [,2] [,3] [,4]
[1,] 20.26862723968 1.98409776095 -4.21501351020 -12.6625102757
[2,] 2.38765287020 1.28655442285 0.07598583361 -11.7919510406
[3,] -5.1855529806 2.11086215444 -1.38615119306 -12.8315789302
[4,] -2.86032075459 1.15479552568 -23.55720485569 -12.1569875918
[5,] 145.43394037545 1.31334455201 -216.15493978072 -9.7583566078
[6,] 68.15219222490 0.51288518377 -88.49607577815 -779.8609668030
[7,] 18.30925574426 -1.40459327477 -354.64137873417 -278.3134848211
[8,] 20.89341534372 3.43861894093 -10.60824199593 -14.2296267695
[9,] 0.64098186374 0.02226095477 -0.97210259317 0.2875842317
[10,] 0.02247174954 0.06629281727 0.06140611074 -1.9144937039
[11,] -3.80888610724 0.12568711841 -27.09423285764 -13.8731375524
[12,] 37.95499308980 -1.14953984533 -217.50885100808 -9.7275105196
[13,] 143.24439499695 -0.57586560469 -242.25185180908 -274.5034263670
[14,] 101.20200302147 1.35360109677 -206.14641179684 -7.5798165443
[15,] -30.45187266047 1.15113413360 29.28072169454 280.9296112476
[16,] 15.37164835196 1.26358693220 15.05258143017 -147.5215948723
[17,] 68.33993182866 0.52276340064 58.01743974096 -180.0309206090
[18,] 24.04194814351 -0.36301830646 -418.98741080128 -1163.9051169387
      [,5] [,6] [,7] [,8]
[1,] 1.30255372410 -84.4634256723 1.2350412314 2.389993251084
[2,] 1.79824209242 0.6779596624 2.0235641336 2.864198749932
[3,] 0.73749225007 4.4980709029 -2.4195802111 3.369435177742
[4,] 1.26433664565 9.0794405761 2.4771845190 3.104071868407
[5,] -0.30428614127 -712.3014162891 4.2249842034 1.592874329153
[6,] 0.06838158334 15.2124814279 2.0672397196 1.874275565433
[7,] -0.07064580572 -7.4879404397 3.8932703085 2.256141936933
[8,] 0.46664940670 -1.5788330178 1.6126879878 2.370483043966
[9,] -1.84620678525 -1.8700388385 0.9127200923 -0.001013202215
[10,] 1.37378519549 1.2233383212 -1.2300424184 -1.757894195465
[11,] 0.46022152040 -0.7475249540 14.6520289711 1.845910952708
[12,] 2.50368930711 26.0563159331 3.6337255768 -2.406745572030
[13,] 0.85174247572 -708.8739065194 1.7866343716 -1.190241223912
[14,] 0.28635046928 -709.5875839125 1.8894612749 -0.341263601485
[15,] 1.10462814506 35.2338003774 2.3335174450 -0.406102118852
[16,] 0.93167150795 -2.3276156420 5.4529357228 -0.396604103267
[17,] 0.29962956184 6.4378904917 -62.8899443828 1.193303844885
[18,] 0.67729658018 18.6091379915 3.5815392142 -0.900293754216
```

```
# In the second matrix,
# rows represent the 8
# hidden nodes and one
# constant input, while the
# column represents the
# single output, income2.
```

```
[[1]][[2]]
      [,1]
[1,] 1.13981565180
[2,] -16.25557629348
[3,] -1.03901347047
[4,] 53.00991185515
[5,] 17.66250969761
[6,] -3.27460636456
[7,] 3.62327351390
[8,] 0.04100428353
[9,] 0.39716010807
```

REFERENCES

1. Tom M. Mitchell, *Machine Learning*, McGraw-Hill, New York, 1997.
2. Russell D. Reed and Robert J. MarksII, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, MIT Press, Cambridge, MA, 1999.
3. C. L. Blake and C. J. Merz, UCI Repository of Machine Learning Databases, University of California, Department of Information and Computer Science, Irvine, CA, 1998, <http://www.ics.uci.edu/~mllearn/MLRepository.html>. Accessed March 17, 2014.
4. Ronny Kohavi, Scaling up the accuracy of naïve Bayes classifiers: A decision tree hybrid, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, OR, 1996.

EXERCISES

1. Suppose that you need to prepare the data in Table 6.10 for a neural network algorithm. Define the indicator variables for the *occupation* attribute.
2. Clearly describe each of these characteristics of a neural network:
 - a. Layered
 - b. Feedforward
 - c. Completely connected
3. What is the sole function of the nodes in the input layer?
4. Should we prefer a large hidden layer or a small one? Describe the benefits and drawbacks of each.
5. Describe how neural networks function nonlinearly.
6. Explain why the updating term for the current weight includes the *negative* of the sign of the derivative (slope).
7. Adjust the weights W_{0B} , W_{1B} , W_{2B} , and W_{3B} from the example on back-propagation in the text.
8. Refer to Exercise 7. Show that the adjusted weights result in a smaller prediction error.
9. True or false: Neural networks are valuable because of their capacity for always finding the global minimum of the SSE.
10. Describe the benefits and drawbacks of using large or small values for the learning rate.
11. Describe the benefits and drawbacks of using large or small values for the momentum term.

HANDS-ON ANALYSIS

For Exercises 12–14, use the data set *churn*. Normalize the numerical data, recode the categorical variables, and deal with the correlated variables.

12. Generate a neural network model for classifying *churn* based on the other variables. Describe the topology of the model.
13. Which variables, in order of importance, are identified as most important for classifying *churn*?

14. Compare the neural network model with the CART and C4.5 models for this task in Chapter 6. Describe the benefits and drawbacks of the neural network model compared to the others. Is there convergence or divergence of results among the models?
For Exercises 15–17, use the *ClassifyRisk* data set.
15. Run an NN model predicting income based only on age. Use the default settings and make sure there is one hidden layer with one neuron.
16. Consider the following quantity: (weight for Age-to-Neuron1) + (weight for Bias-to-Neuron1) * (weight for Neuron 1-to-Output node). Explain whether this makes sense, given the data, and why.
17. Make sure the target variable takes the flag type. Compare the sign of (weight for Age-to-Neuron1) + (weight for Bias-to-Neuron1) * (weight for Neuron 1-to-Output node) for the good risk output node, as compared to the bad loss output node. Explain whether this makes sense, given the data, and why.
IBM/SPSS Modeler Analysis. For Exercises 18–19, use the *nn1* data set.
18. Set your neural network build options as follows: Use a Multilayer Perceptron and customize number of units in Hidden Layer 1 to be 1 and Hidden Layer 2 to be 0. For Stopping Rules, select ONLY Customize number of maximum training cycles. Start at 1 and go to about 20. For Advanced, deselect Replicate Results.
19. Browse your model. In the Network window of the Model tab, select the Style: Coefficients. Record the Pred1-to-Neuron1 weight and the Pred2-to-Neuron1 weight for each run. Describe the behavior of these weights. Explain why this is happening. ■

HIERARCHICAL AND k -MEANS CLUSTERING

10.1	THE CLUSTERING TASK	209
10.2	HIERARCHICAL CLUSTERING METHODS	212
10.3	SINGLE-LINKAGE CLUSTERING	213
10.4	COMPLETE-LINKAGE CLUSTERING	214
10.5	k-MEANS CLUSTERING	215
10.6	EXAMPLE OF k-MEANS CLUSTERING AT WORK	216
10.7	BEHAVIOR OF MSB, MSE, AND PSEUDO-F AS THE k-MEANS ALGORITHM PROCEEDS	219
10.8	APPLICATION OF k-MEANS CLUSTERING USING SAS ENTERPRISE MINER	220
10.9	USING CLUSTER MEMBERSHIP TO PREDICT CHURN	223
	THE R ZONE	224
	REFERENCES	226
	EXERCISES	226
	HANDS-ON ANALYSIS	226

10.1 THE CLUSTERING TASK

Clustering refers to the grouping of records, observations, or cases into classes of similar objects. A *cluster* is a collection of records that are similar to one another and dissimilar to records in other clusters. Clustering differs from classification in that there is no target variable for clustering. The clustering task does not try to classify, estimate, or predict the value of a target variable. Instead, clustering algorithms seek to segment the entire data set into relatively homogeneous subgroups or clusters,

where the similarity of the records within the cluster is maximized, and the similarity to records outside this cluster is minimized.

For example, the Nielsen PRIZM segments, developed by Claritas, Inc., represent demographic profiles of each geographic area in the United States, in terms of distinct lifestyle types, as defined by zip code. For example, the clusters identified for zip code 90210, Beverly Hills, California, are

- Cluster # 01: Upper Crust Estates
- Cluster # 03: Movers and Shakers
- Cluster # 04: Young Digerati
- Cluster # 07: Money and Brains
- Cluster # 16: Bohemian Mix

The description for Cluster # 01: Upper Crust is “The nation’s most exclusive address, Upper Crust is the wealthiest lifestyle in America, a Haven for empty-nesting couples between the ages of 45 and 64. No segment has a higher concentration of residents earning over \$100,000 a year and possessing a postgraduate degree. And none has a more opulent standard of living.”

Examples of clustering tasks in business and research include

- Target marketing of a niche product for a small-capitalization business that does not have a large marketing budget
- For accounting auditing purposes, to segment financial behavior into benign and suspicious categories
- As a dimension-reduction tool when a data set has hundreds of attributes
- For gene expression clustering, where very large quantities of genes may exhibit similar behavior

Clustering is often performed as a preliminary step in a data mining process, with the resulting clusters being used as further inputs into a different technique downstream, such as neural networks. Due to the enormous size of many present-day databases, it is often helpful to apply clustering analysis first, to reduce the search space for the downstream algorithms. In this chapter, after a brief look at hierarchical clustering methods, we discuss in detail *k*-means clustering; in Chapter 11, we examine clustering using Kohonen networks, a structure related to neural networks.

Cluster analysis encounters many of the same issues that we dealt within the chapters on classification. For example, we shall need to determine

- How to measure similarity
- How to recode categorical variables
- How to standardize or normalize numerical variables
- How many clusters we expect to uncover

For simplicity, in this book, we concentrate on Euclidean distance between records:

$$d_{\text{Euclidean}}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

where $\mathbf{x} = x_1, x_2, \dots, x_m$ and $\mathbf{y} = y_1, y_2, \dots, y_m$ represent the m attribute values of two records. Of course, many other metrics exist, such as *city-block distance*:

$$d_{\text{cityblock}}(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|$$

or *Minkowski distance*, which represents the general case of the foregoing two metrics for a general exponent q :

$$d_{\text{Minkowski}}(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|^q$$

For categorical variables, we may again define the “different from” function for comparing the i th attribute values of a pair of records:

$$\text{different}(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{otherwise} \end{cases}$$

where x_i and y_i are categorical values. We may then substitute $\text{different}(x_i, y_i)$ for the i th term in the Euclidean distance metric above.

For optimal performance, clustering algorithms, just like algorithms for classification, require the data to be normalized so that no particular variable or subset of variables dominates the analysis. Analysts may use either the *min-max normalization* or *Z-score standardization*, discussed in earlier chapters:

$$\text{Min-max normalization: } X^* = \frac{X - \min(X)}{\text{Range}(X)}$$

$$\text{Z-score standardization: } X^* = \frac{X - \text{mean}(X)}{\text{SD}(X)}$$

All clustering methods have as their goal the identification of groups of records such that similarity within a group is very high while the similarity to records in other groups is very low. In other words, as shown in Figure 10.1, clustering algorithms seek to construct clusters of records such that the *between-cluster variation* is large

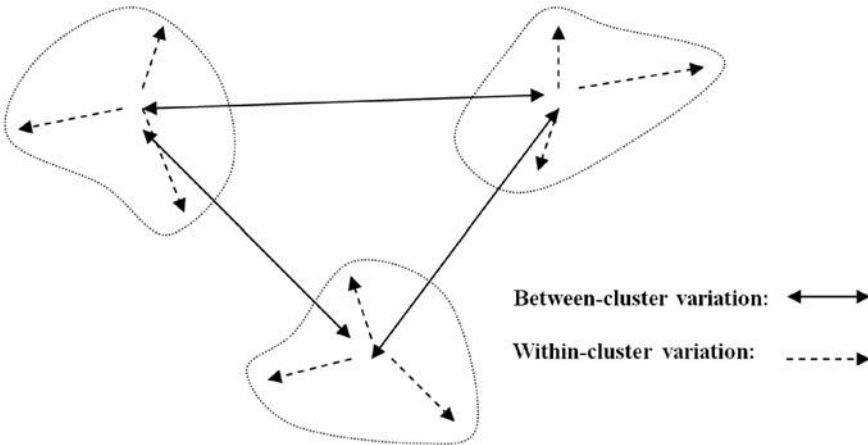


Figure 10.1 Clusters should have small within-cluster variation compared to the between-cluster variation.

compared to the *within-cluster variation*. This is somewhat analogous to the concept behind analysis of variance.

10.2 HIERARCHICAL CLUSTERING METHODS

Clustering algorithms are either hierarchical or nonhierarchical. In *hierarchical clustering*, a treelike cluster structure (*dendrogram*) is created through recursive partitioning (divisive methods) or combining (agglomerative) of existing clusters. *Agglomerative clustering methods* initialize each observation to be a tiny cluster of its own. Then, in succeeding steps, the two closest clusters are aggregated into a new combined cluster. In this way, the number of clusters in the data set is reduced by one at each step. Eventually, all records are combined into a single huge cluster. *Divisive clustering methods* begin with all the records in one big cluster, with the most dissimilar records being split off recursively, into a separate cluster, until each record represents its own cluster. Because most computer programs that apply hierarchical clustering use agglomerative methods, we focus on those.

Distance between records is rather straightforward once appropriate recoding and normalization have taken place. But how do we determine *distance between clusters* of records? Should we consider two clusters to be close if their nearest neighbors are close or if their farthest neighbors are close? How about criteria that average out these extremes?

We examine several criteria for determining distance between arbitrary clusters A and B:

- *Single linkage*, sometimes termed the *nearest-neighbor approach*, is based on the minimum distance between any record in cluster A and any record in cluster B. In other words, cluster similarity is based on the similarity of the most similar members from each cluster. Single linkage tends to form long, slender clusters, which may sometimes lead to heterogeneous records being clustered together.
- *Complete linkage*, sometimes termed the *farthest-neighbor approach*, is based on the maximum distance between any record in cluster A and any record in cluster B. In other words, cluster similarity is based on the similarity of the most dissimilar members from each cluster. Complete-linkage tends to form more compact, sphere-like clusters, with all records in a cluster within a given diameter of all other records.
- *Average linkage* is designed to reduce the dependence of the cluster-linkage criterion on extreme values, such as the most similar or dissimilar records. In average linkage, the criterion is the average distance of all the records in cluster A from all the records in cluster B. The resulting clusters tend to have approximately equal within-cluster variability.

Let us examine how these linkage methods works, using the following small, one-dimensional data set:

2 5 9 15 16 18 25 33 33 45

10.3 SINGLE-LINKAGE CLUSTERING

Suppose that we are interested in using *single-linkage* agglomerative clustering on this data set. Agglomerative methods start by assigning each record to its own cluster. Then, single linkage seeks the minimum distance between any records in two clusters. Figure 10.2 illustrates how this is accomplished for this data set. The minimum cluster distance is clearly between the single-record clusters which each contain the value 33, for which the distance must be zero for any valid metric. Thus, these two clusters are combined into a new cluster of two records, both of value 33, as shown in Figure 10.2. Note that, after step 1, only nine ($n - 1$) clusters remain. Next, in step 2, the clusters containing values 15 and 16 are combined into a new cluster, since their distance of 1 is the minimum between any two clusters remaining.

Here are the remaining steps:

- *Step 3:* The cluster containing values 15 and 16 (cluster {15, 16}) is combined with cluster {18}, since the distance between 16 and 18 (the closest records in each cluster) is two, the minimum among remaining clusters.
 - *Step 4:* Clusters {2} and {5} are combined.
 - *Step 5:* Cluster {2, 5} is combined with cluster {9}, since the distance between 5 and 9 (the closest records in each cluster) is 4, the minimum among remaining clusters.
 - *Step 6:* Cluster {2, 5, 9} is combined with cluster {15, 16, 18}, since the distance between 9 and 15 is 6, the minimum among remaining clusters.
 - *Step 7:* Cluster {2, 5, 9, 15, 16, 18} is combined with cluster {25}, since the distance between 18 and 25 is 7, the minimum among remaining clusters.
 - *Step 8:* Cluster {2, 5, 9, 15, 16, 18, 25} is combined with cluster {33, 33}, since the distance between 25 and 33 is 8, the minimum among remaining clusters.
 - *Step 9:* Cluster {2, 5, 9, 15, 16, 18, 25, 33, 33} is combined with cluster {45}.
- This last cluster now contains all the records in the data set.

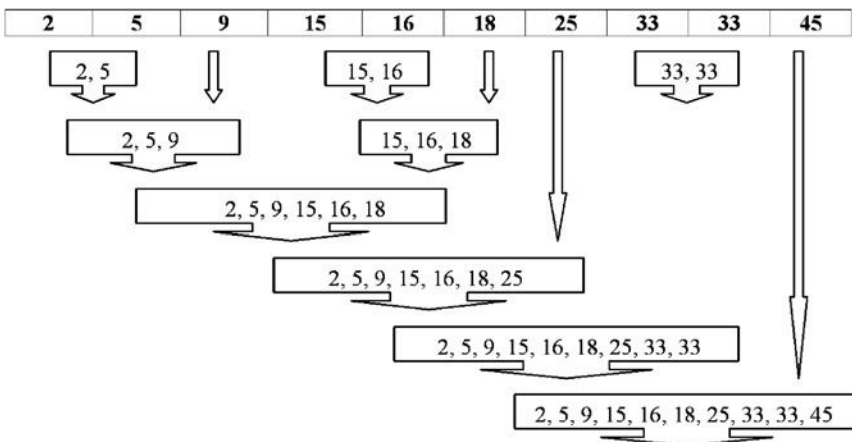


Figure 10.2 Single-linkage agglomerative clustering on the sample data set.

10.4 COMPLETE-LINKAGE CLUSTERING

Next, let us examine whether using the complete-linkage criterion would result in a different clustering of this sample data set. Complete linkage seeks to minimize the distance among the records in two clusters that are farthest from each other. Figure 10.3 illustrates complete-linkage clustering for this data set.

- *Step 1:* Since each cluster contains a single record only, there is no difference between single linkage and complete linkage at step 1. The two clusters each containing 33 are again combined.
- *Step 2:* Just as for single linkage, the clusters containing values 15 and 16 are combined into a new cluster. Again, this is because there is no difference in the two criteria for single-record clusters.
- *Step 3:* At this point, complete linkage begins to diverge from its predecessor. In single linkage, cluster {15, 16} was at this point combined with cluster {18}. But complete linkage looks at the farthest neighbors, not the nearest neighbors. The farthest neighbors for these two clusters are 15 and 18, for a distance of 3. This is the same distance separating clusters {2} and {5}. The complete-linkage criterion is silent regarding ties, so we arbitrarily select the first such combination found, therefore combining the clusters {2} and {5} into a new cluster.
- *Step 4:* Now cluster {15, 16} is combined with cluster {18}.
- *Step 5:* Cluster {2, 5} is combined with cluster {9}, since the complete-linkage distance is 7, the smallest among remaining clusters.
- *Step 6:* Cluster {25} is combined with cluster {33, 33}, with a complete-linkage distance of 8.
- *Step 7:* Cluster {2, 5, 9} is combined with cluster {15, 16, 18}, with a complete-linkage distance of 16.
- *Step 8:* Cluster {25, 33, 33} is combined with cluster {45}, with a complete-linkage distance of 20.
- *Step 9:* Cluster {2, 5, 9, 15, 16, 18} is combined with cluster {25, 33, 33, 45}. All records are now contained in this last large cluster.

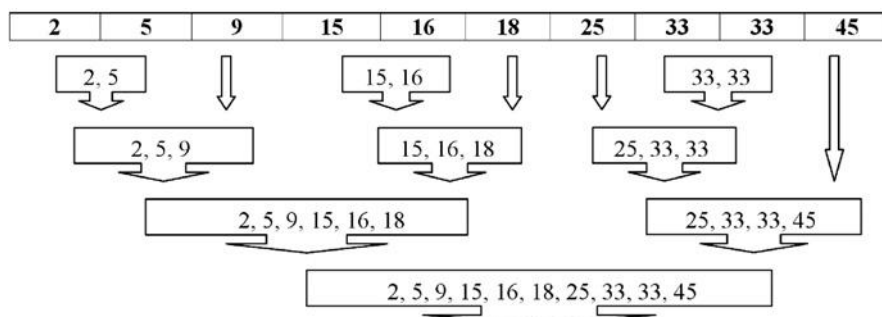


Figure 10.3 Complete-linkage agglomerative clustering on the sample data set.

Finally, with average linkage, the criterion is the average distance of all the records in cluster A from all the records in cluster B. Since the average of a single record is the record's value itself, this method does not differ from the earlier methods in the early stages, where single-record clusters are being combined. At step 3, average linkage would be faced with the choice of combining clusters {2} and {5}, or combining the {15, 16} cluster with the single-record {18} cluster. The average distance between the {15, 16} cluster and the {18} cluster is the average of $|18 - 15|$ and $|18 - 16|$, which is 2.5, while the average distance between clusters {2} and {5} is of course 3. Therefore, average linkage would combine the {15, 16} cluster with cluster {18} at this step, followed by combining cluster {2} with cluster {5}. The reader may verify that the average-linkage criterion leads to the same hierarchical structure for this example as the complete-linkage criterion. In general, average linkage leads to clusters more similar in shape to complete linkage than does single linkage.

10.5 *k*-MEANS CLUSTERING

The *k*-means clustering algorithm [1] is a straightforward and effective algorithm for finding clusters in data. The algorithm proceeds as follows:

- *Step 1:* Ask the user how many clusters k the data set should be partitioned into.
- *Step 2:* Randomly assign k records to be the initial cluster center locations.
- *Step 3:* For each record, find the nearest cluster center. Thus, in a sense, each cluster center “owns” a subset of the records, thereby representing a partition of the data set. We therefore have k clusters, C_1, C_2, \dots, C_k .
- *Step 4:* For each of the k clusters, find the cluster *centroid*, and update the location of each cluster center to the new value of the centroid.
- *Step 5:* Repeat steps 3 to 5 until convergence or termination.

The “nearest” criterion in step 3 is usually Euclidean distance, although other criteria may be applied as well. The cluster centroid in step 4 is found as follows. Suppose that we have n data points $(a_1, b_1, c_1), (a_2, b_2, c_2), \dots, (a_n, b_n, c_n)$, the *centroid* of these points is the center of gravity of these points and is located at point $(\sum a_i/n, \sum b_i/n, \sum c_i/n)$. For example, the points (1,1,1), (1,2,1), (1,3,1), and (2,1,1) would have centroid

$$\left(\frac{1+1+1+2}{4}, \frac{1+2+3+1}{4}, \frac{1+1+1+1}{4} \right) = (1.25, 1.75, 1.00)$$

The algorithm terminates when the centroids no longer change. In other words, the algorithm terminates when for all clusters C_1, C_2, \dots, C_k , all the records “owned” by each cluster center remain in that cluster. Alternatively, the algorithm may terminate when some convergence criterion is met, such as no significant shrinkage in the *mean squared error* (MSE):

$$\text{MSE} = \frac{\text{SSE}}{N - k} = \frac{\sum_{i=1}^k \sum_{p \in C_i} d(p, m_i)^2}{N - k}$$

where SSE represents the *sum of squares error*, $p \in C_i$ represents each data point in cluster i , m_i represents the centroid (cluster center) of cluster i , N is the total sample size, and k is the number of clusters. Recall that clustering algorithms seek to construct clusters of records such that the between-cluster variation is large compared to the within-cluster variation. Because this concept is analogous to the analysis of variance, we may define a *pseudo- F statistic* as follows:

$$F_{k-1,N-k} = \frac{\text{MSB}}{\text{MSE}} = \frac{\text{SSB}/k - 1}{\text{SSE}/N - k}$$

where SSE is defined as above, MSB is the *mean square between*, and SSB is the *sum of squares between* clusters, defined as

$$\text{SSB} = \sum_{i=1}^k n_i \cdot d(m_i, M)^2$$

where n_i is the number of records in cluster i , m_i is the centroid (cluster center) for cluster i , and M is the grand mean of all the data.

MSB represents the between-cluster variation and MSE represents the within-cluster variation. Thus a “good” cluster would have a large value of the pseudo- F statistic, representing a situation where the between-cluster variation is large compared to the within-cluster variation. Hence, as the k -means algorithm proceeds, and the quality of the clusters increases, we would expect MSB to increase, MSE to decrease, and F to increase.

10.6 EXAMPLE OF *k*-MEANS CLUSTERING AT WORK

Let us examine an example of how the k -means algorithm works. Suppose that we have the eight data points in two-dimensional space shown in Table 10.1 and plotted in Figure 10.4 and are interested in uncovering $k = 2$ clusters.

Let us apply the k -means algorithm step by step.

- *Step 1:* Ask the user how many clusters k the data set should be partitioned into. We have already indicated that we are interested in $k = 2$ clusters.
- *Step 2:* Randomly assign k records to be the initial cluster center locations. For this example, we assign the cluster centers to be $m_1 = (1,1)$ and $m_2 = (2,1)$.
- *Step 3 (first pass):* For each record, find the nearest cluster center. Table 10.2 contains the (rounded) Euclidean distances between each point and each cluster center $m_1 = (1,1)$ and $m_2 = (2,1)$, along with an indication of which cluster center the point is nearest to. Therefore, cluster 1 contains points $\{a, e, g\}$, and cluster 2 contains points $\{b, c, d, f, h\}$.

TABLE 10.1 Data points for k -means example

<i>A</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>E</i>	<i>f</i>	<i>g</i>	<i>h</i>
(1,3)	(3,3)	(4,3)	(5,3)	(1,2)	(4,2)	(1,1)	(2,1)

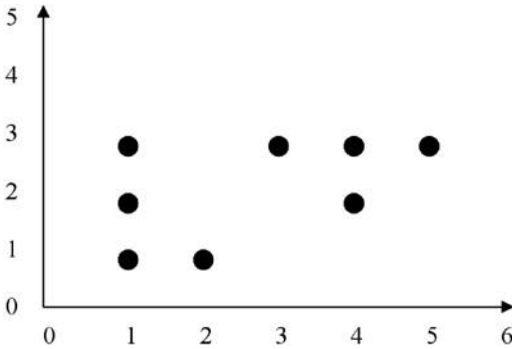


Figure 10.4 How will k -means partition these data into $k = 2$ clusters?

- *Step 4 (first pass)*: For each of the k clusters find the cluster *centroid* and update the location of each cluster center to the new value of the centroid. The centroid for cluster 1 is $[(1 + 1 + 1)/3, (3 + 2 + 1)/3] = (1, 2)$. The centroid for cluster 2 is $[(3 + 4 + 5 + 4 + 2)/5, (3 + 3 + 3 + 2 + 1)/5] = (3.6, 2.4)$. The clusters and centroids (triangles) at the end of the first pass are shown in Figure 10.5. Note that m_1 has moved up to the center of the three points in cluster 1, while m_2 has moved up and to the right a considerable distance, to the center of the five points in cluster 2.
- *Step 5*: Repeat steps 3 and 4 until convergence or termination. The centroids have moved, so we go back to step 3 for our second pass through the algorithm.
- *Step 3 (second pass)*: For each record, find the nearest cluster center. Table 10.3 shows the distances between each point and each updated cluster center $m_1 = (1, 2)$ and $m_2 = (3.6, 2.4)$, together with the resulting cluster membership. There has been a shift of a single record (h) from cluster 2 to cluster 1. The relatively large change in m_2 has left record h now closer to m_1 than to m_2 , so that record h now belongs to cluster 1. All other records remain in the same clusters as previously. Therefore, cluster 1 is $\{a, e, g, h\}$, and cluster 2 is $\{b, c, d, f\}$.
- *Step 4 (second pass)*: For each of the k clusters, find the cluster *centroid* and update the location of each cluster center to the new value of the centroid. The

TABLE 10.2 Finding the nearest cluster center for each record (first pass)

Point	Distance from m_1	Distance from m_2	Cluster Membership
a	2.00	2.24	C_1
b	2.83	2.24	C_2
c	3.61	2.83	C_2
d	4.47	3.61	C_2
e	1.00	1.41	C_1
f	3.16	2.24	C_2
g	0.00	1.00	C_1
h	1.00	0.00	C_2

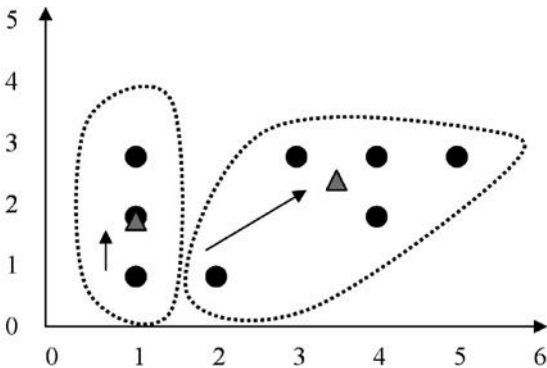


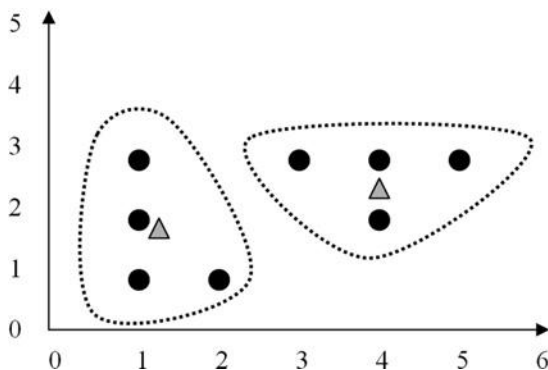
Figure 10.5 Clusters and centroids Δ after first pass through *k*-means algorithm.

new centroid for cluster 1 is $[(1 + 1 + 1 + 2)/4, (3 + 2 + 1 + 1)/4] = (1.25, 1.75)$. The new centroid for cluster 2 is $[(3 + 4 + 5 + 4)/4, (3 + 3 + 3 + 2)/4] = (4, 2.75)$. The clusters and centroids at the end of the second pass are shown in Figure 10.6. Centroids m_1 and m_2 have both moved slightly.

- *Step 5:* Repeat steps 3 and 4 until convergence or termination. Since the centroids have moved, we once again return to step 3 for our third (and as it turns out, final) pass through the algorithm.
- *Step 3 (third pass):* For each record, find the nearest cluster center. Table 10.4 shows the distances between each point and each newly updated cluster center $m_1 = (1.25, 1.75)$ and $m_2 = (4, 2.75)$, together with the resulting cluster membership. Note that no records have shifted cluster membership from the preceding pass.
- *Step 4 (third pass):* For each of the *k* clusters, find the cluster *centroid* and update the location of each cluster center to the new value of the centroid. Since no records have shifted cluster membership, the cluster centroids therefore also remain unchanged.
- *Step 5:* Repeat steps 3 and 4 until convergence or termination. Since the centroids remain unchanged, the algorithm terminates.

TABLE 10.3 Finding the nearest cluster center for each record (second pass)

Point	Distance from m_1	Distance from m_2	Cluster Membership
<i>a</i>	1.00	2.67	C_1
<i>b</i>	2.24	0.85	C_2
<i>c</i>	3.16	0.72	C_2
<i>d</i>	4.12	1.52	C_2
<i>e</i>	0.00	2.63	C_1
<i>f</i>	3.00	0.57	C_2
<i>g</i>	1.00	2.95	C_1
<i>h</i>	1.41	2.13	C_1

Figure 10.6 Clusters and centroids Δ after second pass through k -means algorithm.**TABLE 10.4** Finding the nearest cluster center for each record (third pass)

Point	Distance from m_1	Distance from m_2	Cluster Membership
a	1.27	3.01	C_1
b	2.15	1.03	C_2
c	3.02	0.25	C_2
d	3.95	1.03	C_2
e	0.35	3.09	C_1
f	2.76	0.75	C_2
g	0.79	3.47	C_1
h	1.06	2.66	C_2

10.7 BEHAVIOR OF MSB, MSE, AND PSEUDO- F AS THE k -MEANS ALGORITHM PROCEEDS

Let us observe the behavior of these statistics after step 4 of each pass.

First pass:

$$\begin{aligned}
 \text{SSB} &= \sum_{i=1}^k n_i d(m_i, M)^2 = 3d((1, 2), (2.625, 2.25))^2 + 5d((3.6, 2.4), (2.625, 2.25))^2 \\
 &= 12.975 \\
 \text{MSB} &= \frac{\text{SSB}}{k-1} = \frac{12.975}{2-1} = 12.975
 \end{aligned}$$

$$\begin{aligned}
 \text{SSE} &= \sum_{i=1}^k \sum_{p \in C_i} d(p, m_i)^2 \\
 &= 2^2 + 2.24^2 + 2.83^2 + 3.61^2 + 1^2 + 2.24^2 + 0^2 + 0^2 = 36
 \end{aligned}$$

$$\text{MSE} = \frac{\text{SSE}}{N-k} = \frac{36}{6} = 6$$

$$F = \frac{\text{MSB}}{\text{MSE}} = \frac{12.975}{6} = 2.1625$$

In general, we would expect MSB to increase, MSE to decrease, and F to increase, and such is the case. The calculations are left as an exercise.

Second pass: $MSB = 17.125$, $MSE = 1.313333$, $F = 13.03934$.

Third pass: $MSB = 17.125$, $MSE = 1.041667$, $F = 16.44$.

These statistics indicate that we have achieved the maximum between-cluster variation (as measured by MSB), compared to the within-cluster variation (as measured by MSE).

Note that the k -means algorithm cannot guarantee finding the global maximum pseudo- F statistic, instead often settling at a local maximum. To improve the probability of achieving a global minimum, the analyst may consider using a variety of initial cluster centers. Moore [2] suggests (1) placing the first cluster center on a random data point and (2) placing the subsequent cluster centers on points as far away from previous centers as possible.

One potential problem for applying the k -means algorithm: Who decides how many clusters to search for? That is, who decides k ? Unless the analyst has a priori knowledge of the number of underlying clusters, therefore, an “outer loop” should be added to the algorithm, which cycles through various promising values of k . Clustering solutions for each value of k can therefore be compared, with the value of k resulting in the largest F statistic being selected. Alternatively, some clustering algorithms, such as the BIRCH clustering algorithm, can select the optimal number of clusters.¹

What if some attributes are more relevant than others to the problem formulation? Since cluster membership is determined by distance, we may apply the same axis-stretching methods for quantifying attribute relevance that we discussed in Chapter 7. In Chapter 11, we examine another common clustering method, Kohonen networks, which are related to artificial neural networks in structure.

10.8 APPLICATION OF k -MEANS CLUSTERING USING SAS ENTERPRISE MINER

Next, we turn to the powerful SAS Enterprise Miner 3] software for an application of the k -means algorithm on the *churn* data set from Chapter 3 (available at the book series website; also available from <http://www.sgi.com/tech/mlc/db/>). Recall that the data set contains 20 variables worth of information about 3333 customers, along with an indication of whether or not that customer churned (left the company).

The following variables were passed to the Enterprise Miner clustering node:

- Flag (0/1) variables
 - International Plan and VoiceMail Plan

¹For more on BIRCH clustering, see *Data Mining and Predictive Analytics*, by Daniel Larose and Chantal Larose, John Wiley and Sons, 2015, to appear.

- Numerical variables
 - *Account length, voice mail messages, day minutes, evening minutes, night minutes, international minutes, and customer service calls,*
 - After applying min-max normalization to all numerical variables.

The *Enterprise Miner* clustering node uses SAS's FASTCLUS procedure, a version of the *k*-means algorithm. The number of clusters was set to $k = 3$. The three clusters uncovered by the algorithm varied greatly in size, with tiny cluster 1 containing 92 records, large cluster 2 containing 2411 records, and medium-sized cluster 3 containing 830 records.

Some basic cluster profiling will help us to learn about the types of records falling into each cluster. Figure 10.7 provides a look at the clustering results window of *Enterprise Miner*, containing a pie chart profile of the *International Plan* membership across the three clusters. All members of cluster 1, a fraction of the members of cluster 2, and no members of cluster 3 have adopted the *International Plan*. Note that the left most pie chart represents all records, and is similar to cluster 2.

Next, Figure 10.8 illustrates the proportion of VoiceMail Plan adopters in each cluster. (Note the confusing color reversal for *yes/no* responses.) Remarkably, clusters 1 and 3 contain only VoiceMail Plan adopters, while cluster 2 contains only nonadopters of the plan. In other words, this field was used by the *k*-means algorithm

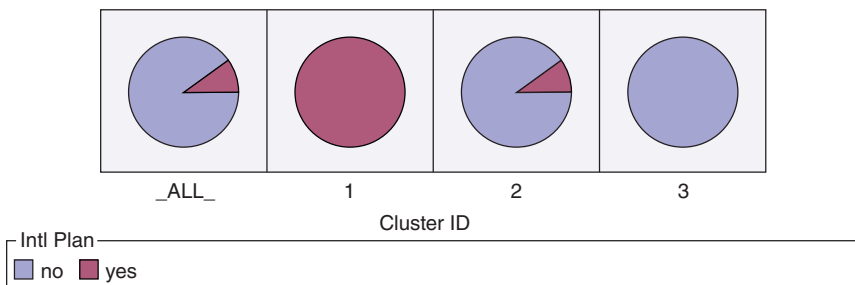


Figure 10.7 Enterprise Miner profile of International Plan adopters across clusters.

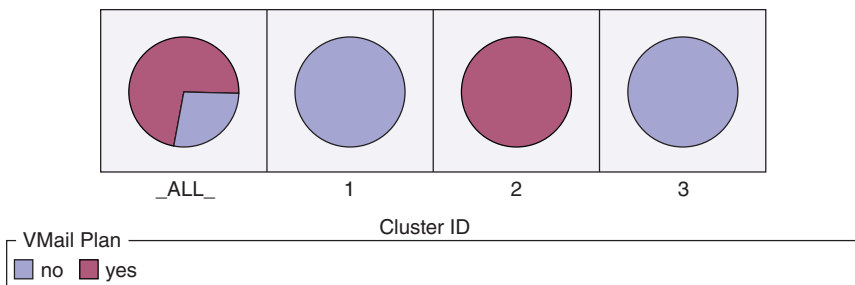


Figure 10.8 VoiceMail Plan adopters and nonadopters are mutually exclusive.

to create a “perfect” discrimination, dividing the data set perfectly among adopters and nonadopters of the International Plan.

It is clear from these results that the algorithm is relying heavily on the categorical variables to form clusters. The comparison of the means of the numerical variables across the clusters in Table 10.5 shows relatively little variation, indicating that the clusters are similar across these dimensions. Figure 10.9, for example, illustrates that the distribution of *customer service calls* (normalized) is relatively similar in each cluster. If the analyst is not comfortable with this domination of the clustering by the categorical variables, he or she can choose to stretch or shrink the appropriate axes, as mentioned earlier, which will help to adjust the clustering algorithm to a more suitable solution.

The clusters may therefore be summarized, using only the categorical variables, as follows:

- *Cluster 1: Sophisticated Users.* A small group of customers who have adopted both the International Plan and the VoiceMail Plan.

TABLE 10.5 Comparison of variable means across clusters shows little variation

Cluster	Freq.	AcctLength_m	VMailMessage	DayMins_mm
1	92	0.4340639598	0.5826939471	0.5360015616
2	2411	0.4131940041	0	0.5126334451
3	830	0.4120730857	0.5731159934	0.5093940185

Cluster	EveMins_mm	NightMins_mm	IntMins_mm	CustServCalls
1	0.5669029659	0.4764366069	0.5467934783	0.1630434783
2	0.5507417372	0.4773586813	0.5119784322	0.1752615328
3	0.5564095259	0.4795138596	0.5076626506	0.1701472557

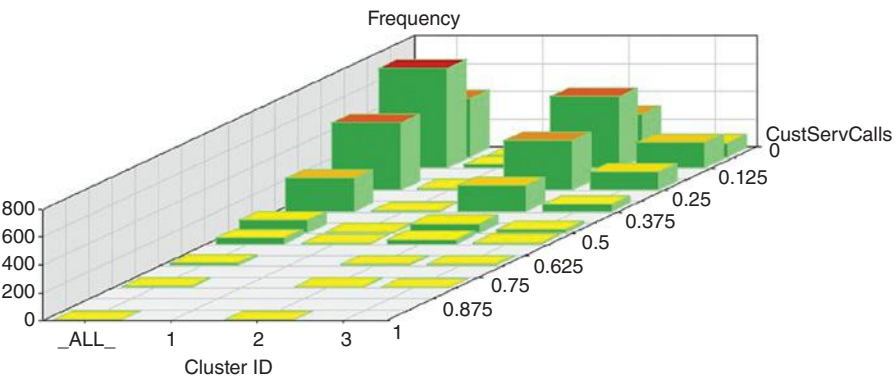


Figure 10.9 Distribution of *customer service calls* is similar across clusters.

- *Cluster 2: The Average Majority.* The largest segment of the customer base, some of whom have adopted the VoiceMail Plan but none of whom have adopted the International Plan.
- *Cluster 3: Voice Mail Users.* A medium-sized group of customers who have all adopted the VoiceMail Plan but not the International Plan.

10.9 USING CLUSTER MEMBERSHIP TO PREDICT CHURN

Suppose, however, that we would like to apply these clusters to assist us in the *churn classification* task. We may compare the proportions of churners directly among the various clusters, using graphs such as Figure 10.10. Here we see that overall (the leftmost column of pie charts), the proportion of churners is much higher among those who have adopted the International Plan than among those who have not. This finding was uncovered in Chapter 3. Note that the churn proportion is higher in cluster 1, which contains International Plan adopters, than in cluster 2, which contains a mixture of adopters and nonadopters, and higher still than cluster 3, which contains no such adopters of the International Plan. Clearly, the company should look at the plan to see why the customers who have it are leaving the company at a higher rate.

Now, since we know from Chapter 3 that the proportion of churners is lower among adopters of the VoiceMail Plan, we would expect that the churn rate for cluster 3 would be lower than for the other clusters. This expectation is confirmed in Figure 10.11.

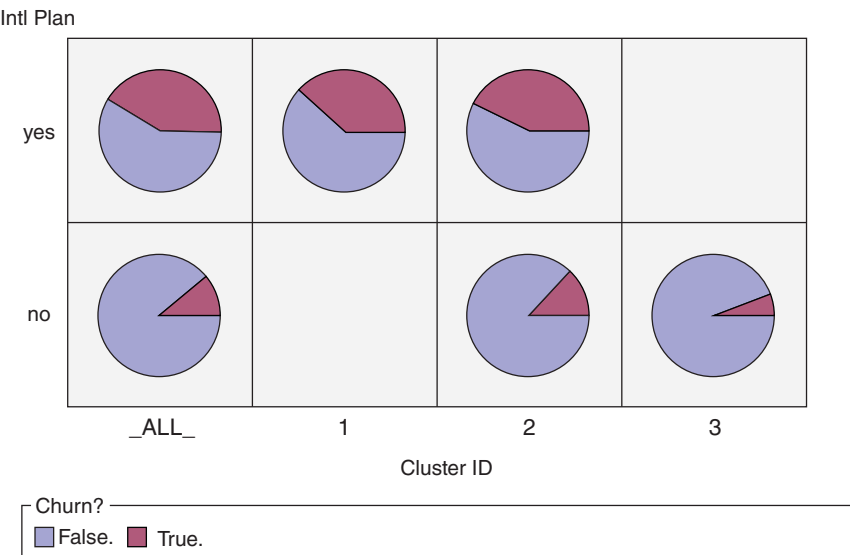


Figure 10.10 Churn behavior across clusters for International Plan adopters and nonadopters.

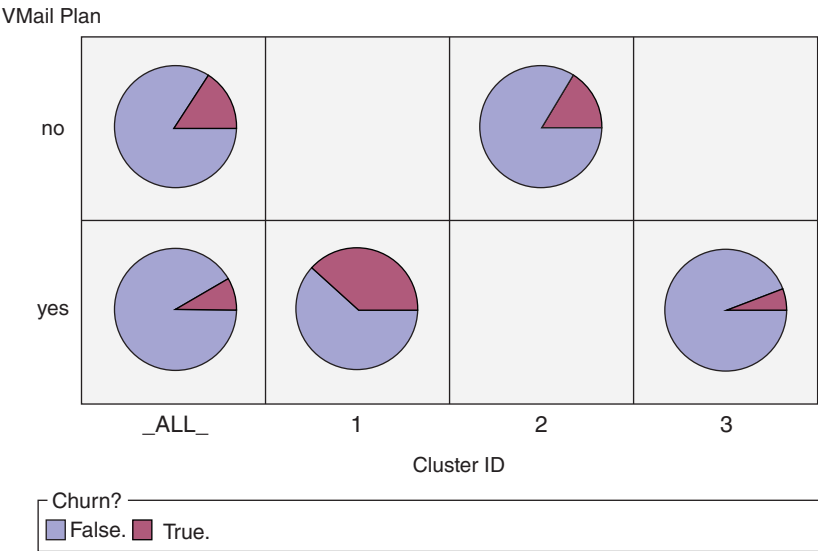


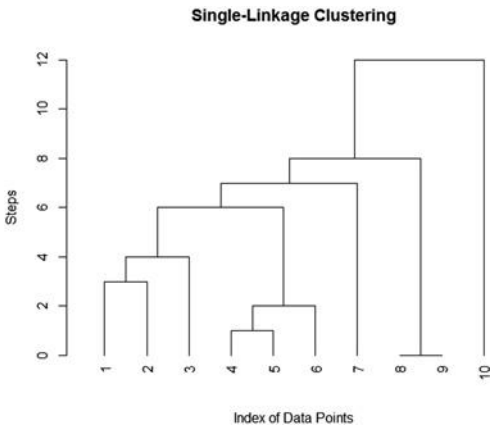
Figure 10.11 Churn behavior across clusters for VoiceMail Plan adopters and nonadopters.

In Chapter 11, we explore using cluster membership as input to downstream data mining models.

THE R ZONE

Single-Linkage Clustering

```
agn <- agnes(data,
  diss = FALSE,
  stand = FALSE,
  method = "single")
# Make and plot the dendrogram
dend_agn <- as.dendrogram(agn)
plot(dend_agn,
  xlab = "Index of Data Points",
  ylab = "Steps",
  main = "Single-Linkage Clustering")
```



Install the required package

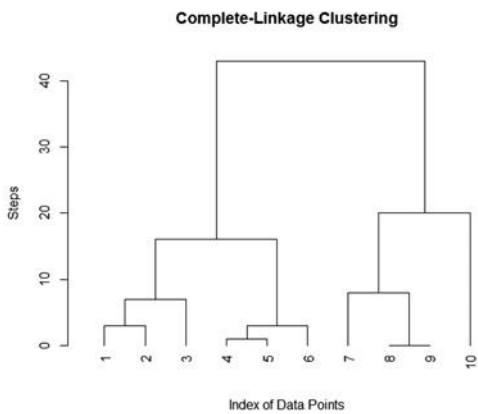
```
install.packages("cluster")
library(cluster)
```

Create the data

```
data <- c(2, 5, 9, 15, 16, 18, 25, 33, 33, 45)
```

Complete-Linkage Clustering

```
agn_complete <- agnes(data,
  diss = FALSE,
  stand = FALSE,
  method = "complete")
# Make and plot the dendrogram
dend_agn_complete <-
  as.dendrogram(agn_complete)
plot(dend_agn_complete,
  xlab = "Index of Data Points",
  ylab = "Steps",
  main = "Complete-Linkage Clustering")
```

**# K-Means clustering**

```
# Create the data matrix
# from Table 10.1
a <- c(1,3)
b <- c(3,3)
c <- c(4,3)
d <- c(5,3)
e <- c(1,2)
f <- c(4,2)
g <- c(1,1)
h <- c(2,1)
m <- rbind(a,b,c,d,e,f,g,h)
km <- kmeans(m, centers = 2)
km
```

```
> km
K-means clustering with 2 clusters of sizes 4, 4

Cluster means:
  [,1] [,2]
1 1.25 1.75
2 4.00 2.75

Clustering vector:
a b c d e f g h
1 2 2 2 1 2 1 1

within cluster sum of squares by cluster:
[1] 3.50 2.75
(between_SS / total_SS = 73.3 %)

Available components:

[1] "cluster"      "centers"      "totss"
[4] "withinss"     "tot.withinss" "betweenss"
[7] "size"
```

REFERENCES

1. James MacQueen, Some methods for classification and analysis of multivariate observations, *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, pp. 281–297, University of California Press, Berkeley, CA, 1967.
2. Andrew Moore, *k-Means and Hierarchical Clustering*, Course Notes, 2001, www.cs.cmu.edu/~cga/ai-course/kmeans.pdf, last accessed March 18, 2014.
3. The SAS Institute, Cary, NC, www.sas.com, accessed March 18, 2014.

EXERCISES

1. To which cluster for the 90210 zip code would you prefer to belong?
2. Describe the goal of all clustering methods.
3. Suppose that we have the following data (one variable). Use single linkage to identify the clusters. Data: 0 0 1 3 3 6 7 9 10 10
4. Suppose that we have the following data (one variable). Use complete linkage to identify the clusters. Data: 0 0 1 3 3 6 7 9 10 10
5. What is an intuitive idea for the meaning of the *centroid* of a cluster?
6. Suppose that we have the following data:

a	B	c	d	e	f	g	h	i	j
(2,0)	(1,2)	(2,2)	(3,2)	(2,3)	(3,3)	(2,4)	(3,4)	(4,4)	(3,5)

Identify the cluster by applying the k -means algorithm, with $k = 2$. Try using initial cluster centers as far apart as possible.

7. Refer to Exercise 6. Show that the ratio of the between-cluster variation to the within-cluster variation decreases with each pass of the algorithm.
8. Once again identify the clusters in Exercise 6 data, this time by applying the k -means algorithm, with $k = 3$. Try using initial cluster centers as far apart as possible.
9. Refer to Exercise 8. Show that the ratio of the between-cluster variation to the within-cluster variation decreases with each pass of the algorithm.
10. Which clustering solution do you think is preferable? Why?
11. Confirm the calculations for the second pass and third pass for MSB, MSE, and pseudo-F for Step Four of the example given in the chapter.

HANDS-ON ANALYSIS

Use the *cereals* data set, included at the book series website, for the following exercises. Make sure that the data are normalized.

12. Using all of the variables except *name* and *rating*, run the k -means algorithm with $k = 5$ to identify clusters within the data.
13. Develop clustering profiles that clearly describe the characteristics of the cereals within the cluster.

14. Rerun the k -means algorithm with $k = 3$.
15. Which clustering solution do you prefer, and why?
16. Develop clustering profiles that clearly describe the characteristics of the cereals within the cluster.
17. Use cluster membership to predict *rating*. One way to do this would be to construct a histogram of *rating* based on cluster membership alone. Describe how the relationship you uncovered makes sense, based on your earlier profiles. ■

KOHONEN NETWORKS

11.1	SELF-ORGANIZING MAPS	228
11.2	KOHONEN NETWORKS	230
11.3	EXAMPLE OF A KOHONEN NETWORK STUDY	231
11.4	CLUSTER VALIDITY	235
11.5	APPLICATION OF CLUSTERING USING KOHONEN NETWORKS	235
11.6	INTERPRETING THE CLUSTERS	237
11.7	USING CLUSTER MEMBERSHIP AS INPUT TO DOWNSTREAM DATA MINING MODELS	242
	THE R ZONE	243
	REFERENCES	245
	EXERCISES	245
	HANDS-ON ANALYSIS	245

11.1 SELF-ORGANIZING MAPS

Kohonen networks were introduced in 1982 by Finnish researcher Tuevo Kohonen [1]. Although applied initially to image and sound analysis, Kohonen networks are nevertheless an effective mechanism for clustering analysis. Kohonen networks represent a type of *self-organizing map* (SOM), which itself represents a special class of neural networks, which we studied in Chapter 9.

The goal of self-organizing maps is to convert a complex high-dimensional input signal into a simpler low-dimensional discrete map [2]. Thus, SOMs are nicely appropriate for cluster analysis, where underlying hidden patterns among records and fields are sought. SOMs structure the output nodes into clusters of nodes, where nodes in closer proximity are more similar to each other than to other nodes that are farther apart. Ritter [3] has shown that SOMs represent a nonlinear generalization of principal component analysis, another dimension-reduction technique.

SOMs are based on *competitive learning*, where the output nodes compete among themselves to be the winning node (or neuron), the only node to be activated by a particular input observation. As Haykin [2] describes it: “The neurons become *selectively tuned* to various input patterns (stimuli) or classes of input patterns in the course of a competitive learning process.” A typical SOM architecture is shown in Figure 11.1. The input layer is shown at the bottom of the figure, with one input node for each field. Just as with neural networks, these input nodes do no processing themselves but simply pass the field input values along downstream.

Like neural networks, SOMs are *feedforward* and *completely connected*. *Feed-forward* networks do not allow looping or cycling. *Completely connected* means that every node in a given layer is connected to every node in the next layer, although not to other nodes in the same layer. Like neural networks, each connection between nodes has a weight associated with it, which at initialization is assigned randomly to a value between zero and 1. Adjusting these weights represents the key for the learning mechanism in both neural networks and SOMs. Variable values need to be normalized or standardized, just as for neural networks, so that certain variables do not overwhelm others in the learning algorithm.

Unlike most neural networks, however, SOMs have no hidden layer. The data from the input layer are passed along directly to the output layer. The output layer is represented in the form of a lattice, usually in one or two dimensions, and typically in the shape of a rectangle, although other shapes, such as hexagons, may be used. The output layer shown in Figure 11.1 is a 3×3 square.

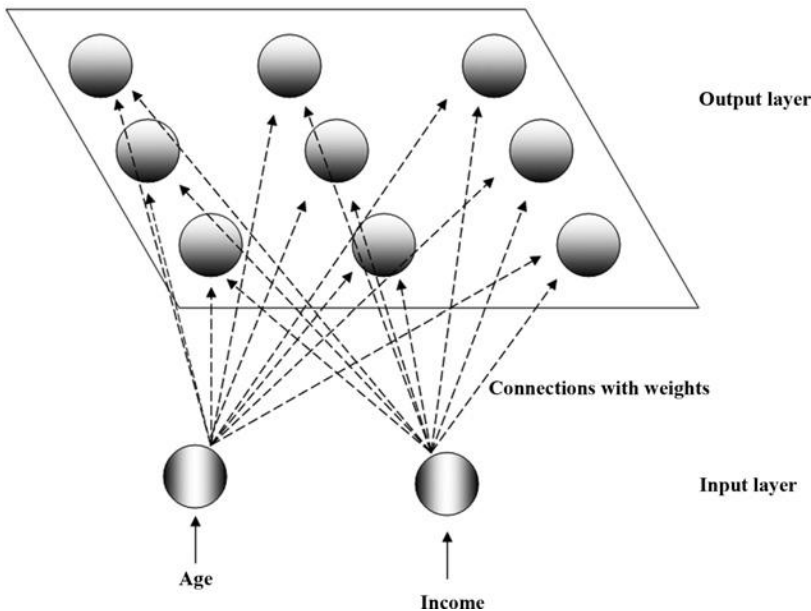


Figure 11.1 Topology of a simple self-organizing map for clustering records by age and income.

For a given record (instance), a particular field value is forwarded from a particular input node to every node in the output layer. For example, suppose that the normalized age and income values for the first record in the data set are 0.69 and 0.88, respectively. The 0.69 value would enter the SOM through the input node associated with *age*, and this node would pass this value of 0.69 to every node in the output layer. Similarly, the 0.88 value would be distributed through the *income* input node to every node in the output layer. These values, together with the weights assigned to each of the connections, would determine the values of a *scoring function* (such as Euclidean distance) for each output node. The output node with the “best” outcome from the scoring function would then be designated as the *winning node*.

SOMs exhibit three characteristic processes:

1. *Competition*. As mentioned above, the output nodes compete with each other to produce the best value for a particular scoring function, most commonly the Euclidean distance. In this case, the output node that has the smallest Euclidean distance between the field inputs and the connection weights would be declared the winner. Later, we examine in detail an example of how this works.
2. *Cooperation*. The winning node therefore becomes the center of a neighborhood of excited neurons. This emulates the behavior of human neurons, which are sensitive to the output of other neurons in their immediate neighborhood. In SOMs, all the nodes in this neighborhood share in the “excitement” or “reward” earned by the winning nodes, that of *adaptation*. Thus, even though the nodes in the output layer are not connected directly, they tend to share common features, due to this neighborliness parameter.
3. *Adaptation*. The nodes in the neighborhood of the winning node participate in adaptation, that is, learning. The weights of these nodes are adjusted so as to further improve the score function. In other words, these nodes will thereby have an increased chance of winning the competition once again, for a similar set of field values.

11.2 KOHONEN NETWORKS

Kohonen networks are SOMs that exhibit *Kohonen learning*. Suppose that we consider the set of m field values for the n th record to be an input vector $\mathbf{x}_n = x_{n1}, x_{n2}, \dots, x_{nm}$, and the current set of m weights for a particular output node j to be a weight vector $\mathbf{w}_j = w_{1j}, w_{2j}, \dots, w_{mj}$. In Kohonen learning, the nodes in the neighborhood of the winning node adjust their weights using a linear combination of the input vector and the current weight vector:

$$w_{ij,\text{new}} = w_{ij,\text{current}} + \eta(x_{ni} - w_{ij,\text{current}}) \quad (11.1)$$

where η , $0 < \eta < 1$, represents the *learning rate*, analogous to the neural networks case. Kohonen [4] indicates that the learning rate should be a decreasing function of training epochs (runs through the data set) and that a linearly or geometrically decreasing η is satisfactory for most purposes.

The algorithm for Kohonen networks (after Fausett [5]) is shown in the accompanying box. At initialization, the weights are randomly assigned, unless firm a priori knowledge exists regarding the proper value for the weight vectors. Also at initialization, the learning rate η and neighborhood size R are assigned. The value of R may start out moderately large but should decrease as the algorithm progresses. Note that nodes that do not attract a sufficient number of hits may be pruned, thereby improving algorithm efficiency.

11.2.1 Kohonen Networks Algorithm

For each input vector \mathbf{x} , do:

- *Competition.* For each output node j , calculate the value $D(w_j, x_n)$ of the scoring function. For example, for Euclidean distance, $D(w_j, x_n) = \sqrt{\sum_i (w_{ij} - x_{ni})^2}$. Find the winning node J that minimizes $D(w_j, x_n)$ over all output nodes.
- *Cooperation.* Identify all output nodes j within the neighborhood of J defined by the neighborhood size R . For these nodes, do the following for all input record fields:
 - *Adaptation.* Adjust the weights:

$$w_{ij,\text{new}} = w_{ij,\text{current}} + \eta(x_{ni} - w_{ij,\text{current}})$$

- Adjust the learning rate and neighborhood size, as needed.
- Stop when the termination criteria are met.

11.3 EXAMPLE OF A KOHONEN NETWORK STUDY

Consider the following simple example. Suppose that we have a data set with two attributes, *age* and *income*, which have already been normalized, and suppose that we would like to use a 2×2 Kohonen network to uncover hidden clusters in the data set. We would thus have the topology shown in Figure 11.2.

A set of four records is ready to be input, with a thumbnail description of each record provided. With such a small network, we set the neighborhood size to be $R = 0$, so that only the winning node will be awarded the opportunity to adjust its weight. Also, we set the learning rate η to be 0.5. Finally, assume that the weights have been randomly initialized as follows:

$$\begin{array}{llll} w_{11} = 0.9 & w_{21} = 0.8 & w_{12} = 0.9 & w_{22} = 0.2 \\ w_{13} = 0.1 & w_{23} = 0.8 & w_{14} = 0.1 & w_{24} = 0.2 \end{array}$$

For the first input vector, $\mathbf{x}_1 = (0.8, 0.8)$, we perform the following competition, cooperation, and adaptation sequence.

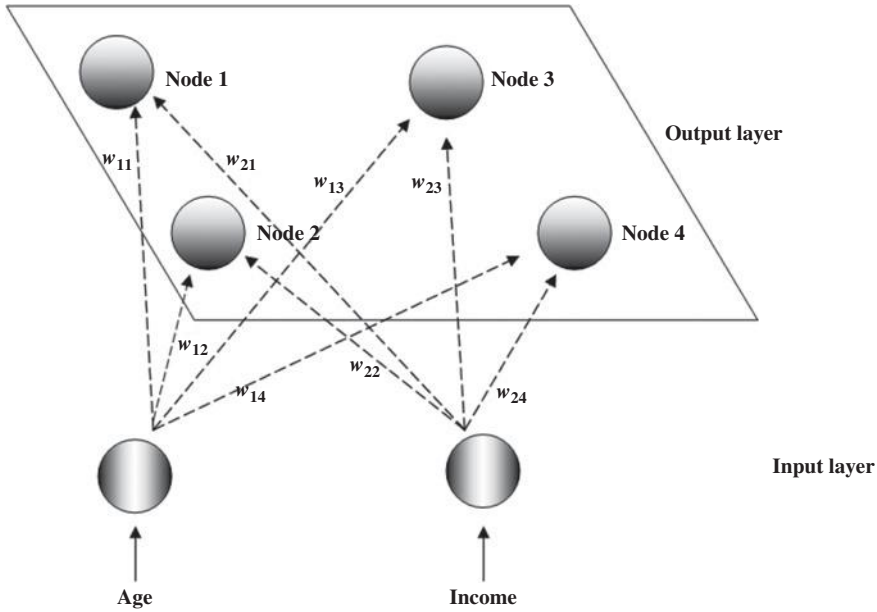


Figure 11.2 Example: topology of the 2×2 Kohonen network.

- *Competition.* We compute the Euclidean distance between this input vector and the weight vectors for each of the four output nodes:

$$\text{Node 1: } D(w_1, x_1) = \sqrt{\sum_i (w_{i1} - x_{1i})^2} = \sqrt{(0.9 - 0.8)^2 + (0.8 - 0.8)^2} = 0.1$$

$$\text{Node 2: } D(w_2, x_1) = \sqrt{(0.9 - 0.8)^2 + (0.2 - 0.8)^2} = 0.61$$

$$\text{Node 3: } D(w_3, x_1) = \sqrt{(0.1 - 0.8)^2 + (0.8 - 0.8)^2} = 0.70$$

$$\text{Node 4: } D(w_4, x_1) = \sqrt{(0.1 - 0.8)^2 + (0.2 - 0.8)^2} = 0.92$$

The winning node for this first input record is therefore node 1, since it minimizes the score function D , the Euclidean distance between the input vector for this record, and the vector of weights, over all nodes.

Note why node 1 won the competition for the first record, $(0.8, 0.8)$. Node 1 won because its weights $(0.9, 0.8)$ are more similar to the field values for this record than are the other nodes' weights. For this reason, we may expect node 1 to exhibit an affinity for records of older persons with high income. In other words, we may expect node 1 to uncover a *cluster* of older, high income persons.

- *Cooperation.* In this simple example, we have set the neighborhood size $R = 0$ so that the level of cooperation among output nodes is nil! Therefore, only the winning node, node 1, will be rewarded with a weight adjustment. (We omit this step in the remainder of the example.)

- *Adaptation.* For the winning node, node 1, the weights are adjusted as follows:

$$w_{ij,\text{new}} = w_{ij,\text{current}} + \eta(x_{ni} - w_{ij,\text{current}})$$

For $j = 1$ (node 1), $n = 1$ (the first record) and learning rate $\eta = 0.5$, this becomes $w_{i1,\text{new}} = w_{i1,\text{current}} + 0.5(x_{i1} - w_{i1,\text{current}})$ for each field:

$$\begin{aligned}\text{For age: } w_{11,\text{new}} &= w_{11,\text{current}} + 0.5(x_{11} - w_{11,\text{current}}) \\ &= 0.9 + 0.5(0.8 - 0.9) = 0.85\end{aligned}$$

$$\begin{aligned}\text{For income: } w_{21,\text{new}} &= w_{21,\text{current}} + 0.5(x_{12} - w_{21,\text{current}}) \\ &= 0.8 + 0.5(0.8 - 0.8) = 0.8\end{aligned}$$

Note the type of adjustment that takes place. The weights are nudged in the direction of the fields' values of the input record. That is, w_{11} , the weight on the *age* connection for the winning node, was originally 0.9, but was adjusted in the direction of the normalized value for *age* in the first record, 0.8. Since the learning rate $\eta = 0.5$, this adjustment is half (0.5) of the distance between the current weight and the field value. This adjustment will help node 1 to become even more proficient at capturing the records of older, high income persons.

Next, for the second input vector, $\mathbf{x}_2 = (0.8, 0.1)$, we have the following sequence.

- *Competition*

$$\begin{aligned}\text{Node 1: } D(w_1, x_2) &= \sqrt{\sum_i (w_{i1} - x_{2i})^2} = \sqrt{(0.9 - 0.8)^2 + (0.8 - 0.1)^2} \\ &= 0.71\end{aligned}$$

$$\text{Node 2: } D(w_2, x_2) = \sqrt{(0.9 - 0.8)^2 + (0.2 - 0.1)^2} = 0.14$$

$$\text{Node 3: } D(w_3, x_2) = \sqrt{(0.1 - 0.8)^2 + (0.8 - 0.1)^2} = 0.99$$

$$\text{Node 4: } D(w_4, x_2) = \sqrt{(0.1 - 0.8)^2 + (0.2 - 0.1)^2} = 0.78$$

Winning node: node 2. Note that node 2 won the competition for the second record, (0.8, 0.1), because its weights (0.9, 0.2) are more similar to the field values for this record than are the other nodes' weights. Thus, we may expect node 2 to "collect" records of older persons with low income. That is, node 2 will represent a cluster of older, low income persons.

- *Adaptation.* For the winning node, node 2, the weights are adjusted as follows:

For $j = 2$ (node 2), $n = 2$ (the first record) and learning rate $\eta = 0.5$, we have $w_{i2,\text{new}} = w_{i2,\text{current}} + 0.5(x_{2i} - w_{i2,\text{current}})$ for each field:

$$\begin{aligned}\text{For age: } w_{12,\text{new}} &= w_{12,\text{current}} + 0.5(x_{21} - w_{12,\text{current}}) \\ &= 0.9 + 0.5(0.8 - 0.9) = 0.85\end{aligned}$$

$$\begin{aligned}\text{For income: } w_{22,\text{new}} &= w_{22,\text{current}} + 0.5(x_{22} - w_{22,\text{current}}) \\ &= 0.2 + 0.5(0.1 - 0.2) = 0.15\end{aligned}$$

Again, the weights are updated in the direction of the field values of the input record. Weight w_{12} undergoes the same adjustment w_{11} above, since the current weights and *age* field values were the same. Weight w_{22} for income is adjusted downward, since the *income* level of the second record was lower than the current *income* weight for the winning node. Because of this adjustment, node 2 will be even better at catching records of older, low income persons.

Next, for the third input vector, $\mathbf{x}_3 = (0.2, 0.9)$, we have the following sequence.

- *Competition*

$$\begin{aligned}\text{Node 1: } D(w_1, x_3) &= \sqrt{\sum_i (w_{i1} - x_{3i})^2} = \sqrt{(0.85 - 0.2)^2 + (0.8 - 0.9)^2} \\ &= 0.66\end{aligned}$$

$$\text{Node 2: } D(w_2, x_3) = \sqrt{(0.85 - 0.2)^2 + (0.15 - 0.9)^2} = 0.99$$

$$\text{Node 3: } D(w_3, x_3) = \sqrt{(0.1 - 0.2)^2 + (0.8 - 0.9)^2} = 0.14$$

$$\text{Node 4: } D(w_4, x_3) = \sqrt{(0.1 - 0.2)^2 + (0.2 - 0.9)^2} = 0.71$$

The winning node is node 3 because its weights (0.1, 0.8) are the closest to the third record's field values. Hence, we may expect node 3 to represent a cluster of younger, high income persons.

- *Adaptation.* For the winning node, node 3, the weights are adjusted as follows:
 $w_{i3, \text{new}} = w_{i3, \text{current}} + 0.5(x_{3i} - w_{i3, \text{current}})$, for each field:

$$\begin{aligned}\text{For age: } w_{13, \text{new}} &= w_{13, \text{current}} + 0.5(x_{31} - w_{13, \text{current}}) \\ &= 0.1 + 0.5(0.2 - 0.1) = 0.15\end{aligned}$$

$$\begin{aligned}\text{For income: } w_{23, \text{new}} &= w_{23, \text{current}} + 0.5(x_{32} - w_{23, \text{current}}) \\ &= 0.8 + 0.5(0.9 - 0.8) = 0.85\end{aligned}$$

Finally, for the fourth input vector, $\mathbf{x}_4 = (0.1, 0.1)$, we have the following sequence.

- *Competition*

$$\begin{aligned}\text{Node 1: } D(w_1, x_4) &= \sqrt{\sum_i (w_{i1} - x_{4i})^2} = \sqrt{(0.85 - 0.1)^2 + (0.8 - 0.1)^2} \\ &= 1.03\end{aligned}$$

$$\text{Node 2: } D(w_2, x_4) = \sqrt{(0.85 - 0.1)^2 + (0.15 - 0.1)^2} = 0.75$$

$$\text{Node 3: } D(w_3, x_4) = \sqrt{(0.15 - 0.1)^2 + (0.85 - 0.1)^2} = 0.75$$

$$\text{Node 4: } D(w_4, x_4) = \sqrt{(0.1 - 0.1)^2 + (0.2 - 0.1)^2} = 0.10$$

The winning node is node 4 because its weights (0.1, 0.2) have the smallest Euclidean distance to the fourth record's field values. We may therefore expect node 4 to represent a cluster of younger, low income persons.

- *Adaptation.* For the winning node, node 4, the weights are adjusted as follows:
 $w_{i4, \text{new}} = w_{i4, \text{current}} + 0.5(x_{4i} - w_{i4, \text{current}})$, for each field:

$$\begin{aligned}\text{For age: } w_{14, \text{new}} &= w_{14, \text{current}} + 0.5(x_{41} - w_{14, \text{current}}) \\ &= 0.1 + 0.5(0.1 - 0.1) = 0.10\end{aligned}$$

$$\begin{aligned}\text{For income: } w_{24, \text{new}} &= w_{24, \text{current}} + 0.5(x_{42} - w_{24, \text{current}}) \\ &= 0.2 + 0.5(0.1 - 0.2) = 0.15\end{aligned}$$

Thus, we have seen that the four output nodes will represent four distinct clusters if the network continues to be fed data similar to the four records shown in Figure 11.2. These clusters are summarized in Table 11.1.

Clearly, the clusters uncovered by the Kohonen network in this simple example are fairly obvious. However, this example does serve to illustrate how the network operates at a basic level, using competition and Kohonen learning.

TABLE 11.1 Four clusters uncovered by Kohonen Network

Cluster	Associated with:	Description
1	Node 1	Older person with high income
2	Node 2	Older person with low income
3	Node 3	Younger person with high income
4	Node 4	Younger person with low income

11.4 CLUSTER VALIDITY

To avoid spurious results, and to assure that the resulting clusters are reflective of the general population, the clustering solution should be validated. One common validation method is to split the original sample randomly into two groups, develop cluster solutions for each group, and then compare their profiles using the methods below or other summarization methods.

Now, suppose that a researcher is interested in performing further inference, prediction, or other analysis downstream on a particular field, and wishes to use the clusters as predictors. Then, it is important that the researcher not include the field of interest as one of the fields used to build the clusters. For example, in the example below, clusters are constructed using the *churn* data set. We would like to use these clusters as predictors for later assistance in classifying customers as churners or not. Therefore, we must be careful not to include the *churn* field among the variables used to build the clusters.

11.5 APPLICATION OF CLUSTERING USING KOHONEN NETWORKS

Next, we apply the Kohonen network algorithm to the *churn* data set from Chapter 3 (available at the book series website; also available from <http://www.sgi.com/tech/mlc/db/>). Recall that the data set contains 20 variables worth

of information about 3333 customers, along with an indication of whether that customer churned (left the company) or not. The following variables were passed to the Kohonen network algorithm, using IBM/SPSS Modeler:

- Flag (0/1) variables
 - International Plan and VoiceMail Plan
- Numerical variables
 - *Account length, voice mail messages, day minutes, evening minutes, night minutes, international minutes, and customer service calls*
 - After applying Z-score standardization to all numerical variables

The topology of the network was as in Figure 11.3, with every node in the input layer being connected with weights (not shown) to every node in the output layer, which are labeled in accordance with their use in the Modeler results. The Kohonen learning parameters were set in Modeler as follows. For the first 20 cycles (passes through the data set), the neighborhood size was set at $R = 2$, and the learning rate was set to decay linearly starting at $\eta = 0.3$. Then, for the next 150 cycles, the neighborhood size was reset to $R = 1$ while the learning rate was allowed to decay linearly from $\eta = 0.3$ to $\eta = 0$.

As it turned out, the Modeler Kohonen algorithm used only six of the nine available output nodes, as shown in Figure 11.4, with output nodes 01, 11, and 21 being pruned. [Note that each of the six clusters is actually of constant value in this plot, such as (0,0), (1,2), and so on. A random shock (x, y agitation, artificial noise) was introduced to illustrate the size of the cluster membership.]

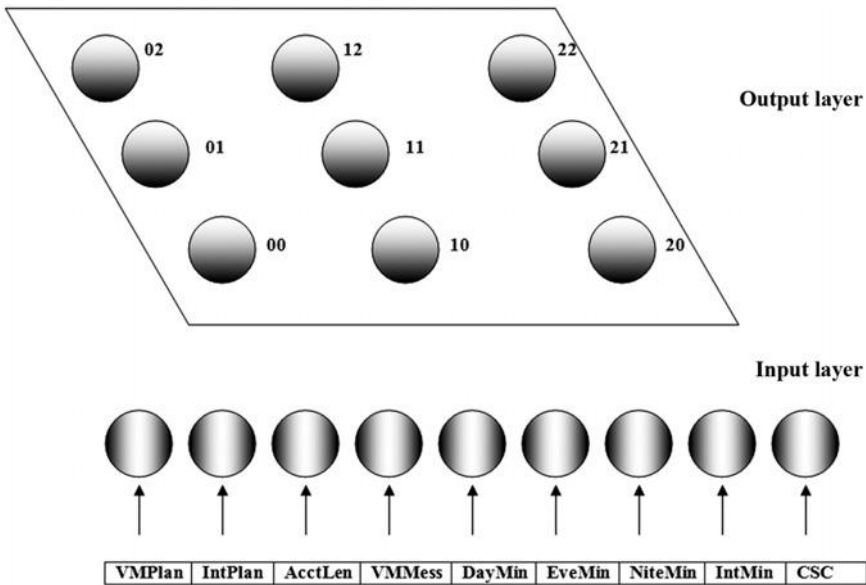


Figure 11.3 Topology of 3 × 3 Kohonen network used for clustering the churn data set.

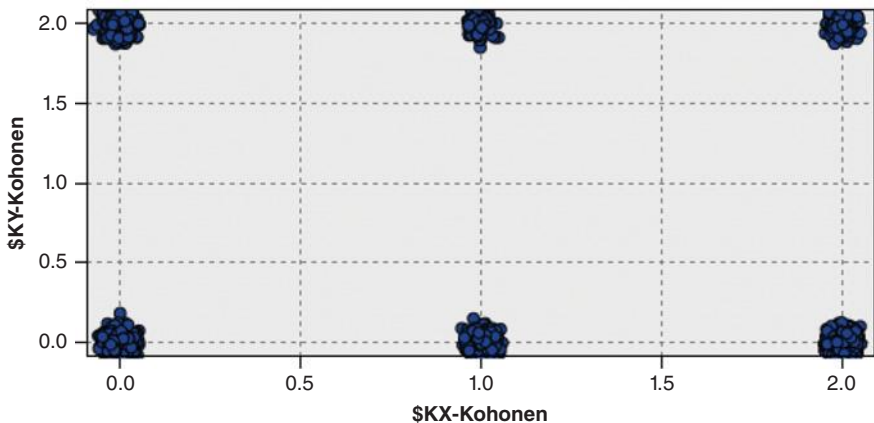


Figure 11.4 Modeler uncovered six clusters.

11.6 INTERPRETING THE CLUSTERS

How are we to interpret these clusters? How can we develop cluster profiles? Consider Figure 11.5, which plots the clusters similar to Figure 11.4, but with panels for whether a customer is an adopter of the International Plan. Figure 11.5 shows that International Plan adopters reside exclusively in Clusters 12 and 22, with the other clusters containing only nonadopters of the International Plan. The Kohonen clustering algorithm has found a high quality discrimination along this dimension, dividing the data set neatly among adopters and nonadopters of the International Plan.

Figure 11.6 shows the VoiceMail Plan adoption status of the cluster members. The three clusters along the bottom row (i.e., Cluster 00, Cluster 10, and Cluster 20) contain only nonadopters of the VoiceMail Plan. Clusters 02 and 12 contain only adopters of the VoiceMail Plan. Cluster 22 contains mostly nonadopters but also some adopters of the VoiceMailPlan.

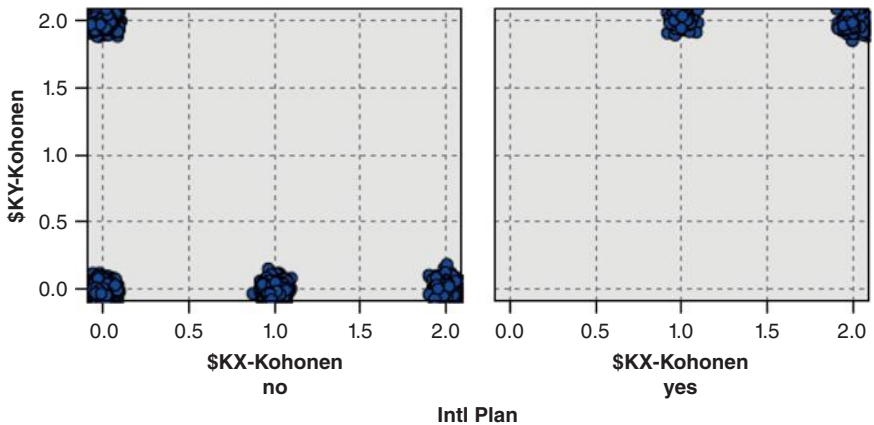


Figure 11.5 International Plan adopters reside exclusively in Clusters 12 and 22.

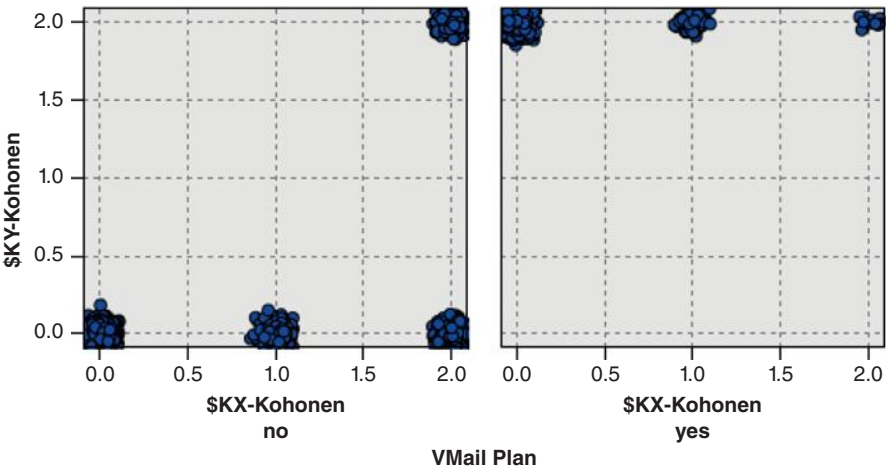


Figure 11.6 Similar clusters are closer to each other.

Recall that because of the neighborliness parameter, clusters that are closer together should be more similar than clusters that are farther apart. Note in Figure 11.5 that all International Plan adopters reside in contiguous (neighboring) clusters, as do all nonadopters. Similarly for Figure 11.6, except that Cluster 22 contains a mixture.

We see that Cluster 12 represents a special subset of customers, those who have adopted both the International Plan and the VoiceMail Plan. This is a well-defined subset of the customer base, which perhaps explains why the Kohonen network uncovered it, even though this subset represents only 2.4% of the customers.

Figure 11.7 provides information about how the values of all the variables are distributed among the clusters, with one column per cluster and one row per variable. The darker rows indicate the more important variables, that is, the variables which proved more useful for discriminating among the clusters. Consider Account Length_Z. Cluster 00 contains customers who tend to have been with the company for a long time, that is, their account lengths tend to be on the large side. Contrast this with Cluster 20, whose customers tend to be fairly new.

For the quantitative variables, the data analyst should report the means for each variable, for each cluster, along with an assessment of whether the difference in means across the clusters is significant. It is important that the means reported to the client appear on the original (untransformed) scale and not on the Z scale or min-max scale, so that the client may better understand the clusters.

Figure 11.8 provides these means, along with the results of an analysis of variance (see Chapter 5) for assessing whether the difference in means across clusters is significant. Each row contains the information for one numerical variable, with one analysis of variance for each row. Each cell contains the cluster mean, standard deviation, standard error (standard deviation/ $\sqrt{\text{cluster count}}$), and cluster count. The degrees of freedom are $df_1 = k - 1 = 6 - 1 = 5$ and $df_2 = N - k = 3333 - 6 = 3327$. The *F*-test statistic is the value of $F = \text{MSTR}/\text{MSE}$ for the analysis of variance for that particular variable, and the Importance statistic is simply $1 - p\text{-value}$, where $p\text{-value} = P(F > F \text{ test statistic})$.

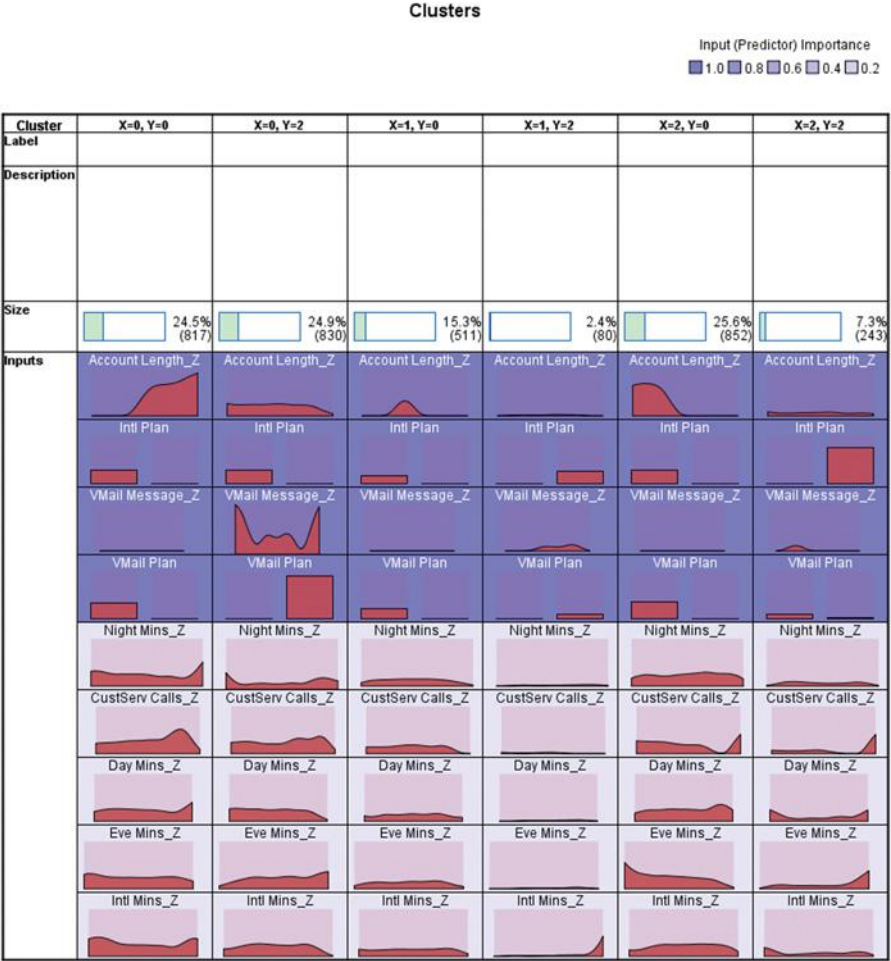


Figure 11.7 How the variables are distributed among the clusters.

Note that both Figure 11.7 and 11.8 concur in identifying account length and the number of voice mail messages as the two most important numerical variables for discriminating among clusters. Next, Figure 11.7 showed graphically that the account length for Cluster 00 is greater than that of Cluster 20. This is supported by the statistics in Figure 11.8, which shows that the mean account length of 141.508 days for Cluster 00 and 61.707 days for Cluster 20. Also, tiny Cluster 12 has the highest mean number of voice mail messages (31.662), with Cluster 02 also having a large amount (29.229). Finally, note that the neighborliness of Kohonen clusters tends to make neighboring clusters similar. It would have been surprising, for example, to find a cluster with 141.508 mean account length right next to a cluster with 61.707 mean account length. In fact, this did not happen.

In general, not all clusters are guaranteed to offer obvious interpretability. The data analyst should team up with a domain expert to discuss the relevance

Grouping field: \$KXY-Kohonen

*Cells contain: Mean, Standard Deviation, Standard Error, Count

Field	X=0, Y=0*	X=0, Y=2*	X=1, Y=0*	X=1, Y=2*	X=2, Y=0*	X=2, Y=2*	F-Test	df	Importance ▾
Account Length	141.508	100.722	100.683	106.962	61.707	103.119	674.616	5, 3327	1.000 ★ Important
	23.917	39.075	8.527	36.221	21.163	39.068			
	0.837	1.356	0.377	4.050	0.725	2.506			
	817	830	511	80	852	243			
VMail Message	0.000	29.229	0.000	31.662	0.000	0.827	7101.682	5, 3327	1.000 ★ Important
	0.000	7.542	0.000	6.240	0.000	3.670			
	0.000	0.262	0.000	0.698	0.000	0.235			
	817	830	511	80	852	243			
Night Mins	196.819	201.483	200.988	204.590	205.786	193.717	3.758	5, 3327	0.998 ★ Important
	49.504	51.374	49.476	53.412	50.421	51.803			
	1.732	1.783	2.189	5.972	1.727	3.323			
	817	830	511	80	852	243			
CustServ Calls	1.696	1.531	1.638	1.425	1.458	1.477	3.597	5, 3327	0.997 ★ Important
	1.378	1.292	1.353	1.329	1.232	1.343			
	0.048	0.045	0.060	0.149	0.042	0.086			
	817	830	511	80	852	243			
Day Mins	176.195	178.695	181.266	189.140	180.253	187.607	2.344	5, 3327	0.961 ★ Important
	54.152	53.576	53.688	51.752	54.926	58.528			
	1.895	1.860	2.375	5.786	1.882	3.755			
	817	830	511	80	852	243			
Eve Mins	201.320	202.366	203.940	207.570	196.409	202.741	2.196	5, 3327	0.948 + Marginal
	51.550	50.477	49.588	48.509	49.766	54.283			
	1.804	1.752	2.194	5.424	1.705	3.482			
	817	830	511	80	852	243			
Intl Mins	10.098	10.153	10.225	10.861	10.312	10.551	2.098	5, 3327	0.937 + Marginal
	2.805	2.787	2.864	2.956	2.766	2.609			
	0.098	0.097	0.127	0.330	0.095	0.167			
	817	830	511	80	852	243			

Figure 11.8 Assessing whether the means across clusters are significantly different.

and applicability of the clusters uncovered using Kohonen or other methods. Here, however, most of these clusters appear fairly clear-cut and self-explanatory.

11.6.1 Cluster Profiles

- *Cluster 00: Loyal Nonadopters.* Belonging to neither the VoiceMail Plan nor the International Plan, customers in large Cluster 00 have nevertheless been with the company the longest, with by far the largest mean account length, which may be related to the largest number of calls to customer service. This cluster exhibits the lowest average minutes usage for day minutes and international minutes, and the second lowest evening minutes and night minutes.
- *Cluster 02: Voice Mail Users.* This large cluster contains members of the VoiceMail Plan, with therefore a high mean number of VoiceMail messages, and no members of the International Plan. Otherwise, the cluster tends toward the middle of the pack for the other variables.
- *Cluster 10: Average Customers.* Customers in this medium-sized cluster belong to neither the VoiceMail Plan nor the International Plan. Except for the second-largest mean number of calls to customer service, this cluster otherwise tends toward the average values for the other variables.

- *Cluster 12: Power Customers.* This smallest cluster contains customers who belong to both the VoiceMail Plan and the International Plan. These sophisticated customers also lead the pack in usage minutes across three categories and are in second place in the other category. They also have the fewest average calls to customer service. The company should keep a watchful eye on this cluster, as they may represent a highly profitable group.
- *Cluster 20: Newbie Nonadopters Users.* Belonging to neither the VoiceMail Plan nor the International Plan, customers in large Cluster 00 represent the company's newest customers, on average, with easily the shortest mean account length. These customers set the pace with the highest mean night minutes usage.
- *Cluster 22: International Plan Users.* This small cluster contains members of the International Plan and only a few members of the VoiceMail Plan. The number of calls to customer service is second lowest, which may mean that they need a minimum of hand-holding. Besides the lowest mean night minutes usage, this cluster tends toward average values for the other variables.

Cluster profiles may be of actionable benefit to companies and researchers. They may, for example, suggest marketing segmentation strategies in an era of shrinking budgets. Rather than targeting the entire customer base for a mass mailing, for example, perhaps only the most profitable customers may be targeted. Another strategy is to identify those customers whose potential loss would be of greater harm to the company, such as the customers in Cluster 12 above. Finally, customer clusters could be identified that exhibit behavior predictive of churning; intervention with these customers could save them for the company.

Suppose, however, that we would like to apply these clusters to assist us in the *churn classification* task. We may compare the proportions of churners among the various clusters, using graphs such as Figure 11.9. From the figure, we can see that customers in Clusters 12 (power customers) and 22 (International Plan users) are in greatest danger of leaving the company, as shown by their higher overall churn proportions. Cluster 02 (VoiceMail Plan users) has the lowest churn rate. The company should take a serious look at its International Plan to see why customers do

Churn			
\$KXY-Kohonen		False.	True.
X=0, Y=0	Count	692	125
	Row %	84.700	15.300
X=0, Y=2	Count	786	44
	Row %	94.699	5.301
X=1, Y=0	Count	440	71
	Row %	86.106	13.894
X=1, Y=2	Count	49	31
	Row %	61.250	38.750
X=2, Y=0	Count	746	106
	Row %	87.559	12.441
X=2, Y=2	Count	137	106
	Row %	56.379	43.621

Figure 11.9 Proportions of churners among the clusters.

not seem to be happy with it. Also, the company should encourage more customers to adopt its VoiceMail Plan, in order to make switching companies more inconvenient. These results and recommendations reflect our findings from Chapter 3, where we initially examined the relationship between churning and the various fields. Note also that Clusters 12 and 22 are neighboring clusters; even though *churn* was not an input field for cluster formation, the type of customers who are likely to churn are more similar to each other than to customers not likely to churn.

11.7 USING CLUSTER MEMBERSHIP AS INPUT TO DOWNSTREAM DATA MINING MODELS

Cluster membership may be used to enrich the data set and improve model efficacy. Indeed, as data repositories continue to grow and the number of fields continues to increase, clustering has become a common method of dimension reduction.

We will illustrate how cluster membership may be used as input for downstream data mining models, using the *churn* data set and the clusters uncovered above. Each record now has associated with it a cluster membership assigned by the Kohonen networks algorithm. We shall enrich our data set by adding this cluster membership field to the input fields used for classifying churn. A CART decision tree model was run, to classify customers as either churners or nonchurners. The resulting decision tree output is shown in Figure 11.10.

The root node split is on whether *DayMin_Z* (the Z-standardized version of day minutes; the analyst should *untransform* these values if this output is meant for the client) is greater than about 1.573. This represents the 142 users who have the highest day minutes, 1.573 standard deviations above the mean. For this group, the second-level split is by cluster, with Cluster 02 split off from the remaining clusters. Note that for high day minutes, the mode classification is *True* (churner), but that within this subset, membership in Cluster 02 acts to protect from churn, since the 31 customers with high day minutes and membership in Cluster 02 have a 100% probability of *not* churning. Recall that Cluster 02, which is acting as a brake on churn behavior, represents *Voice Mail Users*, who had the lowest churn rate of any cluster.



Figure 11.10 Output of CART decision tree for data set enriched by cluster membership.

THE R ZONE

Read in the Churn data set

```
churn <- read.csv(file = "C:/.../churn.txt",
  stringsAsFactors=TRUE)
```

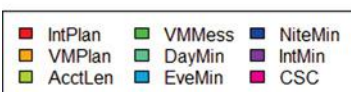
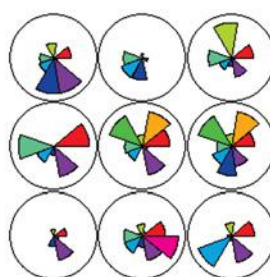
Install the required package and load the library

```
install.packages("kohonen")
library(kohonen)
```

Run the algorithm to get a 3×3 Kohonen network

```
som.churn <- som(data,
  grid = somgrid(3, 3),
  rlen = 200,
  alpha = c(0.03, 0.00),
  radius = 1)
# Make a new color scheme
greys <- function(n, alpha = 1) {
  rev(grey(0:9 / 9))
}
# Plot the make-up of each cluster
plot(som.churn,
  type = c("codes"),
  palette.name = rainbow,
  main = "Cluster Content")
```

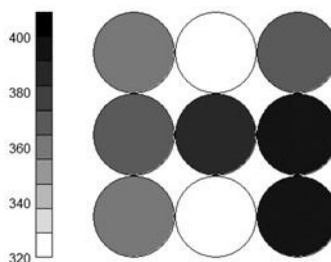
Cluster Content



Plot the counts in each cluster

```
plot(som.churn,
  type = c("counts"),
  palette.name = greys,
  main = "Cluster counts")
```

Cluster counts



Create Flag Variables and standardize numeric variables

```

IntPlan <- VMPlan <- Churn <- c(rep(0, length(churn$Int.l.Plan)))
for (i in 1:length(churn$Int.l.Plan)) {
  if (churn$Int.l.Plan[i]=="yes") IntPlan[i] = 1
  if (churn$VMail.Plan[i]=="yes") VMPlan[i] = 1
  if (churn$Churn[i] == "True") Churn[i] = 1
}
# Standardize
AcctLen <- (churn$Account.Length - mean(churn$Account.Length))/sd(churn$Account.Length)
VMMess <- (churn$VMail.Message - mean(churn$VMail.Message)) /
  sd(churn$VMail.Message)
DayMin <- (churn$Day.Mins - mean(churn$Day.Mins))/sd(churn$Day.Mins)
EveMin <- (churn$Eve.Mins - mean(churn$Eve.Mins))/sd(churn$Eve.Mins)
NiteMin <- (churn$Night.Mins - mean(churn$Night.Mins))/sd(churn$Night.Mins)
IntMin <- (churn$Intl.Mins - mean(churn$Intl.Mins))/sd(churn$Intl.Mins)
CSC <- (churn$CustServ.Calls - mean(churn$CustServ.Calls))/sd(churn$CustServ.Calls)
# Make the variables into one matrix, and make sure the records are the rows
data <- t(rbind(IntPlan, VMPlan, AcctLen, VMMess, DayMin, EveMin, NiteMin, IntMin, CSC))

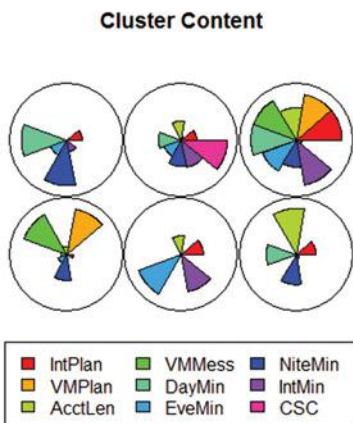
```

Run the algorithm to get a 3×2 Kohonen network

```

som.churn6 <- som(data,
  grid = somgrid(3, 2),
  rlen = 200,
  alpha = c(0.03, 0.00),
  radius = 1)
# Plot the make-up of each cluster
plot(som.churn6,
  type = c("codes"),
  palette.name = rainbow,
  main = "Cluster Content")

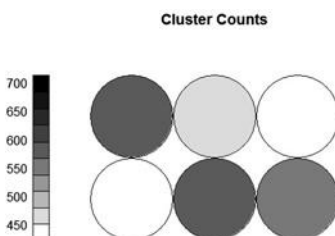
```



```

# Plot the counts in each cluster
plot(som.churn6,
  type = c("counts"),
  palette.name = greys,
  main = "Cluster Counts")

```



REFERENCES

1. Kohonen Tuevo, Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, Vol. 43, pp. 59–69, 1982.
2. Simon Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, Upper Saddle River, NJ, 1990.
3. Helge Ritter, Self-organizing feature maps: Kohonen maps, in M.A. Arbib, ed., *The Handbook of Brain Theory and Neural Networks*, pp. 846–851, MIT Press, Cambridge, MA, 1995.
4. Tuevo Kohonen, *Self-Organization and Associative Memory*, 3rd edn, Springer-Verlag, Berlin, 1989.
5. Laurene Fausett, *Fundamentals of Neural Networks*, Prentice Hall, Upper Saddle River, NJ, 1994.

EXERCISES

1. Describe some of the similarities between Kohonen networks and the neural networks of Chapter 7. Describe some of the differences.
2. Describe the three characteristic processes exhibited by self-organizing maps such as Kohonen networks. What differentiates Kohonen networks from other self-organizing map models?
3. Using weights and distance, explain clearly why a certain output node will win the competition for the input of a certain record.
4. For larger output layers, what would be the effect of increasing the value of R ?
5. Describe what would happen if the learning rate η did not decline?
6. This chapter shows how cluster membership can be used for downstream modeling. Does this apply to the cluster membership obtained by hierarchical and k -means clustering as well?

HANDS-ON ANALYSIS

Use the *adult* data set at the book series website for the following exercises.

7. Apply the Kohonen clustering algorithm to the data set, being careful not to include the *income* field. Use a topology that is not too large, such as 3×3 .
8. Construct a scatter plot (with x/y agitation) of the cluster membership, with an overlay of *income*. Discuss your findings.
9. Construct a bar chart of the cluster membership, with an overlay of *income*. Discuss your findings. Compare to the scatter plot.
10. Construct a bar chart of the cluster membership, with an overlay of *marital status*. Discuss your findings.
11. If your software supports this, construct a web graph of *income*, *marital status*, and the other categorical variables. Fine-tune the web graph so that it conveys good information.
12. Generate numerical summaries for the clusters. For example, generate a cluster mean summary.

13. Using the information above and any other information you can bring to bear, construct detailed and informative cluster profiles, complete with titles.
14. Use cluster membership as a further input to a CART decision tree model for classifying income. How important is clustering membership in classifying income?
15. Use cluster membership as a further input to a C4.5 decision tree model for classifying income. How important is clustering membership in classifying income? Compare to the CART model. ■

ASSOCIATION RULES

- 12.1 AFFINITY ANALYSIS AND MARKET BASKET ANALYSIS 247
- 12.2 SUPPORT, CONFIDENCE, FREQUENT ITEMSETS, AND THE A PRIORI PROPERTY 249
- 12.3 HOW DOES THE A PRIORI ALGORITHM WORK? 251
- 12.4 EXTENSION FROM FLAG DATA TO GENERAL CATEGORICAL DATA 255
- 12.5 INFORMATION-THEORETIC APPROACH: GENERALIZED RULE INDUCTION METHOD 256
- 12.6 ASSOCIATION RULES ARE EASY TO DO BADLY 258
- 12.7 HOW CAN WE MEASURE THE USEFULNESS OF ASSOCIATION RULES? 259
- 12.8 DO ASSOCIATION RULES REPRESENT SUPERVISED OR UNSUPERVISED LEARNING? 260
- 12.9 LOCAL PATTERNS VERSUS GLOBAL MODELS 261
 - THE R ZONE 262
 - REFERENCES 263
 - EXERCISES 263
 - HANDS-ON ANALYSIS 264

12.1 AFFINITY ANALYSIS AND MARKET BASKET ANALYSIS

Affinity analysis is the study of attributes or characteristics that “go together.” Methods for affinity analysis, also known as *market basket analysis*, seek to uncover *associations* among these attributes; that is, it seeks to uncover rules for quantifying the relationship between two or more attributes. Association rules take the form “If *antecedent*, then *consequent*,” along with a measure of the support and confidence associated with the rule. For example, a particular supermarket may find that of the

1000 customers shopping on a Thursday night, 200 bought diapers, and of the 200 who bought diapers, 50 bought beer. Thus, the association rule would be: “If buy diapers, then buy beer,” with a *support* of $50/1000 = 5\%$ and a *confidence* of $50/200 = 25\%$.

Examples of association tasks in business and research include the following:

- Investigating the proportion of subscribers to your company’s cell phone plan that respond positively to an offer of a service upgrade
- Examining the proportion of children whose parents read to them who are themselves good readers
- Predicting degradation in telecommunications networks
- Finding out which items in a supermarket are purchased together, and which items are never purchased together
- Determining the proportion of cases in which a new drug will exhibit dangerous side effects

What types of algorithms can we apply to mine association rules from a particular data set? The daunting problem that awaits any such algorithm is the curse of dimensionality. The number of possible association rules grows exponentially in the number of attributes. Specifically, if there are k attributes, we limit ourselves to binary attributes, we account only for the positive cases (e.g., *buy diapers = yes*), there are on the order of $(k)2^{k-1}$ possible association rules. Consider that a typical application for association rules is market basket analysis and that there may be *thousands* of binary attributes (*buy beer? buy popcorn? buy milk? buy bread?* etc.), the search problem appears at first glance to be utterly hopeless. For example, suppose that a tiny convenience store has only 100 different items, and a customer could either buy or not buy any combination of those 100 items. Then there are $2^{100} \cong 1.27 \times 10^{30}$ possible association rules that await your intrepid search algorithm.

The *a priori algorithm* for mining association rules, however, takes advantage of structure within the rules themselves to reduce the search problem to a more manageable size. Before we examine the *a priori* algorithm, however, let us consider some basic concepts and notation for association rule mining. We begin with a simple example.

Suppose that a local farmer has set up a roadside vegetable stand and is offering the following items for sale: {asparagus, beans, broccoli, corn, green peppers, squash, tomatoes}. Denote this set of items as I . One by one, customers pull over, pick up a basket, and purchase various combinations of these items, subsets of I . (For our purposes, we do not keep track of how much of each item is purchased, just whether or not that particular item is purchased.) Suppose Table 12.1 lists the transactions made during one fine fall afternoon at this roadside vegetable stand.

12.1.1 Data Representation for Market Basket Analysis

There are two principal methods of representing this type of market basket data: using either the transactional data format or the tabular data format. The *transactional data format* requires only two fields, an *ID* field and a *content* field, with each

TABLE 12.1 Transactions made at the roadside vegetable stand

Transaction	Items Purchased
1	Broccoli, green peppers, corn
2	Asparagus, squash, corn
3	Corn, tomatoes, beans, squash
4	Green peppers, corn, tomatoes, beans
5	Beans, asparagus, broccoli
6	Squash, asparagus, beans, tomatoes
7	Tomatoes, corn
8	Broccoli, tomatoes, green peppers
9	Squash, asparagus, beans
10	Beans, corn
11	Green peppers, broccoli, beans, squash
12	Asparagus, beans, squash
13	Squash, corn, asparagus, beans
14	Corn, green peppers, tomatoes, beans, broccoli

record representing a single item only. For example, the data in Table 12.1 could be represented using transactional data format as shown in Table 12.2.

In the *tabular data format*, each record represents a separate transaction, with as many 0/1 flag fields as there are items. The data from Table 12.1 could be represented using the tabular data format, as shown in Table 12.3.

TABLE 12.2 Transactional data format for the roadside vegetable stand data

Transaction ID	Items
1	Broccoli
1	Green peppers
1	Corn
2	Asparagus
2	Squash
2	Corn
3	Corn
3	Tomatoes
:	:

12.2 SUPPORT, CONFIDENCE, FREQUENT ITEMSETS, AND THE A PRIORI PROPERTY

Let D be the set of transactions represented in Table 12.1, where each transaction T in D represents a set of items contained in I . Suppose that we have a particular set of items A (e.g., beans and squash), and another set of items B (e.g., asparagus). Then

TABLE 12.3 Tabular data format for the roadside vegetable stand data

Transaction	Asparagus	Beans	Broccoli	Corn	Green Peppers	Squash	Tomatoes
1	0	0	1	1	1	0	0
2	1	0	0	1	0	1	0
3	0	1	0	1	0	1	1
4	0	1	0	1	1	0	1
5	1	1	1	0	0	0	0
6	1	1	0	0	0	1	1
7	0	0	0	1	0	0	1
8	0	0	1	0	1	0	1
9	1	1	0	0	0	1	0
10	0	1	0	1	0	0	0
11	0	1	1	0	1	1	0
12	1	1	0	0	0	1	0
13	1	1	0	1	0	1	0
14	0	1	1	1	1	0	1

an *association rule* takes the form *if A, then B* (i.e., $A \Rightarrow B$), where the *antecedent* A and the *consequent* B are proper subsets of I , and A and B are mutually exclusive. This definition would exclude, for example, trivial rules such as *if beans and squash, then beans*.

The *support* s for a particular association rule $A \Rightarrow B$ is the proportion of transactions in D that contain both A and B . That is,

$$\text{support} = P(A \cap B) = \frac{\text{number of transactions containing both } A \text{ and } B}{\text{total number of transactions}}.$$

The *confidence* c of the association rule $A \Rightarrow B$ is a measure of the accuracy of the rule, as determined by the percentage of transactions in D containing A that also contain B . In other words,

$$\begin{aligned} \text{confidence} = P(B|A) &= \frac{P(A \cap B)}{P(A)} \\ &= \frac{\text{number of transactions containing both } A \text{ and } B}{\text{number of transactions containing } A} \end{aligned}$$

Analysts may prefer rules that have either high support or high confidence, and usually both. *Strong rules* are those that meet or surpass certain minimum support and confidence criteria. For example, an analyst interested in finding which supermarket items are purchased together may set a minimum support level of 20% and a minimum confidence level of 70%. On the other hand, a fraud detection analyst or a terrorism detection analyst would need to reduce the minimum support level to 1% or less, since comparatively few transactions are either fraudulent or terror-related.

An *itemset* is a set of items contained in I , and a k -*itemset* is an itemset containing k items. For example, {beans, squash} is a 2-itemset, and {broccoli, green peppers, corn} is a 3-itemset, each from the vegetable stand set I . The *itemset frequency* is

simply the number of transactions that contain the particular itemset. A *frequent itemset* is an itemset that occurs at least a certain minimum number of times, having itemset frequency $\geq \phi$. For example, suppose that we set $\phi = 4$. Then itemsets that occur more than four times are said to be *frequent*. We denote the set of frequent k -itemsets as F_k .

MINING ASSOCIATION RULES

The mining of association rules from large databases is a two-step process:

1. Find all frequent itemsets; that is, find all itemsets with frequency $\geq \phi$.
2. From the frequent itemsets, generate association rules satisfying the minimum support and confidence conditions.

The *a priori algorithm* takes advantage of the a priori property to shrink the search space. The *a priori property* states that if an itemset Z is not frequent, then adding another item A to the itemset Z will not make Z more frequent. That is, if Z is not frequent, $Z \cup A$ will not be frequent. In fact, no *superset* of Z (itemset containing Z) will be frequent. This helpful property reduces significantly the search space for the a priori algorithm.

A PRIORI PROPERTY

If an itemset Z is not frequent then for any item A , $Z \cup A$ will not be frequent.

12.3 HOW DOES THE A PRIORI ALGORITHM WORK?

12.3.1 Generating Frequent Itemsets

Consider the set of transactions D represented in Table 12.1. How would the a priori algorithm mine association rules from this data set?

Let $\phi = 4$, so that an itemset is frequent if it occurs four or more times in D . We first find F_1 , the frequent 1-itemsets, which represent simply the individual vegetable items themselves. To do so, we may turn to Table 12.3 and take the column sums, which give us the number of transactions containing each particular vegetable. Since each sum meets or exceeds $\phi = 4$, we conclude that each 1-itemset is frequent. Thus, $F_1 = \{\text{asparagus, beans, broccoli, corn, green peppers, squash, tomatoes}\}$.

Next, we turn to finding the frequent 2-itemsets. In general, to find F_k , the a priori algorithm first constructs a set C_k of candidate k -itemsets by joining F_{k-1} with itself. Then it prunes C_k using the a priori property. The itemsets in C_k that survive the pruning step then form F_k . Here, C_2 consists of all the combinations of vegetables in Table 12.4.

TABLE 12.4 Candidate 2-itemsets

Combination	Count	Combination	Count
Asparagus, beans	5	Broccoli, corn	2
Asparagus, broccoli	1	Broccoli, green peppers	4
Asparagus, corn	2	Broccoli, squash	1
Asparagus, green peppers	0	Broccoli, tomatoes	2
Asparagus, squash	5	Corn, green peppers	3
Asparagus, tomatoes	1	Corn, squash	3
Beans, broccoli	3	Corn, tomatoes	4
Beans, corn	5	Green peppers, squash	1
Beans, green peppers	3	Green peppers, tomatoes	3
Beans, squash	6	Squash, tomatoes	2
Beans, tomatoes	4		

Since $\phi = 4$, we have $F_2 = \{\{\text{asparagus, beans}\}, \{\text{asparagus, squash}\}, \{\text{beans, corn}\}, \text{and } \{\text{beans, squash}\}, \{\text{beans, tomatoes}\}, \{\text{broccoli, green peppers}\}, \{\text{corn, tomatoes}\}\}$. Next, we use the frequent itemsets in F_2 to generate C_3 , the candidate 3-itemsets. To do so, we join F_2 with itself, where *itemsets are joined if they have the first $k - 1$ items in common* (in alphabetical order). For example, $\{\text{asparagus, beans}\}$ and $\{\text{asparagus, squash}\}$ have the first $k - 1 = 1$ item in common, asparagus. Thus, they are joined into the new candidate itemset $\{\text{asparagus, beans, squash}\}$. Similarly, $\{\text{beans, corn}\}$ and $\{\text{beans, squash}\}$ have the first item, beans, in common, generating the candidate 3-itemset $\{\text{beans, corn, squash}\}$. Finally, candidate 3-itemsets $\{\text{beans, corn, tomatoes}\}$ and $\{\text{beans, squash, tomatoes}\}$ are generated in like fashion. Thus, $C_3 = \{\{\text{asparagus, beans, squash}\}, \{\text{beans, corn, squash}\}, \{\text{beans, corn, tomatoes}\}, \{\text{beans, squash, tomatoes}\}\}$.

C_3 is then pruned, using the a priori property. For each itemset s in C_3 , its size $k - 1$ subsets are generated and examined. If *any* of these subsets are not frequent, s cannot be frequent and is therefore pruned. For example, let $s = \{\text{asparagus, beans, squash}\}$. The subsets of size $k - 1 = 2$ are generated, as follows: $\{\text{asparagus, beans}\}$, $\{\text{asparagus, squash}\}$, and $\{\text{beans, squash}\}$. From Table 12.4 we see that each of these subsets is frequent and that therefore $s = \{\text{asparagus, beans, squash}\}$ is not pruned. The reader will verify that $s = \{\text{beans, corn, tomatoes}\}$ will also not be pruned.

However, consider $s = \{\text{beans, corn, squash}\}$. The subset $\{\text{corn, squash}\}$ has frequency $3 < 4 = \phi$, so that $\{\text{corn, squash}\}$ is not frequent. By the a priori property, therefore, $\{\text{beans, corn, squash}\}$ cannot be frequent, is therefore pruned, and does not appear in F_3 . Also consider $s = \{\text{beans, squash, tomatoes}\}$. The subset $\{\text{squash, tomatoes}\}$ has frequency $2 < 4 = \phi$, and hence is not frequent. Again, by the a priori property, its superset $\{\text{beans, squash, tomatoes}\}$ cannot be frequent and is also pruned, not appearing in F_3 .

We still need to check the count for these candidate frequent itemsets. The itemset $\{\text{asparagus, beans, squash}\}$ occurs four times in the transaction list, $\{\text{beans, corn, tomatoes}\}$ occurs only three times. Therefore, the latter candidate itemset is also pruned, leaving us with a singleton frequent itemset in F_3 : $\{\text{asparagus, beans,}$

squash}. This completes the task of finding the frequent itemsets for the vegetable stand data D .

12.3.2 Generating Association Rules

Next, we turn to the task of generating association rules using the frequent itemsets. This is accomplished using the following two-step process, for each frequent itemset s :

GENERATING ASSOCIATION RULES

1. First, generate all subsets of s .
2. Then, let ss represent a nonempty subset of s . Consider the association rule $R : ss \Rightarrow (s - ss)$, where $(s - ss)$ indicates the set s without ss . Generate (and output) R if R fulfills the minimum confidence requirement. Do so for every subset ss of s . Note that for simplicity, a single-item consequent is often desired.

For example, let $s = \{\text{asparagus, beans, squash}\}$ from F_3 . The proper subsets of s are $\{\text{asparagus}\}$, $\{\text{beans}\}$, $\{\text{squash}\}$, $\{\text{asparagus, beans}\}$, $\{\text{asparagus, squash}\}$, $\{\text{beans, squash}\}$. For the first association rule shown in Table 12.5, we let $ss = \{\text{asparagus, beans}\}$, so that $(s - ss) = \{\text{squash}\}$. We consider the rule $R: \{\text{asparagus, beans}\} \Rightarrow \{\text{squash}\}$. The support is the proportion of transactions in which both $\{\text{asparagus, beans}\}$ and $\{\text{squash}\}$ occur, which is 4 (or 28.6%) of the 14 total transactions in D . To find the confidence, we note that $\{\text{asparagus, beans}\}$ occurs in five of the 14 transactions, four of which also contain $\{\text{squash}\}$, giving us our confidence of $4/5 = 80\%$. The statistics for the second rule in Table 12.5 arise similarly. For the third rule in Table 12.5, the support is still $4/14 = 28.6\%$, but the confidence falls to 66.7%. This is because $\{\text{beans, squash}\}$ occurs in six transactions, four of which also contain $\{\text{asparagus}\}$. Assuming that our minimum confidence criterion is set at 60% and that we desire a single consequent, we therefore have the candidate rules shown in Table 12.5. If our minimum confidence were set at 80%, the third rule would not be reported.

Finally, we turn to single antecedent/single consequent rules. Applying the association rule generation method outlined in the box above, and using the itemsets in F_2 , we may generate the candidate association rules shown in Table 12.6.

TABLE 12.5 Candidate association rules for vegetable stand data: two antecedents

If <i>Antecedent</i> , then <i>Consequent</i>	Support	Confidence
If buy asparagus and beans, then buy squash	$4/14 = 28.6\%$	$4/5 = 80\%$
If buy asparagus and squash, then buy beans	$4/14 = 28.6\%$	$4/5 = 80\%$
If buy beans and squash, then buy asparagus	$4/14 = 28.6\%$	$4/6 = 66.7\%$

TABLE 12.6 Candidate association rules for vegetable stand data: one antecedent

If <i>Antecedent</i> , then <i>Consequent</i>	Support	Confidence
If buy asparagus, then buy beans	5/14 = 35.7%	5/6 = 83.3%
If buy beans, then buy asparagus	5/14 = 35.7%	5/10 = 50%
If buy asparagus, then buy squash	5/14 = 35.7%	5/6 = 83.3%
If buy squash, then buy asparagus	5/14 = 35.7%	5/7 = 71.4%
If buy beans, then buy corn	5/14 = 35.7%	5/10 = 50%
If buy corn, then buy beans	5/14 = 35.7%	5/8 = 62.5%
If buy beans, then buy squash	6/14 = 42.9%	6/10 = 60%
If buy squash, then buy beans	6/14 = 42.9%	6/7 = 85.7%
If buy beans, then buy tomatoes	4/14 = 28.6%	4/10 = 40%
If buy tomatoes, then buy beans	4/14 = 28.6%	4/6 = 66.7%
If buy broccoli, then buy green peppers	4/14 = 28.6%	4/5 = 80%
If buy green peppers, then buy broccoli	4/14 = 28.6%	4/5 = 80%
If buy corn, then buy tomatoes	4/14 = 28.6%	4/8 = 50%
If buy tomatoes, then buy corn	4/14 = 28.6%	4/6 = 66.7%

To provide an overall measure of usefulness for an association rule, analysts sometimes multiply the support times the confidence. This allows the analyst to rank the rules according to a combination of prevalence and accuracy. Table 12.7 provides such a list for our present data set, after first filtering the rules through a minimum confidence level of 80%.

Compare Table 12.7 with Figure 12.1, the association rules reported by Modeler’s version of the a priori algorithm, with minimum 80% confidence, and sorted by support \times confidence. The third column, which Modeler calls “Support %,” is actually not what we defined support to be in this chapter (following Han and Kamber [1], Hand *et al.* [2], and other texts). Instead, what Modeler calls “support” is the proportion of occurrences of the antecedent alone rather than the antecedent and the consequent. To find the actual support for the association rule using the Modeler results, multiply the reported “support” times the reported confidence. For example, Modeler reports 50% support and 85.714% confidence for the first association rule,

TABLE 12.7 Final list of association rules for vegetable stand data: ranked by support \times confidence, minimum confidence 80%

If <i>Antecedent</i> , then <i>Consequent</i>	Support	Confidence	Support \times Confidence
If buy squash, then buy beans	6/14 = 42.9%	6/7 = 85.7%	0.3677
If buy asparagus, then buy beans	5/14 = 35.7%	5/6 = 83.3%	0.2974
If buy asparagus, then buy squash	5/14 = 35.7%	5/6 = 83.3%	0.2974
If buy broccoli, then buy green peppers	4/14 = 28.6%	4/5 = 80%	0.2288
If buy green peppers, then buy broccoli	4/14 = 28.6%	4/5 = 80%	0.2288
If buy asparagus and beans, then buy squash	4/14 = 28.6%	4/5 = 80%	0.2288
If buy asparagus and squash, then buy beans	4/14 = 28.6%	4/5 = 80%	0.2288

Consequent	Antecedent	Support %	Confidence %
Beans	Squash	50.0	85.714
Squash	Asparagus	42.857	83.333
Beans	Asparagus	42.857	83.333
Green Peppers	Broccoli	35.714	80.0
Broccoli	Green Peppers	35.714	80.0
Beans	Asparagus Squash	35.714	80.0
Squash	Asparagus Beans	35.714	80.0
Green Peppers	Broccoli Tomatoes	14.286	100.0
Green Peppers	Broccoli Corn	14.286	100.0
Squash	Asparagus Corn	14.286	100.0
Beans	Tomatoes Squash	14.286	100.0

Figure 12.1 Association rules for vegetable stand data, generated by Modeler.

but this really means $50\% \times 85.714\% = 42.857\%$ support, according to the generally accepted definition of support. Be careful with Figure 12.1, because it reports the consequent before the antecedent. Apart from the “support” anomaly, the software’s association rules shown in Figure 12.1 represent the same rules as those we found step by step, and by hand, for the vegetable stand data.

Armed with this knowledge, the vegetable stand entrepreneur can deploy marketing strategies that take advantage of the patterns uncovered above. Why do these particular products co-occur in customers’ market baskets? Should the product layout be altered to make it easier for customers to purchase these products together? Should personnel be alerted to remind customers not to forget item B when purchasing associated item A?

12.4 EXTENSION FROM FLAG DATA TO GENERAL CATEGORICAL DATA

Thus far, we have examined association rules using flag data types only. That is, all of the vegetable stand attributes took the form of Boolean 0/1 flags, resulting in the tabular data format found in Table 12.3, reflecting a straightforward market basket analysis problem. However, association rules are not restricted to flag data types. In particular, the a priori algorithm can be applied to categorical data in general. Let us look at an example.

Recall the normalized *adult* data set analyzed in Chapters 8 and 9. Here in Chapter 12 we apply the a priori algorithm, for the predictor variables *marital-status*,

sex, *workclass*, and the target variable *income* in that same data set, using Modeler. Minimum support of 15%, minimum confidence of 80%, and a maximum of two antecedents are specified, with the resulting association rules shown in Figure 12.2.

Some of these rules contain the nominal variables *Marital Status* and *Work class*, each of which contain several values, so that these attributes are truly non-flag categorical attributes. The a priori algorithm simply finds the frequent itemsets just as before, this time counting the occurrences of the values of the categorical variables rather than simply the occurrence of the flag.

For example, consider the second rule reported in Figure 12.2: “If *Marital status* = *Never-married*, then *income* ≤ 50K,” with confidence 95.319%. There were 8225 instances in the data set where the attribute *Marital Status* took the value *Never-married*, which represents 32.9% of the number of records in the data set. (Again, Modeler refers to this as the “support,” which is not how most researchers define that term.) The support for this rule is (0.329)(0.95319) = 0.3136. That is, 31.362% of the records contained the value *Never-married* for *Marital Status* and the value “≤50K” for *income*, thus making this pairing a frequent 2-itemset of categorical attributes.

Consequent	Antecedent	Support %	Confidence %
Income = ≤50K	Marital status = Never-married	25.184	95.855
	Work Class = Private		
Income = ≤50K	Marital status = Never-married	32.9	95.319
Income = ≤50K	Marital status = Never-married	18.272	94.374
	Sex = Male		
Income = ≤50K	Sex = Female	23.9	90.979
	Work Class = Private		
Income = ≤50K	Sex = Female	33.164	89.193

Figure 12.2 Association rules for categorical attributes found by the a priori algorithm.

12.5 INFORMATION-THEORETIC APPROACH: GENERALIZED RULE INDUCTION METHOD

The structure of association rules, where the antecedent and consequent are both Boolean statements, makes them particularly well suited for handling categorical data, as we have seen. However, what happens when we try to extend our association rule mining to a broader range of data, specifically, numerical attributes?

Of course, it is always possible to discretize the numerical attributes, for example, by arbitrarily defining income under \$30,000 as *low*, income over \$70,000 as *high*, and other income as *medium*. Also, we have seen how both C4.5 and CART handle numerical attributes by discretizing the numerical variables at favorable locations. Unfortunately, the a priori algorithm is not well equipped to handle numeric attributes unless they are discretized during preprocessing. Of course, discretization can lead to a loss of information, so if the analyst has numerical inputs and prefers not to discretize them, he or she may choose to apply an alternative method for mining

association rules: *generalized rule induction* (GRI). The GRI methodology can handle either categorical or numerical variables as inputs, but still requires categorical variables as outputs.

Generalized rule induction was introduced by Smyth and Goodman in 1992 [3]. Rather than using frequent itemsets, GRI applies an information-theoretic approach (as did the C4.5 decision tree algorithm) to determining the “interestingness” of a candidate association rule.

12.5.1 *J*-Measure

Specifically, GRI applies the *J-measure*:

$$J = p(x) \left[p(y|x) \ln \frac{p(y|x)}{p(y)} + [1 - p(y|x)] \ln \frac{1 - p(y|x)}{1 - p(y)} \right]$$

where

- $p(x)$ represents the probability or confidence of the observed value of x . This is a measure of the coverage of the antecedent. How prevalent is this value of the antecedent attribute? You can calculate $p(x)$ using a frequency distribution for the variable in the antecedent.
- $p(y)$ represents the prior probability or confidence of the value of y . This is a measure of the prevalence of the observed value of y in the consequent. You can calculate $p(y)$ using a frequency distribution for the variable in the consequent.
- $p(y|x)$ represents the conditional probability, or posterior confidence, of y given that x has occurred. This is a measure of the probability of the observed value of y given that this value of x has occurred. That is, $p(y|x)$ represents an updated probability of observing this value of y after taking into account the additional knowledge of the value of x . In association rule terminology, $p(y|x)$ is measured directly by the confidence of the rule.
- \ln represents the natural log function (log to the base e).

For rules with more than one antecedent, $p(x)$ is considered to be the probability of the conjunction of the variable values in the antecedent.

As usual, the user specifies desired minimum support and confidence criteria. For GRI, however, the user also specifies how many association rules he or she would like to be reported, thereby defining the size of an association rule table referenced by the algorithm. The GRI algorithm then generates single-antecedent association rules, and calculates J , the value of the *J*-measure for the rule. If the “interestingness” of the new rule, as quantified by the *J*-measure, is higher than the current minimum J in the rule table, the new rule is inserted into the rule table, which keeps a constant size by eliminating the rule with minimum J . More specialized rules with more antecedents are then considered.

How can the behavior of the *J*-statistic be described? Clearly (since $p(x)$ sits outside the brackets), higher values of J will be associated with higher values of $p(x)$. That is, the *J*-measure will tend to favor those rules whose antecedent value is more prevalent, reflecting higher coverage in the data set. Also, the *J*-measure tends toward

higher values when $p(y)$ and $p(y|x)$ are more extreme (near zero or 1). Hence, the J -measure will also tend to favor those rules whose consequent probability, $p(y)$, is more extreme, or whose rule confidence, $p(y|x)$, is more extreme.

The J -measure favors rules with either very high or very low confidence. Why would we be interested in an association rule with extremely low confidence? For example, suppose that we have a rule R : *If buy beer, then buy fingernail polish*, with confidence $p(y|x) = 0.01\%$, which would presumably be favored by the J -measure, since the confidence is so low. The analyst could then consider the *negative form* of R : *If buy beer, then NOT buy fingernail polish*, with confidence 99.99%. Although such negative rules are often interesting (“I guess we better move that fingernail polish out of the beer section . . .”), they are often not directly actionable.

12.6 ASSOCIATION RULES ARE EASY TO DO BADLY

Association rules need to be applied with care, since their results are sometimes deceptive. Let us look at an example. Turning back to the *a priori* algorithm, we asked Modeler to mine association rules from the *adult* database using 10% minimum support, 60% minimum confidence, and a maximum of two antecedents. One association rule is shown from the results, in Figure 12.3.

The results (not shown) include the following association rule: *If Work_Class = Private, then sex = Male*, with 65.63% confidence. Marketing analysts interested in small business owners might be tempted to use this association rule in support of a new marketing strategy aimed at males. However, seen in its proper light, this rule may in fact be worse than useless.

One needs to take into account the raw (prior) proportion of males in the data set, which in this case is 66.84%. In other words, applying this association rule actually *reduces* the probability of randomly selecting a male from 0.6684 to 0.6563. You would have been better advised to pull a name out of a hat from the entire data set than apply this rule.

Why, then, if the rule is so useless, did the software report it? The quick answer is that the default ranking mechanism for Modeler’s *a priori* algorithm is confidence. However, it needs to be emphasized here that data miners should never simply believe the computer output without making the effort to understand the models and mechanisms underlying the results. With the onset of sophisticated point-and-click data mining software, poor analysis costing millions of dollars is more prevalent than ever. In a word, *data mining is easy to do badly*. Insightful human expertise and constant human vigilance are required to translate the nuggets hidden in the database into actionable and profitable results.

Consequent	Antecedent	Support %	Confidence %
sex = Male	workclass = Private	69.54	65.631

Figure 12.3 An association rule that is worse than useless.

With association rules, one needs to keep in mind the prior probabilities involved. To illustrate, we now ask Modeler to provide us with a priori association rules, but this time using the *confidence difference* as the evaluative measure. Here, rules are favored that provide the greatest increase in confidence from the prior to the posterior. One such association rule is shown in Figure 12.4: If *Marital status* = *Divorced* then *Sex* = *Female*. The data set contains 33.16% females, so an association rule that can identify females with 60.029% confidence is useful. The confidence difference for this association rule is $0.60029 - 0.3316 = 0.26869$ between the prior and posterior confidences.

Alternatively, analysts may prefer to use the *confidence ratio* to evaluate potential rules. This is defined as

$$\text{confidence ratio} = 1 - \min \left(\frac{p(y|x)}{p(y)}, \frac{p(y)}{p(y|x)} \right)$$

For example, for the rule: *If Marital Status = Divorced, then Sex = Female*, we have $p(y) = 0.3316$ and $p(y|x) = 0.60029$, so that

$$\min \left(\frac{p(y|x)}{p(y)}, \frac{p(y)}{p(y|x)} \right) = \frac{p(y)}{p(y|x)} = \frac{0.3316}{0.60029} = 0.5524$$

and the *confidence ratio* equals $1 - 0.5524 = 0.4476$. In the exercises we explore further the differences among these rule selection criteria.

Consequent	Antecedent	Support %	Confidence %
Sex = Female	Marital status = Divorced	13.74	60.029

Figure 12.4 This association rule is useful, because the posterior probability (0.60029) is much greater than the prior probability (0.3316).

12.7 HOW CAN WE MEASURE THE USEFULNESS OF ASSOCIATION RULES?

As we have seen, not all association rules are equally useful. Here we are introduced to a measure that can quantify the usefulness of an association rule: *lift*. We define lift as follows:

$$\text{Lift} = \frac{\text{Rule confidence}}{\text{Prior proportion of the consequent}}$$

Recall the supermarket example where, of 1000 customers, 200 bought diapers, and of these 200 customers who bought diapers, 50 also bought beer. The prior proportion of those who bought beer is $50/1000 = 5\%$, while the rule confidence is $50/200 = 25\%$. Therefore, the lift for the association rule, “If buy diapers, then buy beer”, is

$$\text{Lift} = \frac{0.25}{0.05} = 5$$

This may be interpreted as, “Customers who buy diapers are five times as likely to buy beer as customers from the entire data set.” Clearly, this association rule would be useful to a store manager wishing to sell more diapers. Next, suppose, of that 40 of the 1000 customers bought expensive makeup, whereas, of the 200 customers who bought diapers, only 5 bought expensive makeup. In this case, the lift for the association rule “If buy diapers, then buy expensive makeup”, is

$$\text{Lift} = \frac{5/200}{40/1000} = \frac{0.025}{0.04} = 0.625$$

So, customers who buy diapers are only 62.5% as likely to buy expensive makeup as customers in the entire data set.

In general, association rules with lift values different from 1 will be more interesting and useful than rules with lift values close to 1. Why are rules with lift values close to 1 not useful? Recall the definition of confidence for the association rule “If A then B”:

$$\text{Confidence} = P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Then, to obtain lift, we divide by the prior probability of the consequent B, giving us:

$$\text{Lift} = \frac{\text{Rule confidence}}{\text{Prior proportion of the consequent}} = \frac{P(A \cap B)}{P(A)P(B)}$$

Now, events A and B are *independent* when $P(A \cap B) = P(A)P(B)$. Thus, the ratio $\frac{P(A \cap B)}{P(A)P(B)}$ being close to 1 implies that A and B are independent events, meaning that knowledge of the occurrence of A does not alter the probability of the occurrence of B. Such relationships are not useful from a data mining perspective, and thus it makes sense that we prefer our association rules to have a lift value different from 1.

12.8 DO ASSOCIATION RULES REPRESENT SUPERVISED OR UNSUPERVISED LEARNING?

Before we leave the subject of association rules, let us touch on a few topics of interest. First, we may ask whether association rules represent supervised or unsupervised learning. Recall that most data mining methods represent supervised learning, since (1) a target variable is prespecified, and (2) the algorithm is provided with a rich collection of examples where possible association between the target variable and the predictor variables may be uncovered. Conversely, in unsupervised learning, no target variable is identified explicitly. Rather, the data mining algorithm searches for patterns and structure among all the variables. Clustering is perhaps the most common unsupervised data mining method.

Association rule mining, however, can be applied in either a supervised or an unsupervised manner. In market basket analysis, for example, one may simply be interested in “which items are purchased together,” in which case no target variable

would be identified. On the other hand, some data sets are naturally structured so that a particular variable fulfills the role of consequent, and not antecedent (see the *play* example in the exercises). For example, suppose that political pollsters have collected demographic data in their exit polling, along with the subject's voting preference. In this case, association rules could be mined from this data set, where the demographic information could represent possible antecedents, and the voting preference could represent the single consequent of interest. In this way, association rules could be used to help classify the voting preferences of citizens with certain demographic characteristics, in a supervised learning process.

Thus, the answer to the question is that association rules, while generally used for unsupervised learning, may also be applied for supervised learning for a classification task.

12.9 LOCAL PATTERNS VERSUS GLOBAL MODELS

Finally, data analysts need to consider the difference between *models* and *patterns*. A *model* is a global description or explanation of a data set, taking a high level perspective. Models may be descriptive or inferential. Descriptive models seek to summarize the entire data set in a succinct manner. Inferential models aim to provide a mechanism that enables the analyst to generalize from samples to populations. Either way, the perspective is global, encompassing the entire data set. On the other hand, patterns are essentially local features of the data. Recognizable patterns may in fact hold true for only a few variables or a fraction of the records in the data.

Most of the modeling methods we have covered have dealt with global model building. Association rules, on the other hand, are particularly well suited to uncovering local patterns in the data. As soon as one applies the *if* clause in an association rule, one is partitioning a data set so that, usually, most of the records do not apply. Applying the *if* clause “drills down” deeper into a data set, with the aim of uncovering a hidden local pattern which may or may not be relevant to the bulk of the data.

For example, consider the following association rule from Figure 12.4: *if Marital Status = Divorced, then Sex = Female*, with confidence 60.029%. We see that this association rule applies to only 13.74% of the records and ignores the remaining 86.24% of the data set. Even among this minority of records, the association rule ignores most of the variables, concentrating on only two. Therefore, this association rule cannot claim to be global and cannot be considered a model in the strict sense. It represents a pattern that is local to these records and variables only.

Then again, finding interesting local patterns is one of the most important goals of data mining. Sometimes, uncovering a pattern within the data can lead to the deployment of new and profitable initiatives. For example, recall from the *churn* data set (Chapter 3) that those customers who belonged to the VoiceMail Plan were at considerably lower risk of churning than other customers (see Figure 12.5). This finding affected only 922 (27.663%) of the 3333 records and only two of the variables, and is thus to be considered a local pattern. Nevertheless, the discovery of this nugget

Consequent	Antecedent	Support %	Confidence %
Churn? = False.	VMail Plan = yes	27.663	91.323

Figure 12.5 Profitable pattern: VoiceMail Plan adopters less likely to churn.

could lead to policy changes which, if properly deployed, could lead to increased profits for the cell phone company.

THE R ZONE

Read in the Adult data set

```
dat <- read.csv(file = "C:/... /adult.txt",
stringsAsFactors=TRUE)
```

Install and load the required package

```
install.packages("arules")
library(arules)
```

Make the Factors into a Transaction object

```
testing <- as(dat[,c(1, 3, 4, 5, 7, 8, 9, 11, 12, 13, 14)], "transactions")
```

Run the program, view the output sorted by support

```
rules <- apriori(testing,
parameter = list(supp = 0.15,
conf = 0.80,
maxlen = 3))
inspect(sort(rules))
```

```
> inspect(sort(rules))
```

lhs	rhs	support	confidence	lift
1 {marital.status=Married-civ-spouse}	=> {sex=Male}	0.40644	0.8881216677	1.328807331
2 {marital.status=Never-married}	=> {income<=50K.}	0.31360	0.9531914894	1.253144049
3 {sex=Female}	=> {income<=50K.}	0.29580	0.8919310095	1.172605976
4 {workclass=Private, marital.status=Married-civ-spouse}	=> {sex=Male}	0.26300	0.8889940508	1.330112590
5 {workclass=Private, marital.status=Never-married}	=> {income<=50K.}	0.24140	0.9585451080	1.260182357
6 {marital.status=Married-civ-spouse, income<=50K.}	=> {sex=Male}	0.22540	0.8893623737	1.330663675
7 {workclass=Private, sex=Female}	=> {income<=50K.}	0.21744	0.9097907950	1.196085921
8 {income>50K.}	=> {marital.status=Married-civ-spouse}	0.20420	0.8531082888	1.864147122
9 {income>50K.}	=> {sex=Male}	0.20352	0.8502673797	1.272169758
10 {marital.status=Married-civ-spouse, income>50K.}	=> {sex=Male}	0.18104	0.8865817826	1.326503355
11 {sex=Male, income>50K.}	=> {marital.status=Married-civ-spouse}	0.18104	0.8895440252	1.943763712
12 {marital.status=Never-married, sex=Male}	=> {income<=50K.}	0.17244	0.9437390543	1.240717099

REFERENCES

1. Jiawei Han and Micheline Kamber, *Data Mining Concepts and Techniques*, Morgan Kaufmann, San Francisco, CA, 2001.
2. David Hand, Heikki Mannila, and Padhraic Smith, *Principles of Data Mining*, MIT Press, Cambridge, MA, 2001.
3. Padhraic Smyth and Rodney M. Goodman, An information theoretic approach to rule induction from databases, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 4, No. 4, pp. 301–316, August 1992.
4. J. Ross Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, CA, 1993.

EXERCISES

1. Describe the two main methods of representing market basket data. What are the benefits and drawbacks of each?
2. Describe support and confidence. Express the formula for confidence using support.
3. Restate the a priori property in your own words.

For the following several exercises, consider the following data set from Quinlan [4] shown as Table 12.8. The goal is to develop association rules using the a priori algorithm for trying to predict when a certain (evidently indoor) game may be played. Therefore, unlike the vegetable stand example, we may restrict our itemset search to items that include the attribute *play*.

4. Let $\phi = 3$. Generate the frequent 1-itemsets.
5. Let $\phi = 3$. Generate the frequent 2-itemsets.
6. Let $\phi = 3$. Generate the frequent 3-itemsets.
7. Using 75% minimum confidence and 20% minimum support, generate one-antecedent association rules for predicting *play*.

TABLE 12.8 Weather data set for association rule mining

No.	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Rain	Mild	High	False	Yes
5	Rain	Cool	Normal	False	Yes
6	Rain	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
10	Rain	Mild	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Rain	Mild	High	True	No

8. Using 75% minimum confidence and 20% minimum support, generate two-antecedent association rules for predicting *play*.
9. Multiply the observed support times the confidence for each of the rules in Exercises 7 and 8, and rank them in a table.
10. Verify your manually found results using association rule software.
11. For each of the association rules found above by the a priori algorithm, find the *J*-measure. Then order the rules by *J*-measure. Compare the ordering with that from the a priori support \times confidence ordering.
12. Find the value of the *J*-measure for the sixth rule from Figure 12.5.

HANDS-ON ANALYSIS

Use the *churn* data set, given at the book series website, for the following exercises. Use the *Churn_Training_File*. Filter out all variables except the following: VMail Plan, Intl Plan, CustServ Calls, and Churn. Set CustServ Calls to be ordinal. Allow the three predictors to be in either antecedent or consequent, but do not allow Churn to be in the antecedent.

13. Set the minimum antecedent support to 1%, the minimum rule confidence to 5%, and the maximum number of antecedents to 1. Use rule confidence as your evaluation measure.
 - a. Find the association rule with the greatest lift.
 - b. Report the following for the rule in (a).
 - i. Number of instances
 - ii. Support % (as defined in this chapter)
 - iii. Confidence %
 - iv. Rule support %
 - v. Lift
 - vi. Deployability
 - c. Using hand calculations, show how the measures were calculated.
 - d. Explain, in terms of this data, what each of the measures in (c) means (you can skip (i)).
14. Set the minimum antecedent support to 1%, the minimum rule confidence to 5%, and the maximum number of antecedents to 1.
 - a. Generate rules using confidence difference as your evaluation measure with evaluation measure lower bound = 40. Explain what this evaluation measure means.
 - b. For the rules that are generated, use hand calculations to compute the reported evaluation measure, and show that the evaluation measure lower bound has been met.
 - c. Generate rules using confidence difference as your evaluation measure with evaluation measure lower bound = 30.
 - d. Select the rule with the highest deployability. Explain why the deployability of this rule is greater than the rule we found in Question 1a.
15. Set the minimum antecedent support to 1%, the minimum rule confidence to 5%, and the maximum number of antecedents to 1.
 - a. Generate rules using confidence ratio as your evaluation measure with evaluation measure lower bound = 40. Explain what this evaluation measure means.

- b. Select the rule involving Intl Plan. Use hand calculations to compute the reported evaluation measure, and show that the evaluation measure lower bound has been met.
- 16. Compare the results from Exercise 13 with the results from the EDA and decision tree analysis in Chapters 3 and 6. Discuss similarities and differences. Which analysis format do you prefer? Do you find a confluence of results?
- 17. Apply the GRI algorithm to uncover association rules for predicting either churn or nonchurn behavior. Specify reasonable lower bounds for support and confidence.
- 18. Compare the results from the a priori algorithm with those of the GRI algorithm. Which algorithm yields a richer set of rules, and why? Which algorithm is probably preferable for this particular data set? Why? ■

IMPUTATION OF MISSING DATA

13.1	NEED FOR IMPUTATION OF MISSING DATA	266
13.2	IMPUTATION OF MISSING DATA: CONTINUOUS VARIABLES	267
13.3	STANDARD ERROR OF THE IMPUTATION	270
13.4	IMPUTATION OF MISSING DATA: CATEGORICAL VARIABLES	271
13.5	HANDLING PATTERNS IN MISSINGNESS	272
	THE R ZONE	273
	REFERENCE	276
	EXERCISES	276
	HANDS-ON ANALYSIS	276

13.1 NEED FOR IMPUTATION OF MISSING DATA

In this world of big data, the problem of missing data is widespread. It is the rare database that contains no missing values at all. Depending on how the analyst deals with the missing data may change the outcome of the analysis, so it is important to learn methods for handling missing data that will not bias the results.

Missing data may arise from any of several different causes. Survey data may be missing because the responder refuses to answer a particular question, or simply skips a question by accident. Experimental observations may be missed due to inclement weather or equipment failure. Data may be lost through a noisy transmission, and so on.

In Chapter 2 we learned three common methods for handling missing data:

1. Replace the missing value with some constant, specified by the analyst,
2. Replace the missing value with the field mean (for numeric variables) or the mode (for categorical variables),
3. Replace the missing values with a value generated at random from the observed distribution of the variable.

We learned that there were problems with each of these methods, which could generate inappropriate data values that would bias our results. For example, in Chapter 2, a value of 400 cubic inches was generated for a vehicle whose cubic inches value was missing. However, this value did not take into account that the vehicle is Japanese, and there is no Japanese-made car in the database which has an engine size of 400 cubic inches.

We therefore need *data imputation methods* that take advantage of the knowledge that the car is Japanese when calculating its missing cubic inches. In data imputation, we ask “What would be the most likely value for this missing value, given all the other attributes for a particular record?” For instance, an American car with 300 cubic inches and 150 horsepower would probably be expected to have more cylinders than a Japanese car with 100 cubic inches and 90 horsepower. This is called *imputation of missing data*. In this chapter we shall examine methods for imputing missing values for (a) continuous variables, and (b) categorical variables.

13.2 IMPUTATION OF MISSING DATA: CONTINUOUS VARIABLES

In Chapter 5 we introduced multiple regression using the *cereals* data set. It may be worthwhile to take a moment to review the characteristics of the data set by looking back at that chapter. We noted that there were four missing data values:

- Potassium content of Almond Delight
- Potassium content of Cream of Wheat
- Carbohydrates and sugars content of Quaker Oatmeal

Before we use multiple regression to impute these missing values, we must first prepare the data for multiple regression. In particular, the categorical variables must be transformed into 0/1 dummy variables. We did so (not shown) for the variable *type*, turning it into a flag variable to indicate whether or not the cereal was cold cereal. We then derived a series of dummy variables for the variable *manufacturer*, with flags for Kellogg’s, General Mills, Ralston, and so on.

We begin by using multiple regression to build a good regression model for estimating potassium content. Note that we will be using the variable *potassium* as the response, and not the original response variable, *rating*. The idea is to use the set of predictors (apart from potassium) to estimate the potassium content for our Almond Delight cereal. Thus, all the original predictors (minus potassium) represent the predictors, and *potassium* represents the response variable, for our regression model for imputing potassium content. Do not include the original response variable *rating* as a predictor for the imputation.

Because not all variables will be significant for predicting potassium, we apply the stepwise variable selection method of multiple regression. In stepwise regression,¹ the regression model begins with no predictors, then the most significant predictor is

¹For more on regression modeling and multiple regression, including stepwise regression, see *Data Mining Methods and Models*, by Daniel Larose (Wiley, 2006) or *Data Mining and Predictive Analytics*, by Daniel Larose and Chantal Larose (Wiley, 2015, to appear).

entered into the model, followed by the next most significant predictor. At each stage, each predictor is tested whether it is still significant. The procedure continues until all significant predictors have been entered into the model, and no further predictors have been dropped. The resulting model is usually a good regression model, though it is not guaranteed to be the global optimum.

Figure 13.1 shows the multiple regression results for the model chosen by the stepwise variable selection procedure.² The regression equation is

$$\begin{aligned}\text{Estimated potassium} = & -73.11 + 10.137 (\text{Protein}) + 23.515 (\text{Fiber}) \\ & + 1.6444 (\text{Sugars}) + 7.841 (\text{Shelf}) + 70.61 (\text{Weight}) \\ & - 22.1 (\text{Kelloggs})\end{aligned}$$

```

The regression equation is
Potass = - 73.1 + 10.1 Protein + 23.5 Fiber + 1.64 Sugars + 7.84 Shelf
        + 70.6 Weight - 22.1 Kelloggs

74 cases used, 3 cases contain missing values

Predictor      Coef      SE Coef      T      P
Constant      -73.11     18.53     -3.94   0.000
Protein        10.137     2.946      3.44   0.001
Fiber          23.515     1.270     18.52   0.000
Sugars         1.6444     0.7334     2.24   0.028
Shelf          7.841      3.208      2.44   0.017
Weight         70.61     20.95      3.37   0.001
Kelloggs      -22.096     5.534     -3.99   0.000

S = 21.3976    R-Sq = 91.6%    R-Sq(adj) = 90.9%

Analysis of Variance

Source          DF          SS          MS          F          P
Regression       6       336060       56010      122.33    0.000
Residual Error   67        30676         458
Total           73       366736

Predicted Values for New Observations

New Obs      Fit      SE Fit      95% CI      95% PI
    1      77.97      4.41   (69.16, 86.77)   (34.36, 121.57)

Values of Predictors for New Observations

New Obs  Protein  Fiber  Sugars  Shelf  Weight  Kelloggs
    1       2.00   1.00    8.00   3.00    1.00  0.000000

```

Figure 13.1 Multiple regression results for imputation of missing potassium values. (The predicted values section of this output is for Almond Delight only.)

²Note two things about this regression. First, the predictors were chosen using best subsets regression (see reference in footnote 1). Second, for simplicity, Shelf is treated here as a numeric rather than a categorical variable, in order to concentrate on issues of missing data.

To estimate the potassium content for Almond Delight, we plug in Almond Delight's values for the predictors in the regression equation:

$$\begin{aligned}\text{Estimated potassium for Almond Delight} &= -73.11 + 10.137(2) + 23.515(1) \\ &\quad + 1.6444(8) + 7.841(3) + 70.61(1) \\ &\quad - 22.1(0) = 77.9672\end{aligned}$$

That is, the estimated potassium in Almond Delight is 77.9672 mg. This, then, is our imputed value for Almond Delight's missing potassium value: 77.9672 mg.

We may use the same regression equation to estimate the potassium content for Cream of Wheat, plugging in Cream of Wheat's values for the predictors in the regression equation:

$$\begin{aligned}\text{Estimated potassium for Cream of Wheat} &= -73.11 + 10.137(3) + 23.515(1) \\ &\quad + 1.6444(0) + 7.841(2) + 70.61(1) \\ &\quad - 22.1(0) = 67.108\end{aligned}$$

The imputed value for Cream of Wheat's missing potassium value is 67.108 mg.

Next we turn to imputing the missing values for the carbohydrates and sugars content of Quaker Oatmeal. A challenge here is that two predictors have missing values for Quaker Oatmeal. For example, if we build our regression model to impute carbohydrates, and the model requires information for sugars, what value do we use for Quaker Oats sugars, since it is missing? Using the mean or other such *ad hoc* substitute is unsavory, for the reasons mentioned earlier. Therefore, we will use the following approach:

Step 1. Build a regression model to impute carbohydrates; do not include sugars as a predictor.

Step 2. Construct a regression model to impute sugars, using the carbohydrates value found in Step 1.

Thus, the values from Steps 1 and 2 will represent our imputed values for sugars and carbohydrates. Note that we will include the earlier imputed values for potassium.

Step 1: The stepwise regression model for imputing carbohydrates, based on all the predictors except sugars, is as follows (to save space, the computer output is not shown):

$$\begin{aligned}\text{Estimated carbohydrates} &= 6.004 - 1.7741(\text{Fat}) + 0.06557(\text{Calories}) \\ &\quad + 0.9297(\text{Protein}) + 0.013364(\text{Sodium}) \\ &\quad - 0.7331(\text{Fiber}) + 4.406(\text{Nabisco}) + 2.7(\text{Ralston})\end{aligned}$$

(Note that sugars is not one of the predictors.) Then the imputed Step 1 carbohydrates for Quaker Oats is as follows:

$$\begin{aligned}\text{Estimated carbohydrates for Quaker Oats} &= 6.004 - 1.7741(2) + 0.06557(100) \\ &\quad + 0.9297(5) + 0.013364(0) \\ &\quad - 0.7331(2.7) + 4.406(0) \\ &\quad + 2.7(0) = 11.682 \text{ grams}\end{aligned}$$

Step 2: We then replace the missing carbohydrates value for Quaker Oats with 11.682 in the data set. The stepwise regression model for imputing sugars is

$$\begin{aligned}\text{Estimated sugars} &= 0.231 + 0.16307(\text{Calories}) - 1.5664(\text{Fat}) \\ &\quad - 1.04574(\text{Carbohydrates}) - 0.8997(\text{Protein}) \\ &\quad + 1.329(\text{Cups}) + 7.934(\text{Weight}) \\ &\quad - 0.34937(\text{Fiber}) + 1.342(\text{Ralston})\end{aligned}$$

$$\begin{aligned}\text{Estimated sugars for Quaker Oats} &= 0.231 + 0.16307(100) - 1.5664(2) \\ &\quad - 1.04574(11.682) - 0.8997(5) + 1.329(0.67) \\ &\quad + 7.934(1) - 0.34937(2.7) + 1.342(0) \\ &= 4.572 \text{ grams}\end{aligned}$$

We insert 4.572 for the missing sugars value for Quaker Oats in the data set, so that there now remain no missing values in the data set.

Now, ambitious programmers may wish to (a) use the imputed 4.572 grams sugars value to impute a more precise value for carbohydrates, (b) use that more precise value for carbohydrates to go back and obtain a more precise value for sugars, and (c) repeat steps (a) and (b) until convergence. However, the estimates obtained using a single application of Steps 1 and 2 above usually result in a useful approximation of the missing values.

When there are several variables with many missing values, the above step-by-step procedure may be onerous, without recourse to a recursive programming language. In this case, do the following:

- Step 1. Impute the values of the variable with the *fewest* missing values. Use only the variables with no missing values as predictors. If no such predictors are available, use the set of predictors with the fewest missing values (apart from the variable you are predicting, of course).
- Step 2. Impute the values of the variable with the next fewest missing values, using similar predictors as used in Step 1.
- Step 3. Repeat Step 2 until all missing values have been imputed.

13.3 STANDARD ERROR OF THE IMPUTATION

Clients may wish to have an idea of the *precision* of an imputed value. When estimating or imputing anything, analysts should try to provide a measure of the precision of their estimate or imputation. In this case, the *standard error of the imputation*³ is used. The formula for the simple linear regression case is

$$\text{Standard error of the imputation} = \text{SEI} = s \cdot \sqrt{1 + \frac{1}{n} + \frac{(x_p - \bar{x})^2}{(n-1)s_x^2}}$$

³This is from the same formula used to find prediction intervals for the value of a randomly chosen y in simple linear regression.

where s is the standard error of the estimate for the regression, x_p is the value of the known predictor for the particular record, \bar{x} represents the mean value of the predictor across all records, and s_x^2 represents the variance of the predictor values.

For multiple regression (as used here) the formula for SEI is more complex and is best left to the software. Minitab reports SEI as “SE Fit”. In Figure 13.1, where we were imputing Almond Delight’s missing potassium value, the standard error of the imputation is $\text{SEI} = \text{SE Fit} = 4.41$ mg. This is interpreted as meaning that, in repeated samples of Almond Delight cereal, the typical prediction error for imputing potassium, using the predictors in Figure 13.1, is 1.04 mg.

13.4 IMPUTATION OF MISSING DATA: CATEGORICAL VARIABLES

One may use any classification algorithm to impute the missing values of categorical variables. We will illustrate using CART (classification and regression trees, Chapter 8). The data file *classifyrisk* is a small data file containing 6 fields and 246 records. The categorical predictors are *maritalstatus* and *mortgage*; the continuous predictors are *income*, *age*, and number of *loans*. The target is *risk*, a dichotomous field with values *good risk* and *bad loss*. The data file *classifyrisk_missing* contains a missing value for the marital status of record number 19.

To impute this missing value, we apply CART, with *maritalstatus* as the target field, and the other predictors as the predictors for the CART model. Z-score standardization is carried out on the continuous variables. The resulting CART model is shown in Figure 13.2.

Record 19 represents a customer who has the following field values: *loans* = 1, *mortgage* = y, *age_Z* = 1.450, *income_Z* = 1.498, thus representing a customer who

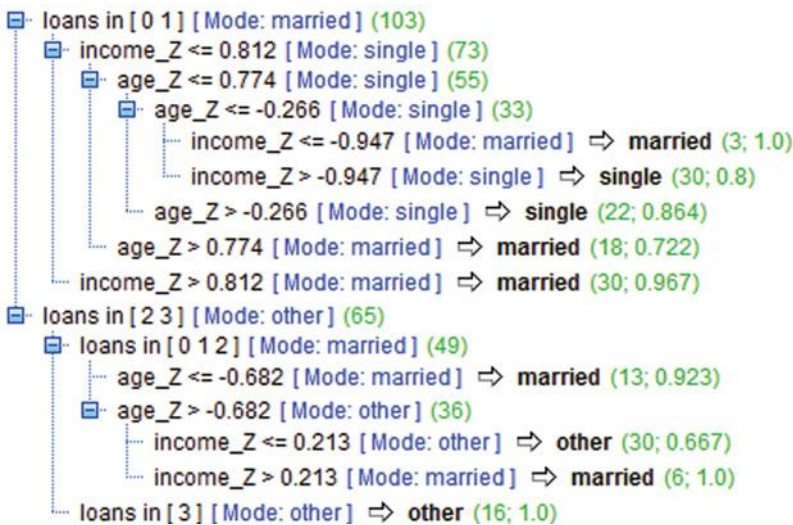


Figure 13.2 CART model for imputing the missing value of *maritalstatus*.

is older than average, with higher income than average, with a mortgage and one other loan. The root node split is on *loans*; we follow the branch down “loans in [0 1]”. The next split checks whether *income_Z* is greater than 0.812. We follow the branch down “*income_Z* > 0.812”, which ends at a leaf node containing 30 records, 96.7% of which have a marital status of *married*. Thus, our imputed value for the marital status of record 19 is *married*, with a confidence level of 96.7%.

13.5 HANDLING PATTERNS IN MISSINGNESS

The analyst should remain aware that imputation of missing data represents replacement. The data value is now no longer missing; rather, its “missingness” has been replaced with an imputed data value. However, there may be information in the pattern of that missingness, information that will be wasted unless some indicator is provided to the algorithm indicating that this data value had been missing. For example, suppose a study is being made of the effect of a new fertility drug on premenopausal women, and the variable *age* has some missing values. It is possible that there is a correlation between the age of the subject, and the likelihood that the subject declined to give their age. Thus, it may happen that the missing values for *age* are more likely to occur for greater values of *age*. Because greater age is associated with infertility, the analyst must account for this possible correlation, by flagging which cases have had their missing ages imputed.

One method to account for patterns in missingness is simply to construct a flag variable, as follows:

$$\text{age_missing} = \begin{cases} 1 & \text{if age value imputed} \\ 0 & \text{otherwise} \end{cases}$$

Add *age_missing* to the model, and interpret its effect. For example, in a regression model, perhaps the *age_missing* dummy variable has a negative regression coefficient, with a very small *p*-value, indicating significance. This would indicate that indeed there is a pattern in the missingness, namely that the effect size of the fertility drug for those cases whose age value was missing tended to be smaller (or more negative). The flag variable could also be used for classification models, such as CART or C4.5.

Another method for dealing with missing data is to reduce the *weight* that the case wields in the analysis. This does not account for the patterns in missingness, but rather represents a compromise between no indication of missingness and completely omitting the record. For example, suppose a data set has ten predictors, and Record 001 has one predictor value missing. Then this missing value could be imputed, and Record 001 assigned a weight, say, of 0.90. Then Record 002, with two of ten field values missing, would be assigned a weight of 0.80. The specific weights assigned depend on the particular data domain and research question of interest. The algorithms would then reduce the amount of influence the records with missing data have on the analysis, proportional to how many fields are missing.

THE R ZONE

Input the data set

```
cereal <- read.csv(file = "C:/... /cereals.txt",
  stringsAsFactors=FALSE,
  header=TRUE,
  sep="\t")
names(cereal)
```

```
> names(cereal)
[1] "Name"      "Manuf"     "Type"
[4] "Calories"  "Protein"   "Fat"
[7] "Sodium"    "Fiber"     "Carbo"
[10] "Sugars"    "Potass"    "Vitamins"
[13] "Shelf"     "Weight"    "Cups"
[16] "Rating"
```

Show what type of variable each one is

```
sapply(cereal, class)
# Name, Manuf, and Type are categorical variables
# The rest are numeric
```

```
> sapply(cereal, class)
      Name      Manuf      Type
"character" "character" "character"
  Calories    Protein    Fat
"numeric"   "numeric"  "numeric"
   Sodium     Fiber    Carbo
"numeric"   "numeric"  "numeric"
   Sugars    Potass   Vitamins
"numeric"   "numeric"  "numeric"
    Shelf    Weight    Cups
"numeric"   "numeric"  "numeric"
   Rating
"numeric"
```

Use the model to estimate missing values

```
# Almond Delight is record 5;
# Cream of Wheat is record 21
predict(reg2, newdata = cereal[5,])
predict(reg2, newdata = cereal[21,])
```

```
> predict(reg2, newdata = cereal[5,])
      5
77.96659
> predict(reg2, newdata = cereal[21,])
     21
67.10763
```

Read in the ClassifyRisk data set. Install and load the required packages

```
crisk <- read.csv(file = "C:/... /classifyrisk.txt",
  stringsAsFactors=TRUE,
  sep="\t",
  header=TRUE)
install.packages(c("rpart", "rpart.plot"))
library(rpart)
library(rpart.plot)
# Make Record 19's marital status missing
crisk[19,4]<-NA
criskna <- crisk
```

Standardize the continuous variables

```
criskna$zincome <- (criskna$income - mean(criskna$income))/sd(criskna$income)
criskna$zage <- (criskna$age - mean(criskna$age))/sd(criskna$age)
criskna$zloans <- (criskna$loans - mean(criskna$loans))/sd(criskna$loans)
```

Create the dummy variables

```
unique(cereal$Type) # needs one indicator
unique(cereal$Manuf) # needs six indicators
cereal$Cold <- c(rep(0, length(cereal$Type)))
cereal$Manuf_N <- cereal$Manuf_Q <- cereal$Manuf_K <- cereal$Manuf_R <-
  cereal$Manuf_G <- cereal$Manuf_P <- c(rep(0, length(cereal$Manuf)))
for (i in 1:length(cereal$Type)) {
  if(cereal$Type[i] == "C") cereal$Cold[i] <- 1
  if(cereal$Manuf[i] == "N") cereal$Manuf_N[i] <- 1
  if(cereal$Manuf[i] == "Q") cereal$Manuf_Q[i] <- 1
  if(cereal$Manuf[i] == "K") cereal$Manuf_K[i] <- 1
  if(cereal$Manuf[i] == "R") cereal$Manuf_R[i] <- 1
  if(cereal$Manuf[i] == "G") cereal$Manuf_G[i] <- 1
  if(cereal$Manuf[i] == "P") cereal$Manuf_P[i] <- 1
}
```

Build the regression model

```
reg1 <- lm(Potass ~ Calories +
  Protein + Fat + Sodium +
  Fiber + Carbo + Sugars +
  Vitamins + Shelf +
  Weight + Cups + Cold +
  Manuf_P + Manuf_R +
  Manuf_G + Manuf_K +
  Manuf_Q + Manuf_N,
  data = cereal)
step1 <- step(reg1,
  direction = "both")
summary(step1)
```

```
> summary(step1)

Call:
lm(formula = Potass ~ Calories + Protein + Fat + Fiber + Carbo +
  Sugars + Vitamins + Shelf + Weight + Cold + Manuf_P + Manuf_K,
  data = cereal)

Residuals:
    Min       1Q   Median       3Q      Max
-44.006 -10.655  -0.201  12.593  55.608

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -41.3976    29.9285  -1.383  0.17164
Calories     -0.7050     0.5326  -1.324  0.19051
Protein      10.7344     3.6037   2.979  0.00415 **
Fat           8.5642     5.7527   1.489  0.14172
Fiber        24.5904     2.0583  11.947 < 2e-16 ***
Carbo         3.4000     2.4375   1.395  0.16811
Sugars        4.6504     2.2746   2.044  0.04523 *
Vitamins     -0.1644     0.1263  -1.301  0.19815
Shelf         9.0983     3.4874   2.609  0.01141 *
Weight       72.9586    36.2560   2.012  0.04861 *
Cold        -40.0808    22.3437  -1.794  0.07780 .
Manuf_P     -11.5908     8.3905  -1.381  0.17219
Manuf_K     -20.7905     6.3381  -3.280  0.00172 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.61 on 61 degrees of freedom
(3 observations deleted due to missingness)
Multiple R-squared:  0.9293, Adjusted R-squared:  0.9154
F-statistic: 66.84 on 12 and 61 DF,  p-value: < 2.2e-16
```

Run the final regression model

```
# Include only predictors
# significant in the
# previous analysis
reg2<- lm(Potass ~ Protein +
  Fiber + Sugars + Shelf +
  Weight + Manuf_K,
  data = cereal)
summary(reg2)
```

```
> summary(reg2)

Call:
lm(formula = Potass ~ Protein + Fiber + Sugars + Shelf + weight +
    Manuf_K, data = cereal)

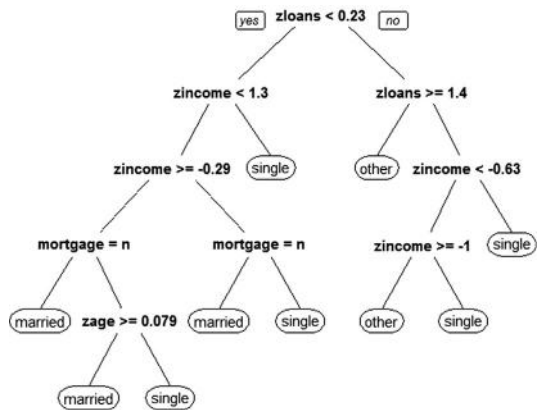
Residuals:
    Min       1Q   Median       3Q      Max
-45.201 -15.013  -0.373  13.470  60.668

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -73.1079    18.5342  -3.944  0.000194 ***
Protein      10.1374     2.9456   3.442  0.001001 **
Fiber       23.5150     1.2698  18.518 < 2e-16 ***
Sugars       1.6444     0.7334   2.242  0.028263 *
Shelf        7.8412     3.2081   2.444  0.017155 *
Weight      70.6058    20.9513   3.370  0.001251 **
Manuf_K     -22.0963     5.5336  -3.993  0.000164 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21.4 on 67 degrees of freedom
(3 observations deleted due to missingness)
Multiple R-squared:  0.9164, Adjusted R-squared:  0.9089
F-statistic: 122.3 on 6 and 67 DF, p-value: < 2.2e-16
```

Apply CART to impute the marital status

```
imp1 <- rpart(marital_status ~
  mortgage + zloans + zage +
  zincome,
  data = criskna,
  model = TRUE,
  method = "class")
# Plot the decision tree
rpart.plot(imp1)
```

**# Predict the marital status of Record 19**

```
predict(imp1, criskna[19,])
```

REFERENCE

1. The classic text on missing data is: Little, Roderick, and Rubin, Donald, *Statistical Analysis with Missing Data*, 2nd edn, Wiley, 2002.

EXERCISES

1. Why do we need to impute missing data?
2. When imputing a continuous variable, explain what we use for the set of predictors, and for the target variable.
3. When imputing a missing value, do we include the original target variable as one of the predictor variables for the data imputation model? Why or why not?
4. Describe what we should do if there are many variables with many missing values.
5. On your own, think of a data set where a potential pattern in missingness would represent good information.
6. State two methods for handling patterns in missingness.

HANDS-ON ANALYSIS

Use the *cereals* data set for Exercises 7–12. Report the standard error of each imputation.

7. Impute the potassium content of Almond Delight using multiple regression.
8. Impute the potassium content of Cream of Wheat.
9. Impute the carbohydrates value of Quaker Oatmeal.
10. Impute the sugars value of Quaker Oatmeal.
11. Insert the value obtained in Exercise 10 for the sugars value of Quaker Oatmeal, and impute the carbohydrates value of Quaker Oatmeal.
12. Compare the standard errors for the imputations obtained in Exercises 9 and 11. Explain what you find.
13. Open the *ClassifyRisk_Missing* data set. Impute the missing value for marital status. Use the *ClassifyRisk_Missing2* data set for Exercises 14–15.
14. Impute all missing values in the data set. Explain the ordering that you are using.
15. Report the standard errors (for continuous values) or confidence levels (for categorical values) for your imputations in Exercise 14. ■

MODEL EVALUATION TECHNIQUES

- 14.1** MODEL EVALUATION TECHNIQUES FOR THE DESCRIPTION TASK **278**
- 14.2** MODEL EVALUATION TECHNIQUES FOR THE ESTIMATION AND PREDICTION TASKS **278**
- 14.3** MODEL EVALUATION TECHNIQUES FOR THE CLASSIFICATION TASK **280**
- 14.4** ERROR RATE, FALSE POSITIVES, AND FALSE NEGATIVES **280**
- 14.5** SENSITIVITY AND SPECIFICITY **283**
- 14.6** MISCLASSIFICATION COST ADJUSTMENT TO REFLECT REAL-WORLD CONCERNS **284**
- 14.7** DECISION COST/BENEFIT ANALYSIS **285**
- 14.8** LIFT CHARTS AND GAINS CHARTS **286**
- 14.9** INTERWEAVING MODEL EVALUATION WITH MODEL BUILDING **289**
- 14.10** CONFLUENCE OF RESULTS: APPLYING A SUITE OF MODELS **290**
 - THE R ZONE **291**
 - REFERENCE **291**
 - EXERCISES **291**
 - HANDS-ON ANALYSIS **291**

As you may recall from Chapter 1, the CRISP cross-industry standard process for data mining consists of six phases, to be applied in an iterative cycle:

- 1.** Business understanding phase
- 2.** Data understanding phase
- 3.** Data preparation phase

4. Modeling phase
5. Evaluation phase
6. Deployment phase

Nestled between the modeling and deployment phases comes the crucial evaluation phase, techniques for which are discussed in this chapter. By the time we arrive at the evaluation phase, the modeling phase has already generated one or more candidate models. It is of critical importance that these models be evaluated for quality and effectiveness *before* they are deployed for use in the field. Deployment of data mining models usually represents a capital expenditure and investment on the part of the company. If the models in question are invalid, the company's time and money are wasted. In this chapter we examine model evaluation techniques for each of the six main tasks of data mining: description, estimation, prediction, classification, clustering, and association.

14.1 MODEL EVALUATION TECHNIQUES FOR THE DESCRIPTION TASK

In Chapter 3 we learned how to apply exploratory data analysis (EDA) to learn about the salient characteristics of a data set. EDA represents a popular and powerful technique for applying the descriptive task of data mining. On the other hand, because descriptive techniques make no classifications, predictions, or estimates, an objective method for evaluating the efficacy of these techniques can be elusive. The watchword is common sense. Remember that data mining models should be as *transparent* as possible. That is, the results of the data mining model should describe clear patterns that are amenable to intuitive interpretation and explanation. The effectiveness of your EDA is best evaluated by the clarity of understanding elicited in your target audience, whether a group of managers evaluating your new initiative or the evaluation board of the U.S. Food and Drug Administration assessing the efficacy of a new pharmaceutical submission.

If one insists on using a quantifiable measure to assess description, one may apply the *minimum descriptive length principle*. Other things being equal, *Occam's razor* (a principle named after the medieval philosopher William of Occam) states that simple representations are preferable to complex ones. The minimum descriptive length principle quantifies this, saying that the best representation (or description) of a model or body of data is the one that minimizes the information required (in bits) to encode (1) the model and (2) the exceptions to the model.

14.2 MODEL EVALUATION TECHNIQUES FOR THE ESTIMATION AND PREDICTION TASKS

For estimation and prediction models, we are provided with both the estimated (or predicted) value \hat{y} of the numeric target variable and the actual value y . Therefore, a natural measure to assess model adequacy is to examine the *estimation error*, or

residual, $|y - \hat{y}|$. Since the average residual is always equal to zero, we cannot use it for model evaluation; some other measure is needed.

The usual measure used to evaluate estimation or prediction models is the *mean square error* (MSE):

$$\text{MSE} = \frac{\sum_i (y_i - \hat{y}_i)^2}{n - p - 1}$$

where p represents the number of model parameters. Models are preferred that minimize MSE. The square root of MSE can be regarded as an estimate of the typical error in estimation or prediction when using the particular model. In context, this is known as the *standard error of the estimate* and denoted by $s = \sqrt{\text{MSE}}$.

For example, consider Figure 14.1 (excerpted from Chapter 5), which provides the Minitab regression output for the estimated nutritional rating based on sugar content for the 76 breakfast cereals with nonmissing sugar values. Both $\text{MSE} = 84.0$ and $s = 9.16616$ are circled on the output. The value of 9.16616 for s indicates that the estimated prediction error from using this regression model to predict nutrition rating based on sugar content alone is 9.16616 rating points.

Is this good enough to proceed to model deployment? That depends on the objectives of the business or research problem. Certainly the model is simplicity itself, with only one predictor and one response; however, perhaps the prediction error is too large to consider deployment. Compare this estimated prediction error with the value of s obtained by the multiple regression in Figure 5.8 in Chapter 5: $s = 7.72769$. The estimated error in prediction for the multiple regression is smaller, but more information is required to achieve this, in the form of a second predictor:

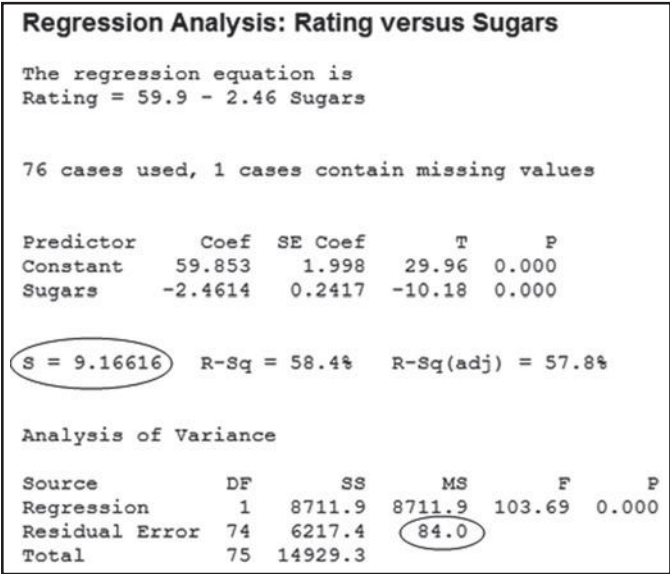


Figure 14.1 Regression results, with MSE and s indicated.

sodium. As with so much else in statistical analysis and data mining, there is a trade-off between model complexity and prediction error. The domain experts for the business or research problem in question need to determine where the point of diminishing returns lies.

In Chapter 9, *Neural Networks*, we examined an evaluation measure that was related to MSE:

$$SSE = \sum_{\text{records}} \sum_{\text{output nodes}} (\text{actual} - \text{output})^2$$

which represents roughly the numerator of MSE above. Again, the goal is to minimize the sum of squared errors over all output nodes. In Chapter 5 we learned another measure of the goodness of a regression model is the *coefficient of determination*,

$$R^2 = \frac{SSR}{SST}$$

R^2 represents the proportion of the variability in the response that is accounted for by the linear relationship between the predictor (or predictors) and the response. For example, in Figure 14.1 we see that $R^2 = 58.4\%$, which means that 58.4% of the variability in cereal ratings is accounted for by the linear relationship between ratings and sugar content. This is actually quite a chunk of the variability, since it leaves only 41.6% of the variability left for all other factors.

14.3 MODEL EVALUATION TECHNIQUES FOR THE CLASSIFICATION TASK

How do we assess how well our classification algorithm is functioning? Classification assignments could conceivably be made based on coin flips, tea leaves, goat entrails, or a crystal ball. Which evaluative methods should we use to assure ourselves that the classifications made by our data mining algorithm are efficacious and accurate? Are we outperforming the coin flips?

In this chapter we examine the following evaluative concepts, methods, and tools: *error rate*, *false positives*, *false negatives*, *error cost adjustment*, *lift*, *lift charts*, and *gains charts*, in the context of the C5.0 model for classifying income from Chapter 8.

14.4 ERROR RATE, FALSE POSITIVES, AND FALSE NEGATIVES

Recall the *Adult* data set from Chapter 8 that we applied a C5.0 model for classifying whether a person's income was low ($\leq \$50,000$) or high ($> \$50,000$), based on a set of predictor variables which included capital gain, capital loss, marital status, and so on. Let us evaluate the performance of that decision tree classification model (with all levels retained, not just three, as in Figure 8.9), using the notions of error rate, false positives, and false negatives.

The general form of the matrix of the correct and incorrect classifications made by a classification algorithm, termed the *contingency table*,¹ is shown in Table 14.1. Table 14.2 contains the statistics from the C5.0 model, with “ $\geq 50K$ ” denoted as the positive classification. The columns represent the predicted classifications, and the rows represent the actual (true) classifications, for each of the 25,000 records. There are 19,016 records whose actual value for the target variable *income* is $\leq 50,000$, and there are 5984 records whose actual value *income* is $>50,000$. The C5.0 algorithm classified 20,758 of the records as having *income* $\leq 50,000$, and 4242 records as having *income* $>50,000$.

Of the 20,758 records whose income is predicted by the algorithm to be $\leq 50,000$, 18,197 of these records actually do have low income. However, the algorithm incorrectly classified 2561 of these 20,758 records as having *income* $\leq 50,000$, when their income is actually $>50,000$.

Now, suppose that this analysis is being carried out for a financial lending firm, which is interested in determining whether or not a loan applicant’s income is $>50,000$. A classification of *income* $>50,000$ is considered to be *positive*, since the lending firm would then proceed to extend the loan to the person in question. A classification of *income* $\leq 50,000$ is considered to be *negative*, since the firm would

TABLE 14.1 General form of the contingency table of correct and incorrect classifications

		Predicted Category		
		0	1	Total
Actual category	0	True negatives: Predicted 0 Actually 0	False positives: Predicted 1 Actually 0	Total actually negative
	1	False negatives: Predicted 0 Actually 1	True positives: Predicted 1 Actually 1	Total actually positive
	Total	Total Predicted negative	Total Predicted positive	Grand total

TABLE 14.2 Contingency table for the C5.0 model

		Predicted Category		
		$\leq 50K$	$>50K$	Total
Actual category	$\leq 50K$	18,197	819	19,016
	$>50K$	2561	3423	5984
	Total	20,758	4242	25,000

¹Also referred to as the *confusion matrix* or the *error matrix*.

proceed to deny the loan application to the person, based on low income (in this simplified scenario). Assume that in the absence of other information, the default decision would be to deny the loan due to low income.

Thus, the 20,758 classifications (predictions) of *income* $\leq 50,000$ are said to be *negatives*, and the 4242 classifications of *income* $> 50,000$ are said to be *positives*. The 2561 negative classifications that were made in error are said to be *false negatives*. That is, a false negative represents a record that is classified as negative but is actually positive. Of the 4242 positive classifications, 819 actually had low incomes, so that there are 819 false positives. A *false positive* represents a record that is classified as positive but is actually negative.

Let TN, FN, FP, and TP represent the numbers of true negatives, false negatives, false positives, and true positives, respectively, in our contingency table. Then we may define our evaluation measures as follows.

$$\text{Overall Error Rate} = \frac{\text{FN} + \text{FP}}{\text{TN} + \text{FN} + \text{FP} + \text{TP}}$$

$$\text{Overall Accuracy} = 1 - \text{Overall Error Rate} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{FN} + \text{FP} + \text{TP}}$$

$$\text{Proportion of False Positives} = \text{PFP} = \frac{\text{FP}}{\text{FP} + \text{TP}}$$

$$\text{Proportion of False Negatives} = \text{PFN} = \frac{\text{FN}}{\text{FN} + \text{TN}}$$

Higher is better for Overall Accuracy, Sensitivity, and Specificity; lower is better for Overall Error Rate, PFP, and PFN.

The *overall error rate*, or simply the *error rate*, is the sum of the false negatives and false positives, divided by the total number of records. Here we have

$$\text{overall error rate} = \frac{\text{FN} + \text{FP}}{\text{TN} + \text{FN} + \text{FP} + \text{TP}} = \frac{2561 + 819}{25,000} = 0.1352$$

To find the *proportion of false negatives*, divide the number of false negatives by the total number of negative classifications. Similarly, to find the *proportion of false positives*, divide the number of false positives by the total number of positive classifications. Here we have

$$\text{proportion of false negatives} = \text{PFN} = \frac{2561}{20,758} = 0.1234$$

$$\text{proportion of false positives} = \text{PFP} = \frac{819}{4242} = 0.1931$$

That is, using the present C5.0 decision tree model, we are more likely to classify an applicant's income incorrectly as high than to classify an applicant's income incorrectly as low. Using error rate, false positive rate, and proportion of false negatives, analysts may compare the accuracy of various models. For example, a C5.0 decision tree model may be compared against a CART decision tree model or a neural network model. Model choice decisions can then be rendered based on the relative rankings of these evaluation measures.

As an aside, in the parlance of hypothesis testing, since the default decision is to find that the applicant has low income, we would have the following hypotheses:

$$H_0 : \text{income} \leq 50,000$$

$$H_a : \text{income} > 50,000$$

where H_0 represents the default, or null, hypothesis, and H_a represents the alternative hypothesis, which requires evidence to support it. A false positive would be considered a *type I error* in this setting, incorrectly rejecting the null hypothesis, while a false negative would be considered a *type II error*, incorrectly accepting the null hypothesis.

14.5 SENSITIVITY AND SPECIFICITY

A good classification model should be *sensitive*, meaning that it should identify a high proportion of the customers who are positive (have high income). *Sensitivity* is defined as

$$\text{Sensitivity} = \frac{\text{Number of true positives}}{\text{Number of actual positives}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

For example, from Table 14.2, we calculate the sensitivity of our income classification model to be:

$$\text{Sensitivity} = \frac{\text{Number of true positives}}{\text{Number of actual positives}} = \frac{3423}{5984} = 0.5720$$

This statistic is interpreted as follows: our classification model has correctly classified 57.20% of the actual high income customers as having high income.

In some fields, such as information retrieval [1], sensitivity is referred to as *recall*. Of course, a perfect classification model would have sensitivity = 1.0 = 100%. However, a null model which simply classified all customers as positive would also have sensitivity = 1.0. Clearly, it is not sufficient to identify the positive responses alone.

A classification model also needs to be *specific*, meaning that it should identify a high proportion of the customers who are negative (have low income). *Specificity* is defined as

$$\text{Specificity} = \frac{\text{Number of true negatives}}{\text{Number of actual negatives}} = \frac{\text{TN}}{\text{FP} + \text{TN}}$$

From Table 14.2, we compute the specificity of our income classification model to be:

$$\text{Specificity} = \frac{\text{Number of true negatives}}{\text{Number of actual negatives}} = \frac{18,197}{19,016} = 0.9569$$

Thus, our classification model has correctly classified 95.69% of the actual low income customers as having low income.

Of course, a perfect classification model would have specificity = 1.0. But so would a model which classifies all customers as low income. A good classification model should have acceptable levels of both sensitivity and specificity, but what

constitutes acceptable varies greatly from domain to domain. Our model specificity of 0.9569 is higher than our model sensitivity of 0.5720, which is probably okay in this instance. In the credit application domain, it may be more important to correctly identify the customers who will default rather than those who will not default, as we discuss next.

14.6 MISCLASSIFICATION COST ADJUSTMENT TO REFLECT REAL-WORLD CONCERNS

Consider this situation from the standpoint of the lending institution. Which error, a false negative or a false positive, would be considered more damaging from the lender’s point of view? If the lender commits a false negative, an applicant who had high income gets turned down for a loan: an unfortunate but not very expensive mistake.

On the other hand, if the lender commits a false positive, an applicant who had low income would be awarded the loan. This error greatly increases the chances that the applicant will default on the loan, which is very expensive for the lender. Therefore, the lender would consider the false positive to be the more damaging type of error and would prefer to minimize the proportion of false positives. The analyst would therefore adjust the C5.0 algorithm’s misclassification cost matrix to reflect the lender’s concerns. Suppose, for example, that the analyst increased the false positive cost from 1 to 2, while the false negative cost remains at 1. Thus, a false positive would be considered twice as damaging as a false negative. The analyst may wish to experiment with various cost values for the two types of errors, to find the combination best suited to the task and business problem at hand.

How did the misclassification cost adjustment affect the performance of the algorithm? Which rate would you expect to increase or decrease, the false negative or the false positive? Do you have an intuition of what might happen to the overall error rate?

Well, we would expect that the proportion of false positives would decrease, since the cost of making such an error has been doubled. Fewer false positives usually mean more false negatives, however. Unfortunately, the overall error rate will probably increase, since there are many more negative predictions made than positive, giving the proportion of false negatives a greater weight in the computation of the overall error rate.

The C5.0 algorithm was rerun, this time including the misclassification cost adjustment. The resulting contingency table is shown in Table 14.3. As expected, the

TABLE 14.3 Contingency table after misclassification cost adjustment

Predicted Category				
Actual category		≤50K	>50K	Total
	≤50K	18,711	305	19,016
	>50K	3307	2677	5984
	Total	22,018	2982	25,000

proportion of false negatives has increased, while the proportion of false positives has decreased. Whereas previously, false positives were more likely to occur, this time the proportion of false positives is lower than the proportion of false negatives. As desired, the proportion of false positives has decreased. However, this has come at a cost. The algorithm, hesitant to classify records as positive due to the higher cost, instead made many more negative classifications, and therefore more false negatives.

$$\text{proportion of false negatives} = \frac{3307}{22,018} = 0.1501, \text{ up from } 0.1234 \text{ previously,}$$

$$\text{proportion of false positives} = \frac{305}{2982} = 0.1023, \text{ down from } 0.1931 \text{ previously.}$$

Unfortunately, the overall error rate has climbed as well:

$$\text{overall error rate} = \frac{3307 + 305}{25,000} = 0.14448, \text{ up from } 0.1352 \text{ previously.}$$

Nevertheless, a higher overall error rate and a higher proportion of false negatives are considered a “good trade” by this lender, who is eager to reduce the loan default rate, which is very costly to the firm. The decrease in the proportion of false positives from 19.31% to 10.23% will surely result in significant savings to the financial lending firm, since fewer applicants who cannot afford to repay the loan will be awarded the loan. Data analysts should note an important lesson here: that we should not be wed to the overall error rate as the best indicator of a good model.

14.7 DECISION COST/BENEFIT ANALYSIS

Company managers may require that model comparisons be made in terms of cost/benefit analysis. For example, in comparing the original C5.0 model before the misclassification cost adjustment (call this *model 1*) against the C5.0 model using the misclassification cost adjustment (call this *model 2*), managers may prefer to have the respective error rates, false negatives and false positives, translated into dollars and cents.

Analysts can provide model comparison in terms of anticipated profit or loss by associating a cost or benefit with each of the four possible combinations of correct and incorrect classifications. For example, suppose that the analyst makes the cost/benefit value assignments shown in Table 14.4. The “–\$300” cost is actually the anticipated average interest revenue to be collected from applicants whose income is actually

TABLE 14.4 Cost/benefit table for each combination of correct/incorrect decision

Outcome	Classification	Actual Value	Cost	Rationale
True negative	$\leq 50,000$	$\leq 50,000$	\$0	No money gained or lost
True positive	$> 50,000$	$> 50,000$	–\$300	Anticipated average interest revenue from loans
False negative	$\leq 50,000$	$> 50,000$	\$0	No money gained or lost
False positive	$> 50,000$	$\leq 50,000$	\$500	Cost of loan default averaged over all loans to $\leq 50,000$ group

>50,000. The \$500 reflects the average cost of loan defaults, averaged over all loans to applicants whose income level is low. Of course, the specific numbers assigned here are subject to discussion and are meant for illustration only.

Using the costs from Table 14.4, we can then compare models 1 and 2:

Cost of Model 1 (False positive cost not doubled):

$$18,197(\$0) + 3423(-\$200) + 2561(\$0) + 819(\$500) = -\$275,100$$

Cost of Model 2 (False positive cost doubled):

$$18,711(\$0) + 2677(-\$200) + 3307(\$0) + 305(\$500) = -\$382,900$$

Negative costs represent profits. Thus, the *estimated cost savings* from deploying model 2, which doubles the cost of a false positive error, is

$$-\$275,100 - (-\$382,900) = \$107,800$$

In other words, the simple data mining step of doubling the false positive cost has resulted in the deployment of a model greatly increasing the company's profit. Is not it amazing what a simple misclassification cost adjustment can mean to the company's bottom line? Thus, even though model 2 suffered from a higher overall error rate and a higher proportion of false negatives, it outperformed model 1 "where it counted," with a lower proportion of false positives, which led directly to a six-figure increase in the company's estimated profit.

14.8 LIFT CHARTS AND GAINS CHARTS

In Chapter 12 we were introduced to the concept of *lift* for association rules. For classification models, lift is a concept, originally from the marketing field, which seeks to compare the response rates with and without using the classification model. Lift charts and gains charts are graphical evaluative methods for assessing and comparing the usefulness of classification models. We shall explore these concepts by continuing our examination of the C5.0 models for classifying income.

Suppose that the financial lending firm is interested in identifying high income persons to put together a targeted marketing campaign for a new platinum credit card. In the past, marketers may have simply canvassed an entire list of contacts without regard to clues about the contact's income. Such blanket initiatives are expensive and tend to have low response rates. It is much better to apply demographic information that the company may have about the list of contacts, build a model to predict which contacts will have high income, and restrict the canvassing to these contacts classified as high income. The cost of the marketing program will then be much reduced and the response rate may be higher.

A good classification model should identify in its positive classifications (the >50,000 column in Tables 14.2 and 14.3), a group that has a higher proportion of positive "hits" than the database as a whole. The concept of *lift* quantifies this.

We define *lift* as the proportion of positive hits in the set of the model's positive classifications, divided by the proportion of positive hits in the data set overall:

$$\text{lift} = \frac{\text{proportion of positive hits in set of positive classifications}}{\text{proportion of positive hits in data set as a whole}}$$

For example, in Table 14.2, model 1 identifies 4242 records as being classified positive (*income* > 50,000). This is the set of positive classifications. Of these 4242, 3423 records are positive hits; that is, the actual value of *income* is > 50,000. This gives us $3423/4242 = 0.8069$ as the proportion of positive hits in the set of positive classifications. Now, in the data set as a whole, 5984 of the 25,000 records have *income* > 50,000, giving us $5984/25,000 = 0.23936$ as the proportion of positive hits in the data set as a whole. The lift, measured at the 4242 positively predicted records, is therefore $0.8069/0.23936 = 3.37$.

Lift is a function of sample size, which is why we had to specify that the lift of 3.37 for model 1 was measured at $n = 4242$ records. When calculating lift, the software will first sort the records by the probability of being classified positive. The lift is then calculated for every sample size from $n = 1$ to $n =$ the size of the data set. A chart is then produced which graphs lift against the percentile of the data set.

Consider Figure 14.2, which represents the lift chart for model 1. Note that lift is highest at the lowest percentiles, which makes sense since the data are sorted according to the most likely positive hits. The lowest percentiles have the highest proportion of positive hits. As the plot moves from left to right, the positive hits tend to get “used up,” so that the proportion steadily decreases until the lift finally equals exactly 1 when the entire data set is considered the sample. Therefore, for any lift chart, the highest lift is always obtained with the smallest sample sizes.

Now, 4242 records represents about the 17th percentile of the 25,000 total records. Note in Figure 14.2 that the lift at about the 17th percentile would be near 3.37, as we calculated above. If our market research project required merely the

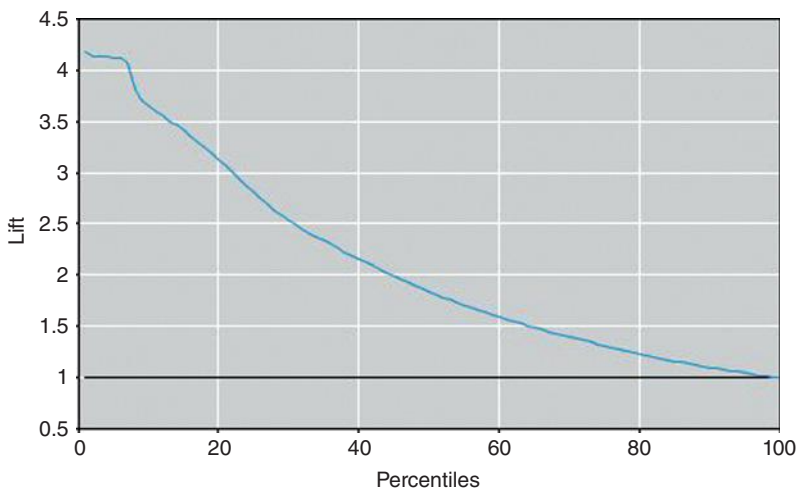


Figure 14.2 Lift chart for model 1: strong lift early, then falls away rapidly.

most likely 5% of records, the lift would have been higher, about 4.1, as shown in Figure 14.2. On the other hand, if the project required 60% of all records, the lift would have fallen off to about 1.6. Since the data are sorted by positive propensity, the further we reach into the data set, the lower our overall proportion of positive hits becomes. Another balancing act is required: between reaching lots of contacts and having a high expectation of success per contact.

Lift charts are often presented in their cumulative form, where they are denoted as *cumulative lift charts*, or *gains charts*. The gains chart associated with the lift chart in Figure 14.2 is presented in Figure 14.3. The diagonal on the gains chart is analogous to the horizontal axis at *lift* = 1 on the lift chart. Analysts would like to see gains charts where the upper curve rises steeply as one moves from left to right and then gradually flattens out. In other words, one prefers a deeper “bowl” to a shallower bowl. How do you read a gains chart? Suppose that we canvassed the top 20% of our contact list (percentile = 20). By doing so, we could expect to reach about 62% of the total number of high income persons on the list. Would doubling our effort also double our results? No. Canvassing the top 40% on the list would enable us to reach approximately 85% of the high income persons on the list. Past this point, the law of diminishing returns is strongly in effect.

Lift charts and gains charts can also be used to compare model performance. Figure 14.4 shows the combined lift chart for models 1 and 2. The figure shows that when it comes to model selection, a particular model may not be uniformly preferable. For example, up to about the 6th percentile, there appears to be no apparent difference in model lift. Then, up to approximately the 17th percentile, model 2 is preferable, providing slightly higher lift. Thereafter, model 1 is preferable.

Hence, if the goal were to canvass up to the top 17% or so of the people on the contact list with high incomes, model 2 would probably be selected. However, if the goal were to extend the reach of the marketing initiative to 20% or more of the likely contacts with high income, model 1 would probably be selected. This question of

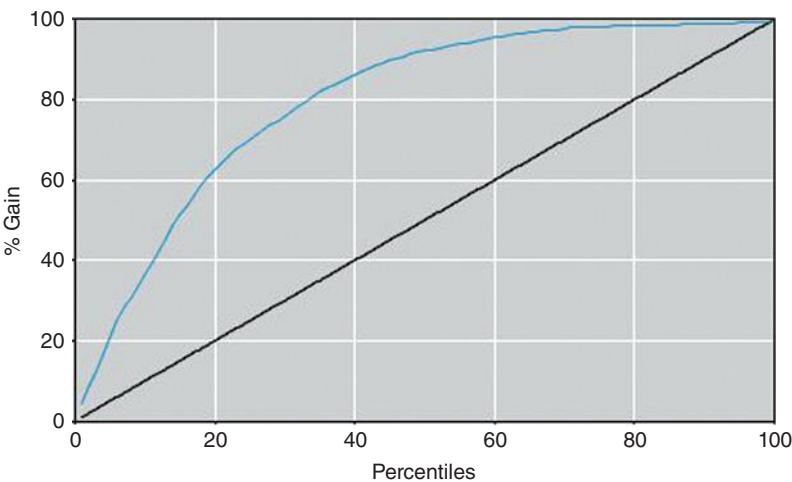


Figure 14.3 Gains chart for model 1.

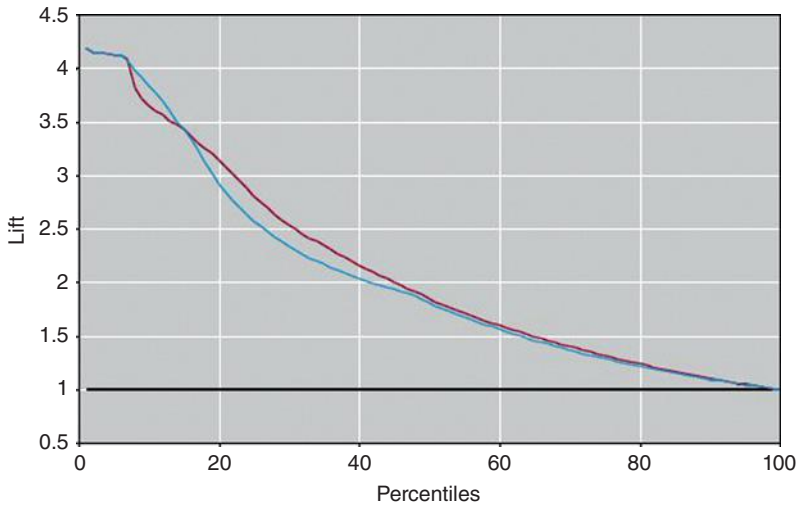


Figure 14.4 Combined lift chart for models 1 and 2.

multiple models and model choice is an important one, which we spend much time discussing in Reference 1.

Finally, when applying misclassification costs in cost/benefit analysis, one should use profit charts, which are discussed in *Data Mining and Predictive Analytics*.²

It is to be stressed that model evaluation techniques should be performed on the test data set, rather than on the training set, or on the data set as a whole. (The entire *Adult* data set was used here so that readers could replicate the results if they so choose.)

14.9 INTERWEAVING MODEL EVALUATION WITH MODEL BUILDING

In Chapter 1 the graphic representing the CRISP-DM standard process for data mining contained a feedback loop between the model building and evaluation phases. In Chapter 6 we presented a *methodology for building and evaluating a data model*. Where do the methods for model evaluation from Chapter 14 fit into these processes?

We would recommend that model evaluation become a nearly “automatic” process, performed to a certain degree whenever a new model is generated. Therefore, at any point in the process, we may have an accurate measure of the quality of the current or working model. Therefore, it is suggested that model evaluation be interwoven seamlessly into the *methodology for building and evaluating a data model* presented in Chapter 6, being performed on the models generated from each of the training set and the test set. For example, when we adjust the provisional model to minimize the error rate on the test set, we may have at our fingertips the proportion

²By Daniel Larose and Chantal Larose, John Wiley and Sons, Inc., 2015, to appear.

of false positives, the proportion of false negatives, the overall error rate, the lift charts, and the gains charts. These evaluative measures can then point the analyst in the proper direction for best ameliorating any drawbacks of the working model.

14.10 CONFLUENCE OF RESULTS: APPLYING A SUITE OF MODELS

In Olympic figure skating, the best-performing skater is not selected by a single judge alone. Instead, a suite of several judges is called upon to select the best skater from among all the candidate skaters. Similarly in model selection, whenever possible, the analyst should not depend solely on a single data mining method. Instead, he or she should seek a *confluence of results* from a suite of different data mining models.

For example, for the *adult* database, Figures 8.5, 8.7, and 9.9 show that the variables listed in Table 14.5 are the most influential (ranked roughly in order of importance) for classifying income, as identified by CART, C5.0, and the neural network algorithm, respectively. Although there is not a perfect match in the ordering of the important variables, there is still much that these three separate classification algorithms have uncovered, including the following:

- All three algorithms identify *Marital_Status*, *education-num*, *capital-gain*, *capital-loss*, and *hours-per-week* as the most important variables, except for the neural network, where *age* snuck in past *capital-loss*.
- None of the algorithms identified either *work-class* or *sex* as important variables, and only the neural network identified *age* as important.
- The algorithms agree on various ordering trends, such as *education-num* is more important than *hours-per-week*.

When we recall the strongly differing mathematical bases on which these three data mining methods are built, it may be considered remarkable that such convincing concurrence prevails among them with respect to classifying income. Remember that CART bases its decisions on the “goodness of split” criterion $\Phi(split)$, that C5.0 applies an information-theoretic approach, and that neural networks base their learning on back-propagation. Yet these three different algorithms represent streams that broadly speaking, have come together, forming a *confluence* of results. In this way, the models act as validation for each other.

TABLE 14.5 Most important variables for classifying income, as identified by CART, C5.0, and the neural network algorithm

CART	C5.0	Neural Network
<i>Marital_Status</i>	<i>Capital-gain</i>	<i>Capital-gain</i>
<i>Education-num</i>	<i>Capital-loss</i>	<i>Education-num</i>
<i>Capital-gain</i>	<i>Marital_Status</i>	<i>Hours-per-week</i>
<i>Capital-loss</i>	<i>Education-num</i>	<i>Marital_Status</i>
<i>Hours-per-week</i>	<i>Hours-per-week</i>	<i>Age Capital-loss</i>

THE R ZONE

Install the required package

```
install.packages("C50")
library("C50")
```

The confusion matrix

After using the C5.0 package, the confusion matrix is included in the output of summary()
See Chapter 8 for data preparation and code to implement the C5.0 package

Add costs to the model

```
#After running data preparation from Chapter 8
x <- dat[,c(2,6, 9, 10, 16, 17, 18, 19, 20)]
y <- dat$income
# Without weights:
c50fit <- C5.0(x, y)
summary(c50fit)
```

Decision Tree		
size	Errors	
78	3286(13.1%)	<<
(a)	(b)	<-classified as
17902	1114	(a): class <=50K.
2172	3812	(b): class >50K.

```
# With weights:
costm <- matrix(c(1, 2, 1, 1),
  byrow = FALSE,
  2, 2)
c50cost <- C5.0(x, y,
  costs = costm)
summary(c50cost)
```

Decision Tree			
size	Errors	Cost	
55	3626(14.5%)	0.16	<<
(a)	(b)	<-classified as	
18762	254	(a): class <=50K.	
3372	2612	(b): class >50K.	

REFERENCE

1. Zdravko Markov and Daniel Larose, *Data Mining the Web, Uncovering Patterns in Web Content, Structure, and Usage*, John Wiley and Sons, New York, 2007.

EXERCISES

HANDS-ON ANALYSIS

Use the *churn* data set at the book series website for the following exercises. Make sure that the correlated variables have been accounted for.

1. Apply a CART model for predicting *churn*. Use default misclassification costs. Determine the following measures.
 - a. Proportion of false positives.
 - b. Proportion of false negatives.
 - c. Overall error rate.
 - d. Overall model accuracy ($1 - \text{overall error rate}$).
 - e. Sensitivity
 - f. Specificity
2. In a typical churn model, in which interceding with a potential churner is relatively cheap but losing a customer is expensive, which error is more costly, a false negative or a false positive (where positive = customer predicted to churn)? Explain.
3. Based on your answer to Exercise 2, adjust the misclassification costs for your CART model to reduce the prevalence of the more costly type of error. Rerun the CART algorithm. Compare the false positive, false negative, sensitivity, specificity, and overall error rate with the previous model. Discuss the trade-off between the various rates in terms of cost for the company.
4. Perform a cost/benefit analysis for the default CART model from exercise 1 as follows. Assign a cost or benefit in dollar terms for each combination of false and true positives and negatives, similar to Table 14.4. Then, using the contingency table, find the overall anticipated cost.
5. Perform a cost/benefit analysis for the CART model with the adjusted misclassification costs. Use the same cost/benefits assignments as for the default model. Find the overall anticipated cost. Compare with the default model, and formulate a recommendation as to which model is preferable.
6. Construct a lift chart for the default CART model. What is the estimated lift at 20%? 33%? 40%? 50%?
7. Construct a gains chart for the default CART model. Explain the relationship between this chart and the lift chart.
8. Construct a lift chart for the CART model with the adjusted misclassification costs. What is the estimated lift at 20%? 33%? 40%? 50%?
9. Construct a single lift chart for both of the CART models. Which model is preferable over which regions?
10. Now turn to a C4.5 decision tree model, and redo Exercises 1–9. Compare the results. Which model is preferable?
11. Next, apply a neural network model to predict churn. Calculate the following measures.
 - a. Proportion of false positives.
 - b. Proportion of false negatives.
 - c. Overall error rate.
 - d. Overall model accuracy ($1 - \text{overall error rate}$).
 - e. Sensitivity
 - f. Specificity

12. Construct a lift chart for the neural network model. What is the estimated lift at 20%? 33%? 40%? 50%?
13. Construct a single lift chart which includes the better of the two CART models, the better of the two C4.5 models, and the neural network model. Which model is preferable over which regions?
14. In view of the results obtained above, discuss the overall quality and adequacy of our *churn* classification models. ■

DATA SUMMARIZATION AND VISUALIZATION

PART 1	SUMMARIZATION 1: BUILDING BLOCKS OF DATA ANALYSIS	294
PART 2	VISUALIZATION: GRAPHS AND TABLES FOR SUMMARIZING AND ORGANIZING DATA	296
PART 3	SUMMARIZATION 2: MEASURES OF CENTER, VARIABILITY, AND POSITION	301
PART 4	SUMMARIZATION AND VISUALIZATION OF BIVARIATE RELATIONSHIPS	304

Here, we present a very brief review of methods for summarizing and visualizing data. For deeper coverage, please see *Discovering Statistics*, by Daniel Larose (second edition, W.H. Freeman, New York, 2013).

PART 1 SUMMARIZATION 1: BUILDING BLOCKS OF DATA ANALYSIS

- **Descriptive statistics** refers to methods for summarizing and organizing the information in a data set.
Consider Table A.1, which we will use to illustrate some statistical concepts.
- The entities for which information is collected are called the **elements**. In Table A.1, the elements are the 10 applicants. Elements are also called **cases** or **subjects**.
- A **variable** is a characteristic of an element, which takes on different values for different elements. The variables in Table A.1 are *marital status*, *mortgage*, *income*, *rank*, *year*, and *risk*. Variables are also called **attributes**.

TABLE A.1 Characteristics of 10 loan applicants

Applicant	Marital Status	Mortgage	Income (\$)	Income Rank	Year	Risk
1	Single	y	38,000	2	2009	Good
2	Married	y	32,000	7	2010	Good
3	Other	n	25,000	9	2011	Good
4	Other	n	36,000	3	2009	Good
5	Other	y	33,000	4	2010	Good
6	Other	n	24,000	10	2008	Bad
7	Married	y	25,100	8	2010	Good
8	Married	y	48,000	1	2007	Good
9	Married	y	32,100	6	2009	Bad
10	Married	y	32,200	5	2010	Good

- The set of variable values for a particular element is an **observation**. Observations are also called **records**. The observation for Applicant 2 is:

Applicant	Marital Status	Mortgage	Income (\$)	Income Rank	Year	Risk
2	Married	y	32,000	7	2010	Good

- Variables can be either *qualitative* or *quantitative*.
 - A **qualitative variable** enables the elements to be classified or categorized according to some characteristic. The qualitative variables in Table A.1 are *marital status*, *mortgage*, *rank*, and *risk*. Qualitative variables are also called **categorical variables**.
 - A **quantitative variable** takes numeric values and allows arithmetic to be meaningfully performed on it. The quantitative variables in Table A.1 are *income* and *year*. Quantitative variables are also called **numerical variables**.
- Data may be classified according to four *levels of measurement*: *nominal*, *ordinal*, *interval*, and *ratio*. Nominal and ordinal data are categorical; interval and ratio data are numerical.
 - **Nominal data** refer to names, labels, or categories. There is no natural ordering, nor may arithmetic be carried out on nominal data. The nominal variables in Table A.1 are *marital status*, *mortgage*, and *risk*.
 - **Ordinal data** can be rendered into a particular order. However, arithmetic cannot be meaningfully carried out on ordinal data. The ordinal variable in Table A.1 is *income rank*.
 - **Interval data** consist of quantitative data defined on an interval without a natural zero. Addition and subtraction may be performed on interval data. The interval variable in Table A.1 is *year*. (Note that there is no “year zero.” The calendar goes from 1 B.C. to 1 A.D.)
 - **Ratio data** are quantitative data for which addition, subtraction, multiplication, and division may be performed. A natural zero exists for ratio data. The interval variable in Table A.1 is *income*.
- A numerical variable that can take either a finite or a countable number of values is a **discrete** variable, for which each value can be graphed as a

separate point, with space between each point. The discrete variable in Table A.1 is *year*.

- A numerical variable that can take infinitely many values is a **continuous variable**, whose possible values form an interval on the number line, with no space between the points. The continuous variable in Table A.1 is *income*.
- A **population** is the set of all elements of interest for a particular problem. A **parameter** is a characteristic of a population. For example, the population is the set of all American voters, and the parameter is the proportion of the population who supports a \$1 per ton tax on carbon.
 - The value of a parameter is usually unknown, but it is a constant.
- A **sample** consists of a subset of the population. A characteristic of a sample is called a **statistic**. For example, the sample is the set of American voters in your classroom, and the statistic is the proportion of the sample who supports a \$1 per ton tax on carbon.
 - The value of a statistic is usually known, but it changes from sample to sample.
- A **census** is the collection of information from every element in the population. For example, the census here would be to find from every American voter whether they support a \$1 per ton tax on carbon. Such a census is impractical, so we turn to statistical inference.
- **Statistical inference** refers to methods for estimating or drawing conclusions about population characteristics based on the characteristics of a sample of that population. For example, suppose 50% of the voters in your classroom support the tax; using statistical inference, we would *infer* that 50% of all American voters support the tax. Obviously, there are problems with this. The sample is neither random nor representative. The estimate does not have a confidence level, and so on.
- When we take a sample for which each element has an equal chance of being selected, we have a **random sample**.
- A **predictor variable** is a variable whose value is used to help predict the value of the *response variable*. The predictor variables in Table A.1 are all the variables except *risk*.
- A **response variable** is a variable of interest whose value is presumably determined at least in part by the set of predictor variables. The response variable in Table A.1 is *risk*.

PART 2 VISUALIZATION: GRAPHS AND TABLES FOR SUMMARIZING AND ORGANIZING DATA

2.1 Categorical Variables

- The **frequency** (or **count**) of a category is the number of data values in each category. The **relative frequency** of a particular category for a categorical variable equals its frequency divided by the number of cases.

TABLE A.2 Frequency distribution and relative frequency distribution

Category of Marital Status	Frequency	Relative Frequency
Married	5	0.5
Other	4	0.4
Single	1	0.1
Total	10	1.0

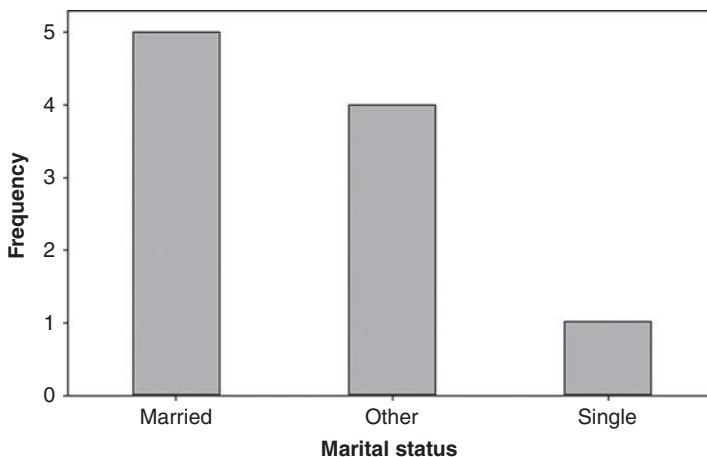
- A **(relative) frequency distribution** for a categorical variable consists of all the categories that the variable assumes, together with the (relative) frequencies for each value. The frequencies sum to the number of cases; the relative frequencies sum to 1.

For example Table A.2 contains the frequency distribution and relative frequency distribution for the variable *marital status* for the data from Table A.1.

- A **bar chart** is a graph used to represent the frequencies or relative frequencies for a categorical variable. Note that the bars do not touch.
 - A **Pareto chart** is a bar chart where the bars are arranged in decreasing order. Figure A.1 is an example of a Pareto chart.
- A **pie chart** is a circle divided into slices, with the size of each slice proportional to the relative frequency of the category associated with that slice. Figure A.2 shows a pie chart of *marital status*.

2.2 Quantitative Variables

- Quantitative data are grouped into **classes**. The **lower (upper) class limit** of a class equals the smallest (largest) value within that class. The **class width** is the difference between successive lower class limits.

Figure A.1 Bar chart for *marital status*.

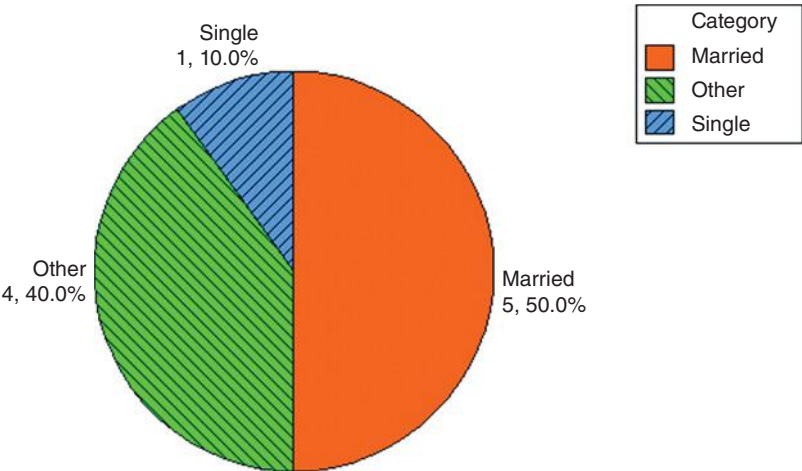


Figure A.2 Pie chart of *marital status*.

- For quantitative data, a **(relative) frequency distribution** divides the data into nonoverlapping classes of equal class width. Table A.3 shows the frequency distribution and relative frequency distribution of the continuous variable *income* from Table A.1.
- A **cumulative (relative) frequency distribution** shows the total number (relative frequency) of data values less than or equal to the upper class limit. See Table A.4.
- A **distribution** of a variable is a graph, table, or formula that specifies the values and frequencies of the variable for all elements in the data set. For example, Table A.3 represents the distribution of the variable *income*.
- A **histogram** is a graphical representation of a (relative) frequency distribution for a quantitative variable. See Figure A.3. Note that histograms represent a simple version of *data smoothing* and can thus vary in shape depending on the number and width of the classes. Therefore, histograms should be interpreted with caution. See *Discovering Statistics*, by Daniel Larose (W.H. Freeman) section 2.4 for an example of a data set presented as *both* symmetric and right-skewed by altering the number and width of the histogram classes.
- A **stem-and-leaf display** shows the shape of the data distribution while retaining the original data values in the display, either exactly or approximately. The

TABLE A.3 Frequency distribution and relative frequency distribution of *income*

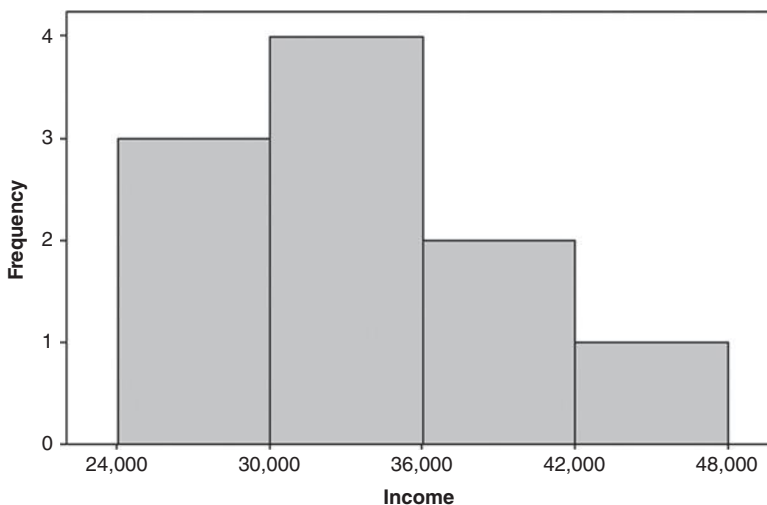
Class of <i>Income</i>	Frequency	Relative Frequency
\$24,000–\$29,999	3	0.3
\$30,000–\$35,999	4	0.4
\$36,000–\$41,999	2	0.2
\$42,000–\$48,999	1	0.1
Total	10	1.0

TABLE A.4 Cumulative frequency distribution and cumulative relative frequency distribution of *income*

Class of <i>Income</i>	Cumulative Frequency	Cumulative Relative Frequency
\$24,000–\$29,999	3	0.3
\$30,000–\$35,999	7	0.7
\$36,000–\$41,999	9	0.9
\$42,000–\$48,999	10	1.0

leaf units are defined to equal a power of 10, and the stem units are 10 times the leaf units. Then each leaf represents a data value, through a stem-and-leaf combination. For example, in Figure A.4, the leaf units (right-hand column) are 1000s and the stem units (left-hand column) are 10,000s. So “2 4” represents $2 \times 10,000 + 4 \times 1000 = \$24,000$, while “2 55” represents two equal incomes of \$25,000 (one of which is exact, the other approximate, \$25,100). Note that Figure A.4, turned 90 degrees to the left, presents the shape of the data distribution.

- In a **dotplot**, each dot represents one or more data values, set above the number line. See Figure A.5.
- A distribution is **symmetric** if there exists an axis of symmetry (a line) that splits the distribution into two halves that are approximately mirror images of each other (Figure A.6a).
- **Right-skewed** data have a longer tail on the right than the left (Figure A.6b). **Left-skewed** data have a longer tail on the left than the right (Figure A.6c).

Figure A.3 Histogram of *income*.

Stem-and-leaf of Income
Leaf Unit = 1000

```
2  4
2  55
3  2223
3  68
4
4  8
```

Figure A.4 Stem-and-leaf display of *income*.

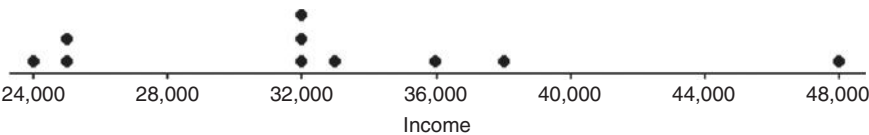


Figure A.5 Dotplot of *income*.

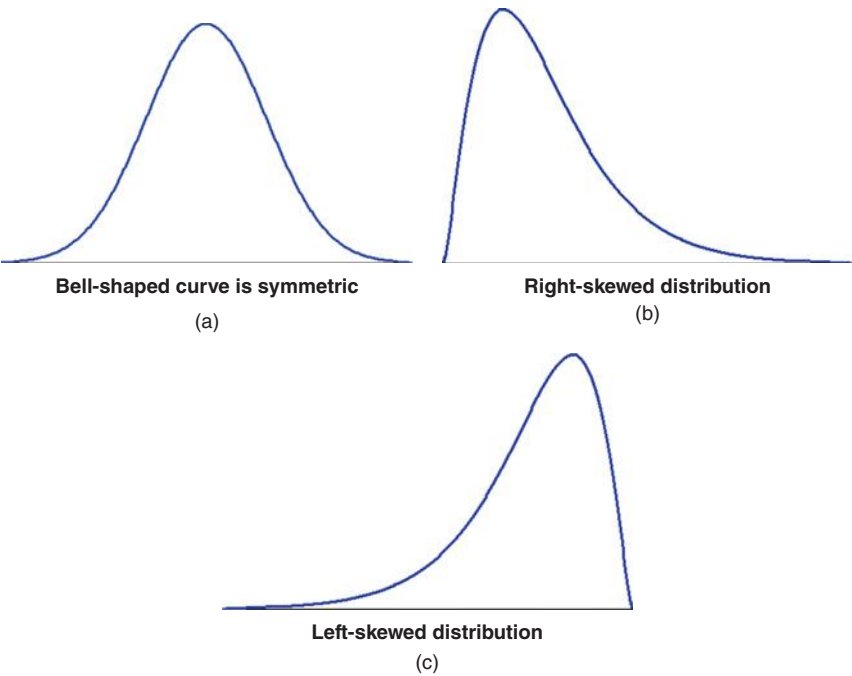


Figure A.6 Symmetric and skewed curves.

PART 3 SUMMARIZATION 2: MEASURES OF CENTER, VARIABILITY, AND POSITION

- The *summation notation* \sum^x means to add up all the data values x . The sample size is n and the population size is N .
- **Measures of center** indicate where on the number line the central part of the data is located. The measures of center we will learn are the *mean*, the *median*, the *mode*, and the *midrange*.
 - The **mean** is the *arithmetic average* of a data set. To calculate the mean, add up the values and divide by the number of values. The mean income from Table A.1 is

$$\frac{38,000 + 32,000 + \dots + 32,200}{10} = \frac{325,400}{10} = \$32,540$$

- The **sample mean** is the arithmetic average of a sample, and is denoted \bar{x} (“*x-bar*”).
- The **population mean** is the arithmetic average of a population, and is denoted μ (“*myu*”, the Greek letter for m).
- The **median** is the middle data value, when there is an odd number of data values and the data have been sorted into ascending order. If there is an even number, the median is the mean of the two middle data values. When the income data are sorted into ascending order, the two middle values are \$32,100 and \$32,200, the mean of which is the median income, \$32,150.
- The **mode** is the data value that occurs with the greatest frequency. Both quantitative and categorical variables can have modes, but only quantitative variables can have means or medians. Each income value occurs only once, so there is no mode. The mode for *year* is 2010, with a frequency of 4.
- The **midrange** is the average of the maximum and minimum values in a data set. The midrange income is

$$\begin{aligned} \text{midrange (income)} &= \frac{(\max(\text{income}) + \min(\text{income}))}{2} = \frac{48,000 + 24,000}{2} \\ &= \$36,000 \end{aligned}$$

- **Skewness and measures of center.** The following are tendencies, and not strict rules.
 - For symmetric data, the mean and the median are approximately equal.
 - For right-skewed data, the mean is greater than the median.
 - For left-skewed data, the median is greater than the mean.
- **Measures of variability** quantify the amount of *variation*, *spread*, or *dispersion* present in the data. The measures of variability we will learn are the *range*, the *variance*, the *standard deviation*, and, later, the *interquartile range*.

- The **range** of a variable equals the difference between the maximum and minimum values. The range of *income* is

$$\text{Range} = \max(\text{income}) - \min(\text{income}) = 48,000 - 24,000 = \$24,000.$$

- A **deviation** is the signed difference between a data value, and the mean value. For Applicant 1, the deviation in *income* equals $x - \bar{x} = 38,000 - 32,540 = 5,460$. For any conceivable data set, the *mean deviation* always equals zero, because the sum of the deviations equals zero.
- The **population variance** is the mean of the squared deviations, denoted as σ^2 (“sigma-squared”):

$$\sigma^2 = \frac{\sum (x - \mu)^2}{N}$$

- The **population standard deviation** is the square root of the population variance: $\sigma = \sqrt{\sigma^2}$.
- The **sample variance** is approximately the mean of the squared deviations, with n replaced by $n - 1$ in the denominator in order to make it an *unbiased estimator* of σ^2 . (An **unbiased estimator** is a statistic whose expected value equals its target parameter.)

$$s^2 = \frac{\sum (x - \bar{x})^2}{n - 1}$$

- The **sample standard deviation** is the square root of the sample variance: $s = \sqrt{s^2}$.
- The variance is expressed in *units squared*, an interpretation that may be opaque to nonspecialists. For this reason, the standard deviation, which is expressed in the original units, is preferred when reporting results. For example, the sample variance of *income* is $s^2 = 51,860,444$ *dollars squared*, the meaning of which may be unclear to clients. Better to report the sample standard deviation $s = \$7201$.
- The sample standard deviation s is interpreted as the size of the *typical deviation*, that is, the size of the typical difference between data values and the mean data value. For example, incomes typically deviate from their mean by \$7201.
- **Measures of position** indicate the relative position of a particular data value in the data distribution. The measures of position we cover here are the *percentile*, the *percentile rank*, the *Z-score*, and the *quartiles*.
 - The ***pth* percentile** of a data set is the data value such that p percent of the values in the data set are at or below this value. The 50th percentile is the median. For example, the median *income* is \$32,150, and 50% of the data values lie at or below this value.
 - The **percentile rank** of a data value equals the percentage of values in the data set that are at or below that value. For example, the percentile rank

of Applicant 1's income of \$38,000 is 90%, since that is the percentage of incomes equal to or less than \$38,000.

- The **Z-score** for a particular data value represents how many standard deviations the data value lies above or below the mean. For a sample, the Z-score is

$$Z\text{-score} = \frac{x - \bar{x}}{s}$$

For Applicant 6, the Z-score is

$$\frac{24,000 - 32,540}{7201} \approx -1.2$$

The income of Applicant 6 lies 1.2 standard deviations below the mean.

- We may also find data values, given a Z-score. Suppose no loans will be given to those with incomes more than 2 standard deviations below the mean. Here, $Z\text{-score} = -2$, and the corresponding minimum income is

$$\text{Income} = Z\text{-score} \cdot s + \bar{x} = (-2)(7201) + 32,540 = \$18,138$$

No loans will be provided to the applicants with incomes below \$18,138.

- If the data distribution is normal, then the **Empirical Rule** states:
 - About 68% of the data lies within 1 standard deviation of the mean,
 - About 95% of the data lies within 2 standard deviations of the mean,
 - About 99.7% of the data lies within 3 standard deviations of the mean.
- The **first quartile (Q1)** is the 25th percentile of a data set; the **second quartile (Q2)** is the 50th percentile (median); and the **third quartile (Q3)** is the 75th percentile.
- The **interquartile range (IQR)** is a measure of variability that is not sensitive to the presence of outliers. $IQR = Q3 - Q1$.
- In the **IQR method for detecting outliers**, a data value x is an outlier if either
 - $x \leq Q1 - 1.5(IQR)$, or
 - $x \geq Q3 + 1.5(IQR)$.
- The **five-number summary** of a data set consists of the *minimum*, $Q1$, the *median*, $Q3$, and the *maximum*.
- The **boxplot** is a graph based on the five-number summary, useful for recognizing symmetry and skewness. Suppose for a particular data set (not from Table A.1), we have $Min = 15$, $Q1 = 29$, $Median = 36$, $Q3 = 42$, and $Max = 47$. Then the boxplot is shown in Figure A.7.
 - The box covers the “middle half” of the data from $Q1$ to $Q3$.
 - The left whisker extends down to the minimum value which is not an outlier.
 - The right whisker extends up to the maximum value that is not an outlier.
 - When the left whisker is longer than the right whisker, then the distribution is left-skewed. And vice versa.

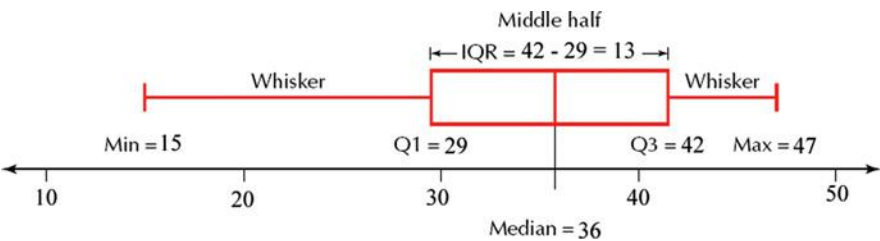


Figure A.7 Boxplot of left-skewed data.

- When the whiskers are about equal in length, the distribution is symmetric. The distribution in Figure A.7 shows evidence of being left-skewed.

PART 4 SUMMARIZATION AND VISUALIZATION OF BIVARIATE RELATIONSHIPS

- A **bivariate relationship** is the relationship between two variables.
- The relationship between two categorical variables is summarized using a **contingency table**, which is a crosstabulation of the two variables, and contains a cell for every combination of variable values (i.e., for every contingency). Table A.5 is the contingency table for the variables *mortgage* and *risk*. The total column contains the **marginal distribution** for *risk*, that is, the frequency distribution for this variable alone. Similarly the total row represents the marginal distribution for *mortgage*.
- Much can be learned from a contingency table. The *baseline proportion* of *bad risk* is $2/10 = 20\%$. However, the proportion of *bad risk* for applicants without a mortgage is $1/3 = 33\%$, which is higher than the baseline; and the proportion of *bad risk* for applicants with a mortgage is only $1/7 = 14\%$, which is lower than the baseline. Thus, whether or not the applicant has a mortgage is useful for predicting risk.
- A **clustered bar chart** is a graphical representation of a contingency table. Figure A.8 shows the clustered bar chart for *risk*, clustered by *mortgage*. Note that the disparity between the two groups is immediately obvious.
- To summarize the relationship between a quantitative variable and a categorical variable, we calculate summary statistics for the quantitative variable for each level of the categorical variable. For example, Minitab provided the following

TABLE A.5 Contingency table for *mortgage* versus *risk*

		Mortgage		Total
		Yes	No	
Risk	Good	6	2	8
	Bad	1	1	2
	Total	7	3	10

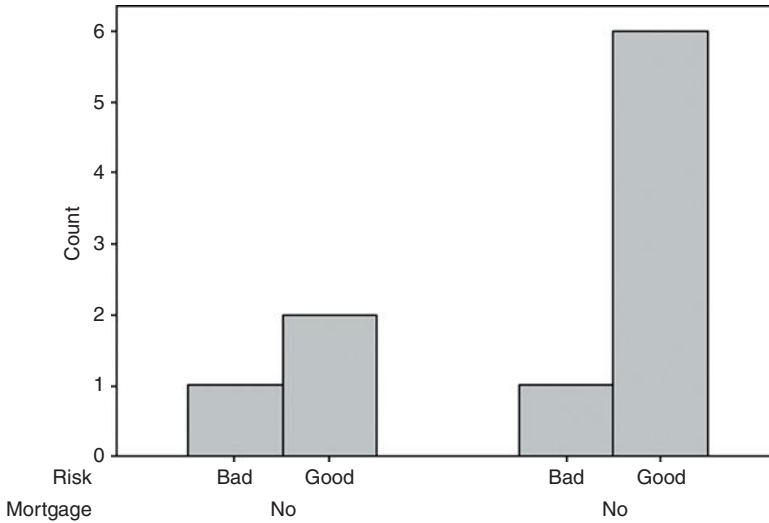


Figure A.8 Clustered bar chart for *risk*, clustered by *mortgage*.

summary statistics for *income*, for records with *bad risk* and for records with *good risk*. All summary measures are larger for *good risk*. Is the difference significant? We need to perform a hypothesis test to find out (Chapter 4).

Descriptive Statistics: Income

Variable	Risk	Mean	StDev	Minimum	Median	Maximum
Income	Bad	28050	5728	24000	28050	32100
	Good	33663	7402	25000	32600	48000

- To visualize the relationship between a quantitative variable and a categorical variable, we may use an **individual value plot**, which is essentially a set of vertical dotplots, one for each category in the categorical variable. Figure A.9 shows the individual value plot for *income* versus *risk*, showing that incomes for *good risk* tend to be larger.

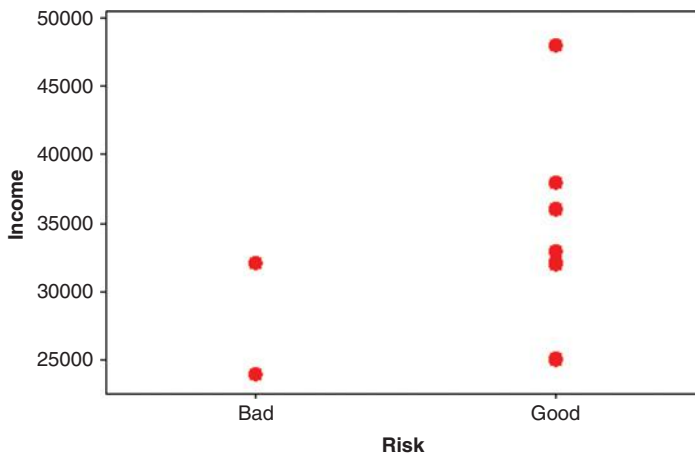


Figure A.9 Individual value plot of *income* versus *risk*.

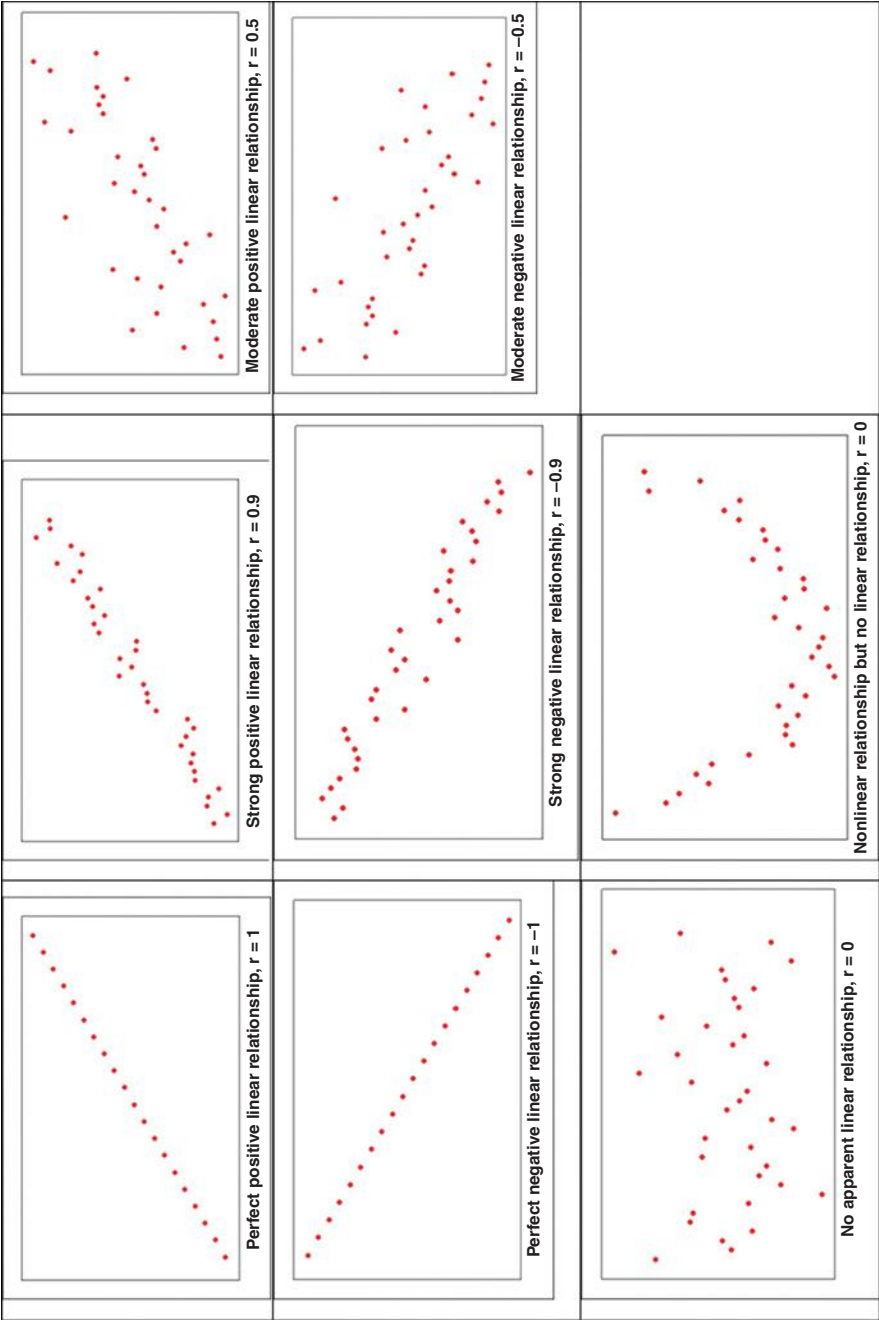


Figure A.10 Some possible relationships between x and y .

- A **scatter plot** is used to visualize the relationship between two quantitative variables, x and y . Each (x, y) point is graphed on a Cartesian plane, with the x axis on the horizontal and the y axis on the vertical. Figure A.10 shows eight scatter plots, showing some possible types of relationships between the variables, along with the value of the *correlation coefficient* r .
- The **correlation coefficient** r quantifies the strength and direction of the linear relationship between two quantitative variables. The correlation coefficient is defined as

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{(n - 1) s_x s_y}$$

where s_x and s_y represent the standard deviation of the x -variable and the y -variable, respectively. $-1 \leq r \leq 1$.

- In data mining, where there are a large number of records (over 1000), even small values of r , such as $-0.1 \leq r \leq 0.1$ may be statistically significant.
- If r is positive and significant, we say that x and y are *positively correlated*. An increase in x is associated with an increase in y .
- If r is negative and significant, we say that x and y are *negatively correlated*. An increase in x is associated with a decrease in y .

INDEX

- Adaptation in Kohonen networks, 230
- Affinity analysis, 247–249
- Agglomerative methods, 209–214
- Analysis of variance (ANOVA), 115–117
- Anomalous fields, 71–72
- Antecedent, 250
- Appendix: Data Summarization and Visualization, 294–307
 - attribute, 294
 - bar chart, 297
 - bivariate relationship, 304
 - boxplot, 303
 - case, 294
 - categorical variable, 296
 - census, 296
 - class limits, 297
 - class width, 297
 - clustered bar chart, 304
 - contingency table, 304
 - continuous variable, 296
 - correlation coefficient r , 307
 - count, 296
 - cumulative frequency distribution, 298
 - descriptive statistics, 294
 - deviation, 302
 - discrete variable, 295
 - distribution, 298
 - dotplot, 299
 - element, 294
 - Empirical Rule, 303
 - five-number summary, 303
 - frequency, 296
 - histogram, 298
 - individual value plot, 305
 - interquartile range (IQR), 303
 - interval data, 295
 - IQR method of detecting outliers, 303
 - left-skewed, 299
 - levels of measurement, 295
 - marginal distribution, 304
 - mean, 301
 - measures of center, 301
 - measures of position, 302
 - measures of variability, 301
 - median, 301
 - midrange, 301
 - mode, 301
 - nominal data, 295
 - numerical variable, 295
 - observation, 295
 - ordinal data, 295
 - parameter, 296
 - Pareto chart, 297
 - percentile, 302
 - percentile rank, 302
 - pie chart, 297
 - population, 296
 - population mean, 301
 - population standard deviation, 302
 - population variance, 302
 - predictor variable, 296
 - qualitative variable, 295
 - quantitative variable, 295
 - quartile, 303
 - random sample, 296
 - range, 302
 - ratio data, 295
 - record, 295
 - relative frequency, 296
 - relative frequency distribution, 297, 298
 - response variable, 296
 - right-skewed, 299
 - sample, 296
 - sample mean, 301
 - sample standard deviation, 302
 - sample variance, 302

Discovering Knowledge in Data: An Introduction to Data Mining, Second Edition.

By Daniel T. Larose and Chantal D. Larose.

© 2014 John Wiley & Sons, Inc. Published 2014 by John Wiley & Sons, Inc.

- Appendix: Data Summarization and Visualization (*Continued*)
 - scatter plot, 307
 - skewness, 301
 - statistic, 296
 - statistical inference, 296
 - stem-and-leaf display, 298
 - subject, 294
 - summation notation, 301
 - symmetric, 299
 - unbiased estimator, 302
 - variable, 294
 - Z-score, 303
- Application of clustering using Kohonen networks, 235–236
- Application of *k*-means clustering using SAS Enterprise Miner, 220–223
- Application of neural network modeling, 202–203
- Apriori algorithm, 251
- Apriori property, 251
- Assessing strength of evidence against null hypothesis, 101–102
- Association, 14, 247–261
- Association rules, 14, 247–261
 - affinity analysis, 247–249
 - antecedent, 250
 - apriori algorithm, 251
 - confidence, 250
 - consequent, 250
 - data representation for market basket analysis, 248–249
 - tabular data format, 248–249
 - transactional data format, 248–249
 - definition of, 250
 - extension to general categorical data, 255
 - generalized rule induction (GRI), 256–258
 - itemset, 250
 - itemset frequency, 250
 - itemset, frequent, 251
 - local patterns versus global models, 261
 - market basket analysis, 247–249
 - measuring usefulness of, 259–260
 - procedure for mining, 251
 - supervised or unsupervised learning, 260
 - support, 250
- Association rules, confidence difference method, 259
- Association rules, confidence ratio method, 259
- Association rules, definition of, 250
- Association rules, measuring usefulness of, 259–260
- Attribute, 294
- Average linkage, 212
- Back-propagation, 194–197
- Back-propagation rules, 195
- Back-propagation, example of, 196
- Balancing the training data set, 144–145
- Bank of America, 2
- Bar chart, 55, 297
- Behavior of MSB, MSE, and pseudo-F, 219–220
- Between cluster variation (BCV), 211
- Bias-variance trade-off, 142–144
- Binary trees, 168
- Binning, 38, 72
- Binning based on predictive value, 72–74
- Binning numerical variables, 38
- Bivariate relationship, 304
- Boxplot, 303
- C4.5 algorithm, 174–178
- Candidate splits, 170
- CART, 168–173
- CART optimality measure, 168
- Case, 294
- Categorical variable, 296
- Census, 296
- Chi-square test for goodness of fit, 114
- Choosing *k* for *k*-nearest neighbor, 160
- CIO Magazine, 2
- City block distance, 211
- Class limits, 297
- Class width, 297
- Classification, 10–12, 149–161, 165–182, 187–203
- Classification and regression trees (CART), 168–173
- Classification error, 171
- Cluster centroid, 215
- Cluster membership as input to downstream models, 242–243
- Cluster profiles, 240
- Cluster validity, 235
- Clustered bar chart, 304
- Clustering, 12–14, 209–223, 228–242

- Clustering, hierarchical methods, 209–214
 - agglomerative methods, 209–214
 - average linkage, 212
 - complete linkage, 212
 - dendrogram, 212
 - divisive methods, 212
 - single linkage, 212
- Clustering, *k*-means *see k*-means clustering
- Combination function for *k*-nearest neighbor, 156
- Combination function for neural networks, 192
- Comparison bar chart, 55–56
- Comparison of the CART and C4.5 algorithms, 180–183
- Competition for Kohonen networks, 230
- Competitive learning, 229
- Complete linkage, 212
- Confidence for decision rules, 180
- Confidence for association rules, 250
- Confidence interval estimate, 95, 98
- Confidence intervals for the mean, 94–97
- Confidence intervals for the mean value of *y* given *x*, 125
- Confidence intervals for the proportion, 98–99
- Confidence level, 95
- Confluence of results, 290
- Consequent, 250
- Contingency table, 56, 281, 304
- Continuous variable, 296
- Cooperation for Kohonen networks, 230
- Correlated predictor variables, 77–80
- Correlation, 77, 123, 307
- Correlation coefficient *r*, 307
- Count, 296
- CRISP-DM, cross industry standard process, 4–6
- Cross-validation, 139–141
- Cumulative frequency distribution, 298

- Dangers of extrapolation, 123
- Data Cleaning, *see* Data pre-processing
- Data Mining
 - definition of, xi, 2
 - fallacies of, 6–7
 - need for human direction, 3
 - tasks, 8
- Data pre-processing, 17–41
 - binning numerical variables, 38
 - data cleaning, 17–19
 - data transformation, 27, 28–34
 - decimal scaling, 28
 - flag variables, 36
 - handling missing data, 19–22
 - identifying misclassifications, 22
 - ID fields, 41
 - index field, 39
 - measures of center and spread, 23–26, 301–303
 - min-max normalization, 26
 - need for, 17
 - outliers, graphical methods for identifying, 22–23
 - outliers, numerical methods for identifying, 35
 - reclassifying categorical variables, 39
 - removal of duplicate records, 41
 - removing variables that are not useful, 39
 - transformations to achieve normality, 28–34
 - transforming categorical variables into numeric, 37
 - variables that should not be removed, 40
 - why pre-process data, 27–28
 - Z-score standardization, 27
- Data representation for market basket analysis, 248–249
- Data transformation, *see* Data pre-processing
- Database considerations for *k*-nearest neighbor, 158
- Decimal scaling, 28
- Decision cost/benefit analysis, 285–286
- Decision nodes, 165
- Decision rules, 179–180
- Decision trees, 165–183
 - C4.5 algorithm, 174–179
 - entropy, 174
 - entropy as noise, 175
 - entropy reduction, 174
 - information gain, 175
 - information as signal, 175
 - classification and regression trees (CART), 168–174
 - binary trees, 168
 - candidate splits, 170
 - CART optimality measure, 168
 - classification error, 171
 - tree pruning, 174

- Decision trees (*Continued*)
 - comparison of the CART and C4.5 algorithms, 180–183
 - decision nodes, 165
 - decision rules, 179–180
 - confidence for, 180
 - support for, 180
 - leaf nodes, 165
 - requirements for, 167
- Definition of association rules, 250
- Definition of data mining, xi, 2
- Dendrogram, 212
- Deriving new flag variables, 74–76
- Deriving new numerical variables, 77
- Description, 8
- Description task, model evaluation
 - techniques, 278
- Descriptive statistics, 294
- Deviation, 302
- “Different from” function, 154
- Discrete variable, 295
- Distance function (distance metric), 153–155
 - city block distance, 211
 - Euclidian distance, 153, 210
 - Minkowski distance, 211
- Distribution, 298
- Divisive methods, 212
- Dotplot, 299
- Element, 294
- Empirical rule, 303
- Entropy, 174
- Entropy reduction, 174
- Entropy as noise, 175
- Error rate, overall, 280–283
- Error responsibility in neural networks, 195
- Establishing baseline performance, 145–146
- Estimation, 8–10, 92–99, 118–122, 125–126, 190
 - in neural networks, 190
 - in regression, 118–122, 125–126
 - in univariate statistics, 93–99
- Estimation and prediction using *k*-nearest neighbor, 159–160
- Estimation and prediction using neural networks, 190–191
- Estimation error (prediction error or residual), 121, 278
- Euclidian distance, 153–154, 210
- Example of a Kohonen network study, 231–235
- Example of *k*-means clustering, 216–219
- Exploratory data analysis, 51–81
 - anomalous fields, 71–72
 - binning based on predictive value, 72–74
 - correlated predictor variables, 77–80
 - deriving new flag variables, 74–76
 - deriving new numerical variables, 77
 - exploring categorical variables, 55–62
 - comparison bar chart, 55–56
 - contingency table, 56
 - web graph, 62
 - exploring multivariate relationships, 69–71
 - interaction, 70
 - exploring numerical variables, 62–69
 - getting to know the data set, 52–55
 - selecting interesting subsets of the data, 71
 - versus hypothesis testing, 51–52
- Exploring categorical variables, 55–62
- Exploring multivariate relationships, 69–71
- Exploring numerical variables, 62–69
- Extension to general categorical data, 255
- Extrapolation, 123–125
- Fallacies of data mining, 6–7
- False negatives, 204, 282
- False positives, 204, 282
- Five-number summary, 303
- Flag variables, 36
- Frequency, 296
- Gains charts, 286–289
- Generalized rule induction (GRI), 256–258
- Getting to know the data set, 52–55
- Gradient descent method, 194–195
- Handling missing data, 19–22
- Hidden layer, 191
- Hierarchical clustering, 212–215
- Histogram, 22–23, 298
- Histogram, normalized, 63
- Histogram, overlay, 63
- How confident are we in our estimates, 94
- Hypothesis testing for the mean, 99–101
- Hypothesis testing for the proportion, 104–105

- Hypothesis testing in regression, 122
- Hypothesis testing using confidence intervals, 102–104
- ID fields, 41
- Identifying misclassifications, 22
- Imputation of missing data, 266–273
 - for categorical variables, 272–272
 - for continuous variables, 267–270
 - need for, 266–267
 - patterns in missingness, 272–273
 - standard error of the imputation, 270
- Imputation of missing data for categorical variables, 272
- Imputation of missing data for continuous variables, 267–270
- Index field, 39
- Indicator variables (flag variables, dummy variables), 36–37, 74–77
- Indicator variables for neural networks, 189
- Individual value plot, 305
- Information gain, 174–175
- Information as signal, 175
- Input layer, 191
- Instance-based learning, 150
- Interaction, 70
- Interquartile range (IQR), 35, 303
- Interval data, 295
- Interweaving model evaluation with model building, 289
- IQR method of detecting outliers, 303
- Itemset, 250
- Itemset frequency, 250
- Itemset, frequent, 251
- J*-measure, 257
- k*-Fold cross-validation, 140
- k*-Means clustering, 215–224
 - application of *k*-means clustering using SAS Enterprise Miner, 220–223
 - behavior of MSB, MSE, and pseudo-F, 219–220
 - cluster centroid, 215
 - example of *k*-means clustering, 216–219
 - using cluster membership to make predictions, 223
- k*-Means clustering, application of, using SAS Enterprise Miner, 223–224
- k*-Means clustering, example of, 216–219
- k*-Nearest neighbor algorithm, 149–161
 - choosing *k*, 160
 - combination function, 156–158
 - database considerations, 158
 - distance function (distance metric), 153–155
 - “different from” function, 154
 - Euclidian distance, 154
 - similarity, 153–155
 - triangle inequality, 153
 - estimation and prediction, 159–160
 - instance-based learning, 150
- Kohonen networks, 228–243
 - adaptation in Kohonen networks, 230
 - application of clustering using Kohonen networks, 235–236
 - cluster membership as input to downstream models, 242–243
 - cluster profiles, 240
 - cluster validity, 235
 - competition in Kohonen networks, 230
 - cooperation in Kohonen networks, 230
 - example of a Kohonen network study, 231–235
- Leaf nodes, 165
- Learning rate for neural networks, 195
- Least squares, 119
- Left-skewed, 299
- Levels of measurement, 295
- Lift, 286–289
- Lift charts, 286–289
- Linkage, average, 212
- Linkage, complete, 212
- Linkage, single, 212
- Local patterns versus global models, 261
- Margin of error, 97–98
- Marginal distribution, 304
- Market basket analysis, 247–249
- Mean, 301
- Mean square error (MSE), 117, 279
- Measures of center, 301
- Measures of position, 302
- Measures of variability, 301–302
- Measuring quality of regression model, 123
- Measuring usefulness of association rules, 259–260
- Median, 301
- Methodology for building, 141

- Midrange, 301
- Minimum descriptive length principle, 278
- Minkowski distance, 211
- Min-max normalization, 26
- Mode, 301
- Model evaluation techniques, 278–291
 - confluence of results, 290
 - for the classification task, 280–290
 - contingency table, 281
 - decision cost/benefit analysis, 285–286
 - error rate, overall, 280–283
 - false negatives, 204
 - false positives, 204
 - gains charts, 286–289
 - lift, 286–289
 - lift charts, 286–289
 - type I error, 283
 - type II error, 283
 - for the description task, 278
 - minimum descriptive length principle, 278
 - Occam's razor, 278
 - for the estimation and prediction tasks, 278–280
 - estimation error, 278
 - mean square error (MSE), 279
 - residual, 279
 - standard error of the estimate, 279
 - interweaving model evaluation with model building, 289
 - sensitivity, 283
 - specificity, 283
- Model evaluation techniques for the classification task, 280–290
- Model evaluation techniques for the description task, 278
- Model evaluation techniques for the estimation and prediction tasks, 278–280
- Momentum term, 199–201
- Multicollinearity, 80
- Multiple regression, 126–131
- Multivariate statistics, 110–130
 - analysis of variance (ANOVA), 115–117
 - chi-square test for goodness of fit, 114
 - confidence intervals for the mean value of y given x , 125
 - dangers of extrapolation, 123
 - extrapolation, 123
 - hypothesis testing in regression, 122
 - measuring quality of regression model, 123
 - multiple regression, 126–131
 - prediction intervals for a randomly chosen value of y given x , 125
 - simple linear regression, 118–125
 - correlation, 123
 - estimation error, 121
 - least squares, 119
 - prediction error, 121
 - regression line, 119
 - residual, 121
 - slope, 119
 - y -intercept, 119
 - test for homogeneity of proportions, ‘, 112–114
 - two-sample test for means, 110
 - two-sample test for proportions, 111
 - verifying regression model assumptions, 127–130
- Need for data pre-processing, 17
- Need for human direction, 3
- Need for imputation of missing data, 266–267
- Neural networks, 188–203
 - application of neural network modeling, 202–203
 - back-propagation, 194–197
 - estimation and prediction using neural networks, 190–191
 - gradient descent method, 194
 - learning rate, 195
 - momentum term, 199
 - neurons, 188
 - sensitivity analysis, 201–202
 - sigmoid activation function, 193
 - simple example of a neural network, 191–193
 - termination criteria, 198
- Neurons, 188
- Nominal data, 295
- Numerical variable, 295
- Observation, 295
- Occam's razor, 278
- Ordinal data, 295
- Outliers, graphical methods for identifying, 22–23

- Outliers, numerical methods for identifying, 35
- Overfitting, 141–142
- Parameter, 93, 296
- Pareto chart, 297
- Patterns in missingness, 272–273
- Percentile, 302
- Percentile rank, 302
- Pie chart, 297
- Point estimate, 94
- Point estimation, 94
- Population, 93, 296
- Population mean, 301
- Population standard deviation, 302
- Population variance, 302
- Prediction, 10, 91–105, 110–130, 159–160
- Prediction error (estimation error, residual), 121
- Prediction intervals for a randomly chosen value of y given x , 125
- Prediction task, model evaluation techniques, 278–280
- Predictor variable, 296
- Preparing to model the data, 138–146
 - balancing the training data set, 144–145
 - bias-variance tradeoff, 142–144
 - establishing baseline performance, 145–146
 - cross-validation, 139–141
 - k -fold cross-validation, 140
 - methodology for building and evaluating a data model, 141
 - test data set, 140
 - training data set, 140
 - two-fold cross-validation, 139
 - validating the partition, 140
 - overfitting, 141
 - statistical and data mining methodology, 139
 - supervised versus unsupervised methods, 138–139
 - supervised methods, 139
 - unsupervised methods, 138
- Procedure for mining association rules, 251
- Qualitative variable, 295
- Quantitative variable, 295
- Quartiles, 35, 303
- Random sample, 296
- Range, 302
- Ratio data, 295
- Reclassifying categorical variables, 39
- Record, 295
- Reducing the margin of error, 97–98
- Regression line, 119
- Regression, simple linear, 118–125
- Relative frequency, 296
- Relative frequency distribution, 297, 298
- Removal of duplicate records, 41
- Removing variables that are not useful, 39
- Requirements for decision trees, 167
- Residual (estimation error, prediction error), 121, 279
- Response variable, 296
- Right-skewed, 299
- Sample, 93, 296
- Sample mean, 301
- Sample standard deviation, 302
- Sample variance, 302
- Sampling error, 95
- Scatter plot, 307
- Selecting interesting subsets of the data, 71
- Self-organizing maps (SOMs), 228–230
- Sensitivity, 283
- Sensitivity analysis for neural networks, 201–202
- Sigmoid activation function, 193
- Similarity, 153–155
- Simple example of a neural network, 191–193
- Simple linear regression, 118–125
- Single linkage, 212
- Skewness, 301
- Slope, 119
- Specificity, 283
- Standard deviation, 26, 302
- Standard error of the estimate, s , 123, 279
- Standard error of the imputation, 270
- Statistic, 93, 296
- Statistical and data mining methodology, 139
- Statistical inference, 93–105, 296
- Stem-and-leaf display, 298
- Subject, 294
- Summation notation, 301
- Supervised methods, 139
- Supervised or unsupervised learning, 260

- Supervised versus unsupervised methods, 138–139
- Support, 180, 250
- Support for decision rules, 180
- Symmetric, 299
- Tabular data format, 248–249
- Tasks, data mining, 8–14
 - association, 14, 247–261
 - classification, 10–12, 149–161, 165–182, 187–203
 - clustering, 12–14, 209–223, 228–242
 - description, 8
 - estimation, 8–10, 93–99, 118–122, 125–126, 190
 - prediction, 10, 91–105, 110–130, 159–160
- Termination criteria, 198
- Test data set, 140
- Test for homogeneity of proportions, 112–114
- Training data set, 140
- Transactional data format, 248–249
- Transformations to achieve normality, 28–34
- Transforming categorical variables into numeric, 37
- Tree pruning, 174
- Triangle inequality, 153
- Two-fold cross-validation, 139
- Two-sample test for means, 110
- Two-sample test for proportions, 111
- Type I error, 283
- Type II error, 283
- Unbiased estimator, 302
- Underfitting, 141–142
- Univariate Statistics, 91–105
 - assessing strength of evidence against null hypothesis, 101–102
 - confidence intervals, 94–99
 - confidence level, 95
 - margin of error, 97–98
 - for the mean, 94–97
 - for the proportion, 98–99
 - how confident are we in our estimates, 94
 - hypothesis testing for the mean, 99–101
 - hypothesis testing for the proportion, 104–105
 - hypothesis testing using confidence intervals, 102–104
 - reducing the margin of error, 97–98
 - statistical inference, 93–105
 - estimation, 92–98
 - parameter, 93
 - point estimate, 94
 - point estimation, 94
 - population, 93
 - sample, 93
 - statistic, 93
 - sampling error, 95
- Unsupervised methods, 138
- Using cluster membership to make predictions, 223
- Validating the partition, 140
- Variable, 294
- Variables that should not be removed, 40
- Verifying regression model assumptions, 127–130
- Voting, simple unweighted, 156
- Voting, weighted, 156
- Web graph, 62
- Why pre-process data, 27–28
- y-Intercept, 119
- Z-score, 303
- Z-score standardization, 27