A General Introduction to Data Analytics

A General Introduction to Data Analytics

João Mendes Moreira
University of Porto

André C. P. L. F. de Carvalho
University of São Paulo

Tomáš Horváth
Eötvös Loránd University in Budapest
Pavol Jozef Šafárik University in Košice

WILEY

This edition first published 2019
© 2019 John Wiley & Sons, Inc.

All rights reserved. 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 or otherwise, except as permitted by law. Advice on how to obtain permission to reuse material from this title is available at http://www.wiley.com/go/permissions.

The right of João Moreira, André de Carvalho, and Tomáš Horváth to be identified as the author(s) of this work has been asserted in accordance with law.

Registered Office
John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, USA

Editorial Office
111 River Street, Hoboken, NJ 07030, USA

For details of our global editorial offices, customer services, and more information about Wiley products visit us at www.wiley.com.

Wiley also publishes its books in a variety of electronic formats and by print-on-demand. Some content that appears in standard print versions of this book may not be available in other formats.

Limit of Liability/Disclaimer of Warranty
In view of ongoing research, equipment modifications, changes in governmental regulations, and the constant flow of information relating to the use of experimental reagents, equipment, and devices, the reader is urged to review and evaluate the information provided in the package insert or instructions for each chemical, piece of equipment, reagent, or device for, among other things, any changes in the instructions or indication of usage and for added warnings and precautions. While the publisher and authors have used their best efforts in preparing this work, they make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives, written sales materials or promotional statements for this work. The fact that an organization, website, or product is referred to in this work as a citation and/or potential source of further information does not mean that the publisher and authors endorse the information or services the organization, website, or product may provide or recommendations it may make. This work is sold with the understanding that the publisher is not engaged in rendering professional services. The advice and strategies contained herein may not be suitable for your situation. You should consult with a specialist where appropriate. Further, readers should be aware that websites listed in this work may have changed or disappeared between when this work was written and when it is read. Neither the publisher nor authors shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

Library of Congress Cataloging-in-Publication Data

Names: Moreira, João, 1969– author. | Carvalho, André Carlos Ponce de Leon Ferreira, author. | Horváth, Tomáš, 1976– author.
Title: A general introduction to data analytics / by João Mendes Moreira, André C. P. L. F. de Carvalho, Tomáš Horváth.
Description: Hoboken, NJ : John Wiley & Sons, 2019. | Includes bibliographical references and index. |
Identifiers: LCCN 2017060728 (print) | LCCN 2018005929 (ebook) | ISBN 9781119296256 (pdf) | ISBN 9781119296263 (epub) | ISBN 9781119296249 (cloth)
Subjects: LCSH: Mathematical statistics–Methodology. | Electronic data processing. | Data mining.
Classification: LCC QA276.4 (ebook) | LCC QA276.4 .M664 2018 (print) | DDC 519.50285–dc23
LC record available at https://lccn.loc.gov/2017060728

Cover image: © agsandrew/Shutterstock
Cover design by Wiley

Printed in the United States of America.

Set in 10/12pt Warnock by SPi Global, Pondicherry, India

10 9 8 7 6 5 4 3 2 1

To the women at home that make my life better: Mamã, Yá and Yé – João
To my family, Valeria, Beatriz, Gabriela and Mariana – André
To my wife Danielle – Tomáš

Contents

Preface *xiii*
Acknowledgments *xv*
Presentational Conventions *xvii*
About the Companion Website *xix*

Part I Introductory Background *1*

1 What Can We Do With Data? *3*
1.1 Big Data and Data Science *4*
1.2 Big Data Architectures *5*
1.3 Small Data *6*
1.4 What is Data? *7*
1.5 A Short Taxonomy of Data Analytics *9*
1.6 Examples of Data Use *10*
1.6.1 Breast Cancer in Wisconsin *11*
1.6.2 Polish Company Insolvency Data *11*
1.7 A Project on Data Analytics *12*
1.7.1 A Little History on Methodologies for Data Analytics *12*
1.7.2 The KDD Process *14*
1.7.3 The CRISP-DM Methodology *15*
1.8 How this Book is Organized *16*
1.9 Who Should Read this Book *18*

Part II Getting Insights from Data *19*

2 Descriptive Statistics *21*
2.1 Scale Types *22*
2.2 Descriptive Univariate Analysis *25*
2.2.1 Univariate Frequencies *25*

2.2.2 Univariate Data Visualization *27*
2.2.3 Univariate Statistics *32*
2.2.4 Common Univariate Probability Distributions *38*
2.3 Descriptive Bivariate Analysis *40*
2.3.1 Two Quantitative Attributes *41*
2.3.2 Two Qualitative Attributes, at Least one of them Nominal *45*
2.3.3 Two Ordinal Attributes *46*
2.4 Final Remarks *47*
2.5 Exercises *47*

3 Descriptive Multivariate Analysis *49*
3.1 Multivariate Frequencies *49*
3.2 Multivariate Data Visualization *50*
3.3 Multivariate Statistics *59*
3.3.1 Location Multivariate Statistics *59*
3.3.2 Dispersion Multivariate Statistics *60*
3.4 Infographics and Word Clouds *66*
3.4.1 Infographics *66*
3.4.2 Word Clouds *67*
3.5 Final Remarks *67*
3.6 Exercises *68*

4 Data Quality and Preprocessing *71*
4.1 Data Quality *71*
4.1.1 Missing Values *72*
4.1.2 Redundant Data *74*
4.1.3 Inconsistent Data *75*
4.1.4 Noisy Data *76*
4.1.5 Outliers *77*
4.2 Converting to a Different Scale Type *77*
4.2.1 Converting Nominal to Relative *78*
4.2.2 Converting Ordinal to Relative or Absolute *81*
4.2.3 Converting Relative or Absolute to Ordinal or Nominal *82*
4.3 Converting to a Different Scale *83*
4.4 Data Transformation *85*
4.5 Dimensionality Reduction *86*
4.5.1 Attribute Aggregation *88*
4.5.1.1 Principal Component Analysis *88*
4.5.1.2 Independent Component Analysis *91*
4.5.1.3 Multidimensional Scaling *91*
4.5.2 Attribute Selection *92*
4.5.2.1 Filters *92*
4.5.2.2 Wrappers *93*
4.5.2.3 Embedded *94*

4.5.2.4 Search Strategies *95*
4.6 Final Remarks *96*
4.7 Exercises *96*

5 Clustering *99*
5.1 Distance Measures *100*
5.1.1 Differences between Values of Common Attribute Types *101*
5.1.2 Distance Measures for Objects with Quantitative Attributes *103*
5.1.3 Distance Measures for Non-conventional Attributes *104*
5.2 Clustering Validation *107*
5.3 Clustering Techniques *108*
5.3.1 K-means *110*
5.3.1.1 Centroids and Distance Measures *110*
5.3.1.2 How K-means Works *111*
5.3.2 DBSCAN *115*
5.3.3 Agglomerative Hierarchical Clustering Technique *117*
5.3.3.1 Linkage Criterion *119*
5.3.3.2 Dendrograms *120*
5.4 Final Remarks *122*
5.5 Exercises *123*

6 Frequent Pattern Mining *125*
6.1 Frequent Itemsets *127*
6.1.1 Setting the *min_sup* Threshold *128*
6.1.2 Apriori – a Join-based Method *131*
6.1.3 Eclat *133*
6.1.4 FP-Growth *134*
6.1.5 Maximal and Closed Frequent Itemsets *138*
6.2 Association Rules *139*
6.3 Behind Support and Confidence *142*
6.3.1 Cross-support Patterns *143*
6.3.2 Lift *144*
6.3.3 Simpson's Paradox *145*
6.4 Other Types of Pattern *147*
6.4.1 Sequential patterns *147*
6.4.2 Frequent Sequence Mining *148*
6.4.3 Closed and Maximal Sequences *148*
6.5 Final Remarks *149*
6.6 Exercises *149*

7 Cheat Sheet and Project on Descriptive Analytics *151*
7.1 Cheat Sheet of Descriptive Analytics *151*
7.1.1 On Data Summarization *151*

7.1.2 On Clustering *151*
7.1.3 On Frequent Pattern Mining *153*
7.2 Project on Descriptive Analytics *154*
7.2.1 Business Understanding *154*
7.2.2 Data Understanding *155*
7.2.3 Data Preparation *155*
7.2.4 Modeling *157*
7.2.5 Evaluation *158*
7.2.6 Deployment *158*

Part III Predicting the Unknown *159*

8 Regression *161*
8.1 Predictive Performance Estimation *164*
8.1.1 Generalization *164*
8.1.2 Model Validation *165*
8.1.3 Predictive Performance Measures for Regression *169*
8.2 Finding the Parameters of the Model *171*
8.2.1 Linear Regression *171*
8.2.1.1 Empirical Error *173*
8.2.2 The Bias-variance Trade-off *175*
8.2.3 Shrinkage Methods *177*
8.2.3.1 Ridge Regression *179*
8.2.3.2 Lasso Regression *180*
8.2.4 Methods that use Linear Combinations of Attributes *181*
8.2.4.1 Principal Components Regression *181*
8.2.4.2 Partial Least Squares Regression *182*
8.3 Technique and Model Selection *182*
8.4 Final Remarks *183*
8.5 Exercises *184*

9 Classification *187*
9.1 Binary Classification *188*
9.2 Predictive Performance Measures for Classification *192*
9.3 Distance-based Learning Algorithms *199*
9.3.1 K-nearest Neighbor Algorithms *199*
9.3.2 Case-based Reasoning *202*
9.4 Probabilistic Classification Algorithms *203*
9.4.1 Logistic Regression Algorithm *205*
9.4.2 Naive Bayes Algorithm *207*
9.5 Final Remarks *208*
9.6 Exercises *210*

10 Additional Predictive Methods *211*
10.1 Search-based Algorithms *211*
10.1.1 Decision Tree Induction Algorithms *212*
10.1.2 Decision Trees for Regression *217*
10.1.2.1 Model Trees *218*
10.1.2.2 Multivariate Adaptive Regression Splines *219*
10.2 Optimization-based Algorithms *221*
10.2.1 Artificial Neural Networks *222*
10.2.1.1 Backpropagation *224*
10.2.1.2 Deep Networks and Deep Learning Algorithms *230*
10.2.2 Support Vector Machines *233*
10.2.2.1 SVM for Regression *237*
10.3 Final Remarks *238*
10.4 Exercises *239*

11 Advanced Predictive Topics *241*
11.1 Ensemble Learning *241*
11.1.1 Bagging *243*
11.1.2 Random Forests *244*
11.1.3 AdaBoost *245*
11.2 Algorithm Bias *246*
11.3 Non-binary Classification Tasks *248*
11.3.1 One-class Classification *248*
11.3.2 Multi-class Classification *249*
11.3.3 Ranking Classification *250*
11.3.4 Multi-label Classification *251*
11.3.5 Hierarchical Classification *252*
11.4 Advanced Data Preparation Techniques for Prediction *253*
11.4.1 Imbalanced Data Classification *253*
11.4.2 For Incomplete Target Labeling *254*
11.4.2.1 Semi-supervised Learning *254*
11.4.2.2 Active Learning *255*
11.5 Description and Prediction with Supervised Interpretable Techniques *255*
11.6 Exercises *256*

12 Cheat Sheet and Project on Predictive Analytics *259*
12.1 Cheat Sheet on Predictive Analytics *259*
12.2 Project on Predictive Analytics *259*
12.2.1 Business Understanding *260*
12.2.2 Data Understanding *260*
12.2.3 Data Preparation *265*
12.2.4 Modeling *265*
12.2.5 Evaluation *265*
12.2.6 Deployment *266*

Part IV Popular Data Analytics Applications *267*

13 Applications for Text, Web and Social Media *269*
13.1 Working with Texts *269*
13.1.1 Data Acquisition *271*
13.1.2 Feature Extraction *271*
13.1.2.1 Tokenization *272*
13.1.2.2 Stemming *272*
13.1.2.3 Conversion to Structured Data *275*
13.1.2.4 Is the Bag of Words Enough? *276*
13.1.3 Remaining Phases *277*
13.1.4 Trends *277*
13.1.4.1 Sentiment Analysis *278*
13.1.4.2 Web Mining *278*
13.2 Recommender Systems *278*
13.2.1 Feedback *279*
13.2.2 Recommendation Tasks *280*
13.2.3 Recommendation Techniques *281*
13.2.3.1 Knowledge-based Techniques *281*
13.2.3.2 Content-based Techniques *282*
13.2.3.3 Collaborative Filtering Techniques *282*
13.2.4 Final Remarks *289*
13.3 Social Network Analysis *291*
13.3.1 Representing Social Networks *291*
13.3.2 Basic Properties of Nodes *294*
13.3.2.1 Degree *294*
13.3.2.2 Distance *294*
13.3.2.3 Closeness *295*
13.3.2.4 Betweenness *296*
13.3.2.5 Clustering Coefficient *297*
13.3.3 Basic and Structural Properties of Networks *297*
13.3.3.1 Diameter *297*
13.3.3.2 Centralization *297*
13.3.3.3 Cliques *299*
13.3.3.4 Clustering Coefficient *299*
13.3.3.5 Modularity *299*
13.3.4 Trends and Final Remarks *299*
13.4 Exercises *300*

Apendix A: Comprehensive Description of the CRISP-DM Methodology *303*
References *311*
Index *315*

Preface

We are living in a period of history that will certainly be remembered as one where information began to be instantaneously obtainable, services were tailored to individual criteria, and people did what made them feel good (if it did not put their lives at risk). Every year, machines are able to do more and more things that improve our quality of life. More data is available than ever before, and will become even more so. This is a time when we can extract more information from data than ever before, and benefit more from it.

In different areas of business and in different institutions, new ways to collect data are continuously being created. Old documents are being digitized, new sensors count the number of cars passing along motorways and extract useful information from them, our smartphones are informing us where we are at each moment and what new opportunities are available, and our favorite social networks register to whom we are related or what things we like.

Whatever area we work in, new data is available: data on how students evaluate professors, data on the evolution of diseases and the best treatment options per patient, data on soil, humidity levels and the weather, enabling us to produce more food with better quality, data on the macro economy, our investments and stock market indicators over time, enabling fairer distribution of wealth, data on things we purchase, allowing us to purchase more effectively and at lower cost.

Students in many different domains feel the need to take advantage of the data they have. New courses on data analytics have been proposed in many different programs, from biology to information science, from engineering to economics, from social sciences to agronomy, all over the world.

The first books on data analytics that appeared some years ago were written by data scientists for other data scientists or for data science students. The majority of the people interested in these subjects were computing and statistics students. The books on data analytics were written mainly for them. Nowadays, more and more people are interested in learning data analytics. Students of economics, management, biology, medicine, sociology, engineering, and some other subjects are willing to learn about data analytics. This book

intends not only to provide a new, more friendly textbook for computing and statistics students, but also to open data analytics to those students who may know nothing about computing or statistics, but want to learn these subjects in a simple way. Those who have already studied subjects such as statistics will recognize some of the content described in this book, such as descriptive statistics. Students from computing will be familiar with a pseudocode.

After reading this book, it is not expected that you will feel like a data scientist with ability to create new methods, but it is expected that you might feel like a data analytics practitioner, able to drive a data analytics project, using the right methods to solve real problems.

João Mendes Moreira
University of Porto, Porto, Portugal

André C. P. L. F. de Carvalho
University of São Paulo, São Carlos, Brazil

Tomáš Horváth
Eötvös Loránd University in Budapest
Pavol Jozef Šafárik University in Košice
October, 2017

Acknowledgments

The authors would like to thank Bruno Almeida Pimentel, Edésio Alcobaça Neto, Everlândio Fernandes, Victor Alexandre Padilha and Victor Hugo Barella for their useful comments.

Over the last several months, we have been in contact with several people from Wiley: Jon Gurstelle, Executive Editor on Statistics; Kathleen Pagliaro, Assistant Editor; Samantha Katherine Clarke and Kshitija Iyer, Project Editors; and Katrina Maceda, Production Editor. To all these wonderful people, we owe a deep sense of gratitude, especially now this project has been completed.

Lastly, we would like to thank our families for their constant love, support, patience, and encouragement.

J. A. T.

Presentational Conventions

Definition The definitions are presented in the format shown here.

Special sections and formats Whenever a method is described, three different sections are presented:

- Assessing and evaluating results: how can we assess the results of a method? How to interpret them? This section is all about answering these questions.
- Setting the hyper-parameters: each method has its own hyper-parameters that must be set. This section explains how to set them.
- Advantages and disadvantages: a table summarizes the positive and negative characteristics of a given method.

About the Companion Website

This book is accompanied by a companion website:

www.wiley.com/go/moreira/dataanalytics

The website includes:
- Presentation slides for instructors

Part I

Introductory Background

1

What Can We Do With Data?

Until recently, researchers working with data analysis were struggling to obtain data for their experiments. Recent advances in the technology of data processing, data storage and data transmission, associated with advanced and intelligent computer software, reducing costs and increasing capacity, have changed this scenario. It is the time of the Internet of Things, where the aim is to have everything or almost everything connected. Data previously produced on paper are now on-line. Each day, a larger quantity of data is generated and consumed. Whenever you place a comment in your social network, upload a photograph, some music or a video, navigate through the Internet, or add a comment to an e-commerce web site, you are contributing to the data increase. Additionally, machines, financial transactions and sensors such as security cameras, are increasingly gathering data from very diverse and widespread sources.

In 2012, it was estimated that, each year, the amount of data available in the world doubles [1]. Another estimate, from 2014, predicted that by 2020 all information will be digitized, eliminated or reinvented in 80% of processes and products of the previous decade [2]. In a third report, from 2015, it was predicted that mobile data traffic will be almost 10 times larger in 2020 [3]. The result of all these rapid increases of data is named by some the "data explosion".

Despite the impression that this can give – that we are drowning in data – there are several benefits from having access to all these data. These data provide a rich source of information that can be transformed into new, useful, valid and human-understandable knowledge. Thus, there is a growing interest in exploring these data to extract this knowledge, using it to support decision making in a wide variety of fields: agriculture, commerce, education, environment, finance, government, industry, medicine, transport and social care. Several companies around the world are realizing the gold mine they have and the potential of these data to support their work, reduce waste and dangerous and tedious work activities, and increase the value of their products and their profits.

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

The analysis of these data to extract such knowledge is the subject of a vibrant area known as data analytics, or simply "analytics". You can find several definitions of analytics in the literature. The definition adopted here is:

Analytics The science that analyze crude data to extract useful knowledge (patterns) from them.

This process can also include data collection, organization, pre-processing, transformation, modeling and interpretation.

Analytics as a knowledge area involves input from many different areas. The idea of generalizing knowledge from a data sample comes from a branch of statistics known as inductive learning, an area of research with a long history. With the advances of personal computers, the use of computational resources to solve problems of inductive learning become more and more popular. Computational capacity has been used to develop new methods. At the same time, new problems have appeared requiring a good knowledge of computer sciences. For instance, the ability to perform a given task with more computational efficiency has become a subject of study for people working in computational statistics.

In parallel, several researchers have dreamed of being able to reproduce human behavior using computers. These were people from the area of artificial intelligence. They also used statistics for their research but the idea of reproducing human and biological behavior in computers was an important source of motivation. For instance, reproducing how the human brain works with artificial neural networks has been studied since the 1940s; reproducing how ants work with ant colony optimization algorithm since the 1990s. The term machine learning (ML) appeared in this context as the "field of study that gives computers the ability to learn without being explicitly programmed," according to Arthur Samuel in 1959 [4].

In the 1990s, a new term appeared with a different slight meaning: data mining (DM). The 1990s was the decade of the appearance of business intelligence tools as consequence of the data facilities having larger and cheaper capacity. Companies start to collect more and more data, aiming to either solve or improve business operations, for example by detecting frauds with credit cards, by advising the public of road network constraints in cities, or by improving relations with clients using more efficient techniques of relational marketing. The question was of being able to mine the data in order to extract the knowledge necessary for a given task. This is the goal of data mining.

1.1 Big Data and Data Science

In the first years of the 20th century, the term big data has appeared. Big data, a technology for data processing, was initially defined by the "three Vs", although some more Vs have been proposed since. The first three Vs allow us to define

a taxonomy of big data. They are: volume, variety and velocity. Volume is concerned with how to store big data: data repositories for large amounts of data. Variety is concerned with how to put together data from different sources. Velocity concerns the ability to deal with data arriving very fast, in streams known as data streams. Analytics is also about discovering knowledge from data streams, going beyond the velocity component of big data.

Another term that has appeared and is sometimes used as a synonym for big data is data science. According to Provost and Fawcett [5], big data are data sets that are too large to be managed by conventional data-processing technologies, requiring the development of new techniques and tools for data storage, processing and transmission. These tools include, for example, MapReduce, Hadoop, Spark and Storm. But data volume is not the only characterization of big data. The word "big" can refer to the number of data sources, to the importance of the data, to the need for new processing techniques, to how fast data arrive, to the combination of different sets of data so they can be analyzed in real time, and its ubiquity, since any company, nonprofit organization or individual has access to data now.

Thus big data is more concerned with technology. It provides a computing environment, not only for analytics, but also for other data processing tasks. These tasks include finance transaction processing, web data processing and georeferenced data processing.

Data science is concerned with the creation of models able to extract patterns from complex data and the use of these models in real-life problems. Data science extracts meaningful and useful knowledge from data, with the support of suitable technologies. It has a close relationship to analytics and data mining. Data science goes beyond data mining by providing a knowledge extraction framework, including statistics and visualization.

Therefore, while big data gives support to data collection and management, data science applies techniques to these data to discover new and useful knowledge: big data collects and data science discovers. Other terms such as knowledge discovery or extraction, pattern recognition, data analysis, data engineering, and several others are also used. The definition we use of data analytics covers all these areas that are used to extract knowledge from data.

1.2 Big Data Architectures

As data increase in size, velocity and variety, new computer technologies become necessary. These new technologies, which include hardware and software, must be easily expanded as more data are processed. This property is known as scalability. One way to obtain scalability is by distributing the data processing tasks into several computers, which can be combined into clusters of computers. The reader should not confuse clusters of computers

with clusters produced by clustering techniques, which are techniques from analytics in which a data set is partitioned to find groups within it.

Even if processing power is expanded by combining several computers in a cluster, creating a distributed system, conventional software for distributed systems usually cannot cope with big data. One of the limitations is the efficient distribution of data among the different processing and storage units. To deal with these requirements, new software tools and techniques have been developed.

One of the first techniques developed for big data processing using clusters was MapReduce. MapReduce is a programming model that has two steps: map and reduce. The most famous implementation of MapReduce is called Hadoop.

MapReduce divides the data set into parts – chunks – and stores in the memory of each cluster computer the chunk of the data set needed by this computer to accomplish its processing task. As an example, suppose that you need to calculate the average salary of 1 billion people and you have a cluster with 1000 computers, each with a processing unit and a storage memory. The people can be divided into 1000 chunks – subsets – with data from 1 million people each. Each chunk can be processed independently by one of the computers. The results produced by each these computers (the average salary of 1 million people) can be averaged, returning the final salary average.

To efficiently solve a big data problem, a distributed system must attend the following requirements:

- Make sure that no chunk of data is lost and the whole task is concluded. If one or more computers has a failure, their tasks, and the corresponding data chunk, must be assumed by another computer in the cluster.
- Repeat the same task, and corresponding data chunk, in more than one cluster computer; this is called redundancy. Thus, if one or more computer fails, the redundant computer carries on with the task.
- Computers that have had faults can return to the cluster again when they are fixed.
- Computers can be easily removed from the cluster or extra ones included in it as the processing demand changes.

A solution incorporating these requirements must hide from the data analyst the details of how the software works, such as how the data chunks and tasks are divided among the cluster computers.

1.3 Small Data

In the opposite direction from big data technologies and methods, there is a movement towards more personal, subjective analysis of chunks of data, termed "small data". Small data is a data set whose volume and format allows its processing and analysis by a person or a small organization. Thus, instead of

collecting data from several sources, with different formats, and generated at increasing velocities, creating large data repositories and processing facilities, small data favors the partition of a problem into small packages, which can be analyzed by different people or small groups in a distributed and integrated way.

People are continuously producing small data as they perform their daily activities, be it navigating the web, buying a product in a shop, undergoing medical examinations and using apps in their mobiles. When these data are collected to be stored and processed in large data servers they become big data. To be characterized as small data, a data set must have a size that allows its full understanding by an user.

The type of knowledge sought in big and small data is also different, with the first looking for correlations and the second for causality relations. While big data provide tools that allow companies to understand their customers, small data tools try to help customers to understand themselves. Thus, big data is concerned with customers, products and services, and small data is concerned with the individuals that produced the data.

1.4 What is Data?

But what is data about? Data, in the information age, are a large set of bits encoding numbers, texts, images, sounds, videos, and so on. Unless we add information to data, they are meaningless. When we add information, giving a meaning to them, these data become knowledge. But before data become knowledge, typically, they pass through several steps where they are still referred to as data, despite being a bit more organized; that is, they have *some* information associated with them.

Let us see the example of data collected from a private list of acquaintances or contacts.

Information as presented in Table 1.1, usually referred to as tabular data, is characterized by the way data are organized. In tabular data, data are organized in rows and columns, where each column represents a characteristic of the data and each row represents an occurrence of the data. A column is referred to as an attribute or, with the same meaning, a feature, while a row is referred to as an instance, or with the same meaning, an object.

Instance or Object Examples of the concept we want to characterize.

Example 1.1 In the example in Table 1.1, we intend to characterize people in our private contact list. Each member is, in this case, an instance or object. It corresponds to a row of the table.

Attribute or Feature Attributes, also called features, are characteristics of the instances.

Table 1.1 Data set of our private contact list.

Contact	Age	Educational level	Company
Andrew	55	1.0	Good
Bernhard	43	2.0	Good
Carolina	37	5.0	Bad
Dennis	82	3.0	Good
Eve	23	3.2	Bad
Fred	46	5.0	Good
Gwyneth	38	4.2	Bad
Hayden	50	4.0	Bad
Irene	29	4.5	Bad
James	42	4.1	Good
Kevin	35	4.5	Bad
Lea	38	2.5	Good
Marcus	31	4.8	Bad
Nigel	71	2.3	Good

Example 1.2 In Table 1.1, contact, age, education level and company are four different attributes.

The majority of the chapters in this book expect the data to be in tabular format; that is, already organized by rows and columns, each row representing an instance and each column representing an attribute. However, a table can be organized differently, having the instances per column and the attributes per row.

There are, however, data that are not possible to represent in a single table.

Example 1.3 As an example, if some of the contacts are relatives of other contacts, a second table, as shown in Table 1.2, representing the family relationships, would be necessary. You should note that each person referred to in Table 1.2 also exists in Table 1.1, i.e., there are relations between attributes of different tables.

Data sets represented by several tables, making clear the relations between these tables, are called relational data sets. This information is easily handled using relational databases. In this book, only simple forms of relational data will be used. This is discussed in each chapter whenever necessary.

Table 1.2 Family relations between contacts.

Friend	Father	Mother	Sister
Eve	Andrew	Hayden	Irene
Irene	Andrew	Hayden	Eve

Example 1.4 In our example, data is split into two tables, one with the individual data of each contact (Table 1.1) and the other with the data about the family relations between them (Table 1.2).

1.5 A Short Taxonomy of Data Analytics

Now that we know what data are, we will look at what we can do with them. A natural taxonomy that exists in data analytics is:

- Descriptive analytics: summarize or condense data to extract patterns
- Predictive analytics: extract models from data to be used for future predictions.

In descriptive analytics tasks, the result of a given method or technique,[1] is obtained directly by applying an algorithm to the data. The result can be a statistic, such as an average, a plot, or a set of groups with similar instances, among other things, as we will see in this book. Let us see the definition of method and algorithm.

Method or technique A method or technique is a systematic procedure that allows us to achieve an intended goal.

A method shows how to perform a given task. But in order to use a language closer to the language computers can understand, it is necessary to describe the method/technique through an algorithm.

Algorithm An algorithm is a self-contained, step-by-step set of instructions easily understandable by humans, allowing the implementation of a given method. They are self-contained in order to be easily translated to an arbitrary programming language.

Example 1.5 The method to obtain the average age of my contacts uses the ages of each (we could use other methods, such as using the number of contacts for each different age). A possible algorithm for this very simple example is shown next.

1 These two terms are used interchangeably in this book.

Algorithm An algorithm to calculate the average age of our contacts

1: INPUT: A: a vector of size N with the ages of all contacts.
2: $S \leftarrow 0$ ▷ Initialize the sum S to zero
3: **for** $i = 1$ **to** N **do** ▷ Iterate through all the elements of A.
4: $S \leftarrow S + A_i$ ▷ Add the current (ith) element of A to S.
5: $\overline{A} \leftarrow S/N$ ▷ Divide the sum by the number N of contacts.
6: return($\overline{A}$) ▷ Return the result, i.e. the average age of the N contacts.

In the limit, a method can be straightforward. It is possible, in many cases, to express it as a formula instead of as an algorithm.

Example 1.6 For instance, the average could be expressed as: $\overline{A} = \sum_{i=1}^{N} A_i/N$.

We have seen an algorithm that describes a descriptive method. An algorithm can also describe predictive methods. In this last case it describes how to generate a model. Let us see what a model is.

Model A model in data analytics is a generalization obtained from data that can be used afterwords to generate predictions for new given instances. It can be seen as a prototype that can be used to make predictions. Thus, model induction is a predictive task.

Example 1.7 If we apply an algorithm for induction of decision trees to provide an explanation of who, among our contacts, is a good company, we obtain a model, called a decision tree, like the one presented in Figure 1.1. It can be seen that people older than 38 years are typically better company than those whose age is equal or less than 38 more than 80% of people aged 38 or less are bad company, while more than 80% of people older than 38 are good company. This model could be used to predict whether a new contact is or not a good company. It would be enough to know the age of that new contact.

Now that we have a rough idea of what analytics is, let us see real examples of problems in data analytics.

1.6 Examples of Data Use

We will describe two real-world problems from different areas as an introduction to the different subjects that are covered in this book. Many more could be presented. One of the problems is from medicine and the other is

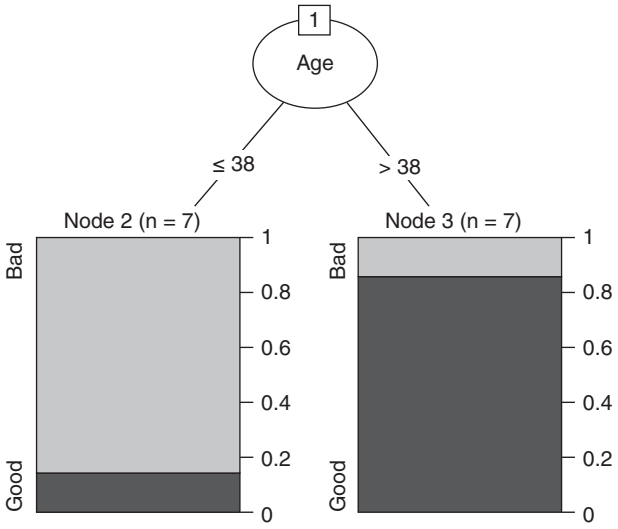


Figure 1.1 A prediction model to classify someone as either good or bad company.

from economics. The problems were chosen with a view to the availability of relevant data, because the problems involved will be solved in the project chapters of the book (Chapters 7 and 12).

1.6.1 Breast Cancer in Wisconsin

Breast cancer is a well-known problem that affects mainly women. The detection of breast tumors can be performed through a biopsy technique known as fine-needle aspiration. This uses a fine needle to sample cells from the mass under study. Samples of breast mass obtained using fine-needle aspiration were recorded in a set of images [6]. Then, a dataset was collected by extracting features from these images. The objective of the first problem is to detect different patterns of breast tumors in this dataset, to enable it to be used for diagnostic purposes.

1.6.2 Polish Company Insolvency Data

The second problem concerns the prediction of the economic wealth of Polish companies. Can we predict which companies will become insolvent in the next five years? The answer to this question is obviously relevant to institutions and shareholders.

1.7 A Project on Data Analytics

Every project needs a plan. Or, to be precise, a methodology to prepare the plan. A project on data analytics does not imply only the use of one or more specific methods. It implies:

- understanding the problem to be solved
- defining the objectives of the project
- looking for the necessary data
- preparing these data so that they can be used
- identifying suitable methods and choosing between them
- tuning the hyper-parameters of each method (see below)
- analyzing and evaluating the results
- redoing the pre-processing tasks and repeating the experiments
- and so on.

In this book, we assume that in the induction of a model, there are both hyper-parameters and parameters whose values are set. The values of the hyper-parameters are set by the user, or some external optimization method. The parameter values, on the other hand, are model parameters whose values are set by a modeling or learning algorithm in its internal procedure. When the distinction is not clear, we use the term parameter. Thus, hyper-parameters might be, for example, the number of layers and the activation function in a multi-layer perceptron neural network and the number of clusters for the k-means algorithm. Examples of parameters are the weights found by the backpropagation algorithm when training a multi-layer perceptron neural network and the distribution of objects carried out by k-means. Multi-layer perceptron neural networks and k-means will be explained later in this book.

How can we perform all these operations in an organized way? This section is all about methodologies for planning and developing projects in data analytics.

A brief history of methodologies for data analytics is presented first. Afterwards, two different methodologies are described:

- a methodology from Academia, KDD
- a methodology from industry, CRISP-DM.

The latter is used in the cheat sheet and project chapters (Chapters 7 and 12).

1.7.1 A Little History on Methodologies for Data Analytics

Machine learning, knowledge discovery from data and related areas experienced strong development in the 1990s. Both in academia and industry, the research on these topics was advancing quickly. Naturally, methodologies for projects in these areas, now referred to as data analytics, become a necessity.

In the mid-1990s, both in academia and industry, different methodologies were presented.

The most successful methodology from academia came from the USA. This was the KDD process of Usama Fayyad, Gregory Piatetsky-Shapiro and Padhraic Smyth [7]. Despite being from academia, the authors had considerable work experience in industry.

The most successful tool from industry, was and still is the CRoss-Industry Standard Process for Data Mining (CRISP-DM) [8]. Conceived in 1996, it later got underway as an European Union project under the ESPRIT funding initiative. The project had five partners from industry: SPSS, Teradata, Daimler AG, NCR Corporation and OHRA, an insurance company. In 1999 the first version was presented. An attempt to create a new version began between 2006 and 2008 but no new discoveries are known from these efforts. CRISP-DM is nowadays used by many different practitioners and by several corporations, in particular IBM. However, despite its popularity, CRISP-DM needs new developments in order to meet the new challenges from the age of big data.

Other methodologies exist. Some of them are domain-specific: they assume the use of a given tool for data analytics. This is not the case for SEMMA, which, despite has been created by SAS, is tool independent. Each letter of its name, SEMMA, refers to one of its five steps: Sample, Explore, Modify, Model and Assess.

Polls performed by kdnuggets [9] over the years (2002, 2004, 2007 and 2014) show how methodologies on data analytics have been used through time (Figure 1.2).

Next, the KDD process and the CRISP-DM methodologies are described in detail.

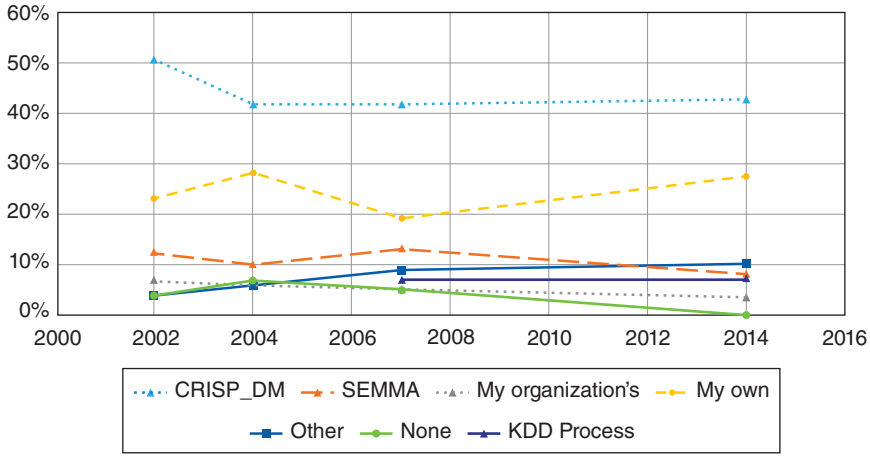


Figure 1.2 The use of different methodologies on data analytics through time.

1.7.2 The KDD Process

Intended to be a methodology that could cope with all the processes necessary to extract knowledge from data, the KDD process proposes a sequence of nine steps. In spite of the sequence, the KDD process considers the possibility of going back to any previous step in order to redo some part of the process. The nine steps are:

1) *Learning the application domain:* What is expected in terms of the application domain? What are the characteristics of the problem; its specificities? A good understanding of the application domain is required.
2) *Creating a target dataset:* What data are needed for the problem? Which attributes? How will they be collected and put in the desired format (say, a tabular data set)? Once the application domain is known, the data analyst team should be able to identify the data necessary to accomplish the project.
3) *Data cleaning and pre-processing:* How should missing values and/or outliers such as extreme values be handled? What data type should we choose for each attribute? It is necessary to put the data in a specific format, such as a tabular format.
4) *Data reduction and projection:* Which features should we include to represent the data? From the available features, which ones should be discarded? Should further information be added, such as adding the day of the week to a timestamp? This can be useful in some tasks. Irrelevant attributes should be removed.
5) *Choosing the data mining function:* Which type of methods should be used? Four types of method are: summarization, clustering, classification and regression. The first two are from the branch of descriptive analytics while the latter two are from predictive analytics.
6) *Choosing the data mining algorithm(s):* Given the characteristics of the problem and the characteristics of the data, which methods should be used? It is expected that specific algorithms will be selected.
7) *Data mining:* Given the characteristics of the problem, the characteristics of the data, and the applicable method type, which specific methods should be used? Which values should be assigned to the hyper-parameters? The choice of method depends on many different factors: interpretability, ability to handle missing values, capacity to deal with outliers, computational efficiency, among others.
8) *Interpretation:* What is the meaning of the results? What is the utility for the final user? To select the useful results and to evaluate them in terms of the application domain is the goal of this step. It is common to go back to a previous step when the results are not as good as expected.
9) *Using discovered knowledge:* How can we apply the new knowledge in practice? How is it integrated in everyday life? This implies the integration of the new knowledge into the operational system or in the reporting system.

For simplicity sake, the nine steps were described sequentially, which is typical. However, in practice, some jumps are often necessary. As an example, steps 3 and 4 can be grouped together with steps 5 and 6. The way we pre-process the data depends on the methods we will use. For instance, some methods are able to deal with missing values, others not. When a method is not able to deal with missing values, those missing values should be included somehow or some attributes or instances should be removed. Also, there are methods that are too sensitive to outliers or extreme values. When this happens, outliers should be removed. Otherwise, it is not necessary to remove them. These are just examples on how data cleaning and pre-processing tasks depend on the chosen method(s) (steps 5 and 6).

1.7.3 The CRISP-DM Methodology

CRoss-Industry Standard Process for Data Mining (CRISP-DM) is a six-step method, which, like the KDD process, uses a non-rigid sequential framework. Despite the six phases, CRISP-DM is seen as a perpetual process, used throughout the life of a company in successive iterations (Figure 1.3).

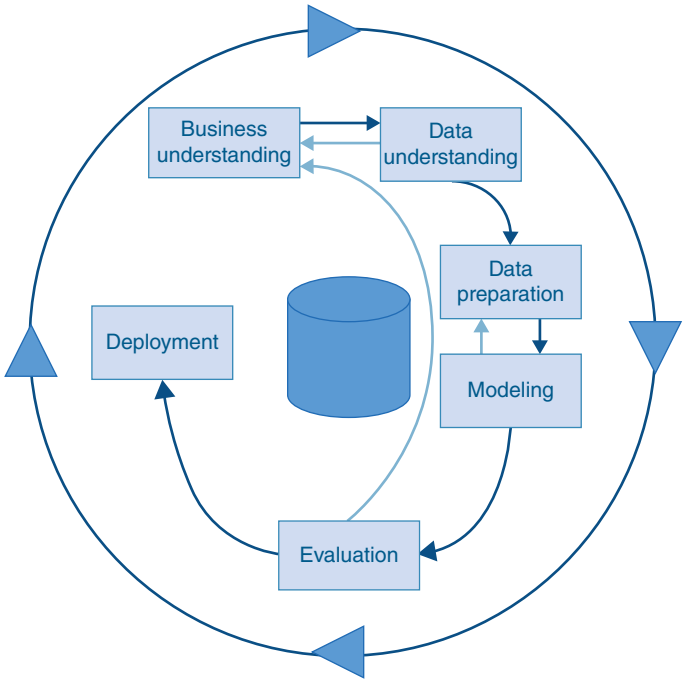


Figure 1.3 The CRISP-DM methodology (adapted from http://www.crisp-dm.org/).

The six phases are:

1) *Business understanding:* This involves understanding the business domain, being able to define the problem from the business domain perspective, and finally being able to translate such business problems into a data analytics problem.
2) *Data understanding:* This involves collection of the necessary data and their initial visualization/summarization in order to obtain the first insights, particularly but not exclusively, about data quality problems such as missing data or outliers.
3) *Data preparation:* This involves preparing the data set for the modeling tool, and includes data transformation, feature construction, outlier removal, missing data fulfillment and incomplete instances removal.
4) *Modeling:* Typically there are several methods that can be used to solve the same problem in analytics, often with specific data requirements. This implies that there may be a need for additional data preparation tasks that are method specific. In such case it is necessary to go back to the previous step. The modeling phase also includes tuning the hyper-parameters for each of the chosen method(s).
5) *Evaluation:* Solving the problem from the data analytics point of view is not the end of the process. It is now necessary to understand how its use is meaningful from the business perspective; in other words, that the obtained solution answers to the business requirements.
6) *Deployment:* The integration of the data analytics solution in the business process is the main purpose of this phase. Typically, it implies the integration of the obtained solution into a decision-support tool, website maintenance process, reporting process or elsewhere.

A more comprehensive description of the CRISP-DM methodology is presented in Appendix A. This will certainly be useful to help you to develop the projects at the end of each of Parts II and III of the book, as explained in Section 1.8.

1.8 How this Book is Organized

The book has two main parts: on descriptive (Part II) and predictive (Part III) analytics respectively.

Parts II and III will finish with a cheat sheet and project chapter (Chapter 7 for Part II, and Chapter 12 for Part III) where the contents of each part are summarized and a project is proposed using one of the two real-world problems presented above (Section 1.6). These projects will be developed using the CRISP-DM methodology, as described in Section 1.7.3 and Appendix A, the latter being a more detailed description. In all other chapters, including this one, we will use as example a small data set from an idealized private list of

contacts in order to better explain the methods. The data set will be presented in the chapters as necessary.

All chapters, excluding this one, the cheat sheet and project chapters, will have exercises. In this book there is no specific software for the examples and exercises. This book was conceived of as a 13-week course, covering one chapter per week. The content of each part is described next.

Part I includes the present chapter, where introductory concepts, a brief methodological description and some examples are presented.

Part II presents the main methods of descriptive analytics and data preprocessing. There are five families of methods/tools covered; one per chapter. The first one, in Chapter 2, is on descriptive statistics. It aims to describe data in a way that us humans can better extract knowledge from. However, the methods described only apply to data with a maximum of two attributes. Chapter 3 extends the discussion in Chapter 2 to an arbitrary number of attributes. The methods described are known as multivariate descriptive statistics methods. Chapter 4 describes methods that are typically used in the data preparation phase of the CRISP-DM methodology, concerning data quality issues, converting data to different scales or scale types and reducing data dimensionality. Chapter 5 describes methods involving clustering, an important technique that aims to find groups of similar instances. Clustering is used in a large number of fields, most notably marketing, in order to obtain segments of clients with similar behavior. Chapter 6 is about a family of descriptive methods known as frequent pattern mining, which aims to capture the most frequent patterns. It is particularly common in the retail market, where it is used in market basket analysis.

Part III presents the main methods of predictive analytics. Chapter 8 is about regression; that is, the prediction of a quantitative attribute. It covers generalization, performance measures for regression and the bias–variance trade-off. It also presents some of the most popular algorithms for regression: multivariate linear regression, ridge and lasso regression, principal component regression and partial least squares regression. In Chapter 9, the binary classification problem is introduced, together with performance measures for classification and methods based on probabilities and distance measures. In Chapter 10 more advanced and state-of-the-art methods of prediction are described: decision trees, artificial neural networks and support vector machines. Chapter 11 presents the most popular algorithms for ensemble learning. Then, a discussion on algorithm bias is presented. Classification tasks other than than binary classification are discussed, as well as other topics relevant for prediction such as imbalanced data classification, semi-supervised learning and active learning. Finally, a discussion on the use of supervised interpretable techniques for descriptive and predictive analysis is presented.

Part IV has only Chapter 13, which discusses briefly applications for text, the web and social media, using both descriptive and predictive approaches.

1.9 Who Should Read this Book

Anyone who aims to extract knowledge from data, whatever they are, could and should read this book. This is a book where the main concepts of data analytics can be understood, and not only by people with a background in engineering and the exact sciences.

You do not need to know statistics – but it helps – or programming. You do not need to be a student of computer sciences, or even a student! You only need to study a little. This book was conceived of as being for bachelor's or master's students, levels where these kinds of analytic tools are relevant. In our experience, more and more people are interested in analyzing data. So this book was written in order to introduce the main tools for data analytics. It aims to be understandable by any university student whichever their background is.

It is expected that after reading this book you will be able to develop a project on analytics, assuming that you are already familiar with the business area of the project. You should be able to identify the necessary data, pre-process and clean it, choose the methods suitable to the project, tune and apply them, evaluate the results in terms of the project purpose, and give the necessary instructions to a development team for deployment.

In order to suit an audience without a background in computer science and/or quantitative methods, the book is particularly focused on explaining the concepts in an intuitive way. Whenever possible, graphics are used to explain the methods. Special attention is given to what must be considered in order to make the right methodological choices. For instance, knowing the meaning of a hyper-parameter allow us to define a strategy for tuning its value.

In summary, it is not expected that after reading this book you will be able to develop new methods or algorithms. But is it expected that you can correctly use appropriate methods to deal with data analytics problems.

Part II

Getting Insights from Data

2

Descriptive Statistics

During the time of the Roman emperor Caesar Augustus, an edict to survey the population was issued. It was expected that everyone was covered: the whole population. It is like that in all studies we do about a population, whatever the population is. It can be the people in a country, the employees of an organization, the animals of a zoo, the cars of an institution, the R&D institutions in a country, all the nails produced in a given machine, and so on. But in many situations it is difficult or even impossible to survey *all* the population. For instance, to collect all the nails ever produced in a machine is typically impossible in practice. In other situations the cost of a survey of all the population can be prohibitive, for instance a survey of all the population ahead of an election. It is theoretically possible, but certainly prohibitive in terms of costs unless the country has few citizens.

That is why sampling is important. By analyzing a subset of the population it is possible to estimate in a quantified way particular values for the population. An example would be the proportion of votes intended for a given party. Generalizing the knowledge obtained from a sample to all of a population is called statistical inference (or induction), since it involves inferring information about the population from a sample. It is important to be aware that we can get many different samples from the same population. So, when we infer a value for the population the inferred value will be different for different samples. But the value obtained from considering all the population is necessarily unique for that population. Of course, as larger the sample is, the closer will be this estimate to the value for the population.

While induction generalizes from the sample to the population, deduction particularizes from the population to the sample. For instance, a deductive problem would be as follows. Given the population of an university, what is the probability of selecting people from two different continents in a random sample of size 10? In other words, knowing the population, the goal is to deduce the nature of a sample of size 10. Yes, probabilities are about deduction.

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

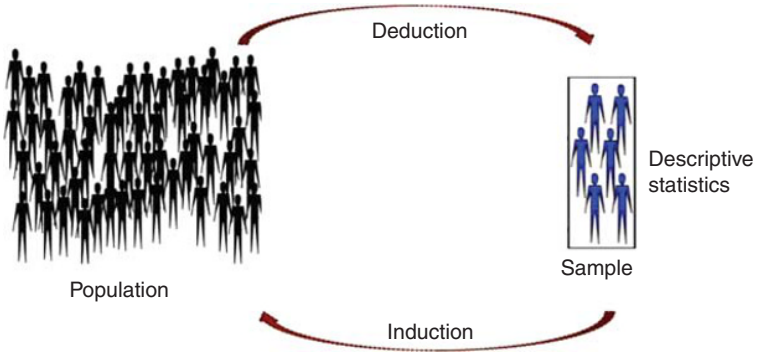


Figure 2.1 The main areas of statistics.

But let us go back to the typical situations we find in data analytics projects. Once we have a data sample, typically obtained through SQL language, we want to get some insights about it. But data are usually too big to look at it as they are. What knowledge can we get from hundreds or thousands of instances? Maybe we do not so much get knowledge but a headache. Descriptive statistics is the branch of statistics that, as the name suggests, sets out methods to describe data samples, through summarization and visualization. Figure 2.1 depicts the relationship between the concepts described so far.

The ways we have to describe and visualize data are usually categorized according to the number of attributes we are considering. The analysis of single attributes is called univariate analysis, for pairs of attributes it is bivariate analysis, and for groups of more than two attributes it is multivariate analysis.

This chapter will show how a data set can be described by descriptive statistics and by visualization techniques for single attributes and pairs of attributes. Several univariate and bivariate statistical formulae and data visualization techniques will be presented. We start by describing the different scale types that exist to describe data.

2.1 Scale Types

Before describing the scale types, let us consider an excerpt from our private list of contacts (Table 2.1). In this chapter we will use the name of the contact, the maximum temperature registered last week in their town, their weight, height and gender, together with the information on how good their company is.

There are two large families of scale types: qualitative and quantitative. Qualitative scales categorize data in a nominal or ordinal way. Nominal data cannot be ordered according to how big or small a certain characteristic is. But ordinal data can.

Table 2.1 Data set of our private list of contacts with weight and height.

Friend	Max temp (°C)	Weight (kg)	Height (cm)	Gender	Company
Andrew	25	77	175	M	Good
Bernhard	31	110	195	M	Good
Carolina	15	70	172	F	Bad
Dennis	20	85	180	M	Good
Eve	10	65	168	F	Bad
Fred	12	75	173	M	Good
Gwyneth	16	75	180	F	Bad
Hayden	26	63	165	F	Bad
Irene	15	55	158	F	Bad
James	21	66	163	M	Good
Kevin	30	95	190	M	Bad
Lea	13	72	172	F	Good
Marcus	8	83	185	F	Bad
Nigel	12	115	192	M	Good

Example 2.1 The name of the contact is expressed on a nominal scale, while the information on how good their company is can be expressed on an ordinal scale because we can define an order of magnitude, ranging from good to bad. Good expresses a higher level of fellowship than bad. This notion of magnitude does not exist in the names.

There are two types of scale for quantitative data: absolute (ratios) and relative (intervals). The difference between them is that in absolute scales there is an absolute zero while in relative scales there is no absolute zero.

Example 2.2 When the attribute "height" is zero it means there is no height. This is also true for the weight. But for the temperature, when we have 0°C it does not mean there is no temperature. When we talk about weight, we can say that Bernhard weighs twice as much as Irene, but we cannot say that the maximum temperature last week in Dennis' home town was twice that in Eve's. This is why we usually use a change in temperature to characterize how the temperature varied in a given day instead of a ratio.

The information we can get depends on the scale type we use to express the data. In fact we can order the four scale types in the following way: the most informative one is the absolute scale, and then the relative, the ordinal and the nominal scale types (see Figure 2.2).

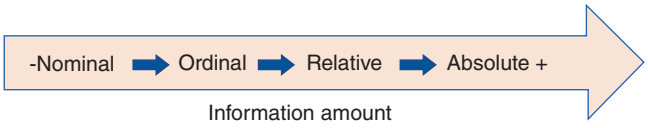


Figure 2.2 The relation between the four types of scales: absolute, relative, ordinal and nominal.

We can also express the data according to the operations we can perform on their values. The only operations we can apply to two nominal values are related to their similarity, in other words to see if they are equal ($=$) or different ($\neq$). For two ordinal values, we can also check their order, to see if one is larger than ($>$), larger than or equal to ($\geq$), smaller than ($<$) or smaller than or equal to ($\leq$) the other. For two relative values, as well as the operations valid for ordinal values, we can also see how much we need to add to ($+$) or subtract from ($-$) one value to get to the other. Finally, for two absolute values, in addition to all previous operations, we can also see how many times a value is larger ($\times$) or smaller ($\div$) than the other.

This all means that when we have data expressed on an absolute scale we can convert it to any of the other scales. When we have data expressed on a relative scale we can convert it in any scale of the two qualitative scale types. When we have data expressed on an ordinal scale we can express it in a nominal scale. But we should be aware that converting a more informative scale into a less informative one involves loss of information.

Converting a less informative scale into a more informative one is also possible, although the level of information obtained after the conversion will be necessarily limited by the information contained in the original scale. However, this is sometimes required for methods that expect quantitative values for the hyper-parameters.

Example 2.3 As an example, consider the attribute "weight" expressed in an absolute scale in kilograms. We can convert it to any other scale:

- *Relative:* The weight can be converted to a relative scale by, for instance, subtracting a value of 10. The old zero becomes -10 and the new zero is the old 10. That means that the new zero does not mean anymore that there is no weight. The new 80kg is no longer twice the new 40kg. Try to figure out why.
- *Ordinal:* We can define, for instance, levels of fatness: "fat" when the weight is larger than 80kg, "normal" when the weight is larger than 65kg but less than or equal to 80kg, and "thin" when the weight is less than or equal to 65kg. With this classification, we still have the possibility to define groups of people as being more or less fat. Why have we chosen 65 and 80kg as the levels of fatness? There was no special reason, but there is a rationale, as will see in Section 4.2.

- *Nominal:* We can transform the previous classification – fat, normal and thin – into B, A and C, respectively. With such a classification it is not possible to order the contacts according to how fat they are, because B, A and C do not quantify anything.

In software packages we must choose the data type for each attribute. These types depend on the software package. Common types are text, character, factor, integer, real, float, timestamp and date. There are several others. A scale type and a data type are different concepts, although they are related. For instance, a quantitative scale type implies the use of numeric data types (integer, real or float are specific numeric data types). Among the numeric data types, some express discrete values such as the integer data type, while others express continuous values such as the float and the real data types.

Be attentive: an attribute can be expressed as a number but the scale type does not have to be quantitative. It can be ordinal or even nominal. Think about a card you have with a numeric code. What kind of quantitative information does it contain? The answer is "nothing": it is just a key. Its value may eventually express how old the card is but, typically, nothing more than that. If it was a code with letters it would contain the same information.

2.2 Descriptive Univariate Analysis

In descriptive univariate analysis, three types of information can be obtained: frequency tables, statistical measures and plots. These different approaches are described in this section.

2.2.1 Univariate Frequencies

In order to illustrate some additional measures, we will use again our sample dataset of contacts (Table 2.1). Two different samples taken randomly from the same population would have typically two different empirical distribution functions.

A frequency is basically a counter. The absolute frequency counts how many times a value appears. The relative frequency counts the percentage of times that value appears.

Example 2.4 The absolute and relative frequency for the attribute "company" can be seen in Table 2.2.

The absolute and relative frequencies for the attribute "height" and the respective cumulative frequencies are as shown in Table 2.3. The absolute and relative cumulative frequencies are, respectively, the number and the

Table 2.2 Univariate absolute and relative frequencies for "company" attribute.

Company	Absolute frequency	Relative frequency
Good	7	50%
Bad	7	50%

Table 2.3 Univariate absolute and relative frequencies for height.

Height	Abs. freq.	Rel. freq.	Abs. cum. freq.	Rel. cum. freq.
158	1	1/14 = 7.14%	1	7.14%
163	1	7.14%	2	14.28%
165	1	7.14%	3	21.42%
168	1	7.14%	4	28.56%
172	2	14.29%	6	42.85%
173	1	7.14%	7	49.99%
175	1	7.14%	8	57.13%
180	2	14.29%	10	71.42%
185	1	7.14%	11	78.56%
190	1	7.14%	12	85.70%
192	1	7.14%	13	92.84%
195	1	7.14%	14	99.98%

percentage of occurrences less than or equal to a given value. The value of the absolute cumulative frequency of the last row is always the total number of instances, while the value of the relative cumulative frequency of the last row is always 100%, although perhaps with some decimal differences due to rounding of intermediate values, as shown in Table 2.3.

While for qualitative scales this information can be useful if there are not too many classes – different values for the attribute "company" – for quantitative scales, the number of repetitions is typically low, implying many values with a low number of observations. This is especially uninformative when using plots, as we will see next.

The relative frequencies define distribution functions; that is, they describe how data are distributed. The column "rel. freq." in Table 2.3 is an example of an empirical frequency distribution while the column "rel. cum. freq." is an example of an empirical cumulative distribution function. They are referred to as "empirical" because they are obtained from a sample.

Distribution functions from populations can be either probability distribution functions or probability density functions, depending on the data type of the attribute. A discrete attribute, such as the integer data type, has a probability mass function, while a continuous attribute, such as the real data type, has a probability density function.[1] The reason for this distinction is that in a continuous space the probability of being an exact value is zero. This might seem strange, but let us think about it. If you store the data with all the decimal places, how tall are you? 177 cm? No. I guess you are rounding your height to centimeters. Be exact, which is your real height? 177 plus some fraction of a centimeter that you are not able to define precisely. That is it: if everybody was able to define their height with maximal precision (that is infinite precision) there would be no one with the same height, or if there was, the probability would be so so low that it would be roughly zero. That is the reason why, for continuous attributes, probability density functions are used. They count relative *densities* while probability distribution functions count relative *frequencies*. A property of probability density functions is that its area is always one, representing 100%.

2.2.2 Univariate Data Visualization

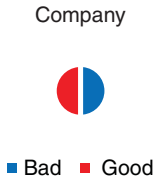
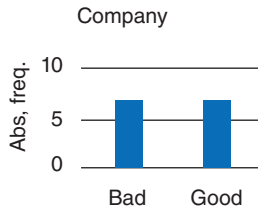
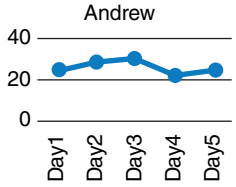
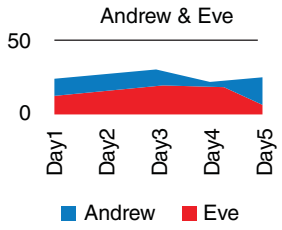
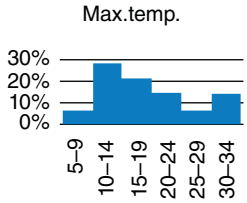
Table 2.4 shows the most common types of univariate plots and their applicability to the different types of scales. Additional details of each of these five different types of charts are discussed next:

Pie chart These are used typically for nominal scales. It is not advisable to use them with scales where the notion of order exists – in other words for ordinal and quantitative scales – although this is possible.

Bar charts These are used typically for qualitative scales. When the notion of order exists, the classes should be displayed in the horizontal bar, typically in increasing order of magnitude. Many authors argue that bar charts are better for comparing values between different classes than pie charts because it is easier to see that one bar is bigger than another than to see that one pie slice is larger than another. In some situations bar charts are also used with quantitative scales, for example where the possible value of an attribute is limited in size. For example, the result of counting the number of times each of the six faces of a die appears could be readily represented in a bar chart (integer numbers from 1 to 6). Another example could be the number of students with a given mark on a 0–20 integer scale (integers from 0 to 20).

1 The terms mass and density functions are inspired by the similar relationship between mass and density.

Table 2.4 Univariate plots.

Plot	Qualitative	Quantitative	Observation	Plot draft
Pie	Yes	No	Company relative frequency	
Bar	Yes	Not always	Company absolute frequency	
Line	No	Yes	Andrew's 5-day max. temperatures	
Area	No	Yes	Andrew and Eve 5-day max. temperatures	
Histogram	No	Yes	Max. last day temperatures of the 14 contacts	

Line charts Like area charts, these are used when the horizontal bar uses a quantitative scale with equal lag between observations. In particular, they are used to deal with the notion of time. Indeed, line charts are quite useful to

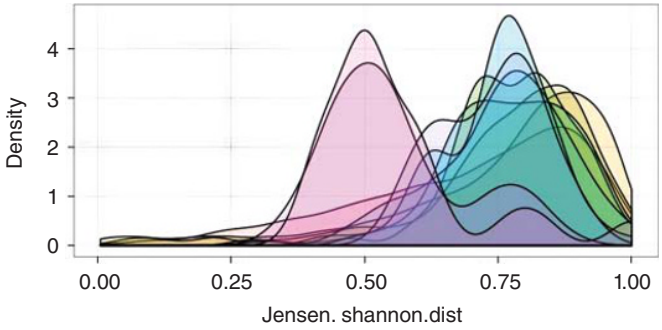


Figure 2.3 An example of an area chart used to compare several probability density functions.

represent time series, graphs of values obtained over regular time sequences. In the example, five consecutive observations of the maximum daily temperature at Andrew's town are shown (the data used for these plots is not in Table 2.1). In real life we see often line plots to analyze the evolution of assets in the stock market or, say, to analyze child mortality rates in a given country over time, or how the GDP of a country has evolved over time.

Area charts Area charts are used to compare time series and distribution functions. Figure 2.3 shows several probability density functions. Understanding data distributions give us strong insights about an attribute. We are able to see, for instance, that data are more concentrated in some values or that other values are rare.

Histograms These are used to represent empirical distributions for attributes with a quantitative scale. Histograms are characterized by grouping values in cells, reducing in this way the sparsity that is common in quantitative scales. You can see in Figure 2.4 that the histogram is more informative than the bar chart.

An important decision in drawing a histogram is to define the number of cells (in the histogram in Table 2.4 we used eleven). The number of cells can have an important effect on the plot. The suitable value is problem dependent. As a rule of thumb, it is around the square root of the number of values. When cells are defined it is usual and advisable to have no space between the columns in order to preserve the idea of continuity that is assumed in a histogram. It is easier, although not always the best option, to use equal-sized cells. When a cell has a larger width, the height of the cell should decrease by the same proportion in order to preserve the area of the cell. For instance, if the width increases to double that of the standard cells, its height should be half of the expected height.

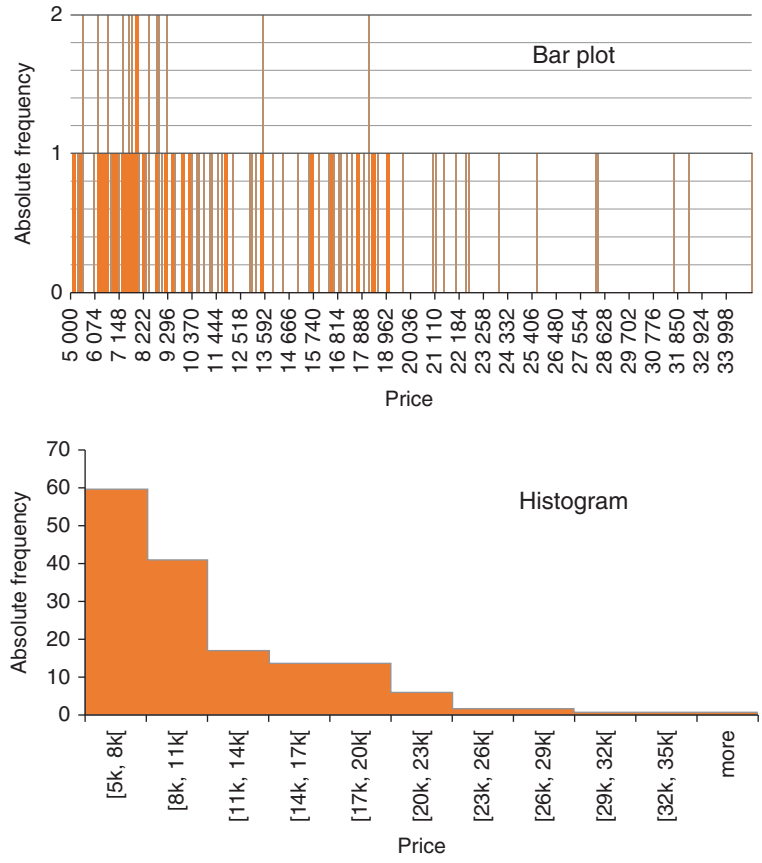


Figure 2.4 Price absolute frequency distributions with (histogram) and without (bar chart) cell definition.

A last note about histograms: do not forget to use good sense. It is nonsense to define cell limits like [8.8, 13.3[. It is good sense to use easy-to-memorize limits such as [10, 15[.

All distribution functions we have seen up to now were based on relative or absolute frequencies of data samples. Let us now examine cumulative distribution functions and consider how different empirical and probabilistic ones are. But remember that empirical distributions are based on samples while probability distributions are about populations.

In Figure 2.5 we see an empirical cumulative distribution based on a sample taken from a population with a known probability density distribution, which

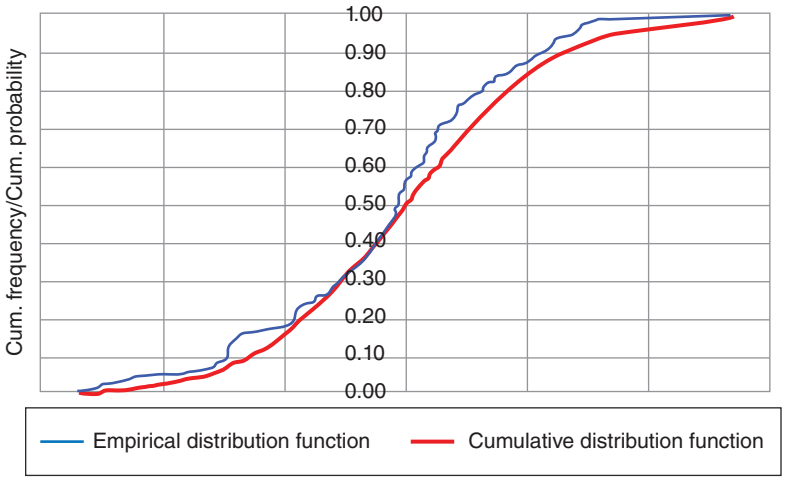


Figure 2.5 Empirical and probability distribution functions.

is also depicted. The step-wise nature of the empirical cumulative probability distribution is typical and easily understandable. There are two reasons:

- the empirical distribution has only some of the values of the population, so there are jumps
- the values are usually obtained at a certain predefined level of precision (for instance, the height can be represented in centimeters) creating jumps between numbers that do not exist in the population (with infinite precision).

Cumulative functions are very informative despite it being necessary to get used to reading them. To help you: the more horizontal the line is, the less frequent the values in the horizontal bar are; the more vertical the line is, the more frequent these values in the horizontal bar are.

How can we design the probability distribution of a population? Do we need access to all instances of that population? A large number of situations in real life follows some already known and well defined function. So, although the answer is problem dependent, in many cases the answer is no – we do not need to access all instances of a given population, as we will see in Section 2.2.4. Before then, some of the most used descriptors of a sample or population are described. These descriptors are known as statistics. Statistics for a single attribute are known as univariate statistics.

As motivaton for Section 2.3, we can also represent frequencies for combined values of two different attributes in a single chart. This is illustrated in Figure 2.6, where the frequencies for the target value of "company" in Table 2.1 is split by gender. This kind of plot is known as stacked bar plot.

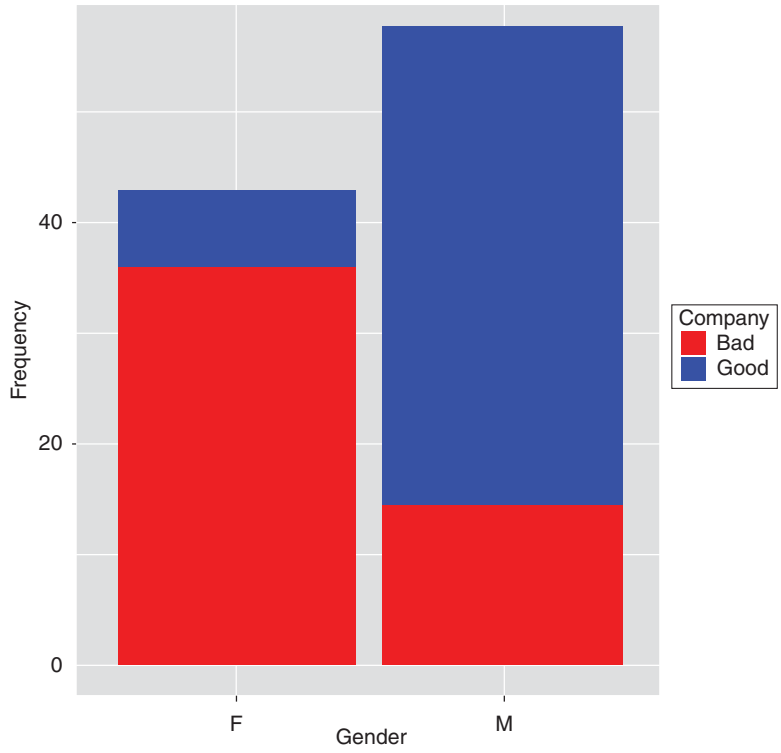


Figure 2.6 Stacked bar plot for "company" split by "gender".

2.2.3 Univariate Statistics

A statistic is a descriptor. It describes numerically a characteristic of the sample or the population. There are two main groups of univariate statistics: location statistics and dispersion statistics.

Location univariate statistics Location statistics identify a value in a certain position. Some well known location univariate statistics are the minimum, the maximum or the mean. Let us look at some of the more important ones:

- minimum: the lowest value
- maximum: the largest value
- mean: the average value, obtained by summing all values and dividing the result by the number of values
- node: the most frequent value;
- first quartile: the value that is larger than 25% of all values

- median or second quartile: the value that is larger than 50% of all values; the value that splits the sequence into two equal-sized sub-sequences
- third quartile: the value that is larger than 75% of all values.

The mean (or average), median and mode are known as measures of central tendency, because they return a central value from a set of values.

Example 2.5 Let us use as an example the attribute "weight" from our data set (Table 2.1). The values for each of the statistics described above are shown in Table 2.5.

Graphically they are positioned as shown in Figure 2.7.

However, there are other more popular ways to express graphically the location statistics. An example is the box-plot. Box-plots present the minimum, the first quartile, the median, the third quartile and the maximum, in this order obviously, bottom-up or from left to right.

Example 2.6 For our example, a box-plot for the attribute "height" is presented in Figure 2.8. The bottom and top points of the plot are respectively the minimum and the maximum. The bottom and the top of each box are respectively the first and the third quartiles. The horizontal line in the middle of the box is the median. The closer each of the points is, the more frequent the values between these points are. See, for example, the distance between the first quartile and the median. Twenty-five percent of the values are found between these two statistics.

Mean, median and mode are measures of central tendency. When should we use each of them? Table 2.6 shows which central tendency statistics can be used according to the type of scale.

Table 2.5 Location univariate statistics for weight.

Location statistic	Weight (kg)
Min	55.00
Max	115.00
Average	79.00
Mode	75.00
First quartile	65.75
Median or second quartile	75.00
Third quartile	87.50

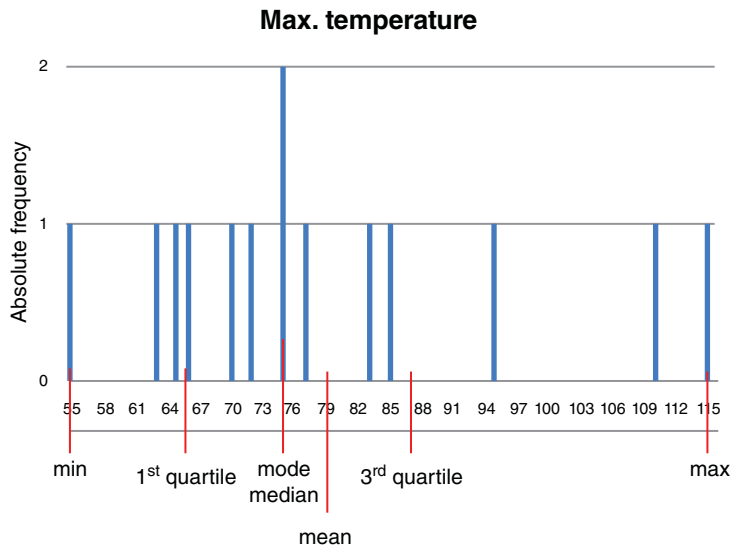


Figure 2.7 Location statistics on the absolute frequency plot for the attribute "weight".

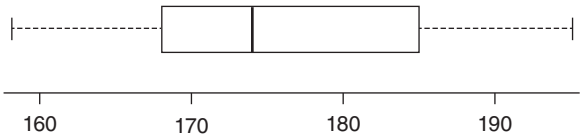


Figure 2.8 Box-plot for the attribute "height".

Table 2.6 Central tendency statistics according to the type of scale.

	Nominal	Ordinal	Quantitative
Mean	No	Eventually*	Yes
Median	No	Yes	Yes
Mode	Yes	Yes	Yes

*See below.

Some additional observations should be made:

- Box-plots can also be used to describe how symmetric/skewness the distribution of an attribute is. As illustrated in Figure 2.8, the values of the attribute "height" are skewed right, meaning that the values above the median are more extended in the plot than the values below the median. If the median is close to the center of the box, the data distribution is typically symmetric, and the values are similarly distributed in the low part and in the high part.

- The median or the mode are more robust as a central tendency statistic than the mean in the presence of extreme values or strongly skewed distributions. Figure 2.9 shows a symmetric and an asymmetric distribution and the location of the central tendency statistics for each. As we can see, the median, the mode and the mean have the same value in symmetric distributions with a single mode. Distributions with a single mode are called unimodal distributions.
- The mode is not useful when the data are very sparse; that is, when there are very few observations per value. This is quite common when we use quantitative scales, especially from continuous data types.
- The median is easy to obtain when the number n of observations is odd. You only need to order the observations according to the values. The median is the value in the position $(n + 1)/2$. But if n is an even number, the median will be the average of the values in the positions $n/2$ and $(n/2) + 1$.
- Despite the mean being, strictly speaking, unsuitable for ordinal scales, it is used in some cases, namely when using the Likert ordinal scale. This explains the "eventually" note in Table 2.4. The Likert scale is very popular for surveys. It uses an ordered scale, say integers from 1 to 7, expressing a level of highest disagreement (1) to highest agreement (7). Although, in the example in Figure 2.10, these values represent an order, they can also be interpreted as a quantity of agreement/disagreement. In this case the Likert scale can be seen in some ways as a quantitative scale. However, this is a debatable point and there is no agreement among statisticians. Discussion of the use of the mean and the Likert scale can be found in the literature.
- Plots can also be combined. A combination of a box-plot and a histogram for the "length" attribute is shown in Figure 2.11. Each vertical bar above the horizontal axis corresponds to one of the values of the height attribute.

There are statistics for samples and for populations. Given a population, there is only one value for a given statistic for that population. The same happens with samples. Given a sample, there is only one value of the statistic for that sample. Since for one population we can have several samples, for that population there is only one population value for the given statistic but several sample values for the same statistic: one per sample.

There are different notations for the statistics depending whether they are population or sample statistics. The statistics calculated so far use the same calculations for samples and populations but the notation is different. You should be particularly aware of the notation of the mean because it will be used in other formulas. The *population* mean of attribute x is denoted as μ_x while the *sample* mean value is denoted as $\overline{x}$.

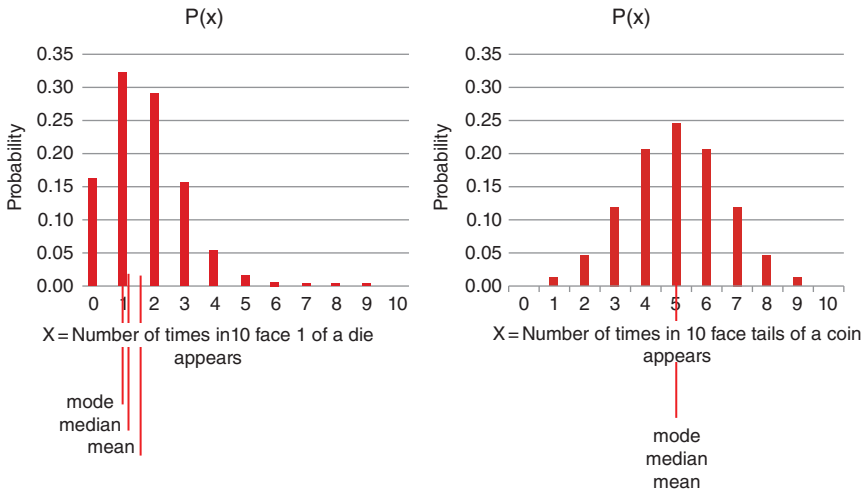


Figure 2.9 Central tendency statistics in asymmetric and symmetric unimodal distributions.

Please circle the number that better fits your experience with the given information
I am satisfied with it Strongly disagree 1 2 3 4 5 Strongly agree
It is simple to use Strongly disagree 1 2 3 4 5 Strongly agree
It has good graphics Strongly disagree 1 2 3 4 5 Strongly agree
It is in accordance to my expectations Strongly disagree 1 2 3 4 5 Strongly agree
Everything make sense Strongly disagree 1 2 3 4 5 Strongly agree

Figure 2.10 An example of the Likert scale.

Dispersion univariate statistics A dispersion statistic measures how distant different values are. The most common dispersion statistics are:

- amplitude: the difference between the maximum and the minimum values
- interquartile range: is the difference between the values of the third and first quartiles
- mean absolute deviation: a measure for the mean absolute distance between the observations and the mean. Its mathematical formula for the population is:

$$MAD_x = \frac{\sum_{i=1}^{n} |x_i - \mu_x|}{n}, \tag{2.1}$$

where n is the number of observations and μ_x is the mean value of the population. In this case the distance between an observation and the mean contributes to the MAD in linear proportion to that distance. For example, one observation with a distance to the mean of 4 will increase the MAD by the same amount as two observations each with a distance to the mean of 2.

- standard deviation: another measure for the typical distance between the observations and their mean. Its mathematical formula for the population is:

$$\sigma_x = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu_x)^2}{n}}, \tag{2.2}$$

where n is the number of observations and μ_x is the mean value of the population. In this case the distance between an observation and the mean contributes to the standard deviation in quadratic proportion to that distance. For example, one observation with a distance to the mean of 4 will increase σ more than two observations each with a distance to the mean of 2. The square of the sample deviation is termed the variance and is denoted as σ^2. It measures how spread out the population values are around the mean.

All of these dispersion statistics are only valid for quantitative scales.

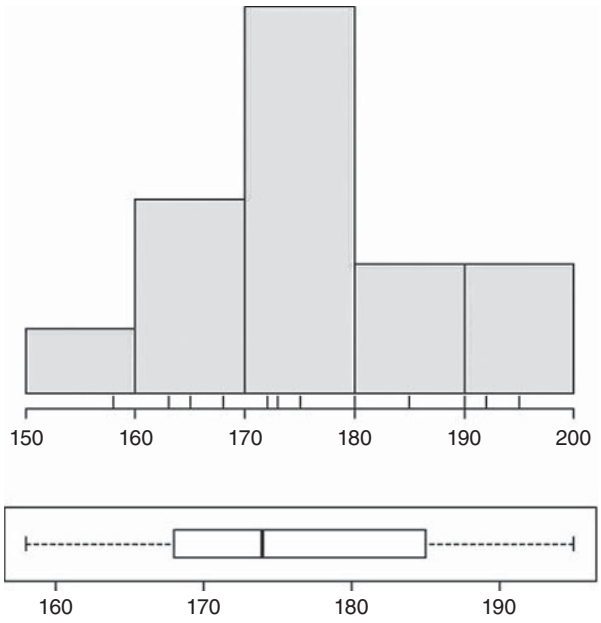


Figure 2.11 Combination of a histogram and a box-plot for the "height" attribute.

Table 2.7 Dispersion univariate statistics for the "weight" attribute.

Dispersion statistic	Weight (kg)
Amplitude	60.00
Interquartile range	21.75
$\overline{MAD}$	14.31
s	17.38

The formulas for the mean absolute deviation and the standard deviation assume that we know the value of μ_x. However, when we have a sample, instead of μ_x, we typically have $\overline{x}$. When this happens, instead of n independent values in relation to $\overline{x}$ we have only $n-1$. Consider an example with three observations. Two of them have values 1 and 2. We know that the average of the 3 observations is 2. What is the third value, denoted as x? It is $(1+2+x)/3 = 2$, so $1+2+x = 6$, consequently the third value x is $x = 3$. That is, we only have 2, that is $n-1$ independent values in relation to $\overline{x}$. For that reason, we calculate the sample mean absolute deviation $\overline{MAD}$ and the sample standard deviation as shown in the following formulas:

$$\overline{MAD}_x = \frac{\sum_{i=1}^{n} |x_i - \overline{x}|}{n-1}, \tag{2.3}$$

and

$$s_x = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}}, \tag{2.4}$$

The sample variance is denoted as s^2 and is, as expected, the square of s.

Example 2.7 Using again as an example the "weight" attribute, dispersion statistics can be calculated, as shown in Table 2.7. The equations for the sample mean absolute deviation and the sample standard deviation are used since the example data set is a sample.

2.2.4 Common Univariate Probability Distributions

Each attribute has its own probability distribution. Many common attributes follow functions for which the distribution is already known. There are many different common probability distributions, which can be found in any introductory book on statistics [10].

We present two of these distributions: the uniform and the normal, the latter also known as the Gaussian. Both are continuous distributions and have known probability density functions.

The uniform distribution The uniform distribution is a very simple distribution. The frequency of occurrence of the values is uniformly distributed in a given interval of values. An attribute x that follows a uniform distribution with parameters a and b, respectively the minimum and maximum values of the interval, is denoted as:

$$x \sim \mathcal{U}(a, b) \tag{2.5}$$

Knowing the distribution it is possible to design its probability density function (Figure 2.12) and calculate probabilities. Probabilities measure the likelihood of an attribute taking a value or a range of values. A probability is to a population as a relative frequency is to a sample. In a sample we talk about proportions – relative frequencies – while at the population level we talk about probabilities. In a continuous population the probability of being equal to a given value is always zero, as explained before. So, in continuous distributions the probabilities are calculated per interval. Let us use as an example the generation of a random number between 0 and 1, a function available in many calculators. In this case we would say that the random number $x \sim \mathcal{U}(a = 0, b = 1)$. The probability of $x < 0.3$ is given by the proportion of the area taken by this term, as shown in Figure 2.12.

It is represented mathematically by the formula:

$$P(x < x_0) = \begin{cases} 0, \text{if } x_0 < a; \\ \frac{x_0 - a}{b - a}, \text{if } a \leq x_0 \leq b; \\ 1, \text{if } x_0 > b. \end{cases} \tag{2.6}$$

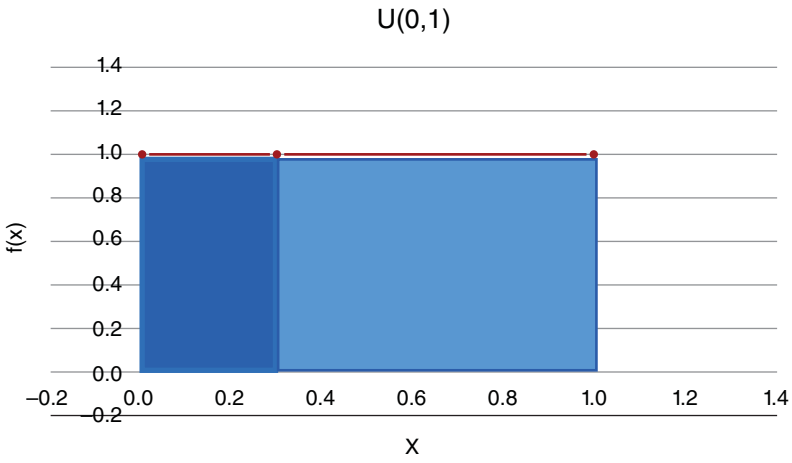


Figure 2.12 The probability density function, $f(x)$ of $x \sim \mathcal{U}(0, 1)$.

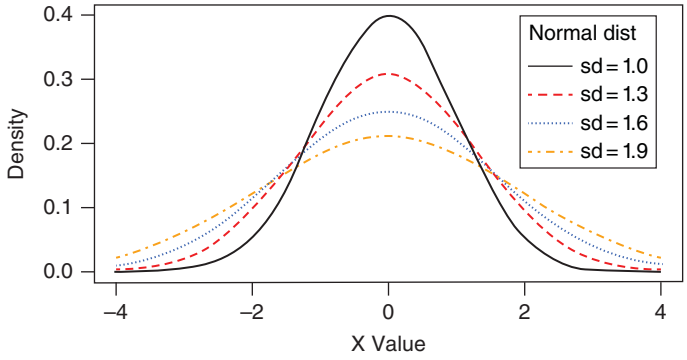


Figure 2.13 The probability density function for different standard deviations, $\mathcal{N}(0, \sigma = sd)$.

The mean and the variance of the uniform population can be obtained using the following formulas, respectively:

$$\mu_x = \frac{a+b}{2} \tag{2.7}$$

$$\sigma_x^2 = \frac{(b-a)^2}{12} \tag{2.8}$$

The normal distribution The normal distribution, also known as Gaussian distribution, is the most common distribution, at least for continuous attributes. This happens due to an important theorem in statistics, known as the central limit theorem, which is the basis of many of the methods used in inductive learning. Physical quantities that are expected to be the sum of many independent factors (say, people's heights or the perimeter of 30-year-old oak trees) typically have approximately normal distributions. The normal distribution is a symmetric and continuous distribution, as shown in Figure 2.13. It has two parameters: the mean and the standard deviation. While the mean localizes the highest point of the bell-shaped distribution, the standard deviation defines how thin or wide the bell shape of the distribution is. An attribute x that follows the normal distribution is denoted as $x \sim \mathcal{N}(\mu_x, \sigma_x)$.

2.3 Descriptive Bivariate Analysis

This section is about pairs of attributes, and their relative behavior. It is organized according to the scale types of the attributes: quantitative, nominal and ordinal. When one of the attributes of the pair is qualitative – that is, nominal or ordinal – and the other is quantitative, box plots can be used, as discussed in Section 2.2.3.

2.3.1 Two Quantitative Attributes

In a data set whose objects have n attributes, each object can be represented in a n-dimensional space: a space with n axes, each axis representing one of the attributes. The position occupied by an object is given by the value of its attributes.

There are several visualization techniques that can visually show the distribution of points with two quantitative attributes. One of these techniques is an extension of the histogram called a three-dimensional histogram.

Example 2.8 Figure 2.14 shows the histogram for the attributes "weight" and "height", illustrating how frequently two values of these attributes occur in Table 2.1.

However, depending on the frequencies for particular combinations of the two attributes, some bars can be hidden.

Another option is the use of scatter plots. Scatter plots illustrate how the values of two attributes are correlated. They make it possible to see how an attribute varies according to the variability of the other attribute.

Example 2.9 Figure 2.15 shows the scatter plot for the attributes "weight" and "height". It can be seen as a general tendency that people with larger weights are taller, and people with smaller weights are shorter.

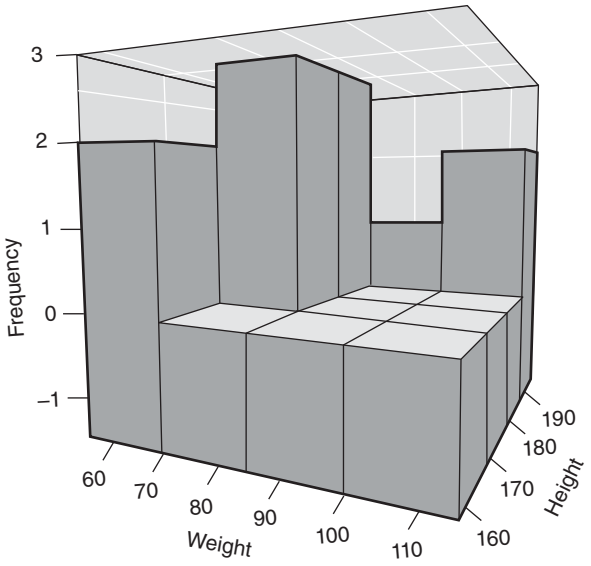


Figure 2.14 3D histogram for attributes "weight" and "height".

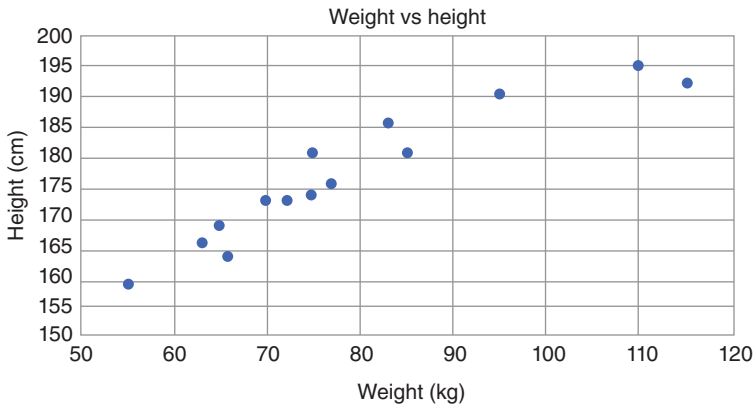


Figure 2.15 Scatter plot for the attributes "weight" and "height".

The degree to which these relations exist – that is, how an attribute varies when a second attribute is changed – is measured by the covariance between them. When two attributes have a similar variation, the covariance has a positive value. If the two attributes vary in the opposite way, the covariance is negative. The value depends on the magnitude of the attributes. If they seem to have independent variation, the covariance value tends to zero. It must be observed that only linear relations are captured. The variance can be seen as a special case of covariance: it is the covariance of an attribute with itself.

Equation (2.9) shows how the covariance between two attributes, x_i and x_j is calculated. In this equation, x_{ki} and $\overline{x}_i$ are, respectively, the kth value and the average of the attribute x_i.

$$s_{ij} = \text{cov}\,(x_i, x_j) = \frac{1}{n-1}\sum_{k=1}^{n}(x_{ki} - \overline{x}_i)(x_{kj} - \overline{x}_j) \tag{2.9}$$

Although covariance is a useful measure to show how the values of two attributes relate to each other, the size of the range of values of attributes influences the covariance values obtained. Of course you can always normalize the attributes to the same interval. However, there is also a similar measure that is not affected by this deficiency: the correlation measure. The linear correlation between two attributes, also known as Pearson correlation, gives a clearer indication of how similar the attributes are, and is usually preferred to the covariance.

Figure 2.16 illustrates three examples of correlations between two attributes, A and B: a positive correlation, a negative correlation and a lack of correlation.

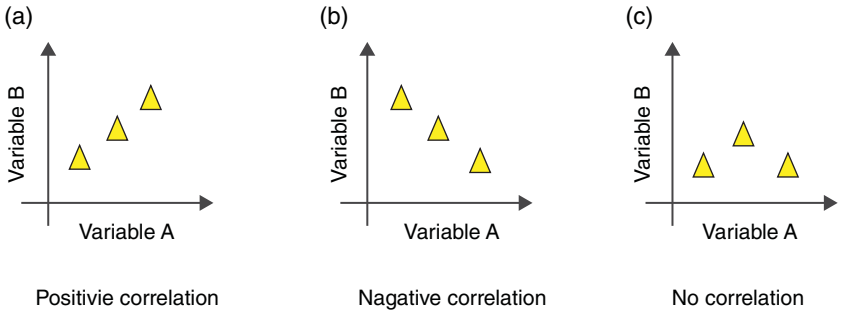


Figure 2.16 Three examples of correlation between two attributes.

It can be seen from that the more correlated are the attributes, the closer the points are to being in a straight line.

To calculate the linear correlation between two attributes, x_i and x_j, we can use Equation (2.10), where $cov(x_i, x_j)$ is the covariance equation and s_i and s_j are the sample standard deviations of the attributes x_i and x_j, respectively.

$$r_{ij} = \text{cor}(x_i, x_j) = \frac{\text{cov}(x_i, x_j)}{s_i s_j} \tag{2.10}$$

The Pearson correlation evaluates the linear correlation between the attributes. If the points are in an increasing line, the Pearson correlation coefficient will have a value of 1. If the points are in a decreasing line, its value will be -1. A value of 0 is when the points form a horizontal line or a cloud without any increasing or decreasing tendency, meaning the nonexistence of a Pearson correlation between the two attributes. Positive values mean the existence of a positive tendency between the two attributes; as it becomes closer to a straight line, the Pearson correlation value becomes closer to 1. Similarly, negative values mean the existence of a negative tendency, the Pearson correlation becoming closer to -1 as the tendency becomes closer to a straight line.

Example 2.10 In our example, the value of the Pearson correlation between weight and height is $\rho_{x,y} = 0.94$ which is quite high.

There are different correlation functions. The most frequently used are the one we have described – the Pearson correlation – and Spearman's rank correlation. Both have values in the range $[-1, 1]$.

Spearman's rank correlation, as the name suggests, is based on rankings. Instead of evaluating how linear is the shape formed by the points, it compares ordered lists of each of the two attributes. The formula is similar to the one

used to calculate the Pearson correlation coefficient, but instead of using the values, it uses the order of the values in the rank, *rx* and *ry* respectively.

$$r_{x,y} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}[(rx_i - \overline{rx}) \times (ry_i - \overline{ry})]}{s_{rx} \times s_{ry}}, \tag{2.11}$$

where n is again the number of pairs.

Example 2.11 The ranking order for the weight and the height are shown in Table 2.8. When there is a draw, the value used is the average of the positions they would have occupied if there was no draw. The order is ascending rank. The value obtained is $r_{x,y} = 0.96$, which is even higher than $\rho_{x,y} = 0.94$.

It is important to understand how different these two coefficients are. Figure 2.17 shows an example where Spearman's rank correlation coefficient is 1 while the Pearson correlation coefficient is 0.96. Do you understand why? Were you able to find an example with attributes x and y where $\rho_{x,y} = 1$ and $r_{x,y} < 1$?

Table 2.8 The rank values for the attributes "weight" and "height".

Weight	Height
1.0	1.0
4.0	2.0
2.0	3.0
3.0	4.0
5.0	5.5
6.0	5.5
7.5	7.0
9.0	8.0
7.5	9.5
11.0	9.5
10.0	11.0
12.0	12.0
14.0	13.0
13.0	14.0

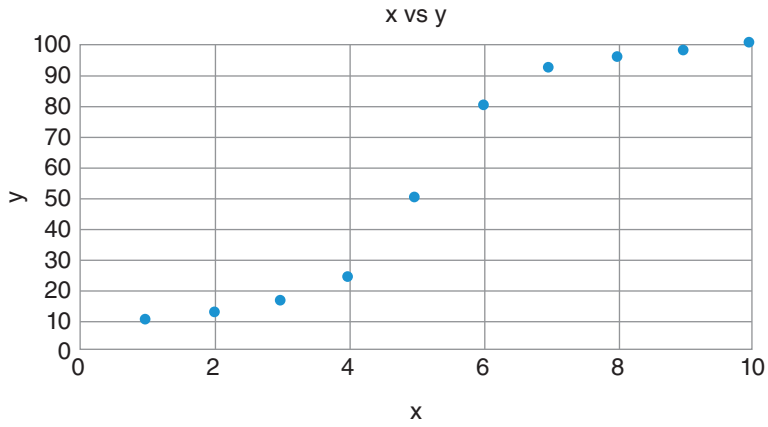


Figure 2.17 The scatter plot for the attributes *x* and *y*.

2.3.2 Two Qualitative Attributes, at Least one of them Nominal

When the attributes are both qualitative with at least one nominal, contingency tables are used. Contingency tables present the joint frequencies, facilitating the identification of interactions between the two attributes. They have a matrix-like format, with cells in a square and labels at the left and top. On the right most column are the totals per row while in the bottom most row are the totals per column. The bottom right-hand corner has the total number of values.

Example 2.12 Figure 2.18 shows the contingency table for the attributes "gender" and "company". This example uses absolute joint frequencies. Relative joint frequencies could also be used. It can be seen that six out of seven people who are rated as good company are men, while only one woman is considered good company. Two of the seven people rated as bad company are men while five are women. The same data can be read as six out of eight men are good company while two are bad company. Five out of six women are bad company while one is good company. There are eight men and six women, totaling fourteen people, seven of whom are good company. The other seven are bad company.

Mosaic plots are based on contingency tables, showing the same information in a more appealing visual way. The areas displayed are proportional to their relative frequency.

Example 2.13 Figure 2.19 uses the same data as in Figure 2.18. The bar for the men (M) is larger than that for the women (F) according to the frequency

		Company		
		Good	Bad	
Gender	Male	6	2	8
	Female	1	5	6
		7	7	14

Figure 2.18 Contingency table with absolute joint frequencies for "company" and "gender".

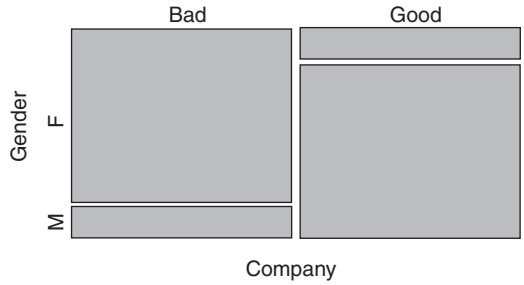


Figure 2.19 Mosaic plot for "company" and "gender".

of men and women. The rectangle with the largest area corresponds to men who are good company; it is the cell with largest frequency in the contingency table (Figure 2.18).

2.3.3 Two Ordinal Attributes

Any of the methods previously described for bivariate analysis can also be used in the presence of two ordinal attributes. However:

- Spearman's rank correlation should be used instead of the Pearson correlation.
- Scatter plots with ordinal attributes usually have the problem that there are many values falling at the same point, making it impossible to evaluate the number of values per point. In order to avoid this problem, some software packages use a jitter effect which add a random deviation to the values, making it possible to evaluate how large the cloud is.
- Contingency tables can be used and mosaic plots too. The values should be in increasing order.

2.4 Final Remarks

This chapter has described how the main characteristics of a data set can be summarized by simple statistical measures, visualization plots and probability distributions. It concentrated on data sets with one or two attributes. Regarding the statistical measures, frequency, localization and dispersion measures, such as the mean, median, mode, the quartiles, amplitude, variance and standard deviation were introduced. The visualization plots illustrate some of these measures, in plots such as histograms, box plots, scatter plots and mosaic plots. The use of a small number of attributes was intentional, to make it easier to describe some of the more important measures.

We know by now that most real data sets have more than two attributes. Chapter 3 describes how to analyze multivariate data: that is, with more than two attributes.

2.5 Exercises

1 What are the most appropriate scales for the following examples?
- university students' exam marks
- level of urgency in the emergency room of a hospital
- classification of the animals in a zoo
- carbon dioxide levels in the atmosphere.

2 Present the absolute and relative frequencies and respective cumulative frequencies for the attribute "weight" in Table 2.1.

3 Choose the most appropriate plot for the following attribute(s) from Table 2.1:
- weight
- gender
- weight per gender.

4 Draw a histogram for the "height" attribute from Table 2.1

5 Calculate the location and the dispersion statistics for the attributes "weight" and "gender" from Table 2.1.

6 Which measure of central tendency would you choose, and why, for:
- bus travel times in a given route
- student exam marks
- waist size of trousers sold in a shop.

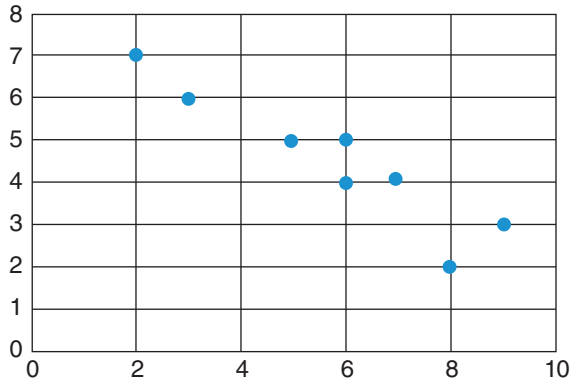


Figure 2.20 Scatter plot.

7 Which are the probability distributions of the following attributes?
- weights of adult men.
- values randomly generated with equal probability between 0 and 3.

8 How do you classify the type of relation between the two attributes in the scatter plot shown in Figure 2.20?

9 Create the contingency table for the attributes "gender" and "company" in Table 2.1.

10 Given the list of contacts in Table 2.1, calculate the covariance and the correlation between the "maxtemp" and "weight" predictive attributes.

3

Descriptive Multivariate Analysis

In real life, as we saw in our contacts data set, the number of attributes is usually more than two. It can be tens, hundreds or even more. Actually, in biology, for example, data sets with several hundreds, or even thousands, of attributes are very common. When the analysis of a data set explores more than two attributes, it is termed "multivariate analysis". As in univariate and bivariate analysis, frequency tables, statistical measures and plots can be used or adapted for multivariate analysis.

As a result, some of the methods we have outlined for univariate and bivariate analysis in Chapter 2 can be either directly used or modified for use with an arbitrary number of attributes. Naturally, the larger the number of attributes, the more difficult the analysis becomes. It must be observed that all methods used for more than two attributes can also be used for two or one attributes.

In order to illustrate the methods described in this chapter for multivariate analysis, let us add a new attribute to the data set of the excerpt of our private list of contacts from Chapter 2, as illustrated in Table 3.1. Since this table has seven columns, our multivariate analysis can use up to seven attributes. The columns (attributes) are the name of the contact, the maximum temperature registered in the previous month in their home town, their weight, height, how long we have known them (years), and their gender, finishing with our rating of how good their company is.

Next, we will describe simple multivariate methods from the three data analysis approaches seen in the last chapter – frequency, visualization and statistical – and show how they can be applied to this data set.

3.1 Multivariate Frequencies

The multivariate frequency values can be computed independently for each attribute. We can represent the frequency values for each attribute by a matrix, in which the number of rows is the number of values assumed by the

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

Table 3.1 Data set of our private list of contacts with weight and height.

Contact	Maxtemp	Weight	Height	Years	Gender	Company
Andrew	25	77	175	10	M	Good
Bernhard	31	110	195	12	M	Good
Carolina	15	70	172	2	F	Bad
Dennis	20	85	180	16	M	Good
Eve	10	65	168	0	F	Bad
Fred	12	75	173	6	M	Good
Gwyneth	16	75	180	3	F	Bad
Hayden	26	63	165	2	F	Bad
Irene	15	55	158	5	F	Bad
James	21	66	163	14	M	Good
Kevin	30	95	190	1	M	Bad
Lea	13	72	172	11	F	Good
Marcus	8	83	185	3	F	Bad
Nigel	12	115	192	15	M	Good

attribute and the columns are frequency values, just as in Table 2.3 for the attribute "height".

As seen in Chapter 2, depending on the attribute values being discrete or continuous, the attribute values are defined by, respectively, a probability mass function or a probability density function. Thus, different procedures are used for qualitative and quantitative value scales. Nevertheless, for each attribute, the following frequency measures can be taken:

- absolute frequency
- relative frequency
- absolute cumulative frequency
- relative cumulative frequency.

3.2 Multivariate Data Visualization

We have already seen, for univariate and bivariate analysis, it is easier to understand data and experimental results when they are illustrated using visualization techniques. However, most of the plots covered in Chapter 2 cannot be used with more than two attributes.

The good news is that some of the previous plots can be extended to represent a small number of extra attributes. Additionally, new visualization approaches and techniques are continuously being created to deal with new types of data, new approaches to results interpretation and new data

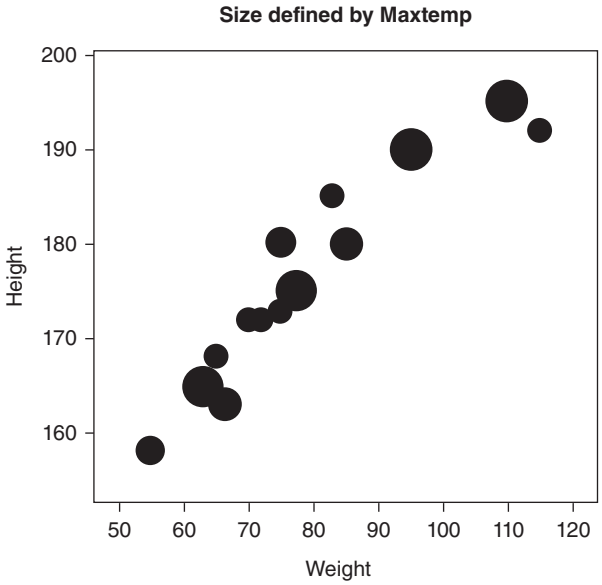


Figure 3.1 Plot of objects with three attributes.

analysis tasks. Depending on the number of attributes, and the need to represent spatial and/or temporal aspects of the data, different plots can be used. This section explores how multivariate data can be visually represented in different ways and the main benefits of each of these alternatives.

When the multivariate data has three attributes, or one can only analyze three attributes from a multivariate data set, the data can still be visualized in a bivariate plot, associating the values of the third attribute with how each data object is represented in the plot. If the third attribute is quantitative, the value can be represented by the size of the object representation in the plot.

Example 3.1 In Figure 3.1, the size of each object in the plot is proportional to the value of the third attribute for this object.

If, on the other hand, the third attribute is qualitative, its value can be represented in the plot by either the color or by the shape of the object. The number of colors or shapes will be the number of values the attribute can assume. In classification tasks, color and shape are usually employed to represent the class labels.

As an example, Figure 3.2 shows two plots where the third attribute is qualitative. On the right-hand size it represents each qualitative value as a different shape. On the left-hand size, it represents each qualitative value by a different color.

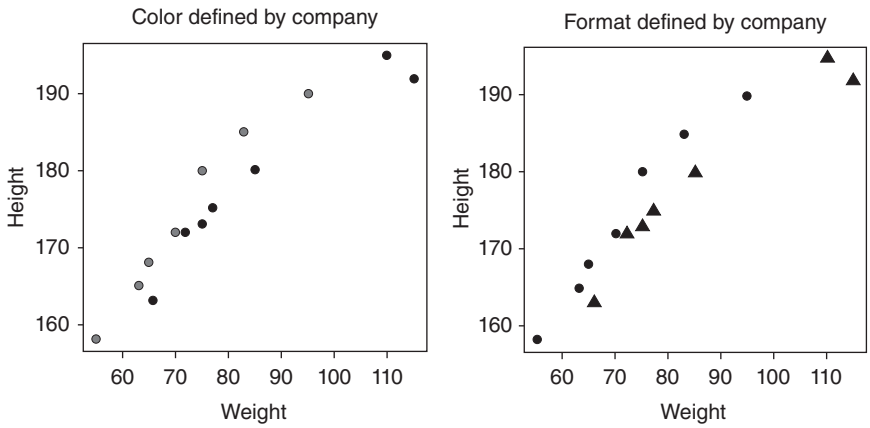


Figure 3.2 Two alternatives for a plot of three attributes, where the the third attribute is qualitative.

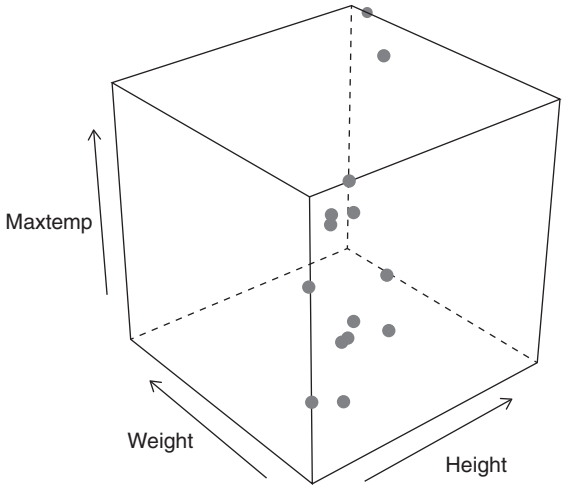


Figure 3.3 Plot for three attributes from the contacts data set.

Another approach to representing three attributes is to use a three-dimensional plot, where each axis is associated with one of the attributes. It makes more sense to use this approach if the three attributes are quantitative, because the values of the corresponding attributes can be presented on each axis assuming an order among them. In Figure 3.3, we show an example using a three-dimensional plot for three quantitative attributes from the conacts data set.

One may ask how can we represent the relationships between more than three attributes. A straightforward approach would be to modify the three-dimensional graph from Figure 3.3, representing the fourth attribute

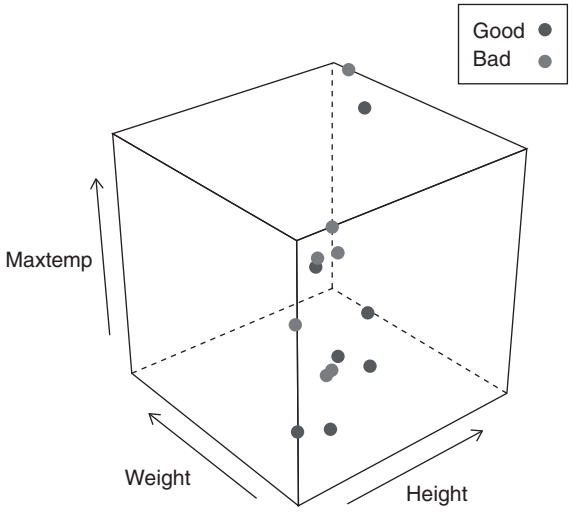


Figure 3.4 Plot for four attributes of the friends data set using color for the forth attribute.

through the size, color or shape of the plotted object. This is shown in in Figure 3.4, using different colors for the attribute.

Although we can also do some tricks to represent more than two predictive attributes in the previous plots by using 3D versions and different formats and colors, not all plots will allow this, or, when they do, the resulting plot can be very confusing. For instance, depending on the plot chosen, color and shape will not preserve the original order and magnitude of the quantitative values, only the different values. Additionally, some qualitative values will not be naturally represented by different object sizes.

Therefore, if we have more than four attributes, a different plot should be used. There are also some plots specifically for more than two quantitative attributes. These usually describe the quantitative attributes of a data set. One of the most popular is the parallel coordinates plot, also known as a profile plot. Each object in a data set is represented by a sequence of lines crossing several equally spaced parallel vertical axes, one for each attribute. The lines for each object are concatenated, and each object is then represented by consecutive straight lines, with up and down slopes. For a particular object, these lines connect with the vertical axis at a position proportional to the value of the attribute associated with the axis. The larger the value of the attribute, the higher the position.

Example 3.2 Figure 3.5 is a parallel coordinate plot for four of our contacts, using three quantitative predictive attributes. The quantitative attributes occupy positions on the vertical axes related to their values. Each object is represented by a sequence of lines crossing the vertical axes at heights that represent the attribute values. It is easy to see the values of the attributes for

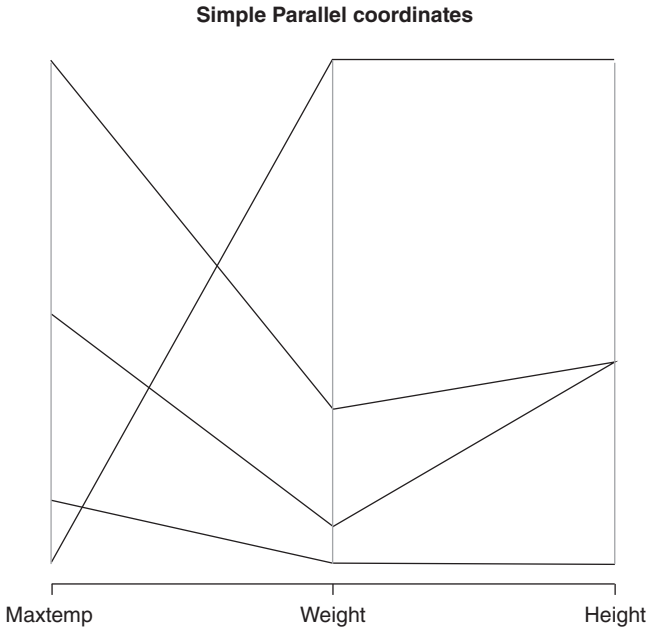


Figure 3.5 Parallel coordinate plot for three attributes.

each object. According to this plot, the attribute values of three of the objects have a similar pattern, which is very different from the profile of the fourth. The plot also shows the minimum and maximum values for each attribute: the highest and the lowest values on each vertical axis.

As we add more objects and more attributes, the lines tend to cross each other, making the analysis more difficult, as shown in Figure 3.6. This figure also shows that qualitative attributes can be represented in parallel coordinate plots. In this case, the qualitative attribute "gender" has been incorporated. Since it has has only two values, "M" and "F", all the objects go to one of two positions on the vertical axis.

Although the plot looks confusing, we can make the analysis of the data in a parallel coordinate plot easier by assigning a color or style to each class and using it for plotting the lines of corresponding objects. Thus, the sequences of lines for the objects from the same class will have the same color or style. On the left-hand side of Figure 3.7, we show a modified version of the previous plot, using a solid line for contacts who are good company and dotted lines for those who are bad company. Even with this modification, the analysis of the information in the plot is still not easy.

The ease of interpretation of these plots depends on the sequence of the attributes used. If the lines from different objects keeping crossing each other, it can be very difficult to extract information from the plot. Changing the order in

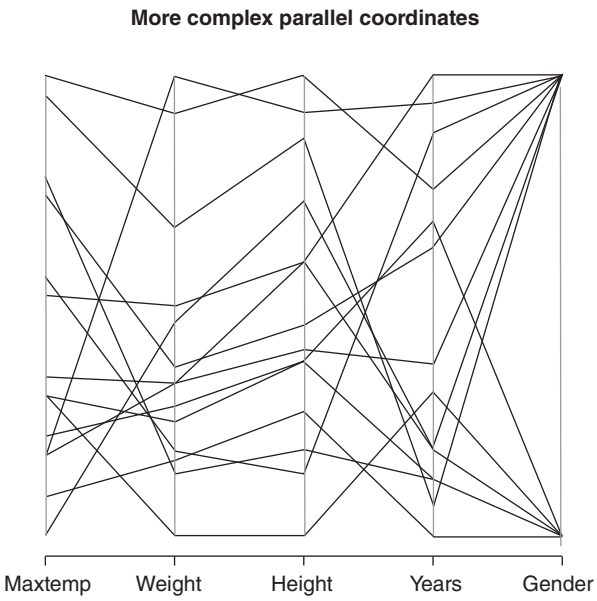


Figure 3.6 Parallel coordinate plot for five attributes.

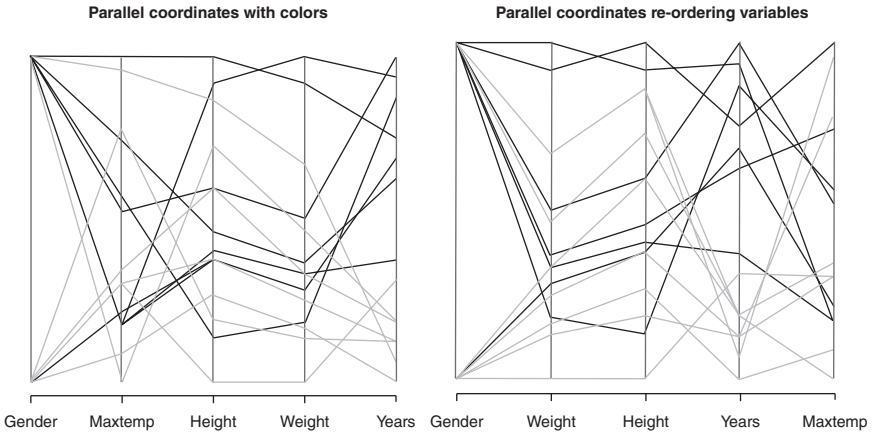


Figure 3.7 Parallel coordinate plots for multiple attributes: left, using a different style of line for contacts who are good and bad company; right, with the order of the attributes changed as well.

which the attributes are plotted can lead to fewer crossings. On the right-hand side of Figure 3.7, the order of the attributes along the horizontal axis has been changed, making it somewhat easier to understand.

Each line in a parallel coordinate plot represents an object. If you do not have many objects and would like to look at each of them individually, you can

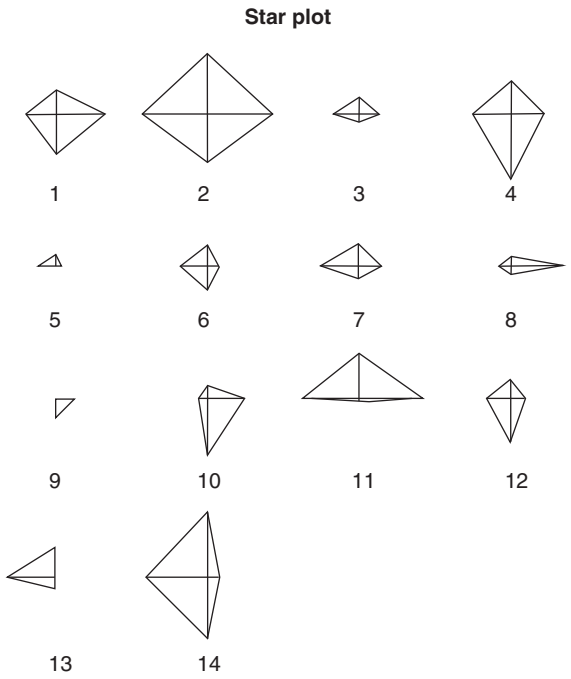


Figure 3.8 Star plot with the value of each attribute for each object in our contacts data set.

use another plot, known as the star plot (it is also known as a spider plot or a radar chart). Figure 3.8 shows star plots for four quantitative attributes (maxtemp, height, weight and years). To avoid the predominance of attributes with larger values in the plot, all attributes have their values normalized to the interval [0.0, 1.0]. When the value of an attribute is close to 0.0, its corresponding star will be too close to the center to be seen. This is clear in the star labeled "Irene". As can be seen in Table 3.1, Irene has the lowest values for some of the attributes.

Qualitative attributes can also be represented in a star plot. However, since they have small numbers of values, points for qualitative attributes will have few variations. We can also label each star in the star plot. Figure 3.9 shows two star plots for five attributes (maxtemp, height, weight, years and gender), mixing quantitative and qualitative attributes. In the star plot on the left, each star is labeled with the contact's name. In the star plot on the right, each object is labeled with its class.

Even with these modifications, using the stars to identify differences among the attribute values of our contacts is still not very intuitive. Taking advantage of the facility of human beings to recognize faces, Herman Chernoff proposed the use of faces to represent objects [11], an approach now referred to as Chernoff faces. Each attribute is associated with a different feature of a human face. If the number of attributes is smaller than the number of features, each attribute

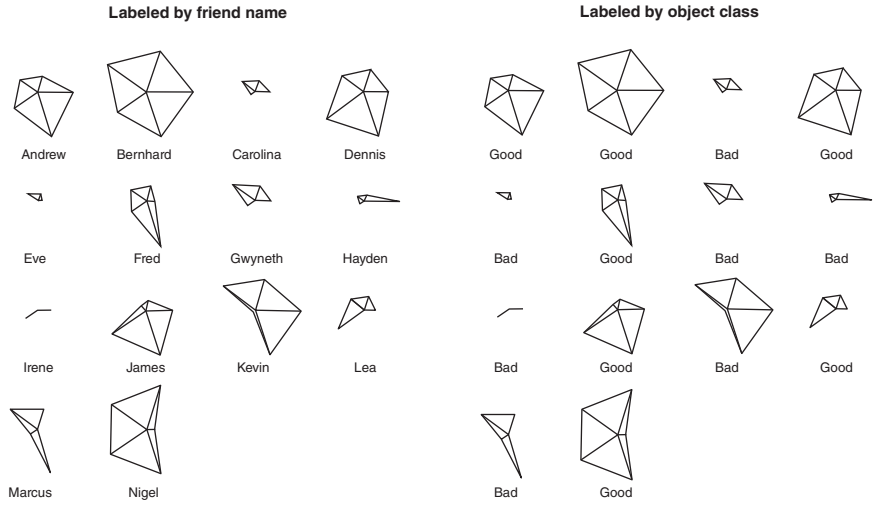


Figure 3.9 Star plot with the value of each attribute for each object in contacts data set.

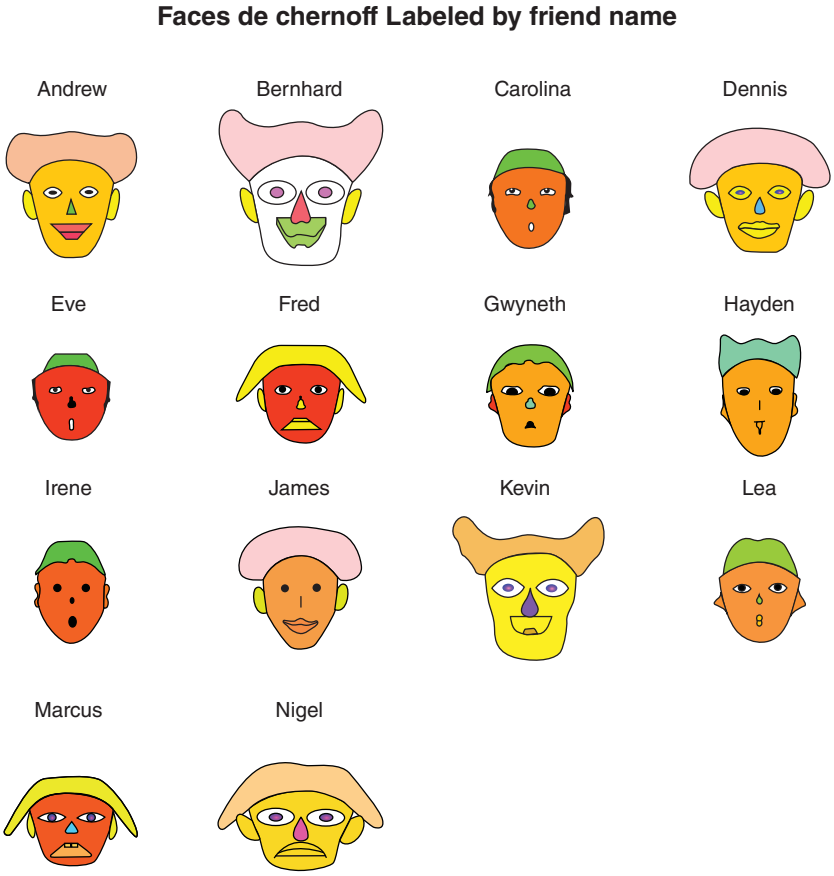


Figure 3.10 Visualization of the objects in our contacts data set using Chernoff faces.

can be associated with a different feature. Figure 3.10 shows how our contacts data set can be represented by Chernoff faces, using as attributes "maxtemp", "height", "weight", "years" and "gender".

Chernoff faces are also useful for clustering, as will be seen in Chapter 5, where they can be used to illustrate the key attributes of each cluster.

There are several other useful plots that allow you to see different aspects of a data set. For example, you can use a streamograph to see how the data distribution changes over time. Data visualization is a very active area of research, and has expanded rapidly in recent decades. New plots to make data analysis simpler and more comprehensive are being continuously created.

A trend in this area is the development and use of interactive visualization plots, where the user interacts with the plot to make the visual information more useful. For example, the user can manipulate the viewing position of a

Table 3.2 Location univariate statistics for quantitative attributes.

Location statistics	Maxtemp	Weight	Height	Years
Min	8.00	55.00	158.00	0.00
Max	31.00	115.00	195.00	16.00
Average	18.14	79.00	176.29	7.14
Mode	15.00	75.00	172.00	2.00
First quartile	12.25	67.00	169.00	2.25
Median or second quartile	15.50	75.00	174.00	5.50
Third quartile	24.00	84.50	183.75	11.75

three-dimensional plot. For further information on data visualization we recommend the reader look at the data visualization literature.

3.3 Multivariate Statistics

At first sight, the extraction of statistical measures from more than two attributes can seem complicated. However multivariate statistics are just a simple extension of the univariate statistics seen in the previous chapter. As we will see in the next sections, some of the statistical measures previously described for univariate and bivariate analysis, such as the mean and standard deviation, can easily be extended to multivariate analysis.

3.3.1 Location Multivariate Statistics

To measure the location statistics when there are several attributes we just measure the location of each attribute. Thus, the multivariate location statistical values can be computed independently for each attribute. These values can be represented by a numeric vector whose number of elements is equal to the number of attributes.

Example 3.3 As an example of the location statistics for more than two attributes, the main location statistics for the four attributes "maxtemp", "height", "weight" and "years" from the data set in Table 3.1 are illustrated by Table 3.2 as a matrix, in which each row has a statistical measure for the four attributes. To use a standard format, all values are represented as real numbers.

A simple plot that we saw in the previous chapter for univariate analysis – the box plot – can also be used to present relevant information about the attributes in a multivariate data set. If the number of attributes is not too large, a set of box plots, one for each attribute, can be used.

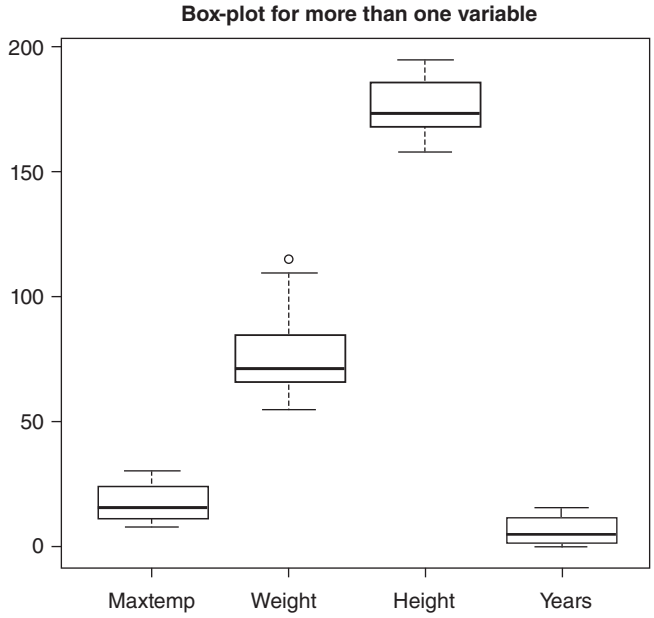


Figure 3.11 Set of box plots, one for each attribute.

Example 3.4 Figure 3.11 shows the equivalent for the quantitative attributes of our excerpt from the contacts data set. It is possible to see how the values of these attributes vary. The box plots show that there is a higher interval of values for the "weight" than for the "years" attribute, and the median of the "weight" attribute is close to the center of values than the median of the "maxtemp" values. It must be noted that it does not make sense to draw a box plot for a qualitative dataset, since, apart from mode, the other statistics only apply to numerical values.

When the number of attributes is large, say more than 10, it is difficult to analyze the information present in all the box plots.

3.3.2 Dispersion Multivariate Statistics

For multivariate statistics, dispersion statistics, such as the amplitude, interquartile range, mean absolute deviation and standard deviation, as seen in Chapter 2, can be independently defined for each attribute.

Example 3.5 Table 3.3 shows an example of multivariate dispersion statistics for the attributes "maxtemp", "height", "weight" and "years" from the data set in Table 3.1. As in the example for location multivariate statistics, the dispersion

Table 3.3 Dispersion univariate statistics for quantitative attributes.

Dispersion statistics	Maxtemp	Weight	Height	Years
Amplitude	23.00	60.00	37.00	16.00
Interquartile range	11.75	17.50	14.75	9.50
$\overline{MAD}$	7.41	14.09	11.12	6.67
s	7.45	17.38	11.25	5.66

statistics can be shown in a matrix, each of the four rows representing a statistical measure for the four attributes.

The previous described statistics measure the dispersion of each attribute independently. We can also measure how the the values of a attribute vary with those of another attribute. For example, if the value for attribute A increases from one person to another, does the attribute B also increase? If so, we say that they have similar variation, and so one is directly proportional to the other. If the variation is in the opposite direction – when attribute A increases, attribute B decreases – we say that the two attributes have an opposite variation, and they are inversely proportional. If neither of these situations is observed, there is probably no relationship between the two attributes.

The relationship between two attributes is evaluated using the covariance or correlation, as discussed in Section 2.3.1. The covariance measure for all pairs in a set of attributes can be represented using a covariance matrix. In these matrices, the attributes are listed in the rows and in the columns, in the same order.

Example 3.6 As an example, in Table 3.4, we show the covariance matrix for four attributes from our contacts data set. Each element shows the covariance of a pair of attributes, giving a good picture of the dispersion in a data set. The main diagonal of the matrix shows the variance of each attribute. This matrix is also symmetric, in the sense that the values above the main diagonal are the same as the values below. This shows that the order of the attributes in the calculation of the covariance is irrelevant. It is also possible to see that weight and height have high covariance.

Example 3.7 In Table 3.5, we show how each pair of attributes is correlated. We use the four quantitative attributes from our contacts data set. In this Pearson correlation matrix, each element shows the Pearson correlation for a pair of attributes. The values on the main diagonal of the matrix are all equal to 1, meaning that each attribute is perfectly correlated with itself.

Table 3.4 Covariance matrix for quantitative attributes.

	Maxtemp	Weight	Height	Years
Maxtemp	55.52	34.46	20.19	5.82
Weight	34.46	302.15	184.62	42.39
Height	20.19	184.62	126.53	14.03
Years	5.82	42.39	14.03	31.98

Table 3.5 Pearson correlation matrix for quantitative attributes.

	Maxtemp	Weight	Height	Years
Maxtemp	1.00	0.27	0.24	0.14
Weight	0.27	1.00	0.94	0.43
Height	0.24	0.94	1.00	0.22
Years	0.14	0.43	0.22	1.00

In Chapter 2 we saw how to plot the linear correlation between two attributes. We can use a similar plot to illustrate the correlation of all pairs from a set of attributes using a matrix of several scatter plots, with one scatter plot for each pair of attributes. Like correlations, scatter plots can be applied to an arbitrary number of pairs of ordinal or quantitative attributes to create a scatter plot matrix, which is also known as a draftsman's display.

Example 3.8 Figure 3.12 shows the scatter plots for all pair of attributes in the contacts data set in Table 2.1, with gender as a target attribute: each object is labelled with its class – in our case using a different shape. The plots show how the predictive attributes correlate for different classes.

Note that the same information is presented above and below the main diagonal, since the correlation between attributes x and y is the same as the correlation between y and x. As well as the position of each object being set according to the values of two attributes, the plot presents, on the vertical and horizontal axes, the values of each attribute. The first row of the matrix shows the Pearson correlation between the attribute "maxtemp" and the three other attributes: "weight", "height" and "years". Similarly, the second row shows the Pearson correlation between the attribute "weight" and the attributes "maxtemp", "height" and "years".

It is possible to see that the predictive attributes "height" and "weight" have a positive linear correlation, since when the value of one of them increases, the value of the other also increases.

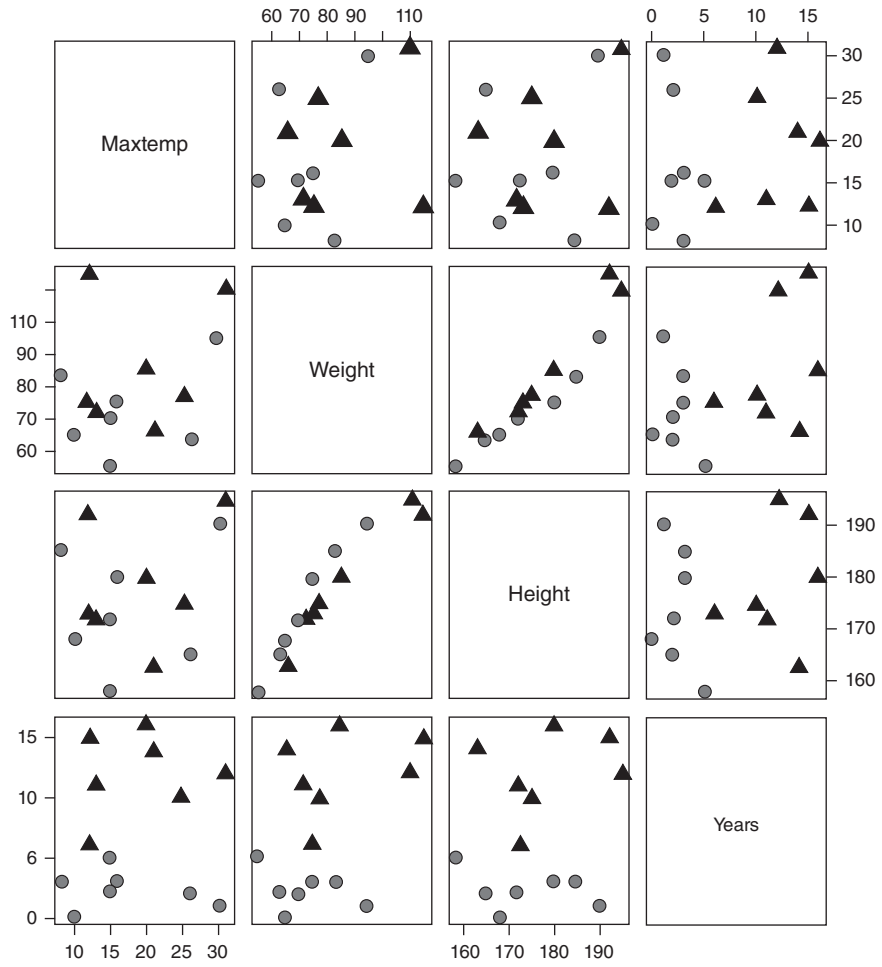


Figure 3.12 Matrix of scatter plots for quantitative attributes.

We have already noted that the same information is presented above and below the main diagonal. There is a version of a scatter plot matrix that takes advantage of this redundancy to show both the scatter plots for each pair of attributes and the corresponding correlation value, for example, the Pearson correlation coefficient. An example of this scatter plot can be seen in Figure 3.13

We can use a simpler plot to give a summary of the information in the scatter plot matrix. The linear correlation matrix can be plotted in a correlogram, as shown in Figure 3.14. In this figure, the darker the square associated with two attributes, the more correlated they are. Thus there is a high correlation between the attributes "height" and "weight", and low correlation for the other

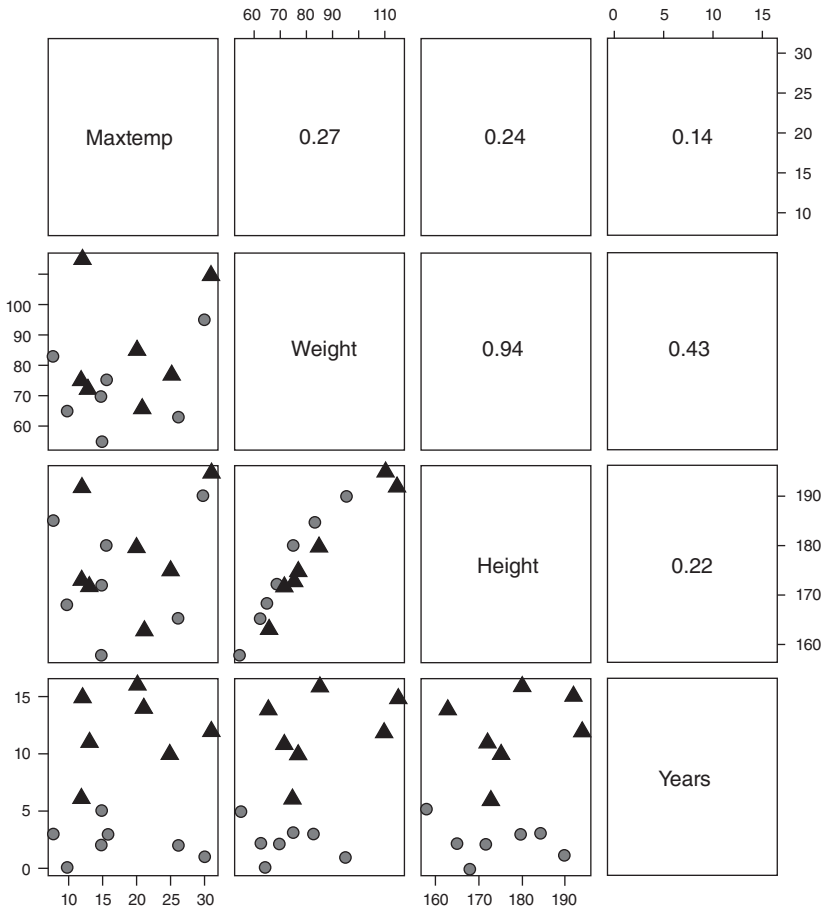


Figure 3.13 Matrix of scatter plots for quantitative attributes with additional Pearson correlation values.

pairs of attributes. Different colors can be used to indicate positive and negative correlations.

As in the scatterplot matrix, there is a symmetry below and above the main diagonal in the correlogram. Thus, correlation values can be plotted instead of the colored boxes above or below the main diagonal.

Another common plot for multivariate data is the heatmap, which represents a table of values by a matrix of boxes, each value corresponding to one box. Each row (or column) of the matrix is associated with a color. Different values in the row (or column) are represented by different tones of the row (or column) color. Heatmaps have been widely used to analyze gene expression in bioinformatics.

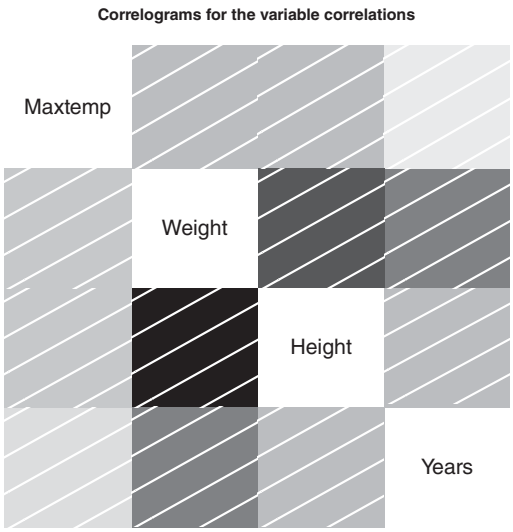


Figure 3.14 Correlograms for Pearson correlation between the attributes "maxtemp", "weight", "height" and "years".

Example 3.9 As an example of its use, Figure 3.15 illustrates a heatmap for the short version of our contacts data set. Each of the four positions in the vertical axis is associated with one object. Each position in the horizontal axis is associated with a different attribute. One color is associated with each attribute. In this example, for a given object, the darker the color shade, the smaller the value of the attribute for the object.

The objects at the top and left-hand side of the heatmap are called dendrograms, and will be discussed in detail in Chapter 5. These represent groupings of the attributes (at the top) and of the objects (on the left) according to their similarity.

We previously mentioned that most plots for multivariate analysis were developed for use with quantitative data. Since the conversion of qualitative ordinal attributes is straightforward, these can also be easily used in the plots. With the increasing importance of the analysis of nominal qualitative data, new plots have been created. One example is the mosaic plot, described in the previous chapter, which can illustrate the frequency of combinations of up to three quantitative attributes. To do this, quantitative values, such frequency, are extracted from the qualitative attribute.

The examples of visualization plots to illustrate information in a data set given so far have used a small number of attributes. However, as mentioned at the beginning of this chapter, many real problems have tens, hundreds, or even thousands of attributes. Although statistical measures can be extracted

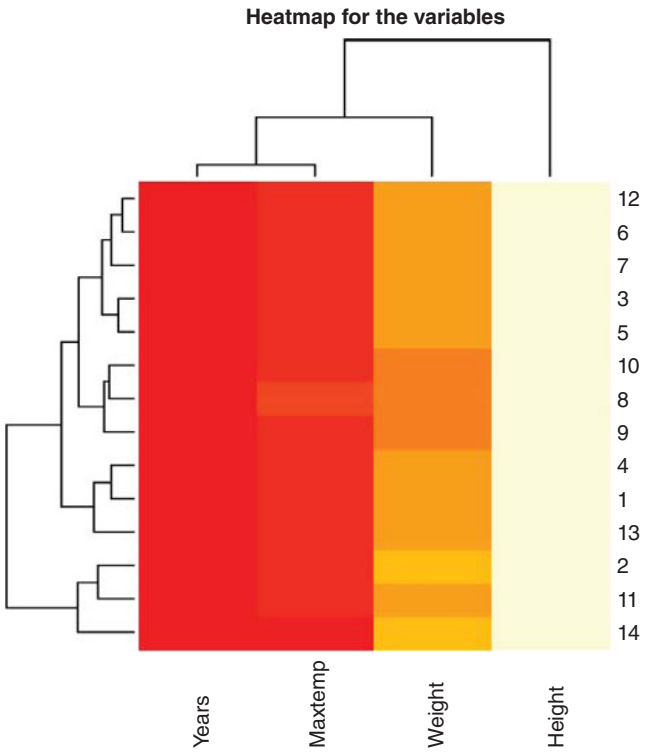


Figure 3.15 Heatmap for the short version of the contacts data set.

from high-dimensional data sets, the user will either receive a brief summary of the information present in the data or will not be able to analyze it or will get swamped by the large amount of information.

3.4 Infographics and Word Clouds

3.4.1 Infographics

Currently, it is common to highlight important facts by using infographics. It is important to understand the difference between data visualization and infographics. Although both techniques transform data into an image, the infographic approach is subjective, is produced manually and is customized for a particular data set. Data visualization, on the other hand, is objective, automatically produced and can be applied to many data sets. We have seen several examples of data visualization in this chapter. An example of an infographic can be seen in Figure 3.16.

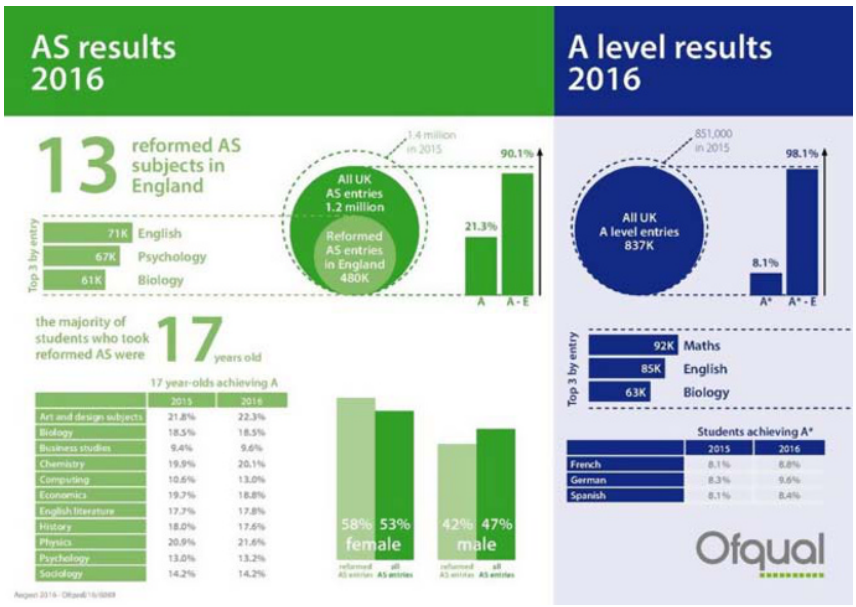


Figure 3.16 Infographic of the level of qualifications in England (Contains public sector information licensed under the Open Government Licence v3.0.).

3.4.2 Word Clouds

A visualization tool frequently used in text mining to illustrate text data is the word cloud, which represents how often each word appears in a given text. The higher the frequency of a word in the text, the larger its size in a word cloud. Since articles and prepositions occur very often in a text, and numbers are not text, these are usually removed before the word cloud tool is applied to a text. Another text process operation, stemming, which replaces a word in a text by its stem is also applied to the text before the word cloud tool is used. Figure 3.17 shows the result of applying a word cloud tool to the last paragraph. It can be seen that the words whose stem appear more often in the previous text are represented in a larger font size: this is the case for the words "text", "word" and "cloud".

3.5 Final Remarks

This chapter has extended univariate and bivariate analysis, as covered in the previous chapter, to the analysis of more than two attributes. Frequency measures, data visualization techniques and statistical measures for multivariate analysis have been described.



Figure 3.17 Text visualization using a word cloud.

A large number of other powerful methods to analyze multivariate data exist. However, these methods are out of the scope of this book: only the most frequently used techniques have been covered. The other methods are usually presented in more advanced books on multivariate analysis, which are only suitable for those with a subject more advisable for those with knowledge on introductory statistics [10, 12].

The next chapter discusses the importance of data set quality and how it can affect the following steps in analytics, presenting the main problems found in low quality data, techniques to deal with them, operations to modify the type, scale and distribution of data, the relationship between data dimensionality and data modeling, and how to deal with high-dimensional data.

3.6 Exercises

1 Why can some of the techniques used in univariate and bivariate analysis not be used for multivariate analysis?

2 What is the limit of the information provided by multivariate plots?

3 Suppose instead of Chernoff faces you use house drawings to represent objects. Describe five features from the drawing you would use to represent the objects.

4 What is the main problem of parallel coordinates and how can this problem be minimized?

5 Describe the absolute and relative frequencies and respective cumulative frequencies for three quantitative attributes from Table 3.1.

6 How do you classify the type of relations between the set of attributes with the scatter plots shown in Figure 3.12?

7 How are the matrix of a scatter plot and a correlogram related?

8 What can be seen in a heatmap?

9 Why is it difficult to use qualitative attributes in a scatter plot?

10 In what situations is it better to use infographics to represent information present in a data set?

4

Data Quality and Preprocessing

Depending on the type of data scale, different data quality and preprocessing techniques can be used. We will now describe data quality issues, and follow this with sections on converting to different scale types and to different scales. We also talk about data transformation and dimensionality reduction.

4.1 Data Quality

The quality of the models, charts and studies in data analytics depends on the quality of the data being used. The nature of the application domain, human error, the integration of different data sets (say, from different devices), and the methodology used to collect data can generate data sets that are noisy, inconsistent, or contain duplicate records.

Today, even though there is a large number of robust descriptive and predictive algorithms available to deal with noisy, incomplete, inconsistent or redundant data, an increasing number of real applications have their findings harmed by poor-quality data. In data sets collected directly from storage systems (actual data), it is estimated that noise can represent 5% or more of the total data set [13]. When these data are used by algorithms that learn from data – ML algorithms – the analysis problem can look more complex than it really is if there is no data pre-processing. This increases the time required for the induction of assumptions or models and resulting in models that do not capture the true patterns present in the data set.

The elimination or even just the reduction of these problems can lead to an improvement in the quality of knowledge extracted by data analysis processes. Data quality is important and can be affected by internal and external factors.

- Internal factors can be linked to the measurement process and the collection of information through the attributes chosen.

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

- External factors are related to faults in the data collection process, and can involve the absence of values for some attributes and the voluntary or involuntary addition of errors to others.

The main problems affecting data quality are now briefly described. They are associated with missing values, and with inconsistency, redundancy, noise and outliers in a data set.

4.1.1 Missing Values

In real-life applications, it is common that some of predictive attribute values for some of the records may be missing in the data set. There are several causes of missing values, among them:

- attributes values only recorded some time after the start of data collection, so that early records do not have a value
- the value of an attribute being unknown at time of collection
- distraction, misunderstanding or refusal at time of collection
- attribute not required for particular objects
- non-existence of a value
- fault in the data collection device
- cost or difficulty of assigning a class label to an object in classification problems.

Since many data analysis techniques were not designed to deal with a data set with missing values, the data set must be pre-processed. Several alternatives approaches have been proposed in the literature, including:

- *Ignore missing values:*
  - Use for each object only the attributes with values, without paying attention to missing values. This does not require any change in the modeling algorithm used, but the distance function should ignore the values of attributes with at least one missing value;
  - Modify a learning algorithm to allow it to accept and work with missing values.
- *Remove objects:* Use only those objects with values for all attributes.
- *Make estimates:* Fill the missing values with estimates based on values for this attribute in the other objects.

The simplest alternative is just to remove objects with missing values in a large number of attributes.Objects should not be discarded when there is a risk of losing important data. Another simple alternative is to create a new, related, attribute, with Boolean values: the value will be true if there was a missing value in the related attribute, and false otherwise.

The filling of missing data is the most common approach. The simplest approach here is to create a new value for the attribute to indicate that the

correct value was missing. This alternative is mainly used for qualitative attributes. The most efficient alternative is to estimate a value. Several methods can be used:

- Fill with a location value: the mean or median for quantitative and ordinal attributes, and the mode for nominal values. The mean is just the average of the values and the mode is the quantitative value that appears most often in the attribute. The median is the value that is greater than half of the values and lower than the remaining half.
- For classification tasks, we can use the previous method, namely using only instances from the same class to calculate the location statistic. In other words, if we intend to fill the value of attribute at of instance i that belongs to class $C1$, we will use only instances from the class $C1$ that do not have missing values in the at attribute.
- A learning algorithm can be used to as a prediction model giving a replacement value for one that is missing in a particular attribute. The learning algorithm uses all other attributes as predictors and the one to be filled as the target.

Of these methods, the first method is the simplest and has the lowest processing cost. The second method has a slightly higher cost, but gives a better estimate of the true value. The third method can further improve the estimate, with a higher cost.

Example 4.1 As an example of how to deal with missing values, let us consider the data set in Table 4.1. Suppose that, due to a data transmission problem, part of our contact data sent to a colleague was missing. Table 4.1 shows how missing values in the data set can be filled, using the mode for qualitative values

Table 4.1 Filling of missing values.

Data with missing values				Data without missing values			
Food	**Age**	**Distance**	**Company**	**Food**	**Age**	**Distance**	**Company**
Chinese	51	Close	Good	Chinese	51	Close	Good
			Good	Chinese	53	Close	Good
Italian	82		Good	Italian	82	Close	Good
Burgers	23	Far	Bad	Burgers	23	Far	Bad
Chinese	46		Good	Chinese	46	Close	Good
Chinese			Bad	Chinese	31	Far	Bad
Burgers		Very close	Good	Burgers	53	Very far	Good
Chinese	38	Close	Bad	Chinese	38	Close	Bad
Italian	31	Far	Good	Italian	31	Far	Good

and rounded averages for quantitative values, considering the objects from the same class: those with the same label for the target attribute "Company".

It should be noted that the absence of a value in an instance can be important information about that instance. There are also situations where the attribute *must* have a missing value, say the apartment number for a house address. In this case, instead of being missing, the value is actually non-existent. It is difficult to automatically deal with non-existent values. A possible approach is to create another related attribute to indicate when the value in the other attribute is non-existent.

4.1.2 Redundant Data

While missing values are a lack of data, redundant data is the excess of it. Redundant objects are those that do not bring any new information to a data set. Thus, they are irrelevant data. They are objects very similar to other objects.

Redundancy occurs mainly in the whole set of attributes. Redundant data could be due to small mistakes or noise in the data collection, such as the same addresses for people whose names differ by just a single letter.

In the extreme, redundant data can be duplicate data. Deduplication is a preprocessing technique whose goal is to identify and remove copies of objects in a data set, as shown in Table 4.2.

The presence of duplicate objects will make the ML technique overweight such objects than others in the data set.

Table 4.2 Removal of redundant objects.

Data with redundant objects				Data without redundant objects			
Food	**Age**	**Distance**	**Company**	**Food**	**Age**	**Distance**	**Company**
Chinese	51	Close	Good	Chinese	51	Close	Good
Italian	43	Very close	Good	Italian	43	Very close	Good
Italian	43	Very close	Good	—	—	—	—
Italian	82	Close	Good	Italian	82	Close	Good
Burgers	23	Far	Bad	Italian	82	Close	Good
Chinese	46	Very far	Good	Chinese	46	Very far	Good
Chinese	29	Too far	Bad	Chinese	29	Too far	Bad
Chinese	29	Too far	Bad	—	—	—	—
Burgers	42	Very far	Good	Burgers	42	Very far	Good
Chinese	38	Close	Bad	Chinese	38	Close	Bad
Italian	31	Far	Good	Italian	31	Far	Good

It must be mentioned that redundancy can also occur in the predictive attributes, when the values for a predictive attribute can be derived from the values from other predictive attributes.

4.1.3 Inconsistent Data

A data set can also have inconsistent values. The presence of inconsistent values in a data set usually reduces the quality of the model induced by ML algorithms. Inconsistent values can be found in the predictive and/or target attributes.

An example of an inconsistent value in a predictive attribute is a zip code that does not match the city name. This inconsistency can be due to a mistake or a fraud.

In predictive problems, inconsistent values in the target attribute can lead to ambiguity, since it allows two objects with the same predictive attribute values to share different target values. Inconsistencies in target attributes can be due to labeling errors.

Some inconsistencies are easily detected. For example, some attribute values might have a known relationship to others, say that the value of attribute A is larger than the value of attribute B. Other attributes might only be allowed to have a positive value. Inconsistencies in these cases are easily identified.

Example 4.2 To show an example of inconsistent values, we will go back to the data set in Table 2.1. The left-hand side of Table 4.3 shows a new version of the data set with inconsistent values for some of the data. These values are highlighted in the table.

Table 4.3 Data set of our private list of contacts with weight and height.

Friend	Maxtemp (°C)	Weight (kg)	Height (cm)	Gender	Company
Andrew	25	77	175	M	Good
Bernhard	31	**1100**	195	M	Good
Carolina	15	70	172	F	Bad
Dennis	20	**45**	**210**	M	Good
Eve	10	65	168	F	Bad
Fred	12	75	173	M	Good
Gwyneth	16	75	**10**	F	Bad
Hayden	26	63	165	F	Bad
Irene	15	55	158	F	Bad
James	21	66	163	M	Good
Kevin	**300**	95	190	M	Bad
Lea	13	72	**1072**	F	Good
Marcus	8	83	185	F	Bad
Nigel	12	115	192	M	Good

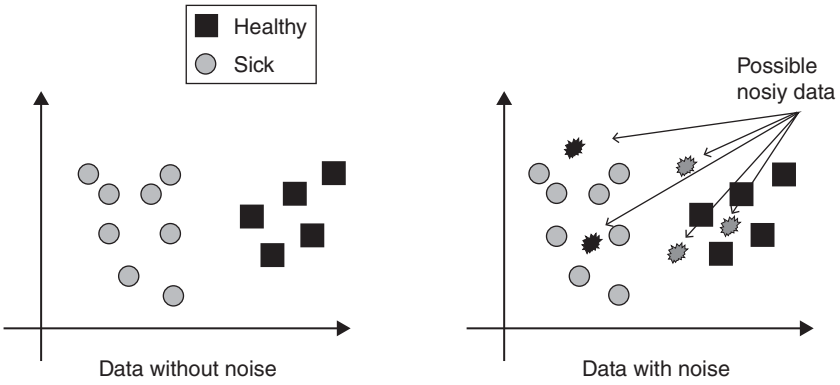


Figure 4.1 Data set with and without noise.

A good policy to deal with inconsistent values in the predictive attribute is to treat them as missing values. Inconsistencies in predictive and in target attributes can also be caused by noise.

4.1.4 Noisy Data

There are several definitions of noise in the literature. A simple definition is that noisy data are data that do not meet the set of standards expected for them. Noise can be caused by incorrect or distorted measurements, human error or even contamination of the samples.

Noise detection can be performed by adaptation of classification algorithms or by the use of noise filters for data preprocessing. It is usually performed with noise filters, which can look for noise in either the predictive attributes or in the target attribute.

Example 4.3 Figure 4.1 illustrates an example of a data set with and without noise. The noise can be present in either the predictive or in the label attributes.

Since noise detection in predictive attributes is more complex and can be affected by relationship between predictive attributes, most filters have been developed for target attributes.

Many label noise filters are based on the k-NN algorithm. They detect a noisy object by looking at the label of the k most similar objects. They assume that an object is likely to be noisy if its class label is different from the class label of the closest objects.

It is important to observe that it is not usually possible to be sure that an object is noisy. Apparently noisy data can be correct objects that do not follow the current standard.

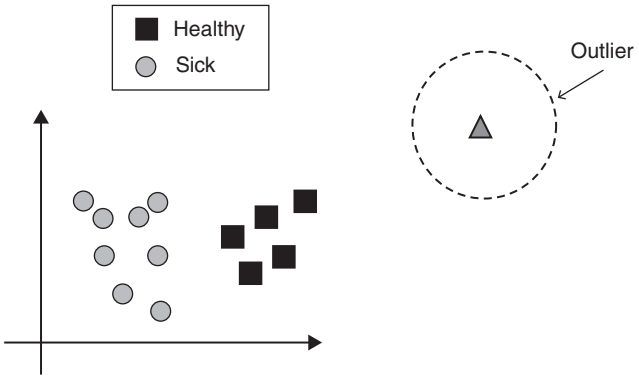


Figure 4.2 Data set with outliers.

4.1.5 Outliers

In a data set, outliers are anomalous values or objects. They can also be defined as objects whose values for one or more predictive attributes are very different from the values found in the same predictive attributes of other objects.

In contrast to noisy data points, outliers can be legitimate values. There are several data analysis applications whose main goal is to find outliers in a data set. Particularly in anomaly detection tasks, the presence of outliers can indicate the presence of noise.

Example 4.4 Figure 4.2 illustrates an example of a dataset with the presence of outliers.

A simple yet effective method to detect outliers in quantitative attributes is based in the interquartile range. Let $Q1$ and $Q3$ be the first quartile and the third quartile, respectively. The interquartile range is given by $IQ = Q3 - Q1$. Values below $Q1 - 1.5 \times IQ$ or above $Q3 + 1.5 \times IQ$ are considered too far away from central values to be reasonable. Figure 4.3 shows an example.

4.2 Converting to a Different Scale Type

As previously mentioned, some ML algorithms can use only data of a particular scale type. The good news is that it is possible to convert data from a qualitative scale to a quantitative scale, and vice versa.

To better illustrate such conversions, let us consider another data set related with to work colleagues or classmates, showing their favorite food, age, how far from us they live and if they are good or bad company. Table 4.4 illustrates this data set. Since we will not convert names, we do not show this column in the table.

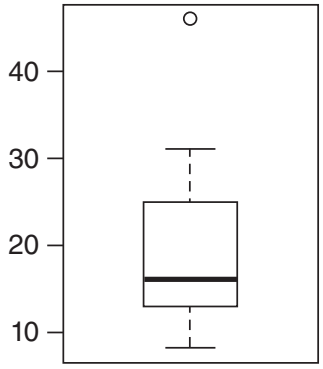


Figure 4.3 Outlier detection based on the interquartile range distance.

Table 4.4 Food preferences of our colleagues.

Food	Age	Distance	Company
Chinese	51	Close	Good
Italian	43	Very close	Good
Italian	82	Close	Good
Burgers	23	Far	Bad
Chinese	46	Very far	Good
Chinese	29	Too far	Bad
Burgers	42	Very far	Good
Chinese	38	Close	Bad
Italian	31	Far	Good

We will now show how conversions can be applied to this data set. Next, the main conversion procedures used are briefly described.

4.2.1 Converting Nominal to Relative

Since the nominal scale does not assume an order between its values, to keep this information, nominal values should be converted to relative or binary values.

The most common conversion is called "1-of-n", also known as canonical or one-attribute- per-value conversion, which transforms n values of a nominal attribute into n binary attributes. A binary attribute has only two values, 0 or 1.

Example 4.5 Table 4.5 illustrates an example of this conversion for an attribute associated with color, whose possible nominal values are *Green*, *Yellow* and *Blue*.

Table 4.5 Conversion from nominal scale to relative scale.

Nominal	Relative
Green	001
Yellow	010
Blue	100

Table 4.6 Conversion from the nominal scale to binary values.

Original data				Converted data					
Food	Age	Distance	Company	F1	F2	F3	Age	Distance	Company
Chinese	51	Close	Good	0	0	1	51	2	1
Italian	43	Very close	Good	0	1	0	43	1	1
Italian	82	Close	Good	0	1	0	82	2	1
Burgers	23	Far	Bad	1	0	0	23	3	0
Chinese	46	Very far	Good	0	0	1	46	4	1
Chinese	29	Too far	Bad	0	0	1	29	5	0
Burgers	42	Very far	Good	1	0	0	42	4	1
Chinese	38	Close	Bad	0	0	1	38	2	0
Italian	31	Far	Good	0	1	0	31	3	1

Table 4.6 shows how another data set concerning out contacts can be converted, turning all predictive attributes that are qualitative, apart from the name, to quantitative ones.

It is important to observe that each resulting sequence of n 0 and 1 values – binary values – is not just one predictive attribute, but n predictive attributes, one for each possible nominal value. Thus, we transform 1 predictive attribute with n nominal values into n numeric predictive attributes, each with the value 1, meaning the presence of the corresponding nominal value, or 0, meaning the absence of the corresponding nominal value.

However, when you use canonical conversion from nominal to numeric scales, we increase the number of predictive attributes and, if we have a large number of nominal values, we end up with a large number of predictive attributes with the value 0. Data sets with a large number of 0 values are referred to as "sparse" data sets. Some analytical techniques have difficulty dealing with sparse data sets.

Table 4.7 Conversion from the nominal scale to the relative scale.

Original DNA					Converted DNA				
A	A	T	C	A	0001	0001	0100	0010	0001
T	T	A	C	G	0100	0100	0001	0010	1000
G	C	A	A	C	1000	0010	0001	0001	0010

Table 4.8 Conversion from the nominal scale to the relative scale.

Original DNA					Converted DNA																
A	A	T	C	A	0	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1
T	T	A	C	G	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	1	0	0	0
G	C	A	A	C	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0

This is notably the case for biological sequences. DNA, RNA and amino acid sequences are long strings with hundreds or thousands of single letters. Each position in the sequence can be seen as a predictive attribute. For DNA sequences, each position can have four possible values: the letters *A*, *C*, *T* and *G*. RNA sequences also have four possible values for each position, but with *U* instead of *T*. For amino acid sequences, each position can have 20 possible values.

Example 4.6 Table 4.7 shows a simple example for three DNA sequences with fivenucleotides each: AATCA, TTACG and GCAAC. We encode the nucleotides A, C, T and G by 0001, 0010, 0100 and 1000, respectively.

It must be recalled that if we use the 1-of-n encoding, a binary number with *n* values corresponds to *n* predictive attributes, one for each value. For DNA sequences, we would multiply the number of predictive attributes by four. For amino acid sequences, we would multiply by 20. This encoding, therefore, results in even sparser data sets. The final number of predictive attributes is illustrated in Table 4.8.

There are some alternative ways to minimize this increase in the number of predictive attributes and the resulting sparse data set. One of these alternatives is to convert each nominal value to the frequency with which the value occurs in the predictive attribute. For DNA sequences, this results in four values, one for the frequency of each nucleotide. This alternative is often used to convert DNA and amino acid sequences to quantitative values. Doing so for the same DNA

Table 4.9 Conversion from the nominal scale to the relative scale.

Original DNA	Converted DNA			
AATCA	0.6	0.2	0.2	0.0
TTACG	0.2	0.2	0.4	0.2
GCAAC	0.4	0.4	0.0	0.2

Table 4.10 Conversion from the ordinal scale to the relative or absolute scale.

Nominal	Natural number	Gray code	Thermometer code
Small	0	00	000
Medium	1	01	001
Large	2	11	011
Very large	3	10	111

sequences, would give the values shown in Table 4.9, for the DNA sequences AATCA, TTACG and GCAAC.

4.2.2 Converting Ordinal to Relative or Absolute

For ordinal values, the conversion is more intuitive, since we can convert to natural numbers, starting with the value 0 for the smallest value and, for each subsequent value, adding 1 to the previous value.

As previously mentioned, some algorithms may work only with binary values. If we want to convert ordinal values to binary values, we can use the gray code, which keeps the distance between two consecutive values as a different value in one of the binary values. In this case we also change one attribute to n attributes, but n can be smaller than the number of values. Another binary code, called the thermometer code, starts with a binary vector with only 0 values and substitutes one 0 value by 1, from right to left, as the ordinal value increases. In this case, n is equal to the number of ordinal values minus 1.

Table 4.10 illustrates the conversion of four values of an ordinal attribute: small, medium, large and very large, using three conversions: to natural numbers, to gray code, and to thermometer code.

If the quantitative natural value starts with the value 0, the conversion is to an absolute scale. But if you want to use the values as relative, you can start the natural values with values larger than 0.

For the gray code, it is possible to see that each two consecutive ordinal values differ by the value of 1 binary value – bit – only. Any combination of binary values with this property can be used.

4.2.3 Converting Relative or Absolute to Ordinal or Nominal

Quantitative values can be converted to nominal or ordinal values. This process is called "discretization" and, depending whether we want to keep the order between the values, will be referred to as "nominal" or "ordinal" discretization.

Discretization is necessary when the learning algorithm can deal only with qualitative values or when one want to reduce the number of quantitative values.

Discretization has two steps. The first step is the definition of the number of qualitative values, which is usually defined by the data analyst. This number of qualitative values is called the number of "bins", where each bin is associated with an interval of quantitative values. Then, given the number of bins, the next step is to define the interval of values to be associated with each bin. This association is usually done with an algorithm. There are two alternatives for the association: by width or by frequency. In the association by width, the intervals will have the same range: the same difference between the largest and smallest values. In the association by frequency, each interval will have the same number of values. It must be noted that a quantitative scale does not necessarily have all possible values.

Table 4.11 illustrates the conversion of nine quantitative values (2, 3, 5, 7, 10, 15, 16, 19, 20) into three bins, whose nominal values are *A*, *B* and *C*, using association by width and association by frequency.

Table 4.11 Conversion from the ordinal scale to the relative scale.

Quantiative	Conversion by width	Conversion by frequency
2	A	A
3	A	A
5	A	A
7	A	B
10	B	B
15	B	B
16	C	C
19	C	C
20	C	C

The chosen intervals for the association by width were [(2, 8), (9, 15), (16, 22)]. For the association by frequency, the chosen intervals were [(2, 5), (7, 15), (16, 20)]. There are usually different alternatives to define the limits of the association by width and by frequency. Note that the upper and lower limits of the interval do not need to be in the data set and that some values, which we assume will never appear, can be left out of the intervals.

4.3 Converting to a Different Scale

Converting data in a scale to another scale of the same type is necessary, in several situations, such as when using distance measures (a subject discussed in Chapter 5). This kind of conversion is typically done in order to have different attributes expressed on the same scale; a process known as "normalization".

We must always be aware that the results can be different depending on the measure in which the values of a given attribute are expressed.

Example 4.7 Let us see the following example: three friends have age and education as follows: Bernhard (age 43, education 2.0), Gwyneth (age 38, education 4.2) and James (age 42 and education 4.1). The ages are expressed in years. Calculating the Euclidean distance between these friends, we obtain the the values in Table 4.12, where we abbreviate Bernhard by B, Gwyneth by G and James by J.

The most similar friends are Bernhard and James, while the most dissimilar are Bernhard and Gwyneth. Let us do the same calculation measuring the ages in decades: 4.3, 3.8 and 4.2 for Bernhard, Gwyneth and James respectively (see Table 4.13).

Now the most similar friends are Gwyneth and James instead of Bernhard and James. Can you understand these results? Why is this happening? If we use years instead of decades to measure ages we obtain larger numbers: much larger than the ones used to measure the educational level. As consequence, the values of the age will be much more influential than those of the educational level in the calculation of the Euclidean distance. A practical approach to avoiding this problem is through data normalization. This is a typical pre-processing task that should occur during the execution of the data preparation phase of the CRISP-DM methodology (see Section 1.7.3) when using distance measures.

The normalization is carried out for each attribute individually. There are two ways to normalize the data: by standardization and by min–max rescaling.

The simplest alternative, min–max rescaling, converts numerical values to values in a given interval. For example, to convert a set of values to values in the interval [0.0, ..., 1.0], you simply subtract the smallest value from all the values

Table 4.12 Euclidean distances of ages expressed in years.

Age in years	B–G	B–J	G–J
Euclidean distance	5.46	2.33	4.00

Table 4.13 Euclidean distance with age expressed in decades.

Age in decades	B–G	B–J	G–J
Euclidean distance	2.26	2.10	0.41

Table 4.14 Normalization using min–max rescaling.

Friend	Age	Education	Rescaled age	Rescaled education
Bernhard	43	2.0	1.0	0.0
Gwyneth	38	4.2	0.0	1.0
James	42	4.0	0.8	0.91

Table 4.15 Normalization using standardization.

Friend	Age	Education	Rescaled age	Rescaled education
Bernhard	43	2.0	0.76	−1.15
Gwyneth	38	4.2	−1.13	0.66
James	42	4.0	0.38	0.49

in the set and divide the new values by the amplitude: the difference between the maximum and minimum of the new values. You can also use other intervals. For example, if you want the values to be in the interval [−1.0, ..., 1.0], you simply multiply the values in the interval [0.0, ..., 1.0] by 2 and then subtract 1.0 from each new value. Table 4.14 illustrates the application of min–max scaling to three values of two attributes – age and education – for the interval [0.0, ..., 1.0].

The second alternative, standardization, first subtracts the average of the attribute values and then divides the result by the standard deviation of these values. As a result, the values of the attribute will have now an average of 0.0 and standard deviation of 1.0. If we apply standardization to the data used for Table 4.14, we will obtain the values shown in Table 4.15.

Table 4.16 Euclidean distance with normalized values.

Normalized	B–G	B–J	G–J
Euclidean distance	2.59	1.73	1.51

Normalized data are very rarely less than −3 or larger than 3. Another thing you should know is that the normalized values obtained for the age would be the same whatever the scale (years or decades) of the original values. If normalization is applied to all attributes, all attributes will have the same importance when calculating the Euclidean distance between objects (Table 4.16).

You can see that now the most similar and dissimilar people are, in this case, the same as when measured the age in decades. In order to denormalize normalized values we should, for each attribute, multiply the normalized value by the original sample standard deviation and then add the original average. This should give the original values.

4.4 Data Transformation

Another important issue for data summarization is transformations that might be necessary to perform to simplify the analysis or to allow the use of particular modeling techniques. Some simple transformations used to improve data summarization are:

- *Apply a logarithmic function to the values of a predictive attribute:* This is usually performed for skewed distributions, when some of the values are much larger (or much smaller) than the others. The logarithm makes the distribution less skewed. Thus, log transformations make the interpretation of highly skewed data easier.
- *Conversion to absolute values:* For some predictive attributes, the value's magnitude is more important than its sign, if the value is positive or negative.

Example 4.8 As an example of the benefits of a log transformation, suppose we have the following data illustrating, with two quantitative attributes, the financial income of our friends and how much they spend in dinners per year. Suppose also that the large majority of our friends have a low or smaller than the average income and that a very small number of our friends have a very high income and spend a large amount of money on dinners. Thus, the values for the two attributes, income and dinner expense, are right skewed. Table 4.17 illustrates, for each friend, their income, in thousands of dollars, and how much money they spend on dinner.

Table 4.17 How much each friend earns as income and spends on dinners per year.

	Salary	Dinner expense
Friend	**$**	**$**
Andrew	17,000	2,200
Bernhard	53,500	4,500
Carolina	69,000	6,000
Dennis	72,000	7,100
Eve	125,400	10,800
Fred	89,400	7,100
Gwyneth	58,750	6,000
Hayden	108,800	9,000
Irene	97,200	9,600
James	81,000	7,400
Kevin	21,300	2,500
Lea	138,400	13,500
Marcus	830,000	92,000
Nigel	1,000,000	120,500

The left-hand side of Figure 4.4 illustrates how the relationship between income and dinner expenses would be plotted. As can be seen, since the data is highly right skewed, data from most of our friends, whose income is lower than the income average and whose dinner expense is lower then the expense average, are mixed up. Thus it is not easy to interpret the data. If we perform a logarithmic transformation, applying a logarithm of base 10 to the values of both attributes gives the values in the plot on the right-hand side of Figure 4.4. Now the data are more spread out, making the visualization of the differences between our friends, and the data interpretation, easier.

4.5 Dimensionality Reduction

As a rule of thumb, if we want to represent more attributes in a clear and easy-to-understand plot, we will need to reduce the level of detail we show for each attribute. In the previous plots, we showed all the values for each attribute. Of course, if the number of objects in a data set is very large, even for two attributes, the plot representing all the objects in the data set will not be clear. In the previous chapter, we saw a plot that reduced the level of information for a attribute and described, even for data sets with a large number of

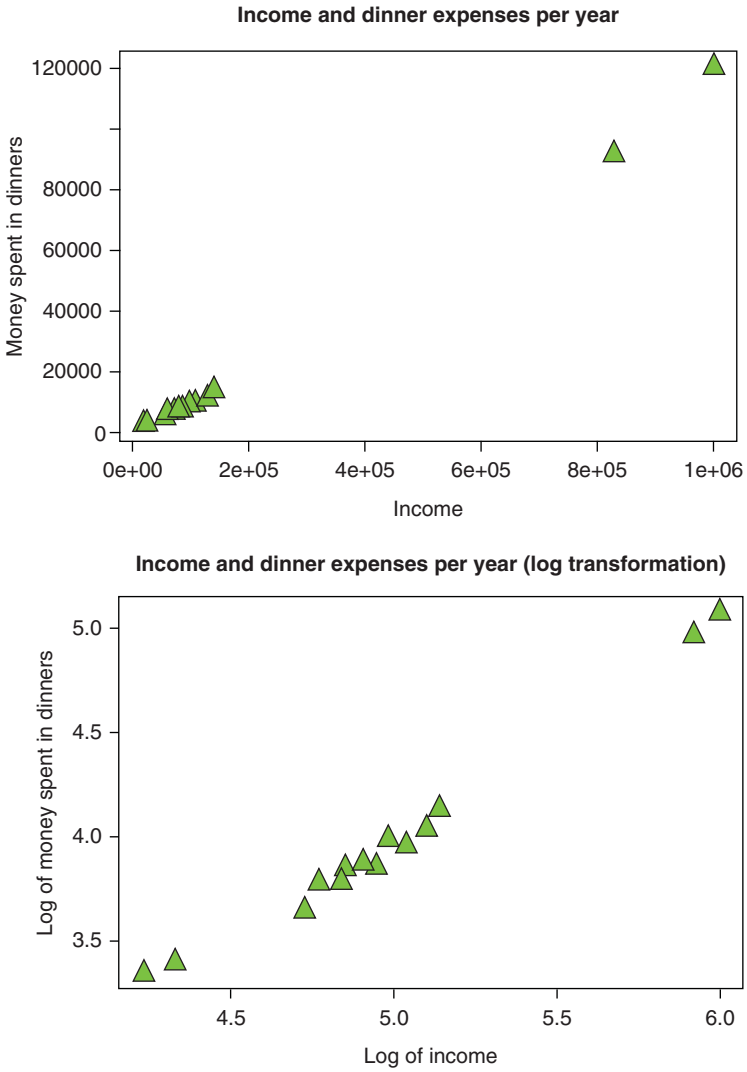


Figure 4.4 Two alternatives for a plot for three attributes the last of which is qualitative.

objects, relevant information regarding the value distribution of the attribute in the data set.

When the number of attributes in a data set is very large, the data space becomes very sparse and the distance between objects becomes very similar, reducing the performance of distance-based ML techniques (discussed later in this book).

The dimensionality reduction of a data set can bring several benefits:

- reducing the training time, decreasing memory needed and improving performance of ML algorithms;
- eliminating irrelevant attributes and reducing the number of noisy attributes;
- allowing the induction of simpler and therefore more easily interpretable models;
- making the data visualization easier to understand and allowing visualization of data sets with a high number of attributes;
- reducing the cost of feature extraction, making ML-based technologies accessible to a larger number of people.

There are two alternatives to reduce the number of attributes: attribute aggregation or attribute selection. In attribute aggregation, we replace a group of attributes with a new attribute: a combination of the attributes in the group. In attribute selection (also called "feature selection"), we select a subset of the attributes that keeps most of the information present in the original data set.

These alternatives are detailed in the next two subsections.

4.5.1 Attribute Aggregation

Attribute aggregation, also known as multidimensional scaling, reduces the data to a given number of attributes, allowing an easier visualization. The selected attributes are those that best differentiate the objects. Attribute aggregation techniques project the original data set into a new, lower-dimensional space, but keeping the relevant information.

Several techniques have been proposed for attribute aggregation in the literature. Most of them work by linearly combining the original attributes, creating a smaller number of attributes, referred to as a set of components.

This set of techniques includes principal component analysis (PCA), independent component analysis (ICA) and multidimensional scaling (MDS). Other techniques can also create non-linear combinations of the original attributes of a data set. As an example, we can mention a variation of PCA called kernel PCA.

Next, the main aspects of some of these techniques are briefly presented.

4.5.1.1 Principal Component Analysis

Proposed in 1901 by Karl Pearson [14], who also proposed the Pearson correlation, PCA is the most frequently used attribute aggregation technique. PCA linearly projects a data set onto another data set whose number of attributes is equal or smaller. Usually, the number is smaller, so as to reduce the dimensionality. The reduction in the number of attributes is obtained by removing redundant information. PCA tries to reduce redundancy by combining the original attributes into new attributes in order to decrease the covariance, and as a result

the correlation between the attributes of a data set. To do this, transformation matrix operations from linear algebra are used [15]. These operations transform the attributes in the original data set, which can have high linear correlation, to attributes that are not linearly correlated. These are called the principal components.

Each principal component is a linear combination of the original attributes. The use of linear combination restricts the possible combinations and makes the procedure simple. The components are ranked according their variance, from the largest to the smallest. Next, a set of components is selected, one by one, starting with the component with the largest variance and following the ranking. At each selection, the variance of the data with the selected components is measured. No new components are selected once the increase in the variance is small or a predefined number of principal components has been selected.

A key concept to understand aggregation techniques is the concept of data projection. A projection transforms a set of attributes in one space to a set of attributes in another space. The original data are named sources and the projected data signals. A simple projection would be to remove some attributes from the original data set, creating a new data, the projected data set, with the remaining attributes. In a projection, we want to keep as much of the information present in the original set of attributes as possible. We can do a more sophisticated projection by, instead of removing some of the attributes, combining some attributes into a new attribute. Ideally, the projection removes redundancy and noise from the original data.

For example, suppose that our original attributes are age, years of undergraduate study and years of postgraduate study. We can transform the two last attributes into a new attribute, "years of study", by adding the two original values.

As another example, when we take a photograph, we are transforming an image in a three-dimensional space to an image in a two-dimensional space. Thus, we are projecting our three-dimensional face image onto a two-dimensional image. By doing so, we lose some information, but we retain in the new space the necessary information to recognize the original image.

PCA is often used to visualize a data set with more than three attributes in a two-dimensional plot.

Example 4.9 Next, we show an example of the application of PCA to a data set. In this example, we apply PCA to the excerpt from our contact list, as presented in Table 3.1. We apply PCA, reducing five attributes – the four quantitative attributes plus the qualitative attribute "gender" converted to a numeric scale – to two principal components, so we can see the data in a two-dimensional plot. Figure 4.5 shows this plot. Each object receives the format and color of its class.

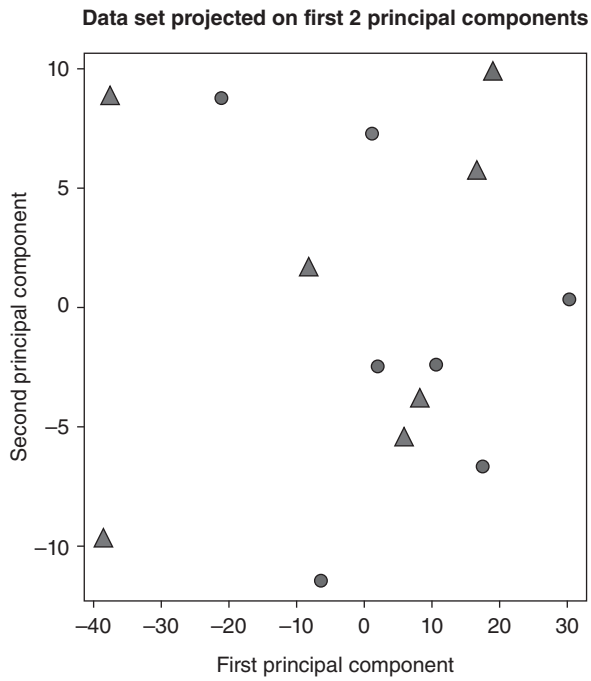


Figure 4.5 Principal components obtained by PCA for the short version of the contacts data set.

Each axis is associated with one of the two principal components selected. For this particular data set, the two components retain more than 90% of the information present in the original data.

In the previous example, the "gender" attribute needed to be converted to a number because PCA works only with quantitative attributes. An approach similar to PCA, but for use with qualitative data, is correspondence analysis [16] and its extension, multiple correspondence analysis [17].

In science, different fields discover similar things, but as they are not always aware of what the others are doing, those involved assume that they have invented a new technique and use a different name for it. This is the case for PCA, which originated in the statistics area, and singular value decomposition (SVD), which originated in the numerical analysis field. The two techniques use different mathematical operations to reach the same point. Thus, they have different names for similar techniques. The linear algebra used by PCA is provided by SVD. PCA can use SVD as another mathematical tool to extract the principal components from a data set.

One of the main strengths of PCA is that it is non-parametric. Since there is no need to choose coefficient values or tune hyper-parameters, the user does not need to be an expert on PCA. On the other hand, this same strength makes

it difficult to use prior information to improve the quality of the principal components. This would be possible if the technique had hyper-parameters. Thus, although PCA is a simple technique, some of the mathematical assumptions involved are very strong. There are two alternatives that can overcome this weakness. One of them is the incorporation of a kernel function, with hyper-parameters to be set. This is the option adopted by kernel PCA. The other approach is provided by ICA.

4.5.1.2 Independent Component Analysis

ICA is very similar to PCA, but the only assumption made by PCA that is also made by ICA is that there is a linear combination of the attributes [18]. By reducing the assumptions, ICA is able to find components with less redundancy than PCA, but at the cost of a higher processing time.

In contrast to PCA, ICA assumes that the original attributes are statistically independent. Thus, ICA tries to decompose the original multivariate data into independent attributes that do not have a Gaussian distribution. In addition, while PCA tries to decrease the covariance between attributes, ICA tries to reduce higher-order statistics, such as kurtosis. Being based on higher-order statistics, ICA is better than PCA for noisy data sets.

Another difference between PCA and ICA is that ICA does not rank the components. This is not a bad feature, since the principal components ranking order found by PCA is not always the best set of components. This is similar to the difference between ranking attributes and selecting a subset of attributes in the attribute selection approach. Components that individually have higher variance when combined do not necessarily result in a better pair of components than some other combination of components with a lower ranking position.

ICA is closely related to a technique called "blind source separation". If ICA is applied to the sound in a party, it would be able to separate the speakers, considering each to be an independent source of sound.

Example 4.10 To illustrate the difference between the components found by PCA and ICA, Figure 4.6 shows the two-dimensional plots for each. This plot uses the same data used in Figure 4.5, where each object receives the format and color of its class.

4.5.1.3 Multidimensional Scaling

Like the previous attribute aggregation techniques, MDS involves a linear projection of a data set [19]. However, while the previous techniques used the values of the attributes of the objects in the original data set, MDS uses the distances between pairs of objects. Since it does not need to know the value of the object attributes, MDS is particularly suitable when it is difficult to extract relevant features to represent the objects. It is just necessary to know how similar pairs of objects are.

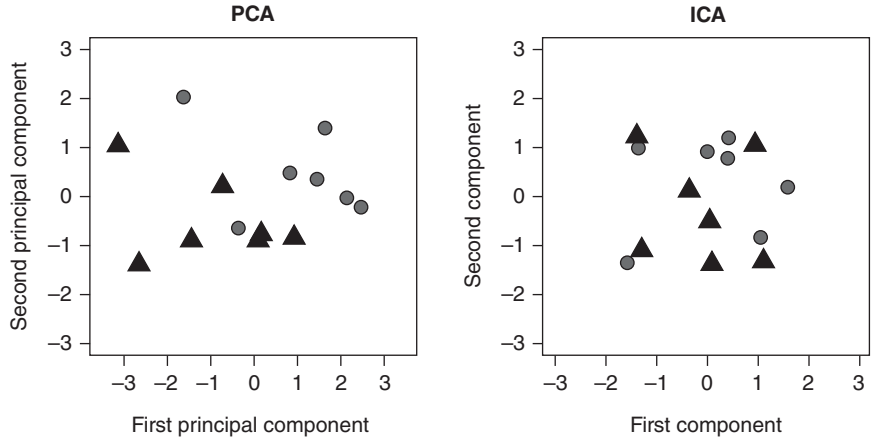


Figure 4.6 Components obtained by PCA and ICA for the short version of the contacts data set.

4.5.2 Attribute Selection

Instead of aggregating attributes, another approach to reduce dimensionality is by selecting a subset of the attributes. This approach can speed up the learning process, since a smaller number of operations will need to be made.

Attribute selection techniques can be roughly divided into three categories: filters, wrappers and embedded.

4.5.2.1 Filters

Filters look for simple, individual, relations between the predictive attribute values and the target attribute. They rank the attributes according to this relation. If the values of a predictive attribute have a strong relation with a label attribute value, for example, high predictive attribute values are related with class A and low values with class B, this predictive attribute receives a high ranking position.

Example 4.11 Using the previously shown excerpt of our data set (Table 2.1) we can show how similar the behavior of each predictive attribute is to the target attribute (Company). To do this, we will apply a statistical measure already described in Chapter 2 – the Pearson correlation – for each pair (predictive attribute, target attribute). Table 4.18 shows, for each predictive attribute, its correlation with the target attribute. The attributes are shown in the decreasing correlation order.

Suppose we want to select the three most relevant predictive attributes. Thus, to select three predictive attributes using this table, the predictive attributes with the highest correlation with the target attribute company – Years, Gender

Table 4.18 Correlation between each predictive attribute and the target attribute.

Predictive attribute	Correlation
Years	0.89
Gender	0.58
Weight	0.40
Height	0.21
Maxtemp	0.14

and Weight – would be selected. We can also check the correlation between pairs of predictive attributes. If two predictive attributes have a high correlation, they may be redundant. Thus, even if they have a high correlation with the target attribute, it is not a good idea to select both of them: both have the same information necessary to predict the target attribute value.

Looking at each predictive attribute individually, two problems may arise. One is that two or more redundant predictive attributes can have a strong relationship with the class attribute, both being selected by predictive filters. The second problem is the inability of filters to identify relations between the class attribute and a combination of predictive attributes. Two positive aspects of filters are that they are not influenced by the classification algorithm used and that they are computed quickly.

4.5.2.2 Wrappers

Wrappers explicitly use a classifier to guide the attribute selection process. As a result, the approach selects the set of predictive attributes that provides the highest predictive performance for the classifier, regardless of the attribute ranking positions. Instead of ranking the predictive attributes, wrappers usually look for the subset of attributes with the best predictive ability, thus capturing the relationship between attributes and reducing the chance of selecting redundant attributes.

Since they need to induce and evaluate several classifiers for each subset of predictive attributes, wrappers usually have a higher computational cost than filters.

Example 4.12 To show a simple example of a wrapper technique, we use again the data from Table 2.1, showing the good predictive performance of a classifier induced by an ML algorithm A if it uses a reduced dataset: a simplification of Table 2.1 with just one predictive attribute. We do this for each one of the predictive attributes. Thus, if we have five predictive attributes,

Table 4.19 Predictive performance of a classifier for each predictive attribute.

Predictive attribute	Predictive performance
Years	0.78
Height	0.46
Gender	0.42
Weight	0.38
Maxtemp	0.14

we reduce the dataset five times, each time using a different attribute to induce a classifier. The predictive performance using each predictive attribute is illustrated in Table 4.19, in decreasing order of predictive performance; that is, the higher the value, the better.

It is possible to see that the order of predictive attributes obtained is different to the one found found using the filter, as shown in Table 4.18. Thus, if we now use the wrapper technique to select the three predictive attributes that would be the best predictor of the target attribute value, they would be "years", "height" and "gender".

We do not need, either for the filter or for the wrapper technique, to check each predictive attribute individually to see how good its prediction would be. As we saw in the filter example, this can lead to redundancy. A better set of predictive attributes can be found if we look for the group of predictive attributes that, when combined, are more correlated to the target attribute, in a filter-based approach, or when used by a classification algorithm would induce the best classification model, in a wrapper-based approach. However, the number of possible groups of predictive attributes is usually much larger than the number of individual predictive attributes, so the processing time to look for the best group is much higher than just looking for the ranking of the best predictive attributes.

By relying on the classifier performance to select predictive attributes, wrappers increase the chances of classifier overfitting. Moreover, the performance of a wrapper is affected by the classification algorithm used and there is no guarantee that a subset selected by a particular classification algorithm will also be a good subset for other classification algorithms.

4.5.2.3 Embedded

In the embedded category, attribute selection is performed as an internal procedure of a predictive algorithm. One predictive technique able to perform embedded attribute selection is a decision tree induction algorithm. These algorithms are covered in Section 10.1.1. When these algorithms are applied to

a data set, they produce a predictive model and select a subset of predictive attributes chosen be the most relevant for classification model induction.

4.5.2.4 Search Strategies

It does not matter if the attribute selection technique is a filter, a wrapper or is embedded, it is necessary to have a criterion to define how the best attribute subset will be determined.

The simplest search strategy is exhaustive search, where all possible subsets of attributes are evaluated and the best subset is selected. If the number of attributes is high, this strategy may take a very long time.

Two more efficient, and still very simple, search strategies for attribute selection are the greedy sequential techniques, forward selection and backward selection.

In forward selection, the selection starts with an empty set of attributes. Then, a model per existing attribute is trained and tested. The attribute used by the model with best predictive performance is selected. Next, a second attribute is added by selecting the best additional attribute from the remaining attributes: the one that gives the best subset of two attributes regarding the measure used to evaluate the attribute subset. If none of these additional attributes improves the attribute subset, none of them is added and the selection stops. If a second attribute is added, the process continues for the remaining attributes, adding one at a time, until the subset stops improving.

The backward strategy works in the opposite direction. It begins with a subset comprising all the attributes. Then attributes are removed, one by one, until the predictive performance of the subset starts to decrease or the number of attributes equals 1. Each time the removal of an attribute is evaluated, the attribute the removal of which causes least harm or most benefit is removed.

There are several variations of the forward and backward attribute selection techniques. One of these, the bidirectional approach, alternates between forward and backward selection.

Other search strategies, like those based on optimization techniques, use more sophisticated and complex mechanisms and, as a result, produce a better attribute subset.

The number of predictive attributes to be selected can be given by the user or defined through an optimization process with two objectives: select the smallest possible set of predictive attributes able to improve or maintain the performance of a model induction technique. Since there is more than one objective, multi-objective optimization techniques can be used.

The curse of dimensionality The curse of dimensionality relates the ratio between the number of objects and number of attributes in a data set to phenomena that can be observed in this data set. As the number of attributes increases, the data volume become more sparse, with no objects in various

regions of the input space. In addition, the distance between objects converge to the same value, which reduces the performance of distance-based predictive and descriptive techniques. The number of objects needed to induce models with high predictive performance increases exponentially with the number of predictive attributes. Two alternatives to overcome this problem are to increase the number of objects or to reduce the number of predictive attributes. Since, in many applications, it is not possible to increase the number of objects, the only feasible alternative is to reduce the input dimension.

4.6 Final Remarks

This chapter discussed aspects of data quality and described preprocessing techniques frequently used in data analytics. The quality of a dataset strongly affects the results of a data quality project. To reduce or remove data quality problems, in a process known as data cleansing, several techniques are available. These techniques allow to problems like missing values, redundant data, data inconsistencies and noise presence to be deal with. Commonly used cleansing techniques for these problems were described and examples of their use were presented. There are several books dedicated exclusively to data cleansing.

This chapter also discussed techniques for data-type conversions, a necessary operation when the values of a predictive attribute need to be of a different type (e.g. the values are nominal but need to be relative). These techniques can convert quantitative data to qualitative data and vice-versa. Motivations and techniques for conversions to different value scales and data transformations to make the data analysis simpler were also discussed.

The last topic covered by this chapter was problems due to the presence of a large number of predictive attributes and how this number can be reduced by aggregating attributes and selecting subsets of attributes.

The next chapter covers one of the two analytics tasks, namely descriptive analytics, describing how data groups can be extracted from a data set using clustering techniques.

4.7 Exercises

1 Are there any similarities in the procedures used to deal with missing values and noisy values? If so, what are they?

2 How can we estimate missing values using location statistics?

Table 4.20 Filling of missing values.

Food	Age	Distance	Company
Chinese	—	far	Good
Italian	—	Very close	Bad
Mediterranean	47	Very close	Good
Italian	82	Far	Good
—	—	Very far	Good
Chinese	29	—	Bad
Burgers	—	Very far	Bad
Chinese	25	Close	Good
Italian	42	Far	Good
Mediterranean	25	Close	Good

3 What are the differences between irrelevant, inconsistent and redundant data?

4 Fill the missing values in Table 4.20.

5 When is an outlier value not a noisy value and when is a noisy value not an outlier?

6 What is the main problem that occurs when qualitative nominal values are converted to quantitative values?

7 If you need to normalize the values of an attribute, when is it better to use min–max scaling and when is it better to use standardization? Apply both techniques to the age values in Table 4.20, before and after filling in the missing values.

8 When faced with the curse of dimensionality, is it better to increase the number of objects or decrease the number of attributes? What are the advantages and disadvantages of each approach?

9 Provide three advantages and disadvantages of using attribute aggregation instead of the original set of attributes.

10 What happens with PCA when more components are selected from the ranking?

5

Clustering

Let us get back to the data set of our contacts. Suppose you want to organize dinners to introduce some of your friends and you decided that, for each meeting, you would invite people of similar education level and age. To this end, you collect the data about your friends shown in Table 5.1. The education level is a quantitative attribute, which can take values of 1 (primary), 2 (high school), 3 (undergraduate), 4 (graduate) or 5 (postgraduate). The decimal numbers represent that a given level is ongoing. For instance, 4.5 represents someone who is in the middle of a postgraduate course.

In the previous chapters, we looked at some simple data analyses we can perform on contact data. Although these analyses provide us with interesting and useful information, we can obtain even more interesting analysis by using other, more sophisticated, data mining techniques. One of these analyses could be to find, among our contacts, groups of friends of similar age and education levels.

We saw in the previous chapters that data analysis modeling tasks can be roughly divided into descriptive and predictive tasks. In this chapter, we will present an important family of techniques for descriptive tasks. They can describe a data set by partitioning it, so that objects in the same group are similar to each other. These "clustering" techniques have been developed and extensively used to partition data sets into groups. Clustering techniques use only predictive attributes to define the partitions. For this reason, they are unsupervised techniques. Figure 5.1 illustrates an example, in which a clustering technique is applied to our data set of ten people, producing two clusters.

The number of clusters is usually not defined beforehand, but is found through trial and error. Human feedback or clustering validation measures are used to find a suitable number of clusters into which to partition a data set. Figure 5.2 illustrates the results of clustering the previous data set into three clusters.

The larger the number of clusters, the more similar the objects inside the cluster are likely to be. The limit is to define the number of clusters to be equal to the number of objects, so each cluster has just one object.

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

Table 5.1 Simple social network data set.

Name	Age	Educational level
Andrew (A)	55	1
Bernhard (B)	43	2
Carolina (C)	37	5
Dennis (D)	82	3
Eve (E)	23	3.2
Fred (F)	46	5
Gwyneth (G)	38	4.2
Hayden (H)	50	4
Irene (I)	29	4.5
James (J)	42	4.1

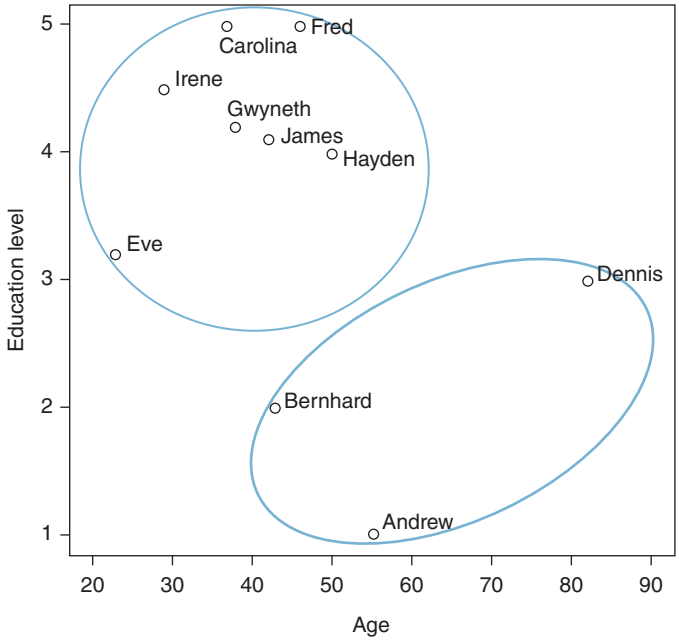


Figure 5.1 Applying clustering to a dataset.

5.1 Distance Measures

Before we define groups of similar data, we must agree on what is similar and what is not similar (dissimilar). We can represent the similarity between two objects by a single number. This will allow us to say, for a particular object,

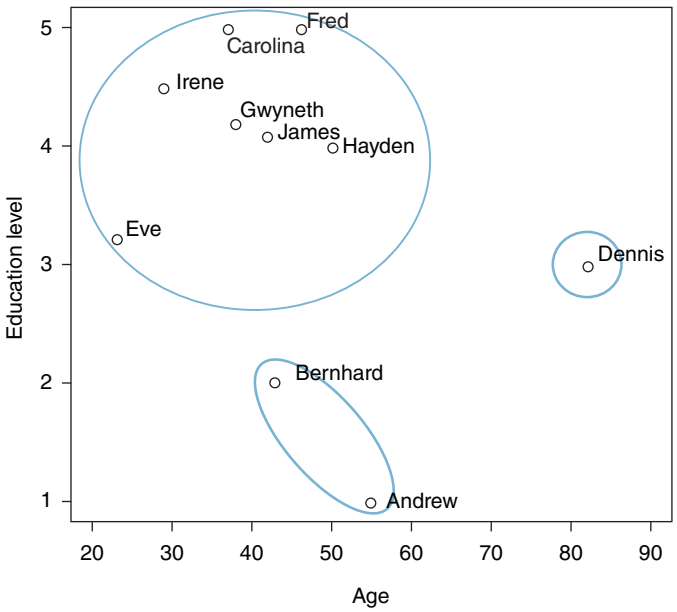


Figure 5.2 Alternative clustering of the data set.

which other objects in the same data set are more similar and which are more dissimilar. A common approach to associate a number with the similarity (and dissimilarity) between two objects is to use distance measures. The most similar objects have the smallest distances between them, and the most dissimilar have the largest distances. The way we compute the distance between objects depends on the scale type of its attributes: whether they are quantitative or qualitative.

5.1.1 Differences between Values of Common Attribute Types

The difference between two values for the same attribute, here named a and b, will be denoted as $d(a, b)$. For quantitative attributes, one can calculate the absolute difference:

$$d(a.b) = |a - b| \tag{5.1}$$

Example 5.1 For example, the difference in age between Andrew ($a = 55$) and Carolina ($b = 37$) is $|55 - 37| = 18$. Note, that even if we change the order of the values ($a = 37$ and $b = 55$) the result is the same.

If the attribute type is qualitative, we use distance measures suitable for the given type. If the qualitative attribute has ordinal values, we can measure the difference in their positions as:

$$d(a, b) = (|pos_a - pos_b|)/(n - 1) \tag{5.2}$$

where n is the number of different values, and pos_a and pos_b are the positions of the values a and b, respectively, in a ranking of possible values.

Example 5.2 In our data set, education level can be considered an ordinal attribute, with larger values meaning a higher level of education. Thus the distance between the education levels of Andrew and Caroline is $|pos1 - pos5|/4 = |1 - 5|/4 = 1$.

Note that ordinal attributes need not be expressed only by numbers. For example, the education level can have values such as "primary", "high school", "undergraduate", "graduate" and "postgraduate". However, these can be readily transformed into numbers (see Section 2.1).

If a qualitative attribute has nominal values, in order to compute the distance between two values we simply determine if they are equal (in which case the difference, or dissimilarity, will be zero) or not (in which case the difference will be one).

$$d(a, b) = \begin{cases} 1\,, & \text{if } a \neq b \\ 0\,, & \text{if } a = b \end{cases} \tag{5.3}$$

Example 5.3 For example, the distance between the names of Andrew and Carolina is equal to 1, since they are different.

In general, computing the distance between two objects consists of aggregating the distances, usually differences, between their corresponding attributes. For our data set, it means aggregating the difference between the ages and education levels of two people.

The most common types of attributes are quantitative, having numeric values, and qualitative (ordinal and nominal), having either Boolean values, words or codes. Qualitative attributes in a data set can be transformed into quantitative attributes (see Chapter 2), thus representing each object (row in the data table) in this data set by a vector of numbers.

An object represented by a vector of m quantitative attributes can be mapped to an m-dimensional space. For our simplified social network data set, which has two attributes, "age" and "education level", each person can be mapped to an object in a two-dimensional space, as illustrated in Figure 5.1. The more similar two objects are, the closer they are when mapped in this space. Let us now look at distance measures for objects with quantitative attributes.

5.1.2 Distance Measures for Objects with Quantitative Attributes

Several distance measures are particular cases of the Minkowski distance. The Minkowski distance for two m-dimensional objects p and q with quantitative attributes is given by:

$$d(p,q) = \sqrt[r]{\sum_{k=1}^{m} |p_k - q_k|^r} \tag{5.4}$$

where m is the number of attributes, while p_k and q_k are the values of the k th attribute for objects p and q, respectively. Variants are obtained using different values for r. For example, for the Manhattan distance, $r = 1$, and for the Euclidean distance, $r = 2$.

The Manhattan distance is also known as the city block or taxicab distance, since if you are in a city and want to go from one place to another, it will measure the distance traveled along the streets. The Euclidean distance may sound familiar to those who know Pythagoras's theorem, which measures the size of the longest side of a right-angled triangle.

Figure 5.3 illustrates the difference between the Euclidean distance and the Manhattan distance. The Euclidean distance, with length 7.07, is represented by the diagonal line. The other highlighted line is the Manhattan distance, of length 10.

There are also other attribute types that are neither quantitative nor qualitative, but are often encountered in data mining. These attribute types, here termed "non-conventional', include:

- biological sequences
- time series
- images
- sound
- video.

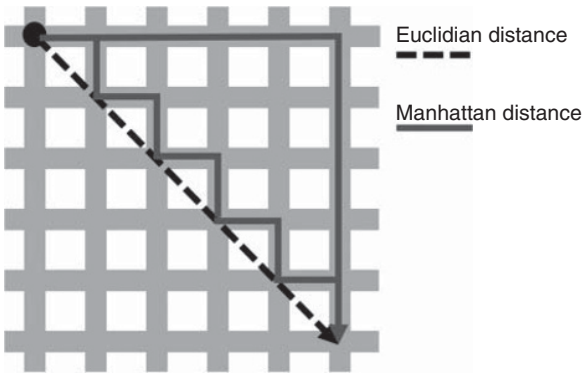


Figure 5.3 Euclidean and Manhattan distances.

All these non-conventional attribute types can be converted into quantitative or qualitative types [20]. Among these non-conventional attribute types, the most common are sequences (text, biological and time series) and images. The topic of distance measures for sequences is discussed next.

5.1.3 Distance Measures for Non-conventional Attributes

The Hamming distance can be used for sequences of values and these values are usually characters or binary values. A binary value (or binary number) is either 1 or 0, meaning true or false, in general. The Hamming distance is the number of positions at which the corresponding characters or symbols in the two strings are different.

For example, the Hamming distance between the strings "James" and "Jimmy" is 3, and between "Tom" and "Tim" is 1. Note, that the Hamming distance is a special case of the Manhattan distance, when the attributes can assume only binary values (0 or 1).

For short sequences that can have different sizes, sequence distance measures, such as the Levenshtein distance, also sometimes referred to as the edit distance, can be used. The edit distance measures the minimum number of operations necessary to transform one sequence into another. The possible operations are: insertion (of a character), removal (of a character) and substitution (of a character by another).

Example 5.4 The edit distance between the strings "Johnny" and "Jonston" is 5, since it is necessary to substitute the characters h, n, n, y with n, s, t, o (four operations), and to add a character n to the end (a fifth operation). A similar idea is used in bioinformatics to compare DNA, RNA and amino acid sequences.

For long texts, such as a text describing a product or a story, each text is converted into an integer vector using a method called the "bag of words". The bag of words method initially extracts a list of words, containing all the words appearing in the texts to be mined. Each text is converted into a quantitative vector, where each position is associated with one of the words found and its value is the number of times this word appeared in the text.

Example 5.5 For example, for the two texts:

- A = "I will go to the party. But first, I will have to work."
- B = "They have to go to the work by bus."

the bag of words will produce the vectors shown in Table 5.2.

Now, the Euclidean distance can be calculated for these 13-dimensional quantitative vectors, measuring the distance between them. Other techniques,

Table 5.2 Example of bag of words vectors.

	I	will	go	to	the	party	but	first	have	work	they	by	bus
A	2	2	1	2	1	1	1	1	1	1	0	0	0
B	0	0	1	2	1	0	0	0	1	1	1	1	1

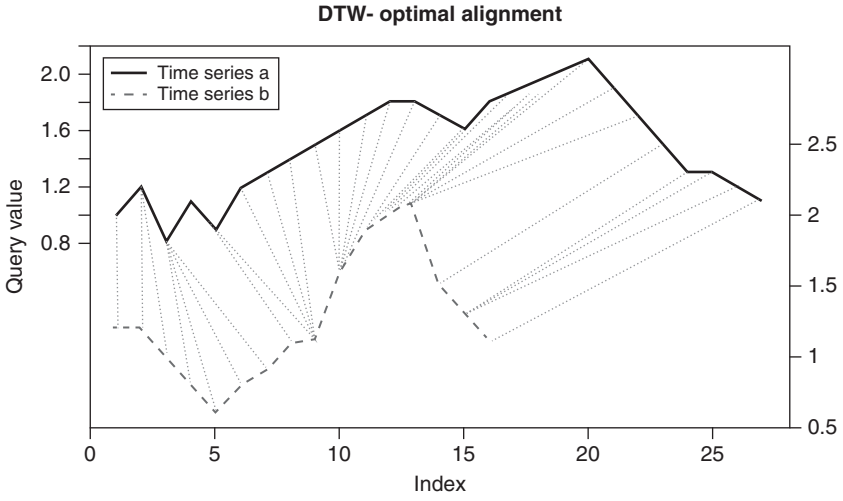
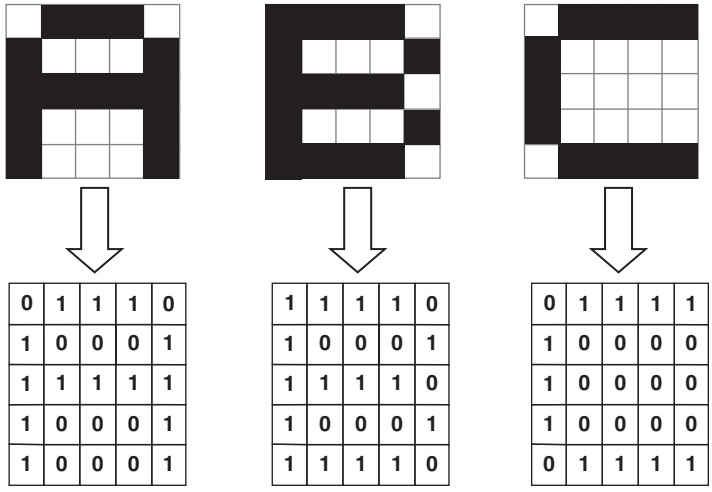


Figure 5.4 Dynamic time warping.

similar to the bag of words, can be used to convert texts (or documents) to quantitative vectors, a subject to be discussed in Chapter 13.

Other commonly found sequences are time series (say, ECG or EEG data in the medical domain). To compute the distance between two time-series, a popular choice is the dynamic time warping distance, which is similar to the edit distance. It returns the value of the best match between the two time series, considering variations and shifts in time and speed. Figure 5.4 illustrates an example of the best alignment between two time series.

To calculate the distance between images, two distinct approaches can be used. In the first, features associated with the application can be extracted from the images. For example, for a face recognition task, the distance between the eyes can be extracted. Ultimately, each image is represented by a vector of real numbers, where each element corresponds to one particular feature. In the second approach, each image is initially converted to a matrix of pixels, where the size of the matrix is associated with the granularity required for the image. Each pixel can then be converted into an integer value. For example, for black and



	1st row					2nd row					3rd row					4th row					5th row				
A	0	1	1	1	0	1	0	0	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0	1
B	1	1	1	1	0	1	0	0	0	1	1	1	1	1	0	1	0	0	0	1	1	1	1	1	0
C	0	1	1	1	1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1	1	1

Figure 5.5 Transformation of images of size 5 × 5 pixels into matrices and vectors.

white images, the darker the pixel, the higher the value. The matrix can be transformed into a vector. The distance measure, in both cases, could be the same as is used for vectors of quantitative attribute values.

Example 5.6 The images in Figure 5.5 represent the letters A, B and C on a canvas of size 5 × 5 pixels. Since the images are black and white (two colors), each can be represented by a binary matrix of 5 rows and 5 columns or a binary vector of size 25, such that the rows of the matrix are copied one after the other. In other words, we map each image onto a 25− dimensional space. We can now use arbitrary distance measures to compute the difference between these images. The Manhattan distance between the images A and B is 6, between the images A and C is 11 and between the images B and C is 9. Note, that in the case of binary values (only 0 or 1) the Manhattan distance is equivalent to the edit distance if 0 and 1 represent characters or letters.

Other types of data such as sound or video are readily converted into a sequence of quantitative vectors by extracting features in the same way as is done for

images. For sound, it might be some low-level signal parameter, for example bandwidth or pitch. A video can be seen as a sequence of images and sounds.

5.2 Clustering Validation

To find good clustering partitions for a data set, regardless of the clustering algorithm used, the quality of the partitions must be evaluated. In contrast to the classification task, the identification of the best clustering algorithm for a given data set lacks a clear definition.

For predictive tasks like classification, the evaluation of predictive models has a clear meaning: how well the predictive models classify objects. The same cannot be said for clustering partitions, since there are many definitions of what is a good partition. However, some validation measures are frequently used in clustering tasks.

Several cluster validity criteria have been proposed; some automatic and others using expert input. The automatic validation measures for clustering partition evaluation can be roughly divided into three categories:

- *External indices:* The external criteria uses external information, such as class label, if available, to define the quality of the clusters in a given partition. Two of the most common external measures are the correct-RAND and Jaccard.
- *Internal indices:* The internal criteria looks for compactness inside each cluster and/or separation between different clusters. Two of the most common internal measures are the silhouette index, which measures both compactness and separation, and the within-groups sum of squares, which only measures compactness.
- *Relative indices:* The relative criterion compares partitions found by two or more clustering techniques or by different runs of the same technique.

The Jaccard external index, the silhouette and the within-groups sum of squares measures, both internal indices, are discussed next.

Silhouette internal index This evaluates compactness inside each cluster, measuring:

- how close to each other the objects inside a cluster are
- the separation of different clusters – how far the objects in each cluster are to the closest object from another cluster.

To do this, it applies the following equation to each object x_i:

$$s(x_i) = \begin{cases} 1 - a(x_i)/b(x_i) & , \quad \text{if } a(x_i) < b(x_i) \\ 0 & , \quad \text{if } a(x_i) = b(x_i) \\ b(x_i)/a(x_i) - 1 & , \quad \text{if } a(x_i) > b(x_i) \end{cases} \tag{5.5}$$

where:

- $a(x_i)$ is the average distance between x_i and all other objects in its cluster
- $b(x_i)$ is the minimum average distance between x_i and all other objects from each other cluster.

The average of all $s(x_i)$ gives the partition silhouette measure value.

Within-groups sum of squares This is also an internal measure but only measures compactness. It sums the squared Euclidean distance between each instance and the centroid of its cluster. From Equation (5.4) we know that the squared Euclidean distance between two instances p and q with m attributes each is given by:

$$sed(p, q) = \sum_{k=1}^{m} |p_k - q_k|^2 \tag{5.6}$$

The within groups sum of squares is given by:

$$s = \sum_{i=1}^{K} \sum_{j=1}^{J_i} sed(p_j, C_i) \tag{5.7}$$

where K is the number of clusters and J_i is the number of instances of cluster i, and C_i is the centroid of cluster i.

Jaccard external measure This is a variation of a similar measure used in classification tasks. It evaluates how uniform the distribution of the objects in each cluster is with respect to the class label. It uses the following equation:

$$J = (M_{11})/(M_{01} + M_{10} + M_{11}) \tag{5.8}$$

where:

- M_{01} is the number of objects in other clusters but with the same label
- M_{10} is the number of objects in the same cluster, but with different labels
- M_{00} is the number of objects in other clusters with different labels
- M_{11} is the number of objects in the same cluster with the same label.

5.3 Clustering Techniques

Since we already know how to measure similarity between records, images, and words, we can proceed to see how to obtain groups/clusters using clustering techniques. There are hundreds of clustering techniques, which can be categorized in various ways. One of them is how the partitions are created, which defines how the data are divided into groups. Most techniques define partitions in one step (partitional clustering), while others progressively define

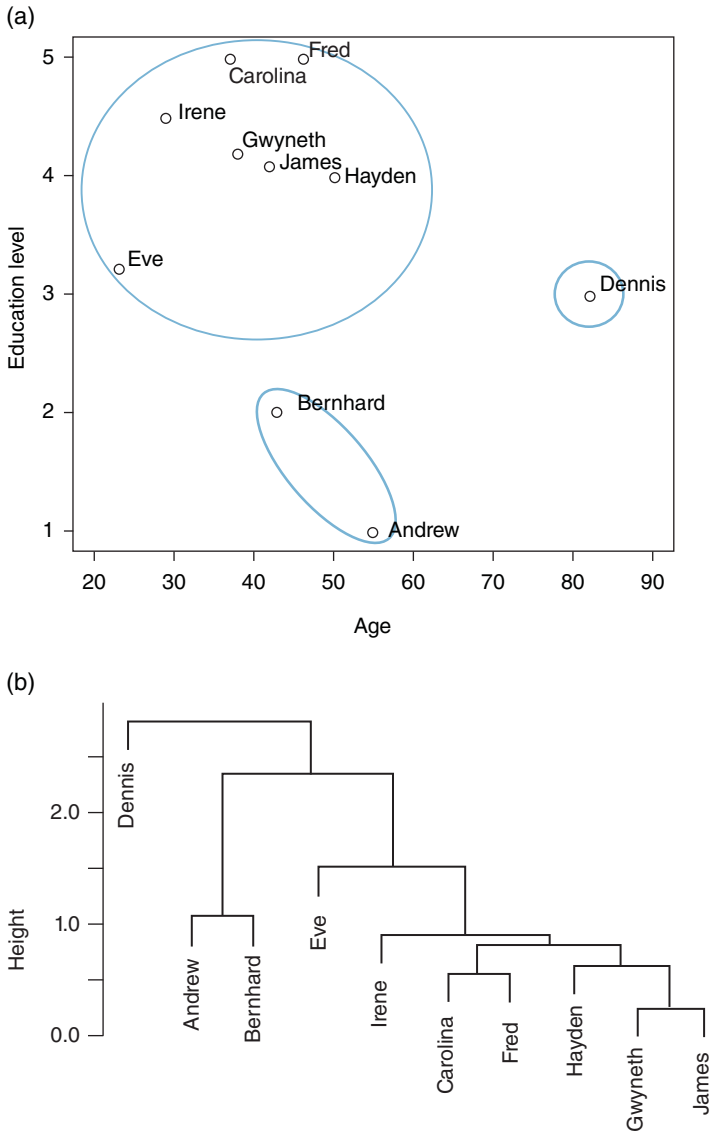


Figure 5.6 (a) Partitional and (b) hierarchical clustering.

partitions, either increasing or decreasing the number of clusters (hierarchical clustering). Figure 5.6 illustrates the difference between partitional and hierarchical clustering. In addition, in many techniques, each object must belong to one cluster (crisp clustering), but for others, each object can belong to a different degree to several clusters (ranging from 0%, does not belong, to 100%, completely belongs).

Another criteria is the approach used to define what a cluster is, determining the elements to be included in the same cluster. According to this criterion, the main types of clusters are [20]:

- *Separation-based:* each object in the cluster is closer to every other object in the cluster than to any object outside the cluster
- *Prototype-based:* each object in the cluster is closer to a prototype representing the cluster than to a prototype representing any other cluster
- *Graph-based:* represents the data set by a graph structure associating each node with an object and connecting objects that belong to the same cluster with an edge
- *Density-based:* a cluster is a region where the objects have a high number of close neighbors (i.e. a dense region), surrounded by a region of low density
- *Shared-property:* a cluster is a group of objects that share a property.

We will present next three different clustering methods representative of different approaches to do clustering. None is better than the others. Each has its own pros and cons, as we will see. These three methods are:

- *K-means:* the most popular clustering algorithm and a representative of partitional and prototype-based clustering methods
- *DBSCAN:* another partitional clustering method, but in this case density-based
- *Agglomerative hierarchical clustering:* a representative of hierarchical and graph-based clustering methods.

5.3.1 K-means

Centroids are a key concept in order to understand k-means. They represent a kind of centre of gravity for a set of instances. We start by describing the concept before explaining how k-means works, how to read the results, and how to set the hyper-parameters.

5.3.1.1 Centroids and Distance Measures

A centroid can also be seen as a prototype or profile of all the objects in a cluster, for example the average of all the objects in the cluster. Thus, if we have several photos of cats and dogs, if we put all the dogs in a cluster and all the cats in another cluster, the centroid in the dog cluster, for example, would be a photo representing the average features in all dog photos. We can observe, therefore, that the centroid of a cluster is not usually one of the objects in the cluster.

Example 5.7 The centroid for the friends Bernhard, Gwyneth and James has the average age and education of the three friends: 41 years (the average of 43, 38 and 42) and 3.4 (the average of 2.0, 4.2 and 4.1) as the education level. As you can see none of the three friends has this age and education level.

In order to have an object of the cluster as a prototype, a medoid is used instead of a centroid. The medoid of a cluster is the instance with the shortest sum of distances to the other instances of the cluster.

Example 5.8 Using the same example, the medoid of the three friends is Andrew because he is the friend with shortest sum of squared distances to the other two friends, a measure called the within-groups sum of squares (see Equation (5.7)). Looking at Table 4.12, James (J) has a within-groups sum of squares of $2.33 + 4 = 6.33$. The other two friends, Bernhard (B) and Gwyneth (G), have higher within-groups sums of squares, of 7.79 and 9.46 respectively.

We have already seen that there are several distance measures that can be used depending on the data we have. The same happens with k-means. By default, it uses the Euclidean distance, assuming that all attributes are quantitative. For problems with other characteristics, a different distance measure should be chosen. The distance measure is one of the main hyper-parameters of k-means.

5.3.1.2 How K-means Works

The way k-means works is shown graphically in Figure 5.7. This follows the k-means algorithm, the pseudocode for which can be seen below.

Algorithm K-means

1: INPUT D the data set
2: INPUT d the distance measure
3: INPUT K the number of clusters
4: Define the initial K centroids (they are usually randomly defined, but can be defined explicitly in some software packages)
5: **repeat**
6: Associate each instance in D with the closest centroid according to the chosen distance measure d
7: Recalculate each centroid using all instances from D associated with it.
8: **until** No instances from D change of associated centroid.

The clusters found by k-means always have a convex shape. A cluster of instances is of convex shape if a line connecting two arbitrary instances in the cluster lies within the cluster (Figure 5.8). Examples include a hypercube, hypersphere or hyperellipsoid. The particular shape depends on the value of r in the Minkowski distance and the number of attributes in the input vector. For example, if $r = 1$ (the taxicab distance), the clusters will be square if the number of attributes is 2, cubes if it is 3, and hypercubes if larger than 3. If $r = 2$ (the Euclidean distance), the clusters will be circles if the number of attributes is 2, spheres if it is 3, and hyperspheres if it is larger than 3.

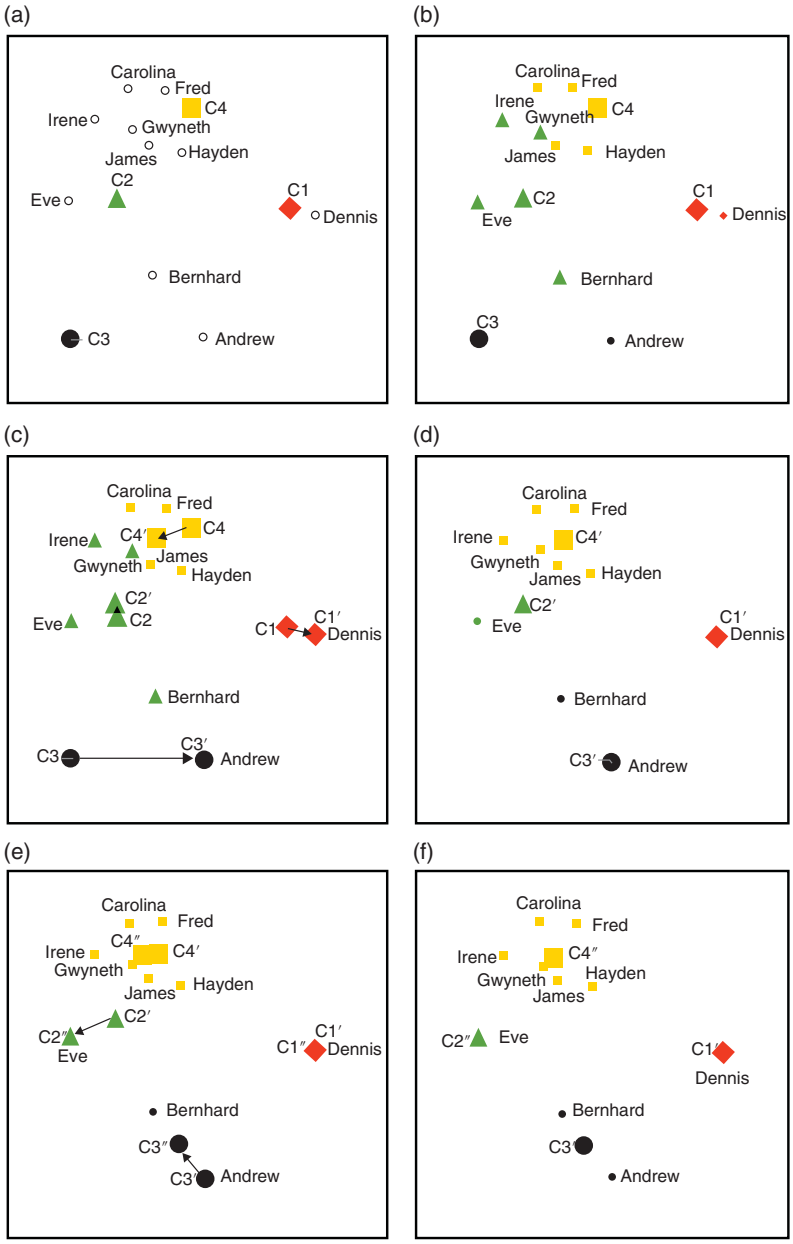


Figure 5.7 K-means with $k = 4$. The big symbols represent the centroids. The example is step-by-step according to the Algorithm K-means: (a) step 4; (b) 1st iteration of step 6; (c) 1st iteration of step 7; (d) 2nd iteration of step 6; (e) 2nd iteration of step 7; and (f) 3rd iteration of step (6). The algorithm stops in the 3rd iteration (f) because there is no instance changing of symbol between the 2nd (d) and the 3rd (f) iterations of step (6).

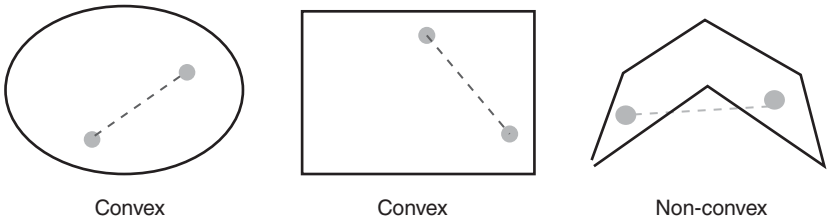


Figure 5.8 Examples of convex and non-convex shapes.

Table 5.3 The clusters to which each friend belongs to according to Figure 5.7a–f.

A	B	C	D	E	F	G	H	I	J
3	3	4	1	2	4	4	4	4	4

Other distance measures will give other convex shapes. For example, if the Mahalanobis distance is used, the shapes will be ellipses if the number of attributes is 2, ellipsoid if it is 3, and hyperellipsoid if it is larger than 3.

If a partition gives clusters with some non-convex shape, another technique should be used to find the partition. You should also be aware that data should be normalized when we use distance measures, as seen in Section 4.3.

The number of centroids that is randomly generated is a second hyper-parameter, usually represented by the letter k. So now you know the reason why this method begins with the letter k. Do you have an idea why there is the term "means" in the name method?

Assessing and evaluating results Once we have clustering results, how can we assess them? This is what we will explain now. Whichever tool you use, there will be the possibility to see what the instances of each cluster are (Table 5.3). Typically the clusters are numbered in an incremental sequence, starting at zero or one.

There is also the possibility to plot clusters, as we have seen in Figure 5.6. However, when the number of attributes is larger than two, the analysis is a bit more complex. There are ways to do it, using, for instance, principal components analysis (Section 4.5.1).

Another way of displaying clustering results is through the analysis of the centroids. However, care is required. Before using clustering methods such as k-means, for instance, quantitative data should be normalized. So, the centroids are typically normalized (see Table 5.4).

Table 5.4 Normalized centroids of the example dataset.

Centroid	Age	Education
1	−0.55	−0.13
2	1.94	−0.38
3	−1.31	−1.97
4	0.7	1.01

We already know from Section 4.3 what normalization is and how to interpret normalized data. In order to analyze the cluster centroids it is important to understand that the centroids are normalized if we have normalized the data before using the k-means. How should we read normalized data? Normalized data varies typically between −3 and 3. The more negative a value is, the lower than the average it is. The more positive a value is, the higher than the average it is. For each attribute, the value shows how much below or above the average of that attribute the centroid is. But, more importantly, it shows which attributes best discriminate two clusters.

Example 5.9 Analyzing our four centroids, it can be seen that cluster 3 is the one with younger and less educated members on average. Cluster 4 has the most educated members, on average. Cluster 2 has older people on average, and cluster 1 is a bit below the average both in terms of age and education level. We can also denormalize the centroids in order to obtain the values in the original measure, as explained in Section 4.3. As an example, the centroid of cluster 2 has for the age $1.94 \times 16.19 + 44.5 = 75.90$ years and as the education level $-0.38 \times 1.31 + 3.6 = 3.10$.

Setting the hyper-parameters We still do not know how to set the hyper-parameters. The distance measure is easy: it depends on the characteristics of the data, a subject previously discussed. What about the value of K? How can we set it?

Let us start with a question. If the number of clusters is equal to the number of instances, what will be the within-groups sum of squares (see Equation (5.7))? Zero! Why? Because each instance will be a centroid of a different cluster. See, for instance, the position of Dennis and C1 in Figure 5.7a–f. So, the more centroids we have, typically, the lower is the within-groups sum of squares. But, nobody aims to have only clusters of size 1, because it is meaningless in terms of data description. To have a single cluster is not desirable either, obviously. If you calculate the within-groups sum of squares (some software packages can do it) for different values of K and you do a line plot, it will be like the one in Figure 5.9.

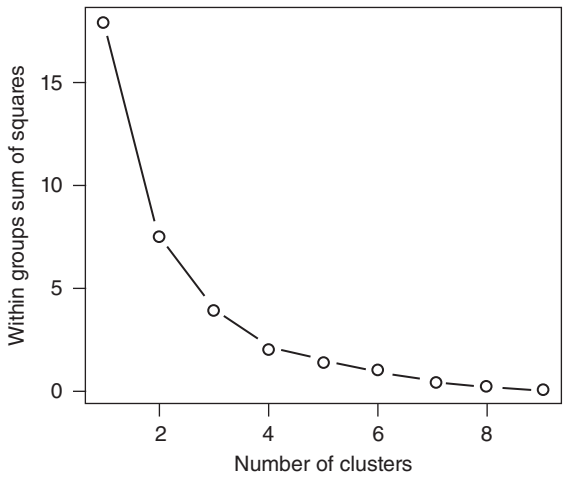


Figure 5.9 Variation of within-groups sum of squares with number of clusters.

Table 5.5 Advantages and disadvantages of k-means.

Advantages	Disadvantages
• Computationally efficient • Good results obtained quite often: global optima	• Typically, each time we run k-means the results are different due to the random initialization of the centroids text • Need to define number of clusters in advance • Does not deal with noisy data and outliers • k-means can only find partitions whose clusters have convex shapes

This is known as the elbow curve. The reason is obvious, although this elbow is somewhat round…The K value you should use is the one for which the curve is approximately straight and horizontal; that is, where the elbow is. In the example, this would be $K = 4$ (or 3). This is a simple but effective way of defining the K value.

Table 5.5 summarizes the key advantages and disadvantages of k-means.

5.3.2 DBSCAN

Like k-means, DBSCAN (density-based spatial clustering of applications with noise) is used for partitional clustering. In contrast to k-means, DBSCAN automatically defines the number of clusters. DBSCAN is a density-based technique, defining objects forming a dense region as belonging to the same cluster. Objects not belonging to dense regions are considered to be noise.

The identification of dense regions is done by first identifying the core instances. A core instance p is an instance that directly reaches a minimum number of other instances. This minimum number is a hyper-parameter. To be considered "directly reachable" an instance q must be at a lower distance from p than a predefined threshold (denoted by epsilon). Epsilon is another hyper-parameter, and is often termed the radius. If p is a core instance, then it forms a cluster together with all instances that are reachable from it, directly or indirectly. Each cluster contains at least one core instance. However, non-core instances can be part of a cluster at its edge, since they cannot be used to reach more instances. DBSCAN also has some randomization due to the necessity of deciding to which core instance a given instance will be attached when there is more than one core instance that can reach it directly. However, despite this, the randomization does not typically have a meaningful impact in the definition of the clusters.

An analysis of our example with DBSCAN is presented in Figure 5.10, using as hyper-parameters the radius as represented in the figure and for a minimum number of instances of two. We can see the existence of one cluster (Carolina, Fred, Gwyneth, Hayden, Irene and James) and four outliers (Andrew, Bernhard, Dennis and Eve).

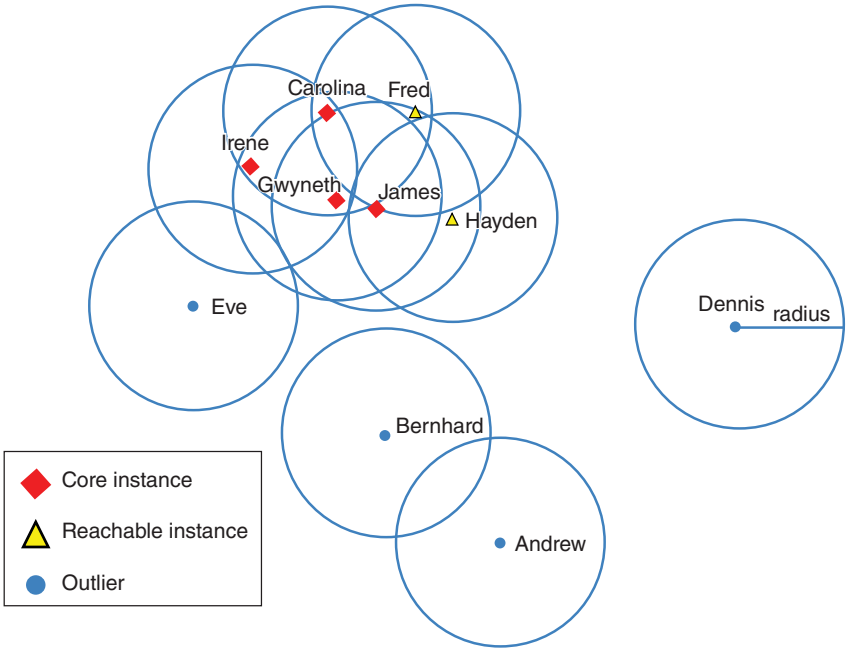


Figure 5.10 DBSCAN for the example data set using a minimum number of instances of two.

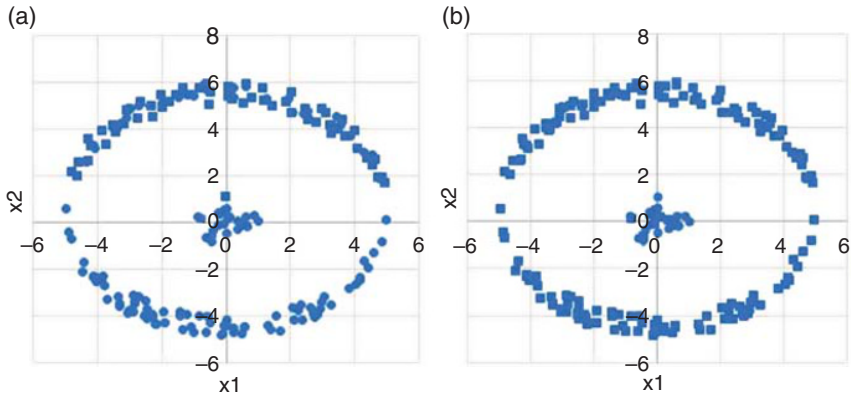


Figure 5.11 Comparison of k-means and DBSCAN: (a) k-means with $k = 2$; (b) DBSCAN with minimum number of instances of 3 and epsilon of 2.

To better visualize how different k-means and DBSCAN are, we can see in Figure 5.11 a different data set that better shows the differences. For the k-means, we have defined the number of clusters as two, while for DBSCAN we used three as the minimum number of instances and set epsilon to two.

Assessing and evaluating results Results of DBSCAN do not have any special characteristic. There are no centroids in DBSCAN. The instances of each cluster can be assessed.

Setting the hyper-parameters The main hyper-parameters of DBSCAN are the minimum number of instances that are necessary to classify a given instance as core, and the maximum distance an instance should be to be considered directly reachable. Despite some research in order to set these hyper-parameters, the common practice is to test different values and analyze whether the results obtained are acceptable taking into consideration the number of clusters and outliers observed. The distance measure is also a hyper-parameter that should be chosen depending on the scale type of the attributes.

Table 5.6 summarizes the key advantages and disadvantages of DBSCAN.

5.3.3 Agglomerative Hierarchical Clustering Technique

Hierarchical algorithms construct clusters progressively. This can be done by starting with all instances in a single cluster and dividing it progressively, or by starting with as many clusters as the number of instances and joining them up step by step. The first approach is top-down while the second is bottom-up.

The agglomerative hierarchical clustering method is a bottom-up approach. We will use our example to present this method. First we need to calculate the

Table 5.6 Advantages and disadvantages of DBSCAN.

Advantages	Disadvantages
• It can detect clusters of an arbitrary shape • Robust to outliers	• Each time we run DBSCAN the results can be a bit different due to some randomness, but the results typically do not vary much between different runs • Computationally more complex than k-means • Difficulty in setting the hyper-parameter values

Table 5.7 First iteration of agglomerative hierarchical clustering.

	A	B	C	D	E	F	G	H	I	J
A	0									
B	1.07	0								
C	3.26	2.33	0							
D	2.26	2.53	3.17	0						
E	2.6	1.54	1.63	3.65	0					
F	3.11	2.31	0.56	2.7	1.98	0				
G	2.67	1.71	0.62	2.87	1.2	0.79	0			
H	2.32	1.59	1.11	2.12	1.78	0.8	0.76	0		
I	3.13	2.1	0.63	3.47	1.06	1.12	0.6	1.35	0	
J	2.51	1.61	0.76	2.61	1.36	0.73	0.26	0.5	0.86	0

distance between all pairs of instances. To do this, we will use normalized data and the Euclidean distance. The diagonal is always zero because the distance between an instance and itself is always zero. Only half of the matrix is presented because the other half is symmetric. For instance, the distance between Andrew and Bernhard is equal to the distance between Bernhard and Andrew. So, half of the matrix is not necessary (see Table 5.7).

Next you find the pair with shortest distance. In this case this is the pair Gwyneth–James. The next step is to create another matrix with one row and one column less. Now, Gwyneth and James will be treated as a single object GJ (see Table 5.8). One question remains. How can we calculate the distance of any instance to GJ? In this example we do it using the average linkage criterion. For instance, the distance between Eve and GJ is the average of the distances between Eve and Gwyneth (1.20) and Eve and James (1.36); that is, 1.28.

The process continues by removing, at each step, one row and one column from the matrix due to the aggregation of two cells. It ends when the size of the

Table 5.8 Second iteration of agglomerative hierarchical clustering.

	A	B	C	D	E	F	GJ	H	I
A	0								
B	1.07	0							
C	3.26	2.33	0						
D	2.26	2.53	3.17	0					
E	2.6	1.54	1.63	3.65	0				
F	3.11	2.31	0.56	2.7	1.98	0			
GJ	2.59	1.66	0.69	2.74	1.28	0.76	0		
H	2.32	1.59	1.11	2.12	1.78	0.8	0.63	0	
I	3.13	2.1	0.63	3.47	1.06	1.12	0.73	1.35	0

matrix is two (two rows and two columns). So, for a problem with n instances, there will be $n - 1$ steps.

In the example above, the distance between Eve and GJ was calculated as the average of the distances between Eve and Gwyneth and Eve and James. Other methods exist, as we will see next.

5.3.3.1 Linkage Criterion

Given two sets of instances, how can we calculate how distant the two sets are? We present four of the most used linkage criteria:

- *The single linkage:* This measures the distance between the closest instances, one from each set. It favors the appearance of a dominant cluster (Figure 5.12a).
- *The complete linkage:* This measures the distance between the most distant instances, one from each set. It favors similar clusters (Figure 5.12b).
- *The average linkage:* This measures the average distance of every pair of instances, each instance of a pair from each set. It is in between the two previous approaches (Figure 5.12c).
- *The Ward linkage:* This measures the increase of the total within-cluster variance after merging. It favors compact clusters.

Let us see how these four linkage criteria behave with our data set (see Figure 5.13).

It is worth noting that the single and average linkage criteria are not too different. The main difference is with Irene. The complete and the Ward criteria are also relatively similar. The major difference again concerns Irene. However, the single and average criteria exhibit more meaningful differences when compared to the complete and Ward criteria. Indeed, for the latter two criteria,

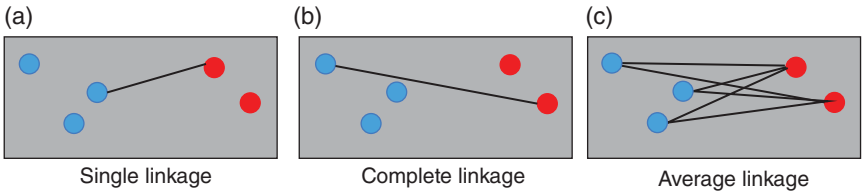


Figure 5.12 Single, complete and average linkage criteria.

there are two well-defined clusters, while with the former two there is an outlier (Dennis).

But to better understand Figure 5.13, let us see now how to read dendrograms. This is an important subject in hierarchical clustering.

5.3.3.2 Dendrograms

Using the minimum distance from each matrix (0.26 and 0.56 are the values for the first and second matrices), a so-called dendrogram can be drawn, as shown in the Figure 5.14. The horizontal branch that unifies Gwyneth and James has a value of 0.26 for the height – the value for the average linkage – while the branch that unifies Carolina and Fred has a value of 0.56 for the height.

From the dendrogram it is very easy to obtain the desired number of clusters. If we want four clusters, we should look at the $4 - 1 = 3$ highest horizontal branches. These are the branches that define four clusters: Dennis is in the first cluster, Andrew and Bernhard in the second, Eve in the third and all others in the fourth. As you can see, it is very easy to interpret this kind of plot. But you should always be aware that the similarity defined by clustering depends strongly on how the distance measure "captures" the concept of dissimilarity.

One of the advantages of dendrograms is that they make identification of outliers easy. These are the cases where the highest branch (largest distance) has only one element on one side and all the others on the other side. This is the case for Dennis, mainly because he is much older than the others.

Assessing and evaluating results Dendrograms are the main tool to analyze the results of hierarchical clustering. There is also the possibility to generate a given number of clusters from the dendrogram. Using this possibility, the instances of each cluster can be assessed, something that a dendrogram already does in an efficient way for data sets that are not too large.

Setting the hyper-parameters The main hyper-parameters for agglomerative hierarchical clustering are the distance measure and the linkage criterion. Both of thse have already been presented, in Sections 5.1 and 5.3.3.1.

Table 5.9 summarizes the main advantages and disadvantages of agglomerative hierarchical clustering.

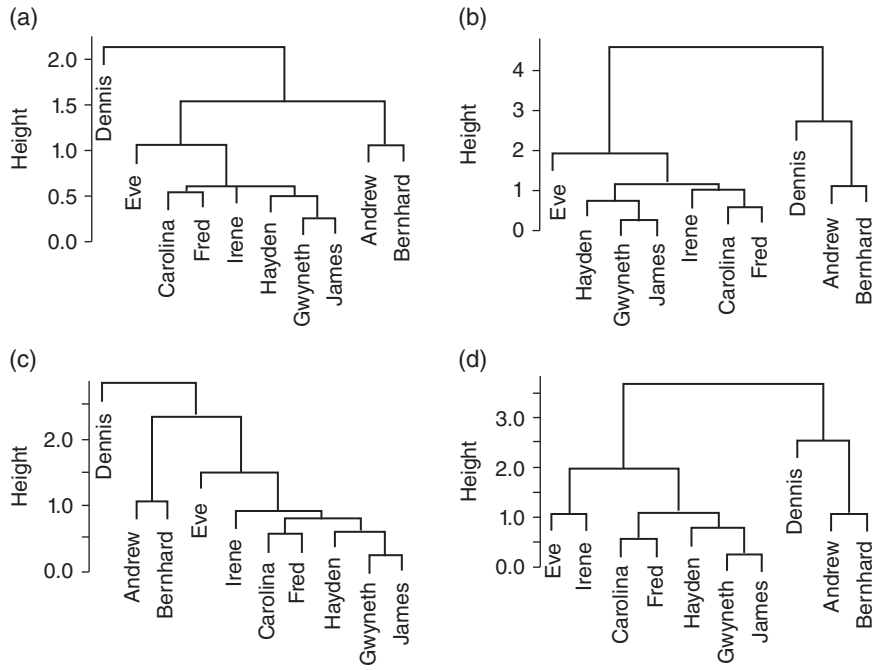


Figure 5.13 Effect of different linkage criteria on sample data set: (a) single; (b) complete; (c) average; (d) Ward.

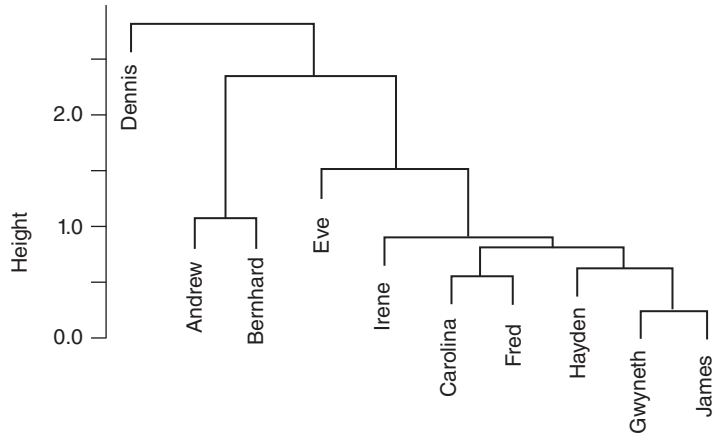


Figure 5.14 Dendrogram defined by agglomerative hierarchical clustering using the average as linkage criterion.

Table 5.9 Advantages and disadvantages of agglomerative hierarchical clustering.

Advantages	Disadvantages
• Easily interpretable, but more confusing for large datasets • Setting of hyper-parameter values is easy	• Computationally more complex than k-means • Interpretation of dendrograms can be, in several domains, quite subjective • Often gives local optima: a typical problem in searching methods, as seen in Section 4.5.2.4.

5.4 Final Remarks

This chapter has presented a common technique used in data analytics, namely cluster analysis. Research on clustering techniques covers a wide range of issues. Some of the current trends in this area are presented here.

One area of current research is the to create clusters by simultaneously grouping the objects and the attributes. Thus, if a data set is represented by a table, such as Table 5.1, while the application of a clustering technique would group the rows (instances/objects), the application of a biclustering technique to this data set would simultaneously group the rows and the columns (attributes). Each bicluster produced is a subset of the rows that have similar values for a subset of columns, or a subset of columns that have similar values for a subset of the rows.

In several real applications, data is continuously generated, producing a data stream. The application of clustering techniques to a data stream at different time instants can produce different partitions. In particular, it is possible that there will be changes in the size and shape of clusters, merging or division of existing clusters, and inclusion of new clusters. A clustering technique used in data stream mining must efficiently deal with the increasing amount of data in terms of processing and memory use.

One particular technique, BIRCH (balanced iterative reducing and clustering using hierarchies), has been designed to fulfill these requirements and has been successfully used for data stream mining. BIRCH has two phases. In the first phase it scans the dataset and incrementally builds a hierarchical data structure, called the CF-tree (clustering feature tree), in which each node stores statistics extracted from the associated cluster, instead of all the data in the cluster, saving computer memory. In the second phase, it applies another clustering algorithm to subclusters in the leaf nodes of the CF-tree.

Some clustering techniques, such as k-means, can produce different partitions in each run because they generate the initial centroids randomly. The partitions obtained using different clustering techniques are typically different. Even if these partitions have different values for validation indexes, they can extract relevant aspects of the data. The ideal situation would be to

find a pattern or "consensus" in the partitions. The use of ensemble techniques allows the combination of different partitions into a consensus partition, for example, trying to keep instances that appear together in most of the partitions in the same cluster.

A high number of attributes, some of them correlated, irrelevant or redundant is a common problem in real data sets. Their presence can increase the computational cost and reduce the quality of the solutions found. Feature selection techniques can determine the most relevant features, resulting in lower costs and better solutions. Although apparently simple, feature selection is one of the key challenges in data analysis. In supervised tasks, the class label can guide the feature selection process. In unsupervised tasks, such as cluster analysis, the lack of knowledge regarding class labels makes the identification of the relevant features in the feature selection process harder. The features selected have a strong influence on the partition found.

It is possible to improve the quality of the partitions found by clustering techniques if we use external information, such as which objects should be (or should not be) in the same cluster. For example, if you know the class label for some objects in the data set, the labeled objects in the same cluster should be from the same class. The use of external information to help clustering is known as semi-supervised clustering. Several real tasks can benefit from the use of semi-supervised learning. One of these is anomaly detection, which looks for objects in a data set partition that differ from most other objects in the same cluster.

5.5 Exercises

1 Using the social network dataset, run the k-means algorithm for different values of k ($k = 2, k = 3$ and $k = 5$).

2 Using the social network dataset, run the DBSCAN algorithm and test different values for the two main hyper-parameters.

3 Using the social network dataset, run the agglomerative hierarchical algorithm and plot the dendrogram generated.

4 What happens if you change the order in which you present the data to the k-means algorithm? Use the social network dataset to find out.

5 Calculate the edit distance between "composition" and "conception".

6 Analyse the dendrogram in Figure 5.15, which is for data obtained by a political party in the United States of America over the course of several elections:

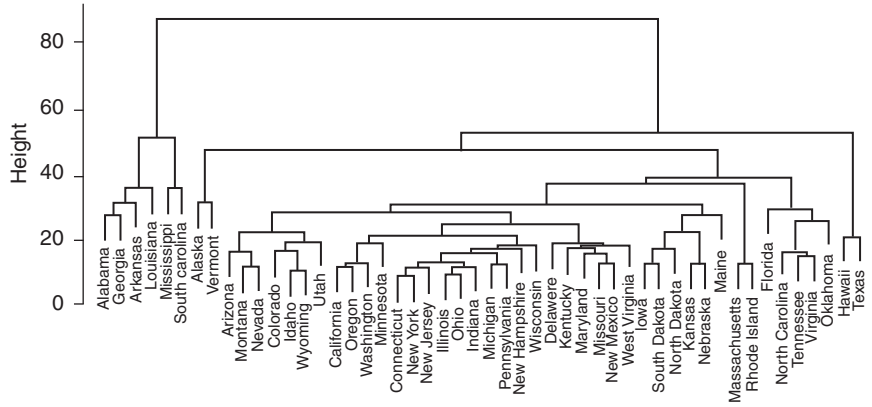


Figure 5.15 Dendrogram.



Figure 5.16 Space invaders.

a) Which state has election results for the party that are more like:
   i) Michigan?
   ii) Wisconsin?
b) Consider only the states from Alaska to Texas. What are the members of each cluster if you want:
   i) 3 clusters?
   ii) 8 clusters?

7 Represent the images of "space invaders" in Figure 5.16 by quantitative vectors and compute the distances between them (i.e. their similarity) using the Euclidean, Manhattan and Hamming (or edit) distance measures.

8 Some clustering algorithms produce partitions whose clusters have arbitrary shapes, while other algorithms restrict the shape of the clusters formed. Describe one advantage and one disadvantage of each group of algorithms compared to the other groups.

9 Write three short texts consisting of at least three sentences. Represent these texts in a bag-of-words format and compute their distances according to this format.

6

Frequent Pattern Mining

In the previous chapter, we showed how to assign objects (friends) with similar attributes (age and education level) to certain sets (clusters). Let us think of the following, somehow inverse, questions:

- Can we find combinations of attributes that are common to many objects?
- Are there any significant (or confident) associations between these combinations?

To put these questions in the context of our example, we might ask if we are able to find combinations of cuisine types that are enjoyed by many of our friends and if these combinations can be somehow assigned a certain confidence. To this end, let us extend our data with information about the preferred cuisine of each person, as shown in Table 6.1.

Before discussing these combinations of attributes or how to measure the confidence level of associations between them, let us take a look on the data in the Table 6.1. At first sight it looks like a normal "object–attribute" table, similar to Table 1.1. However, there are some differences. First, the attributes "age" and "educational level" in Table 1.1 have simple and well-defined domains – numbers – and directly characterize a person. In contrast, the columns in Table 6.1 represent concepts – in our case cuisine types – that can have more complex definitions and domains. We will call these concepts "items". Second, the values in the cells of Table 6.1 do not represent concrete attribute values of objects, but rather the "presence" of a connection between objects (people) and items (cuisine types).

Such data types are common in commercial domains, where each row is called a transaction and represents a purchase or a market basket, while each item represents a product. Non-empty cells, containing a tick mark, indicate that a given item is somehow connected to the given transaction; say that it was ordered or bought. The common name for this type of data is "transactional data" and it is represented in a so-called sparse form, as illustrated in Table 6.2, because the number of items connected to a single transaction is

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

Table 6.1 Data about the preferred cuisine of contacts.

Name	Arabic	Indian	Mediterranean	Oriental	Fast food
Andrew		✓	✓		
Bernhard		✓		✓	✓
Carolina		✓	✓	✓	
Dennis	✓		✓		
Eve				✓	
Fred		✓	✓	✓	
Gwyneth	✓		✓		
Hayden		✓		✓	✓
Irene		✓	✓	✓	
James	✓		✓		

Table 6.2 Transactional data created from Table 6.1.

Transaction id	Items connected to the transaction
Andrew	Indian, Mediterranean
Bernhard	Indian, Oriental, Fast food
Carolina	Indian, Mediterranean, Oriental
Dennis	Arabic, Mediterranean
Eve	Oriental
Fred	Indian, Mediterranean, Oriental
Gwyneth	Arabic, Mediterranean
Hayden	Indian, Oriental, Fast food
Irene	Indian, Mediterranean, Oriental
James	Arabic, Mediterranean

generally much lower than the number of all available items. The same type of data can be found in other domains, too. For example, in recommendation systems, discussed in Chapter 13, rows (transactions) would represent users and columns (items) would represent movies, while a tick mark would be interpreted as meaning the given user and movie are connected; that is, that they have seen, liked or bookmarked it.

The typical tasks of frequent pattern mining introduced in this chapter are:

- *finding frequent itemsets:* this aims to find "itemsets" (see below) that appear together in different transactions
- *finding association rules:* which aims to find interesting relations between itemsets.

We will also briefly describe another task:

- *finding frequent sequences:* which aims to find frequent item sequences, not necessarily contiguous, that appear in the same order.

Frequent pattern mining methods were developed to deal with very large data sets recorded in hypermarkets (tens of thousands of items and millions of transactions) and social media sites (millions of users and videos), just to name two examples. For each of the tasks previously identified, the different methods that exist are usually classified by how efficiently they give their results. The results themselves, however, are expected to be the same, independent of the method used.

6.1 Frequent Itemsets

An arbitrary combination of items is called an "itemset". It is, in essence, an arbitrary subset of the set $\mathcal{I}$ of all items. Let us think a little about the number of possible itemsets (combinations of items). In total, there are $2^{|\mathcal{I}|} - 1$ possible itemsets where $|\mathcal{I}|$ is the number of items in $\mathcal{I}$.

Example 6.1 In our example, $\mathcal{I} = \{$Arabic, Indian, Mediterranean, Oriental, Fast food$\}$ is the set of all of five items considered, so $|\mathcal{I}| = 5$. The subsets $\{$Fast food$\}$, $\{$Indian, Oriental$\}$ and $\{$Arabic, Oriental, Fast food$\}$ are itemsets of size 1, 2 and 3, respectively. The number of all possible itemsets, created from items in $\mathcal{I}$, of length 1 is five, of length 2 and 3 is ten and ten, respectively, while there are five itemsets of length 4 and one itemset of length 5. In total, there are $5 + 10 + 10 + 5 + 1 = 31 = 32 - 1 = 2^5 - 1$ itemsets we can generate from items in $\mathcal{I}$.

First of all, let us define a measure to express the frequency of an itemset in a transactional data, such as Table 6.2, denoted here as $\mathcal{T}$. Such a measure is called "support" and is computed as the ratio between the number of transactions (rows in $\mathcal{T}$) in which the given itemset is present and the number of all transactions in the data.

Example 6.2 The support of the single item "Fast food" in our data $\mathcal{T}$, in Table 6.2, is 2 or 0.2 or 20% since it occurs in two rows out of ten. In other words, the ratio of the number of transactions containing "Fast food" to the number of all transactions is $\frac{2}{10} = 0.2$, which is $0.2 \times 100 = 20\%$. Support for the combination of Indian and Oriental cuisine – the itemset $\{$Indian, Oriental$\}$ – is 5 or 0.5 or 50%; that is, five transactions out of ten contain both the Indian and the Oriental items. We will use the absolute frequency to express the support of itemsets in this chapter – say, 2 or 5 or any other number – but, you should keep in mind that this decision is application dependent and some software

Table 6.3 Combinatorial explosion with growing size of $\mathcal{I}$.

Number of items in $\mathcal{I}$	Number of possible itemsets	Expected runtime
5	31	31 ms
10	1 023	>1 s
20	1 048 576	>17 min
30	1 073 741 823	>12 days
40	$> 10^{12}$	>34 years
50	$> 10^{15}$	>35 millennia

tools may work with fractional or decimal supports, for example 0.2, 0.5 and so on; in other words, relative frequencies.

Frequent itemset mining Given a set of all available items $\mathcal{I}$, transactional data $\mathcal{T}$ and a threshold value *min_sup*, frequent itemset mining aims at finding those itemsets, called "frequent itemsets", generated from $\mathcal{I}$, in which support in $\mathcal{T}$ is at least *min_sup*.

Now, we are able to come up with a so-called "naïve algorithm" to find all frequent itemsets in the transactional data. All we need to do is to generate all the different itemsets from $\mathcal{I}$, count their support in $\mathcal{T}$ and filter out those itemsets for which support is below a pre-defined *min_sup* threshold. While this algorithm would be acceptable in cases when $\mathcal{I}$ contains only a very small number of items, it would suffer from a so-called "exponential explosion" as the number of items grew too large; in other words, the computational cost would be unbearable.

Example 6.3 Let the computational time for generating one itemset and counting its support be 1 ms. For five items ($|\mathcal{I}| = 5$) the naïve algorithm needs to generate 31 itemsets and count their support, so the run time would be 31 ms. The number of different itemsets and the expected runtime for various sizes of $\mathcal{I}$ is depicted in Table 6.3.

6.1.1 Setting the *min_sup* Threshold

Let us discuss the *min_sup* threshold, a hyper-parameter with high importance, which has to be set carefully by the user according to their expectations of the results:

- Setting it to a very low value would give a large number of itemsets that would be too specific to be considered "frequent". These itemsets might apply in too few cases to be useful.

- On the other hand, very high values for *min_sup* would give a small number of itemsets. These would be too generic to be useful. Thus, the resulting information would probably not represent new knowledge for the user.

Another important aspect of the *min_sup* value is whether the number of frequent itemsets that results is small enough for subsequent analysis. Would you like to analyze tens of thousands of itemsets? In Section 6.3, some issues regarding the quality of patterns will also be discussed.

Example 6.4 All itemsets generated from $\mathcal{I}$ = {Arabic, Indian, Mediterranean, Oriental, Fast food}, numbered and organized into a so-called "lattice", are illustrated in Figure 6.1. Each itemset is connected to the subset(s) positioned above it and to the superset(s) positioned below it. The uppermost itemset (with the number 0) is an empty set, which should not be considered an itemset. It is introduced into the lattice only for the sake of completeness. For each itemset, the corresponding transaction identifiers – the names of people from Table 6.2 – are listed too, indicating the support of the given itemset.

There are thirteen itemsets with support greater or equal to 2, and nine itemsets with support greater or equal to 3 (the darker shades of grey in the figure). On the other hand, there are no itemsets with support larger than 7.

Monotonicity The *min_sup* threshold controls the number of resulting frequent itemsets, but is there any way to avoid having to process (generate and test the support of) all the possible itemsets? Fortunately, there are two very simple but effective theorems to help us. These theorems are called the monotonicity theorems or monotonicity rules and say that:

1. If an itemset is frequent then each of its subsets are frequent too:

Example 6.5 Let the itemset {I,M,O}in Figure 6.1, numbered 22 and with support equal to 3, be frequent. Naturally, each transaction containing this itemset is also a transaction that contains each of the subsets of this itemset. This means that the support of each itemset {I,M}, {I,O}, {M, O}, {I}, {M} and {O} (numbered 10, 11, 13, 2, 3 and 4, respectively, is at least 3, too. In fact, the supports of these itemsets are 4, 5, 3, 6, 7 and 6, respectively, all being greater or equal to 3. In other words, if we have a frequent itemset then all the itemsets connected to it through a pathway up in the lattice are frequent, too, with support equal or greater than the support of the given itemset.

2. If an itemset is infrequent then none of its supersets will be frequent:

Example 6.6 Let the itemset {I,F} in Figure 6.1, numbered 12 and with support 2, be infrequent. If we add any other item to this itemset then, obviously,

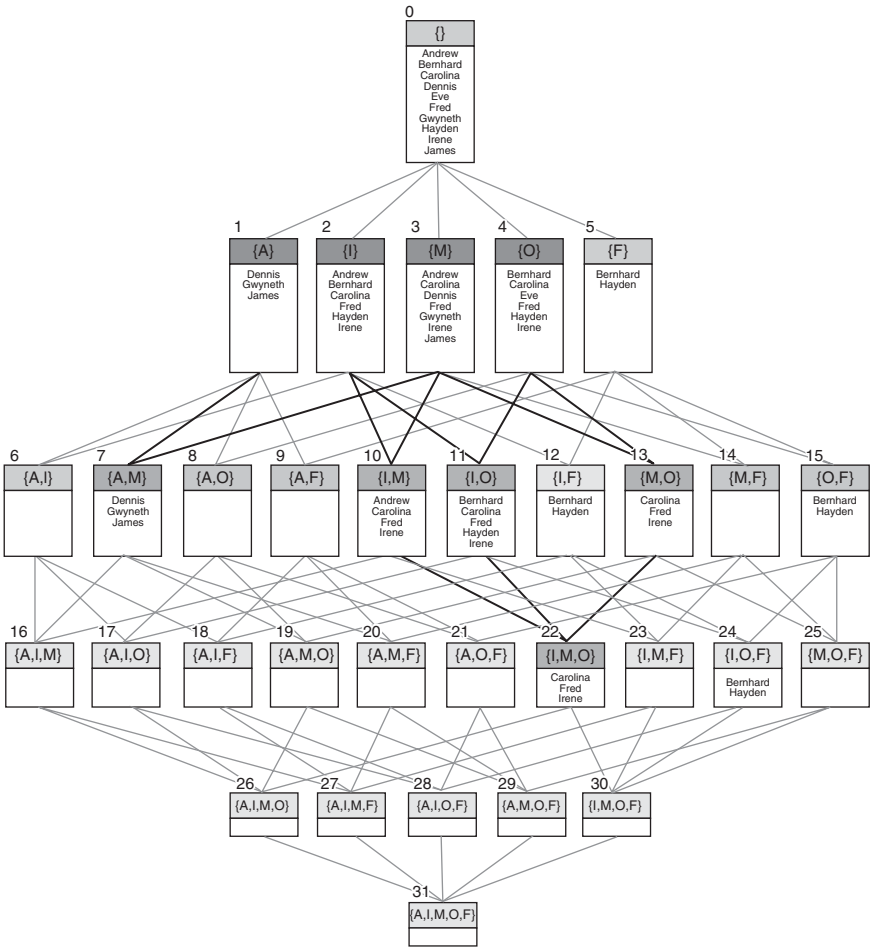


Figure 6.1 Itemsets organized into a lattice according to subset-superset relations. Members of $\mathcal{I}$ are abbreviations for the five cuisine types introduced in Table 6.2, i.e. A for the Arabic cuisine, I for Indian, etc.

fewer transactions will contain that expanded itemset; that is, the support of the expanded itemset will be at most 2. All the supersets of this itemset, reachable from it through a pathway down in the lattice, are infrequent too. These are the itemsets numbered 18, 23, 24, 27, 28, 30 and 31, all with support less than or equal to 2.

A general principle of frequent itemset mining methods is to traverse the itemset lattice (illustrated in Figure 6.1) in order to search for those itemsets with support greater or equal than a predefined *min_sup* threshold. The various methods developed differ in how effectively they traverse through

the lattice of all possible itemsets and how they represent those itemsets. In addition, different implementations utilize various heuristics to optimize the computation time and memory usage. The results themselves, however, are expected to be the same, independent of the method used. In the next subsections three of the main methods are introduced.

6.1.2 Apriori – a Join-based Method

The oldest and most simple technique of mining frequent itemsets involves the generic, so-called "join-based" principle, as set out below.

Example 6.7 Figure 6.2 shows the Apriori principle for our dataset for a minimum support threshold $min_sup = 3$. In the first step of the algorithm, the support of each itemset of length $k = 1$ is computed, resulting in four frequent and one non-frequent itemsets. In the next step, itemsets of length $k = 2$ are generated from the frequent itemsets of length $k = 1$, so no itemset containing item F is considered. Step 2 results in four frequent itemsets, which are used to generate itemsets of length $k = 3$, in the following step.

Algorithm Apriori.

```
1: INPUT T the transactional dataset
2: INPUT min_sup the minimum support threshod
3: Set k = 1
4: Set stop = false
5: repeat
6:     Select all frequent itemsets of length k (with support at least min_sup)
7:     if there are no two frequent itemsets of length k then
8:         stop = true
9:     else
10:        Set k = k + 1
11: until stop
```

One heuristic worth noticing is that only those itemsets of length $k = 2$ are merged in this step, which result in itemsets of length $k + 1 = 3$. For example, merging itemsets {I, M} and {I, O} of length 2 results in {I, M, O} of length 3, so these are merged in this step; {A, M} and {I, O} are not merged since this would result in the itemset {A, I, M, O} of length 4, not 3. Another heuristic is that itemsets are only generated as candidates such that each subset of it is a frequent itemset. For example, itemsets {A, M} and {I, M} are not considered for merging since the resulting itemset {A, I, M} would be a superset of {A, I}, which is not a frequent itemset.

The third iteration results in only one frequent itemset. Since there are no two itemsets of length 3 to merge, the computation stops. There were nine frequent itemsets found for the minimum support threshold $min_sup = 3$.

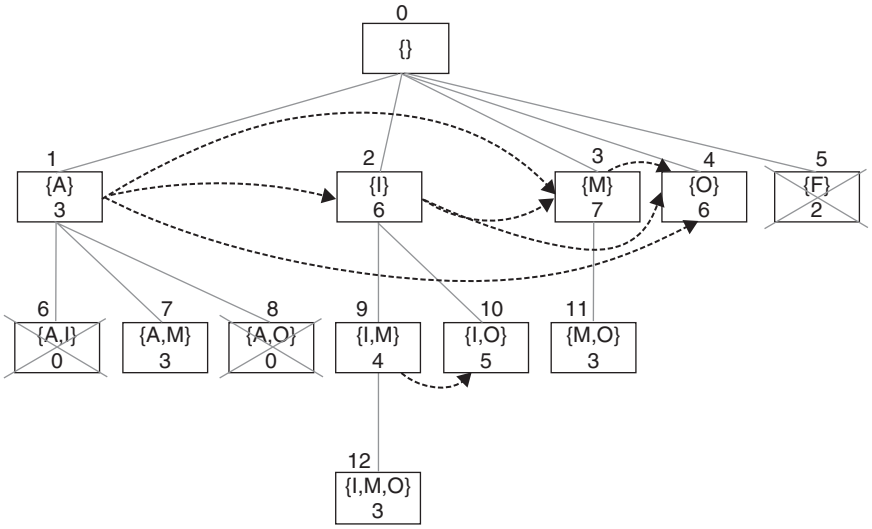


Figure 6.2 The Apriori principle on the data from Table 6.2 (or 6.1), illustrated using an enumeration tree.

As the example shows, by utilizing the two monotonicity theorems mentioned above (all subsets of a frequent itemset are frequent and none of the supersets of a not-frequent itemset is frequent), the amount of computation can be reduced by a large amount. The itemsets that it makes sense to generate and measure the support of in each iteration of the algorithm are called "candidate itemsets". As illustrated in the example above, there were twelve candidate itemsets generated, of which nine turned out to be frequent for the threshold value $min_sup = 3$. Without utilizing the monotonicity theorems, we would have had to generate $2^5 - 1 = 31$ candidate itemsets and check their support. With Apriori, we processed only 12 of them, which is a reduction of more than 60% in the computation time.

The structure used in Figure 6.2 to illustrate the principle of Apriori is called an "enumeration tree". The enumeration tree contains the candidate itemsets ordered by their length (from top to bottom) and also lexicographically (from left to right). This ordering defines an ancestral relationship in which each "parent" itemset is a subset of its "child" itemsets and vice versa, each child itemset being a superset of its parent itemset.

In essence, all frequent itemset mining methods can be considered as variations of the Apriori method, using various strategies to explore the space of candidate itemsets defined by an enumeration tree.

Table 6.4 Transactional data from Table 6.2 in vertical format corresponding to the first iteration ($k = 1$) of the Eclat algorithm.

Item (id)	TID-set (names of persons, in our case)	Cardinality
Arabic	{Dennis, Gwyneth, James}	3
Indian	{Andrew, Bernhard, Carolina, Fred, Hayden, Irene}	6
Mediterranean	{Andrew, Carolina, Dennis, Fred, Gwyneth, Irene, James}	7
Oriental	{Bernhard, Carolina, Eve, Fred, Hayden, Irene}	6
Fast food	{Bernhard, Hayden}	2

6.1.3 Eclat

The main obstacle for the Apriori algorithm is that in every step it needs to scan the whole transactional database in order to count the support of candidate itemsets. Counting support is one of the bottlenecks for frequent itemset mining algorithms, especially if the database does not fit into the memory. There are many technical issues, not relevant here, for why counting support is computationally expensive if the database is large and does not fit into the memory.

But what if the database does fit into the memory? Even then, we have to consider all the transactions and see if the candidate itemset is a subset of the transaction; that is, if all the items from the candidate itemset are present in the transaction. This is not very effective, so other ways to represent transactional data are desirable to speed up the support counting process.

One way to store transactional data is the so-called "vertical format" in which, for every item, a list of transaction identifiers in which that item occurs is stored. Such a list is called a TID-set, the acronym standing for "transaction identifier". The vertical format for the records in Table 6.2 is illustrated in Table 6.4. In this case, the support of an itemset is counted as the ratio of the cardinality (length) of its TID-set to the number of transactions in the database (which is 10 in our example). The support of an itemset created by merging two other itemsets is the cardinality of the intersection of their TID-sets.

Example 6.8 Let us count the support of the itemset {Mediterranean, Oriental}, numbered 13 in Figure 6.1. The TID-set of this itemset is computed by intersecting the TID-set of {Mediterranean} and the TID-set of {Oriental} itemsets. This means that we have to intersect the sets {Andrew, Carolina, Dennis, Fred, Gwyneth, Irene, James} and {Bernhard, Carolina, Eve, Fred, Hayden, Irene}, resulting in a TID-set {Carolina, Fred, Irene} of cardinality 3. Thus the support of the itemset {Mediterranean, Oriental} is 3 or 0.3 (i.e. 3/10), which corresponds to 30%, as shown in the example above.

The principle of Eclat is similar to Apriori in that, first, the frequent itemsets of size $k = 1$ are detected and stored together with their corresponding TID-sets. Then, each frequent itemset of size k is systematically expanded by one item, resulting in an itemset of size $k + 1$. Then k is incremented and the process iterates until there are candidates to expand. The count of the support of candidate itemsets is, however, done by intersecting the TID-sets of the corresponding itemsets.

6.1.4 FP-Growth

If the database is very large, so that it hardly fits into the memory as it is, and the candidate patterns to be generated are long, then counting the support is costly and time consuming. An efficient approach called frequent pattern growth (FP-Growth), was developed to overcome these difficulties. Here we will introduce the method using our transactional data. The main strength of FP-Growth is that it compresses the transactional data into a so-called "FP-Tree" structure, in which support count and itemset generation is fast.

To build an FP-Tree, only two passes of the data are required. In the first pass, all the frequent items and their support are found. In our case, when $min_sup = 3$, these are the items A, I, M and O, with supports $3, 6, 7$ and 6, respectively. In the second pass of the data, items in each transaction are processed in a decreasing order according to their support, i.e. M first, followed by I and O, and finishing with A. Infrequent items are no longer taken into account. Items from the transaction are added to the tree following a path with respect to their order and the counts of the affected nodes are incremented. If a new node is added to the tree, its count is initialized at 1. Finally, a so-called "header table" of items is created, with pointers to the first appearance of these items. The pointers are linked and chained through all appearances of the corresponding items. For counting the support of a given item, we only need to sum the counts in the linked nodes.

Example 6.9 As an example, the process of building an FP-tree on the data from Table 6.2 is illustrated in Figure 6.3. Changes in each step are marked in *italic*. The final FP-tree and the header table are at the bottom. To count the support of an item O, we sum the counts in the three nodes linked and chained to the item O in the header table: the support of O is $3 + 2 + 1 = 6$. Similarly, the support of I is $4 + 2 = 6$, while the supports of M and A are 7 and 3, respectively.

A useful property of the FP-tree is that it contains complete information about frequent itemsets in the data while being compact: the tree is usually much smaller than the data set it was created from. Moreover, the number of transactions containing a given item can be obtained by following the pointers starting from the header table.

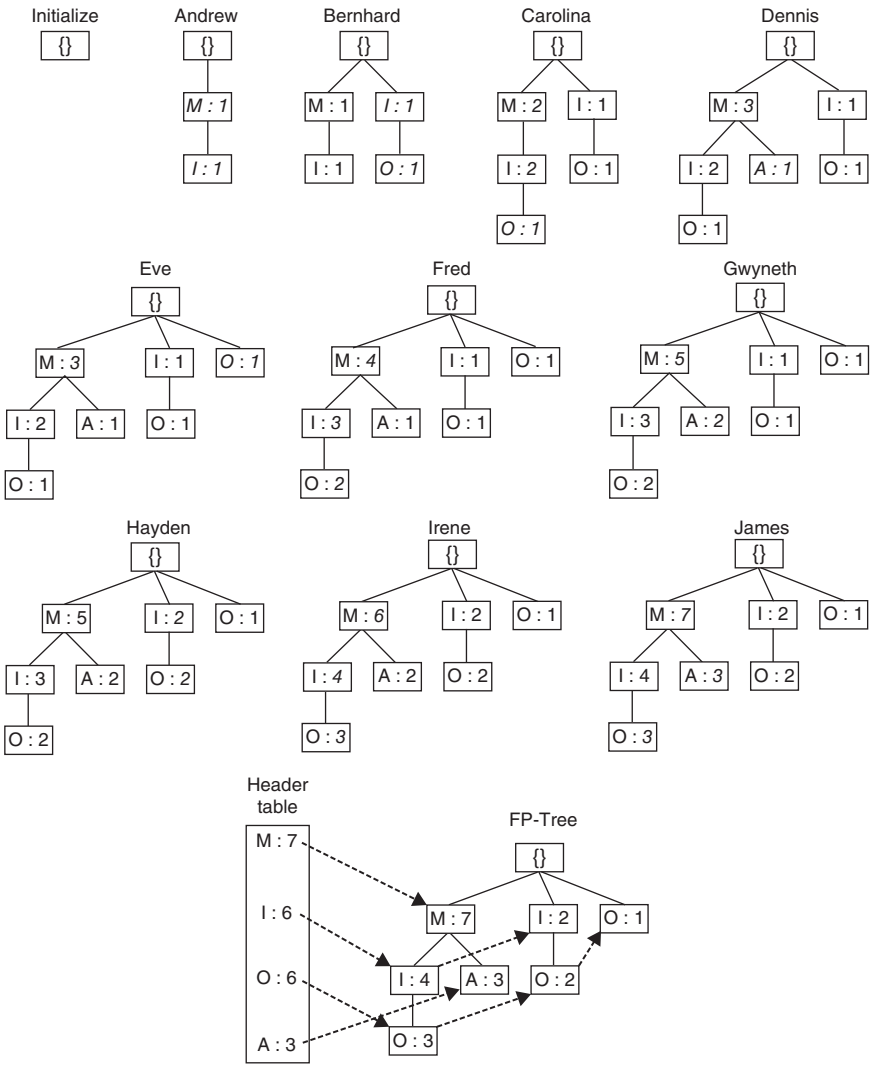


Figure 6.3 Building an FP-tree on the data from Table 6.2 (or 6.1). Each node contains an identifier of an item and its count in the given path.

The process of mining frequent itemsets is based on the observation that the set of all frequent itemsets can be divided into non-overlapping subsets. In our example, these are four subsets such as:

- frequent itemsets containing item A
- frequent itemsets containing item O but not containing item A

- frequent itemsets containing item I but not containing items A and O
- frequent itemsets containing item M but not containing items A, O and I, resulting, in our case, in a single itemset[1] containing only the frequent item M.

Given the decreasing order of items (with respect to their support) in itemsets, these four subsets correspond to itemsets ending in A, I, O and M, respectively. How FP-Growth searches for these four subsets is explained in the next example.

Example 6.10 Figure 6.4 illustrates the process of finding frequent itemsets, given $min_sup = 3$, ending in item O but not containing A.[2] FP-Growth starts at the leaves of the tree containing item O. First, it extracts all the paths in the tree ending in O by following the links from O in the header table and keeping only those nodes that are on direct paths from these leaves to the root node. The resulting subtree is called the prefix-path subtree for O.

Checking the support of O by summing the counts along the leaves gives $3 + 2 + 1 = 6$, so the itemset {O} is frequent and is added to the result. In the next step, the counts of the nodes in the prefix-path subtree are updated, reflecting the number of transactions containing O.

In the next step, as {O} was frequent, itemsets ending in O are searched for. For this reason, a so-called conditional FP-tree is created from the prefix-path subtree for O by simply cutting all the leaves with O and removing all the items the count of which, summed up by chasing the corresponding links from the header table, does not reach the *min_sup* threshold. In our case, the counts of all the resulting items I and M are greater or equal to 3, so no items are deleted. The resulting conditional FP-tree represents those itemsets that appear in transactions together with O, together with their support.

This procedure is applied recursively to the resulting conditional FP-tree of O: a prefix-path subtree for I is created and the counts are updated. The sum of the nodes containing I is $3 + 2 = 5$, so I is added to existing itemsets in the result, forming the frequent itemset {I,O}. Again, recursively, a prefix-path subtree for M is created in which the count for M is 3, so M is added to the existing frequent itemsets {O} and {I,O} resulting in new itemsets {M,O} and {M,I,O}, respectively. Since the conditional FP-tree for M is empty, the generation of itemsets ending in item O stops, resulting in frequent itemsets {M,I,O}, {M,O}, {I,O} and {O}.

---

1 At the end, following the main idea of the algorithm, only a single item, the most frequent one, will be included in the itemset found.

2 Itemsets ending in A should be looked for first. However, the second step of the main procedure, finding itemsets ending in O but not containing A, will be illustrated here.

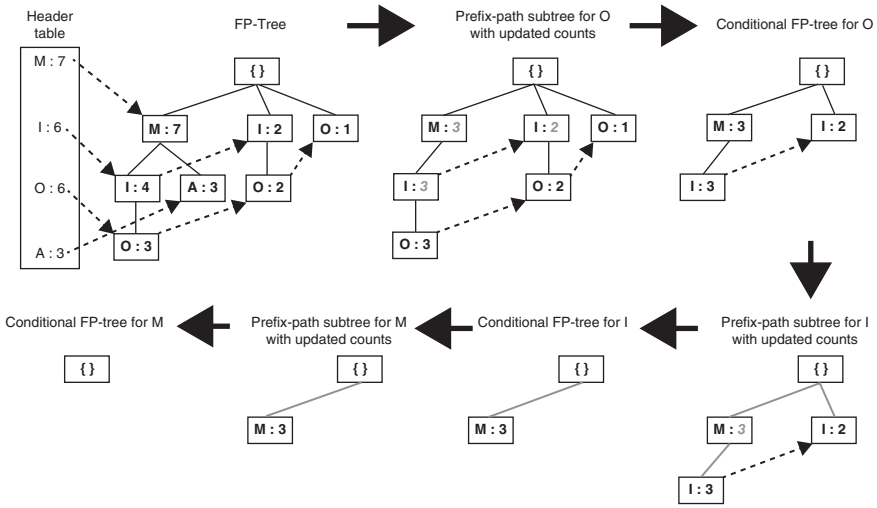


Figure 6.4 The FP-growth process of finding frequent itemsets ending in item O but not containing item A from the FT-Tree (Figure 6.3) built from the data in Table **??**.

6.1.5 Maximal and Closed Frequent Itemsets

In practice, frequent itemset mining, even in case of a reasonably set *min_sup* threshold, often results in a very large number of frequent itemsets, the processing of which is burdensome. However, some itemsets are more representative than others. These itemsets are the maximal and closed frequent itemsets. From them, all the other frequent itemsets can be derived.

Maximal frequent itemset A frequent itemset is maximal if none of its supersets is frequent.

Example 6.11 Maximal frequent itemsets are illustrated in Figure 6.5 by shading color. These are the itemsets {A,M} and {I,M,O}; there are no other frequent itemsets connected to them by a path on the way down in the itemset lattice. In other words, there are no frequent itemsets containing items A and M or I, M and O, respectively, together with other items. We can derive all the frequent itemsets by encountering all the different subsets of maximal frequent itemsets. From {A,M} we can derive the itemsets {A} and {M} while from {I,M,O} we can derive the itemsets {I,M}, {I,O}, {M,O}, {I}, {M} and {O}.

Closed frequent itemset A frequent itemset is closed if none of its supersets has the same support.

Example 6.12 Closed itemsets are illustrated with a solid edge in Figure 6.5. The itemset {A} is not closed since its superset {A,M} has the same support, 3. This means that if a transaction contains an item A then it also contains the item M. Similarly, {M,O} is not closed because it has a superset {I,M,O} with

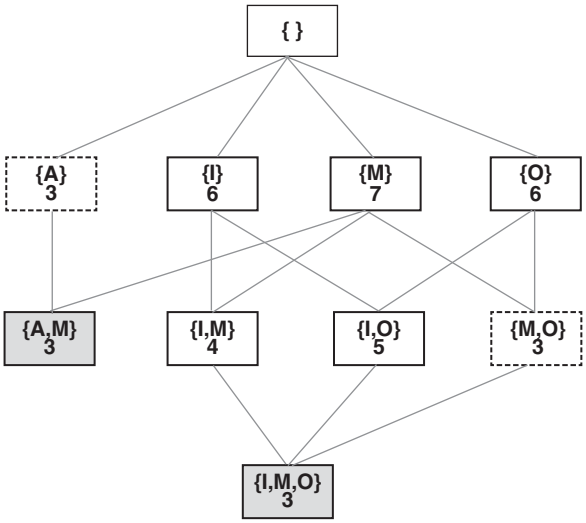


Figure 6.5 Maximal (shaded) and closed (with solid edges) itemsets generated from the data in Table 6.2.

the same support. We can see that if both items M and O occur in a transaction then the item I is also present in that transaction.

It is important to notice that, from the definitions, if an itemset is maximal then it is also closed. However, the opposite is not always true: not all closed itemsets are maximal itemsets.

6.2 Association Rules

In the example of closed frequent itemsets we mentioned that, given our data and the frequent itemsets found, if a transaction contains the items M and O then it implies that the item I is also present in that transaction. In other words, the itemsets {M,O} and {I} are associated in the data according to the "if–then" implication relation.

Association rule An association rule is an implication of the form $\mathcal{A} \Rightarrow \mathcal{C}$, where $\mathcal{A}$ and $\mathcal{C}$ are itemsets that do not share common items. $\mathcal{A}$ is called the antecedent of the rule and $\mathcal{C}$ is the consequent of the rule.

The meaning of an association rule is that if its antecedent is present in some transactions then its consequent should also be present in these transactions. Of course, such an implication usually does not hold absolutely in the data. Thus, we need some measures to express the strength of an association: how valid the rule is according to the data.

Similarly to frequent itemsets, one possible measure of the strength of an association rule $\mathcal{A} \Rightarrow \mathcal{C}$ is its support, a concept introduced for frequent itemsets. For an association rule, this is defined as the support of an itemset formed by joining the antecedent and the consequent of the rule. It is formally expressed as:

$$support(\mathcal{A} \Rightarrow \mathcal{C}) = support(\mathcal{A} \cup \mathcal{C}) \tag{6.1}$$

Example 6.13 According to the data (Table 6.1), the support of the rule {A} ⇒ { M} is the support of the itemset {A,M}, created by joining the antecedent and the consequent of the rule, which is equal to 3. It is important to notice that the support of the rule {A} ⇒ {M} is the same as of the rule {M} ⇒ {A}.

The support is a kind of a quantity or frequency measure, but it is not enough to measure the quality of a rule. The reason is that it gives the same value when we exchange the antecedent and the consequent of a rule. The quality, or the reliability, of a rule can be measured by its confidence, defined as the ratio of the support of the rule to the support of the antecedent of the rule:

$$confidence(\mathcal{A} \Rightarrow \mathcal{C}) = \frac{support(\mathcal{A} \cup \mathcal{C})}{support(\mathcal{A})} \tag{6.2}$$

Example 6.14 According to our transactional data, 3 out of 3 friends who like Arabic food also like Mediterranean food. However, not everyone who likes Mediterranean food also likes Arabic food. In other words, 3 from 7 friends who like Mediterranean food like also Arabic food. As we can sense, the quality of the rule {A} ⇒ {M} is bigger than that of the rule {M} ⇒ {A}, despite they have the same support in the data. The confidence of {A} ⇒ {M} is the support of {A,M} divided by the support of {A}; that is, $3/3 = 1$. The confidence of {M} ⇒ {A} is $support(\{A, M\})/support(\{M\}) = 3/7 = 0.43$.

Since we have already defined what an association rule is and introduced the two measures of support and confidence, we are able to define the problem of mining association rules.

Association rule mining Given a set of all available items $\mathcal{I}$, transactional data $\mathcal{T}$ and threshold values *min_sup* and *min_conf*, association rule mining aims at finding those association rules generated from $\mathcal{I}$ for which support in $\mathcal{T}$ is at least *min_sup* and confidence in $\mathcal{T}$ is at least *min_conf*.

The number of different association rules that can be generated from the data is $3^{|\mathcal{I}|} - 2^{|\mathcal{I}|+1} + 1$ [20]. This is a much larger number than the number of different itemsets we can generate from the data, which is $2^{|\mathcal{I}|} - 1$.

Example 6.15 For our data, with 5 different items, the number of different association rules is $3^5 - 2^{5+1} + 1 = 243 - 64 + 1 = 180$, while the number of different itemsets is only $2^5 - 1 = 32 - 1 = 31$. You can create a table similar to Table 6.3 for association rules.

Because the support of each rule has to meet the *min_sup* threshold, only those rules for which the joined antecedent and consequent parts form a frequent itemset are considered. In accordance with this constraint, the process of mining association rules consists of two phases:

1) Mining frequent itemsets that meet the *min_sup* threshold; this is the most computationally expensive part of the process in general.
2) Generating association rules meeting the *min_conf* threshold from the found frequent itemsets.

As discussed in previous sections, the number of frequent itemsets, given a wisely set *min_sup* threshold, is much smaller than the number of all itemsets. This means that the first step of association rule mining is helpful in reducing the search space. Even so, from each frequent itemset I, we can generate $2^{|I|} - 2$ possible association rules, where $|I|$ is the size of the itemset I. Generating all the different rules for each frequent itemset would be not effective though. However, similar to the monotonicity theorems introduced in Section 6.1,

there is a monotonicity theorem to help in generating association rules. It has been described by Tan *et al.* in the following terms [20]:

Montonicity for association rules If the confidence of the rule $X \Rightarrow Y - X$ is lower than *min_conf* then the confidence of all the rules $X' \Rightarrow Y - X'$, where X' is a subset of X, will be lower than *min_conf* as well. Here, $Y - X$ and $Y - X'$ means that we remove all those items from Y which are present in X and X', respectively.

Example 6.16 Let Y={I,M,O}, X={I,O}, and *min_conf* = 0.75. Then $Y - X = \{\text{I}, \text{M}, \text{O}\} - \{\text{I,O}\} = \{\text{M}\}$. The rule $X \Rightarrow Y - X$ is {I,O} ⇒ {M} with a confidence equal to support({I,O,M}) / support({I,O}) = 3 / 5 = 0.6. Thus the rule $X \Rightarrow Y - X$ does not meet the *min_conf* threshold. Given a subset $X' = \{\text{I}\}$ of X, the rule $X' \Rightarrow Y - X'$ is {I} ⇒ {M,O} with a confidence equal to support({I,M,O}) / support({I}) = 3 / 6 = 0.5. Similarly, for the other subset $X' = \{\text{O}\}$ of X, the rule $X' \Rightarrow Y - X'$ is {O} ⇒ {I,M} with a confidence equal to support({O,I,M}) / support({O}) = 3 / 6 = 0.5. Both of the generated rules of the form $X' \Rightarrow Y - X'$ have confidence lower or equal than the confidence of the rule $X \Rightarrow Y - X$.

In other words, the theorem says that if the confidence of the rule does not satisfy the *min_conf* threshold and we modify this rule by moving one or more items from its antecedent into its consequent, then the confidence of the modified rule will not satisfy the *min_conf* threshold, either. Using this theorem we can define a systematic algorithm to generate association rules for a given frequent itemset Z in the following way:

Example 6.17 The process of mining association rules from the frequent itemset Z = {I,M,O} from our example (Table 6.2) and the lattice structure of these rules is illustrated in Figure 6.6. The *min_conf* threshold is set to 0.75. In the first step, three rules are constructed: {M,O} ⇒ {I}, {I,O} ⇒ {M} and {I,M} ⇒ {O} with confidences of 1.0, 0.6 and 0.75 respectively. Since the confidence of the second rule is less than the *min_conf* threshold, this rule is erased and only the first and the third rules are returned. The items I and O, in the consequences of these two rules, are added to the set C_1, so C_1={{I},{O}} and k is set to 2. There is only one possible itemset V of size $k = 2$ that can be generated from the itemsets {I} and {O} of size $k - 1$ in the set C_1, namely the itemset V = {I,O}. Thus, we generate a rule $Z - V \Rightarrow V$; that is, {I,M,O} − {I,O} ⇒ {I,O}, which is {M} ⇒ {I,O}. The confidence of this rule (0.43) is less than the *min_conf* threshold, so we erase it. Since there are no itemsets added to C_2, we stop the mining process.

Algorithm Association rule generation from a frequent itemset.

```
INPUT Z A frequent itemset
INPUT min_conf the minimum confidence threshold
for all item i in Z do
    Construct a rule Z − {i} ⇒ {i}
    if confidence(Z − {i} ⇒ {i}) ≥ min_conf then
        output Z − {i} ⇒ {i}
        add {i} to the set C_1
Set k = 2
repeat
    for all itemset V of size k generated by joining two itemsets from C_{k−1} do
        Construct a rule Z − V ⇒ V
        if confidence(Z − V ⇒ V) ≥ min_conf then
            output Z − V ⇒ V
            add V to the set C_k
until k < |Z| − 1
```

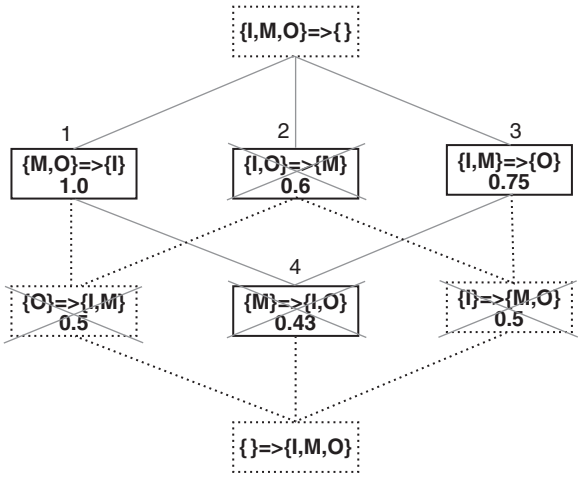


Figure 6.6 Association rules lattice corresponding to the frequent itemset {I,M,O} found in the data (Table 6.2).

6.3 Behind Support and Confidence

In some sense, each pattern reveals a kind of knowledge that might support further decisions of users of these patterns. However, only some patterns are "interesting" enough for the user, representing useful and unexpected knowledge. Evaluation of the interestingness of patterns depends on the application domain and also on the subjective opinion of the user.

Example 6.18 Imagine a data set of records of students at a university. Each item corresponds to a course and each transaction corresponds to a set of courses for which the given student signed up. For a teaching manager analyzing association rules mined from this data, a rule {probability, statistics} ⇒ {data mining} is probably uninteresting, even if it has high support and confidence, because it represents obvious knowledge: students who attended courses on probability and statistics are likely to attend a data mining course too. On the other hand an itemset {basics of biology, introduction to algorithms}, even if having lower support, might be surprising to the teaching manager, triggering a decision to open a new course or study program on computational biology.

Due to the high number of patterns in large data sets, a manual analysis is cumbersome for a human expert; incorporating human knowledge into an automatized evaluation process would also be difficult and its applicability would be domain-dependent. To support the evaluation process, several objective evaluation measures, besides support and confidence, have been developed to assess the quality of association rules, helping the user to select interesting patterns. Since a complete exposition of these measures is out of scope of this chapter, we will focus on a few, important issues about patterns.

6.3.1 Cross-support Patterns

It is not rare in real-world data that the most of the items have relatively low or modest support, while a few of the items have high support. For example, more students at a university attend a course on introductory data analytics than one on quantum computing.

If a pattern contains low-support items and high-support items, then it is called a cross-support pattern. A cross-support pattern can represent interesting relationships between items but also, and most likely, it can be spurious since the items it contains are weakly correlated in the transactions.

To measure an extent to which a pattern $\mathcal{P}$ can be called a cross-support pattern, the so-called support ratio is used. It is defined as:

$$sup_ratio(\mathcal{P}) = \frac{min\{s(i_1), s(i_2), \ldots, s(i_k)\}}{max\{s(i_1), s(i_2), \ldots, s(i_k)\}} \tag{6.3}$$

where $s(i_1), s(i_2), \ldots, s(i_k)$ are the supports of items $i_1, i_2, \ldots, i_k$ contained in $\mathcal{P}$ and *min* and *max* return the minimum and maximum value, respectively, in their arguments. In other words, *sup_ratio* computes the ratio of the minimal support of items present in the pattern to the maximal support of items present in the pattern.

This measure can be used to filter out patterns with *sup_ratio* below or above a user-specified threshold, depending on the interest of a given user.

Example 6.19 As an example, let the supports of items i_1, i_2 and i_3 be $s(i_1) = 0.9$, $s(i_2) = 0.1$ and $s(i_3) = 0.25$, respectively. Given a threshold of 0.2, the itemset $P = \{i_1, i_2\}$ with $sup_ratio(P) = \min\{s(i_1), s(i_2)\}/\max\{s(i_1), s(i_2)\} = 0.1/0.9 = 0.11$ is a cross-support pattern. The itemset $Q=\{i_1, i_3\}$ with $sup_ratio(Q) = 0.25/0.9 = 0.28$ is not a cross-support pattern.

Another way to eliminate cross-support patterns is by setting the *min_sup* threshold high, but this could also eliminate some interesting patterns. For association rules, confidence pruning by setting the *min_conf* threshold high does not help either, since a cross-support pattern can have high confidence too.

6.3.2 Lift

First, let us start with some considerations about the confidence measure and its relationship to the strength of an association rule. We will use a so-called contingency table, which contains some statistics related to an association rule, as introduced in Table 6.5. A contingency table related to two itemsets X and Y appearing in the rule $X \Rightarrow Y$ contains four frequency counts of transactions in which:

- X and Y are present
- X is present and Y is absent
- X is absent and Y is present
- neither X nor Y are present.

Example 6.20 As an example, from Table 6.5 we can see that the support of the rule $X \Rightarrow Y$ is the ratio of the number of transactions in which both X and Y are present to the total number of transactions: $support(X \Rightarrow Y) = 12/100 = 0.12$. The support of the antecedent X of the rule is $support(X) = 16/100 = 0.16$. The confidence of the rule is $confidence(X \Rightarrow Y) = 0.12/0.16 = 0.75$.

Let us now discuss the relationship $\Rightarrow$ between the antecedent X and the consequent Y of the rule $X \Rightarrow Y$, which can sometimes be misleading. One way of thinking about the confidence is as a conditional probability that a randomly selected transaction including all the items in the antecedent of the rule will include all the items in its consequent. One could, however, ask about the statistical relationship between the antecedent and the consequent of the rule: to what extent does the occurrence of the body of the rule in the data influence the probability of the occurrence of its consequent.

Example 6.21 Based on the support (0.12) and the confidence (0.75) of the rule $X \Rightarrow Y$ from the Table 6.5, we might consider this rule as a strong and

Table 6.5 An example of a two-way contingency table for itemsets X and Y.

Frequency counts		Y		
		Present	**Absent**	**Total**
X	**Present**	12	4	16
	Absent	68	16	84
	Total	80	20	100

interesting one. However, the probability of Y occurring in the data is $(12 + 68)/100 = 0.8$, regardless of whether X is present in the data. In other words, the support of Y in the data is 0.8. Thus, the occurrence of X negatively influences the occurrence of Y in the data.

We can see from this example that high confidence and good support of a rule does not necessarily imply a cause and effect between its antecedent and consequent. To measure the effect of the antecedent on the consequent of the rule, a so-called "lift" measure is used. This is defined as:

$$lift(X \Rightarrow Y) = \frac{confidence(X \Rightarrow Y)}{support(Y)} \tag{6.4}$$

The lift values are greater than or equal to zero, and:

- A lift value greater than 1 indicates a positive correlation between the rule's antecedent and consequent; that is, the occurrence of the antecedent has a positive effect on the occurrence of the consequent.
- A lift value smaller than 1 indicates a negative correlation between the rule's antecedent and consequent; that is, the occurrence of the antecedent has a negative effect on the occurrence of the consequent.
- A lift value near 1 indicates that there is no correlation between the rule's antecedent and consequent; that is, the occurrence of the antecedent has almost no effect on the occurrence of the consequent.

Example 6.22 The lift of the rule $X \Rightarrow Y$ from the Table 6.5 is equal to $0.75/0.8 = 0.9375$, indicating that the rule's antecedent and consequent appear less often together than expected. In other words, the occurrence of the antecedent has a negative effect on the occurrence of the consequent.

6.3.3 Simpson's Paradox

As was shown in the previous subsection, it is important to be cautious when interpreting association rules. The relationship observed between the

antecedent and the subsequent of the rule can also be influenced by hidden factors that are not captured in the data or not considered in the analysis.

A related phenomenon, called Simpson's paradox, says that certain correlations between pairs of itemsets (antecedents and consequents of rules) appearing in different groups of data may disappear or be reversed when these groups are combined.

Example 6.23 Consider 800 transcripts of records (transactions) formed by two groups of students, A and B, as illustrated in the Table 6.6. The groups A and B may refer, for example, to students on the physics and biology study programs, respectively, while $X = \{\text{basics of genetics}\}$ and $Y = \{\text{introduction to data analytics}\}$ might be two itemsets, each consisting of a single course.

- In group A, the rule $X \Rightarrow Y$ has a high confidence (0.8) and good lift (1.79) values.
- In group B, the rule $Y \Rightarrow X$ has a high confidence (0.8) and good lift (1.66) values.

It is clear that if we analyze the two groups of students separately, we can discover interesting and strong relationships from them. However, if we combine these two datasets such that the information about the groups (the study programs of the students) becomes a hidden factor, the analysis will give $confidence(X \Rightarrow Y) = 0.44$ and $confidence(Y \Rightarrow X) = 0.48$, with both rules having lift values equal to 1.4. Thus the same rules that were strong when the

Table 6.6 Two-way contingency tables for itemsets *X* and *Y* for two groups A and B of data and the whole data set (combined groups A and B). The presence or absence of itemsets in transactions are marked by Yes and No, respectively.

			Y					*Y*	
Group A		**Yes**	**No**	**Total**	**Group B**		**Yes**	**No**	**Total**
X	**Yes**	20	5	25	***X***	**Yes**	100	150	250
	No	105	150	255		**No**	25	245	270
	Total	125	155	280		**Total**	125	395	520

Combined groups A and B		*Y* **Yes**	**No**	**Total**
X	**Yes**	120	155	275
	No	130	395	525
	Total	250	550	800

groups were analyzed separately will become much weaker. Moreover, given that min_*conf* is set to 0.5 (which is not a very strict constraint), association rule mining would not even find these two rules.

6.4 Other Types of Pattern

There are other types of pattern, such as sequences or graphs. Their mining is based on similar principles as were presented for frequent itemset mining in Section 6.1. In general, approaches to mining frequent sequences and graphs are mostly extensions and modifications of the Apriori, Eclat and FP-Growth methods.

Since a more detailed description of mining these types of patterns is out of the scope of this chapter, we will only present the basic definitions here, focusing on sequential patterns.

6.4.1 Sequential Patterns

The input to sequential pattern mining is a sequence database, denoted by $\mathcal{S}$. Each row consists of a sequence of events consecutively recorded in time. Each event is an itemset of arbitrary length assembled from items available in the data.

Example 6.24 As an example, let the sequence database in Table 6.7 represent shopping records of customers over some period of time. For example, the first row can be interpreted as follows: the customer with ID=1 bought items as follows:

- first visit: items a and b
- second visit: items a, b and c
- third visit: items a, c, d, e
- fourth visit: items b and f.

Let us have two sequences $s_1 = \langle X_1, X_2, \ldots, X_n \rangle$ and $s_2 = \langle Y_1, Y_2, \ldots, Y_m \rangle$, where $n \leq m$. s_1 is called a "subsequence" of s_2 if there exists $1 \leq i_1 < i_2 < \cdots < i_n \leq m$ such that X_1 is a subset of Y_{i_1}, X_2 is a subset of Y_{i_2}, …, and X_n is a subset of Y_{i_n}.

Example 6.25 Let the first sequence from Table 6.7 be $s_2 = \langle Y_1, Y_2, Y_3, Y_4 \rangle$ with $Y_1 = \{a, b\}$, $Y_2 = \{a, b, c\}$, $Y_3 = \{a, c, d, e\}$ and $Y_4 = \{b, f\}$ having $m = 4$. Let $s_1 = \langle X_1, X_2 \rangle$ with $X_1 = \{b\}$, $X_2 = \{a, d, e\}$, thus, $n = 2$. There exists $i_1 = 1$ and $i_2 = 3$ such that X_1 is a subset of $Y_{i_1} = Y_1$ and X_2 is a subset of $Y_{i_2} = Y_3$, so s_1 is a subsequence of s_2. Also, there is another mapping $i_1 = 2$ and $i_2 = 3$ according to which s_1 is a subsequence of s_2.

Table 6.7 Example of a sequence database $\mathcal{S}$ with items a, b, c, d, e, f.

ID	Consecutive events (itemsets) recorded in time
1	$\langle\{a,b\},\{a,b,c\},\{a,c,d,e\},\{b,f\}\rangle$
2	$\langle\{a\},\{a,b,f\},\{a,c,e\}\rangle$
3	$\langle\{a\},\{c\},\{b,e,f\},\{a,d,e\},\{e,f\}\rangle$
4	$\langle\{e,d\},\{c,f\},\{a,c,f\},\{a,b,d,e,f\}\rangle$
5	$\langle\{b,c\},\{a,e,f\}\rangle$

The support of a given sequence s in the sequence database $\mathcal{S}$ is the number of rows in $\mathcal{S}$ of which s is a subsequence. A ratio of this number to the number of all rows in $\mathcal{S}$ can also be used.

Example 6.26 The support of the sequence $\langle\{a\},\{f\}\rangle$ in the sequence database $\mathcal{S}$ in Table 6.7 is $4/5 = 0.8$, since it is a subsequence of 4 rows (with IDs 1, 2, 3 and 4) from all the five rows in $\mathcal{S}$.

6.4.2 Frequent Sequence Mining

Given a set of all available items $\mathcal{I}$, sequential database $\mathcal{S}$ and a threshold value *min_sup*, frequent sequence mining aims at finding those sequences, called frequent sequences, generated from $\mathcal{I}$ for which support in $\mathcal{S}$ is at least *min_sup*. It is important to mention that the number of frequent sequences that can be generated from $\mathcal{S}$ with available items $\mathcal{I}$ is usually much larger than the number of frequent itemsets generated from $\mathcal{I}$.

Example 6.27 As an example, the number of all possible itemsets which can be generated from 6 items a, b, c, d, e and f, regardless of the value of the *min_sup* threshold, is $2^6 - 1 = 64 - 1 = 63$. The numbers of frequent sequences with respect to the sequence database in Table 6.7 are: 6 for $min_sup = 1.0$, 20 for $min_sup = 0.8$, 53 for $min_sup = 0.6$ and 237 for $min_sup = 0.4$.

6.4.3 Closed and Maximal Sequences

Similar to closed and maximal frequent itemsets, closed and maximal sequential patterns can be defined. A frequent sequential pattern s is closed if it is not a subsequence of any other frequent sequential pattern with the same support. A frequent sequential pattern s is maximal if it is not a subsequence of any other frequent sequential pattern.

Example 6.28 Given the sequential database in Table 6.7 and $min_sup = 0.8$, the frequent sequences $\langle\{b,f\}\rangle$ and $\langle\{a\},\{f\}\rangle$, both with support 0.8, are not closed since they are subsequences of $\langle\{a\},\{b,f\}\rangle$ with the same support

0.8, which is a maximal frequent sequence. On the other hand, the frequent sequence $\langle\{a, e\}\rangle$ with support 1.0 is closed since all of its "supersequences" $\langle\{a\}, \{a, e\}\rangle$, $\langle\{b\}, \{a, e\}\rangle$ and $\langle\{c\}, \{a, e\}\rangle$ have less support, at 0.8.

6.5 Final Remarks

The main discussion in this chapter was devoted to frequent itemset mining, describing the three main approaches: Apriori, Eclat and FP-Growth. From the user's point of view, implementations based on these approaches usually differ only in their time and memory requirements, but all of them should return the same result for the same data and the same *min_sup* threshold.

The underlying principles behind these approaches are utilized in association rule or frequent sequence mining. We refer those interested in further details of pattern mining algorithms and a comprehensive overview of state-of-the-art algorithms to the literature [21].

The main obstacle of pattern mining approaches is the large number of resulting patterns, which is mostly too large to allow further analysis by an individual expert. We have to be aware of the careful setting of the *min_sup* and *min_conf* thresholds in order to control not only the amount but also the quality of the resulting patterns. Besides the support and confidence measures for frequent patterns, which also drive the pattern mining methods presented here, there are plenty of other evaluation measures. We described in more detail the lift measure and also discussed the issue of cross-support patterns and Simpson's paradox. A quite detailed discussion of interestingness measures for frequent patterns can be found in the corresponding chapter of the book by Tan *et al.* [20].

Pattern mining is a very important data mining area, with many applications. It is used to analyze businesses, financial and medical transactions (itemsets, association rules, sequential patterns) and web log data (sequences), just to name a few.

6.6 Exercises

1 Which patterns are the following sentences equivalent to? Calculate the support or/and the confidence values of these patterns:
- 25 customers from 100 have bought bread and butter during their visits in the market.
- 5 of these 25 customers also bought honey, while the others also bought milk.
- Every fifth student used to visit the library on Monday, go to the gym and the cafeteria on Tuesday, visit the library and attend a faculty seminar on Wednesday, go to the gym and to the book store on Thursday while going to the theater and to a restaurant on Friday.

2 Find all the frequent sequences in the sequence database introduced in the Table 6.7 given $min_sup = 0.8$.

3 Create a list of ten popular lecture courses at your school. Ask at least ten of your schoolmates to name their five favorite courses from that list and create a transactional database (further referred to as "LectureData") from their answers.

4 Find the frequent itemsets for the threshold $min_sup = 0.2$ according to the Apriori approach in LectureData.

5 Filter the cross-support patterns in the itemsets found in LectureData, for those having cross-support ratios below the threshold of 0.25,

6 Find the association rules for the thresholds $min_conf = 0.6$ and $min_sup = 0.3$ in LectureData.

7 Compute the lift values of the association rules found in the example above.

8 Investigate if Simpson's paradox is present in some of the patterns in LectureData, considering the hidden factor to be the gender of the schoolmates.

9 Download some larger benchmark datasets for frequent itemset mining and compare the runtimes of Apriori, Eclat and FP-Growth implementations on these, considering at least ten different min_sup thresholds.

10 Prepare a short overview of frequent graph mining approaches, using the literature and on-line resources.

7

Cheat Sheet and Project on Descriptive Analytics

This chapter, as with the other cheat sheet and project chapter (Chapter 12), has two different sections: a cheat sheet of the contents in Part II, and the resolution of the project outlined in Section 1.6.1. The main motivation for this chapter is first to remind the reader about what has been learned so far, and second to show where the concepts described in the previous chapters can fit together in the development of a data analytics project, in this case, in descriptive analytics.

7.1 Cheat Sheet of Descriptive Analytics

The main purpose of descriptive analytics is to understand our data, providing relevant knowledge for future decisions in the project development. As the title of Part II of this books suggests, it is about getting insights from data.

The rest of Section 7.1 summarizes the main concepts covered in the previous chapters. For more detail regarding a specific concept, the reader is strongly advised to go back to the corresponding section in the previous chapters.

7.1.1 On Data Summarization

Table 7.1 presents the main aspects of the univariate methods described in Section 2.2; that is, methods used to summarize a single attribute. Table 7.2 presents a summary of bivariate methods, as presented in Section 2.3; that is, methods used to summarize pairs of attributes. Here and in the other tables, + means positive, − negative and −+ in between.

7.1.2 On Clustering

Table 7.3 presents a summary of the distance measures described in Section 5.1, and Table 7.4 presents the advantages and disadvantages of each of the three clustering techniques discussed in Section 5.3.

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

Table 7.1 Summarizing methods for a single attribute.

		Scale type		
		Nominal	**Ordinal**	**Quantitative**
Frequency	Absolute	+	+	+
	Relative	+	+	+
	Cumulative absolute	−	+	+
	Cumulative relative	−	+	+
Plot	Pie	+	−+	−+
	Bar	+	+	−+
	Line	−	−	−+
	Area	−	−	−+
	Histogram	−	−	+
Location statistics	Minimum	−	−+	+
	Maximum	−	−+	+
	Mean	−	−+	+
	Mode	+	+	−+
	Median	−	−+	+
	1st quartile	−	−+	+
	3rd quartile	−	−+	+
Dispersion statistics	Amplitude	−	−	+
	Interquartile range	−	−	+
	Mean absolute deviation	−	−	+
	Standard deviation	−	−	+
PDF	Uniform dstribution	−	−	+
	Normal distribution	−	−	+

Table 7.2 Summarizing methods for two attributes.

Method type	**Method**	**Scale types**
Correlation coefficient	Pearson	Quantitative
	Spearman	Ordinal and/or quantitative
Frequency	Contingency table	Qualitative
Plot	Scatter	Ordinal and/or quantitative
	Mosaic	Qualitative

Table 7.3 Distance measures.

	Eucl	Manh	Hamm	Edit	DTW
Objects with qualitative attributes	−	−	+	+	−
Objects with quantitative attributes	+	+	−	−	−
Sequences with quantitative attributes	−	−	−	−	+
Sequences with qualitative attributes	−	−	−	−	−
Time series of quantitative values	+	+	−	−	+
Time series of qualitative values	−	−	+	+	−
Image	+	+	−	−	−
Sound	+	+	−	−	−
Video	+	+	−	−	−

Eucl, Manh, Hamm, edit and DTW stand for Euclidean, Manhattan, Hamming, Levenshtein/edit and dynamic time warping respectively.

Table 7.4 Clustering methods.

	k-means	DBSCAN	AHC
Computational efficiency	+	−+	−+
Quality of the results	+	+	−+
Randomization	Yes	Almost no	No
Number of hyper-parameters	2	3	2
Robustness to noise and outliers	−	+	+
Shape	Convex	Arbitrary	Arbitrary
Interpretability	−	−	+

AHC, agglomerative hierarchical clustering.

7.1.3 On Frequent Pattern Mining

Table 7.5 shows the time and memory complexity of the three approaches to frequent itemset mining on which association rule mining, sequence mining and graph mining approaches are based. Note that Table 7.5 shows a general picture; the complexities of these approaches depend on their concrete implementation in the data analytics software used.

The main measures for frequent pattern mining presented in this part are illustrated in Table 7.6. Do not forget that these are the most frequently used measures. You can easily find other measures in related literature.

The relation between frequent patterns, closed patterns and maximal patterns is illustrated in Figure 7.1.

Table 7.5 Time complexity and memory requirements of frequent itemset mining approaches.

Method	Time complexity	Memory requirements
Apriori	+	−
Eclat	−+	+
FP-Growth	−	−+

Table 7.6 Measures, generally related to frequent mining approaches.

Pattern type	Support	Confidence	Lift	cross-support ratio
itemsets	+	−	−	+
association rules	+	+	+	+
sequences	+	−	−	−
graphs	+	−	−	−

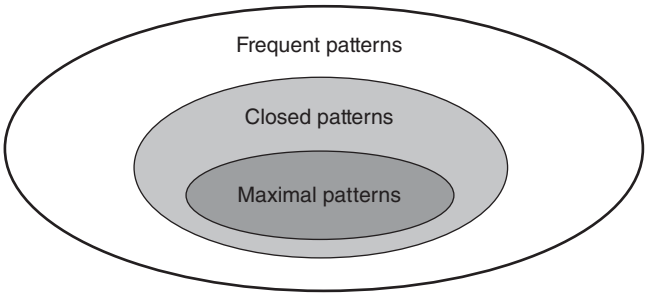


Figure 7.1 Relationship between frequent, closed and maximal patterns.

7.2 Project on Descriptive Analytics

This project will use the CRISP-DM methodology. The data used for this study can be obtained from the UCI machine learning repository [6]. It is the Wisconsin breast cancer data set [22].

7.2.1 Business Understanding

Let us assume that the Wisconsin hospital wants to develop a decision support system to help the diagnosis of breast cancer. To understand the patterns that can exist in breast mass is a primary objective.

Table 7.7 Attributes of the Breast Cancer Wisconsin data set.

	Attribute	Domain
1.	Sample code number	id number
2.	Clump thickness	1–10
3.	Uniformity of cell cize	1–10
4.	Uniformity of cell shape	1–10
5.	Marginal adhesion	1–10
6.	Single epithelial cell size	1–10
7.	Bare nuclei	1–10
8.	Bland chromatin	1–10
9.	Normal nucleoli	1–10
10.	Mitoses	1–10
11.	Class	2 for benign, 4 for malignant

7.2.2 Data Understanding

The hospital has collected digitized images of fine needle aspirates (FNAs) of breast masses. The dataset webpage has information about the data set and the data itself [22]. The attributes of the data set are shown in Table 7.7.

Once you have your data set, you need to understand it, assessing its quality and describing the data using statistics and visualization techniques. Some statistics of the attributes are presented in Table 7.8.

7.2.3 Data Preparation

Examine your data and verify, using the techniques described in Section 4.1, if it exhibits:

- missing values;
- redundant attributes;
- inconsistent data;
- noisy data.

The "bare nuclei" attribute has 16 missing values. This is the unique quality problem of the data set. Since the dataset has 699 instances, to remove 16 of them is not problematic. This is, in this case, a reasonable option.

Another question is to chose the attributes that can be useful to identify patterns in the breast mass. The sample code number is, obviously, irrelevant to identifying patterns in the data, so it should be removed. The target attribute that identifies whether the breast mass is malign or benign should also not be used in order to identify patterns in the breast mass. Indeed, although this

Table 7.8 Statistics of the attributes of the Breast Cancer Wisconsin data set.

Attr	Type	Missing	Min/least	Max/most	Values
1	Polynomial	0	95719 (1)	1182404 (6)	1182404 (6), 1276091 (5), … [643 more]
2	Integer	0	1	10	4.418
3	Integer	0	1	10	3.134
4	Integer	0	1	10	3.207
5	Integer	0	1	10	2.807
6	Integer	0	1	10	3.216
7	Integer	16	1	10	3.545
8	Integer	0	1	10	3.438
9	Integer	0	1	10	2.867
10	Integer	0	1	10	1.589
11	Binomial	0	4 (241)	2 (458)	2 (458), 4 (241)

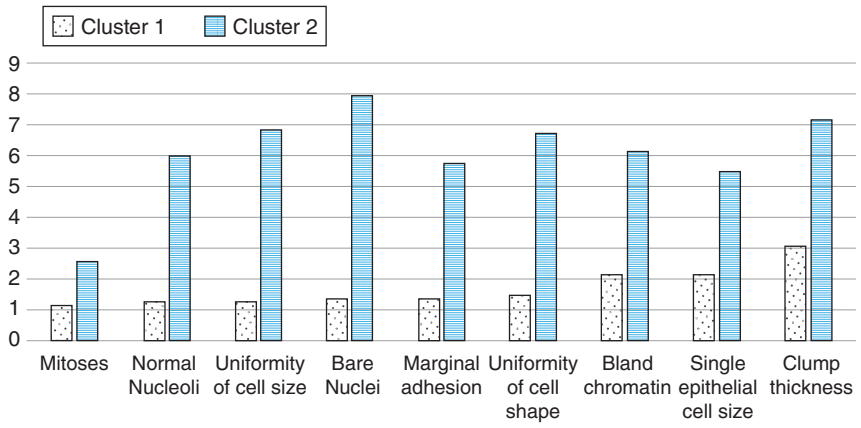


Figure 7.2 Centroids for k-means with $k = 2$, where the benign breast mass is Cluster 1.

attribute is available, we only want to identify patterns based on the images. So, it too should be removed.

Clustering algorithms such as the popular k-means algorithm use distance measures. Distance measures for quantitative attributes should have all attributes using the same scale. Otherwise the attributes with larger values will contribute more to the calculus of the distance than the attributes with lower values. In the Breast Cancer Wisconsin data set, all the attributes that will be used have the same scale, between 1 and 10, so normalization, in this case, is not necessary.

7.2.4 Modeling

Let us try the most popular clustering algorithm, the k-means algorithm. Testing different values of k, it is possible to observe that between $k = 2$ and $k = 4$, one of the clusters is more or less stable and corresponds mainly to a benign breast mass; that is, a benign breast mass has a more homogeneous pattern than a malign breast mass, as shown in Figures 7.2–7.4. Nevertheless, for $k = 4$ the two first clusters correspond roughly to the first cluster when using $k = 2$ and $k = 3$; that is, the two first clusters for $k = 4$ correspond roughly to a benign breast mass.

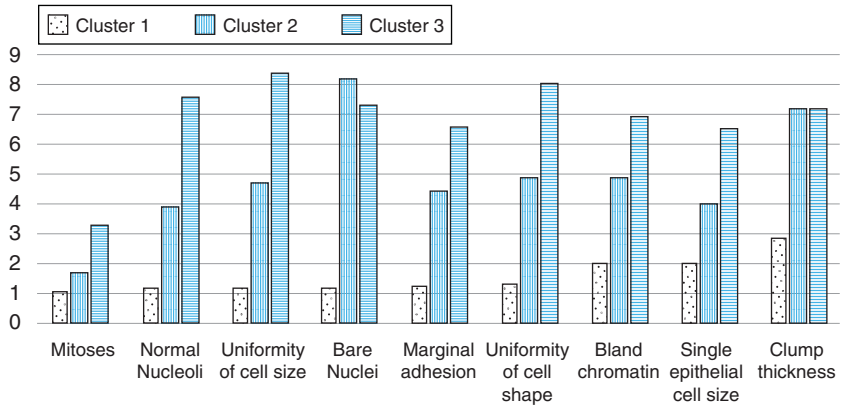


Figure 7.3 Centroids for k-means with $k = 3$, where the benign breast mass is Cluster 1.

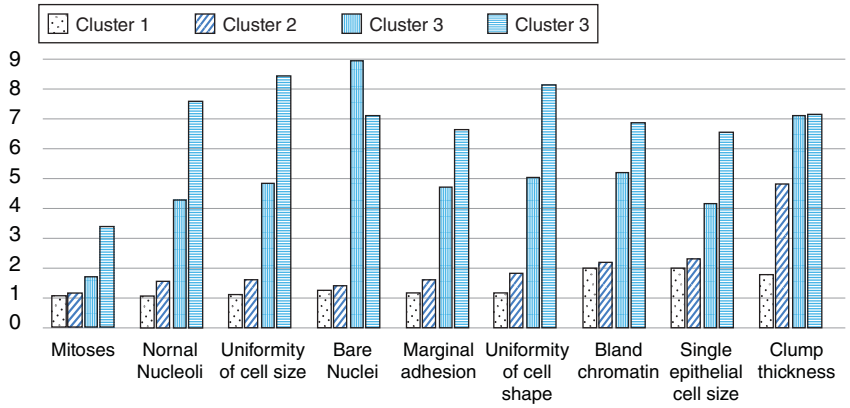


Figure 7.4 Centroids for k-means with $k = 2$, where the benign breast mass is Clusters 1 and 2.

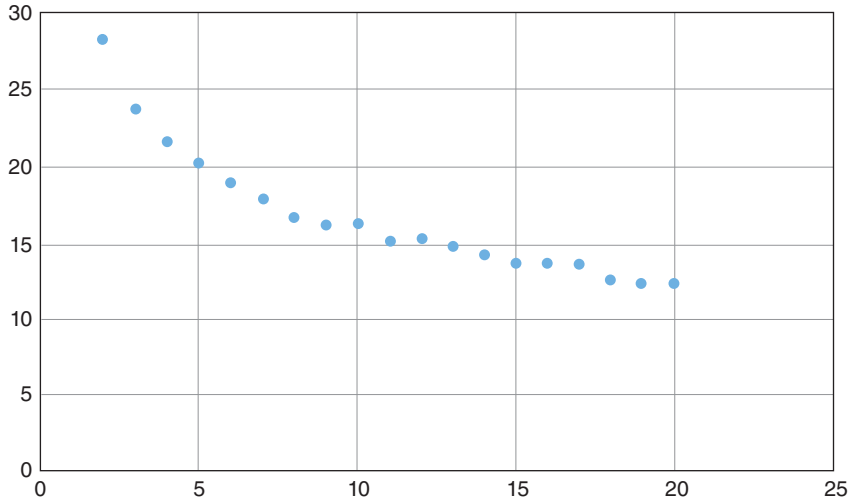


Figure 7.5 The elbow curve for the breast cancer problem using the average of the within-groups sum of squares.

The correct number of clusters can be observed using the elbow curve, although in this case the elbow curve is not very evident (Figure 7.5). Anyway, $k = 4$ was the value used.

7.2.5 Evaluation

For the hospital, the identification of the four patterns of breast mass is relevant to their study and analysis.

7.2.6 Deployment

The deployment phase is in many cases done by people other than data analysts. Making the methods previously used and evaluated accessible to the end user is the goal of this phase. Could these patterns be used in clinical practice?

Part III

Predicting the Unknown

8

Regression

Everybody (or almost everybody) would like to predict what is going to happen in the future. Which numbers will be selected for the lottery? Does the course chosen for a university degree result in professional satisfaction? Is the person you are going out with the right person? We already know that the future is hard to predict. However, there are several real problems where prediction is feasible.

One of the main tasks of analytics, named the predictive task, is the induction of predictive models.

Predictive task Induction of a model able to assign label(s), hopefully correct, to a new, unlabeled, object, given the values of its predictive attributes.

Predictive tasks do not predict what is going to happen in the future, but how likely or probable are the outcomes of a given event. An example of prediction is medical diagnosis. For example, whether a patient with a group of symptoms and results of clinical exams has a particular disease. These predictions are usually not 100% accurate, but they are helpful, especially in providing support for decision making, for example in order to:

- reduce costs
- increase profits
- improve product and service quality
- improve costumer satisfaction
- reduce environmental damage.

Predictive tasks use previously labeled data, data whose outcome is already known, to predict the outcome (or label) for new, unlabeled, data. Predictive techniques usually build or induce a so-called predictive model[1] from the labeled data, a process called inductive learning. Thus the goal of inductive learning is to find the best model – the function or the hypothesis – to map a vector of predictive attribute values of unlabeled instances in data to their correct labels.

1 From now on, the terms "predictive model" and "model" will be used interchangeably.

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

But before explaining what a model is and how it can be induced or, in other words, learned, let us discuss the data. In previous chapters, the data did not contain labels. A label represents a possible outcome of an event and can be of several types. For example, a person can be labeled "child" or "adult" (binary labeling), a car can be of types "family", "sport", "terrain" or "truck" (nominal labels), movies can be rated "worst", "bad", "neutral", "good" and "excellent" (ordinal labels), while houses have prices (quantitative labels).

Second, we have labeled and unlabeled data. In machine learning, the data used to induce a model are called the training data, since they are used by a training algorithm to specify a model that can correctly associate the attributes of each instance to its true label. This process is called training. The data used to test the performance of the induced model are called test data.

An example of an application where a model can be induced from training data is the medical diagnosis example. A hospital might have the records of several patients, and each record would be the results of a set of clinical examinations and the diagnosis for one of the patients. Each clinical examination represents a (predictive) attribute of a patient and diagnosis is the target attribute. The model induced from these training data is then used to predict the most likely diagnosis of new patients belonging to the test set, for which we know the attributes (the results of their clinical examinations), for whom we do yet know the diagnosis. Imagine how helpful such an induced model might be for the doctor who cannot decide between two equally likely diagnoses. Having another "opinion" from the prediction model can mean his decision is better supported.

Once a predictive model is induced on the training set, it can be used to predict the correct label for new data in the next step, which is called deduction. These new data are the test data whose labels are unknown.

Predictive tasks are divided between classification tasks and regression tasks. In classification tasks the main goal is to induce a model able to assign the correct qualitative label, also called the class, to new instances where the label is unknown. Classification is discussed in Chapter 9. The present chapter is about regression.

In 1877 Francis Galton observed reversion toward the mean in experiments on seed size in successive generations of sweet peas. He renamed this phenomenon "regression" in the mid-1880s [23]. Since then, regression has been the process of creating a mathematical function that explains a quantitative output attribute from a set of predictive attributes.

Regression task A predictive task whose aim is to assign a quantitative value to a new, unlabeled object, given the values of its predictive attributes.

Example 8.1 Let us use as an example our social network data set (Table 2.1) using the "weight" and the "height" as predictive and target attributes, respectively. The target attribute is another name for the label. The training

Table 8.1 Data set of our contact list, with weights and heights.

Friend	Weight (kg)	Height (cm)
Andrew	77	175
Bernhard	110	195
Carolina	70	172
Dennis	85	180
Eve	65	168
Fred	75	173
Gwyneth	75	180
Hayden	63	165
Irene	55	158
James	66	163
Kevin	95	190
Lea	72	172
Marcus	83	185
Nigel	115	192

data – friends whose height (the label) we know – are in Table 8.1. From this, a simple linear regression model $height = 128.017 + 0.611 \times weight$ is induced. Actually, this model is an equation of a line with slope 0.611 (the parameter $\hat{\beta}_1$) and intercept 128.017 (the parameter $\hat{\beta}_0$), as represented in a plot on the right side of the Figure 8.1.

Now, let us predict the height of our new friends Omar and Patricia, whose weights are 91 and 58 kg, respectively. What we have to do is to feed their attributes into the induced model. The predicted height of Omar will be $height = 128.017 + 0.611 \times 91 = 183.618$ cm while the predicted height of Patricia is $height = 128.017 + 0.611 \times 58 = 163.455$ cm. Note that Omar and Patricia are the instances belonging to the test set.

Regression methods are used in many different domains:

- *in the stock market:* to predict the value of a share in one week's time
- *transport:* travel-time prediction for a given path
- *higher education:* to predict how many students a given course will have next year
- *survival analysis:* to predict how long a person will live after a given treatment
- *macroeconomics:* to predict the expected unemployment level given a set of proposed policies.

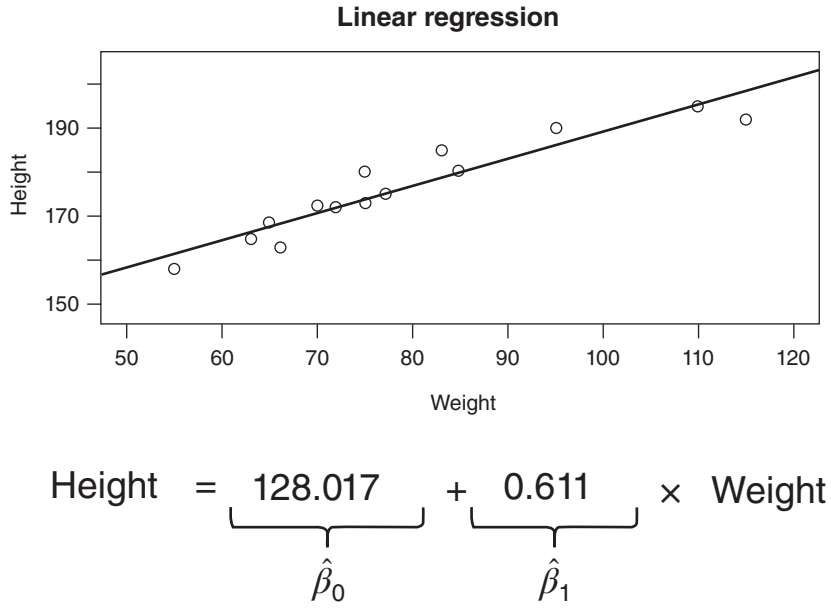


Figure 8.1 Simple linear regression for the social network data.

But before describing regression methods, we will start by describing concepts that are meaningful for both regression and classification, namely generalization and model validation. These topics are included in the subsequent section on performance estimation.

8.1 Predictive Performance Estimation

Before we talk about predictive performance evaluation, it is important to make a distinction between evaluating the predictive learning technique and evaluating the resulting model itself. However, each type of a model has its boundaries and restrictions. For example, linear models are not able to capture more complex, non-linear trends and relationships in the data, although they are easier to interpret. On the other hand, the data itself may contain noise or other types of errors (outliers or missing values) affecting the quality of the resulting model.

8.1.1 Generalization

When dealing with a predictive task represented by a data set, the main goal is to induce from this data set a model able to correctly predict new objects of the same task. We want to minimize the number or extent of future mispredictions.

Since we cannot predict the future, what we do is to estimate the predictive performance of the model for new data. We do so by separating our data set into two mutually exclusive parts, one for training – model parameter tuning – and one for testing: evaluating the induced model on new data.

We use the training data set to induce one or more predictive models for a technique. We try different configurations of the technique's hyper-parameters. For each hyper-parameter value, we induce one or more models. The hyper-parameter values that induce the models with the best predictive performance in the training set are used to represent the technique's predictive performance. If we induce more than one model for the same hyper-parameter values, we use the average predictive performance for the different models to define the predictive performance that will be associated with the technique.

If we are running experiments with more than one technique for a data set, we use the predictive performance associated with the technique to select the most suitable technique for the data set. Observe that we do not use the test data for technique selection. We assume that the test data is not seen by the technique before the selection. It is only used to confirm if it was a good selection of technique. If we also use the training set for the model induction and technique selection, we end up with an over-optimistic estimate of the technique's performance. This happens because we saw the test data before and used this information to select the technique. In this case, the model will overfit (the phenomenon described above) and will not be generic enough to be able to predict future data with a reasonable precision. In this approach we can select the best hyper-parameter values for each technique and select one or more techniques for use.

Usually, the larger the number of examples in the training set, the better the predictive performance of the technique. The two main issues with performance estimation are how to estimate the model performance for new data and what performance measure will be used in this estimation. These two aspects are covered in this section.

8.1.2 Model Validation

The main goal of a predictive model is the prediction of the correct label for new objects. As previously mentioned, what we can do is estimate the predictive performance using the model's predictive performance on a test set of data. In this process, we use predictive performance estimation methods. There are different model validation methods, which are usually based on data sampling [24].

The simplest method, holdout, divides the data set into two subsets:

- the training set is used for training
- the test set is used for testing.

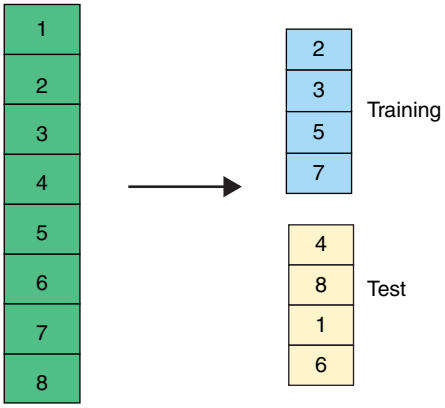


Figure 8.2 Holdout validation method.

The main deficiency of this method is that the predictive performance of the predictive models is strongly influenced by the data selected for the training and test sets. The data partition in the holdout method for a data set with eight objects is illustrated in Figure 8.2.

An alternative that reduces the problems with the holdout method is to sample the original data set several times, creating several partitions, each partition with one training set and one test set. Thus, the training predictive performance is the average of the predictive performances for multiple training sets. The same idea is used to define the test predictive performance. By using several partitions, we reduce the chance of having the predictive performance influenced by the partition used and, as a result, have a more reliable predictive performance estimate. The main methods used to create the several partitions are:

- random sub-sampling
- k-fold cross-validation
- leave-one-out
- bootstrap.

The random sub-sampling method performs several holdouts, where the partition of the data set into training and test sets is randomly defined. Figure 8.3 shows how the random sub-sampling validation method works for the same data set with eight objects that was used to illustrate holdout.

A problem with both holdout and random sub-sampling is that half of the data is not used for model induction. This can be a waste if the data set is small or medium-sized. There are variations that use two thirds of the data in the training set and one third in the test set. You reduce the waste, but it is still there.

In the k-fold cross validation method, the original data set is divided into k subsets, called "folds", ideally of equal size. This results in k partitions, where for each partition one of the folds is used as the test set and the remaining folds are used as training sets. The value used for k is usually 10. In this way, we use 90%

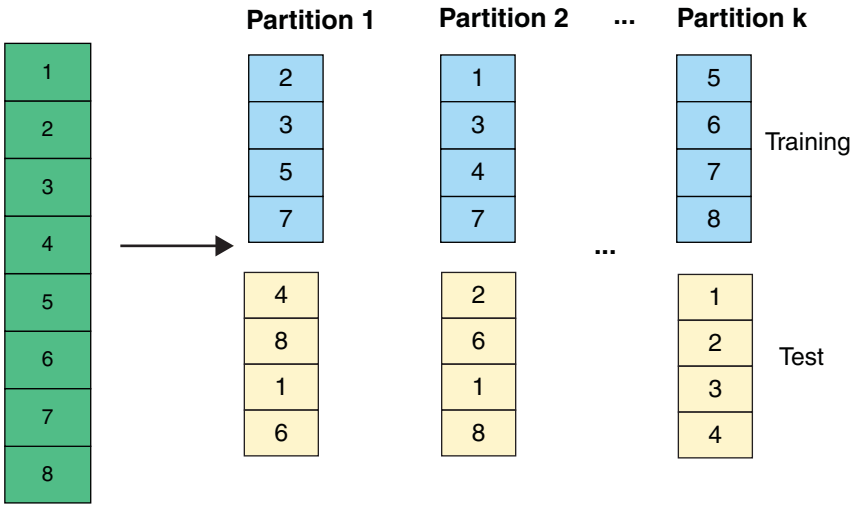


Figure 8.3 Random sub-sampling validation method.

of the data in the training set and 10% in the test set. Thus, we end up with a larger training set than with random sub-sampling. Additionally we guarantee that all objects are used for testing.

In a variation of k-fold cross-validation, called stratified k-fold cross-validation, the folds keep the same proportion of labels as found in the original data set. For regression, this is done by ensuring that the average of the target attribute is similar in all folders, while in classification it is done by ensuring that the number of objects per class are similar for the different folders. The data set partition obtained using k-fold with k equal to 4 for a data set with eight objects can be seen in Figure 8.4.

The leave-one-out method is a special case of k-fold cross validation when k is equal to the number of objects in the data set. Leave-one-out provides a better estimate of the predictive performance for new data than the 10-fold cross validation method. The average estimate provided by leave-one-out tends to the true predictive performance. However, due to its high computational cost, since a number of experiments proportional to the number of objects in the data set must be performed, its use is restricted to small data sets. Moreover, 10-fold cross-validation estimation approximates to leave-one out estimation. Figure 8.5 illustrates the data set partition produced by leave-one-out for a data set with eight objects. Because each partition has only one object in the test set, leave-one-out prediction performance for the test set has a high variance.

If we increase the number of experiments, the average of the predictive performance obtained in these experiments will be a better estimate of the predictive performance for new data. However, if we use the k-fold cross validation

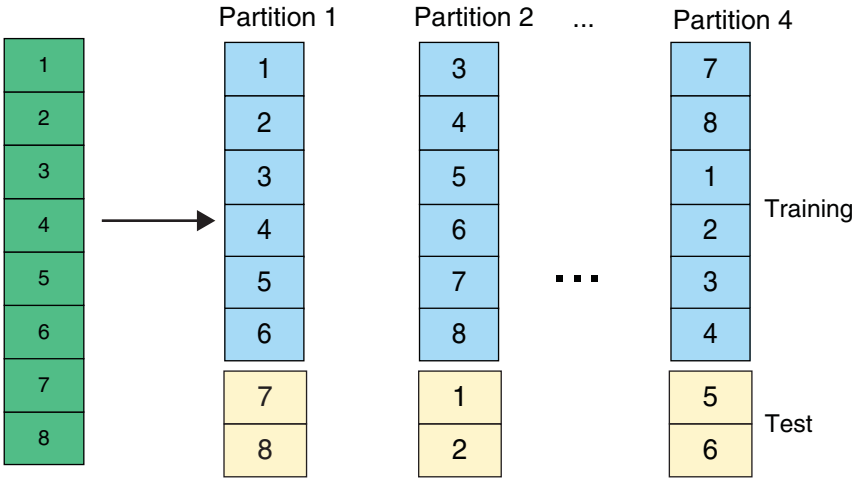


Figure 8.4 k-fold cross validation method.

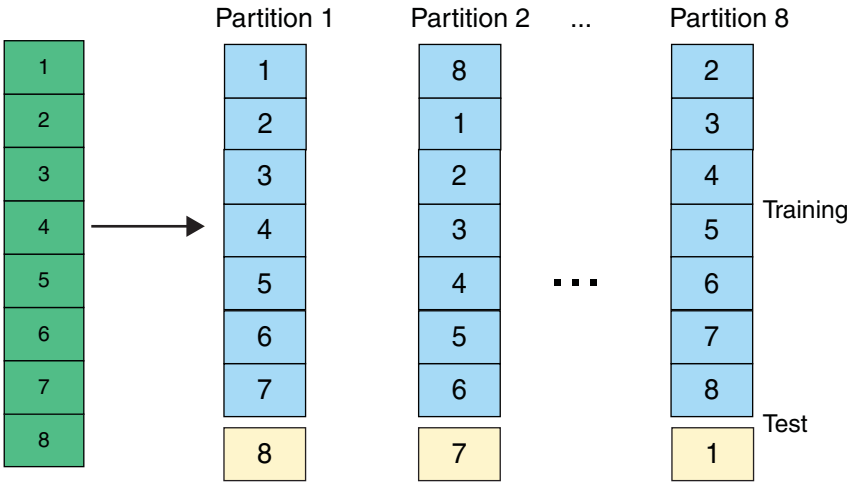


Figure 8.5 Leave one out method.

method, and its variant, leave-one-out, the maximum number of experiments is equal to the number of objects in the data set, and the larger this number, the larger the variation in the test set predictive performance. One way to overcome this problem is to use the bootstrap method.

The bootstrap validation method, like leave-one-out, also works better than 10-fold cross-validation for small data sets. There are many variations of the bootstrap method. In the simplest variation, the training set is defined by

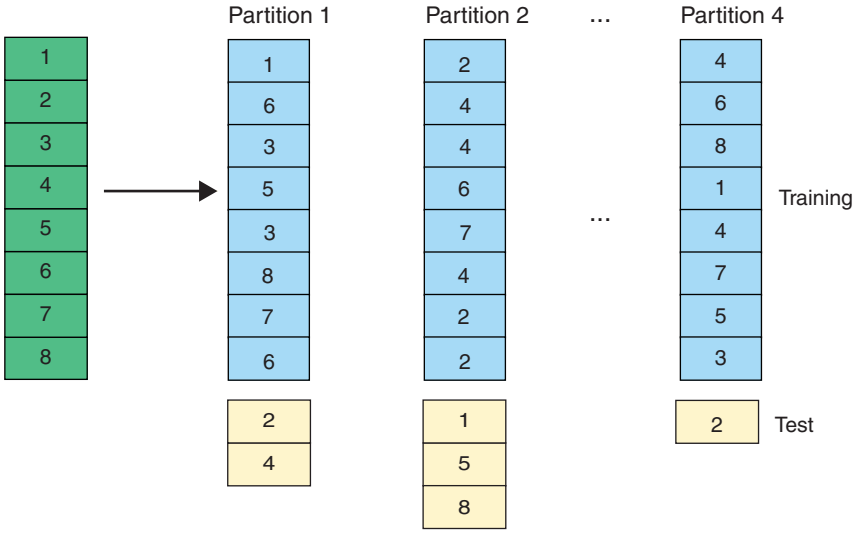


Figure 8.6 Bootstrap method.

sampling from the original data set uniformly and with replacement. Thus one object, after being selected from the original data set, is put back and can be selected again, with the same probability as other objects in the data set. As a result, the same object can be sampled more than once for inclusion in the training set. The objects that were not sampled become the test set. For large data sets, the training set tends to have 63.2% of the objects in the original data set. The other 36.7% of the objects go to the test set. The partitions obtained by applying this bootstrap method to a data set with eight objects can be seen in Figure 8.6.

8.1.3 Predictive Performance Measures for Regression

Consider the linear model discussed at the start of this chapter. This was induced from a training set of 14 instances. The test set consist of two instances, Omar and Patricia, whose predicted heights are 183.618 and 163.455 cm, respectively. The real heights of Omar and Patricia are, however, different from these predicted values, meaning that there are errors in our predictions. More concretely, let the real, measured, heights of Omar and Patricia be 176 and 168 cm, respectively.

Let $S = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, n\}$ denote the set of n instances on which the prediction performance of a model is measured. Here, $\mathbf{x}$ is in bold because it is a vector of predictive attribute values and y is in lowercase because it is a known, single value, the target value. The predicted target attribute values will be denoted as $\hat{y}_1, \ldots, \hat{y}_n$.

Example 8.2 In our case, the set S, on which the performance is measured, contains two instances, so $n = 2$. These instances are, $(\mathbf{x}_1, y_1) = ((Omar, 91), 176)$ and $(\mathbf{x}_2, y_2) = ((Patricia, 58), 168)$. The predicted values of the target attribute are $\hat{y}_1 = 183.618$ and $\hat{y}_2 = 163.455$.

The quality of the induced model is obtained by comparing the predicted values $\hat{y}_i$ with the respective true values y_i on the given set S. Depending on the way these differences are aggregated, various performance measures can be set up:

- Mean absolute error (MAE)

$$MAE = \frac{1}{n} \times \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{8.1}$$

  The value of MAE has the same unit measure as y.
- Mean square error (MSE)

$$MSE = \frac{1}{n} \times \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{8.2}$$

  Compared to MAE, MSE emphasizes bigger errors more.

The value of MSE has the square of the y unit's measure, and is therefore hard to interpret. Several performance measures that are easier to interpret can be obtained from the MSE, as follows:

- Root mean square error (RMSE)

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{8.3}$$

  This measure has the same unit measure as y, so it is easier to interpret than MSE.
- Relative mean square error (RelMSE)

$$RelMSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y})^2} \tag{8.4}$$

  This measure compares the prediction ability of all $\hat{y}_i$ tested against the average, trivial prediction, $\overline{y}$. The meaning of the values are as follows:
  - 0 if the regression model is perfect;
  - $(0, 1)$ if it is useful;
  - 1 if it is as useful as using the average as the predictor (the trivial prediction);
  - > 1 if it is worse than the trivial prediction.

- Coefficient of variation (CV)

$$CV = \frac{RMSE}{\overline{y}} \tag{8.5}$$

CV is unitless: the result is a percentage of the average. For that reason it is easy to interpret. However, CV only makes sense when the domain of y is IR^+

In our example:

Example 8.3

$MAE = \frac{1}{2} \times (|176 - 183.618| + |168 - 163.455|) = \frac{1}{2} \times (7.618 + 4.545) = 6.082$;

$MSE = \frac{1}{2} \times ((176 - 183.618)^2 + (168 - 163.455)^2) = \frac{1}{2} \times ((-7.618)^2 + 4.545^2) = 39.345$;

$RMSE = \sqrt{MSE} = \sqrt{39.345} = 6.273$;

$\overline{y}$ is the average of the target values in the training set, i.e.

$\overline{y} = \frac{175+195+172+180+168+173+180+165+158+163+190+172+185+192}{14} = 176.286$

$RelMSE = \frac{(176-183.618)^2+(168-163.455)^2}{(176-176.286)^2+(168-176.286)^2} = \frac{78.691}{68.740} = 1.145$;

$CV = \frac{6.273}{176.286} = 0.036$

8.2 Finding the Parameters of the Model

After we choose the type of the model to be used we have to find its best parameters, given the data in the training set. Recall that, during the learning, we do not know the values of the target attributes of the instances in the test set, but only the values of their predictive attributes. In our example, the real heights of Omar and Patricia were "hidden from" the learning algorithm; we used them only to measure the predictive performance of the resulting model.

Each learning technique is, in essence, an optimization algorithm that finds the optimal parameters of the corresponding model given some objective function. It means that the type of the model determines the learning algorithm or, in other words, each learning algorithm is designed to optimize a specific type of model. Several regression algorithms have been proposed, most of them in the area of statistics. Next, we briefly describe the most popular ones.

8.2.1 Linear Regression

The linear regression (LR) algorithm is one of the oldest and simplest regression algorithms. Although simple, it is able to induce good regression models, which are easily interpretable.

Let's take a closer look at our model $height = 128.017 + 0.611 \times weight$. This will allow us to understand the main idea behind LR. Given the notation above, each instance $\mathbf{x}$ is associated with only one attribute, the weight (that's why it is usually called univariate linear regression) and the target attribute y is associated with the height. As we have already shown, this model is the equation of a line in a two-dimensional space. We can see that there are two parameters, $\hat{\beta}_0$ and $\hat{\beta}_1$, such that $\hat{\beta}_1$ is associated with the importance of the attribute x_1, the weight. The other parameter, $\hat{\beta}_0$, is called the intercept and is the value of $\hat{y}$ when the linear model intercepts the y-axis, in other words when $x_1 = 0$.

The result of the optimization process is that this line goes "through the middle" of these instances, represented by points. The objective function, in this case, could be defined as follows. Find the parameters $\hat{\beta}_0, \hat{\beta}_1$ representing a line such that the mean of the squared distance of the points to this line is minimal. Or, in other words, find a model $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \times x_1$, called a univariate linear model, such that the MSE between y_i and $\hat{y}_i$ is minimal, considering all the instances $(\mathbf{x}_i, y_i)$ in the training set where $i = 1, \ldots, n$. Formally, this can be written as:

$$\underset{\hat{\beta}_0,\hat{\beta}_1}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \underset{\hat{\beta}_0,\hat{\beta}_1}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - (\hat{\beta}_0 + \hat{\beta}_1 \times x_{i_1}))^2 \tag{8.6}$$

Given the training data of our example, the optimal values for $\hat{\beta}_0$ and $\hat{\beta}_1$ are 128.017 and 0.611, respectively.

For multivariate linear regression – the linear model generalized for any number p of predictive attributes – the model is expressed as:

$$\hat{y} = \beta_0 + \sum_{j=1}^{p} \beta_j \times x_j \tag{8.7}$$

where p is the number of predictive attributes, $\hat{\beta}_0$ is the value of $\hat{y}$ when all $x_j = 0$ and $\hat{\beta}_i$ is the slope of the linear model according to the jth axis: the variation of $\hat{y}$ per unit of variation of x_j.

Take a quick look at the notation: x_j means the jth attribute of some object $\mathbf{x}$ represented as a tuple $\mathbf{x} = (x_1, \ldots, x_j, \ldots, x_p)$. Since our data consists of more instances $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, the ith instance will be denoted as $\mathbf{x}_i = (x_{i_1}, \ldots, x_{i_j}, \ldots, x_{i_p})$, x_{i_j} corresponding to its jth attribute.

The values of $\beta_0, \beta_1, \ldots, \beta_p$ are estimated using an appropriate optimization method to minimize the following objective function

$$\underset{\hat{\beta}_0,\ldots,\hat{\beta}_p}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \underset{\hat{\beta}_0,\ldots,\hat{\beta}_p}{\operatorname{argmin}} \sum_{i=1}^{n} \left[y_i - \left(\hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j \times x_{i_j} \right) \right]^2 \tag{8.8}$$

for n instances with p predictive attributes.

8.2.1.1 Empirical Error

Remember that we have only the training set available during the learning, thus, the error is also measured on this set while the model is learned. If we denote the (squared) deviation $(y_i - \hat{y}_i)^2$ as $error(y_i, \hat{y}_i)$, the objective functions, introduced in equations (8.6) and (8.8), considering the given instances $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, can be written as

$$\underset{\hat{\beta}_0, \hat{\beta}_1}{\operatorname{argmin}} \sum_{i=1}^{n} error(y_i, \hat{y}_i) \tag{8.9}$$

and

$$\underset{\hat{\beta}_0, \ldots, \hat{\beta}_p}{\operatorname{argmin}} \sum_{i=1}^{n} error(y_i, \hat{y}_i) \tag{8.10}$$

respectively for univariate and multivariate linear regression.

The error measured on the training set is called the empirical error or empirical loss, and measures the deviation between the predicted $(\hat{y}_i)$ and measured (y_i) values for training instances $(\mathbf{x}_i)$.

The assumptions made about the errors – the unexplained variation of y – by the multivariate linear model are:

- they are independent and identically distributed, an assumption that suffers when there is collinearity between predictive attributes
- homoscedasticity: there is homogeneous variance, as depicted in Figure 8.7
- normal distribution: a condition that can be verified by comparing the theoretical distribution and the data distribution.

The breach of these assumptions can result in an inaccurate definition of the coefficients $\hat{\beta}_i$.

Assessing and evaluating results The main result of MLR is the estimates of the $\hat{\beta}_i$ coefficients. The $\hat{\beta}_0$ coefficient is often named *coefficient* or $\hat{\alpha}$. Do not forget that knowing the values of all $\hat{\beta}_i$, for $i = 0, ..., p$, it is possible to estimate the target value of new unlabeled instances. Moreover, the estimates $\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_p$ express the influence of the attribute values $x_1, x_2, \ldots, x_p$ of an instance $\mathbf{x}$ on its target value y. The sign of $\hat{\beta}_i$ corresponds to the importance of the attribute x_i on y. For positive $\hat{\beta}_i$, the value x_i positively influences the value of y, while for negative $\hat{\beta}_i$ the influence of x_i on y is negative.

Advantages and disadvantages of LR The interpretability of linear regression models is one reason for their popularity. Another is that LR has no hyper-parameters. Table 8.2 summarizes the main advantages and disadvantages of LR.

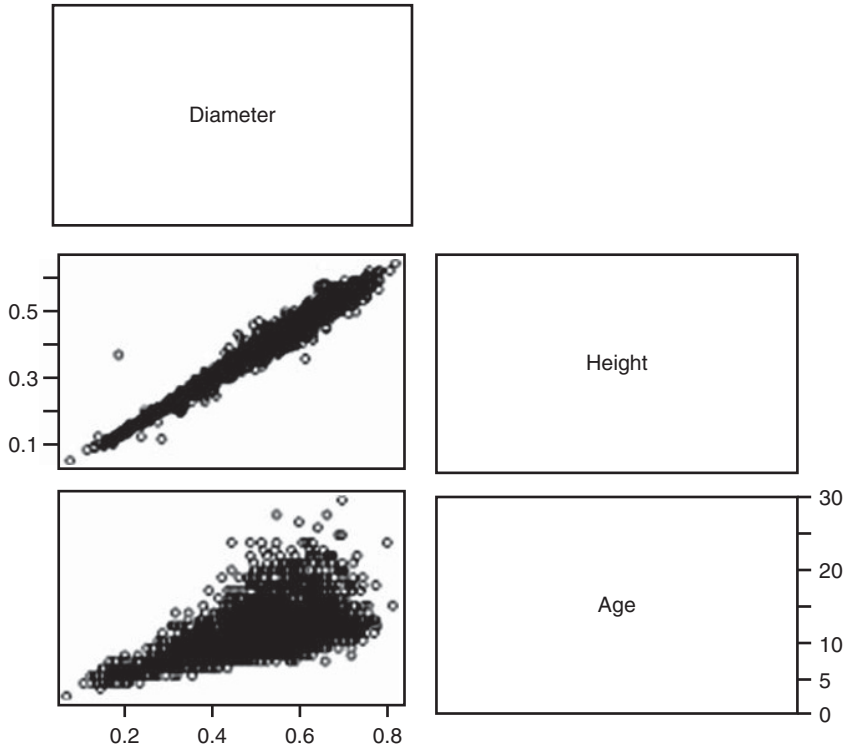


Figure 8.7 Approximately homogeneous variance between diameter and height and nonhomogeneous variance between diameter and age. This data is about the shellfish *Haliotis moluscus*, also known as abalone. It is a food source in several human cultures. The shells are used as decorative items.

Multivariate linear regression suffers when the data set has a large number of predictive attributes. This can be overcome by using one of the following approaches:

- using an attribute selection method to reduce dimensionality (review the attribute selection methods described in Section 4.5.2)
- shrinking the weights
- using linear combinations of attributes instead of the original predictive attributes.

Shrinkage methods and linear combination of attributes is discussed next. But before then, there is a discussion of the bias–variance trade-off, an important concept in understanding shrinkage methods.

Table 8.2 Advantages and disadvantages of linear regression.

Advantages	Disadvantages
• Strong mathematical foundation • Easily interpretable • Hyper-parameter free	• Poor fit if relationship between predictive attributes and target is non-linear • The number of instances must be larger than the number of attributes • Sensitive to correlated predictive attributes • Sensitive to outliers

8.2.2 The Bias-variance Trade-off

Before moving forward, let us discuss the noise in the data, which significantly influences the outcome of the learning process. Noise can have many causes such as, for example, imprecision of the measuring devices or sensors. In our example, the height of each person is an integer, and is therefore probably not precise. We usually do not measure the height on a more accurate scale.

Suppose that the relationship between the instances $\mathbf{x}_i$ and their labels y_i can be expressed, in general, by some hypothetical function f that maps each instance $\mathbf{x}_i$ to some value $f(\mathbf{x}_i)$. However, the real, measured value y_i usually differs from this hypothetical value $f(\mathbf{x}_i)$ by some, usually small, value ϵ_i corresponding to the noise. Thus we get:

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \tag{8.11}$$

where the noise ϵ_i is the component of y_i that cannot be predicted by any model. In addition, we usually assume normally distributed noise with zero mean; that is, $\epsilon \sim N(0, 1)$. In other words, there are cases when y_i is slightly below $f(\mathbf{x}_i)$ (negative noise) and cases when y_i is slightly above $f(\mathbf{x}_i)$ (positive noise), but the average noise (its expected value) should be 0.

However, since f is unknown, the goal of regression is to induce a model $\hat{f}$ that is as close to f as possible:

$$\hat{f} \approx f. \tag{8.12}$$

To be a useful model, $\hat{f}$ should be as close to f as possible, not only for the training instances but also for new unknown instances. In other words, we want to predict the future (test data) and not the past (training data). However, there are some circumstances in which inducing a good model is difficult. First, there is the presence of noise in the data, as discussed above. Another very important issue concerns the limitations of the model used; for example, linear models are not able to capture more complex, non-linear relations and trends in the data, as will be discussed in Chapter 10. Finally, the choice of the training data is,

somewhat random. What if our training data in Figure 8.1 consisted if only half of these people or if it consisted of some other contacts? Would the model be the same? probably not. The question is how would our model look if it were trained on different data, even if only slightly so, from the same population?

As an illustration of this situation, consider four instances chosen from Figure 8.1. In Figure 8.8, four scenarios are shown. In each, there are three training instances, marked by dots, and one test instance, marked by a circle. Three different models are trained:

- the average model (dotted line) is the simplest, predicting the average height of the instances in the training data for new instances;
- the linear regression model (dashed line) is more complex;
- a non-linear model fitting a polynomial of the form $\hat{y} = \beta_0 + \beta_1 \times x + \beta_2 \times x^2$ (solid line) is the most complex.

The MSE of these models on the test instances is shown by the corresponding shaded rectangles.

Let us now analyze the same three models, one at a time. The predictions in each of the scenarios is shown in Figure 8.9. First, look at the most complex: the polynomial models. The models fit the training data precisely: the empirical MSE in all four scenarios is equal to zero. In other words, the bias of the models is very low so that, in these cases, they "copy" the training data precisely. On the other hand, the four models induced on the four sets of training data vary a lot. In other words, the variance of these models is high, meaning they are not particularly stable. Large variance means that the MSE of these models measured on the four test instances (the grey shaded squares on the sides of the lines from test instances to the corresponding models in Figure 8.8) is large. In such cases, we say that polynomial models overfit the data: they have low bias but large variance. Note that this is an example designed to illustrate overfitting using small amounts of data. However, very complex models tend to overfit even with large amounts of data.

At other extreme, the least complex model – the average – shows the opposite behavior. It is quite stable, not changing much for the various scenarios (the models in Figures 8.8b and 8.8c are actually the same); in other words, its variance is low. On the other hand, its empirical MSE is the largest for all four cases, so the bias of the average model is large. In this case we could say that average models underfit the data, having high bias and low variance.

Clearly, neither of these two extremes is good and we need to find a good compromise between model types with low bias and model types with low variance, as illustrated in Figure 8.10. In our example, from Figure 8.8, the linear model seems to be a good choice, since in all the cases it has quite low bias while also keeping the variance low. As can be seen, linear models have the lowest MSE on the test instances in all the four scenarios.

The bias–variance trade-off is depicted on Figure 8.10. Bias and variance are negatively correlated, so minimizing one maximizes the other. It is possible to

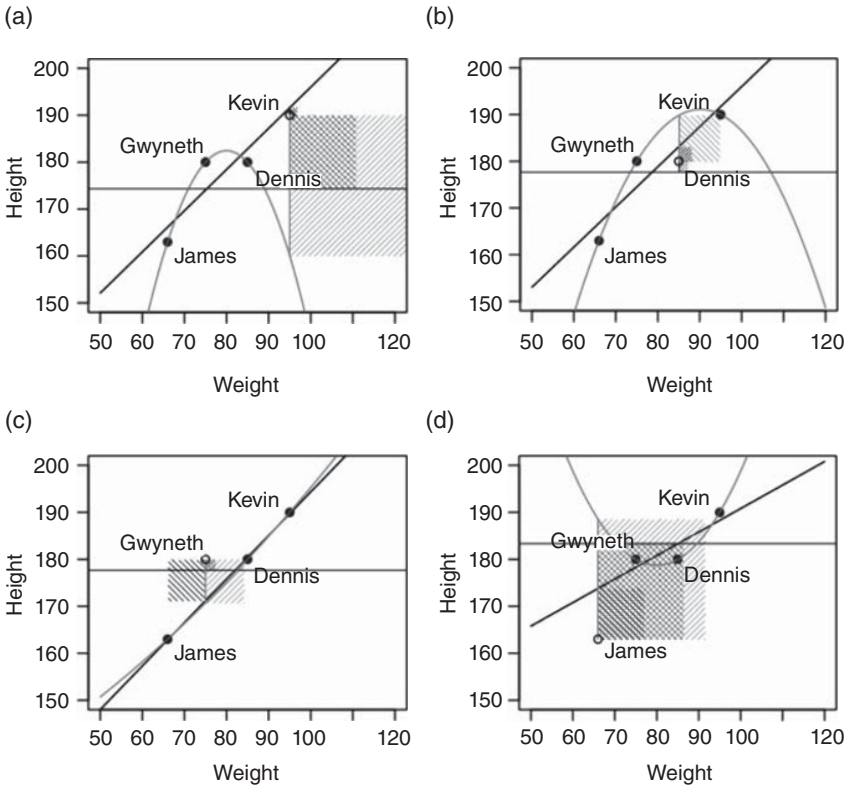


Figure 8.8 Performance of different models when trained and tested on different instances of the same data set. Refer to main text for details.

increase the bias slightly, reducing the overall error, because it is possible to increase the variance a little, reducing the overall error. In fact a deterministic model produces a unique output for the same input. However, we are not measuring the bias of a model but, instead, the bias of a method. Several methods change meaningfully with small changes in the training set.

Why is this important? As we will see, some methods are more focused in reducing the bias component of the error while others are more focused in reducing the variance component of the error. In general, the more complex a model is, the less bias and the larger the variance; the simpler a model is, the more bias and the less variance is expected. Now we will look at methods that can reduce the variance and the overall MSE error by increasing the bias.

8.2.3 Shrinkage Methods

Multivariate linear regression has low bias, but high variance. Shrinkage methods try to minimize the overall error by increasing the bias slightly,

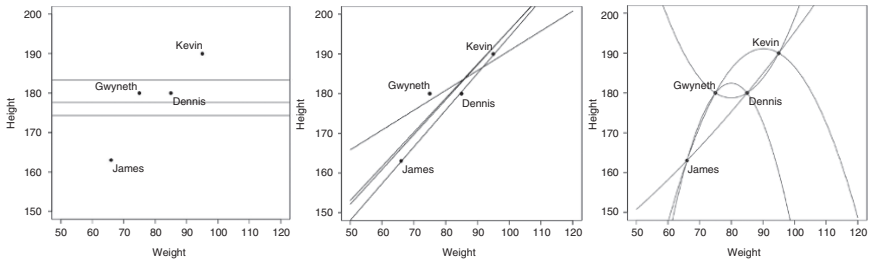


Figure 8.9 Models from the Figure **??**: left, average models; center, linear regression models; right, polynomial models.

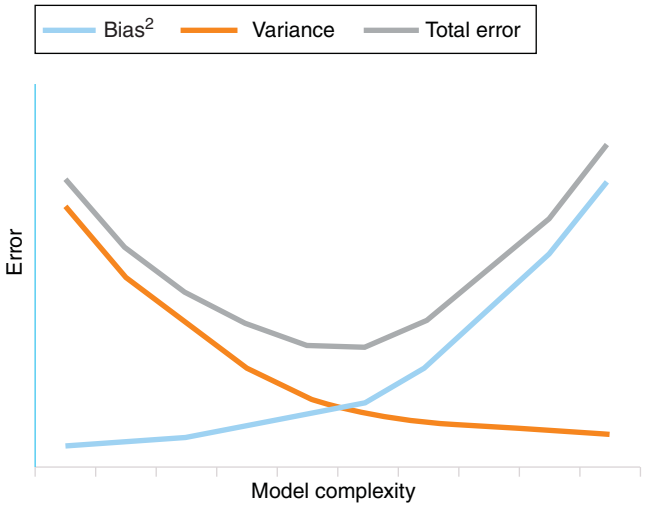


Figure 8.10 The bias–variance trade-off.

while reducing the variance component of the error. Two of the best-known shrinkage methods are ridge and lasso regression.

8.2.3.1 Ridge Regression

Ridge regression increases the bias component of the overall error by adding a penalty term for the coefficients $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ to Equation (8.10), leading to the following objective function for optimization:

$$\underset{\hat{\beta}_0,\ldots,\hat{\beta}_p}{\text{argmin}} \left\{ \sum_{i=1}^{n} error(y_i, \hat{y}_i) + \lambda \times \sum_{j=1}^{p} \hat{\beta}_j^2 \right\} \tag{8.13}$$

for n instances with p predictive attributes. This can be, according to Equation (8.8), rewritten as:

$$\underset{\hat{\beta}_0,\ldots,\hat{\beta}_p}{\text{argmin}} \left\{ \sum_{i=1}^{n} \left[y_i - \left(\hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j \times x_{ij} \right) \right]^2 + \lambda \times \sum_{j=1}^{p} \hat{\beta}_j^2 \right\} \tag{8.14}$$

Assessing and evaluating results As for MLR, the main result of ridge regression is a set of estimates for the $\hat{\beta}_j$ coefficients. Indeed, ridge regression is also a multivariate linear model, but uses a different method to learn the $\hat{\beta}_j$ coefficients.

Setting the hyper-parameters Ridge regression has one hyper-parameter, the λ, that penalizes the $\hat{\beta}_j$ coefficients, i.e., as larger λ is, more costly is to have larger $\hat{\beta}_j$ coefficients. The right value for λ is problem dependent.

Table 8.3 Advantages and disadvantages of ridge regression.

Advantages	Disadvantages
• Strong mathematical foundation • Easily interpretable • Deals better with correlated predictive attributes than ordinary least squares.	• Number of instances must be larger than number of attributes • Sensitive to outliers • Data should be normalized • When relation between predictive and the target attributes is non-linear, uses information poorly

Advantages and disadvantages of ridge regression The advantages and disadvantages of ridge regression are shown in Table 8.3.

8.2.3.2 Lasso Regression

The least absolute shrinkage and selection operator (lasso) regression algorithm is another penalized regression algorithm, that can deal efficiently with high-dimensional data sets. It performs attribute selection by taking into account not only the predictive performance of the induced model, but also the complexity of the model. The complexity is measured by the number of predictive attributes used by the model. It does this by including in the equation of the multivariate linear regression model an additional weighting term, which depends on the sum of the $\hat{\beta}_j$ weights modules. The weight values define the importance and number of predictive attributes in the induced model.

The lasso algorithm usually produces sparse solutions. Sparse means that a large number of predictive attributes have zero weight, resulting in a regression model that uses a small number of predictive attributes. As well as attribute selection, the lasso algorithm also performs shrinkage. Mathematically, the lasso algorithm is very well founded.

The lasso formulation is quite similar to the ridge formulation, as presented in Equations (8.14) and (8.13):

$$\underset{\hat{\beta}_0,\ldots,\hat{\beta}_p}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} error(y_i, \hat{y}_i) + \lambda \times \sum_{j=1}^{p} |\hat{\beta}_j| \right\} \tag{8.15}$$

However, they result in substantially different models. This is because the lasso formulation favors the existence of many zero $\hat{\beta}_j$ coefficients. To illustrate why this happens, let us assume that $\hat{\beta}_1 = 0.2$ and $\hat{\beta}_2 = 0.3$. The ridge approach will have $\sum_{j=1}^{p} \hat{\beta}_j^2 = 0.2^2 + 0.3^2 = 0.04 + 0.09 = 0.13$ while the lasso approach will have $\sum_{j=1}^{p} |\hat{\beta}_j| = |0.2| + |0.3| = 0.5$. But if, instead, $\hat{\beta}_1 = 0.5$ and $\hat{\beta}_2 = 0$, ridge regression will have $\sum_{j=1}^{p} \hat{\beta}_j^2 = 0.5^2 + 0^2 = 0.25 + 0 = 0.25$, a value larger than

Table 8.4 Advantages and disadvantages of the lasso.

Advantages	Disadvantages
• Strong mathematical foundation • Easier interpretation than ordinary least squares or ridge regression because it produces simpler models (with fewer predictive attributes) • Deals better with correlated predictive attributes than ridge regression or ordinary least squares; • Automatically discounts irrelevant attributes	• Number of instances must be larger than number of attributes • Sensitive to outliers • Data should be normalized • When the relationship between predictive and target attributes is non-linear, uses information poorly

before, but the lasso will have $|0.5| + |0| = 0.5$, the same value as before. This example shows that ridge regression promotes shrinkage while lasso promotes attribute selection by setting some of the $\hat{\beta}_j$ weights to 0 but also shrinking some other coefficients.

Assessing and evaluating results As with MLR and ridge regression, the main result of lasso regression are the estimates of the $\hat{\beta}_j$ coefficients. Lasso regression is also a multivariate linear model, but uses a different method to learn the $\hat{\beta}_j$ coefficients.

Setting the hyper-parameters Like ridge regression, lasso regression has one hyper-parameter, λ, which penalizes the $\hat{\beta}_j$ coefficients; that is, the larger λ is, more costly is to have larger $\hat{\beta}_j$ coefficients. The correct value for λ is problem dependent.

Advantages and disadvantages of the lasso The advantages and disadvantages of the lasso are set out in Table 8.4.

8.2.4 Methods that use Linear Combinations of Attributes

A third approach to deal with multiple attributes, some of them correlated, is to create new linear combinations of predictive attributes, as explained in Chapter 3 for principal components analysis. These can be used for the linear regression instead of the original attributes.

8.2.4.1 Principal Components Regression

Principal components regression (PCR) creates linear combinations of predictive attributes. The first principal component – the first linear combination – is

the one that captures the most variance of all the possible linear combinations. The following principal components are those ones that capture the most remaining variability while being uncorrelated with all previous principal components. PCR defines the principal components without evaluating how correlated the principal components generated are with the target attribute. The principal components are used as predictive attributes in the formulation of the multivariate linear regression problem.

8.2.4.2 Partial Least Squares Regression

Partial east squares (PLS) regression starts by evaluating the correlation of each predictive attribute with the target attribute. The first principal component is a linear combination of the predictive attributes, with the weights for each defined according to the strength of their univariate effect on the target attribute. The process follows the one described for PCR. PLS gives similar results to PCR but using fewer principal components. This is due to the process of measuring the correlation of the principal components with the target attribute.

8.3 Technique and Model Selection

In this chapter, we saw some predictive techniques that are widely used in regression tasks. Several techniques have been proposed and new ones are continuously being created for both regression and classification. Ferdandez-Delgado *et al.* have compared 179 classification techniques in 121 data sets [25].

So whenever we have a new predictive task, we face the problem of which predictive technique to choose. We can reduce the alternatives by selecting among those techniques known for their good performance in predictive tasks. But even so, we will still have a large number of options to choose from. The choice of the technique is influenced by our requirements, which can depend on:

- **Memory**
  - *Memory needed for the technique:* For data stream applications, where new data continuously arrive and classification models need to be automatically updated, if the technique is implemented in a portable device, it must fit the available device memory.
  - *Memory needed for the induced model:* The memory necessary to store the model is particularly important when the model is implemented on a small portable device and needs to be stored in a small memory space.

- **Processing cost**
  - *Technique processing cost:* This measure is related with the computational cost of applying the technique to a data set in order to induce a classification model. This is particularly relevant in data stream applications, where the model induction or update must be fast.
  - *Model processing cost:*This is the time the model takes, given the values of the predictive attributes of an object, to predict a class label of an object, which is relevant for applications that need a fast output, such as autonomous driving.
- **Predictive performance**
  - *Technique predictive performance:* This measure estimates the predictive performance of the models induced by the technique. This is the main performance measure and often the main aspect taken into account for technique selection. It is usually the average predictive performance of several models induced by the technique for different samples of a data set. It depends on the predictive performance of the models induced by the technique.
  - *Model predictive performance:* This estimates the predictive performance of a classification model induced by a technique for the classification of new data.
- **Interpretability**
  - *Technique interpretability:* How easy it is to understand the knowledge represented by the models induced by the technique?
  - *Model interpretability:* How easy it is for a human to understand the knowledge represented by a particular model?

8.4 Final Remarks

In this chapter, we have seen some important concepts on prediction: the experimental setup and the need to have training and test data sets to adequately evaluate methods and models. Although these concepts have been described in the regression chapter, they are relevant for both regression and classification.

Another part of the chapter described some of the most popular regression methods. The methods described are based on statistics. Some of them are based on the bias–variance trade-off, others are based on principal components.

Finally, we saw some of the criteria that can be used to select a technique. These requirements can be used for any predictive task, not only regression. Chapter 12 summarizes the different methods presented in Part III using these and other requirements.

The next chapter introduces classification and, in particular, binary classification.

8.5 Exercises

1 Explain in your own words the difference between choosing a model and choosing an algorithm with their respective hyper-parameters.

2 Why should we use different data to train and to test a model?

3 What are the advantages of both coefficient of variation and the relative mean square error when compared to the mean square error?

4 Consider the formula $\hat{y} = 5 + 0.5 \times x$ and explain, using a graphic, the meaning of the coefficients 5 and 0.5.

5 How would you classify a linear model with respect to the bias–variance trade-off?

6 Use forward selection for subset attribute selection with multivariate linear regression (MLR) on the data set in Table 8.5.

Table 8.5 My social network data using the height as target.

Age	Educ. level	Max. temp	Weight	Arabic	Indian	Mediterr.	Oriental	Fast food	Height
55	1.0	25	77	0	1	1	0	0	175
43	2.0	31	110	0	1	0	1	1	195
37	5.0	15	70	0	1	1	1	0	172
82	3.0	20	85	1	0	1	0	0	180
23	3.2	10	65	0	0	0	1	0	168
46	5.0	12	75	0	1	1	1	0	173
38	4.2	16	75	1	0	1	0	0	180
50	4.0	26	63	0	1	0	1	1	165
29	4.5	15	55	0	1	1	1	0	158
42	4.1	21	66	1	0	1	0	0	163
35	4.5	30	95	0	1	0	1	0	190
38	2.5	13	72	0	0	1	0	0	172
31	4.8	8	83	0	0	1	1	0	185
71	2.3	12	115	0	1	0	0	1	192

7 Using the data set in Table 8.5, test the predictive performance of MLR using 10-fold cross validation as the re-sampling technique and the mean square error as the performance measure.

8 Using the same data and the same experimental setup as in the previous question, use ridge regression, with λ values of 0.1, 0.5 and 1.

9 Using the same data and the same experimental setup as in the two previous questions, use the lasso, with λ values of 0.1, 0.5 and 1.

10 Compare the results obtained in the three previous questions in terms of predictive performance, interpretability and model processing cost.

9

Classification

Classification is one of the most common tasks in analytics, and the most common in predictive analytics. Without noticing, we are classifying things all the time. We perform a classification task when:

- we decide if we are going to stay at home, go out for dinner or visit a friend;
- we choose a meal from the menu of a restaurant;
- we decide if a particular person should be added to our social network;
- we decide if someone is a friend.

But classification is not only used to make decisions in our social lives; it is crucial for several activities in the private and public sectors. Examples of classification tasks where ML algorithms can be used include:

- classifying an email as spam or useful
- diagnosing a patient as sick or healthy
- classifying a financial transaction as fraudulent or normal
- identifying a face as belonging to a criminal or to an innocent person
- predicting if someone in our social network would be good dinner company.

Classification task A predictive task where the label to be assigned to a new, unlabeled, object, given the value of its predictive attributes, is a qualitative value representing a class or category.

The difficulty (complexity) of a classification task depends on the data distribution in the training data set. The most common classification task is binary classification, where the target attribute can have one of only two possible values, for example "yes" or "no". Usually, one of these classes is referred to as the positive class and the other as the negative class. The positive class is usually the class of particular interest. As an example, a medical diagnosis classification task can have only two classes: healthy and sick. The "sick" class is the main class of interest, so it is the positive class.

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

9.1 Binary Classification

In addition to being the most common classification task, binary classification is the simplest. Most other classification tasks, like multiclass, multilabel and hierarchical classification, can be decomposed into a set of binary classification tasks. In these cases, the final classification is a combination of the output from binary classifiers. This is discussed in Chapter 11.

Let us look at an example of a binary classification task using the familiar data set of our contacts in a social network tool. Suppose you want to go out for dinner and you want to predict who, among your new contacts, would be a good company. Suppose also that, to make this decision, you can use data from previous dinners with people in your social network and that these data have the following three attributes: name, age and how the last dinner experience with that person was (Table 9.1). The "how was the dinner" attribute is a qualitative attribute that has two possible values: good and bad. Suppose further that Table 9.1 has the values of these three attributes for all your contacts. Figure 9.1 illustrates how the data are distributed in this data set.

Using this data set, you can induce a simple classification model by applying a classification algorithm to identify a vertical line able to separate objects into the two classes. For example, it is possible to induce a classification model represented by a linear equation $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \times x$, which, in our case, since $\hat{\beta}_0 = 0$ and $\hat{\beta}_1 = 1$, would be simplified to *dinner* = *age*, where y is the target attribute *dinner* and x is the predictive attribute *age*. The classification of objects according to this equation can be represented by a simple rule that says: everybody whose age is less than 31 was bad a company at the last dinner. The others are classified as good company. Thus a simple model can be induced, which can be a rule saying:

Table 9.1 Simple labeled social network data set for model induction.

Name	Age	Company
Andrew	51	Good
Bernhard	43	Good
Dennis	82	Good
Eve	23	Bad
Fred	46	Good
Irene	29	Bad
James	42	Good
Lea	38	Good
Mary	31	Bad



Figure 9.1 Simple binary classification task.

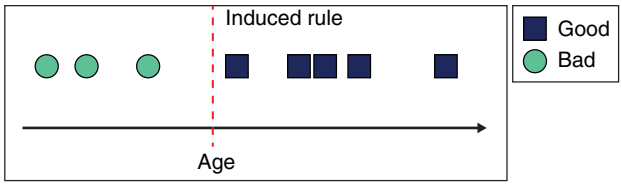


Figure 9.2 Classification model induced for the binary classification task.

Table 9.2 Simple labeled social network data set 2.

Name	Age	Company
Andrew	51	Good
Bernhard	43	Good
Dennis	82	Good
Eve	23	Bad
Fred	46	Good
Irene	29	Bad
James	42	Bad
Lea	38	Good
Mary	31	Good

If person-age < 32
Then dinner will be bad
Else dinner will be good

The application of this rule to the data set in Table 9.1 creates a dividing line separating objects in the two classes, as illustrated in Figure 9.2. Any other vertical line separating the objects from the two classes would be a valid classification model too.

However, this data set is very well-behaved. Classification tasks are usually not that easy. Suppose your data set was actually the data in Table 9.2, which is illustrated in Figure 9.3. A simple rule like the one used for the data set in

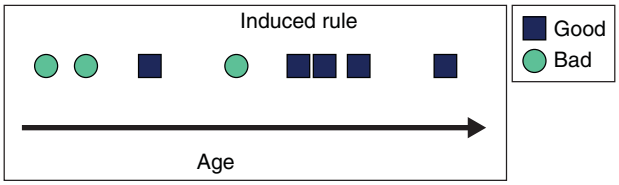


Figure 9.3 New binary classification task.

Table 9.3 Simple labeled social network data set 3.

Name	Age	Education level	Company
Andrew	51	1.0	Good
Bernhard	43	2.0	Good
Eve	23	3.5	Bad
Fred	46	5.0	Good
Irene	29	4.5	Bad
James	42	4.0	Good
Lea	38	5.0	Bad
Mary	31	3.0	Good

Table 9.1 does not work any more; neither would any other rule represented by a vertical line separating objects.

An alternative to deal with this difficulty is to extract an additional predictive attribute from our social network data that could allow the induction of a classification model able to discriminate between the two classes. In this case, we will add the attribute "Education level". The new data set, now with eight objects, can be seen in Table 9.3. The distribution of the data in this data set is shown in Figure 9.4.

We could have used the name as the second predictive attribute. However, it does not make sense to use the name to distinguish people who are good company from those who are not.

Since a classification task with two predictive attributes can be represented by a graph with two axes, the representation is two-dimensional. Up to three predictive attributes, it is possible to visualize the data distribution without a mathematical transformation.

Next, a classification algorithm can be applied to this data set, without the "Name" predictive attribute, to induce a predictive model. The predictive model induced could be represented by a linear equation, of the type $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \times x$, which, in our case, would be represented by a linear function of the form

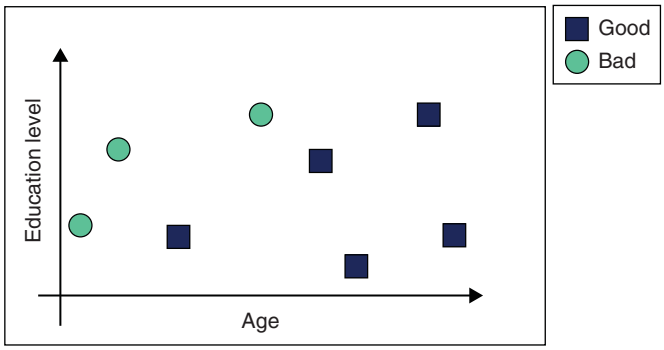


Figure 9.4 Binary classification task with two predictive attributes.

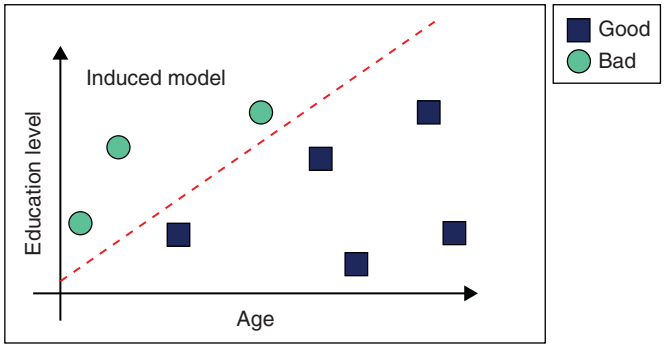


Figure 9.5 Classification model induced by a classification algorithm for the classification task with two predictive attributes.

$dinner = \hat{\beta}_0 + \hat{\beta}_1 \times age$. This is shown in Figure 9.5. As in the simpler classification task, with one predictive attribute, an infinite number of solutions can be found that divide the data into the two classes.

In binary classification tasks with two predictive attributes, when a straight line can separate the objects of the two classes, we say that the classification data set is linearly separable. Figure 9.5 shows a two-dimensional linearly separable binary classification task.

A binary classification data set with three predictive attributes is linearly separable if a plane divides the objects into the two classes. For more than three predictive attributes, a classification data set is linearly separable if a hyperplane separates the objects into the two classes. Figure 9.6 shows a three-dimensional linearly separable binary classification task, where the third axis (third predictive atribute) is the number of years of friendship.

When a classification data set is non-linearly separable, more complex decision borders must be created, so more complex classification models must be

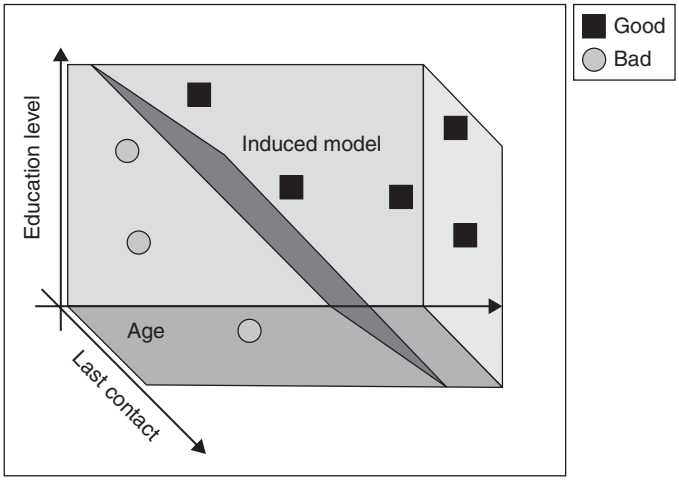


Figure 9.6 Linearly separable classification task with three predictive attributes.

induced. When the decision border becomes more complex, it also becomes more difficult to induce a model that can find this border using simple classification techniques. Algorithms with clever heuristics are necessary to induce functions that will find complex borders. However such algorithms typically tend to overfit, taking into account unnecessary details present in the data set.

9.2 Predictive Performance Measures for Classification

The correct classification rate of objects obtained by a classification model must be better than the correct classification rate obtained by placing all objects in the class with the largest number of objects, the "majority class". A classifier that places every new example in the majority class is known as a majority class classifier.

When dealing with a classification task, it is necessary to evaluate how well the induced model solves the task. Thus evaluation is also important when we have to decide the best learning algorithm for the given task.

The main concern of most data analytic applications is the classification model's predictive performance, which is associated with how frequently the predicted labels are the correct, true, labels. Several measures can be used to assess the predictive performance of a classification model. Most of these measures were developed for binary classification. Thus, without loss of generality, we will discuss predictive performance measures for binary

Figure 9.7 Confusion matrix for a binary classification task.

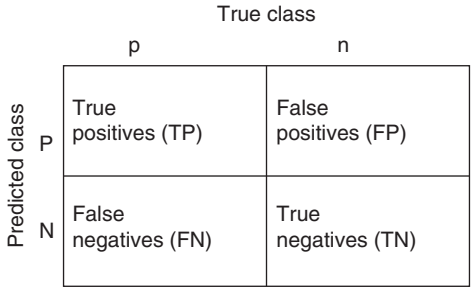
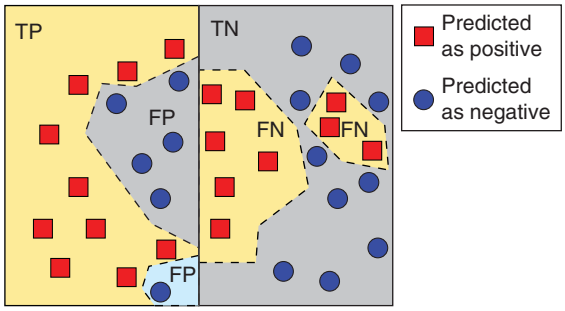


Figure 9.8 Data distribution according to predictive attribute values, true class and predicted class for a fictitious dataset.



classification tasks. The measures discussed here can be easily adapted, or have an equivalent, for classification tasks with more than two classes.

The main classification performance measures can be easily derived from a confusion matrix, a 2×2 matrix that shows, for a set of objects whose classes were predicted by a classifier and for each class, how many objects were correctly classified and how many were misclassified. Figure 9.7 shows a confusion matrix. We can use this table to see where we correctly predicted the class of the objects and where we incorrectly predicted the class.

The two rows represent the predicted classes and the two columns the true, correct, classes. The correct predictions are the values on the main diagonal. This diagonal shows the number of TPs and TNs. The incorrect predictions (FPs and FNs) are shown in the secondary diagonal. The value on the top right is the FPs. Similarly, the value on the bottom left refers to the FNs.

The predicted and true data classes can be illustrated by plotting each example using only two predictive attributes. To better illustrate the data distribution regarding TP, TN, FP and FN, Figure 9.8 uses a fictitious dataset, to show, for each object from each class, a distribution of objects whose classes were correctly predicted and incorrectly predicted.

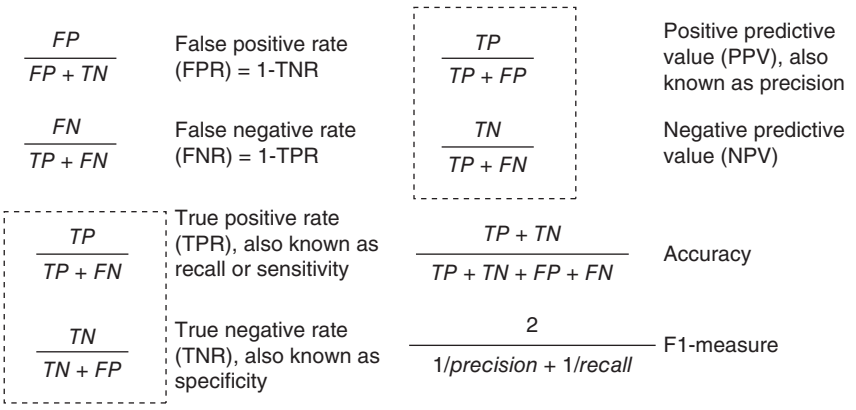


Figure 9.9 Predictive performance measures for a binary classification task.

The predictive performance measures currently used to evaluate classification algorithms and classification models induced by these algorithms are a combination of these four simple measures.

Two simple error measures frequently used in classification tasks, and easily derived from Figure 9.7, are:

- the false positive rate, also known as type I error or false alarm rate
- the false negative rate, also known as type II error or miss.

These measures are useful, but they are not the only measures that can be extracted from the confusion matrix. Figure 9.9 illustrates these and other predictive performance measures that can be extracted from the matrix.

The true positive rate (TPR) or recall measure returns the percentage of true positive objects that are classified as positive by the induced classifier. It reaches its highest value when all objects from the positive class are assigned to the positive class by the induced classifier. Recall is also known as sensitivity. The complement of the recall is the false negative rate (FNR).

The equivalent to the recall measure for the negative class is the true negative rate (TNR) or specificity. This measure returns the percentage of objects from the negative class that are classified by the induced classifier as negative. Thus, its value is higher when all negative examples are correctly predicted as negative by the induced classifier. The complement of the specificity is the false positive rate (FPR).

Other measures to evaluate the predictive performance of induced classifiers are the positive predictive value (PPV), also known as precision, and the negative predictive value. Recall and specificity are non-forgetting measures, because they evaluate how objects from a class were assigned to that class. The precision measure returns the percentage of objects classified as positive that were really positive. It is higher when no negative object is classified as positive

Table 9.4 Extended simple labeled social network data set.

Name	Age	Education level	Company
Paul	48	1.0	Bad
Monica	43	2.0	Good
Lee	82	3.0	Bad
Eve	23	3.0	Bad
Fred	46	5.0	Good
Irene	29	4.5	Bad
James	42	4.1	Good
Lea	38	5.0	Bad
Mary	31	3.0	Good
Peter	41	1.0	Good
Mary	43	2.0	Good
Louis	82	3.0	Good
Jonathan	23	3.5	Bad
Levy	46	5.0	Good
Joseph	17	4.0	Bad
Omar	42	4.0	Good
Lisa	38	4.0	Bad
Elizabeth	31	2.0	Good
Taylor	46	5.0	Good
Yves	29	4.0	Bad

(no negative object was included in the negative class). Precision and recall are the most frequently used of these measures. They are very popular in information retrieval tasks.

To see these measures in practice, suppose we used the data set from Table 9.3 as training data to induce a classification model using a decision tree induction algorithm. We want to evaluate the classification model, which will be a decision tree, to predict the class of new objects. Since we want to evaluate how well our model predicts the class of new objects, we test it using a test set with labeled data. We want to see how well the class predictions from our model match the true classes. Our test set can be seen in Table 9.4. We previously decided that only two predictive attributes are relevant for predicting who will be good company: age and education level.

From this table, just counting the number of objects from each class that are correctly and incorrectly classified, we can derive the confusion matrix in Figure 9.10.

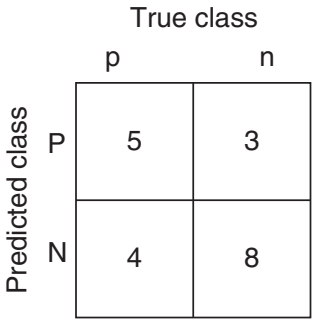


Figure 9.10 Confusion matrix for the example dataset.

Using this confusion matrix, we can easily calculate the predictive performance measures from Figure 9.9, which can be seen in Equations (9.1)–(9.8).

$$FPR = \frac{FP}{FP + TN} = \frac{3}{3 + 8} \tag{9.1}$$

$$FNR = \frac{FN}{TP + FN} \tag{9.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{9.3}$$

$$Specificity = \frac{TN}{TN + FP} \tag{9.4}$$

$$PPR(Precision) = \frac{TP}{TP + FP} \tag{9.5}$$

$$NPR = \frac{TN}{TN + FN} \tag{9.6}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9.7}$$

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{recall}} \tag{9.8}$$

These measures are very useful, but for some situations, like imbalanced data sets, when the number of objects in one class is significantly higher than the number of objects in the other class, they can be biased towards the majority class.

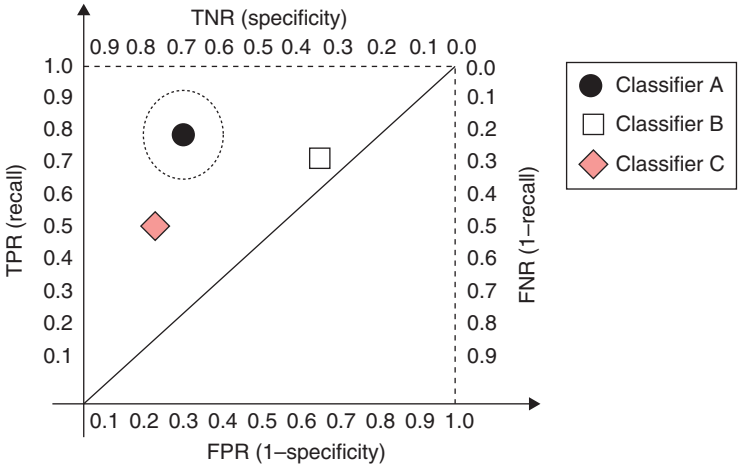


Figure 9.11 ROC graph for three classifiers.

A predictive performance measure that reduces this problem and is often used is to see how many objects from the positive class are classified as positive (measured by TPR, recall) – the more, the better – and how many objects from the negative class are mistakenly classified as positive (measured by FPR, 1 – *specificity*); the fewer, the better. If a classifier classifies all positive objects as positives and no negative object as positive, we have the perfect classifier, a classifier that does not make misclassifications. Thus, when comparing two or more classifiers, we just need to see these two measures. The best classifier will have the highest the TPR and the lowest FPR. A visual plot that helps to compare these measures for several classifiers is the receiver operating characteristics (ROC) graph [26, 27].

As an example, Figure 9.11 shows a ROC graph with the predictive performances of three classifiers. Each classifier is represented by a different symbol in the graph. The ideal classification model would be the classifier with the highest possible TPR, 1.0, and the lowest possible FPR, 0.0. The classifier with the best predictive performance is the classifier whose symbol is closer to the top left-hand corner, classifier A, as highlighted on the graph.

However, for most classifiers, different classification decisions, such as the use of different threshold values to decide the class of a new object, can result in different ROC points in a ROC graph. Therefore, the use of just one ROC point does not clearly illustrate a classifier's performance. In order to reduce this uncertainty, several points are taken, for instance, with different threshold values. By connecting the ROC points of a classifier obtained for different threshold values, we have a curve. The area under this curve, known as the area under the ROC curve (AUC) [26–28] is a better estimate of the classifier's predictive performance.

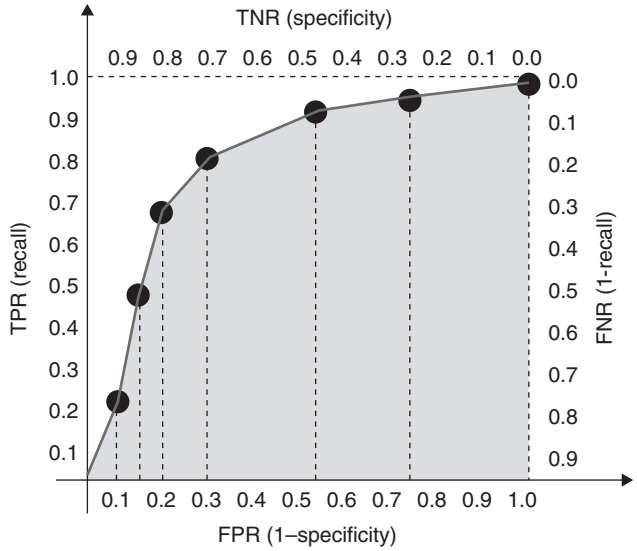


Figure 9.12 Classifier AUC with different ROC values.

Figure 9.12 illustrates a ROC graph with seven ROC points for a given classifier. Connecting these points generates the AUC for the classifier. To calculate the AUC, this figure shows that the area under the curve can be divided into seven trapezoids. Thus the area can be defined by simply adding the areas of the trapezoids.

Figure 9.13 shows different AUC situations. The top left-hand AUC, whose value is 0.5, is the AUC that would be obtained by a classifier that makes random predictions. In the bottom left, we have the AUC of a perfect classifier, with an area equal to 1.0. The top right the figure shows the shape for an AUC with an area of 0.75. To use AUC to select one of a set of possible classifiers, it is necessary only to calculate the AUC of each one, and select the classifier with the largest AUC. The bottom right graph of Figure 9.13 shows the AUC for two classifiers, A and B. Since classifier A has the largest AUC, it would be selected.

Thousands of classification algorithms can be found in the literature. These algorithms follow different principles and can be roughly categorized as:

- distance-based algorithms
- probability-based algorithms
- search-based algorithms
- optimization-based algorithms.

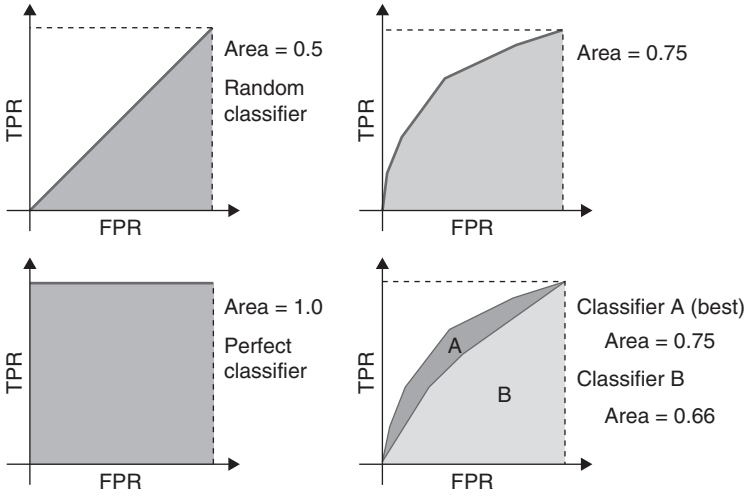


Figure 9.13 Different AUC situations.

The remainder of this chapter describes the main aspects of algorithms from the first two categories. The latter two categories are left to Chapter 10.

9.3 Distance-based Learning Algorithms

The simplest approach to predicting the class of a new object is to see how similar this object is to other, previously labeled, objects. Although very simple, distance-based learning algorithms are effective in several applications. The best-known algorithms based on this approach are the nearest neighbor (k-NN) and the case-based reasoning algorithms, which are introduced next.

9.3.1 K-nearest Neighbor Algorithms

The k-NN algorithm is one of the simplest classification algorithms. This algorithm is based on lazy learning, since it does not have an explicit learning phase. Instead, the algorithm memorizes the training objects, keeping them in memory. Whenever k-NN has to predict the class of a new object, it just identifies the class of the k objects most similar to this object. Since it only uses the class information of those objects most similar to the new object, k-NN uses a local-learning approach. The value of k defines how many neighboring labeled objects are consulted.

The pseudocode of the k-NN algorithm is presented next. It is possible to see that k-NN does not have an explicit learning phase.

Algorithm K-NN test algorithm.

INPUT D_{train}, the training set
INPUT D_{test}, the test set
INPUT d, the distance measure
INPUT x_i objects in the test set
INPUT K, the number of neighbors
INPUT n, the number of objects in the test set
for all object x_i in D_{test} **do**
    for all object x_j in D_{test} **do**
        Find the k objects from D_{train} closest to x_i according to the chosen distance measure d
    Assign x_i the class label most frequent in the k closest objects

To use the k-NN algorithm it is necessary to define the value of k, the number of nearest neighbors that will be considered. A very large value may include neighbors that are very different to the object to be classified. Neighboring objects very far from the new object are not good predictors of its class. It also makes the prediction to tend to the majority class. If the value of k is too small, only objects very similar to the object to be classified will be considered. Thus, the amount of information used to classify the new object may not be enough for a good classification. This can result in a unstable behavior, since noisy objects can decide the classification of new objects. Figure 9.14 shows how the value of k can affect the classification of a new object. It illustrates the neighbors considered by using two different k values and the majority class for each of these situations.

One problem faced by k-NN is that classification of a new object can be slow, since it is necessary to calculate the distance between it and all the objects in the training set. There are two ways to deal with this problem:

- Apply an attribute selection technique to reduce the number of attributes and, as a result, reduce the time necessary to calculate the distance between objects.
- Eliminate some examples from the training set, thus reducing the number of objects whose distance to the new object must be calculated. A way to do this is to store only a set of object prototypes for each class.

Attribute selection techniques were described in Chapter 4. For the elimination of examples, two alternatives can be employed: sequential object elimination and sequential object insertion.

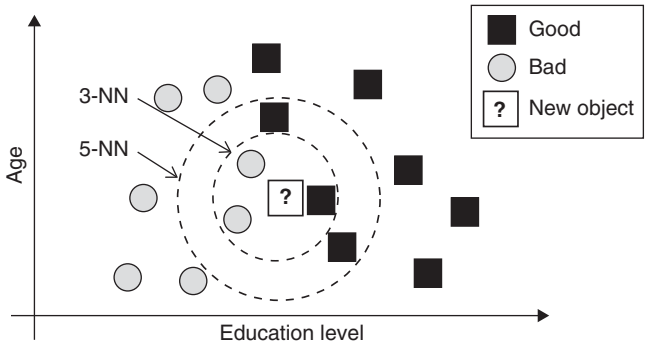


Figure 9.14 Influence of the value of *k* in k-NN algorithm classification.

The sequential elimination method starts with all objects and iteratively discards objects that are correctly classified by the class prototypes. The sequential insertion approach goes in the opposite direction: it starts with only one prototype per class and sequentially adds objects whose class label is incorrectly predicted by the prototypes.

The original k-NN algorithm used majority voting among the classes associated with the *k* nearest neighbors. An efficient variation weighs the vote of each one of the *k* nearest objects by the inverse of the distance between this object and the object to be labeled. Thus, the smaller the distance, the more important is the vote (class label) of the object.

Assessing and evaluating results Since K-NN follows a lazy learning approach, it has no model. For that reason the unique results that can be assessed are the predictions made.

Setting the hyper-parameters The k-NN algorithm has only one input hyper-parameter, the number of nearest neighbors *k*. When using majority voting in a classification problem or the average in a regression problem, the value of *k* should be set using validation data. The best value for *k* is data set dependent. When using the inverse of the distance between the test object and the nearest neighbors to weigh the votes or the average, *k* can be set to the number of training objects.

Advantages and disadvantages of k-NN The advantages and disadvantages of k-NN are set out in Table 9.5.

Finally, it is worth noting that K-NN can also be used for regression. Instead of using majority voting to decide the predicted value of a new object, the average of the label values of the k nearest objects can be used. Just as in classification

Table 9.5 Advantages and disadvantages of k-NN.

Advantages	Disadvantages
• Simplicity • Good predictive power in several problems • Inherently incremental: if new objects are incorporated into the training set, they will be automatically considered when the algorithm needs to predict the class of a new object	• Long time required to classify new objects • Uses only local information to classify new objects • Sensitive to presence of irrelevant attributes • Predictive quantitative attributes should be normalized • Somewhat sensitive to the presence of outliers • No model, so no interpretation possible.

the votes can be weighted by the inverse of the distances, in regression the average can also be weighted by the inverse of the distances.

9.3.2 Case-based Reasoning

Case-based reasoning (CBR) is a distance-based algorithm very application oriented. It tries to solve new problems by finding similar problems and adapting their solutions. For such, it uses a record of previous cases [29]. Each case has two components: the case description (problem to be solved) and the case solution, a solution (experience) used to solve the problem. A typical CBR system has four processes: retrieve, reuse, revise and retain [29]. Figure 9.15 illustrates the role of these processes.

A typical CBR cycle works in the following way. Whenever a user has a new problem to be solved, the user presents the problem description to the CBR system. The system retrieves the k most similar cases, according to the problem description. Another distance-based algorithm, such as k-NN, is usually employed to find the k most similar cases. The solution part of the retrieved cases can be reused to solve the new problem. As such, it may be necessary to revise the solution, adapting it to specific features of the new problem. If the revised solution, or parts of it, may be useful in the future, a new case, with the problem description and its solution, is retained in the case base. Learning occurs in the case revision and case retaining processes.

In recent years there has been growing interest in the use of CBR in analytic and big data problems. Among the applications of CBR to big data, support tools based on CBR have been proposed to help decision making by busy managers.

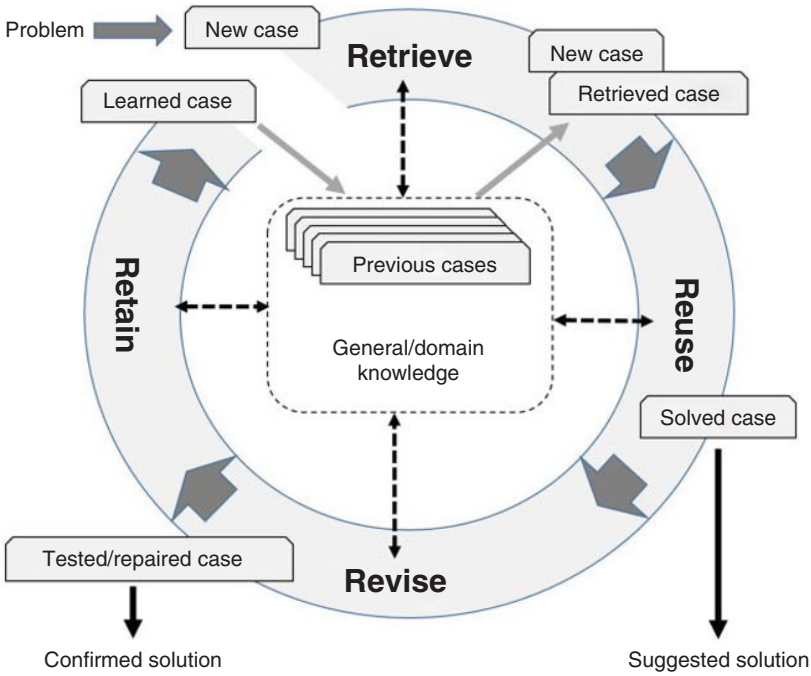


Figure 9.15 Case-based reasoning, adapted from [29].

9.4 Probabilistic Classification Algorithms

The CBR algorithm has good predictive performance in several classification tasks using a deterministic approach. By deterministic we mean that if we run the algorithm several times on the same data set, using the same experimental setup, the results will be the same. However, several classification tasks are not deterministic. This happens because the relationship between the predictive attributes and the target attribute of a data set can be probabilistic, and not all important information may be captured by the predictive attributes used. In real data, the information captured by these attributes is usually incomplete or imprecise.

Thus, in several classification tasks it is important to estimate the probability of an object belonging to each class, given the information we have. Probabilistic ML algorithms can model the probabilistic relationship between the predictive attributes and the target attribute of a data set. To do this, they use a well known statistical approach known as Bayes' theorem or Bayes' rule. Techniques using Bayes' theorem are known as being based on Bayesian statistics.

To explain how Bayes' theory works, suppose there is an epidemic of severe acute respiratory syndrome (SARS) in a city. The city hospital has a small number of physicians, and therefore cannot cope with the very large increase in the number of people looking for a medical diagnosis. The hospital then needs a fast alternative to reduce the burden without decreasing the quality of the diagnoses. This can be achieved by prioritizing those patients whose risk of having contracted SARS is higher.

So far, the hospital physicians have been able to examine and diagnose 100 patients. For each patient, they checked whether the patient has a sore throat and headache. Based on what they have seen, they can say that:

- 85% of the patients with SARS have sore throat.
- 40% of the patients without SARS have headaches.

In Bayesian statistics, these frequencies are called "prior" probabilities.

Suppose further that Mary was not feeling well, so she went to the hospital, with a sore throat, but without a headache. She heard about the SARS epidemic and is very worried. She would like to know what are the chances – probability – that she has SARS. The probability of having a disease given the presence/absence of some symptoms is known as the "posterior" probability. More formally, the posterior probability is the probability of the occurrence of an event given that another event occurred or did not occur. When we want to classify new objects by predicting the target label given the predictive attribute values, we are trying to predict the posterior probability.

Depending on the data set, it is easier to estimate some prior and posterior probabilities than others. In the previous example, defining the posterior probability of having a sore throat given the diagnosis, is easier than defining the posterior probability of having SARS given the presence/absence of a sore throat. Bayes' theorem provides a simple way to estimate the second posterior probability.

For classification tasks, Bayes' theorem calculates the probability of an object x belonging to a class y, given:

- the probability of the class y occuring
- the probability of this object x occurring in class y and
- the probability of object x occuring.

Usually, a simplification of this theorem, without the probability of the object x occuring, is used. Equation (9.9) presents the Bayes' theorem in a formal way.

$$P(y|X) = P(X|y) \times P(y)|P(X) \tag{9.9}$$

In this equation, X is in uppercase because it is a vector of predictive attribute values and y is in lowercase because it is a single value, a class label. Probabilities in the format $P(A|B)$ are called conditional probabilities, since they measure the probability of A occuring conditioned to the occurrence of B, or given that B

occurs. $P(y|X)$ is the probability of class y occurring given that the predictive attribute values are X, and $P(X|y)$ is the probability that predictive attributes with values X occur given that the class is y.

In the terminology of Bayesian probability, $P(y|X)$ is the posterior probability that the object X belongs to the class y, $P(X|y)$ is the likelihood that the object X can be found in the class y, $P(y)$ is the prior probability of class y and $P(X)$ is the evidence we have.

To use Bayes' theorem to predict the class of a new object, parameter values must be obtained from an incredibly large data set. Given the difficulty involved, the parameters are approximated. There are two categories of probabilistic learning algorithms to approximate these parameters: generative and discriminative algorithms.

Discriminative algorithms, like logistic regression, approximate the parameters for the probability of an object X belonging to a class y. Generative algorithms, which include naïve Bayes (NB) algorithms, approximate the parameters of the two other probabilities: the probability of the class y occurring and the probability of the object X occurring in the class y.

In this section, we will briefly explain how logistic regression and NB algorithms work. We will assume, without loss of generality, a classification task with two predictive attributes and two possible classes.

9.4.1 Logistic Regression Algorithm

As previously seen, a simple way to discriminate between objects with two predictive attributes from two classes is to find a straight line that separates the objects of the two classes. This line is defined by a linear function known as a discriminant function. The challenge is to find the equation of this line. The function relates the values of the predictive attributes – two in our example – to a value associated with the class. After some mathematical transformations, it has the form shown in Equation (9.10); that is, it is a linear model (See Section 8.2.1):

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \times X \tag{9.10}$$

However, while in binary classification tasks using probabilistic algorithms the output value of the discriminant function is between 0 and 1, the two possible extremes of the probability that the object belongs to this class, the function in Equation (9.10) can have values between $-\infty$ and $+\infty$. These values can be transformed into probabilities using logistic regression.

Although it has the term regression in its name, logistic regression is used for classification tasks. It estimates the probability of an object belonging to a class. For such, it adjusts a logistic function to a training data set. This logistic function generates a straight line separating the objects of the two classes.

In contrast to the linear function, this function produces values in the [0, 1] interval.

Initially, logistic regression calculates the odds of an object belonging to each of the two classes, which are values in the [0, 1] interval. The ratio of the odds for the two classes is calculated and a logarithmic function is applied to the result. The final result, in the $[-\infty, +\infty]$ interval, is known as the log-odds, or logit. Linear regression, a regression technique described in Section 8.2.1, can then be used to find the discriminant function, a linear function.

Thus, it can be seen that logistic regression induces a linear classifier. It allows estimation of class probabilities using the line equation obtained by a discriminant function.

Since logistic regression returns a probability, a quantitative value, this value has to be transformed into a qualitative value, a class label. In binary problems, when that probability is less than 0.5 the negative class is predicted. Otherwise the positive class is predicted.

Let us use as an example our social network data set as presented in Table 9.4. The age and the educational level are predictive attributes and the good/bad company rating is the target attribute. A simple regression model that can be obtained for this example is:

$$\hat{l} = -0.83338 + 0.03707 \times Age - 0.13133 \times Education\ Level.$$

Now let us apply this equation to a new instance. Andrew is 51 years old and has an education level of 1.0. The logit is as follows:

$$\hat{l} = -0.83338 + 0.03707 \times 51 - 0.13133 \times 1.0 = 0.92535.$$

This value could be any value in IR. Using the logistic distribution function we obtain:

$$P(\hat{y} < 0.92535) = \frac{1}{1 + e^{-\hat{l}}} = \frac{1}{1 + e^{-0.92535}} = 0.716.$$

Since $0.716 \geq 0.5$, the positive class of being good company is predicted.

Logistic regression can be applied to data sets with any number of predictive attributes, which can be qualitative or quantitative. For more than two classes, multinomial logistic regression can be used.

Assessing and evaluating results The main result of logistic regression is the estimates of the linear regression coefficients $\hat{\beta}_i$, which relate the values of the predictive attributes to the logit value. The logit value is transformed into a probability value, which is then transformed into a qualitative value (class).

Setting the hyper-parameters Logistic regression has no hyper-parameters.

Advantages and disadvantages of logistic regression The advantages and disadvantages of logistic regression are set out in Table 9.6.

Table 9.6 Advantages and disadvantages of logistic regression.

Advantages	Disadvantages
• Easily interpretable • No hyper-parameters	• Restricted to linearly separable binary classification tasks • Number of instances must be larger than number of attributes • Sensitive to correlated predictive attributes • Sensitive to outliers

9.4.2 Naive Bayes Algorithm

NB is a generative ML algorithm. Generative classification algorithms induce classification models based on joint probabilities, the probability that a given object belongs to a particular class. To do this, they use Bayes' theorem, defined by Equation (9.9).

The Bayes' theorem is used to calculate the probability of an object belonging to a particular class. Thus, to predict the class of a new object, if we have m classes, we use the Bayes theorem m times, one for each class, as shown in Equation (9.14).

$$P(y_a|X) = P(X|y_a) \times P(y_a)|P(X) \tag{9.11}$$

$$P(y_b|X) = P(X|y_b) \times P(y_b)|P(X) \tag{9.12}$$

$$\ldots \tag{9.13}$$

$$P(y_m|X) = P(X|y_m) \times P(y_m)|P(X) \tag{9.14}$$

Since $P(X)$ is common to all class equations, it can be removed. Taking into account that all the classes have the same *a-priori* probability – that is, all classes i have the same value for $P(y_i)$ – this term can also be eliminated because this elimination does not change the order of the most probable classes. Indeed, the goal is not to have the probability value per class but the classes ordered by decreasing order of probability. Thus, the equations are simplified, as shown in Equation (9.18).

$$P(y_a|X) = P(X|y_a) \tag{9.15}$$

$$P(y_b|X) = P(X|y_b) \tag{9.16}$$

$$\ldots \tag{9.17}$$

$$P(y_m|X) = P(X|y_m) \tag{9.18}$$

The class i with the highest probability value, $P(y_i|X)$, is assigned to the object X. Thus, NB can be used in classification tasks with any number of classes.

To use Bayes' theorem, we need to know the value of $P(X|y_i)$, where X is a vector of p values, one for each predictive attribute of the object. If we consider that the value of some predictive attributes depend on the values of other attributes, the calculus of $P(X|y_i)$ requires several intermediate calculations whose estimate depends on the number of training examples available for each class. For example, if an object has p predictive attributes, $P(X|y_i)$ is defined as in Equation (9.20):

$$P(X|y_i) = P(x_1, x_2, ..., x_p|y_i) = \tag{9.19}$$

$$= P(x_1|x_2, ..., x_p, y_i) \times P(x_2|x_3, ..., x_p, y_i) \times ... \times P(x_p|y_i) \times P(y_i) \tag{9.20}$$

To simplify these calculations, NB assumes that the predictive attributes are independent, so instead of using Equation (9.20), $P(X|y_i)$ is defined by Equation (9.22):

$$P(X|y_i) = P(x_1, x_2, ..., x_p|y_i) \tag{9.21}$$

$$P(X|y_i) = P(x_1|y_i) \times P(x_2|y_i) \times ... \times P(x_p|y_i) \tag{9.22}$$

Assessing and evaluating results The main result of NB algorithms are the p conditional probabilities, as shown in the right-hand term of Equation (9.22). This information is very meaningful as it allows us to obtain the empirical distribution for each predictive attribute per class, as exemplified in Figure 9.16 for the example data set.

Setting the hyper-parameters NB has no hyper-parameters

Advantages and disadvantages of the NB approach The advantages and disadvantages of the NB approach are set out in Table 9.7.

9.5 Final Remarks

Many current data analytics applications are predictive tasks. This is one of the main reasons a very large number of learning techniques for predictive tasks have been developed in recent decades.

Although simple, the learning algorithms in this chapter can induce predictive models with high predictive performance. In many application tasks k-NN has a predictive performance superior to more sophisticated classification

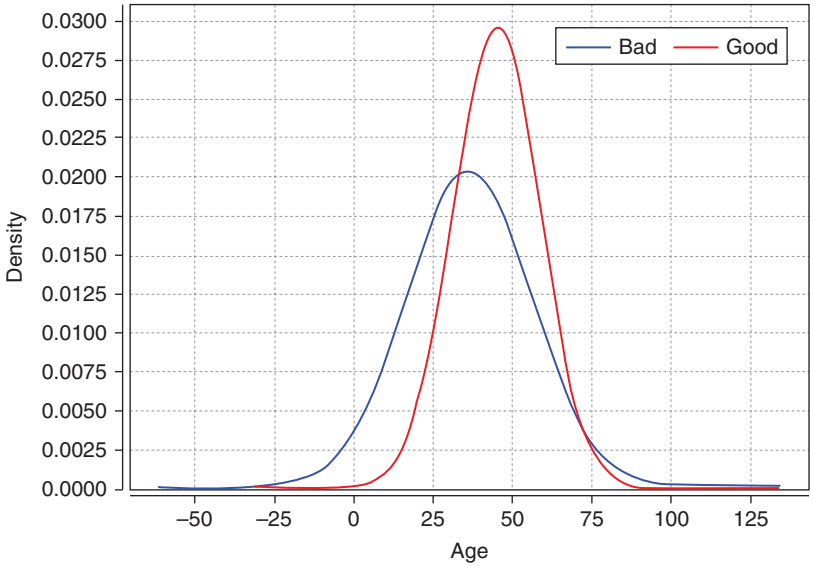


Figure 9.16 The age empirical distributions for people who are good and bad company.

Table 9.7 Advantages and disadvantages of NB

Advantages	Disadvantages
• Good predictive performance in classification tasks where the predictive attributes are independent • Robust to the presence of noise data and irrelevant attributes • Cast training, since it induces classification models by looking at the training set only once • Prediction of class labels for new objects is also fast • Induced models are easy to interpret • No hyper-parameters	• One of the reasons for the fast learning is also a limitation of NB: it does not take into account relation between predictive attributes • Can benefit from feature selection • Struggles to deal with continuous quantitative values in the predictive attributes

algorithms. The same observation applies to logistic regression and NB, which, additionally, are based on important concepts from probability theory. The next chapter describes classification algorithms based on two other learning approaches: search and optimization.

9.6 Exercises

1 Describe five tasks you do every day, which are classification tasks.

2 How can a regressive task be converted to a classification task?

3 When does the extraction of additional predictive attributes help to deal with the classification task and when does it harm the task?

4 How do you relate, in an informal way, and using different terms from those used in the chapter text, the relationship between precision and specificity.

5 Why is accuracy not a suitable predictive performance evaluation measure for imbalanced data sets?

6 Given the social network dataset from Table 9.4, using the first ten objects as the training set and the last ten objects as the test subset, perform the following activities:
   a) Use the k-NN algorithm, with different values of k ($k = 2, k = 3$ and $k = 5$), to predict the class label of the test subset objects.
   b) Perform the same experiments but now exchanging the training subset and the test subset (the last ten objects will be used for training and the first ten objects will be used for testing).

7 Compare the results from the two previous sets of experiments for each value of k. Are they the same? Different? Explain why.

8 Repeat the experiments from the previous question using logistic regression. Compare the experimental results with those obtained using k-NN. What can you say from this comparison?

9 Why is posterior probability used for class label prediction instead of the prior probability?

10 Repeat the experiments from the previous question using the NB algorithm. Compare the experimental results with those obtained using k-NN and logistic regression. What can we say from this comparison?

10

Additional Predictive Methods

Many predictive tasks can be much more complex than those presented in Chapters 8 and 9. As such, they may need more sophisticated learning algorithms. This chapter will describe a new group of predictive learning algorithms – search-based and optimization-based algorithms – which allow us to deal efficiently with more complex classification tasks.

10.1 Search-based Algorithms

Several ML algorithms use algorithms themselves: they perform iterative local searches aiming to induce the best predictive model, a local optimum. This is the case for decision tree induction algorithms (DTIAs), a commonly used approach for designing search-based algorithms. DTIAs induce models with a tree-shaped decision structure where each internal node is associated with one or more predictive attributes and each leaf node is associated with a target value. There are DTIAs for classification tasks, known as classification trees, and for regression tasks, known as regression trees. To distinguish between these uses, DTIAs are also called characteristic tree induction algorithms (CTIAs), reserving the term "decision tree" for classification tasks and keeping the term "regression tree" for regression tasks. In this book we consider that classification and regression are types of decision and therefore we adopt "decision tree" as the general term and use the classification and regression tree terms for the specific tasks.

Another approach for search-based algorithms – rule set induction algorithms (RSIAs) – search for the model, represented by a set of rules, with the best predictive performance. Each rule has an IF-THEN format: IF an object obeys a set of conditions THEN it belongs to a given class. Both approaches induce predictive models using concepts from logic. RSIAs can be seen as a special case of association rules, which are covered in Chapter 6. To make the text more concise, only DTIAs are covered in the book.

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

Table 10.1 Simple labeled social network data set 3.

Name	Pain	Temperature	Outcome
Andrew	no	high	Home
Bernhard	yes	high	Hospital
Mary	no	high	Home
Dennis	yes	low	Home
Eve	yes	high	Hospital
Fred	yes	high	Hospital
Lea	no	low	Home
Irene	yes	low	Home
James	yes	high	Hospital

10.1.1 Decision Tree Induction Algorithms

Some data analytics applications require a predictive model that is easy to understand; that is, an interpretable model. Humans are good at interpreting flowcharts, where data follows a specific branch according to its value. A flowchart is even easier to interpret if it assumes the format of a tree, like a decision tree. Decision trees are frequently used in decision support systems. They show possible decisions and the outcome of each decision.

Something similar is found in ML algorithms that induce decision trees. DTIAs create a tree-like hierarchical structure, where each node is usually associated with a predictive attribute. To see what the decision tree looks like, let us consider the classification task data set in Table 10.1. This data set has three predictive attributes and one target attribute, the latter having two classes associated with medical diagnosis. The two predictive attributes that are relevant for induction of a diagnosis model are:

- if the patient has pain or not
- if the patient has a high or low temperature.

The two classes are: go to a hospital or go home. Figure 10.1 illustrates a possible decision tree structure produced by a DTIA for this data set. It is easy to understand the knowledge represented by this decision tree. Each path from the root node to a leaf node can be seen as a rule.[1] Thus, there are three rules:

- If a patient has pain and high fever, the patient should go to a hospital.
- If a patient has pain and low fever, the patient should go home.
- If a patient has no pain, the patient should go home.

1 This also shows how simple it is to convert a decision tree to a set of rules.

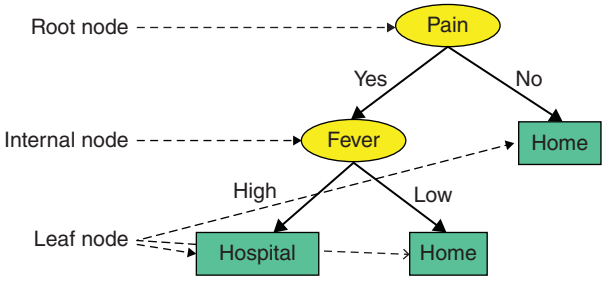


Figure 10.1 Example of a decision tree.

Thus, like RSIAs, in the end, DTIAs also derive a set of rules from a data set, and these rules can be used to predict the label of a new object, be it a class or a numerical value. However, while a DTIA will produce a rule for every possible combination of predictive values, a RSIA only covers some of the combinations. The other combinations are usually covered by a default rule.

Easy model interpretability is an important characteristic of decision trees. There are several ML algorithms for decision tree induction. One of the first to be proposed, the Hunt algorithm, inspired several other DTIAs, including

- the classification and regression tree (CART)
- the iterative dichotomiser 3(ID3)
- C4.5, which is an extension of ID3
- very fast decision trees (VFDT), which are popular for incremental data sets like data streams.

Algorithm Hunt decision tree induction algorithm.

```
INPUT D_train current node training set
INPUT p the impurity measure
INPUT n the number of objects in the training set
if all objects in D_train belongs to the same class y then
    The current node is a leaf node labeled with class y
else
    Select a predictive attribute to split D_train using the impurity measure p
    Split D_train in subsets according to its current values
    Apply Hunt algorithm to each subset
```

The Hunt algorithm is very simple, as shown by its pseudocode. This algorithm calls itself. In computing, algorithms that call themselves are called recursive algorithms. In the algorithm, each new call to itself expands the tree size.

Recursive algorithms solve problems by adopting the "greedy strategy", also known as divide-and-conquer. In this strategy, a problem is solved by dividing

it into simpler, smaller, problems, and then applying the same strategy to them. This division continues until the problem is simple enough to be resolved without the need of further division. DTIAs usually follow the divide-and-conquer strategy.

Starting at a root node, the Hunt algorithm first checks whether all objects in the node have the same class (lowest impurity value, according to the impurity measure used). If they have, this node is a leaf, is labeled with one of the classes, and the algorithm stops. Otherwise, the algorithm divides the current objects in the node using a test predictive attribute (TPA). TPA is a predictive attribute selected from a training data set according to a given criterion. A criterion can be the predictive attribute that divides the current objects into the most homogeneous groups. The highest homogeneity occurs when all objects in a group belong to the same class. To do this, each test result is associated with a branch originating from the current node and ending in one of the previous groups, which are called child nodes. Thus, selected a TPA for the node, each current object, according to the comparison of the TPA with the node's corresponding predictive attribute, follows one of the branches. Next, the algorithm is applied to each child node.

The Hunt algorithm progressively divides the objects into less impure groups. The simplest situation is when all objects in a node belong to the same class. When this occurs, the objects in this node are no longer divided and the node is labeled with the class of its objects.

There are several DTIAs, based on the Hunt algorithm and alternative approaches. The main differences between existing DTIAs are in the following aspects:

- how to choose a predictive attribute for the current tree node;
- once the attribute has been chosen, how to divide the objects among the node branches;
- when to stop applying the algorithm to a node (stop tree growth).

The TPA selected is the attribute in the data set that best divides the objects in the current node. The ideal division would be the one that leaves all the child nodes with objects from only one class. To define the predictive attribute that gives the best division, impurity measures are used. The more homogeneous are the classes in a node, the lower the impurity measure value. Thus, the attribute with the lowest impurity value (most objects have the same class label) is selected. The most common impurity measures used in classification trees are the Gini index, as used in the CART algorithm, and the information gain, as used in the ID3 and C4.5 algorithms. Details of these and other impurity measures for classification trees can be found in the literature [30].

The division of the objects depends on the number of values and the predictive attribute scale of values. The simplest division occurs when the attribute has two values, when two branches are defined and each value is associated with

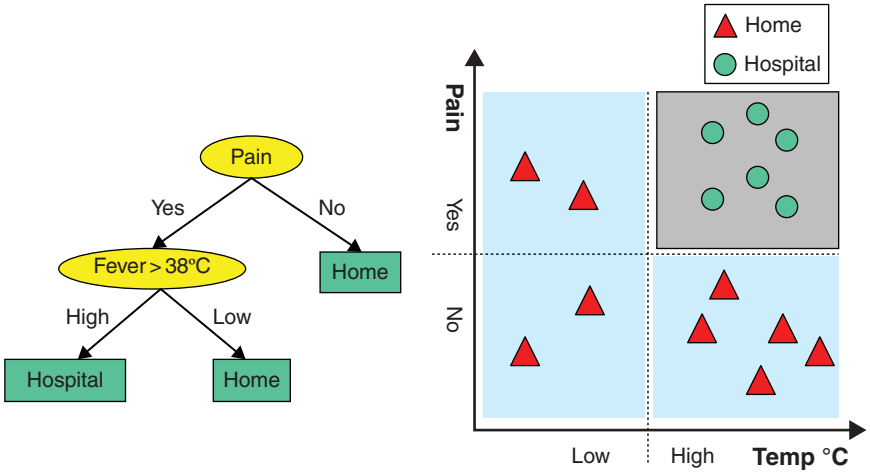


Figure 10.2 Input space partition by a decision tree.

one branch. For predictive attributes with more than two values, the scale and data set domain aspects are taken into consideration. Some DTIAs, like CART, allow only two branches per node. Thus, they induce only binary decision trees. Other algorithms have no such limitation.

Regarding the stop criteria, when to stop dividing the nodes and therefore growing the decision tree, the main criteria adopted are:

- The objects of the current node all have the same class
- The objects of the current node have equal values for the input attributes, but different class labels.
- The number of objects in the node is less than a given value.
- All predictive attributes have been included in the path from the root node to this node.

Every leaf node of a tree has a label that is attributed by majority voting; that is, the majority class of the objects in this leaf node. The prediction of a new unlabeled object involves starting at the root, and following the split rules according to the values of the predictive attributes. The predicted value is the label of the leaf node reached by the object.

Another important aspect of DTIAs is the partition of the input space by hyper-planes parallel to each axis, where each axis is associated with a predictive attribute. For a data set with two predictive attributes, decision trees divide the input space using vertical and horizontal lines. Figure 10.2 illustrates a possible decision tree structure produced by a DTIA for this data set.

If the decision tree induced by an algorithm is deep, measured by the number of nodes in the path from the root node to the farthest leaf node, the tree can exhibit overfitting, paying too much attention to details in the data set, instead

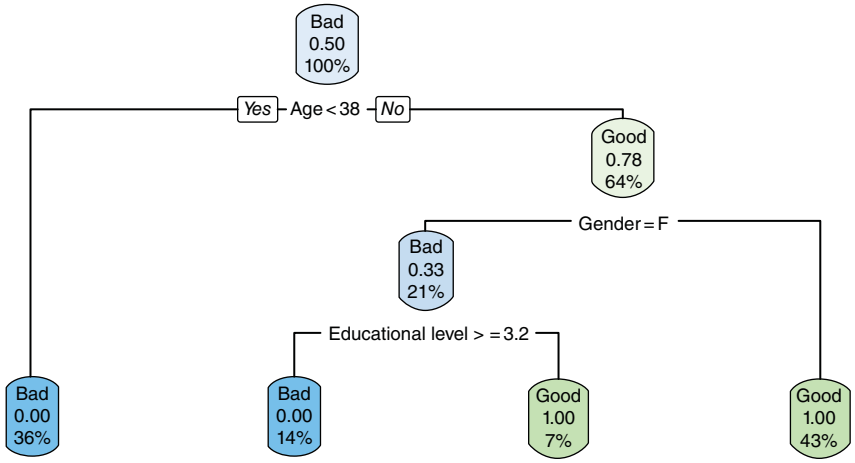


Figure 10.3 Graphical representation of the CART classification tree for our social data set in Table 8.5.

of the main patterns. This problem can be reduced by pruning the trees, like pruning trees in a garden. We cut tree branches that do not significantly affect the predictive performance of the decision tree. The pruning can occur:

- during the tree induction: pre-pruning
- after the tree induction: post-pruning.

Assessing and evaluating results The predictive performance of models induced by DTIAs can be assessed using the predictive performance measures used for regression tasks and for classification tasks.

Decision tree models are interpretable. They can be represented as a graph, as in Figure 10.3, or as a set of rules, as in Figure 10.4. Both figures show the selected predictive attributes, each one with the conditions adopted to divide the objects into two groups. Figure 10.3 also shows the purity value and the percentage of the total set of objects in each group. Figure 10.4 also shows the number of objects from each class and the purity value of each group.

Setting the hyper-parameters Each DTIA can have different hyper-parameters, the values of which need to be set. Some hyper-parameters that can be found in implementations of DTIAs are used to control the tree pruning (both pre- and post-pruning). The most common of these hyper-parameters is the minimum number of objects a leaf node must have. If its value is very low, it can result in overfitting.

Advantages and disadvantages of decision trees The advantages and disadvantages of decision trees are set out in Table 10.2.

```
(node), split, n, loss, vval, (vprob)
* denotes terminal node

1) root 14 7 Bad (0.5000000 0.5000000)
   2) Age< 37.5 5 0 Bad (1.0000000 0.0000000) *
   3) Age>=37.5 9 2 Good (0.2222222 0.7777778)
      6) Gender=F 3 1 Bad (0.6666667 0.3333333)
        12) Educational level>=3.25 2 0 Bad (1.0000000 0.0000000) *
        13) Educational level< 3.25 1 0 Good (0.0000000 1.0000000) *
      7) Gender=M 6 0 Good (0.0000000 1.0000000) *
```

Figure 10.4 Rules representation of the CART classification tree for our social data set from Table 8.5.

Table 10.2 Advantages and disadvantages of decision trees.

Advantages	Disadvantages
• Interpretable both as a graph and as a set of rules • Pre-processing free since robust to outliers, missing data, correlated and irrelevant attributes, and do not need previous normalization	• The definition of a rule to split a node is evaluated locally without enough information to know whether the rule guarantees the global optimum, i.e. the minimum number of objects misclassified after the tree is completed. This is due to the greedy search and can result in a sub-optimal solution. • Since the rules are of the type $x < a$, where x is a predictive attribute and a is a value, a decision tree splits the bi-dimensional space with horizontal and vertical lines, as shown in Figure 10.2, which makes some problems hard to deal with.

10.1.2 Decision Trees for Regression

There are two main differences between regression tree induction algorithms and classification tree induction algorithms:

- The impurity measures are necessarily different for regression and classification. For regression, the most common impurity measure is the variance reduction. This measures the difference between the variance on a node and the sum of the variances in its child nodes. The split is defined by the rule that maximizes this difference.
- The second difference is the labeling of the leaf nodes. While for classification majority voting is used, for regression it is the average. However, this option has an important limitation, which is that is predicts the same value for a relatively large input space. In other words, every unlabeled object that falls in a given leaf node will have the same prediction, as can be seen in Figure 10.5.

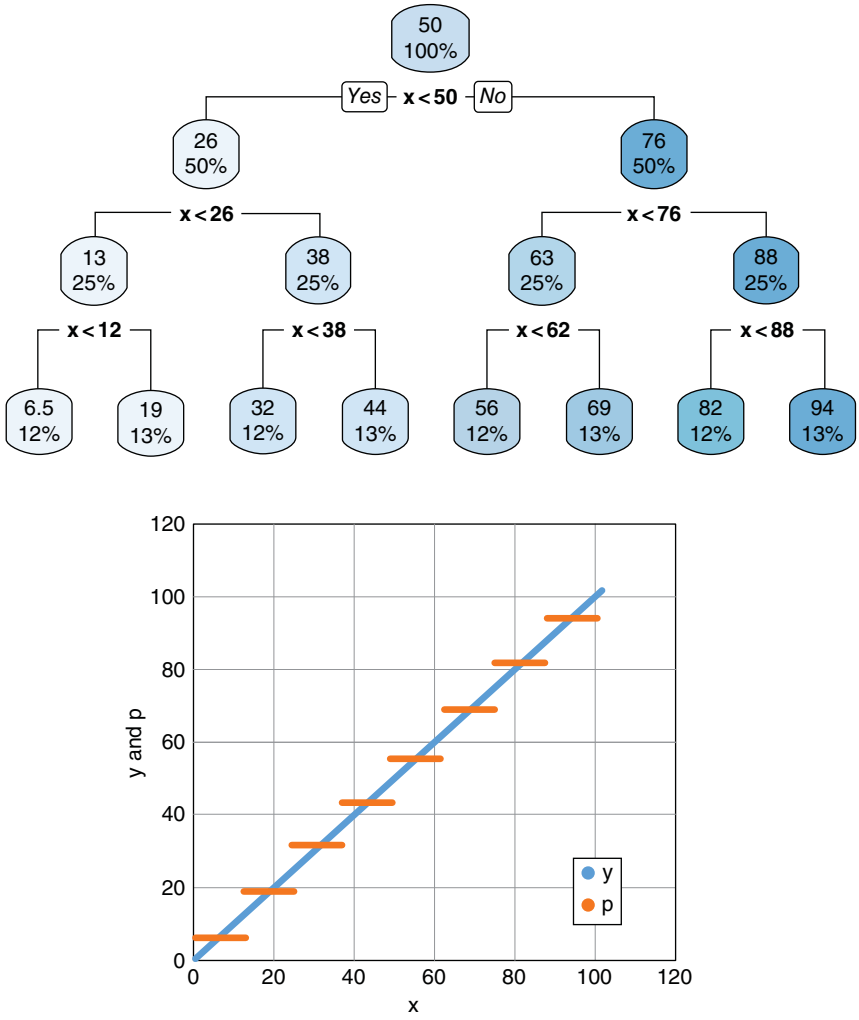


Figure 10.5 Example: $y = x$, where p is the prediction of y produced by a CART decision.

The methods described in the next two sections – model trees and MARS – were specially developed for regression, although some approaches use model trees for classification. Model trees and MARS are able to surpass the limitation of using the average in the leaf nodes of decision trees.

10.1.2.1 Model Trees

As illustrated in Figure 10.5, regression trees use as the prediction the average of the target values of the instances in the leaf node reached by a test instance. Regression model trees, also known as model trees, use multivariate linear

regression (MLR) instead of the average [31]. The same data that is used to calculate the average in a regression tree leaf node is used to fit the MLR in a model tree.

Model trees have a post-pruning step, which verifies for each non-leaf node whether its MLR accuracy is higher than the accuracy given by the subtree. If so, the subtree is pruned. In the example from Figure 10.5, the linear model $\hat{y} = x$ is used in the root node. The result is a tree without branches.

The concept of model trees has also been applied to classification using logistic regression instead of MLR.

10.1.2.2 Multivariate Adaptive Regression Splines

Multivariate adaptive regression splines (MARS) are a technique that sits between multivariate linear regression (MLR) and DTIAs. MLR assumes a linear relation between x and y while CART assumes a constant value at each leaf node. MARS and model trees with MLR in the leaf nodes are better able to better adjust the induced tree to the objects. Figure 10.6 shows the regression trees induced by MLR, CART, model trees with MLR in the leaves, and MARS for the same data set.

Model trees adjust MLR using the instances that fall in the same leaf node. In doing so, there can be, and usually there are, discontinuities in the MLRs of two neighboring leaf-nodes. There is always a point common to two neighboring "stretches" of the MARS model. This can be seen in Figure 10.6. In Figure 10.7 this difference is magnified and consequently becomes more apparent.

MARS can model interactions between predictive attributes. The presence of interactions between two predictive attributes indicates that the effect of one predictive attribute on the target attribute is different for different values of the other predictive attribute. Mathematically, this is expressed by multiplying the two predictive attributes. The possibility of expressing in the model the existence of interactions between predictive attributes increases the ability of the model to adjust to the data. However, it also increases the complexity of the model, and the chances of overfitting. The degree of interaction defines the number of attributes that are multiplied between them. There is an hyper-parameter that defines the maximum degree of interaction. The larger its value, the more flexible but also more complex the model can be.

A model induced in the MARS approach has the form:

$$\hat{f}(X) = \hat{\beta}_0 + \sum_{i=1}^{k} \hat{\beta}_i \times B_i(X) \tag{10.1}$$

where B_i is a hinge function, or a product of hinge functions when there are interactions between predictive attributes. A hinge function can have two forms:

- $max\{0; x - c\}$
- $max\{0; c - x\}$.

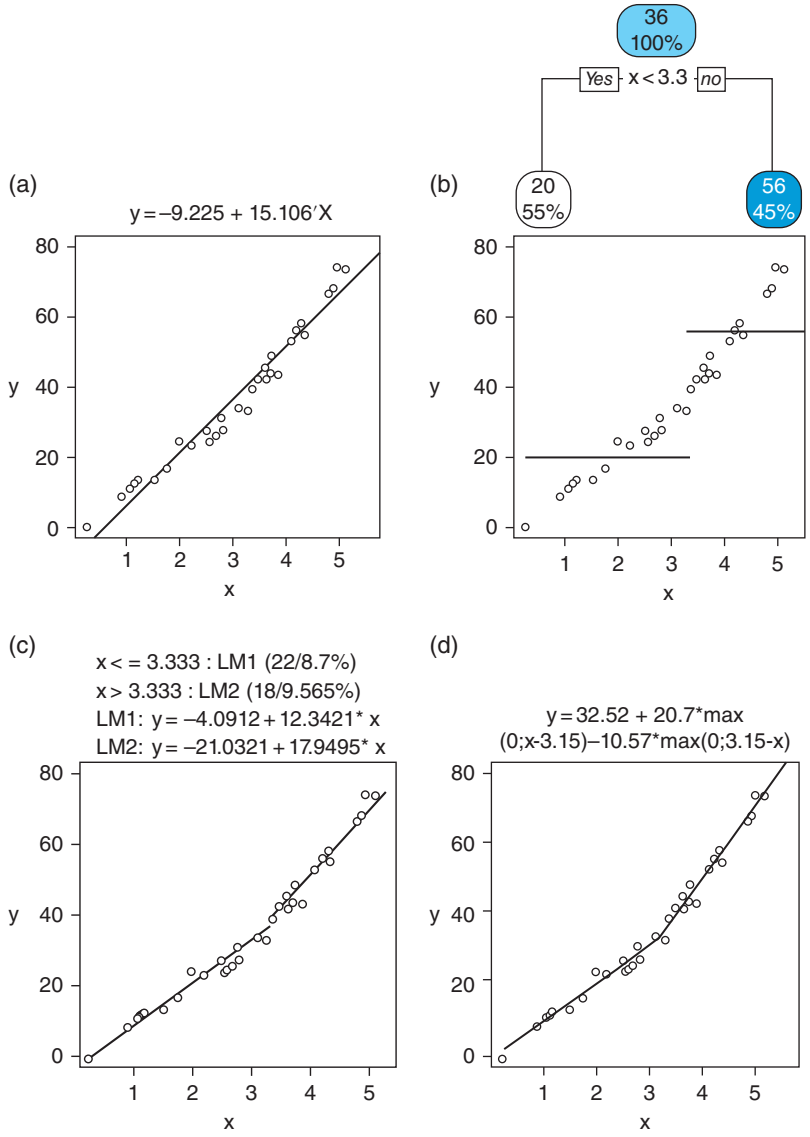


Figure 10.6 Solving an artificial data set with (a) MLR, (b) CART, (c) model trees and (d) MARS.

In these equations, x is an attribute from the vector of predictive attributes X, and c is a constant.

The MARS algorithm has two steps: a forward step, where at each iteration a term is added to the model, and a backward step, where the model is pruned by eliminating some of the terms.

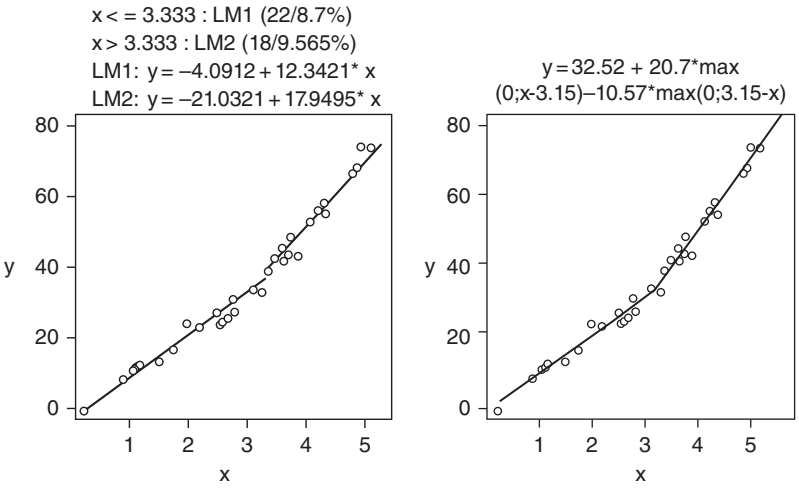


Figure 10.7 Detail of a model tree and MARS using an artificial data set.

Table 10.3 Advantages and disadvantages of MARS.

Advantages	Disadvantages
• Interpretable • Embed attribute selection; • Do not need previous normalization • Less sensitive to the presence of outliers than MLR	• Correlated attributes can have some effect on the model obtained • Outliers are better managed by MARS than by MLR, but still can affect the model

Assessing and evaluating results Like MLR, MARS models are interpretable, but typically are more complex. Figure 10.6 shows an example of the behavior of a MARS model.

Setting the hyper-parameters The main hyper-parameter of MARS is the degree, which that defines the maximum number of interacting hinge functions. This value typically has the default value of 1. Integer values larger than 1 can be tested by increasing the order.

Advantages and disadvantages of MARS The advantages and disadvantages of MARS are set out in Table 10.3.

10.2 Optimization-based Algorithms

This section will describe the main aspects of two very popular learning techniques based on optimization of a given function: artificial neural networks and support vector machines.

10.2.1 Artificial Neural Networks

Artificial neural networks (ANNs) are distributed computer systems based on the nervous system. Our nervous system learns empirically according to the errors we make. For instance, if we put our hand in a very hot place, our nervous system detects it and we feel pain as a result of our mistake. Next time we will be more cautious in putting our hand in a hot place. An ANN tries to simulate this idea mathematically using a network architecture composed by artificial neurons, also known as processing units.

Each artificial neuron has a set of input connections, to receive input values from an input attribute vector or from other neurons. A weight is associated with each input connection, emulating the synapses in the nervous system. A neuron defines its output value by applying an activation function to the weighted sum of its inputs. Different activation functions have been used in the neural network community, ranging from simple linear functions to complex non-linear functions. This output value can be the ANN output or can be sent to other neurons, when the network has more than one neuron layer.

The network weight values are defined by a learning algorithm, which updates the weights according to an objective function. The training optimizes, for each neuron, objective function parameters (neural weights) in order to minimize the prediction error, the difference between the neuron output value produced by the activation function and the desired output value. This is the reason ANN learning algorithms are optimization-based.

The oldest ANN still in use is the perceptron network, proposed by Frank Rosenblatt [32]. Perceptron has a single neuron and is able to learn binary classification tasks. The activation function used by the perceptron network is a bipolar step function, since it has two possible output values: $+1$ when the weighted sum of its input values is above a threshold value, or -1 otherwise. For this reason, this step function is also known as threshold function or signed function. If the lowest value is 0 instead of -1, the function becomes a binary step function.

The weights of a perceptron network are updated by the perceptron learning algorithm. This algorithm modifies the weights in order to reduce the prediction error of the perceptron network, a learning paradigm called error-correcting learning. An error occurs when the output from the perceptron activation function for a given input attribute vector is different from the expected, true, output value.

The algorithm used to train a perceptron network is shown below. In this training algorithm, the training objects are presented one by one to the perceptron network, where x is the predictive attribute vector and y is the true output for that vector in D. Each presentation of all the training examples is called a training cycle.

After training, the network weight values can be used as a classification model to predict the class of new objects in a test set. This is called the test phase. To

predict the class of new objects in a test set using the predictive model induced by the algorithm, a similar, simpler algorithm can be used. This is illustrated in the algorithm below.

In 1962, Albert Novikoff proved the perceptron convergence theorem, which states that if a perceptron network can learn a classification task, its learning algorithm will find the right network weights for this task [33]. It must be noted that the predictive and label output values must be quantitative, so qualitative values in predictive attributes of a data set must be converted to quantitative values.

Algorithm Perceptron training.

1: INPUT D_{train} training set
2: INPUT x_i object in the training set
3: INPUT y_i neuron predicted output value for the object x_i
4: INPUT d_i neuron desired output value for the object x_i
5: INPUT n the number of objects in the training set
6: INPUT m the number of predictive attributes of the objects
7: Define the initial weights with the value 0
8: **repeat**
9: Initialize Errors with the value 0
10: **for all** objects x_i in D_{train} **do**
11: Calculate the neuron output y_i when the neuron receives as input x_i
12: **if** $y_i \neq d_i$ **then**
13: Update the m weights of x_i
14: **until** There are no differences (prediction errors)

Algorithm Perceptron test.

1: INPUT D_{test} test set
2: INPUT x_i object in the test set
3: INPUT y_i neuron predicted output value for the object x_i
4: INPUT n the number of objects in the test set
5: INPUT m the number of predictive attributes of the objects
6: Use the induced perceptron network with weights defined in a training phase
7: **for all** objects x_i in D_{test} **do**
8: Calculate the neuron output y_i when the neuron receives as input x_i
9: Label x_i with the class associated with the y_i value

However, the perceptron learning algorithm is limited to linearly separable classification tasks. For non-linearly separable problems, a network with more than one layer is necessary. However, when the perceptron was proposed, a

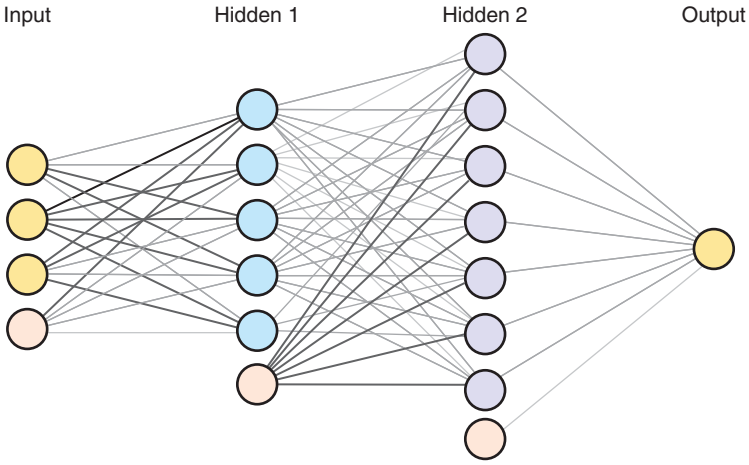


Figure 10.8 MLP neural network with two hidden layers.

learning algorithm to train such networks was unknown. A few years later, a learning algorithm to train a multi-layer perceptron (MLP), known as a backpropagation algorithm, was independently proposed by different groups [34–37]. In a MLP network, the last layer of neurons is named the output layer and the others, the hidden layers. Figure 10.8 illustrates a typical MLP network.

10.2.1.1 Backpropagation

Backpropagation uses gradient descent to update the MLP network weights. Gradient descent identifies the direction of weight update that leads to the largest reduction in the errors made by an MLP network. A commonly used analogy is: suppose you are standing in a particular position in a rugged landscape, with valleys and hills, and you want to move to the landscape's lowest position. Suppose further that your eyes are closed. How can you move to the lowest point? A simple and intuitive approach would be to try several possible directions with one of your feet and move in the direction that makes you go to the lowest nearby position.

It must be pointed out that if an MLP network uses a linear activation function in the neurons of the hidden layers, it can be shown by matrix manipulations that this network would be equivalent to several single-layer perceptron networks. Thus, MLP networks must use non-linear activation functions in their hidden layers. By using non-linear activation functions, MLP networks trained by the backpropagation algorithm can solve non-linearly separable problems. They use the gradient descent of the activation functions, which needs therefore to be continuously differentiable. The use of gradient

descent makes the learning faster and more efficient. The algorithm, as shown below, has the pseudocode of the backpropagation algorithm, where X is the predictive attribute vector, T is a vector with the true output of each output neuron for the vector X from D, and Y is the vector of the y value produced by each output neuron for X.

The backpropagation algorithm, shown below, works as follows. When an attribute vector is presented to an MLP network, each neuron in the first hidden layer applies an activation function to its weighted input values. The value of this function is used as input by the neurons in the next layer. The same process continues, layer by layer, until the output layer is reached. The activation function values from the neurons in the output layer are the network label attribute values. These values are compared with the true values and if they are very different the weights of the network are updated.

The updating starts in the output neurons where the predicted value is different from the true value. The weight updated goes from the last layer to the first hidden layer, in a backward direction. The error for each neuron in each hidden layer is estimated using the errors in the neurons of the next layer weighted by the weight values of the current layer neuron and the neurons in the next layer that presented an error. The amount of the update value depends on a hyper-parameter called the learning rate. It must be larger than zero. The larger it is, the faster the learning process is, but the higher the risk of getting a local optimum instead of the desired global optimum.

The classification model that is learned depends on the number of layers and the number of nodes per layer. The larger these numbers are, the more complex the models that can be induced are. When a MLP network is trained by backpropagation, one or two hidden layers are used. Each neuron in the first layer of an MLP network learns the position and orientation of a hyperplane, dividing the input space. In the second hidden layer, each neuron combines a subset of the hyperplanes found in the previous layer, producing a convex region in the input space, encompassing some of the training objects. The nodes in the next layer, usually an output layer, combine these convex regions, producing regions of arbitrary shapes.

During the MLP training, the hyperplanes' position and orientation, and the position and shape of the convex and non-convex regions keep changing, as illustrated, for two predictive attributes and two classes, by Figure 10.9. Figure 10.10 shows how the layers and nodes of an MLP network divide the input space for two predictive attributes, for the same data set as previously.

The training for MLP networks is a search for the model that best approximates the true, unknown, model that generated the training data set. This search is guided by the network predictive performance. In doing so, the training can produce an overfitted model, which is a model that pays too much attention to details, instead of the underlying patterns in the training data. This leads to complicated, overfitted models. When overfitting occurs, instead of

Algorithm Backpropagation training for MLP networks.

INPUT D_{train} training set
INPUT Y_{true} true output vector
INPUT Y_{pred} predicted output vector
INPUT x_i object in the training set
INPUT y_i neuron predicted output value for the object x_i
INPUT d_i neuron desired output value for the object x_i
INPUT n the number of objects in the training set
Initialize Errors with random values in the range [−0.5, +0.5]
repeat
    Initialize $\text{Error}_{Total} = 0$
    for all objects x_i in D_{train} **do**
        for all network layer, starting in the first hidden layer, forward **do**
            for all neuron in the current layer **do**
                Calculate the neuron output y_i when the neuron receives as input x_i or the output from the neurons in the previous layer
        $\text{Error}_{Partial} = Y_{true} - Y_{pred}$
        $\text{Error}_{Total} = \text{Error}_{Total} + \text{Error}_{Partial}$
        for all network layer, starting in the output layer, backward **do**
            for all neuron in the current layer **do**
                if error > 0 **then**
                    Update weight values
until Predictive performance is acceptable

learning from the data, the model memorizes the data. As consequence, it has low generalization ability. A model generalizes well when it correctly classifies data not present in the training set.

There are some alternatives that will reduce the occurrence of overfitting. One of them, termed "early stops", separates part of the training data set to validate the predictive performance at several time stamps. This data subset, called a validation set, is not used by the backpropagation algorithm directly in the MLP network training, but instead is used to evaluate, from time to time, the network predictive performance. At first, the network predictive performance for training data and validation data keeps improving. When the predictive performance for the validation data starts to decrease, overfitting is starting and the training process is stopped.

There are several training algorithms for MLP networks, allowing the user to select the most suitable for a given predictive task. Because they are very flexible in dividing the input space, MLP networks have exhibited high predictive performance in various classification tasks. Unlike decision tree induction algorithms, which can only use hyperplanes parallel to the axes representing

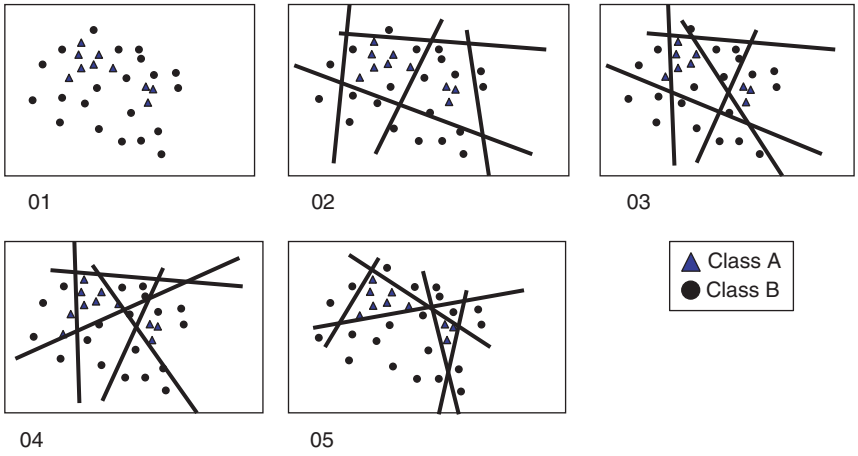


Figure 10.9 How the MLP training affects the orientation and position of separating hyperplanes and the orientation and position of convex and non-convex separating regions.

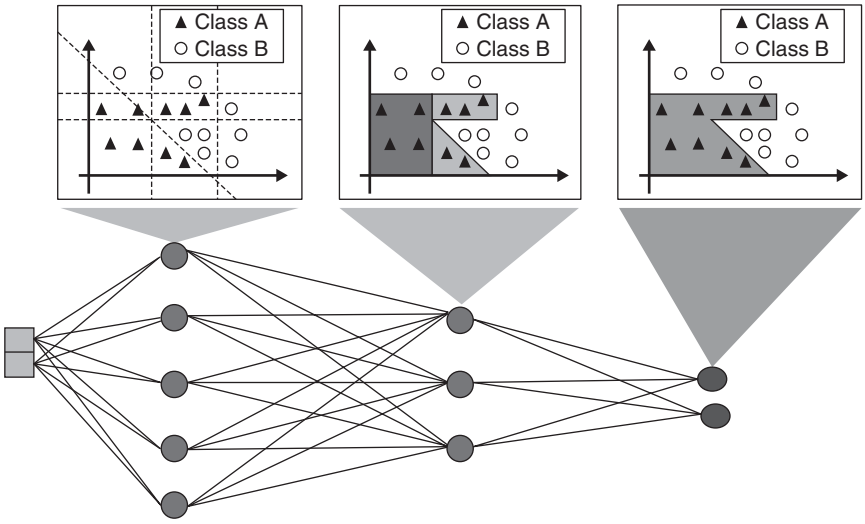


Figure 10.10 Orientation and position of separating hyperplanes, and orientation and position of convex and non-convex separating regions of a trained MLP network.

predictive attributes, MLP networks can produce separation hyperplanes with any orientation.

Figure 10.11 illustrates separating regions that could be created by a MLP network trained with backpropagation and a decision tree induction algorithm for a data set with two predictive attributes and three classes.

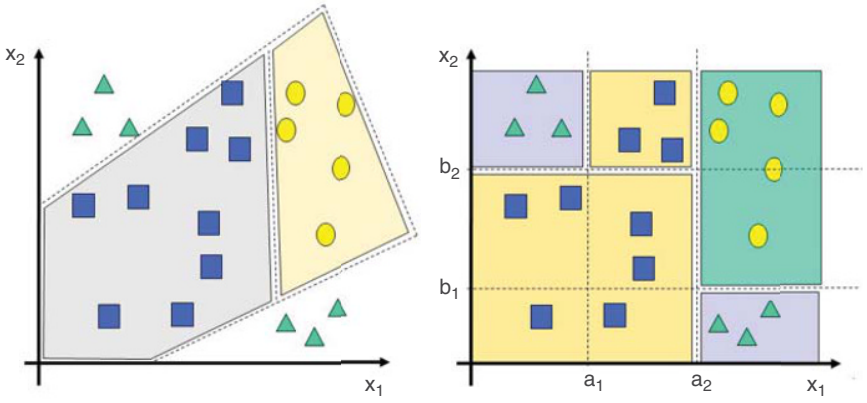


Figure 10.11 Separation regions created by (left) an MLP network trained with backpropagation and (right) a decision tree induction algorithm for the same data set.

As in the perceptron network, after the training of an MLP network, we have a classification model, which is represented by the weights learned. The induced classification model can easily be used to predict the class of new objects from a test set. The algorithm below shows how the test phase works for an MLP network. The test procedure does not depend on the learning algorithm used to train the MLP network.

Algorithm Classification of new objects using an MLP neural network.

1: INPUT D_{test} test set
2: INPUT x_i object in the test set
3: INPUT y_i neuron predicted output value for the object x_i
4: INPUT n the number of objects in the test set
5: INPUT m the number of predictive attributes of the objects
6: Use the induced MLP network with weights defined in a training phase
7: **for all** objects x_i in D_{test} **do**
8: **for all** network layer, starting in the first hidden layer, forward **do**
9: **for all** neuron in the current layer **do**
10: Calculate the neuron output y_i when the neuron receives as input x_i or the output from the neurons in the previous layer
11: Label x_i with the class associated with the y_i values predicted by the neurons in the output layer

Several other ANNs have been proposed, mainly for classification tasks. But there are also ANN techniques for clustering, regression and time-series analysis. There is also currently great interest in "deep learning", a set of learning techniques for neural networks of several layers.

Assessing and evaluating results The knowledge acquired by a neural network after its training is represented by the weight values associated with the network connections. A learned concept can be represented by several weight values and a weight value can represent part of several learned concepts. This distributed knowledge representation makes the interpretation of the model very difficult. The predictive performance of the induced model can be estimated by using a suitable predictive performance measure.

Setting the hyper-parameters The main hyper-parameters of ANN and a training algorithm like backpropagation with momentum are:

- *the number of layers:* the larger the number of layers the more complex are the features extracted by the ANN;
- *the number of nodes:* the larger the number of nodes, the higher the chances of overfitting.
- *the learning rate:* this defines how large the weight updates are; the value must be larger than zero and the larger it is the faster the learning process is but also the larger the risk of stopping at a local optimum;
- *the momentum term:* this uses the previous weight updates to improve the search;
- *the number of epochs:* how many times the training set will be presented;
- *the activation function:* this defines the output from each neuron in the ANN given the the input received and the neuron weights, and also defines the solution landscape; different activation functions can be used for classification and regression tasks, including:
  - linear function
  - step function
  - sigmoid function
  - sigmoid or logistic function
  - hyperbolic tangent function
  - rectified linear function.

Advantages and disadvantages of ANNs The advantages and disadvantages of ANNs are set out in Table 10.4.

Most classification tasks solved by MLP networks usually have 1 or 2 hidden layers. A single layer with a large enough number of neurons can, with arbitrary accuracy, approximate any multivariate continuous function [38]. It can be easily shown that, with two hidden layers, given a representative training data set, a large enough number of neurons, and the proper activation function in each layer, any classification function can be learned. However, in contrast to the perceptron training algorithm, where if a function can be learned, it *will* be learned, there is no assurance with backpropagation that a function that can be learned by a two hidden layer MLP network *will* be learned. The good news is that the

Table 10.4 Advantages and disadvantages of ANNs.

Advantages	Disadvantages
• Good performance in many real-life problems • Easily applied to multiclass and multilabel classification tasks • Model the structure and functioning of the nervous system • Very robust in the presence of noise	• Models difficult to interpret • Training algorithms are stochastic: different models usually obtained in different runs of the algorithm • Training usually has high computational cost • Lack of strong mathematical foundation

use of an additional layer can increase the chance that the desired classification function will be learned. Assuming that each layer transform the previous data representation, a large number of layers would provide a composition of several transformations, allowing the learning of very complex classification functions.

The bad news is that, although ANNs with much more than two layers have been investigated for several years, until recently there was neither an efficient learning algorithm nor the necessary computational resources for their training. Given the available resources, MLP training algorithms, like backpropagation, did not perform well for ANNs of many layers. The main reason is that backpropagation updates the weights of each node using the prediction error of the node. The prediction error is easily calculated for the neurons in the output layer, since the predicted value and the expected value are available. For the neurons in the last hidden layer backwards, the expected value is not known, and it is therefore estimated from the errors of the neurons in the next layer it is connected to. Thus, as the weight update performed by backpropagation goes from the output layer to previous layers, the information regarding the classification error for the current object is increasingly deteriorated.

10.2.1.2 Deep Networks and Deep Learning Algorithms

With current computational resources, like the introduction of graphics processing units (GPUs), and access to very large training sets, this deterioration problem has became less critical. Deep learning (DL) algorithms, which are learning algorithms able to train MLP networks with at least two hidden layers [39], known as deep networks, have become widespread. MLP networks with a small number of hidden layers, usually just one, are called shallow MLP networks. In the next section, some of the main deep MLP network architectures and DL algorithms will be briefly described.

In recent years, the very good results obtained using deep networks in several applications has attracted the strong interest of both research institutions and companies. According to experimental results reported in the data analytics literature, deep networks trained by DL algorithms have exhibited predictive

performance superior to other ML algorithms in several application domains, including:

- image recognition
- natural language processing
- speech recognition
- natural language processing
- games
- drug design.

It should be noted that the use of deep networks and their training by DL algorithms is not new. They were originally proposed and applied to real problems some decades ago. For example, at the end of the 1970s, Neocognitron, a hierarchical multi-layer ANN in which several neuron layers simulated the virtual cortex was successfully applied to the problem of handwritten character recognition [40].

In many predictive tasks, features need to be extracted from the raw data before a learning algorithm is used. These features, which are used as predictive attributes, should represent aspects of the data that are relevant to the predictive tasks and, at the same time, ignore irrelevant aspects. The extraction of these features by human experts usually requires domain knowledge and engineering skills and results in high costs.

One of the main reasons for the success of DL is the division of the network into two stages. The first stage encompasses the first layers and is often pre-trained, usually with an unsupervised approach. The subsequent training of the layers in the second second stage is usually supervised.

The layers in the first stage are trained to extract relevant features from a raw data set. One of the main contributions of DL with pre-training is the automatic extraction of relevant features by general-purpose learning algorithms [41]. Thus, pre-training has been frequently used to extract features for complex classification tasks, where the extraction of features by a domain expert would not necessarily lead to features that would support the induction of efficient classification models.

The first network layer extracts simple features from the raw data. Successive layers use the features extracted from the previous layer to extract more sophisticated, complex features. Thus, the training process creates, from the first to the last layer, increasingly complex data representation levels. Each representation level can be seen as a module or layer that performs simple and non-linear processing. Each module transforms the representation extracted by the previous module.

The current popularity of DL techniques has resulted in the proposal of several different architectures and learning algorithms. A simple approach for DL is to train MLP networks of several layers using backpropagation. Until recently, in the training of MLP experiments with backpropagation algorithms, the most frequently used activation functions were the sigmoid or hyperbolic tangent.

To improve the learning performance when several layers are used, a new activation function, the rectified linear unit (ReLU) activation function has been used. When the ReLU function is applied to a value, it returns 0 if the value is negative and the value itself otherwise.

The use of the ReLU function makes the updating of the network weights faster: the calculus of the gradient descent, used for weight update for the ReLU function, is faster than for other popular non-linear activation functions. Thus, ReLU speeds up learning, which is particularly welcome for MLP networks with many layers. ReLU has a strong biological motivation and is mathematically simpler.

One of the most popular pre-trained DL techniques is the use of convolutional neural networks (CNNs), also known as ConvNets. CNNs use unsupervised learning to extract features – predictive attributes – from raw data. The first network layer extracts simple features from the raw data. Successive layers use the features extracted from the previous layer to extract more sophisticated, complex features. Thus, the pre-training creates increasingly complex representation levels. Each representation level can be seen as module or layer that performs simple and non-linear processing. Each module transforms the representation extracted by the previous module.

Much of the fame generated by DL algorithms in recent years is due to the impressive performance of CNNs when applied to image recognition. CNNs mimic how an image is processed in the brain, starting with the extraction of very simple features, like lines and curves, and progressively extracting features of increasing complexity. A CNN involves different types of layers organized into two processing stages. The first stage has a pair of layers: a convolutional layer, which extracts feature maps from the input using filters, followed by a pooling layer, which keeps only the most relevant information from the feature maps. As a result, the input data representation becomes successively more abstract. As more modules are used, more complex representations are found. The second stage is usually a conventional MLP network, with its activation layers. The features extracted by the first stage layers can be used as predictive attributes by supervised learning techniques.

Several other DL techniques can be found in the literature, including:

- auto-decoder networks
- deep belief networks
- restricted Boltzmann machines.

In addition to the negative aspects found in other ANNs, DL techniques need a very large number of training objects, since many different variations of the objects are necessary to extract relevant features from the training data set.

Assessing and evaluating results The information provided by the models learned by ANNs is the learned weight values, as shown in Figure 10.8, just as was the case for the architectures used for DL. This information is hard to

interpret. The predicted values can be assessed and evaluated using a suitable performance measure.

Setting the hyper-parameters The hyper-parameters for a deep network and a DL algorithm are very similar to those used to define an MLP network architecture and how a learning algorithm such as backpropagation will work. There are other hyper-parameters to select, depending on the deep network used. For CNN networks, the following hyper-parameters can be tuned:

- the number of CNN (convolutional and pooling) layers;
- the number of fully connected layers;
- the number of neurons in the fully connected layers;
- the filter size in the convolutional layers;
- the max-pool size in the pooling layers;
- the training algorithm;
- the learning rate: this defines how large the weight updates are; the value must be larger than zero and the larger it is, the faster the learning process, but the higher the risk of stopping at a local optimum;
- the momentum term, which uses the previous weight updates to improve the search;
- the number of epochs: how many times the training set will be presented;
- the activation function, which defines the output from each neuron in the ANN and, for the output layer, the ANN output. It also defines the solution landscape. Different activation functions can be used for classification and regression tasks, including:
  - linear function
  - step function
  - sigmoid function
  - sigmoid or logistic function
  - hyperbolic tangent function
  - rectified linear function.

Advantages and disadvantages of deep learning networks The advantages and disadvantages of DL are set out in Table 10.5.

MLP networks, be they shallow or deep, have two drawbacks. The first is the choice of good hyper-parameter values. Usually the selection of these values is carried out by trial and error or using an optimization metaheuristic, both of which are time consuming. The second negative aspect is the poor interpretability of the induced models.

10.2.2 Support Vector Machines

Two of the deficiencies of ANN learning algorithms – generation of different models for different runs and a lack of a mathematical foundation – are

Table 10.5 Advantages and disadvantages of DL.

Advantages	Disadvantages
• Exhibits most of the positive aspects of ANNs • deep network models have exhibited very good performance in many real problems, superior to several state-of-the-art ML algorithms • Very robust in the presence of noise • Able to extract relevant features from raw data • Similar to several features found in the nervous system.	• Exhibits most of the deficiencies of ANNs • DL needs of a large number of training examples • Lack of strong mathematical foundation

overcome by support vector machines (SVMs), also known as support vector networks [42, 43]. Based on statistical learning theory, SVMs have a strong mathematical foundation.

SVMs were originally designed for regression and binary classification tasks. When applied to binary classification tasks, the SVM learning algorithm looks for classification models that combine high predictive accuracy and low model complexity. To this end, training objects from different classes are selected to be support vectors. These vectors define a decision border able to maximize the separation margin between the border and the objects of the two classes. For this reason, they are also known as "large margin classifiers". The margin maximization reduces the number of possible models and increases the model generalization ability, and thus the occurrence of overfitting. Figure 10.12 illustrates the separation margins defined by support vectors, as selected by a SVM.

The other famous binary classifier, the perceptron ANN, is not concerned with the place and direction of the decision border, provided that the border separates the objects of the two classes.

Figure 10.13 illustrates decision borders that are found, for the same training data set, by the perceptron and SVM learning algorithms.

To increase the size of the separation margin, some objects can be allowed inside the separation margin. Figure 10.14 shows how the SVM margins can be increased by allowing some objects inside it. These objects are referred to as "slack variables".

Like perceptron networks, SVMs, in their original design, can only deal with linearly separable tasks. However, SVM can use kernel functions to transform non-linearly separable data, from its original space, into a higher dimensional space, where it becomes linearly separable. Figure 10.15 illustrates how this transformation can occur.

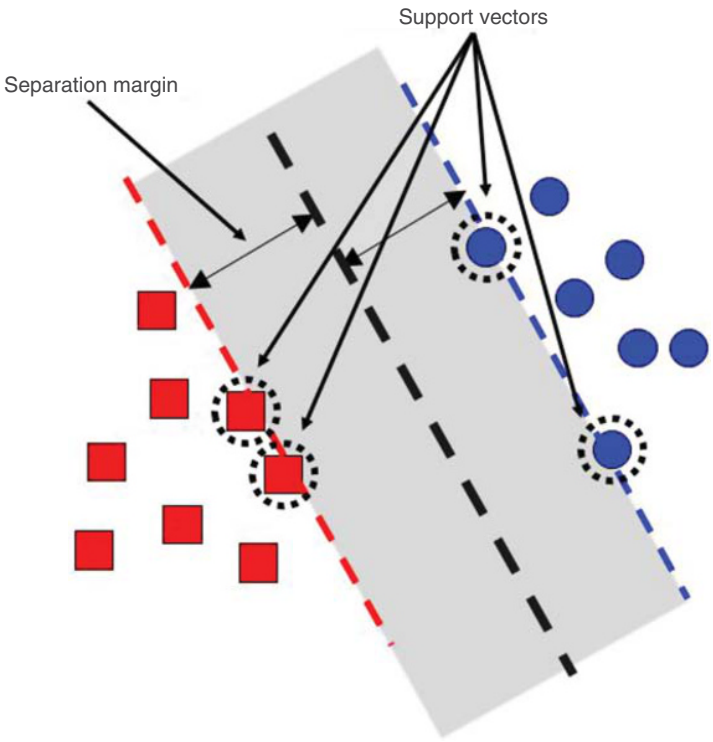


Figure 10.12 Large margin defined by SVM support vectors.

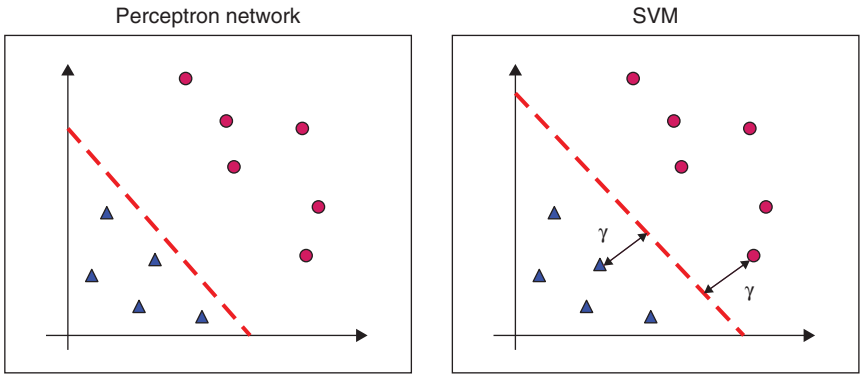


Figure 10.13 Decision borders found by (left) perceptron and (right) SVM learning algorithms.

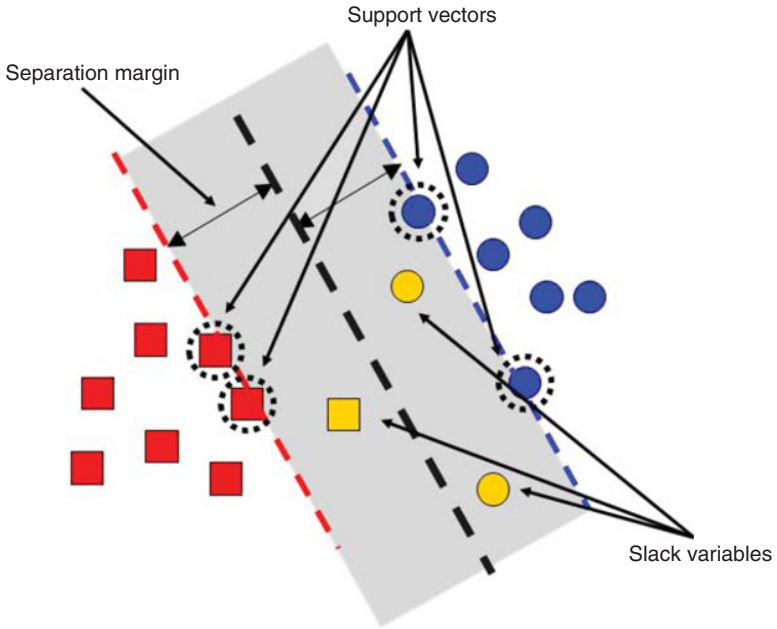


Figure 10.14 Increase of separation margin by allowing some objects to be placed inside the separation margin.

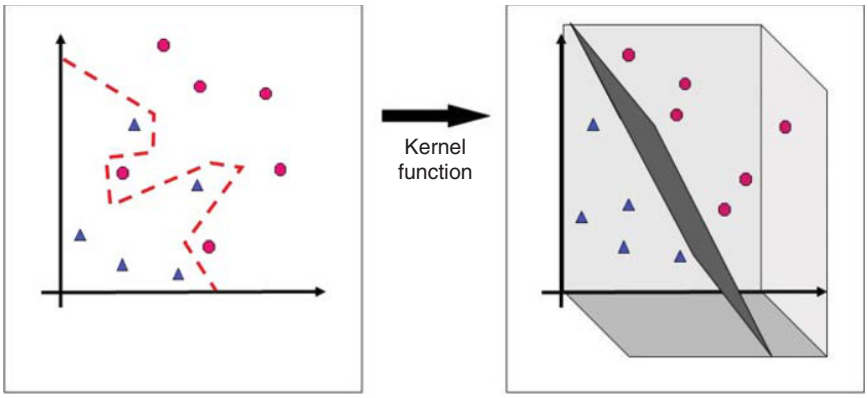


Figure 10.15 Example of the use of a kernel function to transform a non-linearly separable classification task in a two-dimensional space to a linearly separable classification task in a three-dimensional space.

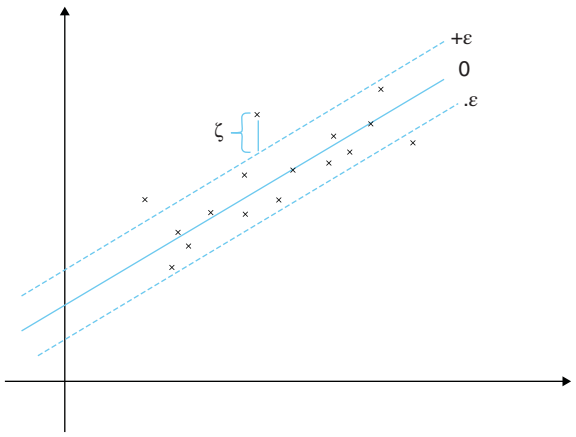


Figure 10.16 The soft-margin in a regression problem and how ζ is calculated.

As previously mentioned, SVMs are also limited to binary classification tasks. However, as will be seen in Chapter 11, some strategies have been proposed to allow the application of SVMs to multi-class classification tasks.

10.2.2.1 SVM for Regression

The main difference between the classification and the regression SVM problem is on the definition of the error function and, as consequence of this, the way the optimization problem is formulated as a linear programming problem. In both classification and regression problems, a soft margin error function is used. However, these functions have different definitions in the regression and classification formulations.

In both problems, ζ is the distance of a slack instance to the soft margin. In classification, the optimization problem is the maximization of the separation margin, as shown in Figure 10.14. However, in regression, the optimization problem is the minimization of the ϵ margin, as shown in Figure 10.16.

Assessing and evaluating results The information about the learned model consists of the support vectors: the objects that define the soft margin. Such information is hard to interpret. The predicted values can be assessed and evaluated as usual with a suitable predictive performance measure.

Setting the hyper-parameters The main hyper-parameters of SVM are:

- C: the cost of constraint violation for the classification setting; the larger its value, the more complex the margin will be, typically with more support vectors. Values of C that are too large can promote overfitting.

Table 10.6 Advantages and disadvantages of SVM.

Advantages	Disadvantages
• SVM models have strong theoretical foundations • SVM models have exhibited good predictive performance in many problems	• Very sensitive to hyper-parameter values • Computational cost depends on the number of support vectors of the model, which can be large in some problems • Original technique can only deal with binary classification tasks

- ν: similar to the C hyper-parameter but ranging between 0 and 1. For this reason, it is easier to set the value of ν than the value of C. Some implementations of SVM use the ν hyper-parameter instead of C.
- ϵ: controls the width of the ϵ-insensitive zone for the regression setting, as shown in Figure 10.16. Its value can affect the number of support vectors. Typically, bigger ϵ values produce models with fewer support vectors, resulting in flatter estimates.
- Kernel: the kernel should be chosen together with their own hyper-parameters. Some of the most commonly used kernels and their respective hyper-parameters are:
  - *linear:* defines a linear margin and has no specific hyper-parameters;
  - *radial basis function:* also known as the Gaussian kernel: the probability distribution function of the Gaussian (normal) distribution (Section 2.2.4). It has γ as its hyper-parameter.

Advantages and disadvantages of SVM The Advantages and disadvantages of SVM are set out in Table 10.6.

10.3 Final Remarks

Despite the increasing success of DTIAs, ANNs and SVMs, and the growing interest in DL, there are several aspects of these approaches that need to be addressed.

Most current applications assume that all relevant data are available, which is generally not true, since in almost all applications, new data are being continuously generated in data streams. Data stream mining poses several challenges to current techniques. Learning algorithms for data stream mining need to have the capacity to learn quickly, to learn after looking at each training example just once and to automatically adapt models to occurrence changes in the class

profiles (concept drift detection) and to the appearance of new classes (novelty detection). As a result, predictive models must also be able to forget outdated knowledge.

Additionally, it is not enough for the learning techniques to induce predictive models with good predictive performance. In many applications, it must be possible to interpret the knowledge learned by the induced models. The large number of techniques available makes the choice of the most suitable technique for a new task a predictive problem in itself. Additionally, particularly in applications that can result in human harm, it must be possible to mathematically prove that the models are robust. For many common learning techniques, there is a lack of a consistent mathematical foundation, preventing assessment of the robustness.

10.4 Exercises

1 How can a decision tree be transformed into a set of classification rules?

2 Suggest a new criterion to split the training objects reaching an internal node in a decision tree.

3 Give two advantages and two disadvantages of using decision tree induction algorithms instead of MLP neural networks.

4 How can a DTIA find a decision border similar to the decision border found by a perceptron network?

5 Is it possible to train MLP neural networks with the perceptron learning algorithm? If so, how is this done? If not, why not?

6 Does the idea of an MLP network where all activation functions are linear make sense? Why?

7 What is gradient descent and why does the backpropagation algorithm use it?

8 What happens to the classification error estimate of the neurons in an MLP network as the number of layers is increased?

9 What is the role of the first layers in a DL technique?

10 Given the social network dataset from Table 9.4, using the first ten objects as a training set and the last ten objects as a test subset, perform the following activities:

a) Use the C4.5 (or J48) learning algorithm, with default values for the hyper-parameters, to predict the class label of the test subset objects.
b) Perform the same experiments using an MLP trained with back-propagation, varying the number of hidden layers and the number of neurons per hidden layer. For the other hyper-parameters, use the default values.
c) Repeat the experiments using SVM with at least three different kernel functions. For the other hyper-parameters, use the default values.
d) Compare the results from the three sets of experiments.

11

Advanced Predictive Topics

It is now time to talk about somewhat more advanced subjects in predictive analytics. Nevertheless these have been chosen for their usefulness in many real-world problems.

11.1 Ensemble Learning

Suppose you need to select a restaurant for a party with friends. You can choose the restaurant based on your previous experience of local restaurants. Your decision therefore will be largely influenced by the particular experiences you had. But suppose that instead of relying only on your own experience, you also asked some of your friends for their opinion. You can take a vote, with the restaurant recommended by the largest number of people, among you and your friends, being selected. By considering the previous experiences of different people, you can increase the chance of making a good decision. This same idea is adopted by ensemble methods.

As seen in Chapter 9, you can select a classification technique to apply to a data set based on the predictive performance of classification models induced by these techniques for part of the data set. But why restrict ourselves to just one option? When you apply different classification techniques, or even the same technique to a data set, more than one classification model is usually induced. Why not, instead of using just one classification model, combine the class predictions made by more than one classification model?

Several studies point out that the predictive performance in classification tasks can be improved by combining the predictions from multiple classifiers: an ensemble of classifiers. The individual classifiers whose predictions will be combined will be referred to here as the "base" classifiers. Each base classifier can be induced using the same, original, training set, or parts of the original training set.

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

An ensemble can be homogeneous, when the base classification models are induced by the same technique, or heterogeneous, when different techniques are used to induce the base classifiers.

Two important requirements in developing ensembles with good predictive performance are the predictive performance and the predictive diversity of the base classifiers. The diversity requirement means that the base classifiers must be independent, ideally making errors in different, complementary, areas of the input space. The predictive performance means that the base classifiers must have a predictive performance better than a majority class classifier.

There are three different approaches to inducing base classifiers in an ensemble:

- parallel
- sequential
- hierarchical.

The first approach, parallel combination, illustrated in Figure 11.1 is the most common. In this approach, each base classifier is induced using objects from the original training set. Each classifier can be induced using the whole training set, a sample of the training set, or the whole training set but using only a subset of the predictive attributes. This approach tries to explore the similarities and differences in the predictions made by the base classifiers.

In the sequential or serial approach, the induction of a base classifier uses information from the base classifiers previously induced in some way, for instance, by combining the predictions from the previously induced base classifier with the predictive attribute values. This approach, shown in Figure 11.2, can be for hierarchical and/or multilabel classification tasks. Another possible use of this approach is when the first classifier is a simple technique that selects a subset of the classes, leaving to the following classifier the task of selecting a single class from that subset.

Finally, the two previous approaches can be combined, using both parallel and sequential approaches, as illustrated in Figure 11.3.

The classes predicted by each base classifier are combined to give the class predicted by the ensemble. The combination of the classes predicted by the base classifiers is usually arrived at by voting, weighted voting or stacking approaches.

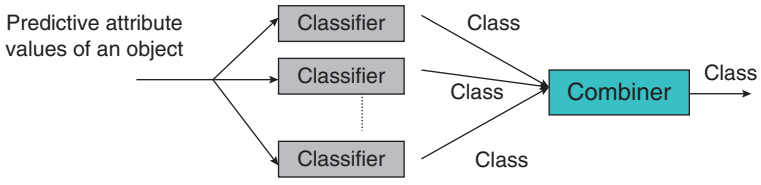


Figure 11.1 Ensemble with a parallel combination of classifiers.

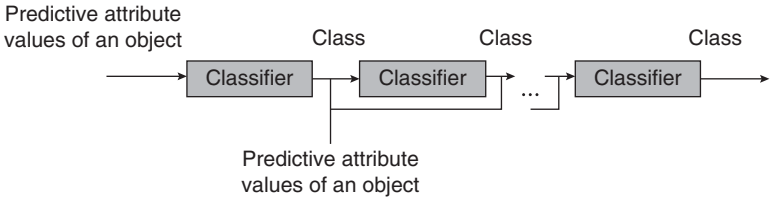


Figure 11.2 Ensemble with a sequential combination of classifiers.

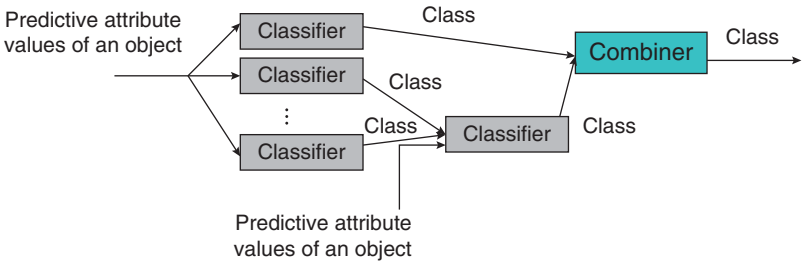


Figure 11.3 Ensemble combining parallel and sequential approaches.

In combination by voting, the class predicted by the largest number of classifiers is the class predicted by the ensemble. In combination by weighted voting, the vote of each classifier is associated with a weight, representing how much the prediction made by a base classifier should be taken into account for the ensemble predicted class. In combination by stacking, a classification technique induces a classifier, using as predictive attributes the class predictions from the base classifiers.

Three of the most common ensemble approaches for classification are:

- bagging: a parallel approach
- random forests: also a parallel approach
- AdaBoost: a sequential approach.

11.1.1 Bagging

In bagging, each base classifier is induced using a sample of the training set. The samples, defined using the bootstrap approach, have the same number of objects as present in the training set. Bagging combines the predictions from the base classifiers using voting. Bagging is suitable for unstable classification techniques, which are those whose predictive performance is affected by changes in the training set composition. Decision trees and artificial neural networks are unstable predictive techniques. Bagging is robust to overfitting when the training set is noisy. The number of generated models is a hyper-parameter for the bagging technique. The larger the number of induced

Table 11.1 Advantages and disadvantages of bagging.

Advantages	Disadvantages
• Typically improves the predictive performance of the base learner since the base learner is an unstable predictor • Almost hyper-parameter free, because the number of trees to be generated does not need to be tuned and decision trees – the most common base learner used – is hyper-parameter free	• Due to the bootstrap sampling, bagging has randomization, but variability of results can be minimized by suitable choice of number of generated models • Computationally more expensive than single model, but can be parallelized

models, the lower is the variance of the prediction. Obviously, increasing the number of base models also increases the computational cost.

Bagging can also be used for regression. The ensemble prediction is obtained by averaging the predictions of the models generated.

Assessing and evaluating results The main results of bagging are the predictions and the generated base models.

Setting the hyper-parameters Bagging has two main hyper-parameters: the number of base models that are to be generated, and the learner to generate these models. Some bagging approaches use a decision tree method as the base learner but some other implementations allow the user to choose the intended base learner. Decision trees and artificial neural networks are the most common base learners due to their instability. The larger the number of generated models the better: 100 models is considered a good option.

Advantages and disadvantages of bagging The advantages and disadvantages of bagging are set out in Table 11.1.

11.1.2 Random Forests

Random forests, as the name suggest, were created to combine several decision trees. As in bagging, each decision tree is created using a different bootstrap sample. However, in contrast to bagging, at each node of the tree, instead of choosing the split rule from all predictive attributes, only a predefined number of attributes, randomly selected at each node, is used. Figure 11.4 illustrates an example of a random forest.

Random forests is a good choice when the number of predictive attributes in a data set is large. They usually give good predictive performance and have some level of interpretability. However, they can have high computational cost.

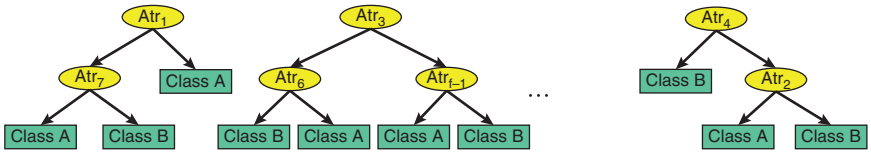


Figure 11.4 Example of a random forest predictive model.

Table 11.2 Advantages and disadvantages of random forests.

Advantages	Disadvantages
• Very good predictive performance in many problems • Easy to define/tune hyper-parameters	• Computationally expensive since the number of recommended trees is large, but, like bagging, can be parallelized • Randomization, but this can be minimized using the recommended number of trees

Random forests use decision trees as the base learner. Like bagging, they can be used for both classification or regression. The variability of the results will be lower, as the number of generated trees increases.

Assessing and evaluating results The main results of a random forest are the predictions and some statistics about the importance of the predictive attributes.

Setting the hyper-parameters The two main hyper-parameters of random forests are the number of base models to generate and the the number of attributes to randomly select at each node. The recommended number of trees is 1000. However, in order to obtain more reliable statistics for the attribute importance, 5000 trees are recommended. The number of attributes to select at each split-node is the main hyper-parameter to be tuned. Its optimum value is problem dependent. As a rule of thumb, the square root of the number of predictive attributes can be used.

Advantages and disadvantages of random forests The advantages and disadvantages of random forests are set out in Table 11.2.

11.1.3 AdaBoost

AdaBoost is one of the most representative boosting set of methods. In AdaBoost, at each training iteration, a base classifier is induced using the training set and each object from the training set is weighted according to how well the base classifier predicted its true class. Thus the more difficult the class

Table 11.3 Advantages and disadvantages of Adaboost.

Advantages	Disadvantages
• Good predictive performance in several problems • Easy to define/tune its hyper-parameter	• Computationally expensive since the number of generated models depends on the number of iterations and there is no parallelization possibile because it is a sequential algorithm • Hard to interpret

of an object is to predict, the larger the weight associated with the object is. The weight of an object defines its probability of being selected for the training of the next base classifier. While the bagging and random forest approaches are parallel ensemble methods, Adaboost is sequential.

Adaboost is suitable when the base classifiers are weak; that is, when their predictive performance is only slightly better than a random guess.

Adaboost is a classification technique. Several studies have adapted it for regression. One of the most popular is gradient boosting, which can be used for both classification and regression. XGBoost, an acronym for "extreme gradient boosting" [44] has a strong statistical basis. Although not new – it was proposed in 2000 – it is one of the most popular techniques in data mining and machine learning.

Assessing and evaluating results The main result of Adaboost is its predictions.

Setting the hyper-parameters The main hyper-parameter of Adaboost is the number of iterations to generate. Freund and Schapire, the authors of the seminal paper on AdaBoost, use 100 iterations in their experiments [45].

Advantages and disadvantages of Adaboost The advantages and disadvantages of Adaboost are set out in Table 11.3.

11.2 Algorithm Bias

To be able to learn, ML algorithms need to make prior assumptions. These assumptions are termed "bias". The bias makes a learning algorithm give preference to a particular set of hypotheses over others. The main biases associated with an ML algorithm are the search bias and the representation bias.

The search bias, also known as the preference bias, defines the order the possible hypotheses are searched in the hypothesis space. For example, the search can prefer smaller, simpler hypotheses over more complex ones.

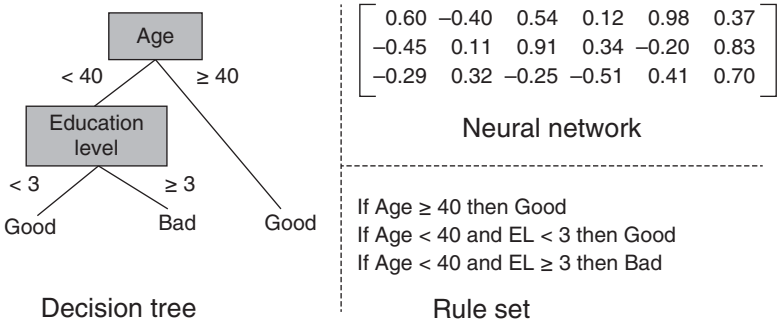
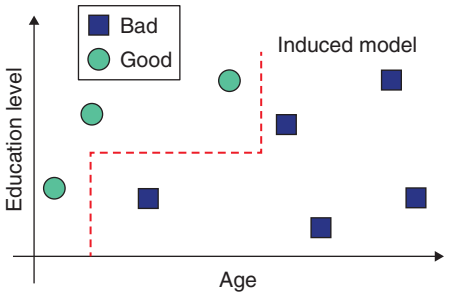


Figure 11.5 Representational bias for three different classification algorithms.

Figure 11.6 Classification model induced by a decision tree algoritm on a data set with two predictive attributes.



The representation bias defines how the hypotheses are represented, constraining the ones that can be found in the search space. For example, the hypothesis space can have only linear functions. Figure 11.5 illustrates a possible hypothesis space for three different classification algorithms.

Several other valid classification models could be induced by the same classification algorithm or other classification algorithms, depending on the representation bias used. Figure 11.6 presents a classification tree. This kind of model has a representation bias different from the algorithm used to induce the linear classifier, as presented in Figure 9.5.

The predictive performance of classification algorithms is mainly affected by the predictive attributes in a data set. Each predictive attribute describes a specific characteristic of a data set. Usually, the more predictive attributes we have for a data set, the better is our description of its main aspects. However, this is not necessarily true. In real data sets, it is common to have irrelevant, inconsistent and redundant attributes. These can degrade the performance of classification algorithms. Moreover, the predictive performance of a classification algorithm degrades when the ratio between the number of predictive attributes and the number of objects is high. This problem is known as the curse of dimensionality.

11.3 Non-binary Classification Tasks

So far, we have presented classification tasks with only two classes. However, not all classification tasks are binary. Not only does the number of classes vary, but also the relationship between them. These "non-binary" classification tasks include one-class classification, multi-class classification, ranking classification, multi-label classification and hierarchical classification. These classification tasks are described next.

11.3.1 One-class Classification

In some classification tasks, the main goal is the identification of objects in a specific class, usually called the "normal" class. These tasks are called one-class or unary classification tasks. In the training phase, only objects from one of the classes are used for the induction of a classification model. In the test phase, objects not belonging to the normal class should be labeled by the model as "not-normal" or as outlier.

Figure 11.7 shows an example of a one-class classification task. In this example, from sick patient data, the training set has examples belonging to only one class.

Some classification algorithms have been developed or modified to deal with one-class classification. They are used when there are few examples from the negative class or when obtaining negative examples is difficult or expensive. As a side effect, using examples from just one class to induce a model that will later be used to classify examples from more than one class will harm predictive accuracy.

Examples of one-class classification tasks are:

- computer network intrusion detection, where the normal classes are secure operations

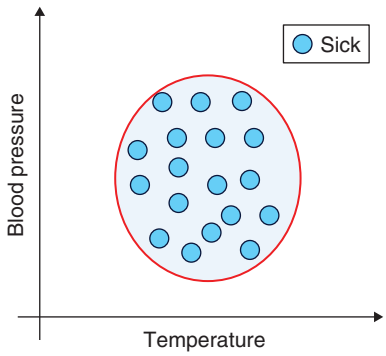


Figure 11.7 Example of a one-class classification task.

- invasive species detection, where the normal class is the presence of a species in a particular region
- credit card transaction fraud detection, where the objects in the normal class are legitimate credit card transactions.

11.3.2 Multi-class Classification

At the opposite end of the spectrum regarding the number of classes are multi-class classification tasks: those where the number of classes an object can belong to is larger than 2. An example of a multi-class classification task can be seen in Figure 11.8. In this figure, the learning algorithm receives during its training, examples from three classes.

Some of the most frequently used classification algorithms are limited to binary classification, for example the SVMs. Since some binary classification algorithms have very good predictive accuracy, it would be good if they could also be used for multi-class classification tasks. There are two alternatives to this end. One is to modify the algorithms' internal procedures to make them able to deal with multi-class classification. The other, more commonly used, is to use decompositional strategies.

A decompositional strategy decomposes the original multi-class classification task into several binary classification tasks, for which any binary classification algorithm can be used. The binary classification outputs are then combined to obtain a multi-class classification. The two main decomposition strategies are one-against-one and one-against-all.

In the one-against-one strategy, the original multi-class task is decomposed into all possible pairs of classes and each pair becomes a binary classification task among two classes. For the one-against-all strategy, a binary classification task is created for each class, which will be one of the classes. The other class will comprise the objects from all other classes.

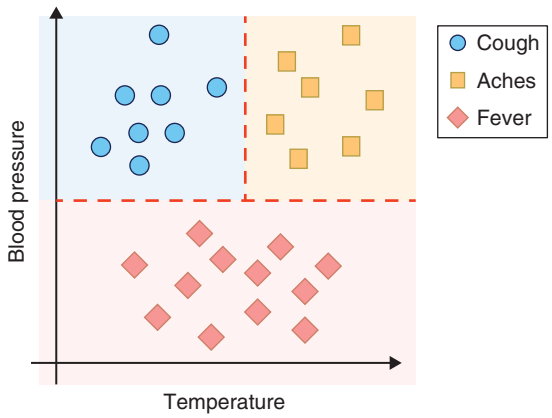


Figure 11.8 Example of a multi-class classification task.

Among multi-class classification tasks, we can include:

- manuscript number and letter classification, where each number and each letter is one class;
- face recognition, where each person is one class;
- heart disease diagnosis, where each disease is one class.

11.3.3 Ranking Classification

In multi-class classification, the final output is one among a set of possible classes. There is a special case of multi-class classification where the output is a ranking of the existing classes, or a subset of these classes, known as a ranking classification.

In the class ranking task, the classes are ordered by classification relevance. Thus, they are ranked in such a way that the top class is the class the classifier is most certain of assigning to the unlabeled object presented. The classification certainty decreases as the class ranking position goes down.

Figure 11.9 presents an example of a ranking classification class, using the same three classes as in the multi-class classification example of Figure 11.8. Now, instead of one of the three classes, the classification algorithm assigns each training object to a ranking of the classes. When a new, unlabeled object is presented for classification, a ranking of the classes is produced by the induced classification model.

Ranking classification is frequently used in applications like recommendation systems, information recovery and search engines. The predictive performance in these tasks is measured by a ranking comparison between the true ranking and the predicted ranking.

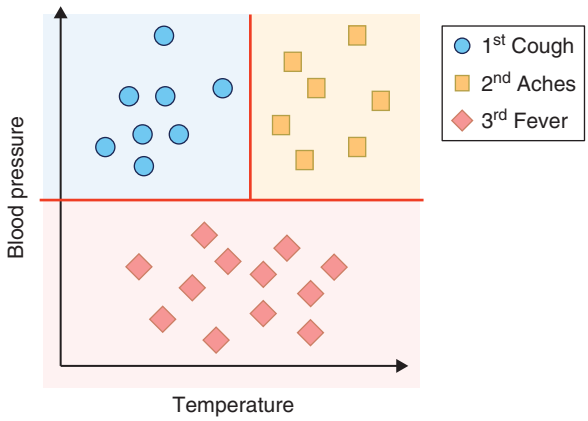


Figure 11.9 Example of a ranking classification task.

11.3.4 Multi-label Classification

Multi-class classification tasks exclusively associate a single class label to each object. For this reason, these tasks are referred to as single-label tasks. Another special case of multi-class classification tasks is multi-label classification, where each object can simultaneously belong to more than one class [46].

Traditional classification algorithms were not designed to deal with multi-label classification. Therefore, the data set must be transformed. Two main approaches are adopted. In the first approach, the original multi-label classification task is decomposed into single-label classification tasks. This transformation can be either independent of or dependent on the classification algorithm used. In the second case, single-label classifiers have their internal procedures modified or a new algorithm is designed. The second approach transforms the set of labels in a multi-labeled object into a single label by creating a new label to represent this set of labels.

An example of multi-label classification, using the same classes as in the previous examples, can be seen in Figure 11.10. Note that in this classification task, some objects have two class labels and one object has three class labels.

Examples of multi-label classification tasks include newspaper article classification, scene classification, music genre classification, protein function classification and website classification [47].

Multi-label classification is related to ranking classification, since both techniqes can assign more than one label to an object. In ranking classification, two or more objects can have the same ranking position. The difference is the order of the labels assigned to an object for ranking classification, and the attribute number of labels assigned to each object in multi-label classification.

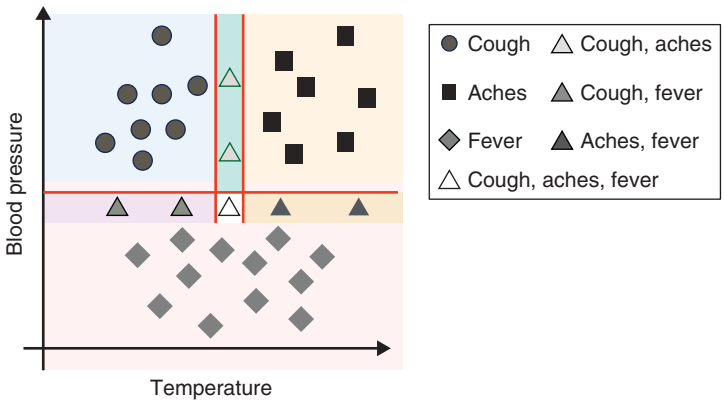


Figure 11.10 Example of a multi-label classification task.

11.3.5 Hierarchical Classification

The vast majority of classification tasks are flat classification tasks, where each sample is associated with a class belonging to a finite set of classes, excluding hierarchical relationships. However, for some classification tasks, known as hierarchical classification tasks, the classes can be structured into a hierarchical structure, with subclasses and superclasses [48].

In these tasks, each unlabeled object can be classified in the class associated with any node (prediction to any node) or in one of the leaf nodes of the hierarchical structure (mandatory leaf node prediction).

An example of a hierarchical classification task is presented in Figure 11.11. In this example, the classes goes from the most generic, at the tree root, to the most specific, at the leaves. In the training process, the label to be associated with each object is a particular node (class) in the class hierarchy.

In hierarchical classification tasks, a learning algorithm induces a model that captures the most relevant relationships between the classes found in the set of training data, considering the hierarchical relationships between the classes.

In these tasks, each unlabeled object can be classified in a class associated with a node (prediction to any node) or in one of the leaf nodes of the hierarchical structure (mandatory leaf node prediction). The closer to the tree root is the class associated with an unlabeled object, the lower the classification error rate tends to be. On the other hand, the classification obtained is less specific, and therefore less useful.

Several strategies have been proposed for the induction of models for hierarchical classification. These can be grouped into four main approaches:

- transformation into a flat classification problem
- hierarchical prediction using flat sorting algorithms
- ranking top-down
- one-shot classification or "Big Bang" [48].

The top-down strategy performs the classification in stages, from the root node to the leaf nodes. The one-shot strategy generates a unique classifier for the

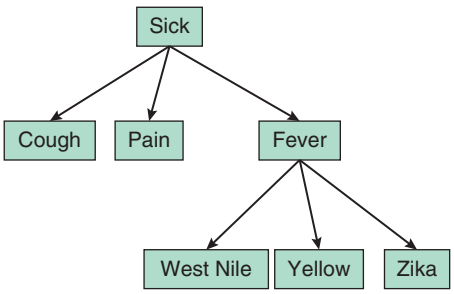


Figure 11.11 Example of a hierarchical classification task.

entire hierarchy, which increases the complexity of the induced models. It is important to note that many hierarchical classification tasks are also multi-label classification [49].

Among hierarchical classification applications, it is possible to mention protein function prediction, classification of songs according to style, text classification, web page classification and image classification.

11.4 Advanced Data Preparation Techniques for Prediction

The data available to create and test predictive models are unsuitable, in many problems, to give good predictive models. Some of the most common undesirable characteristics are:

- *imbalanced data:* when the possible values of the target attribute are unequally represented;
- *incomplete target labeling:* when there are unlabeled objects.

Approaches to solve these problems are described next.

11.4.1 Imbalanced Data Classification

One problem frequently faced in classification tasks is data imbalance. The data sets used in classification tasks usually have different numbers of objects in each class. If the numbers are similar and the data in each class are good representations of the possible data distribution in the class, there is no problem. However, if at least one of the classes has many more examples than the other class(es), the induced model tends to favor the class(es) with more examples. In a binary classification task, if one class has a number of examples that is significantly larger than the other, this class is called the majority class.

Several techniques have been proposed to deal with class imbalance. Most of them are restricted to binary classification tasks, so these are the strategies discussed here. However, they can be readily adapted to other classification tasks. The data set from Table 9.3, illustrated in Figure 9.4 is an imbalanced binary classification data set. The two most common strategies to deal with this situation are undersampling and oversampling.

The undersampling strategy reduces the number of objects in the majority class, in order to approximate the number of objects in the two classes. A deficiency of this strategy is that important objects from the majority can be lost.

The oversampling strategy increases the number of objects in the minority class, by duplicating some objects of the minority class or by creating new objects from existing ones. This second approach makes it possible to create objects that would never be found in the real problem.

A method that uses such approach is the synthetic minority oversampling technique (SMOTE). It selects k neighbors in relation to each minority object. Then, for a given percentage p of oversampling chosen, around $p/100$ from the k nearest neighbors are chosen. An object is synthetically generated by choosing randomly for each attribute a value between the attribute value of the object under consideration from the minority class and the attribute value of its neighbor.

Imbalanced data is usually associated with classification tasks. However, there are also approaches to deal with imbalanced data for regression problems.

11.4.2 For Incomplete Target Labeling

When there are unlabeled objects in our data set, two approaches can be used:

- *semi-supervised learning:* a learning schema that can use both the labeled and the unlabeled objects in the learning process;
- *active learning:* a technique to choose the unlabeled objects that are expected to be more useful in the learning process if manually labeled.

These two approaches will be described next.

11.4.2.1 Semi-supervised Learning

For the application of a classification technique to a data set, each object in the training set must be labeled with one of the classification task classes. In many situations, the class labeling is performed by humans. The assignment of a class label to the training set objects can therefore have a high cost in both money and time. The cost can be so high that the labeling of a large training set can become unfeasible.

Consider, for example, the labeling of data for a credit risk analysis based on personal credit score. The class labels are "able to pay" and "not able to pay". The personal credit score of a person is defined by their financial history and current personal data. We need a credit analyst to label whether each training object should be categorized as able to pay or not able to pay. The predictive performance of classification model induced with these data is strongly influenced by the quality of the class label assignment. On the other hand, the acquisition of unlabeled data is usually inexpensive.

An alternative way to reduce the labeling cost is to use semi-supervised learning, which is a learning process situated between unsupervised learning and supervised learning. Semi-supervised learning techniques can be applied to partially labeled data sets: data sets having only a fraction of the objects labeled. In predictive tasks, semi-supervised learning usually combines transductive learning with inductive learning. The transductive learning process predicts the class label to be associated with unlabeled training objects. The inductive learning process, as previously seen, uses a labeled training set to induce a predictive

model able to predict the class of new objects. Semi-supervised learning tries to improve the quality of the inductive learning – particularly when the data labeling is costly, resulting in a small labeled training set – by also using information present in the unlabeled training data.

To see how semi-supervised learning works, let us consider a simple example: suppose you have a training data set of 200 objects, only 30 of which have a class label. In semi-supervised learning, we start with transductive learning: we use these 30 labeled objects to induce a classification model. Next, we apply this classifier to the 170 unlabeled training objects, in order to predict their class label. In this process, we also measure how confident we are in each of the 170 class label predictions. The 30 objects for which we have the most confidence in the label prediction are incorporated into the labeled training set, with the class label predicted for them. The other training objects remain unlabeled. Now, we have a training set of 60 labeled and 140 unlabeled. We perform the transductive learning again, now with the larger labeled training set, to predict the class label of the 140 unlabeled training objects. This process continues until all training objects have a class label. In the final round, the class predicted for the last 20 unlabeled objects is assigned. Once all training set objects are labeled, inductive learning is applied to the training data to induce a predictive model.

Semi-supervised learning can also be used for descriptive learning, when some of the objects, typically a small percentage, are labeled. As an example, in clustering, the labeled objects can be used to define the ones that should be in the same cluster and the ones that should not be in the same cluster, improving the quality of the cluster partitioning.

11.4.2.2 Active Learning

The predictive performance of classification models is strongly affected by the quality of the data labeling process. The cost of data labeling can be reduced by selecting the most promising objects for the labeling. Such as approach is the use of active learning strategies [50].

Active learning investigates less costly strategies for object labeling. Active learning techniques have been successfully used to select data to be labeled. Unlike semi-supervised learning, active learning uses a strategy to select examples whose labeling will be most beneficial for the induction of a classification model. Examples of these situations are choosing objects closer to the decision border or objects that are very different to the labeled objects.

11.5 Description and Prediction with Supervised Interpretable Techniques

So far, all of the situations we described for use of supervised learning have been predictive tasks. In spite of having access to class label information, supervised

classification techniques can also be used in descriptive tasks. In this case, they can be trained with all of the objects in the data set, since the induced model will not be used for prediction. The purpose now is to describe patterns present in a data set.

The interpretation that results will be of the data that are available. The larger and more representative the data set, the closer to the real situation will be the interpretation. To be able to describe the data set, the model needs to be interpretable by the user. This excludes techniques like neural networks and support vector machines.

Example 11.1 An example of the use of a supervised technique for a descriptive task is the application of a decision tree induction algorithm to our social network data set. This will result in a decision tree, like the one illustrated in Figure 10.3. This decision tree describes the main aspects of the "company" classes by showing the patterns of predictive attributes for the objects in each class.

Although the model obtained has the same kind of information whatever the task is – descriptive or predictive – the experimental setup used is different for each of these tasks. In a descriptive task, it is not necessary to evaluate the predictive performance. It is therefore not necessary to use resampling techniques, as described in Section 8.1.2 or performance measures, as described in Section 8.1.3 for regression and in Section 9.2 for classification.

Evaluation measures for models with descriptive purposess have as their main principle the measurement of how well the model fits the data. However, using more flexible models, such as decision trees, it is possible to obtain models that adjust fully to the data. For that reason, the evaluation measures should also take into account the complexity of the model. Examples of such evaluation measures are the Akaike information criterion (AIC) and the Bayesian information criterion (BIC).

11.6 Exercises

1 Give one advantage and one disadvantage of the parallel approach over the sequential approach for generating the base learners of an ensemble of classifiers.

2 Why can we use any predictor in bagging and AdaBoost, but only decision trees in random forests?

3 Why do some ensemble methods work better with unstable predictors?

4 What are the search bias and the representation bias of decision trees?

5 Describe three multi-class classification problems in your daily life.

6 Explain to what extent multi-label and ranking classifications are related.

7 How could the activities "walking", "running", "sitting down" and "lying" be classified hierarchically?

8 Given the social network dataset from Table 9.4, using the first ten objects as the training set and the last ten objects as the test subset, perform the following activities:

a) Use bagging with ten decision trees to predict the class label of the test subset objects.
b) Use a random forest with ten decision trees to predict the class labels of the test subset objects and test 1, 2 and 3 as the number of attributes to be selected randomly at each split-node.
c) Repeat the experiments using AdaBoost with decision trees. For the hyper-parameters, use the default values.
d) Compare the results from the three sets of experiments.

12

Cheat Sheet and Project on Predictive Analytics

Like the previous project chapter, this chapter has two sections: a cheat sheet of the contents in Part III; and the resolution of the project described in Section 1.6.2.

12.1 Cheat Sheet on Predictive Analytics

A summarization of the methods described in Part III is now presented in Table 12.1. Each method is classified according to the following criteria:

- *Method:* name of the method
- *CR:* classification and/or regression
- *#hp:* number of hyper-parameters
- *PrP:* pre-processing, which can have the following values
  - CS: center and scale or "normalization"
  - COR: remove correlated features
  - FS: feature selection
- *PC:* processing cost
- *Int:* interpretability.

Here and in the later tables, + means positive, − negative and −+ is in between.

12.2 Project on Predictive Analytics

This project relates to the CRISP-DM methodology. The data used for this study can be obtained in the UCI machine learning repository [6], easily obtainable in the web, entitled "Polish companies bankruptcy data" [51].

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

Table 12.1 A cheat sheet on predictive algorithms.

Method	CR	#hp	PrP	PC	Int
Multivariate linear regression	R	0	COR/FS	+	+
Ridge regression	R	1	COR/CS/FS	+	+
Least absolute shrinkage and selection operator	R	1	COR/CS/FS	+	+
Principal components regression	R	2	COR/CS	+	−+
Partial least squares	R	3	COR/CS	+	−+
K−nearest neighbors	CR	2	CS/FS		−
Logistic regression	C	0	COR/CS/FS	+	+
Naïve Bayes	C	0	COR/FS	+	−+
Decision tree induction algorithm (C4.5)	CR	10		+	+
Model tree induction algorithm	CR	2		+	+
Multivariate adaptive regression splines	R	2		+	+
Artificial neural networks	CR	6	FS	−	−
Deep learning (CNN)	CR	10		−	−
Support vector machines	CR	3	FS	−+	−
Bagging	CR	2		−	−+
Random forest	CR	2		−	−+
AdaBoost	C	2		−	−

12.2.1 Business Understanding

Investors, banks and many other institutions and shareholders have an interest in predicting how viable a company is. The business objective is to predict whether a given company will be insolvent in the next five years.

12.2.2 Data Understanding

The data collected has the .arff extension. This extension is used by the data mining software WEKA[52], but can be read with any text editor. We will use only the first of five files available, which is the data for the first year. The target attribute is binary (0/1). The predictive attributes are described in Table 12.2.

Once you have your data set, you need to understand it, assessing its quality and describing the data using statistics and visualization techniques. Some statistics of the attributes are presented in Table 7.8.

A quick analysis of Table 12.3 allows us to identify some problems:

- *missing values:* 844 instances without a target label and almost all attributes having missing values.

Table 12.2 Predictive attributes of the Polish company insolvency data set.

Attribute	Description
X1	net profit/total assets
X2	total liabilities/total assets
X3	working capital/total assets
X4	current assets/short-term liabilities
X5	[(cash + short-term securities + receivables − short-term liabilities)/(operating expenses − depreciation)] * 365
X6	retained earnings/total assets
X7	EBIT/total assets
X8	book value of equity/total liabilities
X9	sales/total assets
X10	equity/total assets
X11	(gross profit + extraordinary items + financial expenses)/total assets
X12	gross profit/short-term liabilities
X13	(gross profit + depreciation)/sales
X14	(gross profit + interest)/total assets
X15	(total liabilities × 365)/(gross profit + depreciation)
X16	(gross profit + depreciation)/total liabilities
X17	total assets/total liabilities
X18	gross profit/total assets
X19	gross profit/sales
X20	(inventory × 365)/sales
X21	sales (n)/sales ($n - 1$)
X22	profit on operating activities/total assets
X23	net profit/sales
X24	gross profit (in three years)/total assets
X25	(equity − share capital)/total assets
X26	(net profit + depreciation)/total liabilities
X27	profit on operating activities/financial expenses
X28	working capital/fixed assets
X29	logarithm of total assets
X30	(total liabilities − cash)/sales
X31	(gross profit + interest)/sales
X32	(current liabilities × 365)/cost of products sold
X33	operating expenses/short-term liabilities
X34	operating expenses/total liabilities

(*Continued*)

Table 12.2 (Continued)

Attribute	Description
X35	profit on sales/total assets
X36	total sales/total assets
X37	(current assets − inventories)/long-term liabilities
X38	constant capital/total assets
X39	profit on sales/sales
X40	(current assets − inventory − receivables)/short-term liabilities
X41	total liabilities/((profit on operating activities + depreciation) × (12/365))
X42	profit on operating activities/sales
X43	rotation receivables + inventory turnover in days
X44	(receivables × 365)/sales
X45	net profit/inventory
X46	(current assets − inventory)/short-term liabilities
X47	(inventory × 365)/cost of products sold
X48	EBITDA (profit on operating activities − depreciation)/total assets
X49	EBITDA (profit on operating activities − depreciation)/sales
X50	current assets/total liabilities
X51	short-term liabilities/total assets
X52	(short-term liabilities × 365)/cost of products sold)
X53	equity/fixed assets
X54	constant capital/fixed assets
X55	working capital
X56	(sales − cost of products sold)/sales
X57	(current assets − inventory − short-term liabilities)/(sales − gross profit − depreciation)
X58	total costs /total sales
X59	long-term liabilities/equity
X60	sales/inventory
X61	sales/receivables
X62	(short-term liabilities × 365)/sales
X63	sales/short-term liabilities
X64	sales/fixed assets

Table 12.3 Statistics of the Polish company insolvency data set.

Attr	Type	Missing	Min/least	Max/most	Average/values
X1	Real	3	−189.560	453.770	0.314
X2	Real	3	−141.410	1452.200	2.624
X3	Real	3	0	3876.100	5.553
X4	Real	30	−440.550	1099.500	1.826
X5	Real	8	−189.450	453.780	0.354
X6	Real	3	−23.207	331.460	0.800
X7	Real	3	−607.402	13315	2.093
X8	Real	25	−141.410	1452.200	2.624
X9	Real	1	0	3876.100	5.553
X10	Real	3	−440.550	1099.500	1.826
X11	Real	39	−189.450	453.780	0.354
X12	Real	30	−23.207	331.460	0.800
X13	Real	0	−607.420	13315	2.093
X14	Real	3	−189.560	453.770	0.314
X15	Real	2	−5611900	3599100	1802.696
X16	Real	25	−42.322	405.330	0.871
X17	Real	25	−0.413	1529.900	3.752
X18	Real	3	−189.560	453.770	0.314
X19	Real	0	−622.060	2156.800	0.562
X20	Real	0	0	7809200	1162.128
X21	Real	1622	−1325	27900	10.368
X22	Real	3	−216.800	454.640	0.288
X23	Real	0	−634.590	2156.800	0.424
X24	Real	124	−189.560	831.660	0.540
X25	Real	3	−459.560	1353.300	1.264
X26	Real	25	−21.793	612.880	0.831
X27	Real	311	−14790	2040800	1321.989
X28	Real	34	−490.090	1570	2.703
X29	Real	3	0.176	9.386	4.195
X30	Real	43	−149.070	152860	23.705
X31	Real	297	−622	2156.800	0.500
X32	Real	636	0	351630	237.064
X33	Real	763	0	884.200	7.473
X34	Real	818	−280.260	884.200	3.931
X35	Real	830	−169.470	445.470	0.356

(*Continued*)

Table 12.3 (Continued)

Attr	Type	Missing	Min/least	Max/most	Average/values
X36	Real	841	0.000	3876.000	6.447
X37	Real	3265	−525.520	398920	190.201
X38	Real	841	−20.340	1099.500	2.188
X39	Real	839	−14.335	2156.500	0.440
X40	Real	869	−101.270	1014.600	0.883
X41	Real	914	−11.976	813.140	0.664
X42	Real	840	−35.214	2156.800	0.435
X43	Real	840	0	30393000	5034.468
X44	Real	840	0	22584000	3728.024
X45	Real	948	−0.599	5986.800	8.252
X46	Real	869	−101.260	1017.800	1.960
X47	Real	869	0	47794	80.008
X48	Real	843	−218.420	405.590	0.186
X49	Real	842	−9001	31.639	−1.408
X50	Real	865	0	261.500	2.161
X51	Real	846	0	21.261	0.385
X52	Real	872	0	354.360	0.462
X53	Real	875	−130.470	180440	107.276
X54	Real	875	−82.303	180440	108.245
X55	Real	844	−589300	4398400	10496.129
X56	Real	844	−1108300	1	−179.139
X57	Real	845	−15.813	71.053	0.291
X58	Real	844	−0.004	1108300	180.140
X59	Real	845	−256.990	119.580	0.290
X60	Real	953	0.000	361820	132.159
X61	Real	862	0.000	21110	16.433
X62	Real	844	0	25016000	4164.117
X63	Real	869	0.000	1042.200	8.635
X64	Real	875	0.000	294770	218.049
CLASS	Binomial	844	1 (194)	0 (5989)	0 (5989), 1 (194)

- *redundant attributes:* 24 pairs of attributes with absolute correlation values larger than 0.8: 22 positively correlated and 2 negatively correlated.
- *noisy data:* almost all attributes have extreme values; either noise or outliers.

Moreover the data set is clearly unbalanced since it has 5989 instances of the negative class and only 194 of the positive class.

12.2.3 Data Preparation

The following steps were taken:

1) The 844 instances without a target label were removed. All of them were from the majority class: companies that were no insolvent.
2) All predictive attributes were normalized using the z-norm.
3) In order to remove extreme values, the 167 (17 class 1 and 150 class 0) normalized values with absolute values larger than 5 were removed.
4) All missing values were filled with the average of the respective predictive attribute.

Nothing was done in order to remove correlated attributes because forward feature selection is used in the modeling phase, a process that favors the removal of correlated attributes.

12.2.4 Modeling

Modeling was done with three different algorithms:

- K-nearest neighbors: $k = 15$ was used, without tuning, and the Euclidean distance was the distance measure. A hold-out split using 70% of the data was used to train the model and the remaining 30% was used to evaluate the forward feature selection. The selected features were X6, X11, X24, X27 and X60.
- C4.5 decision tree: a 25% confidence threshold was used for pruning and 2 was taken as the minimum number of instances per leaf.
- Random forests: 500 trees were generated.

The 70% partition of the hold-out split was used to do 10-fold cross validation. All three algorithms used the same partitions of the 10-fold cross validation. C4.5 and random forests were used both with all attributes and using only the five attributes selected by forward search. These five attributes were selected using forward selection with the k-nearest neighbor algorithm.

Many more experiments and algorithms could have been performed/used. The use of approaches for unbalanced data sets could have been tested. Several other algorithms could have been used.

12.2.5 Evaluation

In classification, there are a large number of measures for evaluation, the choice depends on the objectives. The most important goal in this problem is to predict correctly as many actual insolvencies as possible. This is class 1. Observing

Table 12.4 K-NN confusion matrix using five predictive attributes.

	true 0	true 1	class precision
pred 0	5812	106	98.21%
pred 1	14	84	85.71%
class recall	99.76%	44.21%	

Table 12.5 C4.5 confusion matrix using five predictive attributes.

	true 0	true 1	class precision
pred 0	5799	100	98.30%
pred 1	27	90	76.92%
class recall	99.54%	47.37%	

Table 12.6 Random forest confusion matrix using all predictive attributes.

	true 0	true 1	class precision
pred 0	5794	90	98.47%
pred 1	32	100	75.76%
class recall	99.45%	52.63%	

Tables 12.4–12.6, the random forest was the best approach, able to predict 100 out of 190 of the insolvencies that occurred in practice. New experiments should be done until an acceptable error rate be achieved.

12.2.6 Deployment

The use of this kind of result could be displayed as a web page, for instance. The deployment phase would therefore involve web site construction using the prediction model prepared by the data science team.

Part IV

Popular Data Analytics Applications

13

Applications for Text, Web and Social Media

This chapter will describe three current fields of data analytics that are attracting a great deal of attention due to their wide application in different domains:

- text mining, which extracts knowledge from texts
- social network analysis, which looks for information that can be extracted from social relations
- recommendation systems, which use previous selections from the same user or from other users to recommend items like books, movies and tourist packages.

We devote a section to each one of these applications.

13.1 Working with Texts

Suppose now that your social network has grown considerably, so you now have hundreds of contacts. As a result, everyday you receive hundreds of messages on your smartphone. Up till now, you have read all your messages, but you are spending a good part of your day reading them and, as your number of contacts increases, so the number of messages to read increases too. Would it not be nice to have a filter able to indicate messages you need to read and those you do not need to read? You can do this using text mining.

With the increasing use of social network tools and emails, associated with the fast expansion of blogs and texts available in the Internet, we have a large amount of data in text format. Texts are the most common way to exchange information in our society. Much precious information can be hidden in these texts. While humans can easily extract meaningful information from a text, the same is not true of computer software.

Consider, for example, a simple note you wrote about the food preferences of one of your friends, Fred, who is vegetarian. Figure 13.1 shows the note you wrote down in your social network a while ago.

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

> Fred very much likes having dinner in Chinese restaurants. Since Fred is vegetarian, he doesn't eat meat. In order to have sufficient protein, Fred is always looking for other foods that have protein levels similar to those found in meat.

Figure 13.1 Text about the food preferences of a friend.

How can you automatically extract useful knowledge from this text? We saw in the previous chapter several data analytics techniques, in particular, machine learning algorithms, that can extract knowledge from data. However, these techniques can only be applied to data in tabular format, which is not the case for the text in Figure 13.1. Texts, like images, movies and sounds, do not have a tabular format. To distinguish between these two formats, tabular data is referred to as "structured" data and the other data formats are called "unstructured" data.

An area very close to data mining is text mining, which provides several techniques specifically developed to extract knowledge from raw texts written in a natural language. We can say that while data mining is associated with data, text mining is associated with text [53].

The origins of text mining goes back to document indexing tasks in the area of information retrieval. Information retrieval is usually concerned with the retrieval of information from on-line documents. They are a key area in web search engines, which use similarity between documents to identify relevant web sites.

Text mining is a very active area of data analytics. It investigates and provides tools to extract knowledge from texts. Text mining is an important part of several other tasks, like information retrieval, spam detection, sentiment analysis, recommender systems and web mining. For these applications, a key aspect is how to measure the similarity between texts.

As in data mining, text mining tasks can be descriptive or predictive. Descriptive text mining tasks include looking for groups of similar documents, and looking for texts about similar issues and words that frequently appear together in texts. Predictive tasks include classification of documents into one or more topics and identification of spam in emails and sentiment analysis in text messages.

This section will focus on predictive text mining, also known as text categorization and document classification. The terms "text" and "document" will be used interchangeably in this section.

As already mentioned, most data mining techniques expect the data to be in a attribute–value tabular format, so they cannot be directly applied to textual data. However, several techniques have been developed to extract structured data from raw text. Thus, one of the first steps in text mining is the transformation of texts into tabular, attribute–value format. In this section we will

describe some of these techniques and show how can transform a text into an attribute–value table, a tabular format.

To illustrate how text mining works, let us go back to our automatic message classification task. Suppose we want to divide the messages we receive into two groups: work and family. To this end, we can use data analytical tools to induce a model able to automatically classify our messages into one of these two groups.

The predictive text mining process is very similar to a data mining process. The main difference is the of transformation of unstructured data into structured data and the use of preprocessing techniques specific to text. In summary, a text mining task comprises five phases:

- data acquisition
- feature extraction
- data preprocessing
- model induction
- evaluation and interpretation of the results.

The last three phases are carried out using data mining techniques. After the first step, the data will be in structured format. Therefore, we will concentrate here on the first two steps.

13.1.1 Data Acquisition

We saw in the beginning of this book that we first need to collect a data set with representative objects: objects similar to those we believe we will receive in the future. Of course, we cannot be sure of the characteristics of future messages, but if we can collect a diverse range of objects we have a good chance of having a representative sample. If we have a very large number of messages, and good storage and processing capacity, we could just collect all messages from a given period, say the previous 12 months. A collection of texts or documents is known as a corpus. Each text in the corpus will be converted to a structured object.

If the texts come from different sources, they may have different formats, like ASCII or Unicode text format or the Extensible Markup Language (XML) format, which is the standard document exchange format. An XML file has key words – tags – used to mark some parts of document. These tags can give meaningful information about the content in these parts, for example indicating the title of the document, the authors, the date, the topics and the abstract. The tags can be use to identify the part of document to be mined. Texts that are not natural language, like email and web site addresses, are easily detected and can be removed if necessary.

13.1.2 Feature Extraction

Once all texts have undergone this process, each text or document will be a stream of characters, which can include words, numbers, white spaces, punctuation characters and special characters. As in a predictive data mining task,

Table 13.1 Training set of labeled texts.

Message received	Class
I like my sister's birthday party	Family
I liked the company party	Work
I am not bringing them from school	Family
I will talk and bring the contract	Work
I talked to other companies	Work
My wife is having contractions	Family

we separate a subset of the texts to become our training data set. These are the texts we will use to induce a predictive model, which can then be applied to new texts, after their conversion to quantitative values.

13.1.2.1 Tokenization

The next step is to extract, for each text, a sequence of words from the stream of characters. In this procedure, called tokenization, each word in the sequence is called a lexical token. Words are detected by looking at white space and punctuation characters. If a word appears more than once in the text, its token will appear more than once in the sequence of tokens. This procedure of representing a text by a set of tokens, where each token can appear more than once, is known in the information retrieval and natural processing language fields as a "bag of words". Depending on the context of the document, some special characters may also be tokens. For some applications, a whole phrase can be a token.

Example 13.1 To illustrate how text mining works, let us suppose that we have a small training set with six short messages received from our family and work colleagues. In text classification tasks, each text in the training set is labeled by one or more topics. To make the example simpler, let us suppose that we have assigned one topic to each training set message, which will become the message class label. In this cases, the topics are "Family" and "Work". Table 13.1 shows the texts and their labels.

In this example, we have a small number of short texts. In practice, we usually have a large collection of texts. For message exchange applications and sentiment analysis, the texts are usually short. For other applications, long texts are more common.

13.1.2.2 Stemming

The tokens can be several variations of a word, like plurals, verb inflections and gender forms. Thus, if we represent each text by all the tokens that appear in all

texts from the training set, we will probably end up with a very large number of tokens. However, most texts will have just a small fraction of all these tokens.

We said before that when we transform the texts into a table of quantitative values, each word, or now each token, will become a predictive attribute. Since just a small proportion of the tokens will be present in a particular text, the predictive attributes whose token does not appear in the text of an object will have the value 0. This will make the whole table very sparse, since most of its predictive attribute values will be equal to 0.

To avoid having a very large number of tokens, which can result in a very sparse data set, we look for a common base form able to represent many of the variations of a token. In one of the simplest methods, each token is converted to its stem, a process called "stemming", which uses a stemming algorithm, also called a "stemmer". There are different stemmers for different natural languages. Even for one language, like English, we can have different stemmers. Among the stemming algorithms for the English language, one of the most common is the Porter stemming algorithm or stemmer [54].

Example 13.2 For example, Table 13.2 illustrates the stems resulting from the application of the Porter stemmer to variations of four words.

Thus, the Porter stemmer will convert the words "studies", "studied" and "studying' to the stem "studi". Stemming is a very simple method that usually just removes affixes of words. An affix at the beginning of a word is a prefix and in the end of a word is a suffix.

Now, let us apply the same stemming algorithm to the text messages from our training set of labeled texts. Table 13.3 shows the stems of each object, together with the object label.

These simple operations resulted in a significant reduction in the number of tokens, representing each token by its stem.

There is also a variation of stemming, called lemmatization. Lemmatization is a more sophisticated method to extract the common base forms of words, since it uses a vocabulary and takes grammatical aspects of language into account,

Table 13.2 Results of stemming.

Original words	Stems
studied, studying, student, studies, study	studi, studi, student, studi, studi
miner, mining, mine	miner, mine, mine
vegetable, vegetarian, vegetate	veget, vegetarian, veget
eating, ate, eats, eater	eat, ate, eat, eater,

Table 13.3 Results of applying a stemmer.

Message received after stemming	Class
I, like, my, sister, birthday, parti	Family
I, like, the, compani, parti	Work
I, am, not, bring, them, from, school,	Family
I, will, talk, and, bring, the, contract	Work
I, talk, to, other, compani	Work
My, wife, is, have, contract	Family

performing a morphological analysis. The lemmatization of a word returns the dictionary form of a word, which is called a lemma.

The stemming in the previous example reduced the number of different tokens by extracting their stems. Even so, we still have 23 stems. Since each stem will become a predictive attribute, each object will have 23 predictive attributes. Since most texts have 5 stems, around 75% of the predictive attributes will not be present in most objects and will result in 103 predictive attributes with value equal to 0. We will still have a sparse table after the conversion to the tabular format. The good news that there are still other procedures to reduce the number of stems and, as a result, the number of predictive attributes.

The number of stems can be further reduced by removing stop words. Stop words are words that are very common in texts. Therefore, they have low discriminative power and will probably not be useful for the next text mining phases. Some examples of stop words are:

- adjectives (good, bad, large…)
- adverbs (fast, nicely, not…)
- articles (a, an, the)
- negations (none, not, never…)
- pronouns (I, he, my, his, yours, ours…)
- prepositions (at, by, for, from, in, like, on, to…)
- conjunctions (and, but, or, with,…)
- frequent verbs (are, be, is, was, has, have…)
- qualifiers (a little, less, more, other, probably, same, somewhat, very, yet…).

It is important to note that one word can have different meanings. For example, the word "like" can be, according to the context, a verb, a preposition or a conjunction. Several words can be adverbs in some texts and adjectives in others. In simple stop word detection techniques, one of the meanings is assumed.

The decision of which stop words to remove depends on the application. For instance, in sentiment analysis applications, the presence of adjectives and

Table 13.4 Stems after removal of stop words.

Stems after removal	Class
sister, birthday, parti	Family
compani, parti	Work
bring, school,	Family
talk, bring, contract	Work
talk, compani	Work
wife, contract	Family

negations can be important. What usually occurs in text mining is the selection of the most adequate subset of the stop list for the application.

In addition, some stems can also occur very rarely in a text and therefore will probably not be useful for the induction of a predictive model. It has been estimated that half of the words in a corpus appear just once [55]. Thus stems with very low frequency in the corpus can be removed.

Some text mining applications also use stop *phrases*, where a whole phrase that appears very often in the text and has low discriminative power can be removed. Stop words can be identified and removed before the feature extraction step. However, their removal should take place after all previous transformations.

Example 13.3 If we remove the stop words, considering the word "like" a stop word, we can transform the composition of stems for each object from the previous example to the objects illustrated by Table 13.4.

With the removal of the stop words, we have reduced the number of different stems from 23 to 9. Most of the objects have just 2 stems, so the quantitative table will still be sparse, but with a much smaller number of predictive attributes with the value 0. There was a reduction from 103 to 40 predictive attributes with the value 0.

13.1.2.3 Conversion to Structured Data

The next step in text mining is to transform the text mining task into a data mining task. To do this, information in unstructured format – text – must be converted to structured format, namely a table with quantitative values.

We initially perform this conversion in the text training set: the texts we will use to induce a predictive model, which can then be applied to new texts.

The value of the predictive attribute associated with a stem can be as simple as a binary value indicating the presence of the stem in the text. For example,

Table 13.5 Objects with their stems.

birthday	bring	compani	contract	parti	school	sister	talk	wife	Class
1	0	0	0	1	0	1	0	0	Family
0	0	1	0	1	0	0	0	0	Work
0	1	0	0	0	1	0	0	0	Family
0	1	0	1	0	0	0	1	0	Work
0	0	1	0	0	0	0	1	0	Work
0	0	0	1	0	0	0	0	1	Family

a value of 1 would represent the presence of the stem in the text, with 0 for its absence. This procedure simplifies the implementation and the data analysis. However, the number of times each stem appears in the text can be important information for text classification.

Therefore, the usual procedure records the number of times a particular stem appears in the message. The most common way to do this is to use the bag of words method, as introduced in Section 5.1.3. This method extracts the stems that appear in each text, as filtered by tokenization and stemming processes. Each stem obtained from the texts, from any class, will become a predictive attribute. Thus each text will be represented by an object with n predictive attributes, one for each stem that was found in the training set texts. The values will be the number of occurrences of the stem.

Example 13.4 Since the texts used are very short, no stem occurs more than once in each stemmed text after the removal of stop words. Thus, all predictive attributes will have a binary value: 1 for the presence of the stem in the object and 0 when it does not appear in the object. As a reminder, each text is transformed to an attribute–value object. Table 13.5 illustrates the tabular format obtained.

Now the data look more like the data sets we saw in the previous chapters of the book.

13.1.2.4 Is the Bag of Words Enough?

Sometimes, the occurrence of each word is not enough, and can even be misleading. Looking at the text in Figure 13.1, the word meat appears twice as frequently as the word vegetarian. By purely counting the words, our data analytics method may conclude that Fred *likes* meat. Moreover, if we have a negative before a word, the negation is not taken into account.

In more sophisticated text mining approaches, techniques from natural language processing can be used. Despite more accurate text interpretation, the use of these techniques for large texts will slow down the text mining process.

13.1.3 Remaining Phases

Once we have the data in tabular format, we have the next phases: data preprocessing, model induction, and evaluation and interpretation of the results. Conventional data mining techniques are used in theses phases.

Preprocessing may include dimensionality reduction. In text mining applications, after all the steps we have presented – tokenization, stemming and removal of stop words and words with very low frequency – we still can have a large number of predictive attributes and very sparse data. Dimensionality reduction techniques are therefore often used to further reduce the number of predictive attributes and the sparsity.

The evaluation measures for text mining tasks are usually the same as those used in data mining tasks. In some applications, information retrieval measures are also used.

13.1.4 Trends

Text mining is a very hot topic in data analytics and several text mining tools and applications have been developed and commercialized. Current trends in text mining include:

- its combination with image processing techniques to extract knowledge from old printed documents and books
- combination with natural language processing techniques for text understanding
- identification of text idioms and translation of texts to other idioms
- discovery of authorship of academic texts
- identification of plagiarism in documents, books, news and academic articles
- extraction of information from news published by the media to summarize news received from different sources and to provide personal selections of news
- monitoring of health-related literature to discover new knowledge for improving medical diagnoses
- sentiment analysis, mining opinion in text messages.

Another frequent application is the extraction of metadata from texts, which is important information present in the text. As an example, suppose we want to filter job offers. The metadata extracted could be company name, country, address, web site, email address and phone number, closing date for

applications, desired applicant characteristics, job requirements, salary, skills and starting date.

Two very important applications of text mining are sentiment analysis and web mining. The main aspects of these two applications are described next.

13.1.4.1 Sentiment Analysis

A special case of text mining is the analysis of short texts exchanged through social network tools. This analysis, known as sentiment analysis or opinion mining, is usually carried out on texts with a limited number of characters. In such cases, the use of bag of words approach usually leads to good results with low processing cost.

Sentiment analysis techniques have been used to analyze attitudes, evaluations, opinions and sentiments of users, regarding entities, events, issues, public figures, products, services and topics. They have been been used successfully in applications like marketing assessments of new products, predicting fights between rival football supporters and discovery of voting trends in election campaigns.

13.1.4.2 Web Mining

Another common text mining application is the analysis of texts from web pages, a field known as web mining. A very large collection of texts can be found in web pages, from sources as distinct as logs and web sites for academic institutions and publications, electronic commerce, news agencies, newspapers and government.

However in contrast to plain texts, web pages are usually written in a special structure, defined by a markup language; for example, Hypertext Markup Language (HTML). Markup languages provide extra information beyond the text, such as audio files, images, videos, comments, metadata and hyperlinks to other web pages, all of which can be used for knowledge extraction. Thus, once the text has been extracted from a web page, text mining techniques can be applied to it, and possibly to the extra information as well.

However, this extra information can also be a burden. First, it can contain irrelevant information and even information that can harm rather than help the knowledge extraction process. Second, since web pages can have very different structures, it is not easy to automatically extract data from them.

13.2 Recommender Systems

Popular applications of data analytics include recommender systems (RSs). We often come across these in everyday life: Netflix recommends movies, Facebook recommends new friends, Youtube recommends videos to see, e-shops

recommend products to buy, Amazon recommends books, newspapers recommend articles to read, and so on.

Companies want to sell more goods, increase users' satisfaction and loyalty and also to better understand users' needs and interests. On the other hand, users want to find what they want with relatively little effort; this is extremely important because of the information overload we face while searching and browsing. Employing recommendation techniques (RTs) to web catalogs, e-shops, social networks and other applications is a reasonable step to achieve these goals of both providers and users.

The RT field emerged from that of information retrieval and machine learning but despite there being common ground, there are also some differences. In information retrieval, data are unstructured and cover various topics, but RTs are focused on repositories concerning a single topic (such as movies). Moreover, while machine learning deals with easily measurable and objective evaluation measures (say, the mean squared error), it is not so straightforward to measure the quality of a recommendation (user satisfaction). However, information retrieval and machine learning techniques are commonly used and adapted for developing RTs.

13.2.1 Feedback

The basic concept of RT, besides the user and the item, is feedback, but let us talk briefly about the former two concepts first.

Users can be defined by their attributes or characteristics, such as age, income, marital status, education, profession or nationality, but also by their preferred sport, hobbies, or favorite movies. This information is often obtained by means of questionnaires, but because of the sensitivity of such information, user attributes are usually hard to obtain.

On the other side of the coin, the items are also characterized by their attributes, such as, for movies, the title, genre, year, director, actors, budget or award nominations. In contrast to user attributes, this information is not sensitive, but might sometimes be costly to obtain. For example, if we need to derive them from textual descriptions, someone has to input the data.

When the user interacts with an item, it can be viewed as some kind of feedback about the user's interest in the given item. Depending on the way the feedback was obtained, we distinguish

- *explicit feedback:* when the user is asked by the system to express a preference on items directly, say by rating them on a scale or ranking them;
- *implicit feedback:* when information about a user's preference is obtained by watching their interaction with the system, say by recording which items were viewed, listened to, scrolled past, bookmarked, saved, purchased, linked to, copied, and so on.

Explicit feedback is more precise, but puts a cognitive burden on users. Just imagine that someone asked you to rate your teachers on a scale; it would probably take some time for you to decide in some cases. On the other hand, implicit feedback does not burden users but is not so precise. For example, if you signed up for a lecture course with a certain teacher it does not necessarily mean that you like that teacher.

13.2.2 Recommendation Tasks

Before we start with the definition of recommendation tasks, let us introduce some notations. Let $u \in \{u_1, u_2, \ldots, u_n\}$ be a specific user from the set of n users. Similarly, $i \in \{i_1, i_2, \ldots, i_m\}$ will correspond to a specific item from the set of m items. The recorded feedback of the user u on item i will be denoted as r_{ui}, corresponding to a certain rating or ranking.

The task of recommendation is, given the set of users and items as well the recorded feedback, to induce a model $\hat{f}$ that predicts for each user–item pair (u, i) a value $\hat{r}_{ui}$ representing the rating of the user u for the item i. Does that sound familiar? Probably yes, since it looks like a prediction task, as introduced in Chapter 8.

Depending on the type of feedback, we distinguish two recommendation tasks:

- rating prediction where the feedback is explicit; that is, r_{ui} are numbers representing ratings of users for items, and $\hat{f}(u, i) = \hat{r}_{ui}$ expresses the predicted rating of the user u for an item i;
- item recommendation where the feedback is implicit; that is, r_{ui} are zeros and ones representing the absence or presence, respectively, of an interaction between users and items, and $\hat{f}(u, i) = \hat{r}_{ui}$ expresses the predicted likelihood of a "positive" implicit feedback (ranking score) of the user u for an item i.

Example 13.5 Let's collect the preferences of our friends for some movies, so we can recommend them something new to watch. The scenario of item recommendation is illustrated in Table 13.6, in which we have collected information about which of five movies were seen by four of our friends. A value of 1 in a cell corresponds to positive implicit feedback. For example Eve saw *Titanic* but didn't see *Forrest Gump*. The recommendation task, in this case, would be to learn the likelihood of positive feedback from users on movies they have not yet seen, for example, the likelihood of positive feedback from James for *Titanic* and *Forrest Gump*. The movie with highest predicted likelihood would then be the one that James would be most likely to enjoy. Note, however, that for implicit feedback only positive feedback is recorded. This makes sense, since the fact that a user has watched a movie might indicate that she was sufficiently interested in that movie to watch it and therefore, in some way, preferred it to other movies. But positive feedback does not necessarily mean that the user

Table 13.6 Item recommendation scenario.

	Titanic	Pulp Fiction	Iron Man	Forrest Gump	The Mummy
Eve	1	1	1		1
Fred	1	1		1	1
Irene	1	1	1	1	
James		1	1		1

Table 13.7 Rating prediction scenario.

	Titanic	Pulp Fiction	Iron Man	Forrest Gump	The Mummy
Eve	1	4	5		3
Fred	5	1		5	2
Irene	4	1	2	5	
James		3	4		4

also *liked* the given item. These assumptions are, however, fuzzy. This is a reason that implicit feedback cannot be considered precise.

More precise feedback is obtained when we ask users to rate items, as illustrated in Table 13.7 where the numbers in the cells correspond to the ratings (number of stars) assigned to the movies by our friends. The task here is to predict the ratings of users for movies they have not yet seen. For example, what rating would James give to *Titanic* and *Forrest Gump*? The highest predicted rating would indicate the movie that James would be expected to like more than any other.

13.2.3 Recommendation Techniques

We distinguish three main types of RS, and their hybrid combinations. The use of any particular one depends on the domain and the data available; that is, what information about users and items is available or if feedback is considered. These three types are described in the following subsections.

13.2.3.1 Knowledge-based Techniques

In knowledge-based RTs, recommendations are based on knowledge of users' needs and preferences. Items' attributes (say the price and the type of a car, the number of airbags, the trunk size, and so on), users' requirements (say "the maximum acceptable price of a car is $8,000" and "the car should be safe and suitable for a family") and the domain knowledge describing some dependencies

between user requirements and item properties (say, "a family car should have big trunk") or between user requirements (say, "if a safe family car is required, the maximum acceptable price must be more than $2,000") are utilized.

In these types of RS, the recommendation process is interactive; the user iteratively specifies her requirements according to the items recommended in the given state of the "conversation" with the system. Recommendations are derived by identifying those products in the catalog that match the user's requirements, and ranking them according to how well they match those requirements.

The drawback of knowledge-based RTs is the high cost of preparing the underlying knowledge base; this is domain dependent. For each domain, a specific knowledge base, and thus a domain expert, is needed, making these types of RS inflexible and therefore less popular.

13.2.3.2 Content-based Techniques

In content-based RTs, the attributes of items and some feedback from users is necessary. Users' interests are learned by supervised machine learning techniques. In other words, a model of the feedback (the target attribute) of a given user is learned from the attributes (the explanatory variables) of items rated or ranked by the user in the past. Using the derived predictive model, ratings or rankings for items not yet seen by the user are predicted. The list of recommended items is then prepared, based on these predicted ratings or rankings. For each user, a separate predictive model, a classifier or a regressor, is learned.

The advantage of these types of RT is that user attributes are not needed. On the other hand, the predictive model is usually learned from small number of instances, particularly for new users. Also, it might be that the user only rates specific movies; say, only a specific genre or movies featuring specific actors. Thus, for these but also other reasons, the learned predictive model is sensitive to overfitting and might narrow the recommendation to a specific space of items.

Example 13.6 The training data to learn the predictive, content-based, model of Eve from Table 13.7 are illustrated in the upper part of Table 13.8. The regression tree model learned from these data is shown in Figure 13.2. The predicted rating of Eve for the movie *Forrest Gump* is 1 and is shown in bold in the bottom part of Table 13.8.

13.2.3.3 Collaborative Filtering Techniques

Collaborative filtering is the most popular RT. It recognizes similarities between users according to their feedback and recommends items preferred by like-minded users. Collaborative filtering techniques can provide good results even in cases when no user or item attributes are available. We distinguish two main types of collaborative filtering: neighborhood-based and model-based techniques.

Data for a content-based RT for the user Eve from Table 13.7.

	Genre$_1$	Genre$_2$	Year	Country	Length (min)	Director	Actor$_1$	Actor$_2$	Rating
	Drama	Romance	1997	USA	194	J. Cameron	L. DiCaprio	K. Winslet	1
Pulp Fiction	Drama	Crime	1994	USA	154	Q. Tarantino	J. Travolta	U. Thurman	4
Iron Man	Action	Adventure	2008	USA	126	J. Favreau	R. Downey Jr.	G. Paltrow	5
The Mummy	Fantasy	Adventure	1999	USA	125	S. Sommers	B. Fraser	R. Weisz	3
Forrest Gump	**Drama**	**Romance**	**1994**	**USA**	**142**	**R. Zemeckis**	**T. Hanks**	**R. Wright**	**1**

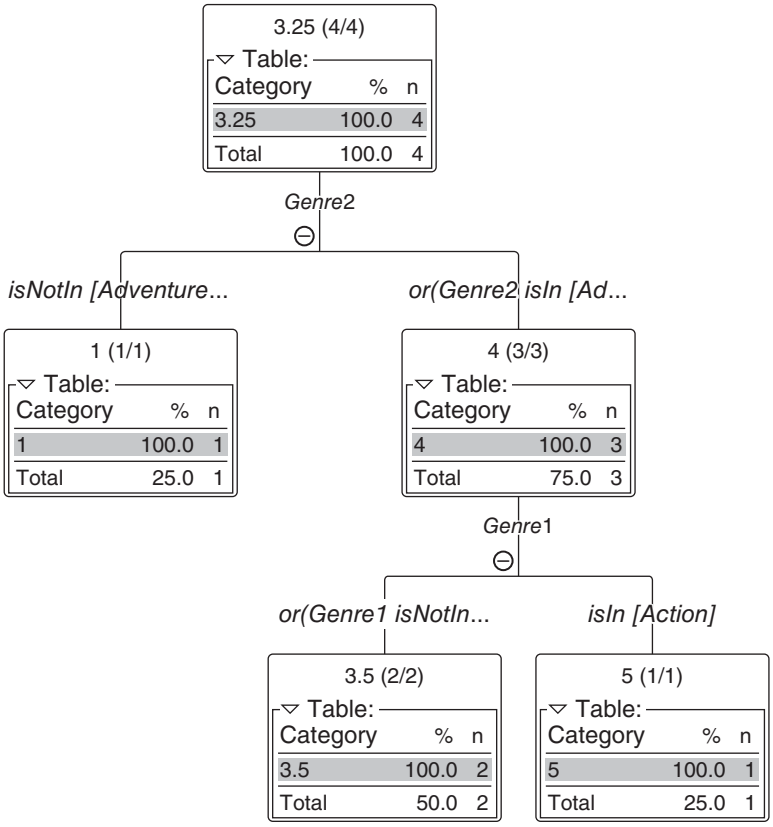


Figure 13.2 Regression tree model learned from the training data in Table 13.8.

Neighborhood-based techniques utilize vector similarity measures to compute similarities between users or items. We distinguish user-based and item-based collaborative filtering techniques.

User-based collaborative filtering In user-based techniques, each user is represented by a vector of feedbacks. In our example, from Tables 13.6 or 13.7 James would be represented by vectors $(?, 1, 1, ?, 1)$ or $(?, 3, 4, ?, 4)$, respectively, corresponding to the ratings he gave the five movies in the catalog. Note that the user vectors are usually very sparse, given that users provide feedback only on a small number of items compared to the total number of items in the catalog.

To predict the feedback of user u for item i, the first step is to get the feedback of the k users who are most similar to user u, with respect to their feedbacks, and have also provided some feedback on item i. We denote the set of these k-nearest neighbors as $\mathcal{N}_i^{u,k}$.

In case of item recommendation – that is, the example from Table 13.6 – the predicted likelihood of positive feedback of the user u on the item i is computed as an average similarity of each user v from $\mathcal{N}_i^{u,k}$ to user u

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i^{u,k}} sim(u, v)}{k} \tag{13.1}$$

where $sim(u, v)$ is some vector similarity measure, say a cosine vector similarity. Any other distance measure discussed in Chapter 5 can be used.

Cosine Vector Similarity The cosine vector similarity measure is a popular vector similarity measure that is used when the vectors to compare are sparse. To compute the similarity of sparse vectors, the missing values in both vectors are substituted with 0. Having two vectors $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{y} = (y_1, \ldots, y_m)$, the cosine vector similarity is computed as

$$sim^{cv}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{m} x_i y_i}{\left(\sum_{i=1}^{m} x_i^2 \sum_{i=1}^{m} y_i^2\right)^{\frac{1}{2}}} \tag{13.2}$$

Cosine vector similarity is a popular measure in text mining and RSs.

Example 13.7 The computed cosine vector similarities between the users in Table 13.6 are shown in Table 13.9. Based on these data, according to Equation (13.1), the predicted likelihoods of positive feedback of James for *Titanic* and *Forrest Gump* are computed as follows:

$$\mathcal{N}_{Titanic}^{James,2} = \{Eve, Fred\} \text{and } \mathcal{N}_{ForrestGump}^{James,2} = \{Fred, Irene\}$$

then

$$\begin{aligned}
\hat{r}_{James\ Titanic} &= \frac{sim^{cv}(James, Eve) + sim^{cv}(James, Fred)}{2} \\
&= \frac{0.87 + 0.58}{2} \\
&= 0.725 \\
\hat{r}_{James\ ForrestGump} &= \frac{sim^{cv}(James, Fred) + sim^{cv}(James, Irene)}{2} \\
&= \frac{0.58 + 0.58}{2} \\
&= 0.58
\end{aligned}$$

Thus, James will be more likely to prefer the *Titanic* than *Forrest Gump*.

In rating prediction, we must be aware of the phenomenon called bias. Users' ratings are usually biased, meaning that some users, when they are providing

Table 13.9 Cosine vector similarities between the users from Table 13.6.

$sim^{cv}(u,v)$	Eve	Fred	Irene	James
Eve	1.0	0.75	0.75	0.87
Fred		1.0	0.75	0.58
Irene			1.0	0.58
James				1.0

Since the similarity values are symmetrical, the values below the diagonal would mirror the values above the diagonal.

ratings, are more pessimistic while other are more optimistic than the average. Thus, bias should be taken into consideration when computing the similarity of users. A good choice would be to utilize some correlation measures for computing the similarity between the feedbacks of two users. An example would be the Pearson correlation, introduced in Chapter 2, and denoted here as sim^{pc}. Since the feedbacks are not only 0 (in case of no feedback) or 1, as in the case of item recommendation, but numbers (see Table 13.7), the model predicting the rating of user u for item i is

$$\hat{r}_{ui} = \overline{r}_u + \frac{\sum_{v \in \mathcal{N}_i^{u,k}} sim(u,v) \cdot (r_{vi} - \overline{r}_v)}{\sum_{v \in \mathcal{N}_i^{u,k}} |sim(u,v)|} \tag{13.3}$$

where $\overline{r}_u$ and $\overline{r}_v$ are the average ratings of users u and v (computed from the training data), respectively, and $sim(u,v)$ is a similarity measure, such as the Pearson correlation $sim^{pc}(u,v)$.

Example 13.8 The computed Pearson correlation similarities between the users from Table 13.7 are shown in Table 13.10. Based on these data, according to Equation (13.3), the predicted likelihoods of positive feedback of James on the item Titanic is computed as follows (J, I and F are abbreviations for James, Irene and Fred, respectively): $\mathcal{N}_{Titanic}^{J,2} = \{I, F\}$ and $\overline{r}_J = \frac{3+4+4}{3} = 3.67$, $\overline{r}_I = \frac{4+1+2+5}{4} = 3$, $\overline{r}_F = \frac{5+1+5+2}{4} = 3.25$ $\hat{r}_{J\ Titanic} = \overline{r}_J + \frac{sim^{pc}(J,I)\cdot(r_{I\ Titanic}-\overline{r}_I)+sim^{pc}(J,F)\cdot(r_{F\ Titanic}-\overline{r}_F)}{|sim^{pc}(J,I)|+|sim^{pc}(J,F)|} = 3.67 + \frac{0.6\cdot(4-3)+0.565\cdot(5-3.25)}{0.6+0.565} = 1.36$ what is the predicted rating of James for the movie Titanic.

Item-based Collaborative Filtering Similar to user-based collaborative filtering, there are also item-based techniques. The difference is that instead of considering similar users we consider similar items. Thus, the vector similarity measures will be computed from the columns of the user–item matrix (Tables 13.6 and 13.7). Also, the bias of items is considered, reflecting their popularity amongst the users (some movies are blockbusters and popular while some are received negatively by the audience).

Table 13.10 Pearson correlation similarities between users in Table 13.7.

$sim^{PC}(u, v)$	Eve	Fred	Irene	James
Eve	1.0	−0.716	−0.762	−0.005
Fred	–	1.0	0.972	0.565
Irene	–	–	1.0	0.6
James	–	–	–	1.0

Since the similarity values are symmetrical, the values below the diagonal would mirror the values above the diagonal.

Model-based Collaborative Filtering The basic idea of model-based collaborative filtering techniques is to map the users and the items into a common latent space. The dimensions of this space, often called factors, represent some implicit properties of items and users' interests in these implicit properties. We introduced some dimension reduction techniques, such as principal component analysis, in Chapter 4, and these could be used for model-based collaborative filtering. However, we will introduce an other, very simple technique called low-rank matrix factorization. We will illustrate the main ideas of matrix factorization in a rating prediction example, but there are factorization models for item recommendation too.

For the input, there is the user–item rating matrix denoted as R; that is, Table 13.7, with n rows (number of users) and m columns (number of items). Only some of the cells are non-empty (recorded feedbacks). The non-empty cells are our training data, while the empty cells will be filled by future feedback. The goal is to fill the empty cells of this matrix with numbers that are as close to users' future feedbacks as possible. In other words, we should find a good regression model with a good bias–variance trade-off such that the error in the data is minimal.

Now imagine two matrices W and H with the following dimensions: W has n rows and k columns while H has m columns and k rows. The uth row $\mathbf{w}_u$ of W would correspond to a vector representing the user u in some k-dimensional latent space. Similarly, the ith column $\mathbf{h}_i$ of H would correspond to a vector representing the item i in the same k-dimensional space.

Multiplying[1] the two matrices W and H would result in a matrix $\hat{R} = W \cdot H$ with the same dimensions as of the rating matrix R. Now the goal is to find the matrices W and H such that the error

1 Multiplying a matrix W of size n×k with a matrix H of size k×m results in a matrix $\hat{R}$ of size n×m. The cell $\hat{r}_{ui}$ in the uth row and ith column of $\hat{R}$ is computed as the dot product of the uth row $\mathbf{w}_u$ of W and ith column $\mathbf{h}_i$ of H, i.e. $\mathbf{w}_u \cdot \mathbf{h}_i = \sum_{j=1}^{k} (w_{u_j} \times h_{i_j})$.

$$error(R, \hat{R}) = \sum_{r_{ui} \in R} (r_{ui} - \mathbf{w}_u \cdot \mathbf{h}_i)^2 \tag{13.4}$$

is minimized. Here $\hat{r}_{ui} = \mathbf{w}_u \cdot \mathbf{h}_i$ is the predicted rating of the user u for the item i. This, however, corresponds to a linear model introduced in Chapter 8 with the parameters W and H that could result from the minimization of the following objective function:

$$\underset{W,H}{\operatorname{argmin}} \sum_{r_{ui} \in R} (r_{ui} - \mathbf{w}_u \cdot \mathbf{h}_i)^2 + \lambda(\| W\|^2 + \| H\|^2) \tag{13.5}$$

Does this seem familiar? Look at Equations 8.8, 8.10, 8.13 and 8.14 to see the similarities.

Example 13.9 For $k = 2$, the factorization of the matrix R (Table 13.7) under certain settings[2] results in two matrices:

$W=$

1.1995242	1.1637173
1.8714619	−0.02266505
2.3267753	0.27602595
2.033842	0.539499

the rows of which correspond to latent representations of the users Eve, Fred, Irene and James, respectively, and

$H=$

1.6261001	1.1259034	2.131041	2.2285593	1.6074764
−0.40649664	0.7055319	1.0405376	0.39400166	0.49699315

the columns of which correspond to latent representations of the items Titanic, Pulp Fiction, Iron Man, Forrest Gump and The Mummy, respectively. Multiplying these matrices we get

$\hat{R} =$

1.477499	2.171588	3.767126	3.131717	2.506566
3.052397	2.091094	3.964578	4.161733	2.997066
3.671365	2.814469	5.245668	5.294111	3.877419
3.087926	2.670543	4.895569	4.745101	3.537480

from which we can see that the predicted rating of James for the movies Titanic and Forrest Gump are 3.09 and 4.75, respectively.

Graphical representations of user and item factors – that is, the k-dimensional representation of users and items corresponding to rows and columns of W and H, respectively – are shown in Figure 13.3. The closer the users are to each other, more similar their interest should be. In addition, the closer a user is to an item, the more likely she will prefer it. Note that the

2 The factorization technique used (stochastic gradient descent) has its own hyper-parameters, the discussion of which is out of the scope of this chapter.



Figure 13.3 Representation of users and items in a common, latent two-dimensional space from Example 13.9.

example provided here was created without carefully tuning the settings of the factorization algorithm and thus the results might be not precise, although they are adequate for illustration purposes.

13.2.4 Final Remarks

There are several important issues one has to keep in mind when developing and implementing an RS, some of which we will outline in this section.

Before integrating the developed RT into a live system like an e-shop, it is advised to do an off-line evaluation on simulated user-behavior data. Such an experiment is low cost, takes little time and can done on a large scale, although it answers only a few questions, such as determining the predictive power of the developed technique and the running time.

The other, more expensive and time consuming, step in evaluating an RS involves user studies. In contrast to off-line evaluations, interactions with real users of the system are observed and analyzed by means of questionnaires: say,

how did they like the delivered recommendations and whether they were satisfied. User studies are usually small scale and expensive because we will need to reward the test subjects somehow.

The final phase in the evaluation of an RS is on-line evaluation. This is done in the following way: a small part of the traffic in the system is redirected to the developed recommendation technique and users' behavior is observed: whether they change ratings to better grades, whether they stay in the system for longer periods, and so on. This can be, however, risky, because if the customers are not satisfied with the result of the new recommendation technique we might loose some of them. Thus, it is good to run on-line testing after off-line testing and only if the user studies show promising results.

There are various properties an RS should have, which can be tested and evaluated in on-line and off-line experiments as well in user studies. Besides scalability, robustness and predictive accuracy, the RS should have good coverage: it should be able to recommend a large proportion of items for a large proportion of users. In other words, it should not only work for a small portion of items and users. An other issue is the novelty of recommendations. This concerns whether the system recommends items to a user that they would not find on their own. This idea is connected to the serendipity property: how surprising the recommendations for a user are. For example, the newest movie with the user's favorite actor might be a novelty but is not surprising since they would eventually find out about it on their own. In addition, it is worth thinking about the diversity of recommendations, which measures how "colorful" the palette of recommended items are. Probably we would not be very satisfied if the recommendation was restricted to only a specific genre of movies or movies with a particular actor.

The so-called cold-start problem arises when a new user or new item appears in the system; there is no (or not enough) feedback recorded. In this case, the easiest solution is to recommend the most popular items to the user. If we have some additional information about the user available, we might utilize some generic knowledge base: in other words, using knowledge-based recommendation. Also, additional information about items can be utilized. There have been many attempts to mitigate the cold-start problem, but their detailed description is out of the scope of this chapter.

A new direction in RS research concerns context-based and group recommendation. In the former case, the recommendation technique considers so called "context": additional information besides the user and item attributes that can be relevant for recommendation. For example, people watch "Christmas movies" during the Christmas holidays, or they might watch different movies at weekends than on weekdays. Group recommendation concerns people with whom the user is involved when requesting recommendations. For example, we might like to watch different movies when we are alone, with our partner, with our children or with our friends. Detecting the context of the

user is not easy, and usually some readily measurable context is considered: the time, the location or the weather.

Finally, we have to notice that a very important "partner" of any RT is a good user interface. Even the results of the best recommendation techniques can be spoiled in the user experience when delivered through a bad user interface. In addition, we have to keep in mind that each domain comes with its specialties which, if incorporated into an RS, can contribute to the uniqueness of the developed system. More information and deeper discussion of RSs and their applications can be found in the literature [56].

13.3 Social Network Analysis

Social network analysis (SNA) has gained importance in recent years due to the popularity of social networks. SNA has its roots in sociology, but the study of networks has now extended far from the social sciences. Networks of various characteristics have been studied in physics, neuroscience, economics, computer science and engineering as well, just to name a few areas in which relationships between entities play important roles.

Since a complete description of all the existing methods extends out of the scope of this book, only a few foundational concepts will be introduced here, considering just simple networks.

13.3.1 Representing Social Networks

Each network is a type of a graph, consisting of nodes and relations, also called edges, between the nodes. Edges can be directed or undirected, and can be assigned with a weight, as illustrated in Figure 13.4. If there are multiple types of relation between nodes, for example if two people can be connected by a "friend", "family" or "colleague" relationship, we call these edges "multiplex".

For representing a directed or undirected graph, a suitable structure is a so-called adjacency matrix, denoted here as A, whose rows and columns represent nodes. Each cell A_{ij} in ith row and jth column indicates if there is an edge from the ith node to the jth node. For undirected edges, A is symmetrical with respect to its diagonal.

Example 13.10 An example social network, containing as nodes our friends from the example introduced in Table 1.1, is illustrated in Figure 13.5. Edges, for simplicity, are undirected and unweighted and represent, say, which pairs of friends have already had dinner together.

As we can see, there are three different parts of the network, corresponding to different network types. The nodes A, B, C and D form a kind of a star according to which this sub-network is centralized, with the node D being in the center.

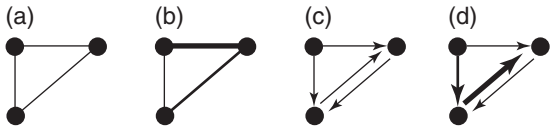


Figure 13.4 Undirected (a), weighted (b) where the thickness of an edge is proportional to its weight, directed (c), and, directed and weighted (d) edge types in networks.

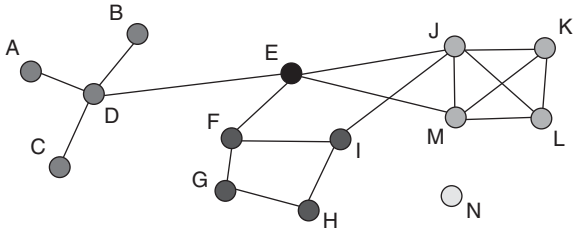


Figure 13.5 An example social network. The nodes correspond to our friends from the example in Table 1.1; each person is indicated by their initial.

Considering only the nodes F, G, H and I, a kind of a ring is formed, where each node is connected to its two neighbors. The nodes J, K, L and M form a complete, dense network such that each node is connected to each other. Finally, there is the node N not (yet) connected to any other node in the network.

The network, or graph, in Figure 13.5 can be represented in an adjacency matrix, as shown in Table 13.11. Note, that for sparse networks there can be other ways of representing the adjacency matrix, for example the sparse form introduced in Chapter 6, where a boolean matrix from Table 6.1 was represented in sparse form in Table 6.2.

Representing a graph in an adjacency matrix A has advantages: we can employ efficient and fast matrix operations to get useful information about the graph. For example, if we multiply A by itself – that is, raise A to the second power – will show us the number of paths, or sequence of edges, between pairs of nodes that are of length two.

Example 13.11 The adjacency matrix from Table 13.11 squared is shown in Table 13.12. From this matrix we can see that there is one path of length two between nodes A and B, through node D; in other words, the sequence of edges between nodes A and D and nodes D and B. This sequence is of length two. Also, there is one path of length two from A to D and back from D to A, as can be read from the cell in the first row and first column of Table 13.12.

Similarly, if we take the third power of A, we get the number of paths between pairs of nodes that are of length three. The fourth power of A will show the number of paths between pairs of nodes that are of length four, and so on.

Table 13.11 The adjacency matrix for the network in Figure 13.5.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A	0	0	0	1	0	0	0	0	0	0	0	0	0	0
B	0	0	0	1	0	0	0	0	0	0	0	0	0	0
C	0	0	0	1	0	0	0	0	0	0	0	0	0	0
D	1	1	1	0	1	0	0	0	0	0	0	0	0	0
E	0	0	0	1	0	1	0	0	0	1	0	0	1	0
F	0	0	0	0	1	0	1	0	1	0	0	0	0	0
G	0	0	0	0	0	1	0	1	0	0	0	0	0	0
H	0	0	0	0	0	0	1	0	1	0	0	0	0	0
I	0	0	0	0	0	1	0	1	0	1	0	0	0	0
J	0	0	0	0	1	0	0	0	1	0	1	1	1	0
K	0	0	0	0	0	0	0	0	0	1	0	1	1	0
L	0	0	0	0	0	0	0	0	0	1	1	0	1	0
M	0	0	0	0	1	0	0	0	0	1	1	1	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13.12 The adjacency matrix from the Table 13.11 squared showing the counts of paths of length two between pairs of nodes.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A	1	1	1	0	1	0	0	0	0	0	0	0	0	0
B	1	1	1	0	1	0	0	0	0	0	0	0	0	0
C	1	1	1	0	1	0	0	0	0	0	0	0	0	0
D	0	0	0	4	0	1	0	0	0	1	0	0	1	0
E	1	1	1	0	4	0	1	0	2	1	2	2	1	0
F	0	0	0	1	0	3	0	2	0	2	0	0	1	0
G	0	0	0	0	1	0	2	0	2	0	0	0	0	0
H	0	0	0	0	0	2	0	2	0	1	0	0	0	0
I	0	0	0	0	2	0	2	0	3	0	1	1	1	0
J	0	0	0	1	1	2	0	1	0	5	2	2	3	0
K	0	0	0	0	2	0	0	0	1	2	3	2	2	0
L	0	0	0	0	2	0	0	0	1	2	2	3	2	0
M	0	0	0	1	1	1	0	0	1	3	2	2	4	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13.13 Basic properties of nodes from the network in Figure 13.5.

Node	A	B	C	D	E	F	G
Degree	1	1	1	4	4	3	2
Closeness	0.0196	0.0196	0.0196	0.025	0.0286	0.025	0.0204
Betweenness	0	0	0	30	37.17	14.5	2.17
Clust_coef	–	–	–	0	0.17	0	0
Node	**H**	**I**	**J**	**K**	**L**	**M**	**N**
Degree	2	3	5	3	3	4	0
Closeness	0.0196	0.0238	0.0270	0.0217	0.0217	0.0256	0.0054
Betweenness	1.33	10.67	18	0	0	6.17	0
Clust_coef	0	0	0.4	1	1	0.67	–

–, not defined.

13.3.2 Basic Properties of Nodes

The basic properties of a network are derived from the relevant properties of its nodes. This will be the main focus of this section. Since a network is determined by its nodes and the connections between them, the basic properties of nodes are concerned with connections. The number and the structure of a node's connections determine its influence or, in other words, its power in the network.

13.3.2.1 Degree

This, the most basic measure, captures the number of connections of the node. For a network with undirected edges, the degree of a node is the sum of the corresponding row or the corresponding column in the adjacency matrix. The degrees of nodes from the network in Figure 13.5 are shown in Table 13.13.

For directed edges, we distinguish so-called "in-degree" and "out-degree" scores. The in-degree score of a node captures the number of nodes from which there is an edge pointing to the given node. The out-degree of a node is the number of nodes to which there is an edge pointing from the given node. The out-degrees or in-degrees of the nodes of a network can be acquired by summing the corresponding rows or columns, respectively, of the adjacency matrix.

13.3.2.2 Distance

The distances between nodes are important characteristic since they determine the way information diffuses in the network. The distance between two nodes is computed as the minimum number of edges the information has to take from one node to the other. As is illustrated in the distance matrix

Table 13.14 The distance matrix – distances between nodes – for the graph in Figure 13.5.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A	0	2	2	1	2	3	4	5	4	3	4	4	3	∞
B	2	0	2	1	2	3	4	5	4	3	4	4	3	∞
C	2	2	0	1	2	3	4	5	4	3	4	4	3	∞
D	1	1	1	0	1	2	3	4	3	2	3	3	2	∞
E	2	2	2	1	0	1	2	3	2	1	2	2	1	∞
F	3	3	3	2	1	0	1	2	1	2	3	3	2	∞
G	4	4	4	3	2	1	0	1	2	3	4	4	3	∞
H	5	5	5	4	3	2	1	0	1	2	3	3	3	∞
I	4	4	4	3	2	1	2	1	0	1	2	2	2	∞
J	3	3	3	2	1	2	3	2	1	0	1	1	1	∞
K	4	4	4	3	2	3	4	3	2	1	0	1	1	∞
L	4	4	4	3	2	3	4	3	2	1	1	0	1	∞
M	3	3	3	2	1	2	3	3	2	1	1	1	0	∞
N	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

in Table 13.14, if there is no connection between two nodes, the computed distance is infinite. Note that for a graph with undirected edges, the distance matrix is symmetric with respect to its diagonal, but this does not hold for graphs with directed edges.

13.3.2.3 Closeness

This measure reflects how accessible a node is in the network. Larger values indicate that the given node is well connected to the other nodes of the network. For a given node v, its closeness measure

$$closeness(v) = \frac{1}{\sum_{u \neq v} distance(u, v)} \tag{13.6}$$

is computed (from the distance matrix) as 1 divided by the sum of distances between the given node v and all the other nodes $u \neq v$ in the network. If there is no connection between two nodes, instead of an infinite value, the number of nodes in the network is substituted into the computation. Closeness is sensitive to and decreases with the size of the network.

Example 13.12 The closeness value of node A from Figure 13.5, based on the distances introduced in the first row of Table 13.14, is computed as $1/(2 + 2 + 1 + 2 + 3 + 4 + 5 + 4 + 3 + 4 + 4 + 3 + 14) = 0.019607843$. Here, instead of $distance(A, N) = \infty$, $distance(A, N) = 14$ is used (the network has 14 nodes).

13.3.2.4 Betweenness

This measure is used to assess how important the position of a node v in the network is. It is computed as:

$$betweenness(v) = \sum_{u \neq v \neq t} \frac{nsp_v(u,t)}{nsp(u,t)} \tag{13.7}$$

where u and t are pairs of nodes different from v, $nsp(u,t)$ is the number of shortest paths from node u to node t and $nsp_v(u,t)$ is the number of shortest paths from u to t that go through node v. Betweenness measures the degree to which information has to flow through a particular node in the network.

Example 13.13 Since the computation of betweenness scales with the number of nodes in the network, for illustration purposes we will introduce the computation only on the part of our social network shown in Figure 13.6.

The betweenness of node E is zero since none of the shortest paths between any other pairs of nodes cross through E. Now, let us count the betweenness for node G: $nsp(F,H) = 2$ because there are two paths both with distance 2 (both are shortest) between nodes F and H, namely the path F→G→H and the path F→I→H. However, only one of these passes through node G, so $nsp_G(F,H) = 1$. Similarly, $nsp(E,H) = 2$ and $nsp_G(E,H) = 1$. Since there are no other shortest paths passing through node G, we can calculate its betweenness value according to Equation (13.7) as:

$$\begin{aligned}
&\frac{nsp_G(E,F)}{nsp(E,F)} + \frac{nsp_G(E,H)}{nsp(E,H)} + \frac{nsp_G(E,I)}{nsp(E,I)} \\
&\quad + \frac{nsp_G(F,H)}{nsp(F,H)} + \frac{nsp_G(F,I)}{nsp(F,I)} + \frac{nsp_G(H,I)}{nsp(H,I)} \\
&\quad = \frac{0}{1} + \frac{1}{2} + \frac{0}{1} + \frac{1}{2} + \frac{0}{1} + \frac{0}{1} \\
&\quad = 1
\end{aligned}$$

Similarly, the betweenness of the other nodes can be computed, resulting in $betweenness(F) = 3.5$, $betweenness(H) = 0.5$, and $betweenness(I) = 1$.

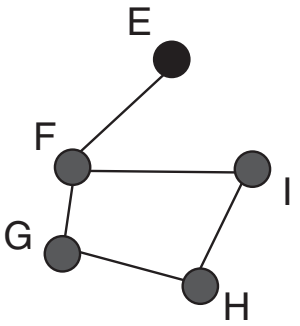


Figure 13.6 Part of a social network from Figure 13.5.

13.3.2.5 Clustering Coefficient

Some group research indicates that triads – three nodes connected to form a triangle – are important formations from which a wide range of interesting social relations can be derived. The clustering coefficient measures the tendency of a node v to be included in a triad and can be defined as

$$clust_coef(v) = \frac{\sum_{u \neq v \neq t} triangle(u, v, t)}{\sum_{u \neq v \neq t} triple(u, v, t)} \tag{13.8}$$

where $triangle(u, v, t) = 1$ if the nodes u, v and t are connected to form a triangle and $triangle(u, v, t) = 0$ otherwise. In addition, $triple(u, v, t) = 1$ if the nodes u and t are both connected to the node v, otherwise, $triple(u, v, t) = 0$.

If $degree(v) < 2$, the clustering coefficient is either equal to zero or not defined.

Example 13.14 The clustering coefficient of node E of the network in Figure 13.5 is computed as follows: four nodes are connected to E, forming six triples in total, such that the values of $triple(D, E, F)$, $triple(D, E, M)$, $triple(D, E, J)$, $triple(F, E, M)$, $triple(F, E, J)$ and $triple(M, E, J)$ are equal to 1 adding up to six. However, there is only one triangle formed, such that the value of $triangle(M, E, J)$ is equal to 1. Thus, $clust_coef(E) = 1/6 = 0.17$.

13.3.3 Basic and Structural Properties of Networks

The above mentioned properties (see Table 13.13), also called "node centrality scores", are related to individual nodes of a network and characterize their "power" or "position" in the network. However, there are also some basic and structural properties focusing on the whole network or some of its interesting parts. These will be discussed below.

13.3.3.1 Diameter

The diameter of a network is defined as the longest of all the distances between its nodes. This measure indicates how easily the nodes of a network are reachable. The diameter of the network from Figure 13.5 is the longest distance present in the distance matrix, which is equal to 5: the distance between nodes A and H.

13.3.3.2 Centralization

As shown in Table 13.13, centrality scores are uneven for the nodes of the graph. To measure this unevenness, network-level centrality scores for network N with n nodes can be computed as

$$C(N) = \sum_{v} (\max_{u} c(u) - c(v)) \tag{13.9}$$

where $\max_u c(u)$ is the maximum centrality score from all nodes u of the network (including node v), $c(v)$ is the centrality score of node v and c is either the degree, closeness or betweenness measure.

Example 13.15 Let us discuss the closeness centrality scores of the three networks illustrated by Figure 13.7.

The closeness scores of nodes A, B, C and D are 0.2, 0.2, 0.2 and 0.33, respectively, the maximum of which is 0.33. Thus the closeness centrality score of the network on the left-hand side is computed as $C^{closeness}(left) = (0.33 - 0.2) + (0.33 - 0.2) + (0.33 - 0.2) + (0.33 - 0.33) = 0.4$. For the other two networks, the closeness scores are equal for all their nodes; that is, 0.25 for nodes F, G, H and I, and 0.33 for the nodes J, K, L and M. Thus, $C^{closeness}(middle) = 4 \times (0.25 - 0.25) = 0 = 4 \times (0.33 - 0.33) = C^{closeness}(right)$ where "middle" and "right" correspond to the networks in the middle and on the right-hand side. From these results we can see that the left-hand network is more centralized than the other networks. Regarding the network from Figure 13.5, its degree centrality $C^{degree}(N) = 0.187$, its closeness centrality $C^{closeness}(N) = 0.202$ and its betweenness centrality $C^{betweenness}(N) = 0.395$.

What does a large centralization score for a network mean? In Example 13.15 above, we saw some extreme networks:

- the most centralized, the star
- the least centralized, the ring
- fully connected.

In a star network, one node is maximally central while all the others are minimally central nodes. On the other hand, in a ring network or fully connected network, all the nodes are equally central. Usually, network centrality measures are normalized onto the interval $[0, 1]$ so that the scores for two networks are easier to compare.

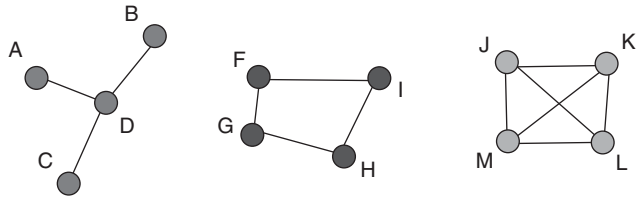


Figure 13.7 Example with three networks.

13.3.3.3 Cliques

A clique is a subset of nodes such that every two nodes in the subset are connected. Example of cliques of size three, containing three nodes, in the network from Figure 13.5 are the following subsets: $\{E, J, M\}$, $\{J, K, L\}$, $\{J, K, M\}$, $\{J, L, M\}$ and $\{K, L, M\}$. There is one clique of size four, namely the subset $\{J, K, L, M\}$.

13.3.3.4 Clustering Coefficient

This measure expresses the probability that the triples in the network are connected to form a triangle. It is computed, in a similar way to the clustering coefficient of nodes (see Subsection 13.3.2.5), as the ratio of the number of triangles to the number of connected triples in the network. The clustering coefficients of both the ring and star networks from the Example 13.15 above are equal to zero while the clustering coefficient of the fully connected network from the same example is equal to one. The clustering coefficient of the network from Figure 13.5 is 0.357.

13.3.3.5 Modularity

Modularity expresses the degree to which a network displays cluster structures (often called communities). A high modularity score of a network means that its nodes can be divided into groups such that nodes within these groups are densely connected while the connections between these groups are not dense. The modularity of the network in Figure 13.5 is 0.44.

13.3.4 Trends and Final Remarks

The social media landscape has greatly increased in recent years and several social network analytics tools have been developed within various disciplines of science and engineering. Special interest is devoted to the dynamics of social networks, in other words how the network evolves over time.

In this section, we focused on social network analysis rather than on mining. The reason is that, as discussed in Section 13.1 on text mining, only after understanding the basic principles, such as tokenization, stemming or the bag of words, are we able to extract knowledge from the text and, consequently, transform the text into structured form for use in clustering, pattern mining and prediction. Similarly, by understanding the basic properties of nodes and networks, we might be able to extract useful features that can be used in further mining using predictive or descriptive machine learning methods. The most common directions in social network mining are set out next.

With increasing use of social network sites such as Facebook or LinkedIn, there is a need for link prediction, a task closely related to the classification and regression techniques discussed earlier in this book. The aim of link prediction

is to predict which connections will be more likely to emerge between the nodes of the network. It is also related to the problem of inferring missing connections in a network.

An active field of study is the use of text mining techniques in the context of opinion and sentiment analysis. There are various applications, such as analyzing societal sentiment and opinion on actual events, or tracking the origins of fake news.

Visualization of social networks is another field of study to which lots of attention has been devoted recently. As has been shown, this is not an easy task since good visualization must make both clusters and outliers identifiable as well giving the ability to follow the links.

Other fields of research in social network analytics include community detection or mining interaction patterns, closely related to and utilizing the clustering and frequent pattern mining techniques discussed in Chapters 5 and 6 of this book.

For further study of social network mining, please refer to the on-line textbook by Hannemann and Riddle [57] or the more technical introduction by Zafarani *et al.* [58].

13.4 Exercises

1 Create a small dataset consisting of storylines (summaries) of 20 movies from some movie database, such that it includes ten science fiction movies and ten romantic comedies. Extract the features from these texts.

2 Use clustering techniques on the movie data and analyze the results.

3 Induce some classification model on the movie data (the training set) and assess its accuracy on a further five movies (the test set).

4 Ask 30 of your friends to rate some of the movies in the movie database created in the first exercise on a scale from 1 (very bad) to 5 (excellent) and compute the sparsity of the resulting matrix (of dimensions 30×20).

5 Use text mining techniques to develop a content-based model for recommending movies to three friends based on their ratings and the storylines of the rated movies.

6 Perform clustering of your friends based on their similarity of ratings.

7 Using k-nearest neighbor collaborative filtering to recommend movies to your three friends.

8 Create a social network of your friends from the exercise above, such that two friends are connected if their ratings for at least one movie differ at most by one. Create the adjacency matrix.

9 Compute the basic properties of nodes of the created network: degrees, the distance matrix, and the closeness, betweenness and clustering coefficients.

10 Compute the basic and structural properties of the network: diameter, centralization scores, cliques, clustering coefficient and modularity.

Appendix A

A Comprehensive Description of the CRISP-DM Methodology

In Section 1.7 we gave an overview of the CRISP-DM methodology. We will see now in more detail the tasks in each phase and the outputs. This appendix is necessary to better follow the projects in Chapters 7 and 12.

A.1 Business Understanding

This involves understanding the business domain, being able to define the problem from the business domain perspective, and being able to translate such a business problem into a data analytics problem. The business understanding phase has the following tasks and outputs:

1) *Determine business objectives:* the client–the person/institution that will pay you for the project or your boss if it is an internal project–certainly has a good understanding of the business and a clear idea of its objectives. The objective of this task is to understand this, uncovering important factors that can influence the final outcome. The outputs should be as follows:
   - The background: the situation about the business in the beginning of the project should be registered.
   - The business objectives: when a project on data analytics starts there is a motivation/objective behind it. The description of these objectives should be registered together with all related business details that seem relevant for the project.
   - The business success criteria: the success of a project should be as far as possible quantified. Sometimes it is not possible to do so, due to the subjective nature of the objective. In any case, the criteria/process to determine the business success of the project should be identified.
2) *Assess situation:* since an overview of the business was prepared in the previous task, it is now the time to detail the information about existing resources, constraints, assumptions, requirements, risks, contingencies, costs and benefits. The outputs are:

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

- Inventory of resources: the resources relevant for a project on data analytics are mainly human and computational. From computational resources, data repositories, such as databases or data warehouses, information systems, computers and other types of software are all meaningful computational resources.
- Requirements, assumptions, and constraints: there are typically requirements on the project calendar and on the results, and also legal and security requirements. During this kind of project it is usually necessary to make assumptions about, for instance, data availability in a certain date or expected changes in the business dependent on political measures, among others. All of these factors should be identified and registered. Constraints can also exist on data availability and usability, the kind of software that can be used, or the computational constraints for high-performance computing.
- Risks and contingencies: when a risk is identified, a contingency plan should be defined. A typical risk is a third-party dependency that can delay the project.
- Terminology: a glossary of terms for the business area and on the data analytics area.
- Costs and benefits: to list the expected costs and the expected benefits of the project, preferably in a quantified way.

3) *Determine data mining goals:* this is extensible to data analytics goals. The goal is to translate the problem from business to technical terms. For instance, if the business objective is 'to increase client loyalty', the data analytics object could be to 'predict churn clients'. The outputs are:
   - Data mining goals: describing how the data mining/analytics results are are able to help meet the business objectives. In the previous example, how does predicting churn clients help increase client loyalty?
   - Data mining success criteria: identifies the criteria for which the data mining/analytics result is considered successful. Using the same example, a possible success criterion would be to predict churn clients with an accuracy of at least 60%.
4) *Produce project plan:* to prepare the plan for the project. The outputs are:
   - Project plan: despite Dwight D. Eisenhower's famous quote, 'Plans are nothing; planning is everything', it is important to prepare a good initial plan. It should contain all tasks to be done, their duration, resources, inputs, outputs and dependencies. An example of a dependency is, for instance, that data preparation should be done before the modeling phase. This kind of dependency is often a cause of risks due to time delays. When there is evidence of risks, action recommendations should be written in the plan. The plan should describe the tasks of each phase up to the evaluation phase. At the end of each phase, a review of the plan should be scheduled. Indeed, Eisenhower was right!

- Initial assessment of tools and techniques: an initial selection of methods and tools should be prepared. The details of the next phases, especially the data preparation one, can depend on this choice.

A.2 Data Understanding

Data understanding involves the collection of the necessary data and its initial visualization/summarization in order to obtain the first insights about it, particularly, but not exclusively, about data quality problems such as missing values, outliers and other non-conformities.

The data understanding phase has the following tasks and respective outputs:

1) *Collect initial data:* data for an initial inspection should be collected from the project resources previously identified. Quite often, SQL queries are used to do this. When the data comes from multiple sources it is necessary to integrate them somehow. This can be quite costly. The output of this task is:
   - Initial data collection report: the identification of the data sources and all work necessary to collect it, including all technical aspects, such as any SQL queries used, or any steps taken to merge data from different sources.
2) *Describe data:* collection of basic information about the data. The output is:
   - Initial data collection report: the data usually comes in one or more data tables. For each table, the number of instances selected, the number of attributes and the data type of each attribute should be registered.
3) *Explore data:* Examination of the data, but using correct methods. Descriptive data analytics methods are adequate for this task (see Part II, but mainly Chapter 2). Interpretable models from Part III, such as decision trees, can also be a good option. The output is:
   - Data exploration report: where the relevant details discovered while exploring the data are reported. It may (and sometimes should) include plots in order to present visually details about the data that are important to report.
4) *Verify data quality:* the goal is to identify and quantify the existence of incomplete data when only part of the domain exits (for instance, data of only one faculty when the goal is to study data from the whole university), missing values, or errors in the data, such as a person with an age of 325 years. The output is:
   - Data quality report: where the results of verifying the data quality are reported. Not only should the problems found with the data be reported but also possible ways to solve them. Typically, the solutions for this kind of problems imply good knowledge both of the business subject and of data analytics.

A.3 Data Preparation

Data preparation: includes all tasks necessary in order to prepare the data set to be fed by the modeling tool. Data transformation, feature construction, outlier removal, missing values completion and incomplete instance removal are some of the most common tasks in the data preparation phase.

The data preparation phase has the following tasks and respective outputs:

1) *Select data:* based on their relevance to the project goals, the quality of the data and the existence of technical constraints such as on the data volume or data types, data is selected in terms of attributes and instances.
   - Rationale for inclusion/exclusion: where the rationale used to select the data is reported.
2) *Clean data:* The methods that are expected to be used in the modeling phase can imply specific preprocessing tasks. Typically, data subsets without missing values are selected, or techniques to fill missing values or remove outliers are applied. The output is:
   - Data cleaning report: describes how the problems identified in the data quality report of the data understanding phase were addressed. The impact of transformations made during the data cleaning task on the results in the modeling phase should be considered.
3) *Construct data:* construction of new attributes, new instances, or transformed values by converting, for instance, a Boolean attribute into a 0/1 numerical attribute. The output is:
   - Derived attributes: attributes that are obtained by doing some kind of calculation from existing attributes. An example is to obtain a new attribute named "day type", with three possible values, "Saturday", "Sunday" or "working day", from another attribute of the type timestamp (with the date and the time).
   - Generated records: new records/instances are created. These can be used, for instance, to generate artificial instances of a certain type as an way to deal with unbalanced data sets (see Section 11.4.1).
4) *Integrate data:* In order to have the data in tabular format it is often necessary to integrate data from different tables. The output is:
   - Merged data: an example is the integration of personal data from an university student with information about his/her academic career. This could be done easily if the academic information has an instance per student. But if there are several academic instances per student, for instance one per course in which the student has enrolled, it is still possible to generate a unique instance per student by calculating values such as the average classification or the number of courses the student is enrolled in.

5) *Format data:* this refers to transformations done to the data without transforming its meaning but that are necessary to meet the requirements of the modeling tool. the output is:
   - Reformatted data: some tools have specific assumptions, say the necessity of the attribute to predict being the last one. Other assumptions exist.

The outputs of the data preparation phase are:

- *Data set:* one or more data sets to be used in the modeling phase or the major analysis work of the project.
- *Data set description:* describes the data sets that will be used in the modeling phase or the major analysis work of the project.

A.4 Modeling

Typically, there are several methods to solve the same problem in analytics, some of which will need additional data preparation tasks that are method specific. In such a case it is necessary to go back to the data preparation phase. The modeling phase also includes tuning the hyper-parameters for each of the chosen method(s).

The modeling phase has the following tasks and respective outputs:

1) *Select modeling technique:* in the business understanding phase, the methods or, to be more precise, the families of methods to be used were already identified. Now, it is necessary to choose which specific methods will be used. As an example, in the business understanding phase we might have chosen decision trees, but now we need to decide whether we want to use CART, C5.0 or another technique.
   - Modeling technique: description of the technique to be used.
   - Modeling assumptions: several methods make assumptions about the data, such as nonexistence of missing values, non-existence of outliers, non-existence of irrelevant attributes (not useful for the prediction task). All existing assumptions should be described.
2) *Generate test design:* the definition of the experimental setup must be prepared. This is epecially important for predictive data analytics in order to avoid over-fitting.
   - Test design: describe the planned experimental setup. Resampling approaches should take into consideration the existing amount of data, how unbalanced the data set is (in classification problems) or whether the data arrives as a continuous stream.
3) *Build model:* use the method(s) to obtain one or more models when the problem is predictive or to obtain the desired description(s) when the problem is descriptive.

- Hyper-parameter settings: each method typically has several hyper-parameters. The values used for each hyper-parameter and the process to define them should be described.
- Models: the obtained models or results.
- Model description: description of the models or results, taking into account how interpretable they are.

4) *Assess model:* typically several models/results are generated. It is necessary then to rank these according to a chosen evaluation measure. That evaluation is normally done from the data-analytics point of view, although some business considerations may also be considered.
   - Model assessment: summarize results of this task, listing the qualities of the generated models (say, in terms of accuracy), and ranking their quality in relation to each other.
   - Revised hyper-parameter settings: revise the hyper-parameter settings if necessary according to the model assessment. This will be used in another iteration of the model building task. The iterations of the building and assessment tasks stops when the data analyst believes new iterations are unnecessary.

A.5 Evaluation

To solve the problem from the data-analytics point of view is not the end of the process. It is now necessary to understand how its use is meaningful from the business perspective. In this phase it should be ensured that the solution obtained meets the business requirements.

The evaluation phase has the following tasks and respective outputs:

1) *Evaluate results:* to determine whether the solution obtained meets the business objectives and to evaluate possible non-conformities from the business point of view. If possible, the testing of the model in a real business scenario is helpful. However, this kind of solution is not always possible and, if it is, the costs can be excessive.
   - Assessment of data mining results: describe the assessment results from the business perspective including a final statement about whether the results obtained meet the initially defined business objectives.
   - Approved models: the models that were approved.
2) *Review process:* reviewing all data analysis work in order to verify that it meets the business requirements.
   - Review of process: summarize the review process, highlighting what is missing and what should be repeated.
3) *Determine next steps:* after the review process, the next steps should be decided: to pass to the deployment phase, to repeat some step going back

to a previous phase or to start a new project. Such decisions also depend on the availability of budget and resources.
- List of possible actions: list possible actions, and for each action the pros and cons.
- Decision: describe the decision about whether to proceed and the rationale behind it.

A.6 Deployment

Deployment: the integration of the data analytics solution in the business process is the main purpose of this phase. Typically, it implies integration of the obtained solution in a decision support tool, website maintenance process, reporting process or elsewhere.

The deployment phase has the following tasks and respective outputs:

1) *Plan deployment:* the deployment strategy considers the assessment of the results from the evaluation phase.
   - Deployment plan: summarizes the deployment strategy and describes the procedure to create the necessary models and results.
2) *Plan monitoring and maintenance:* over time, the performance of data analysis methods can change. For that reason it is necessary to define both a monitoring strategy, according to the type of deployment, and a maintenance strategy.
   - Monitoring and maintenance plan: write the monitoring and maintenance plan, step-by-step if possible.
3) *Produce final report:* a final report is written. This can be a synthesis of the project and its experiments or a comprehensive presentation of the data analytics results.
   - Final report: a kind of dossier with all previous outputs summarized and organized.
   - Final presentation: presentation of the final meeting of the project.
4) *Review project:* an analysis of the strong and weak points of the project.
   - Experience documentation: the review is written, including all specific experiences on each phase of the project; everything that can help future data analytics projects.

References

1 Gantz, J. and Reinsel, D. (2012) Big data, bigger digital shadows, and biggest growth in the far east, *Tech. Rep.*, International Data Corporation (IDC).
2 Laney, D. and White, A. (2014) Agenda overview for information innovation and governance, *Tech. Rep.*, Gartner Inc.
3 Cisco Inc. (2016) White paper: Cisco visual networking index: Global mobile data traffic forecast update, 2015–2020, *Tech. Rep.*, Cisco.
4 Simon, P. (2013) *Too Big to Ignore: The Business Case for Big Data*, John Wiley & Sons, Inc.
5 Provost, F. and Fawcett, T. (2013) Data science and its relationship to big data and data-driven decision making. *Big Data*, **1** (1), 51–59.
6 Lichman, M. (2013), UCI machine learning repository. http://archive.ics.uci.edu/ml.
7 Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996) The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, **39** (11), 27–34.
8 Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R. (2000) CRISP-DM 1.0, Step-by-step data mining guide, *report CRISPMWP-1104*, CRISP-DM consortium.
9 Piatestsky, G. (2014), CRISP-DM, still the top methodology for analytics, data mining, or data science projects. http://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-ana lytics-data-mining-data-science-projects.html.
10 Weiss, N. (2014) *Introductory Statistics*, Pearson Education.
11 Chernoff, H. (1973) The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, (**68**), 361–368.
12 Tabachnick, B.G. and Fidell, L.S. (2014) *Using Multivariate Statistics*, Pearson New International Edition.

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

13 Maletic, J.I. and Marcus, A. (2000) Data cleansing: Beyond integrity analysis, in *Proceedings of the Conference on Information Quality*, pp. 200–209.

14 Pearson, K. (1902) On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, **2** (6), 559–572.

15 Strang, G. (2016) *Introduction to Linear Algebra*, Wellesley-Cambridge Press, 5th edn.

16 Benzecri, J. (1992) *Correspondence Analysis Handbook*, Marcel Dekker.

17 Messaoud, R.B., Boussaid, O., and Rabaséda, S.L. (2006) Efficient multidimensional data representations based on multiple correspondence analysis, in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 662–667.

18 Comon, P. (1994) Independent component analysis, a new concept? *Signal Processing*, **36** (3), 287–314.

19 Cox, M. and Cox, T. (2000) *Multidimensional Scaling*, Chapman & Hall/CRC, 2nd edn.

20 Tan, P., Steinbach, M., and Kumar, V. (2014) *Introduction to Data Mining*, Pearson Education.

21 Aggarwal, C. and Jiawei, H. (2014) *Frequent Pattern Mining*, Springer.

22 Wolberg, W.H., Street, W.N., and Mangasarian, O.L. (1995), Breast cancer wisconsin (diagnostic) data set, UCI Machine Learning Repository. URL https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+Cancer+Wisconsin+%28Diagnostic%29.

23 Bulmer, M. (2003) *Francis Galton: Pioneer of heredity and biometry*, The Johns Hopkins University Press.

24 Kohavi, R. (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection, in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, Morgan Kaufmann, San Francisco, CA, USA, pp. 1137–1143. URL http://dl.acm.org/citation.cfm?id=1643031.1643047.

25 Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014) Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, **15** (1), 3133–3181.

26 Swets, J.A., Dawes, R.M., and Monahan, J. (2000) Better decisions through science. *Scientific American*, **283** (4), 82–87.

27 Provost, F. and Fawcett, T. (2013) *Data Science for Business: What you need to know about data mining and data-analytic thinking*, O'Reilly Media, Inc., 1st edn.

28 Flach, P. (2012) *Machine Learning: The art and science of algorithms that make sense of data*, Cambridge University Press.

29 Aamodt, A. and Plaza, E. (1994) Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, **7** (1), 39–59.

30 Rokach, L. and Maimon, O. (2005) Top-down induction of decision trees classifiers – a survey. *IEEE Transactions on Systems, Man and Cybernetics: Part C*, **35** (4), 476–487.

31 Quinlan, J.R. (1992) Learning with continuous classes, in *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, World Scientific, pp. 343–348.

32 Rosenblatt, F. (1958) The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, **65** (6), 65–386.

33 Novikoff, A.B.J. (1962) On convergence proofs on perceptrons, in *Proceedings of the Symposium on the Mathematical Theory of Automata*, vol. XII, pp. 615–622.

34 Werbos, P.J. (1974) *Beyond Regression: New tools for prediction and analysis in the behavioral sciences*, Ph.D. thesis, Harvard University.

35 Parker, D.B. (1985) Learning-logic, *Tech. Rep. TR-47*, Center for Comp. Research in Economics and Management Sci., MIT.

36 LeCun, Y. (1985) Une procédure d'apprentissage pour réseau à seuil asymétrique. *Proceedings of Cognitiva 85, Paris*, pp. 599–604.

37 Rumelhart, D., Hinton, G., and Williams, R. (1986) Learning internal representations by error propagation, in *Parallel Distributed Processing*, vol. 1 (eds D.E. Rumelhart and J.L. McClelland), MIT Press, pp. 318–362.

38 Kolmogorov, A.K. (1957) On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk SSSR*, **114**, 369–373.

39 Goodfellow, I., Bengio, Y., and Courville, A. (2016) *Deep Learning*, MIT Press.

40 Fukushima, K. (1979) Neural network model for a mechanism of pattern recognition unaffected by shift in position – Neocognitron. *Transactions of the IECE*, **J62-A**(10), 658–665.

41 Lecun, Y., Bengio, Y., and Hinton, G. (2015) Deep learning. *Nature*, **521** (7553), 436–444.

42 Cortes, C. and Vapnik, V. (1995) Support-vector networks. *Machine Learning*, **20** (3), 273–297.

43 Burges, C. (1998) A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2** (2), 121–167.

44 Friedman, J.H. (2000) Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, **29**, 1189–1232.

45 Freund, Y. and Shapire, R.E. (1996) Experiments with a new boosting algorithm, in *Proceedings of the 13th International Conference on Machine Learning*, ICML 1996, pp. 148–156.

46 de Carvalho, A. and Freitas, A. (2009) A tutorial on multi-label classification techniques, in *Foundations of Computational Intelligence Volume 5: Function Approximation and Classification* (eds A. Abraham, A.E. Hassanien, and V. Snášel), Springer, pp. 177–195.

47 Tsoumakas, G. and Katakis, I. (2007) Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, **3** (3), 1–13.

48 Freitas, A. and de Carvalho, A. (2007) A tutorial on hierarchical classification with applications in bioinformatics, in *Research and Trends in Data Mining Technologies and Applications* (ed. D. Taniar), Idea Group, pp. 175–208.

49 Ren, Z., Peetz, M., Liang, S., van Dolen, W., and de Rijke, M. (2014) Hierarchical multi-label classification of social text streams, in *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, ACM, pp. 213–222.

50 Settles, B. (2012) *Active Learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool.

51 Zikeba, M. and Tomczak, S.K. and Tomczak, J.M. (2016) Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. *Expert Systems with Applications*, **58**, 93–101.

52 Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I.H. (2009) The WEKA data mining software: An update. SIGKDD Explorations *Newsletter*, **11** (1), 10–18.

53 Weiss, S.M., Indurkhya, N., and Zhang, T. (2015) *Fundamentals of Predictive Text Mining*, Springer, 2nd edn.

54 Porter, M. (1980) An algorithm for suffix stripping. *Program*, **14** (3), 130–137.

55 Witten, I.H., Frank, E., and Hall, M.A. (2011) *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 3rd edn.

56 Ricci, F., Rokach, L., Shapira, B., and Kantor, P. (2010) *Recommender Systems Handbook*, Springer-Verlag, 1st edn.

57 Hanneman, R. and Riddle, M. (2005), *Introduction to Social Networks Methods*. Published online at http://faculty.ucr.edu/~hanneman/nettext/.

58 Zafarani, R., Abbasi, M., and Liu, H. (2014) *Social Media Mining: An introduction*, Cambridge University Press, 1st edn.

Index

a

Absolute cumulative frequency 26, 51
Absolute frequency 25, 26, 28, 30, 34
Absolute scale 23, 24, 81
Activation function 12, 222, 224, 225, 229, 231–233, 239
Active learning 17, 254, 255
AdaBoost 243, 245, 246, 256, 257, 260
Adjacency matrix 291–294, 301
Advanced predictive topics 241–257
Agglomerative hierarchical clustering 110, 117–122, 153
Algorithm bias 17, 246, 247
Amplitude 36, 38, 47, 60, 61, 84, 152
Apriori 131–134, 147, 149, 150, 154
Area chart 28, 29, 152
Area under the ROC curve (AUC) 197–199
Artificial neural networks (ANN) 4, 17, 221–234, 238, 243, 244, 260
Association rule 126, 139–145, 147, 149, 150, 153, 154, 211
Association rule mining 140, 147, 153
Attribute 7
Attribute aggregation 88–92, 97 *see also* Linear combination of attributes
Attribute selection 88, 91–95, 174, 180, 181, 184, 200, 221
Average linkage 118–120

b

Backpropagation algorithm 12, 224–231, 233, 239, 240
Backward selection 95
Bagging 243–246, 256, 257, 260
Bag of words 104, 105, 124, 272, 276–278, 299
Bar chart 27–30, 152
Bayes theorem 203–205, 207, 208
Betweenness 294, 296, 298, 301
Bias 17, 174–177, 179, 183, 184, 202, 246, 247, 257, 285–287
Bias-variance 17, 174–177, 179, 183, 184, 287
Biclustering 122
Big data 4–7, 13, 202
Binary classification 17, 183, 187–194, 205, 207, 222, 234, 237, 238, 249, 253
BIRCH 122
Bivariate analysis 22, 40, 46, 49, 50, 59, 67
Boosting 245, 246
Bootstrap 166, 168, 169, 243–245, 312
Box-plot 33–35, 37, 60
Breast cancer Wisconsin dataset 11, 154–156, 158
Business Understanding 15, 16, 154, 260, 303–305

A General Introduction to Data Analytics, First Edition. João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth.
© 2019 John Wiley & Sons, Inc. Published 2019 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/moreira/dataanalytics

c

C4.5 213, 214, 240, 260, 265, 266
Candidate itemset 132–134
Case-based reasoning 199, 202, 203
Centrality score 297, 298
Centralization 297, 298, 301
Central tendency statistics 33–36
Centroid 108, 110–115, 117, 122, 156, 157
Characteristic tree induction algorithms (CTIA) 211
Cheat sheet on descriptive analytics 151–154
Cheat sheet on predictive analytics 259, 260
Chernoff faces 56, 58, 68
Classification 14, 17, 51, 72, 73, 76, 93–95, 107, 108, 162, 164, 167, 182, 183, 187–257, 259, 265, 270, 271, 299, 300, 307
Classification and regression tree (CART) 211, 213–220, 307
Classification tasks 17, 51, 73, 107, 108, 162, 187–194, 203–212, 216, 222, 223, 226, 228–231, 234, 236–238, 241, 242, 248–254, 271, 272
Clique 299, 301
Closed frequent itemset 138, 139
Closed frequent sequence 148, 149
Closeness 294, 295, 298, 301
Clustering 14, 17, 58, 99–124, 228, 255, 299
Clustering coefficient 297, 299, 301
Clustering techniques 6, 96, 99, 107–123, 151, 300
Clustering validation 99, 107, 108
Coefficient of variation 171, 184
Cold-start problem 290
Collaborative filtering 282–289, 301
Complete linkage 119, 120
Confidence 125, 139–146, 149, 154, 255, 265
Confusion matrix 193–196, 266
Content-based recommendation 282, 283, 300
Content-based techniques 282
Contingency table 45, 46, 48, 144–146, 152
Converting to a different scale 83–85
Converting to a different scale type 77–83
Convex shape 111, 113, 115
Corpus 271, 275
Correlation 7, 42–44, 46, 48, 61–65, 88, 89, 92, 93, 145, 146, 152, 182, 264, 286, 287
Correlograms 65
Covariance 42, 43, 48, 61, 62, 88, 91
Covariance matrix 61, 62
CRISP-DM methodology 12, 13, 15–17, 83, 154, 259, 303–309
Cross-support pattern 143, 144, 149, 150
CTIA 211
Cumulative frequency 26, 50
Curse of dimensionality 95, 97, 247

d

Data 7
Data acquisition 271
Data analytics 4, 5, 9, 10, 12, 13, 15, 16, 18, 21, 71, 96, 122, 143, 146, 151, 153, 208, 212, 230, 269, 270, 276–278, 303–305, 307–309
Data mining 4, 5, 13–15, 99, 103, 143, 246, 260, 270, 271, 275, 277, 304, 308
Data preparation 15–17, 83, 155, 156, 253–255, 265, 304–307
Data quality 16, 17, 71–77, 96
Data science 4, 5, 266
Data streams 5, 122, 182, 183, 213, 238
Data transformation 16, 71, 85, 86, 96, 306

Data type 5, 14, 25, 27, 35, 96, 125, 305
Data Understanding 15, 16, 154, 260, 305
DBSCAN 110, 115–118, 123, 153
Decision support 16, 154, 212, 309
Decision tree induction algorithms (DTIA) 14, 211–217, 219, 226, 238, 239
Decision trees 10, 17, 212, 213, 215–218, 243–245, 256, 257, 305, 307
Decision Trees for Regression 217–221
Deduction 17, 21, 22, 162
Deduplication 74
Deep learning (DL) 228, 230–235, 238, 239, 260
Degree 219, 221, 294, 296–299
Dendrogram 65, 120–124
Density-based clustering 110, 115
Deployment 15, 16, 18, 158, 266, 308, 309
Descriptive analytics 9, 14, 17, 96, 151, 154
Descriptive bivariate analysis 40–46
Descriptive multivariate analysis 49–69
Descriptive statistics 17, 21–48
Descriptive univariate analysis 25
Diameter 297, 301
Dimensionality reduction 71, 86–96, 277
Directed edge 294, 295
Discretization 82
Discriminative algorithms 205
Dispersion multivariate statistics 60–66
Dispersion statistic 32, 36–38, 47, 60, 61, 152
Dispersion univariate statistics 36, 38, 61
Dissimilarity 101, 102, 120
Distance 17, 33, 36, 37, 72–74, 78, 79, 81, 83–85, 87, 91, 96, 97, 100–106, 108, 110, 111, 113, 114, 116–120, 123, 124, 151, 153, 156, 172, 198–202, 237, 265, 285, 294–297, 301
Distance-based learning algorithms 199–202
Distance measure 17, 83, 100–106, 110, 111, 113, 114, 117, 120, 124, 153, 156, 200, 265, 285
Distribution function 25–27, 29–31, 206, 238
Divide-and-conquer 213, 214
Document classification 270
Draftsman's display 62
Dynamic time warping 105, 153

e

Eclat 133, 134, 147, 149, 150, 154
Edge 110, 116, 138, 291, 292, 294, 295
Edit distance 104–106, 123, 124, 153
Elbow curve 115, 158
Embedded 92, 94, 95
Empirical cumulative distribution 26, 30
Empirical distribution 25, 29–31, 208
Empirical error 173–175
Empirical frequency distribution 26
Empirical loss 173
Ensemble 17, 123, 241–246, 256
Enumeration tree 132
Euclidean distance 83–85, 103, 104, 108, 111, 118, 124, 153, 265
Evaluation 15, 16, 107, 142, 143, 149, 158, 164, 192, 210, 256, 265, 266, 271, 277–279, 289, 290, 304, 308, 309
Explicit feedback 279, 280
External index 107

f

False negative rate (FNR) 194, 196–198
False positive rate (FPR) 194, 196–199
Feature 7
Feature extraction 88, 271–277
Feedback 99, 279–282, 284–287, 290
Filter 76, 92–95, 128, 143, 150, 232, 233, 269, 276, 277
FP-tree 134–137
Frequent itemset 126–142, 147–150, 153, 154
Frequent pattern growth method (FP-Growth) 134–137, 147, 149, 150, 154
Frequent pattern mining 17, 125–150, 153, 154, 300
Frequent sequence 127, 147–150

g

Gaussian distribution 40, 91 *see also* Normal distribution
Generalization 10, 17, 164, 165, 226, 234
Generative algorithms 205
Gradient boosting 246
Gradient descent 224, 232, 239, 288
Graph 29, 52, 110, 147, 150, 153, 154, 190, 197, 198, 216, 217, 291, 292, 295, 297
Graph-based clustering 110
Graphics processing units (GPUs) 230
Gray code 81, 82
Greedy 95, 213, 217

h

Hadoop 5, 6
Hamming distance 104, 124, 153
Header table 134–137
Heatmap 64–66, 69
Hierarchical classification 188, 248, 252, 253
Histogram 28–30, 35, 37, 41, 47, 152
Holdout 165, 166
Hunt algorithm 213, 214
Hyper-parameter 12, 14, 16, 18, 90, 91, 110, 111, 113, 114, 116–118, 120, 122, 123, 128, 153, 165, 173, 175, 179, 181, 184, 206–208, 216, 219, 221, 225, 229, 233, 237, 238, 240, 243–246, 257, 259, 288

i

Imbalanced data 17, 196, 210, 253, 254
Incomplete target labeling 253–255
Inconsistent data 75, 76, 155
Independent component analysis (ICA) 88, 91, 92
Induction 10, 12, 21, 22, 71, 88, 94, 95, 161, 165, 166, 183, 188, 190, 195, 211–213, 216, 217, 226–228, 231, 239, 242, 248, 252, 255, 256, 260, 271, 275, 277
Inductive learning 4, 40, 161, 254, 255
Infographics 66, 67, 69
Information retrieval 195, 270, 272, 277, 279
Instance 7
Internal index 107
Interquartile range 36, 38, 60, 61, 77, 78, 152
Item 125–129, 131, 133–144, 147, 148, 174, 269, 279–282, 284–290
Item-based collaborative filtering 284, 286
Item recommendation 280, 281, 285–287
Itemset 127
Iterative dichotomiser 3 (ID3) 213, 214

j

Jaccard measure 107, 108
Join-based frequent pattern mining 131

k

KDD process 12–15
k-fold cross-validation 166, 167
k-means 12, 110–115, 117, 118, 122, 123, 153, 156, 157
k-nearest neighbors (K-NN) 260, 265, 284 *see also* K-NN algorithm
K-NN algorithm 76, 199–202, 208, 210, 266 *see also* k-nearest neighbors (K-NN)
Kernel function 91, 234, 236, 240
Kernel PCA 88, 91
Knowledge-based recommendation 290
Knowledge-based techniques 281, 282
Knowledge discovery 5, 12

l

Label attribute 76, 92, 225 *see also* Target attribute
Lasso 17, 179–181, 185
Least absolute shrinkage and selection operator 174 *see also* Lasso
Levenshtein distance 104, 153 *see also* Edit distance
Lift 144–146, 149, 150, 154
Likert scale 35, 36
Linear combination of attributes 174, 181, 182 *see also* Attribute aggregation
Linear correlation 42, 43, 62, 63, 89
Linearly separable 191, 192, 207, 223, 234, 236
Linear regression 171–175, 181, 206
Line chart 28, 152
Linkage criterion 119, 120
Location multivariate statistics 59, 60
Location statistics 32–34, 59, 96, 152
Location univariate statistics 32, 33, 59
Logistic regression 205–207, 209, 210, 219, 260

m

Machine learning 4, 12, 154, 162, 246, 259, 270, 279, 282, 299
Machine learning algorithms 270
Manhattan distance 103, 104, 106, 124, 153
MapReduce 5, 6
Maximal frequent itemset 138, 139, 148
Maximal frequent sequence 148, 149
Maximum 17, 22, 23, 29, 32, 33, 36, 39, 49, 54, 84, 143, 152, 168, 219, 221, 281, 282, 298
Mean 32–37, 40, 47, 59, 73, 113, 152, 162, 172, 175
Mean absolute deviation 36, 38, 60, 152
Mean Square Error (MSE) 170–172, 176, 177, 184, 185, 279
Median 33–36, 47, 59, 60, 73, 152
Medoid 111
Metadata 277, 278
Minimum 32, 33, 36, 39, 54, 84, 104, 108, 116, 117, 120, 143, 152, 216, 217, 265, 294
Minimum confidence threshold 142
Minimum support threshold 131
Minkowski distance 103, 111
Min–max rescaling 83, 84, 97
Min_sup threshold 128–131, 136, 138, 140, 144, 148–150
Missing values 14, 15, 72–74, 76, 96, 97, 155, 164, 260, 265, 285, 305–307
Mode 33–36, 47, 59, 60, 73, 152
Model-based collaborative filtering 287
Modeling 4, 12, 15, 16, 68, 72, 85, 99, 157, 158, 265, 304, 306–308
Model trees 218–221, 260
Model validation 164, 169
Modularity 299, 301
Monotonicity 129, 132, 140, 141

Multi-class classification 237, 248–250, 257
Multidimensional scaling (MDS) 88, 91
Multi-label classification 248, 251, 253
Multi-layer perception (MLP) 224–233, 239, 240
Multi-layer perceptron network (MLP network) 224–233, 239, 257
Multi-objective optimization 95
Multiplex edge 291
Multivariate adaptive regression splines (MARS) 218–221, 260
Multivariate analysis 22, 49, 59, 65, 67, 68
Multivariate data visualization 50–59
Multivariate frequency 49–50
Multivariate linear model 173, 179, 181
Multivariate linear regression (MLR) 17, 172–174, 177, 180–182, 184, 185, 219–221, 260
Multivariate statistics 59–66

n

Naïve Bayes 205, 207–210, 260
Natural language processing 231, 277
Negative predictive value (NPV) 194
Neighborhood-based collaborative filtering 282, 284
Neuron 222–226, 228, 229, 231, 233, 239, 240
Node 225, 230, 239, 242, 245, 252, 257, 291–301
Noise 71, 72, 74, 77, 89, 96, 115, 153, 164, 175, 209, 230, 234, 264 *see also* Noisy data
Noisy data 71, 76, 77, 88, 89, 91, 96, 97, 115, 155, 200, 243, 264 *see also* Noise
Nominal 22–25, 27, 34, 40, 45, 65, 73, 78–82, 96, 97, 102, 152, 162
Non-binary classification 248–253
Non-linearly separable 191, 223, 224, 234, 236
Normal distribution 40, 152, 238 *see also* Gaussian distribution
Normalization 83–85, 114, 156, 217, 221, 259

o

Object 7
Objective function 171–173, 179, 222, 288
Off-line evaluation 289
One-attribute-per-value 78
One-class classification 248, 249
1-of-n 78, 80
Opinion mining 278
Optimization-based algorithms 198, 211, 221–238
Ordinal 22–25, 27, 34, 35, 40, 46, 62, 65, 73, 81, 82, 102, 152, 162
Outliers 14–16, 72, 77, 114–118, 120, 153, 164, 175, 180, 181, 202, 207, 217, 221, 264, 300, 305–307
Overfitting 94, 165, 176, 192, 215, 216, 219, 225, 226, 229, 234, 237, 243, 282

p

Parallel coordinates 53–55, 68
Parameter 12, 39, 40, 107, 163, 165, 171, 172, 205, 222, 288
Partial least squares 17, 182, 260
Path 134–136, 138, 163, 212, 215, 292, 293, 296
Pearson correlation 42–44, 46, 61–65, 88, 92, 152, 286, 287
Perceptron convergence theorem 223
Perceptron network 222–224, 228, 234, 235, 239

Performance measures for classification 17, 192–199, 216, 256
Performance measures for regression 17, 169–171, 216, 256
Pie chart 27, 28, 152
Pointer 134
Polish Company Insolvency Data 11, 259, 261–264
Population 21, 22, 25, 27, 30–32, 35–37, 39, 40, 176
Population mean 35
Positive predictive value (PPV) 194
Precision 27, 31, 165, 194–196, 210, 266
Predictive analytics 9, 14, 16, 17, 187, 241, 259
Predictive model 95, 107, 161, 162, 165, 166, 208, 211, 212, 239, 245, 253, 255, 272, 275, 282
Predictive performance estimation 164–171
Predictive task 10, 99, 107, 161, 162, 164, 182, 183, 187, 208, 211, 226, 231, 254, 255, 270
Prefix-path 136, 137
Principal component analysis (PCA) 88–92, 97, 113, 181, 287
Principal component regression 17, 181, 182, 260
Principal components 89–91, 182, 183
Probabilistic classification algorithms 203–208
Probabilities 17, 21, 39, 204, 206–208
Probability density function 27, 29, 38–40, 50
Probability distribution 27, 30, 31, 38–40, 47, 48, 238
Project on descriptive analytics 154–158
Project on predictive analytics 259–266
Properties of networks 297–299
Properties of nodes 294–297, 299, 301
Prototype-based clustering 110

q

Qualitative scale 22, 24, 26, 27, 77
Quantitative scale 25–19, 35, 37, 77, 82
Quartile 32–34, 36, 47, 59, 77, 152

r

Radar chart 56
Radial basis function 238
Random forests 243–245, 256, 265
Random sub-sampling 166, 167
Ranking 43, 44, 89, 91–94, 97, 102, 250–252, 279, 280, 282, 308
Ranking classification 248, 250, 251, 257
Rating 49, 206, 279–288, 290, 300, 301
Rating prediction 280, 281, 285, 287
Recall 194–198, 266
Receiver operating characteristics (ROC) 197, 198, 202
Recommendation systems 126, 250, 269 *see also* Recommender systems
Recommendation tasks 280, 281
Recommendation technique 279, 281–291
Recommender systems 270, 278–291 *see also* Recommendation systems
Rectified linear unit (ReLU) 232
Recursive algorithms 213
Redundant data 71, 74, 96, 97, 155
Regression 14, 17, 161–185, 201, 202, 206, 211, 216–221, 228, 229, 233, 234, 237, 238, 244–246, 254, 256, 287, 299
Regression tasks 162, 182, 211, 216, 229, 233
Relative cumulative frequency 26, 50

Relative frequency 25, 26, 28, 39, 45, 50
Relative mean square error (RelMSE) 170, 171, 184
Relative scale 23, 24, 79–82
Ridge regression 179–181, 185, 260
Root mean square error (RMSE) 170, 171
Rule set induction algorithms (RSIA) 211, 213

S

Sample 4, 11, 13, 21, 22, 25, 26, 30–32, 35, 37–39, 43, 76, 85, 121, 155, 166, 169, 183, 242–244, 252, 271
Sample mean 35, 38
Sample mean absolute deviation 38
Sample standard deviation 38, 43, 85
Sample variance 38
Scale types 17, 22–25, 40, 71, 152
Scatter plot matrix 62, 63
Scatter plots 41, 42, 45–48, 62, 63, 69
Search-based algorithms 211–221
Search strategies 95, 96
Semi-supervised clustering 123
Semi-supervised learning 17, 123, 254, 255
SEMMA 13
Sensitivity 194 *see also* Recall
Sentiment analysis 270, 272, 274, 277, 278, 300
Separation-based clustering 110
Sequence database 147, 148, 150
Sequential patterns 147–149
Shallow networks 230, 233
Shared-property clustering 110
Shrinkage methods 174, 177–181
Silhouette index 107, 108
Similarity 24, 65, 100–102, 108, 120, 124, 270, 284–287, 300
Simple linear regression 163, 164, 171
Simpson's paradox 145–147, 149, 150
Single linkage 119, 120
Singular value decomposition (SVD) 89, 90
Small data 6, 7, 16, 167, 168, 300
Social network analysis 269, 291–300
Spark 5
Spearman's rank correlation 43, 44, 46, 152
Specificity 194, 196–198, 210
Spider plot 56
Standard deviation 37, 38, 40, 47, 59, 60, 84, 152
Standardization 83, 84, 97
Star plot 56, 57
Statistical inference 21
Stemmer 273, 274
Stemming 67, 272–277, 299
Stemming algorithm 273
Step function 222
Stop words 274–277
Storm 5
Streamograph 58
Structured data 270, 271, 275, 276
Subsequence 147, 148
Summarization 14, 16, 22, 85, 151, 259, 305
Supervised interpretable techniques 17, 255, 256
Support 127–134, 136, 138–145, 148, 149, 154
Support ratio 143
Support vectors 234–238
Support vector machines (SVM) 17, 221, 233–238, 240, 249, 256, 260
SVM for regression 237, 238
Synthetic minority oversampling technique (SMOTE) 254

t

Tabular data 7, 14, 270
Target attribute 62, 74, 75, 76, 92–94, 155, 162, 167, 169–172, 180–182, 187, 188, 203, 206, 212, 219, 253, 260, 282 *see also* Label attribute

Technique and model selection 182, 183
Test data 162, 165, 175, 183
Text categorization 270
Text classification 253, 272, 276
Text mining 67, 269–278, 285, 300
TID-set 133, 134
Token 272–274
Tokenization 272, 276, 277, 299,
Training 12, 88, 162, 165, 166, 209, 210, 222, 223, 225–233, 245, 248, 249, 252
Training data set 162, 165–169, 171–173, 175–177, 187, 195, 199–202, 205, 209, 210, 213, 214, 223, 225, 226, 229, 230, 232–234, 240–243, 245, 248, 252, 254, 255, 257, 272, 273, 275, 276, 282, 284, 286, 297, 300
Transaction 125–127, 129, 130, 133, 134, 136, 138, 139, 143, 144, 146, 149
Transactional data 125–128, 131, 133, 134, 140, 150
Transductive learning 254, 255
Triad 297
True negative rate (TNR) 194, 197, 198
True positive rate (TPR) 194, 197–199

u

Unary classification 248
Underfitting 176
Undirected edge 291, 294, 295
Uniform distribution 39
Unimodal distribution 35, 36
Univariate analysis 22, 25, 59
Univariate data visualization 27–32
Univariate frequencies 25–27
Univariate linear model 172
Univariate plot 27, 28
Univariate statistics 32–38, 59
User 7, 12, 14, 58, 59, 66, 90, 95, 126–129, 142, 143, 149, 158, 202, 226, 244, 256, 269, 278–291
User-based collaborative filtering 284, 286
User-item matrix 286

v

Variance 37, 40, 42, 47, 61, 89, 91, 119, 167, 173, 174, 176, 177, 179, 182, 217, 244
Very fast decision trees (VFDT) 213
Visualization 5, 16, 22, 27, 41, 47, 49, 50, 58, 65–68, 86, 88, 155, 260, 300, 305

w

Ward linkage 119
Web mining 270, 278
Weighted edge 292
Within-groups sum of squares 107, 108, 111, 114, 115, 158
Word cloud 66–68
Wrapper 92–95

x

XGBoost 246