David L. Olson
Dursun Delen

Advanced Data Mining Techniques

Springer

David L. Olson · Dursun Delen

Advanced Data Mining Techniques

Springer

Dr. David L. Olson
Department of Management Science
University of Nebraska
Lincoln, NE 68588-0491
USA
dolson3@unl.edu

Dr. Dursun Delen
Department of Management
Science and Information Systems
700 North Greenwood Avenue
Tulsa, Oklahoma 74106
USA
dursun.delen@okstate.edu

ISBN: 978-3-540-76916-3 e-ISBN: 978-3-540-76917-0

Library of Congress Control Number: 2007940052

© 2008 Springer-Verlag Berlin Heidelberg

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Cover design: WMX Design, Heidelberg

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

I dedicate this book to my grandchildren.

David L. Olson

I dedicate this book to my children, Altug and Serra.

Dursun Delen

Preface

The intent of this book is to describe some recent data mining tools that have proven effective in dealing with data sets which often involve uncertain description or other complexities that cause difficulty for the conventional approaches of logistic regression, neural network models, and decision trees. Among these traditional algorithms, neural network models often have a relative advantage when data is complex. We will discuss methods with simple examples, review applications, and evaluate relative advantages of several contemporary methods.

Book Concept

Our intent is to cover the fundamental concepts of data mining, to demonstrate the potential of gathering large sets of data, and analyzing these data sets to gain useful business understanding. We have organized the material into three parts. Part I introduces concepts. Part II contains chapters on a number of different techniques often used in data mining. Part III focuses on business applications of data mining. Not all of these chapters need to be covered, and their sequence could be varied at instructor design.

The book will include short vignettes of how specific concepts have been applied in real practice. A series of representative data sets will be generated to demonstrate specific methods and concepts. References to data mining software and sites such as www.kdnuggets.com will be provided.

Part I: Introduction

Chapter 1 gives an overview of data mining, and provides a description of the data mining process. An overview of useful business applications is provided.

Chapter 2 presents the data mining process in more detail. It demonstrates this process with a typical set of data. Visualization of data through data mining software is addressed.

Part II: Data Mining Methods as Tools

Chapter 3 presents memory-based reasoning methods of data mining. Major real applications are described. Algorithms are demonstrated with prototypical data based on real applications.

Chapter 4 discusses association rule methods. Application in the form of market basket analysis is discussed. A real data set is described, and a simplified version used to demonstrate association rule methods.

Chapter 5 presents fuzzy data mining approaches. Fuzzy decision tree approaches are described, as well as fuzzy association rule applications. Real data mining applications are described and demonstrated

Chapter 6 presents Rough Sets, a recently popularized data mining method.

Chapter 7 describes support vector machines and the types of data sets in which they seem to have relative advantage.

Chapter 8 discusses the use of genetic algorithms to supplement various data mining operations.

Chapter 9 describes methods to evaluate models in the process of data mining.

Part III: Applications

Chapter 10 presents a spectrum of successful applications of the data mining techniques, focusing on the value of these analyses to business decision making.

University of Nebraska-Lincoln David L. Olson

Oklahoma State University Dursun Delen

Contents

Part I INTRODUCTION

1 Introduction....3
- What is Data Mining?....5
- What is Needed to Do Data Mining....5
- Business Data Mining....7
- Data Mining Tools....8
- Summary....8

2 Data Mining Process....9
- CRISP-DM....9
  - Business Understanding....11
  - Data Understanding....11
  - Data Preparation....12
  - Modeling....15
  - Evaluation....18
  - Deployment....18
- SEMMA....19
  - Steps in SEMMA Process....20
- Example Data Mining Process Application....22
- Comparison of CRISP & SEMMA....27
- Handling Data....28
- Summary....34

Part II DATA MINING METHODS AS TOOLS

3 Memory-Based Reasoning Methods....39
- Matching....40
  - Weighted Matching....43
- Distance Minimization....44
- Software....50
- Summary....50
- Appendix: Job Application Data Set....51

4 Association Rules in Knowledge Discovery 53
Market-Basket Analysis 55
Market Basket Analysis Benefits 56
Demonstration on Small Set of Data 57
Real Market Basket Data 59
The Counting Method Without Software 62
Conclusions 68

5 Fuzzy Sets in Data Mining 69
Fuzzy Sets and Decision Trees 71
Fuzzy Sets and Ordinal Classification 75
Fuzzy Association Rules 79
Demonstration Model 80
Computational Results 84
Testing 84
Inferences 85
Conclusions 86

6 Rough Sets 87
A Brief Theory of Rough Sets 88
Information System 88
Decision Table 89
Some Exemplary Applications of Rough Sets 91
Rough Sets Software Tools 93
The Process of Conducting Rough Sets Analysis 93
1 Data Pre-Processing 94
2 Data Partitioning 95
3 Discretization 95
4 Reduct Generation 97
5 Rule Generation and Rule Filtering 99
6 Apply the Discretization Cuts to Test Dataset 100
7 Score the Test Dataset on Generated Rule set (and measuring the prediction accuracy) 100
8 Deploying the Rules in a Production System 102
A Representative Example 103
Conclusion 109

7 Support Vector Machines 111
Formal Explanation of SVM 112
Primal Form 114

Dual Form 114
Soft Margin 114
Non-linear Classification 115
Regression 116
Implementation 116
Kernel Trick 117
Use of SVM – A Process-Based Approach 118
Support Vector Machines versus Artificial Neural Networks 121
Disadvantages of Support Vector Machines 122

8 Genetic Algorithm Support to Data Mining 125
Demonstration of Genetic Algorithm 126
Application of Genetic Algorithms in Data Mining 131
Summary 132
Appendix: Loan Application Data Set 133

9 Performance Evaluation for Predictive Modeling 137
Performance Metrics for Predictive Modeling 137
Estimation Methodology for Classification Models 140
Simple Split (Holdout) 140
The *k*-Fold Cross Validation 141
Bootstrapping and Jackknifing 143
Area Under the ROC Curve 144
Summary 147

Part III APPLICATIONS

10 Applications of Methods 151
Memory-Based Application 151
Association Rule Application 153
Fuzzy Data Mining 155
Rough Set Models 155
Support Vector Machine Application 157
Genetic Algorithm Applications 158
Japanese Credit Screening 158
Product Quality Testing Design 159
Customer Targeting 159
Medical Analysis 160

Predicting the Financial Success of Hollywood Movies 162
Problem and Data Description 163
Comparative Analysis of the Data Mining Methods 165
Conclusions 167

Bibliography 169

Index 177

Part I
INTRODUCTION

1 Introduction

Data mining refers to the analysis of the large quantities of data that are stored in computers. For example, grocery stores have large amounts of data generated by our purchases. Bar coding has made checkout very convenient for us, and provides retail establishments with masses of data. Grocery stores and other retail stores are able to quickly process our purchases, and use computers to accurately determine product prices. These same computers can help the stores with their inventory management, by instantaneously determining the quantity of items of each product on hand. They are also able to apply computer technology to contact their vendors so that they do not run out of the things that we want to purchase. Computers allow the store's accounting system to more accurately measure costs, and determine the profit that store stockholders are concerned about. All of this information is available based upon the bar coding information attached to each product. Along with many other sources of information, information gathered through bar coding can be used for data mining analysis.

Data mining is not limited to business. Both major parties in the 2004 U.S. election utilized data mining of potential voters.[1] Data mining has been heavily used in the medical field, to include diagnosis of patient records to help identify best practices.[2] The Mayo Clinic worked with IBM to develop an online computer system to identify how that last 100 Mayo patients with the same gender, age, and medical history had responded to particular treatments.[3]

Data mining is widely used by banking firms in soliciting credit card customers,[4] by insurance and telecommunication companies in detecting

---

[1] H. Havenstein (2006). IT efforts to help determine election successes, failures: Dems deploy data tools; GOP expands microtargeting use, *Computerworld* 40: 45, 11 Sep 2006, 1, 16.

[2] T.G. Roche (2006). Expect increased adoption rates of certain types of EHRs, EMRs, *Managed Healthcare Executive* 16:4, 58.

[3] N. Swartz (2004). IBM, Mayo clinic to mine medical data, *The Information Management Journal* 38:6, Nov/Dec 2004, 8.

[4] S.-S. Weng, R.-K. Chiu, B.-J. Wang, S.-H. Su (2006/2007). The study and verification of mathematical modeling for customer purchasing behavior, *Journal of Computer Information Systems* 47:2, 46–57.

fraud,[5] by telephone companies and credit card issuers in identifying those potential customers most likely to churn,[6] by manufacturing firms in quality control,[7] and many other applications. Data mining is being applied to improve food and drug product safety,[8] and detection of terrorists or criminals.[9] Data mining involves statistical and/or artificial intelligence analysis, usually applied to large-scale data sets. Traditional statistical analysis involves an approach that is usually directed, in that a specific set of expected outcomes exists. This approach is referred to as *supervised* (hypothesis development and testing). However, there is more to data mining than the technical tools used. Data mining involves a spirit of knowledge discovery (learning new and useful things). Knowledge discovery is referred to as *unsupervised* (knowledge discovery) Much of this can be accomplished through automatic means, as we will see in decision tree analysis, for example. But data mining is not limited to automated analysis. Knowledge discovery by humans can be enhanced by graphical tools and identification of unexpected patterns through a combination of human and computer interaction.

Data mining can be used by businesses in many ways. Three examples are:

1. *Customer profiling*, identifying those subsets of customers most profitable to the business;
2. *Targeting*, determining the characteristics of profitable customers who have been captured by competitors;
3. *Market-basket analysis*, determining product purchases by consumer, which can be used for product positioning and for cross-selling.

These are not the only applications of data mining, but are three important applications useful to businesses.

---

[5] R.M. Rejesus, B.B. Little, A.C. Lovell (2004). Using data mining to detect crop insurance fraud: Is there a role for social scientists? *Journal of Financial Crime* 12:1, 24–32.

[6] G.S. Linoff (2004). Survival data mining for customer insight, *Intelligent Enterprise* 7:12, 28–33.

[7] C. Da Cunha, B. Agard, A. Kusiak (2006). Data mining for improvement of product quality, *International Journal of Production Research* 44:18/19, 4041–4054.

[8] M. O'Connell (2006). Drug safety, the U.S. Food and Drug Administration and statistical data mining, *Scientific Computing* 23:7, 32–33.

[9] ___., Data mining: Early attention to privacy in developing a key DHS program could reduce risks, *GAO Report 07-293*, 3/21/2007.

What is Data Mining?

Data mining has been called exploratory data analysis, among other things. Masses of data generated from cash registers, from scanning, from topic-specific databases throughout the company, are explored, analyzed, reduced, and reused. Searches are performed across different models proposed for predicting sales, marketing response, and profit. Classical statistical approaches are fundamental to data mining. Automated AI methods are also used. However, systematic exploration through classical statistical methods is still the basis of data mining. Some of the tools developed by the field of statistical analysis are harnessed through automatic control (with some key human guidance) in dealing with data.

A variety of analytic computer models have been used in data mining. The standard model types in data mining include regression (normal regression for prediction, logistic regression for classification), neural networks, and decision trees. These techniques are well known. This book focuses on less used techniques applied to specific problem types, to include association rules for initial data exploration, fuzzy data mining approaches, rough set models, support vector machines, and genetic algorithms. The book will also review some interesting applications in business, and conclude with a comparison of methods.

But these methods are not the only tools available for data mining. Work has continued in a number of areas, which we will describe in this book. This new work is generated because we generate ever larger data sets, express data in more complete terms, and deal with more complex forms of data. Association rules deal with large scale data sets such as those generated each day by retail organizations such as groceries. Association rules seek to identify what things go together. Research continues to enable more accurate identification of relationships when coping with massive data sets. Fuzzy representation is a way to more completely describe the uncertainty associated with concepts. Rough sets is a way to express this uncertainty in a specific probabilistic form. Support vector machines offer a way to separate data more reliably when certain forms of complexity are present in data sets. And genetic algorithms help identify better solutions for data that is in a particular form. All of these topics have interesting developments that we will try to demonstrate.

What is Needed to Do Data Mining

Data mining requires identification of a problem, along with collection of data that can lead to better understanding, and computer models to provide statistical or other means of analysis. This may be supported by visualization

tools, that display data, or through fundamental statistical analysis, such as correlation analysis.

Data mining tools need to be versatile, scalable, capable of accurately predicting responses between actions and results, and capable of automatic implementation. Versatile refers to the ability of the tool to apply a wide variety of models. Scalable tools imply that if the tools works on a small data set, it should also work on larger data sets. Automation is useful, but its application is relative. Some analytic functions are often automated, but human setup prior to implementing procedures is required. In fact, analyst judgment is critical to successful implementation of data mining. Proper selection of data to include in searches is critical. Data transformation also is often required. Too many variables produce too much output, while too few can overlook key relationships in the data. Fundamental understanding of statistical concepts is mandatory for successful data mining.

Data mining is expanding rapidly, with many benefits to business. Two of the most profitable application areas have been the use of customer segmentation by marketing organizations to identify those with marginally greater probabilities of responding to different forms of marketing media, and banks using data mining to more accurately predict the likelihood of people to respond to offers of different services offered. Many companies are using this technology to identify their blue-chip customers so that they can provide them the service needed to retain them.[10]

The casino business has also adopted data warehousing and data mining. Harrah's Entertainment Inc. is one of many casino organizations who use incentive programs.[11] About 8 million customers hold Total Gold cards, which are used whenever the customer plays at the casino, or eats, or stays, or spends money in other ways. Points accumulated can be used for complementary meals and lodging. More points are awarded for activities which provide Harrah's more profit. The information obtained is sent to the firm's corporate database, where it is retained for several years. Trump's Taj Card is used in a similar fashion. Recently, high competition has led to the use of data mining. Instead of advertising the loosest slots in town, Bellagio and Mandalay Bay have developed the strategy of promoting luxury visits. Data mining is used to identify high rollers, so that these valued customers can be cultivated. Data warehouses enable casinos to estimate the lifetime value of players. Incentive travel programs, in-house

[10] R. Hendler, F. Hendler (2004). Revenue management in fabulous Las Vegas: Combining customer relationship management and revenue management to maximize profitability, *Journal of Revenue & Pricing Management* 3:1, 73–79.

[11] G. Loveman (2003). Diamonds in the data mine, *Harvard Business Review* 81:5, 109–113.

promotions, corporate business, and customer follow-up are tools used to maintain the most profitable customers. Casino gaming is one of the richest data sets available. Very specific individual profiles can be developed. Some customers are identified as those who should be encouraged to play longer. Other customers are identified as those who are discouraged from playing. Harrah's found that 26% of its gamblers generated 82% of its revenues. They also found that their best customers were not high rollers, but rather middle-aged and senior adults who were former professionals. Harrah's developed a quantitative model to predict individual spending over the long run, and set up a program to invite back $1,000 per month customers who had not visited in 3 months. If a customer lost in a prior visit, they would be invited back to a special event.[12]

Business Data Mining

Data mining has been very effective in many business venues. The key is to find *actionable* information, or information that can be utilized in a concrete way to improve profitability. Some of the earliest applications were in retailing, especially in the form of market basket analysis. Table 1.1 shows the general application areas we will be discussing. Note that they are meant to be representative rather than comprehensive.

Table 1.1. Data mining application areas

Application area	Applications	Specifics
Retailing	Affinity positioning,	Position products effectively
	Cross-selling	Find more products for customers
Banking	Customer relationship management	Identify customer value, Develop programs to maximize revenue
Credit Card Management	Lift	Identify effective market segments
	Churn	Identify likely customer turnover
Insurance	Fraud detection	Identify claims meriting investigation
Telecommunications	Churn	Identify likely customer turnover
Telemarketing	On-line information	Aid telemarketers with easy data access
Human Resource Management	Churn	Identify potential employee turnover

[12] S. Thelen, S. Mottner, B. Berman (2004). Data mining: On the trail to marketing gold, *Business Horizons* 47:6 Nov–Dec, 25–32.

Data Mining Tools

Many good data mining software products are being used, ranging from well-established (and expensive) Enterprise Miner by SAS and Intelligent Miner by IBM, CLEMENTINE by SPSS (a little more accessible by students), PolyAnalyst by Megaputer, and many others in a growing and dynamic industry. WEKA (from the University of Waikato in New Zealand) is an open source tool with many useful developing methods. The Web site for this product (to include free download) is www.cs.waikato.ac.nz/ml/*weka*/. Each product has a well developed Web site.

Specialty products cover just about every possible profitable business application. A good source to view current products is www.KDNuggets.com. The UCI Machine Learning Repository is a source of very good data mining datasets at http://www.ics.uci.edu/~mlearn/MLOther.html.[13] That site also includes references of other good data mining sites. Vendors selling data access tools include IBM, SAS Institute Inc., Microsoft, Brio Technology Inc., Oracle, and others. IBM's Intelligent Mining Toolkit has a set of algorithms available for data mining to identify hidden relationships, trends, and patterns. SAS's System for Information Delivery integrates executive information systems, statistical tools for data analysis, and neural network tools.

Summary

This chapter has introduced the topic of data mining, focusing on business applications. Data mining has proven to be extremely effective in improving many business operations. The process of data mining relies heavily on information technology, in the form of data storage support (data warehouses, data marts, and or on-line analytic processing tools) as well as software to analyze the data (data mining software). However, the process of data mining is far more than simply applying these data mining software tools to a firm's data. Intelligence is required on the part of the analyst in selection of model types, in selection and transformation of the data relating to the specific problem, and in interpreting results.

[13] C.J. Merz, P.M. Murphy. *UCI Repository of Machine Learning Databases*. http://www.ics.uci.edu/~mlearn/MLOther.html Irvine, CA: University of California, Department of Information and Computer Science.

2 Data Mining Process

In order to systematically conduct data mining analysis, a general process is usually followed. There are some standard processes, two of which are described in this chapter. One (CRISP) is an industry standard process consisting of a sequence of steps that are usually involved in a data mining study. The other (SEMMA) is specific to SAS. While each step of either approach isn't needed in every analysis, this process provides a good coverage of the steps needed, starting with data exploration, data collection, data processing, analysis, inferences drawn, and implementation.

CRISP-DM

There is a Cross-Industry Standard Process for Data Mining (CRISP-DM) widely used by industry members. This model consists of six phases intended as a cyclical process (see Fig. 2.1):

- *Business Understanding* Business understanding includes determining business objectives, assessing the current situation, establishing data mining goals, and developing a project plan.
- *Data Understanding* Once business objectives and the project plan are established, data understanding considers data requirements. This step can include initial data collection, data description, data exploration, and the verification of data quality. Data exploration such as viewing summary statistics (which includes the visual display of categorical variables) can occur at the end of this phase. Models such as cluster analysis can also be applied during this phase, with the intent of identifying patterns in the data.
- *Data Preparation* Once the data resources available are identified, they need to be selected, cleaned, built into the form desired, and formatted. Data cleaning and data transformation in preparation of data modeling needs to occur in this phase. Data exploration at a greater depth can be applied during this phase, and additional models utilized, again providing the opportunity to see patterns based on business understanding.

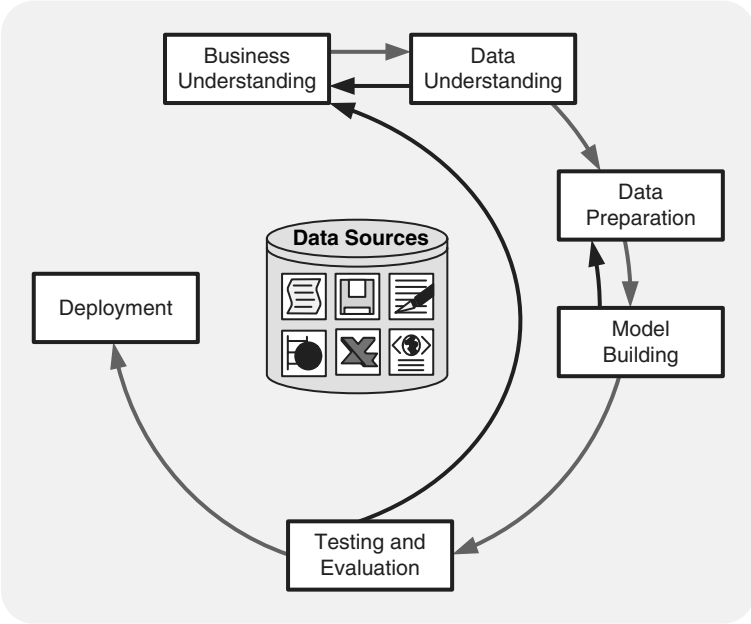


Fig. 2.1. CRISP-DM process

- *Modeling* Data mining software tools such as visualization (plotting data and establishing relationships) and cluster analysis (to identify which variables go well together) are useful for initial analysis. Tools such as generalized rule induction can develop initial association rules. Once greater data understanding is gained (often through pattern recognition triggered by viewing model output), more detailed models appropriate to the data type can be applied. The division of data into training and test sets is also needed for modeling.
- *Evaluation* Model results should be evaluated in the context of the business objectives established in the first phase (business understanding). This will lead to the identification of other needs (often through pattern recognition), frequently reverting to prior phases of CRISP-DM. Gaining business understanding is an iterative procedure in data mining, where the results of various visualization, statistical, and artificial intelligence tools show the user new relationships that provide a deeper understanding of organizational operations.
- *Deployment* Data mining can be used to both verify previously held hypotheses, or for knowledge discovery (identification of unexpected and useful relationships). Through the knowledge discovered in the earlier phases of the CRISP-DM process, sound models can be obtained

that may then be applied to business operations for many purposes, including prediction or identification of key situations. These models need to be monitored for changes in operating conditions, because what might be true today may not be true a year from now. If significant changes do occur, the model should be redone. It's also wise to record the results of data mining projects so documented evidence is available for future studies.

This six-phase process is not a rigid, by-the-numbers procedure. There's usually a great deal of backtracking. Additionally, experienced analysts may not need to apply each phase for every study. But CRISP-DM provides a useful framework for data mining.

Business Understanding

The key element of a data mining study is knowing what the study is for. This begins with a managerial need for new knowledge, and an expression of the business objective regarding the study to be undertaken. Goals in terms of things such as "What types of customers are interested in each of our products?" or "What are typical profiles of our customers, and how much value do each of them provide to us?" are needed. Then a plan for finding such knowledge needs to be developed, in terms of those responsible for collecting data, analyzing data, and reporting. At this stage, a budget to support the study should be established, at least in preliminary terms.

In customer segmentation models, such as Fingerhut's retail catalog business, the identification of a business purpose meant identifying the type of customer that would be expected to yield a profitable return. The same analysis is useful to credit card distributors. For business purposes, grocery stores often try to identify which items tend to be purchased together so it can be used for affinity positioning within the store, or to intelligently guide promotional campaigns. Data mining has many useful business applications, some of which will be presented throughout the course of the book.

Data Understanding

Since data mining is task-oriented, different business tasks require different sets of data. The first stage of the data mining process is to select the related data from many available databases to correctly describe a given business task. There are at least three issues to be considered in the data selection. The first issue is to set up a concise and clear description of the problem. For example, a retail data-mining project may seek to identify

spending behaviors of female shoppers who purchase seasonal clothes. Another example may seek to identify bankruptcy patterns of credit card holders. The second issue would be to identify the relevant data for the problem description. Most demographical, credit card transactional, and financial data could be relevant to both retail and credit card bankruptcy projects. However, gender data may be prohibited for use by law for the latter, but be legal and prove important for the former. The third issue is that selected variables for the relevant data should be independent of each other. Variable independence means that the variables do not contain overlapping information. A careful selection of independent variables can make it easier for data mining algorithms to quickly discover useful knowledge patterns.

Data sources for data selection can vary. Normally, types of data sources for business applications include *demographic data* (such as income, education, number of households, and age), *socio-graphic data* (such as hobby, club membership, and entertainment), *transactional data* (sales records, credit card spending, issued checks), and so on. The data type can be categorized as quantitative and qualitative data. *Quantitative data* is measurable using numerical values. It can be either discrete (such as integers) or continuous (such as real numbers). *Qualitative data*, also known as categorical data, contains both nominal and ordinal data. Nominal data has finite non-ordered values, such as gender data which has two values: male and female. Ordinal data has finite ordered values. For example, customer credit ratings are considered ordinal data since the ratings can be excellent, fair, and bad. Quantitative data can be readily represented by some sort of probability distribution. A probability distribution describes how the data is dispersed and shaped. For instance, normally distributed data is symmetric, and is commonly referred to as bell-shaped. Qualitative data may be first coded to numbers and then be described by frequency distributions. Once relevant data are selected according to the data mining business objective, data preprocessing should be pursued.

Data Preparation

The purpose of data preprocessing is to clean selected data for better quality. Some selected data may have different formats because they are chosen from different data sources. If selected data are from flat files, voice message, and web text, they should be converted to a consistent electronic format. In general, data cleaning means to filter, aggregate, and fill in missing values (*imputation*). By filtering data, the selected data are examined for outliers and redundancies. Outliers differ greatly from the majority

of data, or data that are clearly out of range of the selected data groups. For example, if the income of a customer included in the middle class is $250,000, it is an error and should be taken out from the data mining project that examines the various aspects of the middle class. Outliers may be caused by many reasons, such as human errors or technical errors, or may naturally occur in a data set due to extreme events. Suppose the age of a credit card holder is recorded as "12." This is likely a human error. However, there might actually be an independently wealthy pre-teen with important purchasing habits. Arbitrarily deleting this outlier could dismiss valuable information.

Redundant data are the same information recorded in several different ways. Daily sales of a particular product are redundant to seasonal sales of the same product, because we can derive the sales from either daily data or seasonal data. By aggregating data, data dimensions are reduced to obtain aggregated information. Note that although an aggregated data set has a small volume, the information will remain. If a marketing promotion for furniture sales is considered in the next 3 or 4 years, then the available daily sales data can be aggregated as annual sales data. The size of sales data is dramatically reduced. By smoothing data, missing values of the selected data are found and new or reasonable values then added. These added values could be the average number of the variable (mean) or the mode. A missing value often causes no solution when a data-mining algorithm is applied to discover the knowledge patterns.

Data can be expressed in a number of different forms. For instance, in CLEMENTINE, the following data types can be used.

- *RANGE* Numeric values (integer, real, or date/time).
- *FLAG* Binary – Yes/No, 0/1, or other data with two outcomes (text, integer, real number, or date/time).
- *SET* Data with distinct multiple values (numeric, string, or date/time).
- *TYPELESS* For other types of data.

Usually we think of data as real numbers, such as age in years or annual income in dollars (we would use RANGE in those cases). Sometimes variables occur as either/or types, such as having a driver's license or not, or an insurance claim being fraudulent or not. This case could be dealt with using real numeric values (for instance, 0 or 1). But it's more efficient to treat them as FLAG variables. Often, it's more appropriate to deal with categorical data, such as age in terms of the set {young, middle-aged, elderly}, or income in the set {low, middle, high}. In that case, we could group the data and assign the appropriate category in terms of a string,

using a set. The most complete form is RANGE, but sometimes data does not come in that form so analysts are forced to use SET or FLAG types. Sometimes it may actually be more accurate to deal with SET data types than RANGE data types.

As another example, PolyAnalyst has the following data types available:

- *Numerical* Continuous values
- *Integer* Integer values
- *Yes/no* Binary data
- *Category* A finite set of possible values
- *Date*
- *String*
- *Text*

Each software tool will have a different data scheme, but the primary types of data dealt with are represented in these two lists.

There are many statistical methods and visualization tools that can be used to preprocess the selected data. Common statistics, such as max, min, mean, and mode can be readily used to aggregate or smooth the data, while scatter plots and box plots are usually used to filter outliers. More advanced techniques (including regression analyses, cluster analysis, decision tree, or hierarchical analysis) may be applied in data preprocessing depending on the requirements for the quality of the selected data. Because data preprocessing is detailed and tedious, it demands a great deal of time. In some cases, data preprocessing could take over 50% of the time of the entire data mining process. Shortening data processing time can reduce much of the total computation time in data mining. The simple and standard data format resulting from data preprocessing can provide an environment of information sharing across different computer systems, which creates the flexibility to implement various data mining algorithms or tools.

As an important component of data preparation, data transformation is to use simple mathematical formulations or learning curves to convert different measurements of selected, and clean, data into a unified numerical scale for the purpose of data analysis. Many available statistics measurements, such as mean, median, mode, and variance can readily be used to transform the data. In terms of the representation of data, data transformation may be used to (1) transform from numerical to numerical scales, and (2) recode categorical data to numerical scales. For numerical to numerical scales, we can use a mathematical transformation to “shrink” or “enlarge” the given data. One reason for transformation is to eliminate differences in variable scales. For example, if the attribute “salary” ranges from

"$20,000" to "$100,000," we can use the formula $S = (x - min)/(max - min)$ to "shrink" any known salary value, say $50,000 to 0.6, a number in [0.0, 1.0]. If the mean of salary is given as $45,000, and standard deviation is given as $15,000, the $50,000 can be normalized as 0.33. Transforming data from the metric system (e.g., meter, kilometer) to English system (e.g., foot and mile) is another example. For categorical to numerical scales, we have to assign an appropriate numerical number to a categorical value according to needs. Categorical variables can be ordinal (such as less, moderate, and strong) and nominal (such as red, yellow, blue, and green). For example, a binary variable {yes, no} can be transformed into "1 = yes and 0 = no." Note that transforming a numerical value to an ordinal value means transformation with order, while transforming to a nominal value is a less rigid transformation. We need to be careful not to introduce more precision than is present in the original data. For instance, Likert scales often represent ordinal information with coded numbers (1–7, 1–5, and so on). However, these numbers usually don't imply a common scale of difference. An object rated as 4 may not be meant to be twice as strong on some measure as an object rated as 2. Sometimes, we can apply values to represent a block of numbers or a range of categorical variables. For example, we may use "1" to represent the monetary values from "$0" to "$20,000," and use "2" for "$20,001–$40,000," and so on. We can use "0001" to represent "two-store house" and "0002" for "one-and-half-store house." All kinds of "quick-and-dirty" methods could be used to transform data. There is no unique procedure and the only criterion is to transform the data for convenience of use during the data mining stage.

Modeling

Data modeling is where the data mining software is used to generate results for various situations. A cluster analysis and visual exploration of the data are usually applied first. Depending upon the type of data, various models might then be applied. If the task is to group data, and the groups are given, discriminant analysis might be appropriate. If the purpose is estimation, regression is appropriate if the data is continuous (and logistic regression if not). Neural networks could be applied for both tasks.

Decision trees are yet another tool to classify data. Other modeling tools are available as well. We'll cover these different models in greater detail in subsequent chapters. The point of data mining software is to allow the user to work with the data to gain understanding. This is often fostered by the iterative use of multiple models.

Data Treatment

Data mining is essentially the analysis of statistical data, usually using enormous data sets. The standard process of data mining is to take this large set of data and divide it, using a portion of the data (the *training set*) for development of the model (no matter what modeling technique is used), and reserving a portion of the data (the *test set*) for testing the model that's built. In some applications a third split of data (*validation set*) is used to estimate parameters from the data. The principle is that if you build a model on a particular set of data, it will of course test quite well. By dividing the data and using part of it for model development, and testing it on a separate set of data, a more convincing test of model accuracy is obtained.

This idea of splitting the data into components is often carried to additional levels in the practice of data mining. Further portions of the data can be used to refine the model.

Data Mining Techniques

Data mining can be achieved by Association, Classification, Clustering, Predictions, Sequential Patterns, and Similar Time Sequences.[1]

In *Association*, the relationship of a particular item in a data transaction on other items in the same transaction is used to predict patterns. For example, if a customer purchases a laptop PC (X), then he or she also buys a mouse (Y) in 60% of the cases. This pattern occurs in 5.6% of laptop PC purchases. An association rule in this situation can be "X implies Y, where 60% is the confidence factor and 5.6% is the support factor." When the confidence factor and support factor are represented by linguistic variables "high" and "low," respectively, the association rule can be written in the fuzzy logic form, such as: "where the support factor is low, X implies Y is high." In the case of many qualitative variables, fuzzy association is a necessary and promising technique in data mining.

In *Classification*, the methods are intended for learning different functions that map each item of the selected data into one of a predefined set of classes. Given the set of predefined classes, a number of attributes, and a "learning (or training) set," the classification methods can automatically predict the class of other unclassified data of the learning set. Two key research problems related to classification results are the evaluation of misclassification and prediction power. Mathematical techniques that are often used to construct classification methods are binary decision trees, neural networks, linear programming, and statistics. By using binary

[1] D.L. Olson, Yong Shi (2007). *Introduction to Business Data Mining*, Boston: McGraw-Hill/Irwin.

decision trees, a tree induction model with a "Yes–No" format can be built to split data into different classes according to its attributes. Models fit to data can be measured by either statistical estimation or information entropy. However, the classification obtained from tree induction may not produce an optimal solution where prediction power is limited. By using neural networks, a neural induction model can be built. In this approach, the attributes become input layers in the neural network while the classes associated with data are output layers. Between input layers and output layers, there are a larger number of hidden layers processing the accuracy of the classification. Although the neural induction model often yields better results in many cases of data mining, since the relationships involve complex nonlinear relationships, implementing this method is difficult when there's a large set of attributes. In linear programming approaches, the classification problem is viewed as a special form of linear program. Given a set of classes and a set of attribute variables, one can define a cut-off limit (or boundary) separating the classes. Then each class is represented by a group of constraints with respect to a boundary in the linear program. The objective function in the linear programming model can minimize the overlapping rate across classes and maximize the distance between classes. The linear programming approach results in an optimal classification. However, the computation time required may exceed that of statistical approaches. Various statistical methods, such as linear discriminant regression, quadratic discriminant regression, and logistic discriminant regression are very popular and are commonly used in real business classifications. Even though statistical software has been developed to handle a large amount of data, statistical approaches have a disadvantage in efficiently separating multiclass problems in which a pair-wise comparison (i.e., one class versus the rest of the classes) has to be adopted.

Cluster analysis takes ungrouped data and uses automatic techniques to put this data into groups. Clustering is unsupervised, and does not require a learning set. It shares a common methodological ground with Classification. In other words, most of the mathematical models mentioned earlier in regards to Classification can be applied to Cluster Analysis as well.

Prediction analysis is related to regression techniques. The key idea of prediction analysis is to discover the relationship between the dependent and independent variables, the relationship between the independent variables (one versus Another, one versus the rest, and so on). For example, if sales is an independent variable, then profit may be a dependent variable. By using historical data from both sales and profit, either linear or nonlinear regression techniques can produce a fitted regression curve that can be used for profit prediction in the future.

Sequential Pattern analysis seeks to find similar patterns in data transaction over a business period. These patterns can be used by business analysts to identify relationships among data. The mathematical models behind Sequential Patterns are logic rules, fuzzy logic, and so on. As an extension of Sequential Patterns, *Similar Time Sequences* are applied to discover sequences similar to a known sequence over both past and current business periods. In the data mining stage, several similar sequences can be studied to identify future trends in transaction development. This approach is useful in dealing with databases that have time-series characteristics.

Evaluation

The data interpretation stage is very critical. It assimilates knowledge from mined data. Two issues are essential. One is how to recognize the business value from knowledge patterns discovered in the data mining stage. Another issue is which visualization tool should be used to show the data mining results. Determining the business value from discovered knowledge patterns is similar to playing "puzzles." The mined data is a puzzle that needs to be put together for a business purpose. This operation depends on the interaction between data analysts, business analysts and decision makers (such as managers or CEOs). Because data analysts may not be fully aware of the purpose of the data mining goal or objective, and while business analysts may not understand the results of sophisticated mathematical solutions, interaction between them is necessary. In order to properly interpret knowledge patterns, it's important to choose an appropriate visualization tool. Many visualization packages and tools are available, including pie charts, histograms, box plots, scatter plots, and distributions. Good interpretation leads to productive business decisions, while poor interpretation analysis may miss useful information. Normally, the simpler the graphical interpretation, the easier it is for end users to understand.

Deployment

The results of the data mining study need to be reported back to project sponsors. The data mining study has uncovered new knowledge, which needs to be tied to the original data mining project goals. Management will then be in a position to apply this new understanding of their business environment.

It is important that the knowledge gained from a particular data mining study be monitored for change. Customer behavior changes over time, and what was true during the period when the data was collected may have already change. If fundamental changes occur, the knowledge uncovered is

no longer true. Therefore, it's critical that the domain of interest be monitored during its period of deployment.

SEMMA

In order to be applied successfully, the data mining solution must be viewed as a process rather than a set of tools or techniques. In addition to the CRISP-DM there is yet another well-known methodology developed by the SAS Institute, called SEMMA. The acronym SEMMA stands for *s*ample, *e*xplore, *m*odify, *m*odel, *a*ssess. Beginning with a statistically representative sample of your data, SEMMA intends to make it easy to apply exploratory statistical and visualization techniques, select and transform the most significant predictive variables, model the variables to predict outcomes, and finally confirm a model's accuracy. A pictorial representation of SEMMA is given in Fig. 2.2.

By assessing the outcome of each stage in the SEMMA process, one can determine how to model new questions raised by the previous results, and thus proceed back to the exploration phase for additional refinement of the data. That is, as is the case in CRISP-DM, SEMMA also driven by a highly iterative experimentation cycle.

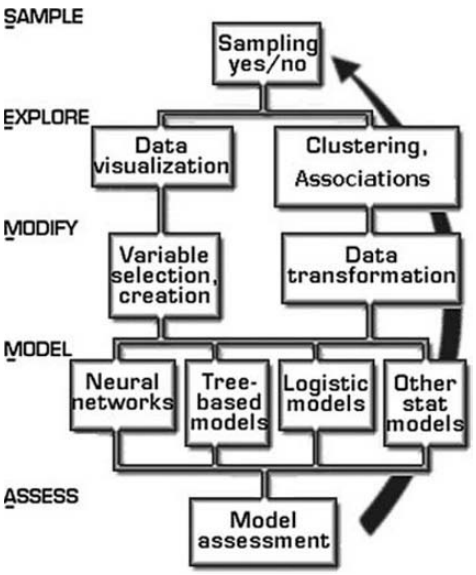


Fig. 2.2. Schematic of SEMMA (original from SAS Institute)

Steps in SEMMA Process

Step 1 (Sample): This is where a portion of a large data set (big enough to contain the significant information yet small enough to manipulate quickly) is extracted. For optimal cost and computational performance, some (including the SAS Institute) advocates a sampling strategy, which applies a reliable, statistically representative sample of the full detail data. In the case of very large datasets, mining a representative sample instead of the whole volume may drastically reduce the processing time required to get crucial business information. If general patterns appear in the data as a whole, these will be traceable in a representative sample. If a niche (a rare pattern) is so tiny that it is not represented in a sample and yet so important that it influences the big picture, it should be discovered using exploratory data description methods. It is also advised to create partitioned data sets for better accuracy assessment.

- Training – used for model fitting.
- Validation – used for assessment and to prevent over fitting.
- Test – used to obtain an honest assessment of how well a model generalizes.

Step 2 (Explore): This is where the user searched for unanticipated trends and anomalies in order to gain a better understanding of the data set. After sampling your data, the next step is to explore them visually or numerically for inherent trends or groupings. Exploration helps refine and redirect the discovery process. If visual exploration does not reveal clear trends, one can explore the data through statistical techniques including factor analysis, correspondence analysis, and clustering. For example, in data mining for a direct mail campaign, clustering might reveal groups of customers with distinct ordering patterns. Limiting the discovery process to each of these distinct groups individually may increase the likelihood of exploring richer patterns that may not be strong enough to be detected if the whole dataset is to be processed together.

Step 3 (Modify): This is where the user creates, selects, and transforms the variables upon which to focus the model construction process. Based on the discoveries in the exploration phase, one may need to manipulate data to include information such as the grouping of customers and significant subgroups, or to introduce new variables. It may also be necessary to look for outliers and reduce the number of variables, to narrow them down to the most significant ones. One may also need to modify data when the “mined”

data change. Because data mining is a dynamic, iterative process, you can update data mining methods or models when new information is available.

Step 4 (Model): This is where the user searches for a variable combination that reliably predicts a desired outcome. Once you prepare your data, you are ready to construct models that explain patterns in the data. Modeling techniques in data mining include artificial neural networks, decision trees, rough set analysis, support vector machines, logistic models, and other statistical models – such as time series analysis, memory-based reasoning, and principal component analysis. Each type of model has particular strengths, and is appropriate within specific data mining situations depending on the data. For example, artificial neural networks are very good at fitting highly complex nonlinear relationships while Rough sets analysis is know to produce reliable results with uncertain and imprecise problem situations.

Step 5 (Assess): This is where the user evaluates the usefulness and the reliability of findings from the data mining process. In this final step of the data mining process user assesses the models to estimate how well it performs. A common means of assessing a model is to apply it to a portion of data set put aside (and not used during the model building) during the sampling stage. If the model is valid, it should work for this reserved sample as well as for the sample used to construct the model. Similarly, you can test the model against known data. For example, if you know which customers in a file had high retention rates and your model predicts retention, you can check

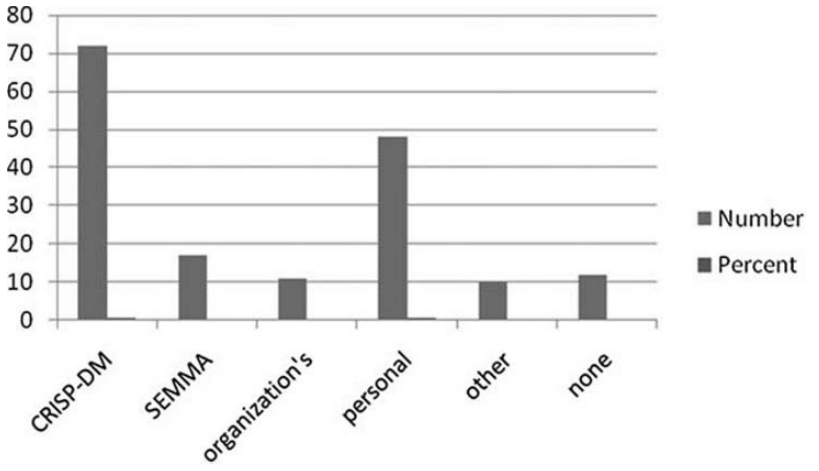


Fig. 2.3. Poll results – data mining methodology (conducted by KDNuggets.com on April 2004)

to see whether the model selects these customers accurately. In addition, practical applications of the model, such as partial mailings in a direct mail campaign, help prove its validity. The data mining web-site KDNuggets provided the data shown in Fig. 2.3 concerning relative use of data mining methodologies.

The SEMMA approach is completely compatible with the CRISP approach. Both aid the knowledge discovery process. Once models are obtained and tested, they can then be deployed to gain value with respect to business or research application.

Example Data Mining Process Application

Nayak and Qiu (2005) demonstrated the data mining process in an Australian software development project.[2] We will first relate their reported process, and then compare this with the CRISP and SEMMA frameworks.

The project owner was an international telecommunication company which undertook over 50 software projects annually. Processes were organized for Software Configuration Management, Software Risk Management, Software Project Metric Reporting, and Software Problem Report Management. Nayak and Qiu were interested in mining the

Table 2.1. Selected attributes from problem reports

Attribute	Description
Synopsis	Main issues
Responsibility	Individuals assigned
Confidentiality	Yes or no
Environment	Windows, Unix, etc.
Release note	Fixing comment
Audit trail	Process progress
Arrival date	
Close date	
Severity	Text describing the bug and impact on system
Priority	High, Medium, Low
State	Open, Active, Analysed, Suspended, Closed, Resolved, Feedback
Class	Sw-bug, Doc-bug, Change-request, Support, Mistaken, Duplicate

[2] R. Nayak, Tian Qiu (2005). A data mining application: Analysis of problems occurring during a software project development process, *International Journal of Software Engineering* 15:4, 647–663.

data from the Software Problem Reports. All problem reports were collected throughout the company (over 40,000 reports). For each report, data was available to include data shown in Table 2.1:

The data mining process reported included goal definition, data preprocessing, data modeling, and analysis of results.

1. Goal Definition

Data mining was expected to be useful in two areas. The first involved the early estimation and planning stage of a software project, company engineers have to estimate the number of lines of code, the kind of documents to be delivered, and estimated times. Accuracy at this stage would vastly improve project selection decisions. Little tool support was available for these activities, and estimates of these three attributes were based on experience supported by statistics on past projects. Thus projects involving new types of work were difficult to estimate with confidence. The second area of data mining application concerned the data collection system, which had limited information retrieval capability. Data was stored in flat files, and it was difficult to gather information related to specific issues.

2. Data Pre-Processing

This step consisted of attribute selection, data cleaning, and data transformation.

Data Field Selection: Some of the data was not pertinent to the data mining exercise, and was ignored. Of the variables given in Table 2.1, Confidentiality, Environment, Release note, and Audit trail were ignored as having no data mining value. They were, however, used during preprocessing and post-processing to aid in data selection and gaining better understanding of rules generated. For data stability, only problem reports for State values of Closed were selected.

Whenever a problem report was created, the project leader had to determine how long the fix took, how many people were involved, customer impact severity, impact on cost and schedule, and type of problem (software bug or design flaw). Thus the attributes listed below were selected as most important:

- Severity
- Priority
- Class

- Arrival-Date
- Close-Date
- Responsible
- Synopsis

The first five attributes had fixed values, and the Responsible attribute was converted to a count of those assigned to the problem. All of these attributes could be dealt with through conventional data mining tools. Synopsis was text data requiring text mining. Class was selected as the target attribute, with the possible outcomes given in Table 2.2:

Data Cleaning: Cleaning involved identification of missing, inconsistent, or mistaken values. Tools used in this process step included graphical tools to provide a picture of distributions, and statistics such as maxima, minima, mean values, and skew. Some entries were clearly invalid, caused by either human error or the evolution of the problem reporting system. For instance, over time, input for the Class attribute changed from SW-bug to sw-bug. Those errors that were correctable were corrected. If all errors detected for a report were not corrected, that report was discarded from the study.

Data Transformation: The attributes Arrival-Date and Close-Date were useful in this study to calculate the duration. Additional information was required, to include time zone. The Responsible attribute contained information identified how many people were involved. An attribute Time-to-fix was created multiplying the duration times the number of people, and then categorized into discrete values of 1 day, 3 days, 7 days, 14 days, 30 days, 90 days, 180 days, and 360 days (representing over one person-year).

In this application, 11,000 of the original 40,000 problem reports were left. They came from over 120 projects completed over the period 1996–2000. Four attributes were obtained:

Table 2.2. Class outcomes

Sw-bug	Bug from software code implementation
Doc-bug	Bug from documents directly related to the software product
Change-request	Customer enhancement request
Support	Bug from tools or documents, not the software product itself
Mistaken	Error in either software or document
Duplicate	Problem already covered in another problem report

- Time-to-fix
- Class
- Severity
- Priority

Text-mining was applied to 11,364 records, of which 364 had no time values so 11,000 were used for conventional data mining classification.

3. Data Modeling

Data mining provides functionality not provided by general database query techniques, which can't deal with the large number of records with high dimensional structures. Data mining provided useful functionality to answer questions such as the type of project documents requiring a great deal of development team time for bug repair, or the impact for various attribute values of synopsis, severity, priority, and class. A number of data mining tools were used.

- Prediction modeling was useful for evaluation of time consumption, giving sounder estimates for project estimation and planning.
- Link analysis was useful in discovering associations between attribute values.
- Text mining was useful in analyzing the Synopsis field.

Data mining software CBA was used for both classification and association rule analysis, C5 for classification, and TextAnalyst for text mining. An example classification rule was:

IF Severity non-critical AND Priority medium
THEN Class is Document with 70.72% confidence with support value of 6.5%

There were 352 problem reports in the training data set having these conditions, but only 256 satisfied the rule's conclusion.

Another rule including time-to-fix was more stringent:

IF 21 ≤ time-to-fix ≤ 108
AND Severity non-critical AND Priority medium
THEN Class is Document with 82.70% confidence with support value of 2.7%

There were 185 problem reports in the training data set with these conditions, 153 of which satisfied the rule's conclusion.

4. Analysis of Results

Classification and Association Rule Mining: Data was stratified using choice-based sampling rather than random sampling. This provided an equal number of samples for each target attribute field value. This improved the probability of obtaining rules for groups with small value counts (thus balancing the data). Three different training sets of varying size were generated. The first data set included 1,224 problem reports from one software project. The second data set consisted of equally distributed values from 3,400 problem reports selected from all software projects. The third data set consisted of 5,381 problem reports selected from all projects.

Minimum support and confidence were used to control rule modeling. Minimum support is a constraint requiring at least the stated number of cases be present in the training set. A high minimum support will yield fewer rules. Confidence is the strength of a rule as measured by the correct classification of cases. In practice, these are difficult to set ahead of analysis, and thus combinations of minimum support and confidence were used.

In this application, it was difficult for the CBA software to obtain correct classification on test data above 50%. The use of equal density of cases was not found to yield more accurate models in this study, although it appears a rational approach for further investigation. Using multiple support levels was also not found to improve error rates, and single support mining yielded a smaller number of rules. However, useful rules were obtained.

C5 was also applied for classification mining. C5 used cross validation, which splits the dataset into subsets (folds), treating each fold as a test case and the rest as training sets in hopes of finding a better result than a single training set process. C5 also has a boosting option, which generates and combines multiple classifiers in efforts to improve predictive accuracy. Here C5 yielded larger rule sets, with slightly better fits with training data, although at roughly the same level. Cross validation and boosting would not yield additional rules, but would focus on more accurate rules.

Text Mining: Pure text for the Synopsis attribute was categorized into a series of specific document types, such as “SRS – missing requirements” (with SRS standing for software requirement specification), “SRS – ability to turn off sending of SOH”, “Changes needed to SCMP_2.0.0” and so forth. TextAnalyst was used. This product builds a semantic network for text data investigation. Each element in the semantic network is assigned a

weight value, and relationships to other elements in the network, which are also assigned a weight value. Users are not required to specify predefined rules to build the semantic network. TextAnalyst provided a semantic network tree containing the most important words or word combinations (concepts), and reported relations and weights among these concepts ranging from 0 to 100, roughly analogous to probability. Text mining was applied to 11,226 cases.

Comparison of CRISP & SEMMA

The Nayak and Qiu case demonstrates a data mining process for a specific application, involving interesting aspects of data cleaning and transformation requirements, as well as a wide variety of data types, to include text. CRISP and SEMMA were created as broad frameworks, which need to be adapted to specific circumstances (see Table 2.3). We will now review how the Nayak and Qiu case fits these frameworks.

Nayak and Qiu started off with a clearly stated set of goals – to develop tools that would better utilize the wealth of data in software project problem reports.

They examined data available, and identified what would be useful. Much of the information from the problem reports was discarded. SEMMA includes sampling efforts here, which CRISP would include in data preparation, and which Nayak and Qiu accomplished after data transformation. Training and test sets were used as part of the software application.

Table 2.3. Comparison of methods

CRISP	SEMMA	Nayak & Qiu
Business understanding	Assumes well-defined question	Goals were defined Develop tools to better utilize problem reports
Data understanding	*S*ample *E*xplore	Looked at data in problem reports
Data preparation	*M*odify data	Data pre-processing Data cleaning Data transformation
Modeling	*M*odel	Data modeling
Evaluation	*A*ssess	Analyzing results
Deployment		

Data was cleaned, and reports with missing observations were discarded from the study. Data preparation involved data transformation. Specifically, they used two problem report attributes to generate project duration, which was further transformed by multiplying by the number of people assigned (available by name, but only counts were needed). The resultant measure of effort was further transformed into categories that reflected relative importance without cluttering detail.

Modeling included classification and association rule analysis from the first software tool (CBA), a replication of classification with C5, and independent text analysis with TextAnalyst. Nayak and Qiu generated a variety of models by manipulating minimum support and confidence levels in the software.

Evaluation (assessment) was accomplished by Nayak and Qiu through analysis of results in terms of the number of rules, as well as accuracy of classification models as applied to the test set.

CRISP addresses the deployment of data mining models, which is implicit in any study. Nayak and Qiu's models were presumably deployed, but that was not addressed in their report.

Handling Data

A recent data mining study in insurance applied a knowledge discovery process.[3] This process involved iteratively applying the steps that we covered in CRISP-DM, and demonstrating how the methodology can work in practice.

Stage 1. Business Understanding

A model was needed to predict which customers would be insolvent early enough for the firm to take preventive measures (or measures to avert losing good customers). This goal included minimizing the misclassification of legitimate customers.

In this case, the billing period was 2 months. Customers used their phone for 4 weeks, and received bills about 1 week later. Payment was due a month after the date of billing. In the industry, companies typically gave customers about 2 weeks after the due-date before taking action, at which time the phone was disconnected if the unpaid bill was greater than a set

[3] S. Daskalaki, I. Kopanas, M. Goudara, N. Avouris (2003). Data mining for decision support on customer insolvency in the telecommunications business, *European Journal of Operational Research* 145, 239–255.

amount. Bills were sent every month for another 6 months, during which period the late customer could make payment arrangements. If no payment was received at the end of this 6-month period, the unpaid balance was transferred to the uncollectible category.

This study hypothesized that insolvent customers would change their calling habits and phone usage during a critical period before and immediately after termination of the billing period. Changes in calling habits, combined with paying patterns were tested for their ability to provide sound predictions of future insolvencies.

Stage 2. Data Understanding

Static customer information was available from customer files. Time-dependent data was available on bills, payments, and usage. Data came from several databases, but all of these databases were internal to the company. A data warehouse was built to gather and organize this data. The data was coded to protect customer privacy. Data included customer information, phone usage from switching centers, billing information, payment reports by customer, phone disconnections due to a failure to pay, phone reconnections after payment, and reports of permanent contract nullifications.

Data was selected for 100,000 customers covering a 17-month period, and was collected from one rural/agricultural region of customers, a semi-rural touring area, and an urban/industrial area in order to assure representative cross-sections of the company's customer base. The data warehouse used over 10 gigabytes of storage for raw data.

Stage 3. Data Preparation

The data was tested for quality, and data that wasn't useful for the study was filtered out. Heterogeneous data items were interrelated. As examples, it was clear that inexpensive calls had little impact on the study. This allowed a 50% reduction in the total volume of data. The low percentage of fraudulent cases made it necessary to clean the data from missing or erroneous values due to different recording practices within the organization and the dispersion of data sources. Thus it was necessary to cross-check data such as phone disconnections. The lagged data required synchronization of different data elements.

Data synchronization revealed a number of insolvent customers with missing information that had to be deleted from the data set. It was thus necessary to reduce and project data, so information was grouped by account to make data manipulation easier, and customer data was aggregated

by 2-week periods. Statistics were applied to find characteristics that were discriminant factors for solvent versus insolvent customers. Data included the following:

- Telephone account category (23 categories, such as payphone, business, and so on).
- Average amount owed was calculated for all solvent and insolvent customers. Insolvent customers had significantly higher averages across all categories of account.
- Extra charges on bills were identified by comparing total charges for phone usage for the period as opposed to balances carried forward or purchases of hardware or other services. This also proved to be statistically significant across the two outcome categories.
- Payment by installments was investigated. However, this variable was not found to be statistically significant.

Stage 4. Modeling

The prediction problem was classification, with two classes: most possibly solvent (99.3% of the cases) and most possibly insolvent (0.7% of the cases). Thus, the count of insolvent cases was very small in a given billing period. The costs of error varied widely in the two categories. This has been noted by many as a very difficult classification problem.

A new dataset was created through stratified sampling for solvent customers, altering the distribution of customers to be 90% solvent and 10% insolvent. All of the insolvent cases were retained, while care was taken to maintain a proportional representation of the solvent set of data over variables such as geographical region. A dataset of 2,066 total cases was developed.

A critical period for each phone account was established. For those accounts that were nullified, this critical period was the last 15 two-week periods prior to service interruption. For accounts that remained active, the critical period was set as a similar period to possible disruption. There were six possible disruption dates per year. For the active accounts, one of these six dates was selected at random.

For each account, variables were defined by counting the appropriate measure for every 2-week period in the critical period for that observation. At the end of this phase, new variables were created to describe phone usage by account compared to a moving average of four previous 2-week periods. At this stage, there were 46 variables as candidate discriminating factors. These variables included 40 variables measured as call habits over

15 two-week periods, as well as variables concerning the type of customer, whether or not a customer was new, and four variables relating to customer bill payment.

Discriminant analysis, decision trees and neural network algorithms were used to test hypotheses over the reduced data set of 2,066 cases measured over 46 variables. Discriminant analysis yielded a linear model, the neural network came out as a nonlinear model, and the decision tree was a rule-based classifier.

Stage 5. Evaluation

Experiments were conducted to test and compare performance. The data set was divided into a training set (about two-thirds of the 2,066 cases) and test set (the remaining cases). Classification errors are commonly displayed in *coincidence matrices* (called confusion matrices by some). A coincidence matrix shows the count of cases correctly classified, as well as the count of cases classified in each incorrect category. But in many data mining studies, the model may be very good at classifying one category, while very poor at classifying another category. The primary value of the coincidence matrix is that it identifies what kinds of errors are made. It may be much more important to avoid one kind of error than another. For instance, a bank loan officer suffers a great deal more from giving a loan to someone who's expected to repay and does not than making the mistake of not giving a loan to an applicant who actually would have paid. Both instances would be classification errors, but in data mining, often one category of error is much more important than another. Coincidence matrices provide a means of focusing on what kinds of errors particular models tend to make.

A way to reflect relative error importance is through cost. This is a relatively simple idea, allowing the user to assign relative costs by type of error. For instance, if our model predicted that an account was insolvent, that might involve an average write-off of $200. On the other hand, waiting for an account that ultimately was repaid might involve a cost of $10. Thus, there would be a major difference in the cost of errors in this case. Treating a case that turned out to be repaid as a dead account would risk the loss of $190, in addition to alienating the customer (which may or may not have future profitability implications). Conversely, treating an account that was never going to be repaid may involve carrying the account on the books longer than needed, at an additional cost of $10. Here, a cost function for the coincidence matrix could be:

$$\$190 \times (\text{closing good account}) + \$10 \times (\text{keeping bad account open})$$

(Note that we used our own dollar costs for purposes of demonstration, and these were not based on the real case.) This measure (like the correct classification rate) can be used to compare alternative models.

SPSS was used for discriminant analysis, including a stepwise forward selection procedure. The best model included 17 of the available 46 variables. Using equal misclassification costs yielded the coincidence matrix shown in Table 2.4.

Overall classification accuracy is obtained by dividing the correct number of classifications (50 + 578 = 628) by the total number of cases (718). Thus, the test data was correctly classified in 87.5. The cost function value here was:

$$\$190 \times 76 + \$10 \times 14 = \$14{,}580$$

The high proportion of actually solvent cases classified as insolvent was judged to be unacceptable, because it would chase away too many good customers. The experiment was reconducted using a-priori probabilities. This improved output significantly, as shown in the coincidence matrix in Table 2.5.

The test data was correctly classified in 93.0% of the cases. For the training data, this figure was 93.6%. Models usually fit training data a little better than test data, but that's because they were built on training data. Independent test data provides a much better test. The accuracy for insolvent customers, which is very important because it costs so much more, decreased from 78% in the training data to 56% in the test data. The cost function value here was the following:

$$\$190 \times 22 + \$10 \times 28 = \$4{,}460$$

Table 2.4. Coincidence matrix – equal misclassification costs

Telephone bill	Model insolvent	Model solvent	
Actual Insolvent	50	14	64
Actual Solvent	76	578	654
	126	593	718

Table 2.5. Coincidence matrix – unequal misclassification costs

Telephone bill	Model insolvent	Model solvent	
Actual Insolvent	36	28	64
Actual Solvent	22	632	654
	58	660	718

From a total cost perspective, the model utilizing unequal misclassification costs (using real costs) was considered more useful.

The 17 variables identified in the discriminant analysis were used for the other two models. The same training and test sets were employed. The training set was used to build a rule-based classifier model. The coincidence matrix for the test set is shown in Table 2.6.

Thus the test data was correctly classified in 95.26% of the cases. For the training data, this figure was 95.3%. The cost function value here was

$$\$190 \times 8 + \$10 \times 26 = \$1{,}780$$

This was an improvement over the discriminant analysis model.

A number of experiments were conducted with a neural network model using the same 17 variables and training set. The resulting coincidence matrix over the test data is shown in Table 2.7.

The test data was correctly classified in 92.9% of the cases. For the training data, this figure was 94.1%. The cost function value here was

$$\$190 \times 11 + \$10 \times 40 = \$2{,}490$$

However, these results were inferior to that of the decision tree model.

The first objective was to maximize accuracy of predicting insolvent customers. The decision tree classifier appeared to be best at doing that. The second objective was to minimize the error rate for solvent customers. The neural network model was close to the performance of the decision tree model. It was decided to use all three models on a case-by-case basis.

Table 2.6. Coincidence matrix – the rule-based model

Telephone bill	Model insolvent	Model solvent	
Actual Insolvent	38	26	64
Actual Solvent	8	646	654
	46	672	718

Table 2.7. Coincidence matrix – the neural network model

Telephone bill	Model insolvent	Model solvent	
Actual Insolvent	24	40	64
Actual Solvent	11	643	654
	35	683	718

Table 2.8. Coincidence matrix – combined models

Telephone bill	Model insolvent	Model solvent	Unclassified	Total
Actual Insolvent	19	17	28	64
Actual Solvent	1	626	27	654
	20	643	91	718

Stage 6. Deployment

Every customer was examined using all three algorithms. If all three agreed on classification, that result was adopted. If there was disagreement in the model results, the customer was categorized as unclassified. Using this scheme over the test set yielded the coincidence matrix shown in Table 2.8.

Thus, the test data was correctly classified in 89.8% of the cases. But only one actually solvent customer would have been disconnected without further analysis. The cost function value here was

$$\$190 \times 1 + \$10 \times 17 = \$360$$

The steps used in this application match the six stages we have presented. Data Selection relates to Learning the application domain and Creating a target dataset. Data Preprocessing involves Data Cleaning and preprocessing. Data Transformation involves Data Reduction and projection. Data Mining was expanded in the earlier application to include (1) choosing the function of data mining, (2) choosing the data mining algorithms, and (3) data mining. Data Interpretation involves the interpretation and use of discovered knowledge.

Summary

The industry-standard CRISP-DM data mining process has six stages: (1) Business Understanding, (2) Data Understanding, (3) Data Preparation, (4) Modeling, (5) Evaluation, and (6) Deployment. Data selection and understanding, preparation, and model interpretation require teamwork between data mining analysts and business analysts, while data transformation and data mining are conducted by data mining analysts alone. Each stage is a preparation for the next stage. In the remaining

chapters of this book, we'll discuss details regarding this process from a different perspective, such as data mining tools and applications. This will provide the reader with a better understanding as to why the correct process sometimes is even more important than the correct performance of the methodology.

Part II
DATA MINING METHODS AS TOOLS

3 Memory-Based Reasoning Methods

Memory-based reasoning systems are a type of model, supporting the modeling phase of the data mining process. Their unique feature is that they are relatively machine driven, involving automatic classification of cases. It is a highly useful technique that can be applied to text data as well as traditional numeric data domains.

Memory-based reasoning is an empirical classification method.[1] It operates by comparing new unclassified records with known examples and patterns. The case that most closely matches the new record is identified, using one of a number of different possible measures. This method can be applied to textual data as well as numerical data. That makes it very useful in comparing written documents. It has been used in a small number of cases, but has proven useful when conditions fit. Zurada and associates have been active in comparing memory-based reasoning against traditional data mining tools such as decision trees, neural networks, and regression for a variety of applications. They found memory-based reasoning to provide best overall classification when compared with the more traditional approaches in classifying jobs with respect to back disorders.[2] This group found that traditional models did better in an application to predict debt repayment in healthcare.[3] Mixed results were obtained when testing fuzzy logic and memory-based reasoning to evaluate residential property values in real estate.[4]

---

[1] C. Stanfill, D.L. Waltz (1986). Toward memory-based reasoning, *Communications of the ACM* 29:12, Dec, 1213–1223; D. Aha, D. Kibler, M. Albert (1991). Instance-based learning algorithms, *Machine Learning* 6, 37–66.

[2] J. Zurada, W. Karwowski, W. Marras (2004). Classification of jobs with risk of low back disorders by applying data mining techniques, *Occupational Ergonomics* 4:4, 291–305.

[3] J. Zurada, S. Lonial (2005). Comparison of the performance of several data mining methods for bad debt recovery in the healthcare industry, *Journal of Applied Business Research* 21:2, 37–54.

[4] J.M. Zurada, A.S. Levitan, J. Guan (2006). Non-conventional approaches to property value assessment, *Journal of Applied Business Research* 22:3, 1–14.

Matching

While matching algorithms are not normally found in standard data mining software, they are useful in many specific data mining applications. Fuzzy matching has been applied to discover patterns in the data relative to user expectations.[5] Java software has been used to completely automate document matching.[6] Matching can also be applied to pattern identification in geometric environments.[7]

There are a series of *measures* that have been applied to implement memory-based reasoning. The simplest technique assigns a new observation to the preclassified example most similar to it. The *Hamming distance* metric identifies the *nearest neighbor* as the example from the training database with the highest number of matching fields (or lowest number of non-matching fields). *Case-based reasoning* is a well-known expert system approach that assigns new cases to the past case that is closest in some sense. Thus case-based reasoning can be viewed as a special case of the nearest neighbor technique.

Job applicant data: We use a small (appended) database to represent the training database of past job applicants, with ratings of success with the hiring firm. Some of these variables are quantitative and others are nominal. State, degree, and major are nominal. There is no information content intended by state or major. CA, NV, and OR are not expected to have a specific order prior to analysis, nor is major. (The analysis may conclude that there is a relationship between state, major, and outcome, however.) Degree is ordinal, in that MS and MBA are higher degrees than BS. However, as with state and major, the analysis may find a reverse relationship with outcome. So in the context of this example, degree is nominal.

Age and experience are continuous variables. For matching, these variables can be grouped into ranges, such as age under 25, 25–30, and over 30; or 0 experience, 1–2 years experience, and over 2 years experience. More is not necessarily better than less, however. The analysis will reach its conclusion one way or the other.

---

[5] B. Liu, H.-Y. Lee (1999). Finding interesting patterns using user expectations, *IEEE Transactions on Knowledge & Data Engineering* 11:6, Nov/Dec, 817–832.

[6] S.M. Weiss, B.F. White, C.V. Apte, F.J. Damerau (2000). Lightweight document matching for help-desk applications, *IEEE Expert Intelligent Systems & their Applications* 15:2, Mar/Apr, 57–61.

[7] W. Rodriguez, M. Last, A. Kandel, H. Bunke (2001). Geometric approach to data mining, *International Journal of Image & Graphics* 2, Apr, 363–386.

Table 3.1. Past job applicants – Transformed data

Record	Age	State	Degree	Major	Experience	Outcome
1	25–30	CA	BS	Engineering	1–2 years	Excellent
2	>30	NV	Masters	Bus Admin	>2 years	Adequate
3	25–30	CA	Masters	Comp Sci	0	Adequate
4	<25	CA	BS	Info Sys	0	Unacceptable
5	25–30	CA	BS	Info Sys	1–2 years	Minimal
6	25–30	CA	Masters	Bus Admin	0	Excellent
7	25–30	CA	BS	Engineering	>2 years	Adequate
8	25–30	OR	Masters	Comp Sci	1–2 years	Adequate
9	25–30	CA	BS	Info Sys	1–2 years	Minimal
10	<25	CA	BS	Info Sys	1–2 years	Adequate

Transforming this data using these groupings (and grouping degree into categories of BS and Masters) yields the information in Table 3.1.

A simple application of Hamming distance is to categorize new applicants by the number of categories that match. Hamming distance is often measured as the number of negative cases (mismatches rather than matches). This has the feature that a smaller number indicates a closer relationship. Here we use the more direct measure of the number of matches. The results will be identical. For instance, if a 26 year old applicant from California with a BS in Information Systems and 4 years experience applies for a job, the Hamming distance to each of the items in the training set would be as shown in Table 3.2.

In this case, there are three records in the training set of ten with four matches for the new applicant. Record 5 had an outcome of Minimal, record 7 a record of Adequate, and record 9 an outcome of Minimal. The

Table 3.2. Matching past job applicant data – Case 1

Record	Age	State	Degree	Major	Experience	Outcome	Match
1	25–30	CA	BS	Eng	1–2 years	Excellent	3
2	>30	NV	Masters	Bus Ad	>2 years	Adequate	1
3	25–30	CA	Masters	Comp Sci	0	Adequate	2
4	<25	CA	BS	IS	0	Unacceptable	3
5	25–30	CA	BS	IS	1–2 years	Minimal	4
6	25–30	CA	Masters	Bus Ad	0	Excellent	2
7	25–30	CA	BS	Eng	>2 years	Adequate	4
8	25–30	OR	Masters	Comp Sci	1–2 years	Adequate	1
9	25–30	CA	BS	IS	1–2 years	Minimal	4
10	<25	CA	BS	IS	1–2 years	Adequate	3

Table 3.3. Matches of test observations with training observations

Record	1	2	3	4	5	6	7	8	9	10	Outcome
11	1	2	3	3	2	3	1	1	2	2	Inconclusive
12	2	1	2	1	2	1	3	3	2	1	Adequate
13	1	1	1	4	2	1	1	0	2	1	Unacceptable
14	4	1	1	2	3	1	3	1	3	3	Excellent
15	3	2	2	2	3	3	4	1	3	2	Adequate

new applicant could be assessed by this set of measures to be expected to be minimally acceptable as a prospect with a probability of two-thirds.

We can test this approach with the test set of data in the appendix. Each of these observations is tested against all ten of the training set observations, and the count of matches identified. For instance, record 11 has age >30, State CA, Masters degree in information systems, and 0 experience. This has 1 match with observation 1 (State), 2 with observation 2 (Age, Degree), 3 with observation 3 (State, Degree, Experience), and so forth. Table 3.3 shows the matches with training observations.

One of the problems with matching is that it has many inconclusive outcomes. For instance, record 12 had the most matches with two training observations (records 7 and 8). In this case, both outcomes were adequate, so the outcome is clear. But record 11 had the most matches with records 3, 4, and 6, the outcomes of which were adequate, unacceptable, and excellent. Not only are the outcomes inconclusive, they are radically different. This highlights another limitation of the method, in that all five measures are treated equally, even though State probably has nothing to do with outcome, while experience may be critical.

The coincidence matrix for this application (using the four conclusive predictions) is shown in Table 3.4.

This result was terrible. It did not correctly classify any cases. The actually excellent applicant was classified as unacceptable. The actually

Table 3.4. Coincidence matrix for basic matching

Actual	Unacceptable	Minimal	Adequate	Excellent	Total
Unacceptable	0	0	1	0	1
Minimal	0	0	1	0	1
Adequate	0	0	0	1	1
Excellent	1	0	0	0	1
Total	1	0	2	1	4

Table 3.5. Matching past job applicant data – New case

Record	Age	State	Degree	Major	Experience	Outcome	Match
1	25–30	CA	BS	Eng	1–2 years	Excellent	0
2	>30	NV	Masters	Bus Ad	>2 years	Adequate	3
3	25–30	CA	Masters	Comp Sci	0	Adequate	1
4	<25	CA	BS	IS	0	Unacceptable	0
5	25–30	CA	BS	IS	1–2 years	Minimal	0
6	25–30	CA	Masters	Bus Ad	0	Excellent	1
7	25–30	CA	BS	Eng	>2 years	Adequate	1
8	25–30	OR	Masters	Comp Sci	1–2 years	Adequate	1
9	25–30	CA	BS	IS	1–2 years	Minimal	0
10	<25	CA	BS	IS	1–2 years	Adequate	0

unacceptable candidate was classified as adequate. This is the reverse of what a model should do.

We can go ahead and demonstrate the application of the method to a new case. If a 35 year old applicant from Ohio with a Master's degree in Computer Science with 12 years of experience applied, the matches would be as shown in Table 3.5.

Here there clearly is a stronger match with record 2 than with any other record. The approach would therefore predict that this applicant would provide "adequate" performance for the company. However, there are some negative aspects demonstrated in this case. The applicant is quite a bit older than any case in the training set. The applicant also is from Ohio, not represented in the training set (although state probably doesn't have anything to do with job performance). Training sets should be as comprehensive as possible relative to the cases they are used to classify.

Weighted Matching

In this data set, age probably doesn't matter nearly as much as experience, or level of education, or major of study. State definitely wouldn't be expected to matter. Data mining can involve deletion of variables, but the usual attitude is to retain data because you don't know what it may provide. Weighting provides another means to emphasize certain variables over others. All that would change would be that the "Matches" measure could now represent a weighted score for selection of the best matching case. For instance, in this set of data, we could apply the weights shown in Table 3.6.

Table 3.6. Matching weights

Record	Age	State	Degree	Major	Experience	Outcome	Score
Weight	0.09	0.01	0.20	0.30	0.40		
1	25–30	CA	BS	Eng	1–2 years	Excellent	0
2	>30	NV	Masters	Bus Ad	>2 years	Adequate	0.69
3	25–30	CA	Masters	Comp Sci	0	Adequate	0.20
4	<25	CA	BS	IS	0	Unacceptable	0
5	25–30	CA	BS	IS	1–2 years	Minimal	0
6	25–30	CA	Masters	Bus Ad	0	Excellent	0.20
7	25–30	CA	BS	Eng	>2 years	Adequate	0.40
8	25–30	OR	Masters	Comp Sci	1–2 years	Adequate	0.20
9	25–30	CA	BS	IS	1–2 years	Minimal	0
10	<25	CA	BS	IS	1–2 years	Adequate	0

In this case, record 2 still shows up as the best match. But spurious matches, such as with age or state, will have much less influence on the outcome.

Distance Minimization

The next concept uses the distance measured from the observation to be classified to each of the observations in the known data set.

Job applicant data: In this case, the nominal and ordinal data needs to be converted to meaningful ratio data. Categorical data such as age group now must represent distance, where the distance from a given observation to each group indicates equal distance. For instance, the distance to age below 30 should be the same as the distance to the interval between 30 and 39. The best way to treat the age variable is to return to the original values of applicant age. If we attempt to measure the distance from the ten records in the memory database, we immediately encounter the problem of scale. Age is measured in terms ranging from 22 to 33 years, while state and degree are either 0 or 1, a range of 1. Experience ranges from 0 to 5 years. We might think that some of these variables are more important than others in predicting outcome, but initially we have no reason to expect any difference, and to arbitrarily overweight age because of the scale it is measured on is unattractive. Therefore, we want to normalize scales. One way to do that is to subtract the minimum from each observation and divide by the range. This, however, can result in complications if applicants are older than the maximum in the data set. (There are reasons to

want a training database including all possible values for each variable, but even if this is possible, future observations might spill over maxima or under minima). So we might normalize by assigning the expected minima and maxima for each variable. The maximum age considered is 50, and the minimum 20. Ages of 20 and below are assigned a value of 0, ages of 50 and above a value of 1.0, and all ages between a value of (Age–20)/30. For instance, the oldest applicant expected might be 50 years old, and the youngest 20. An applicant 27 years old would then be normalized to be:

$$(27–20)/30 = 0.233$$

Nominal data such as state can be used, but unless there is some way of measuring value on a continuous scale, it is best limited to binary data, where the applicant either is from a specific state or is not. For example, the database here is dominated by Californians. If being Californian were considered a positive social characteristic, one could assign the value CA a 1, and all others a 0.

Degree is ordinal. This variable would be problematic if there were more than two degrees. For instance, if non-degree applicants as well as Ph.Ds were included, the assignment of 0 for no degree, 1 for B.S., 2 for Masters, and 3 for Ph.D. might be the initial inclination of an academician. The first problem that arises is that this implied an equal distance between degrees. Possibly the distance between a Ph.D. and a Masters graduate is greater than that between a B.S. and a Masters. Furthermore, for this application there is no guarantee that the correct valuation might not be 0 for Ph.D., 1 for no degree, 5 for B.S., and 6 for Masters. In the specific case of this data set, there are fortunately only two entries. Therefore, we can assign 0 for Certification and B.S., and 1 for Masters or higher. This does not need to imply any order or value.

The same problem arises in assigning value to major. In this case, we assume that the ideal major would be information systems, although engineering, computer science, and science would also be useful backgrounds. For this specific job, business administration would be useful but not exactly what the job focused upon. The variable major is transformed assigning information systems a value of 1.0, engineering/computer science/science 0.9, business administration 0.7, and other degrees in this small set a value of 0.

The last input variable in the data set was years of experience. This was originally a continuous number, and we can return to that data form. After 5 years, all of the experience needed is considered to be obtained, so experience of 5 years or more is assigned a value of 1.0, and all other entries divided by 5. Coded values are given in Table 3.7.

Table 3.7. Past job applicants – Coded

Record	Age	State	Degree	Major	Experience	Outcome
1	0.233	1	0	0.9	0.4	Excellent
2	0.433	0	1	0.7	1.0	Adequate
3	0.333	1	1	0.9	0.0	Adequate
4	0.067	1	0	1.0	0.0	Unacceptable
5	0.267	1	0	1.0	0.4	Minimal
6	0.200	1	1	0.7	0.0	Excellent
7	0.167	1	0	0.9	0.6	Adequate
8	0.267	0	1	0.9	0.4	Adequate
9	0.167	1	0	1.0	0.4	Minimal
10	0.133	1	0	1.0	0.2	Adequate

The observations in the test data set can be applied by measuring the distance of each to each of the ten observations in the Training data set shown in Table 3.7. This distance can be measured a number of ways. The most commonly used are absolute value and squared value. Table 3.8 gives the coded values for the test set of data.

Table 3.9 shows the calculations for the first test case (record 11) against the first Training set observation (record 1) for the absolute value metric. This distance comes to 1.8.

Table 3.8. Coded test data for job applicants

Record	Age	State	Degree	Major	Experience	Outcome
11	0.533	1	1	1	0	Minimal
12	0.267	0	0	0.9	1	Unacceptable
13	0.133	0	0	1	0	Excellent
14	0.433	1	0	0.9	0.4	Adequate
15	0.200	1	0	0.7	0.6	Minimal

Table 3.9. Calculations for distance from test record 11 to record 1

Variable	Record 11	Record 11 coded	Record 1 coded	Absolute value of difference
Age	36	0.533	0.233	0.3
State	CA	1	1	0
Degree	MS	1	0	1
Major	Information S	1.0	0.9	0.1
Experience	0	0	0.4	0.4
			Sum	1.8

Table 3.10 shows the absolute value, sum of squared distance, and maximum distance measures for record 11 against each of the ten training cases.

In all three measures, the closest fit was with record 3, which had an Adequate outcome. Thus, test record 11 would be assigned an outcome of Adequate by all three measures. Results for this method for all five test cases are shown in Table 3.11.

The coincidence matrix for the absolute value distance metric is shown in Table 3.12.

Table 3.10. Distance measures from record 11 to training data

Training record	Absolute value	Squared distance	Maximum distance
1	1.800	1.260	1.000
2	2.400	2.100	1.000
3	0.300*	0.050*	0.200*
4	1.467	1.218	1.000
5	1.667	1.231	1.000
6	0.633	0.201	0.333
7	2.067	1.504	1.000
8	1.767	1.241	1.000
9	1.767	1.294	1.000
10	1.600	1.200	1.000

Table 3.11. Matches of test set

Record	Actual	Absolute Value	Squared	Maximum
11	Minimal	Adequate	Adequate	Adequate
12	Unacceptable	Adequate	Adequate	Inconclusive (ten ties)
13	Excellent	Unacceptable	Unacceptable	Inconclusive (ten ties)
14	Adequate	Excellent	Excellent	Excellent
15	Minimal	Adequate	Adequate	Inconclusive (two ties)

Table 3.12. Coincidence matrix for absolute value distance metric

Actual	Unacceptable	Minimal	Adequate	Excellent	Inconclusive	Total
Unacceptable	0	0	1	0	0	1
Minimal	0	0	2	0	0	2
Adequate	0	0	0	1	0	1
Excellent	1	0	0	0	0	1
Total	1	0	3	1	0	5

The results of this very small sample are very bad, in that not one correct classification of the test data was obtained. In fact, the one applicant that turned out to be excellent was rated as unacceptable. The squared distance metric produced the same five outcomes. The minimization of maximum distance (often called the Tchebycheff metric) gave the same outcomes as the other distance metrics except for the three cases where it was inconclusive. The Tchebycheff metric suffers from a tendency to have many ties in distance calculations over standardized data (because of the propensity for a distance of 1.0 to appear on some scale).

Keep in mind that this is a very small sample, and the method could improve a great deal with more samples. However, one major flaw in distance methods is that they are adversely affected by spurious variables, which actually have no relationship to the issue at hand. For instance, the applicant's state has no bearing on the actual outcome. Not being from California in this case could have a major impact, especially in a small sample as used to demonstrate the method here.

The first new applicant was a 26 year old Californian with a B.S. in information systems and 4 years of experience. Coded onto the same scales, this would yield values of:

New 1: Age 0.2 State 1 Degree 0 Major 1.0 Experience 0.8

The distance to each of the training records, measured in absolute values, are calculated in Table 3.13.

The applicant is closest to record 7, which had an adequate outcome. A 35 year old applicant from Ohio with a Masters degree in computer science and 12 years experience would be coded:

Table 3.13. Past job applicants – Absolute value distance calculations

Record	Age	State	Degree	Major	Experience	Abs. Val. Calc.	AV
1-Excellent	0.233	1	0	0.9	0.4	0.033+0+0+0.1+0.4	0.533
2-Adequate	0.433	0	1	0.7	1	0.233+1+1+0.3+0.2	2.733
3-Adequate	0.333	1	1	0.9	0	0.133+0+1+0.1+0.8	2.033
4-Unacceptable	0.067	1	0	1	0	0.133+0+0+0+0.8	0.933
5-Minimal	0.267	1	0	1	0.4	0.067+0+0+0+0.4	0.467
6-Excellent	0.200	1	1	0.7	0	0+0+1+0.3+0.8	2.100
7-Adequate	0.167	1	0	0.9	0.6	0.033+0+0+0.1+0.2	0.333*
8-Adequate	0.267	0	1	0.9	0.4	0.067+1+1+0.1+0.4	2.567
9-Minimal	0.167	1	0	1	0.4	0.033+0+0+0+0.4	0.433
10-Adequate	0.133	1	0	1	0.2	0.067+0+0+0+0.6	0.667

New 2: Age 0.5 State 0 Degree 1 Major 0.9 Experience 1.0

The absolute value distance metric for this applicant is closest to the record 2, which had an outcome of adequate.

Distance can be measured a lot of different ways. The most commonly used measure is squared distance, which gives greater emphasis to great distances than to small distances. The distance measure is simply the sum of squares of differences between the record value and the new value over all measures. Table 3.14 shows calculations for the training set of job applicants.

In this case the test data yielded the same result as the absolute value calculation. This often happens, but the squared distance is more affected by extreme differences. That could be a good thing if the measures were accurate, and if the variables included were important. However, in this case, being from a different state would not be expected to be important, but would affect the squared distance calculation more than the absolute value calculation. In this sense, the absolute value calculation is considered more robust. There is a third commonly used distance calculation, to minimize the maximum difference. This would be equivalent to taking the maximum absolute value difference among all of the variables in each group, and selecting that group with the minimum of these maximum absolute values. This distance metric is the one that is most affected by extreme values.

Table 3.14. Past job applicants – Squared distance calculation

Record	Age	State	Degree	Major	Exp.	Abs. Val. Calc.	Sq. D.
1-Excellent	0.233	1	0	0.9	0.4	$0.033^2+0+0+0.1^2+0.4^2$	0.171
2-Adequate	0.433	0	1	0.7	1	$0.233^2+1^2+1^2+0.3^2+0.2^2$	2.184
3-Adequate	0.333	1	1	0.9	0	$0.133^2+0+1^2+0.1^2+0.8^2$	1.668
4-Unacceptable	0.067	1	0	1	0	$0.133^2+0+0+0+0.8^2$	0.658
5-Minimal	0.267	1	0	1	0.4	$0.067^2+0+0+0+0.4^2$	0.164
6-Excellent	0.200	1	1	0.7	0	$0+0+1^2+0.3^2+0.8^2$	1.730
7-Adequate	0.167	1	0	0.9	0.6	$0.033^2+0+0+0.1^2+0.2^2$	0.051*
8-Adequate	0.267	0	1	0.9	0.4	$0.067^2+1^2+1^2+0.1^2+0.4^2$	2.174
9-Minimal	0.167	1	0	1	0.4	0.033+0+0+0+0.4	0.161
10-Adequate	0.133	1	0	1	0.2	0.067+0+0+0+0.6	0.364

Software

Memory-based reasoning is one of the data mining tools available on Darwin, recently acquired by Oracle.[8] The web site www.kdnuggets.com in May 2007 gives one product under the category of memory-based reasoning: PolyAnalyst (by Megaputer Intelligence), whose site describes memory based reasoning as finding the closest past analog of the present situation, and selecting the same solution that was considered right in those past situations. This is also referred to as the nearest-neighbor method. While such methods often do well in many problems, they have the deficiency of not having generalizable rules. Predictions require processing the entire body of available historic data.

Summary

Memory-based reasoning was proposed by Waltz and Kasif (1995) as an important framework for general intelligent applications, with the ability to include probabilistic aspects of problems.[9] Zurada and colleagues have been active in testing its use against conventional data mining techniques. Memory-based reasoning methods are appropriate in domains with a need for very fast response to rapidly changing environments.

The attractive feature of memory-based reasoning methods is their simplicity. This is counterbalanced by their computational and computer storage complexity. It is difficult to find too many software packages supporting this type of analysis. However, association rules, which can be identified through memory-based reasoning, are often the basis of text mining software.

There are some data features that were identified in our examples. It is doubtful that the data will be "well-behaved" in the sense that each useful output category will be equally represented (which would be the ideal statistical state). One way to adjust to this data reality is to assure that the training and test cases include all outcome categories. Further, interpretation of output should consider the proportion of outcomes.

If there are too many matches tied for best, it will help to transform the continuous variables into more groups (or divide categorical variables into more categories). This will provide additional discriminatory power. At the other extreme, of course, too many groups may lead to no matches. It will often be necessary to calibrate this feature.

---

[8] R. Whiting (1999). Oracle boosts data analysis, *Informationweek* 738, 14 June, 30.

[9] D. Waltz, S. Kasif (1995). On reasoning from data, *ACM Computing Surveys* 27:3, Sep, 356–359.

Appendix: Job Application Data Set

This dataset involves 500 past job applicants. Variables are

Age	Integer, 20–65	
State	State of origin	
Degree	Cert	Professional Certification
	UG	Undergraduate Degree
	MBA	Masters in Business Administration
	MS	Masters of Science
	PhD	Doctorate
Major	none	
	Engr	Engineering
	Sci	Science or Math
	Csci	Computer Science
	BusAd	Business Administration
	IS	Information Systems
Experience	Integer	Years of experience in this field
Outcome	ordinal	Unacceptable
		Minimal
		Adequate
		Excellent

Table 3A.1 gives the ten observations in the learning set.

Table 3A.1. Job applicant training data set

Record	Age	State	Degree	Major	Experience	Outcome
1	27	CA	BS	Engineering	2 years	Excellent
2	33	NV	MBA	Bus Admin	5 years	Adequate
3	30	CA	MS	Comp Sci	0	Adequate
4	22	CA	BS	Info Sys	0	Unacceptable
5	28	CA	BS	Info Sys	2 years	Minimal
6	26	CA	MS	Bus Admin	0	Excellent
7	25	CA	BS	Engineering	3 years	Adequate
8	28	OR	MS	Comp Sci	2 years	Adequate
9	25	CA	BS	Info Sys	2 years	Minimal
10	24	CA	BS	Info Sys	1 year	Adequate

Notice that some of these variables are quantitative and others are nominal. State, degree, and major are nominal. There is no information content intended by state or major. State is not expected to have a specific order prior to analysis, nor is major. (The analysis may conclude that there is a relationship between state, major, and outcome, however.) Degree is ordinal, in that MS and MBA are higher degrees than BS. However, as with state and major, the analysis may find a reverse relationship with outcome.

Table 3A.2 gives the test data set for this case.

Table 3A.3 provides a set of new job applicants to be classified by predicted job performance.

Table 3A.2. Job applicant test data set

Record	Age	State	Degree	Major	Experience	Outcome
11	36	CA	MS	Info Sys	0	Minimal
12	28	OR	BS	Comp Sci	5 years	Unacceptable
13	24	NV	BS	Info Sys	0	Excellent
14	33	CA	BS	Engineering	2 years	Adequate
15	26	CA	BS	Bus Admin	3 years	Minimal

Table 3A.3. New job applicant set

Age	State	Degree	Major	Experience
28	CA	MBA	Engineering	0
26	NM	UG	Science	3
33	TX	MS	Engineering	6
21	CA	Cert	None	0
26	OR	Cert	None	5
25	CA	UG	Bus Admin	0
32	AR	UG	Engineering	8
41	PA	MBA	Bus Admin	2
29	CA	UG	Science	6
28	WA	UG	Comp Sci	3

4 Association Rules in Knowledge Discovery

An association rule is an expression of $X \rightarrow Y$, where X is a set of items, and Y is a single item. Association rule methods are an initial data exploration approach that is often applied to extremely large data set. An example is grocery store market basket data.[1] Work on more efficient algorithms continues.[2] Association rules mining provides valuable information in assessing significant correlations. They have been applied to a variety of fields, to include medicine[3] and medical insurance fraud detection.[4] Business applications in addition to market basket analysis include warranty analysis.[5]

The most popular method is the a priori algorithm.[6] A number of methods have been developed to efficiently identify market basket relationships. Agrawal and Srikant presented the a priori algorithm that has been

---

[1] Y.-L. Chen, K. Tang, R.-J. Shen, Y.-H. Hu (2005). Market basket analysis in a multiple store environment, *Decision Support Systems* 40:2, 339–354.

[2] C. Lucchese, S. Orlando, R. Perego (2006). Fast and memory efficient mining of frequent closed itemsets, *IEEE Transactions on Knowledge & Data Engineering* 18:1, 21–36; D. Souliou, A. Pagourtzis, N. Drosinos, P. Tsanakas (2006). Computing frequent itemsets in parallel using partial support trees, *Journal of Systems & Software* 79:12, 1735–1743.

[3] T.P. Exarchos, C. Papaloukas, D.I. Fotiadis, L.K. Michalis (2006). An association rule mining-based methodology for automated detection of ischemic ECG beats, *IEEE Transactions on Biomedical Engineering* 53:8, 1531–1540.

[4] A.-C. Tsoi, S. Zhang, M. Hagenbuchner (2005). Pattern discovery on Australian medical claims data – A systematic approach, *IEEE Transactions on Knowledge & Data Engineering* 17:10, 1420–1435.

[5] J. Buddhakulsomsiri, Y. Siradeghyan, A. Zakarian, X. Li (2006). Association rule-generation algorithm for mining automotive warranty data, *International Journal of Production Research* 44:14, 2749–2770.

[6] R. Agrawal, T. Izmielinski, A. Swami (1993). Database mining: A performance perspective, *IEEE Transactions on Knowledge and Data Engineering* 5:6, 914–925; J. Han, J. Pei, Y. Yin (2000). Mining frequent patterns without candidate generation, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, Dallas, Texas, United States, 1–12.

improved several times.[7] Basic methods were demonstrated by Hipp et al.[8] Vaitchev et al. presented an approach to incremental frequent itemset generation seeking compact rule sets and low storage requirements.[9] Many algorithms have been proposed to find association rules mining in large databases. Most, such as the APriori algorithm identify correlations among transactions consisting of categorical attributes using binary values. Some data mining approaches involve weighted association rules for binary values,[10] or time intervals.[11] Mild and Reutterer[12] and Decker and Monien[13] offered additional computational approaches.

Data structure is an important issue due to the scale of data usually encountered.[14] Structured query language (SQL) has been a fundamental tool in manipulation of database content. Imielinksi and Mannila suggested a short-term research program focusing on efficient algorithms and more intelligent indexing techniques, as well as a long-term research program[15] to increase programmer productivity through more efficient tools to aid knowledge discovery.[16] Knowledge discovery involves ad hoc queries, needing efficient query compilation. Lopes et al. considered functional dependencies

---

[7] R. Agrawal, R. Srikant (1994). Fast algorithms for mining association rules, *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, Santiago, Chile, 487–499.

[8] J. Hipp, U. Güntzer, G. Nakhaeizadeh (2000). Algorithms for association rule mining – A general survey and comparison, *SIGKDD Explorations* 2:1, 58–64.

[9] P. Vaitchev, R. Missaoui, R. Godin, M. Meridji (2002). Generating frequent itemsets incrementally: Two novel approaches based on Galois lattice theory, *Journal of Experimental & Theoretical Artificial Intelligence* 14:2/3, 115–142.

[10] C.H. Cai, W.-C. Fu, C.H. Cheng, W.W. Kwong (1998). Mining association rules with weighted items, *Proceedings of 1998 International Database Engineering and Applications Symposium*, Cardiff, Wales, 68–77.

[11] S.-F. Lu, H. Hu, F. Li (2001). Mining weighted association rules, *Intelligent Data Analysis*, 5, 211–225.

[12] A. Mild, T. Reutterer (2003). An improved collaborative filtering approach for predicting cross-category purchases based on binary market basket data, *Journal of Retailing & Consumer Services* 10:3, 123–133.

[13] R. Decker, K. Monien (2003). Market basket analysis with neural gas networks and self-organising maps, *Journal of Targeting, Measurement & Analysis for Marketing* 11:4, 373–386.

[14] X. Yan, S. Zhang, C. Zhangt (2007). On data structures for association rule discovery, *Applied Artificial Intelligence* 21:2, 57–79.

[15] R.S.T. Lee, J.N.K. Liu (2004). IJADE Web-Miner: An intelligent agent framework for internet shopping, *IEEE Transactions on Knowledge & Data Engineering* 16:4, 461–473.

[16] T. Imielinski, H. Mannila (1996). A database perspective on knowledge discovery, *Communications of the ACM* 39:11, 58–64.

in inference problems.[17] SQL was used by those researchers to generate sets of attributes that were useful in identifying item clusters. Lee and Liu suggested a way to utilize agent technology to expedite Web mining that utilized SQL. Grossman et al. presented a predictive model markup language to aid data mining.[18]

Key measures in association rule mining include support and confidence. Support refers to the degree to which a relationship appears in the data. Confidence relates to the probability that if a precedent occurs, a consequence will occur. The rule $X \rightarrow Y$ has minimum support value *minsup* if *minsup* percent of transactions support $X \cup Y$, the rule $X \rightarrow Y$ holds with minimum confidence value *minconf* if *minconf* percent of transactions that support *X* also support *Y*. For example, from the transactions kept in supermarkets, an association rule such as "Bread and Butter $\rightarrow$ Milk" could be identified through association mining.

Market-Basket Analysis

Market-basket analysis refers to methodologies studying the composition of a shopping basket of products purchased during a single shopping event. This technique has been widely applied to grocery store operations (as well as other retailing operations, to include restaurants). Market basket data in its rawest form would be the transactional list of purchases by customer, indicating only the items purchased together (with their prices). This data is challenging because of a number of characteristics:[19]

- A very large number of records (often millions of transactions per day)
- Sparseness (each market basket contains only a small portion of items carried)
- Heterogeneity (those with different tastes tend to purchase a specific subset of items).

The aim of market-basket analysis is to identify what products tend to be purchased together. Analyzing transaction-level data can identify

[17] S. Lopes, J.-M. Petit, L. Lakhal (2002). Functional and approximate dependency mining: Database and FCA points of view, *Journal of Experimental Theoretical Artificial Intelligence* 14, 93–114.

[18] R.L. Grossman, M.F. Hornick, G. Meyer (2002). Data mining standards initiatives, *Communications of the ACM* 45:8, 59–61.

[19] C. Apte, B. Liu, E.P.D. Pednault, P. Smyth (2002). Business applications of data mining, *Communications of the ACM* 45:8 49–53.

purchase patterns, such as which frozen vegetables and side dishes are purchased with steak during barbecue season. This information can be used in determining where to place products in the store, as well as aid inventory management. Product presentations and staffing can be more intelligently planned for specific times of day, days of the week, or holidays. Another commercial application is electronic couponing, tailoring coupon face value and distribution timing using information obtained from market-baskets.[20] Data mining of market basket data has been demonstrated for a long time,[21] and has been applied in a variety of applications.[22]

Due to increasing size of data sets in databases and data warehouses, researchers have tried different architectural alternatives for efficient coupling of mining with database systems. Experiments with real life datasets on some of the methods such as loose-coupling through a SQL cursor interface, encapsulation of a mining algorithm in a stored procedure, caching the data to a file system on-the-fly and mining, tight-coupling using primarily user-defined functions, and SQL implementations for processing in the DBMS have been conducted and compared.[23] The focal point of research in data mining is increasing efficiency, but simplicity and portability have also been considered.[24]

Market Basket Analysis Benefits

The ultimate goal of market basket analysis is finding the products that customers frequently purchase together. The stores can use this information by putting these products in close proximity of each other and making them more visible and accessible for customers at the time of shopping. These assortments can affect customer behavior and promote the sales for complement items. The other use of this information is to decide about the layout of catalogs and put the items with strong association together in sales catalogs. The advantage of using sales data for promotions and

[20] G.J. Russell, A. Petersen (2000). Analysis of cross category dependence in market basket selection, *Journal of Retailing* 78:3, 367–392.

[21] E. Simoudis (1996). Reality check for data mining, *IEEE Expert Intelligent Systems & Their Applications* 11:5, 26–33.

[22] R. Reed (1995). Household ethnicity, household consumption: Commodities and the Guarani, *Economic Development & Cultural Change* 44:1, 129–145.

[23] S. Sarawagi, S. Thomas, R. Agrawal (1998). Integrating association rule mining with relational database systems: Alternatives and implications, *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, WA, United States, 343–354.

[24] Ibid.

store layout is that the consumer behavior information determines the items with associations. This information may vary based on the area and the assortments of available items in stores and the point of sale data reflects the real behavior of the group of customers that frequently shop at the same store. Catalogs that are designed based on the market basket analysis are expected to be more effective on consumer behavior and sales promotion.

Market basket analysis is an undirected method. It doesn't involve choosing specific products and finding their associations with other items. This method can reveal the associations that may be unknown to the store management. The products that are most important are revealed through this analysis. The current study provides the method of querying for specific products as well. Market basket analysis can be used to identify the items frequently sold to new customers and profiling the customer baskets during a period of time by identifying customers through membership shopping cards.

Demonstration on Small Set of Data

Market basket analysis involves large scale datasets. To demonstrate methods, we will generate a prototypical dataset consisting of 10 grocery items, with 25 market baskets. Table 4.1 shows the raw data.

Table 4.1. Prototypical data

Basket	Milk	Eggs	Bread	Beer	Water	Cola	Apples	Beans	Peas	Diapers
1				X						X
2	X		X			X	X		X	
3	X	X	X	X			X	X		X
4		X			X		X	X	X	
5	X		X	X		X	X			
6			X			X		X		
7						X	X			
8	X	X	X							X
9	X						X		X	X
10	X		X					X		
11		X		X						
12					X		X		X	
13	X		X					X		
14	X		X				X			
15				X				X		
16					X		X			X
17			X	X						
18	X	X	X	X	X	X	X	X	X	X

(Continued)

Table 4.1. (Continued)

Basket	Milk	Eggs	Bread	Beer	Water	Cola	Apples	Beans	Peas	Diapers
19	X		X			X	X	X	X	
20			X			X		X		
21		X		X						
22	X	X	X			X	X	X		X
23	X						X			
24						X	X			
25	X	X	X			X	X			

By sorting on the item with the most volume, counts for cross-sales can be generated as in Table 4.2.

The diagonal contains the total number of market baskets containing each item. It can be seen that most who purchased apples also purchased milk, bread, and cola. Of those that purchased beer, few purchased water. One customer purchased everything, so there are no zeros in this matrix. Realize that this set of data appears to have inflated sales, but it is for purposes of demonstration. Real groceries would have many more items, with many more zeros. But our data can be used to demonstrate key measures in association rules. *Support* is the number of cases for a given pair. The support for apples and milk is 10. For beer and water it is one. Minimum support can be used to control the number of association rules obtained, with higher support requirements yielding fewer pairs. A minimum support of 1 would have 45 pairs, the maximum for 10 items. Minimum support of 15 would have no rules. Minimum support of 10 would have 2 {Apples-Milk 10 cases; Bread-Milk 11 cases}. Note that association rules can go beyond pairs to triplets, sets of four, and so on. But usually pairs are of interest (and much easier to identify).

Table 4.2. Cross-Sales

	Apples	Bread	Milk	Cola	Beans	Eggs	Beer	Diapers	Peas	Water
Apples	15	8	10	8	5	5	3	5	6	4
Bread	8	14	11	8	8	5	4	4	3	1
Milk	10	11	13	6	6	5	3	5	4	1
Cola	8	8	6	10	5	3	2	2	3	1
Beans	5	8	6	5	10	4	3	3	3	2
Eggs	5	5	5	3	4	8	4	4	2	2
Beer	3	4	3	2	3	4	8	3	1	1
Diapers	5	4	5	2	3	4	3	7	2	2
Peas	6	3	4	3	3	2	1	2	6	3
Water	4	1	1	1	2	2	1	2	3	4
TOTAL	15	14	13	10	10	8	8	7	6	4

Another key measure is *confidence*. For a given pair, confidence is the proportion of true conditions to total possible. Bread and milk had the highest support of 11 cases. If bread was in a market basket, the confidence that milk would also be in the basket is 11/14, or 0.786. Confidence is relative to the base case. If milk were in the market basket, the confidence that bread would appear would be 11/13, or 0.846. Minimum confidence levels can be set in most data mining software, again with a higher level yielding fewer pairs.

Improvement is a measure of how much stronger (or weaker) a given pair occurrence is relative to random. For instance, there were 15 cases of apples occurring. There were a total of 25 market baskets, with 95 total items sold. Random would therefore have any particular market basket containing 3.8 items out of 10. Of the 15 market baskets containing any given item, there are on average 2.8 other items, so each item would have an average probability of 2.8/9 or 0.311. For 15 items then, random expected number of complementary items would be 0.311 times 15 or 4.67. Only beer and water fell below this expectation. A minimum improvement parameter of 2 would require 9.33 items, and only milk would pass this limit for apples.

Real Market Basket Data

A set of real grocery store data was obtained that we can use to demonstrate the scale of market-basket analysis.[25] It was part of a sales database of a grocery store chain. Point of sale data from 18 stores in the area was collected in a central system, with each transaction including the store code. The point of sale data was collected in a flat file and each record included the item identifier (UPC) code, the store number, date of sale, basket identifier, and other information.

The first step was data cleaning because some sales records lacked the basket identifier or the UPC code. In this step about 4.8% of the total records were eliminated to create a reliable dataset for analysis. The eliminated data was for discount coupons, not considered a product.

In the second step the text formatted data of transaction records was loaded into a relational database for querying. Customer basket analysis involves finding the number of occurrences of every item to all other items to find possible relationships between them. Two problems in a real world dataset arise. First, there are usually tens of thousands of items in grocery

[25] Data obtained from an anonymous store by M. Nabavi.

stores, and counting all of these items occurrences against each other involves hundreds of millions of calls, each involved in executing calculation methods and database connection operations. Even with today's fast and powerful computers this operation takes a long time and is not practical. Finding efficient algorithms for candidate item generation is a significant part of data mining research. The second problem is that recording the results in commercial database management systems is not possible due to the limits in the number of columns, even though we can save the results in text format. To explain this limitation, we have to look at Fig. 4.1 that represents the data table structure in our study, and this is a generic structure for transaction recording in sales databases. Every tuple includes the UPC code of the item and a transaction identifier. One solution for sorting the items that repeat to be in the same transaction together is extracting them as data elements of a tuple, but this tuple may include a large number of elements, with a limitation on the number of columns in most of commercial database systems at 255, this option is not practical. For example, in case of UPC "0000000006124", this count was 1162 and for UPC "0007066201003" this count was 1101 which means we need a matrix with at least 163 columns.

A two step approach to screen data records in first step was used to identify the potential important items and made the analysis practical. The structure of sales data table items that were important to this analysis is illustrated in Fig. 4.1.

The "UPC code" is unique identifier of sales item, "Store" is the store identifier, "Date" identifies the date of transaction, and "Basket identifier" identifies the sales transaction of a specific basket in a specific date.

Candidate generation through a simple SQL query was used to scan every item in the dataset and count the total number of other items that share the same transaction code with that item. One criterion that is important at this step is the minimum support for groups of repetitions. Minimum support could be implemented in two ways; counting the percentage of transactions, or counting the occurrence of items and then filtering out the counts that do not meet the minimum count. The second way was implemented in this study, setting a minimum number of counts as the minimum support required for candidate items. The results were saved in the

UPC	Store	Date	Transaction identifier	……
…	…	…	…	…

Fig. 4.1. The point of sale data items considered in analysis

```
INSERT INTO Tc
SELECT DISTINCT a1.upc_code AS UPC_1, count(a2.upc_code) AS CO_Count
FROM I AS a1, I AS a2
WHERE a1.Transaction_identifier=a2. Transaction_identifier and a2.upc_code NOT
   LIKE a1.upc_code
GROUP BY a1.upc_code, null;
   HAVING count(a2.upc_code)> minsup
```

Fig. 4.2. Query 1

candidate items table T_c. The SQL query that extracts this set is given in Fig. 4.2.

Results were grouped by UPC codes of items. Sorting the results based on the count of other items that go with each particular item provided a list of items that identify the potential important items and substantially narrowed down the number of candidates for final step of analysis. A test run of all items for counting the number of all items together showed no difference in results. This step narrowed down a list of about 212,000 items to 2,600 top items that can potentially be important for our study. Another query that counted the number of individual items that had occurrences with each item that was not between these top 2,600 items showed no significant results. The advantage of this method for candidate generation is simplicity of the method, standard SQL support in all database systems. As later confirmed in comparison with commercial data mining system results later in this study, this method returns a more complete set of candidates while has the flexibility of adjusting the minimum support indicator *minsup*

The second step after candidate generation was checking each item in the candidate list against each other. There are different alternatives for this step, including stored procedures in database system, and standard application interface to the database system. Since the purpose of this study is providing a portable solution accessible by all platforms, the standard application interface was the proper choice. Developing this simple interface is easy and practical, following the algorithm in this section. This application provided the output as matrix of frequencies in text format. The output data was loaded into commercial data mining system PolyAnalyst to find the associations of items together and later comparing them to the final results of our analysis without using the commercial data mining software. PolyAnalyst supports traditional market basket analysis, which treats data as binary in the sense that products are either purchased together (yes) or not (no). It also can utilize additional data, such as product volume or dollar value. In our case, we applied traditional market basket analysis. The software required three input parameters:

1. Minimum support sets the minimum proportion of transactions that should contain a basket of products in order to be considered. If this number is set high, only those products that appear very often will be considered, which will lead to a small number of product clusters with few products. This would be useful if one-to-one rules of important products were desired. Setting a low minimum support level will generate many rules including many products. We used the default of 10%.
2. Minimum confidence sets the probability that if one item is purchased, another item will be. The higher this value is set, the more discriminating the resulting association rule. However, if minimum confidence is set too high, there is a higher likelihood of no rules being generated. We used the default setting of 65%.
3. Minimum improvement sets how much better the association rule generated is at predicting the presence of a product in a market basket than would be obtained at random. Again, higher settings would result in fewer association rules being generated, while lower settings would yield more rules. We used the default setting of 2.

PolyAnalyst yielded 201 groups of associated products. These groups contained from 2 to 78 products. While this output can be extremely useful to retail organizations, it requires investment in data mining software and more importantly prioritizing item sets from a large number of sets in output. The same information could be obtained through relational database software.

The Counting Method Without Software

This method consists of two steps. The first step identifies the items that are sold with larger number of other items. The purpose of this step is narrowing the list of items down for efficiency of calculations and time of computation. Running “query1” provides a list of items and number of the other items sold with them in same customer baskets. Having the results of “query1” in hand, we have the option of selecting any number of items for the second step, which identifies the specific items that are members of same frequent set. In the current study we selected top 2443 items, which had minimum support in entire data set. The criterion at this step was having 50 and more combined times of sales with other items in transactions. We call this the top items list.

In the second step a SQL query counted the number of occurrences of shopping every item in the top items list and identified the UPC code for

```
SELECT DISTINCT a2.UPC_Code, Count(a2.UPC_Code) AS CO_COUNT
FROM I AS a1, I AS a2
WHERE (((a1. Transaction_identifier)=a2. Transaction_identifier) and
   ((a2.UPC_Code) NOT LIKE a1.UPC_Code) And ((a1.UPC_Code)='[SPECIFIC
   UPC_CODE]'))
GROUP BY a2.UPC_CODE, NULL
```

Fig. 4.3. SQL query

each item along with the number of times these two items are sold together. This query was passed through a loop in the program that sends the query to the database and in every pass an array including all UPC codes in the top items list substitutes the UPC codes for the "*SPECIFIC UPC_CODE*" in the query. The SQL query (Fig. 4.3) has the following structure:

The program algorithm that passes the query to database and returns the results is simple and compares all the items in our top list to find the repetitions of selling items together. The algorithm is illustrated in Fig. 4.4:

The output was a database table with three columns. In every record, an item in the sales list is in the first column, one frequent pair item is in the second field, and the number of times these two items are sold together is in the third field, and this record is taken for every item. The advantage of this method is that by simply sorting the result based on the number of repetitions we can get the most frequent items that are sold together. In case the store managers want to get the results for any specific item, this table can be easily grouped by UPC code in the first field and sorted for number of repetitions.

We compared database queries with a commercial data mining software, PolyAnalyst. The dataset analyzed with PolyAnalyst software produced 78 groups of products, identified by their UPCs. Since the query results from database analysis included the complete set of items and their matching products along with their counts, a proper way of investigating the report

```
LOOP: For each item_i
        IF Transaction includes item_i
        THEN Select transaction_identifier
        LOOP For transaction_identifier
                Select all items as A:{item_1 , item_2 ,....., item_k }
                For each item_n of set A count the number of times for pair {item_i, item_n}set
                in transactions
                Record result in matrix
                Go to the next item
        END LOOP
        Go to the next item
END LOOP
```

Fig. 4.4. Program algorithm

```
SELECT *
FROM Count_Pairs (The result table for matching pairs)
WHERE UPC_1 like 'UPC_CODE' or UPC_2 like 'UPC_CODE'
ORDER BY Count DESC;
```

Fig. 4.5. Query for UPC codes

result was to run queries to find out about the accuracy of the output report. The query that checks the results has the syntax given in Fig. 4.5.

The following set of results from one of the output groups is an example of the groups of UPC codes matched by PolyAnalyst. The UPC codes for one of the output groups, which had a relatively high volume (132) for a low number of products (4), are given below.

04158100134
00000006154
04158100131
04158100129

Table 4.3 is the top part of the query for UPC Code 00000006154. In this table, product item identified by UPC_1 and ID = 1 has been in the same basket with product identified by UPC_2 and ID = 1 for 63 times, but this relationship has not been identified by the commercial data mining software. The next match with 52 counts has been included in the results, which is the UPC_1 and UPC_2 in ID = 2 row.

Table 4.3. Query results for matching items with UPC code "00000006154"

ID	UPC_1	UPC_2	Count
1	00000006154	00000006500	63
2	00000006154	04158100134	52
3	00000006154	04158100129	43
4	00000006154	04158100131	41
5	00000006154	00000006014	38
6	00000006154	07740012043	28
7	00000006154	00000006016	24
8	00000006154	00000006371	21
9	00000006154	04158100130	20
10	00000006154	00000006114	20
11	00000006154	00000006069	19
12	00000006154	70601011297	18
13	00000006154	07870019590	17
14	00000006154	07003833255	16
15	00000006154	07740012013	14
16	00000006154	07740012023	14

Table 4.3. (Continued)

ID	UPC_1	UPC_2	Count
17	00000006154	00000006360	14
18	00000006154	00000006595	14
19	00000006154	00000006098	12
20	00000006154	00000006407	12
21	00000006154	04119691138	12
22	00000006154	07003859126	12
23	00000006154	07003859128	12
24	00000006154	07066201003	12
25	00000006154	04119691077	12
26	00000006154	00000007257	11
27	00000006154	00000006199	11

As an additional test for result accuracy we ran the query with UPC code “04158100134” and as shown in Table 4.4, it returned the item with UPC code in the UPC_2 column. This relationship has been identified by the output report of PolyAnalyst.

Repeating the query with UPC code “04158100131” returns the results shown in Table 4.5. The first row has been identified by the software but not the second row with 18 occurrences, while the software included associations with fewer occurrences in most of the output groups.

Finally, the UPC code “04158100129” in query returns values shown in Table 4.6 with 43 count of associating “04158100129” with item “00000006154” as the most frequent occurrence of selling in the same basket. This relationship is identified by the software.

Table 4.4. Query results for matching items with UPC code “04158100134”

ID	UPC_1	UPC_2	Count
1	04158100134	00000006154	52

Table 4.5. Query results for matching items with UPC code “041588100131”

ID	UPC_1	UPC_2	Count
1	04158100131	00000006154	41
2	04158100131	00000006500	18

Table 4.6. Query results for matching items with UPC code “04158100129”

ID	UPC_1	UPC_2	Count
1	04158100129	00000006154	43
2	04158100129	00000006500	27
3	04158100129	07740012043	12

Database querying for items in this group shows a relatively good identification of items in same customer baskets by the software, with the exception of the most frequent actual count of items in the group; selling “00000006500” with “00000006154” for 63 times. In this case, the relationship has been better identified through database querying.

In the PolyAnalyst report output included two products with UPC code “01780041992” and “00000006402”. The query results using “01780041992” as identifier UPC returns results shown in Table 4.7, which indicates that two other products, not included in the report, have more instances of selling with this item than the item with UPC “00000006402”.

The querying process was repeated for a group including 22 products (Group 1). Some of these items were tested for efficiency of the output report by comparing the set to query results from the database. The UPC codes for Group 1 are given in Table 4.8.

Table 4.7. Query results for matching items with UPC code “01780041992”

ID	UPC_1	UPC_2	Count
608554	01780041992	00000006016	32
608725	01780041992	07066201002	20
608578	01780041992	00000006402	14
608555	01780041992	00000006066	12

Table 4.8. UPC codes for Group 1

07127923200	01980003674	03800004530	07127928100	00000006405
00000007115	04400000659	07800008216	03600029169	00000004635
07003836368	07003858114	00000009641	04174358208	
01600068890	03700036052	05040040369	03450063211	
03651853120	07064003525	01141160152	00000000448	

Table 4.9. Query results for matching items with UPC code “07127923200”

ID	UPC_1	UPC_2	Count
1	07127923200	00000007115	68
2	07127923200	00000007590	38
3	07127923200	00000007436	27
4	07127923200	00000007257	27

The first UPC code tested in query was “07127923200” that returned the results in Table 4.9.

These values show that item “00000007115” with 68 counts was the most common match, while next most common matching item was “00000007590” with 38 counts. On the other hand, some items will have lower matches. Item “07003836368” was tested for all other items in the group, having maximum 3 counts with item “03600029169” and less than that with other items in the group. Results of querying database for this item are shown in Table 4.10. Each row of this table is an output of a separate query on the database.

Table 4.10. Testing item “07003836368” with all other items in the group

ID	UPC_1	UPC_2	Count
1	07003836368	00000007115	1
2	07003836368	04174358208	1
3	07003836368	04400000659	1
4	07003836368	01980003674	2
5	07003836368	00000007115	1
6	07003836368	04174358208	1
7	07003836368	04400000659	1
8	07003836368	01980003674	2
9	07003836368	03651853120	1
10	07003836368	01600068890	2
11	07003836368	07127923200	1
12	07003836368	07003858114	1
13	07003836368	07064003525	2
14	07003836368	03800004530	1
15	07003836368	07800008216	2
16	07003836368	00000009641	2
17	07003836368	05040040369	1
18	07003836368	01140060152	1
19	07003836368	07127928100	1
20	07003836368	03600029169	3

The outcome shows that by simply relying on group outputs from data mining software we may miss important relationships, while the large number of output groups adds to the confusion of user to identify important relationships.

Conclusions

Association rules are basic data mining tools for initial data exploration usually applied to large data sets, seeking to identify the most common groups of items occurring together. In this sense it is much like cluster analysis, but cluster analysis involves different algorithms that can be more computationally intensive and impractical in large datasets. The most common application of association rule mining is market basket analysis, but it has been applied to a growing number of other applications as indicated in our short review.

Market basket analysis can be very useful to small businesses. However, specialty data mining software capable of supporting market basket analysis is expensive, and requires specialists who understand that software. While PolyAnalyst was easy to use, it also involves additional investment, and the output is “black box” in the sense that it is difficult to see why particular products were grouped together.

Generating market basket groups through SQL queries provides a means to obtain association rule data without the need for additional software. The benefit of this method is not just saving the required investment for software acquisition and its training costs, but in the flexibility and accuracy of the results. The associations discovered between items through the SQL queries are more accurate, discovering the relationships that were not identified by the commercial software. This method provides users with a set of associations that are readily available for grouping based on any desired item. In this method the relationships are known and controlled by users, and the number of sale transactions for paired items is available to the user, which is an advantage comparing to the commercial software output that does not describe the association rules.

5 Fuzzy Sets in Data Mining

Real-world application is full of vagueness and uncertainty. Several theories on managing uncertainty and imprecision have been advanced, to include fuzzy set theory,[1] probability theory,[2] rough set theory[3] and set pair theory.[4] Fuzzy set theory is used more than the others because of its simplicity and similarity to human reasoning. Fuzzy modeling provides a very useful tool to deal with human vagueness in describing scales of value. The advantages of the fuzzy approach in data mining is that it serves as an "… interface between a numerical scale and a symbolic scale which is usually composed of linguistic terms."[5] We will discuss some use of fuzzy sets in data mining, demonstrating decision tree and association rule use.

Fuzzy association rules described in linguistic terms help users better understand the decisions they face.[6] Fuzzy set theory is being used more and more frequently in intelligent systems. A fuzzy set A in universe U is defined as $A = \{(x, \mu_A(x)) \mid x \in U, \mu_A(x) \in [0,1]\}$, where $\mu_A(x)$ is a membership function indicating the degree of membership of x to A. The greater the value of $\mu_A(x)$, the more x belongs to A. Fuzzy sets can also be thought of as an extension of the traditional crisp sets and categorical/ordinal scales,

---

[1] L.A. Zadeh (1965). Fuzzy sets, *Information and Control* 8, 338–356.

[2] J. Pearl (1988). Probabilistic reasoning in intelligent systems, *Networks of Plausible Inference* San Mateo, CA: Morgan Kaufmann.

[3] Z. Pawlak (1982). Rough set, *International Journal of Computer and Information Sciences* 11, 341–356.

[4] K.-G. Zhao (1989). Set pair analysis – a new concept and new systematic approach. *Proceedings of National System Theory and Regional Analysis Conference*, Baotou (In Chinese); K.-G. Zhao (2000). *Set Pair Analysis and Its Preliminary Application*, Zhejiang Science and Technology Press (in Chinese).

[5] D. Dubois, E. Hullermeier, H. Prade (2006). A systematic approach to the assessment of fuzzy association rules, *Data Mining and Knowledge Discovery*, July, 13:2, 1–26.

[6] W.-J. Lee, S.-J. Lee (2004). Discovery of fuzzy temporal association rules, *IEEE Transactions on Systems, Man, and Cybernetics – Part B* 34:6, 2330–2342; W. Shitong, K.F.L. Chung, S. Hongbin (2005). Fuzzy taxonomy, quantitative database and mining generalized association rules, *Intelligent Data Analysis* 9, 207–217.

in which each element is either in the set or not in the set (a membership function of either 1 or 0).

Fuzzy set theory in its many manifestations (interval-valued fuzzy sets, vague sets, grey-related analysis, rough set theory, etc.) is highly appropriate for dealing with the masses of data available. We will review some of the general developments of fuzzy sets in data mining, with the intent of seeing some of the applications in which they have played a role in advancing the use of data mining in many fields. We will then review the use of fuzzy sets in two data mining software products, and demonstrate the use of data mining in an ordinal classification task. The results will be analyzed through comparison with the ordinal classification model. Possible adjustments of the model to take into account fuzzy thresholds in ordinal scales will be discussed.

Classification tasks in business applications may be viewed as tasks with classes reflecting the levels of the same property. Evaluating creditworthiness of clients is rather often measured on an ordinal level as, e.g., {excellent}, {good}, {acceptable}, or {poor}.[7] Applicants for a job are divided into accepted and rejected, but sometimes there may be also a pool of applicants left for further analysis as they may be accepted in some circumstances. Different cars may be divided into groups {very good}, {good}, {acceptable}, {unacceptable}. This type of tasks is called "ordinal classification."[8] The peculiarity of ordinal classification is that data items with {better} qualities (characteristics) logically are to be presented in {better} classes: the better the article measures on its characteristics the closer it is to the class {accepted}. Taking into account possible ordinal dependence between attribute values and final classes may lead to a smaller number of rules with the same accuracy and enable the system to extend obtained rules to instances not presented in the training data set.[9]

There are many data mining tools available, to cluster data, to help analysts find patterns, to find association rules. The majority of data mining approaches in classification work with numerical and categorical information. Most data mining software tools offer some form of fuzzy analysis. Modifying continuous data is expected to degrade model accuracy, but

---

[7] A. Ben David (1992). Automated generation of symbolic multiattribute ordinal knowledge-based DSSs: Methodology and applications. *Decision Sciences* 23:6, 157–1372; R. Slowinski (1995). Rough set approach to decision analysis, *AI Expert* 10:3, 19–25.

[8] O.I. Larichev and, H.M. Moshkovich (1994). An approach to ordinal classification problems *International Transactions on Operations Research* 82, 503–521.

[9] H.M. Moshkovich, A.I. Mechitov, D.L. Olson (2002), Rule induction in data mining: Effect of ordinal scales, *Expert Systems with Applications* 22, 303–311.

might be more robust with respect to human understanding. (Fuzzy representations might lose accuracy with respect to numbers that don't really reflect accuracy of human understanding, but may better represent the reality humans are trying to express.) Another approach to fuzzify data is to make it categorical. Categorization of data is expected to yield greater inaccuracy on test data. However, both treatments are still useful if they better reflect human understanding, and might even be more accurate on future implementations. The categorical limits selected are key to accurate model development.[10] Not many data mining techniques take into account ordinal data features.

Fuzzy Sets and Decision Trees

See5, a decision tree software, allows users to select options to soften thresholds through selecting a fuzzy option.[11] This option would insert a buffer at boundaries (which is how PolyAnalyst works as well). The buffer is determined by the software based on analysis of sensitivity of classification to small changes in the threshold. The treatment from there on is crisp, as opposed to fuzzy. Thus, in decision trees, fuzzy implementations seem to be crisp models with adjusted set boundaries.

See5 software was used in an experiment on a real set of credit card data.[12] This dataset had 6,000 observations over 64 variables plus an outcome variable indicating bankruptcy or not. Of the 64 independent variables, 9 were binary and 3 categorical. The problem can be considered to be an ordinal classification task as the two final classes are named as "GOOD" and "BAD" with respect to financial success. This means that majority of the numerical and categorical attributes (including binary ones) may be easily characterized by more preferable values with respect to "GOOD" financial success.

The dataset was balanced to a degree, so that it contained 960 bankrupt outcomes ("BAD") and 5040 not bankrupt ("GOOD"). Winnowing was used in See5, which reduced the number of variables used in models to

[10] This section taken from D.L. Olson, H.M. Moshkovich, A.I. Mechitov (2007). An experiment with fuzzy sets in data mining, *International Conference on Computational Science* Beijing, 27–29.

[11] http://www.rulequest.com

[12] Y. Shi, Y. Peng, G. Kou, Z. Chen (2005). Classifying credit card accounts for business intelligence and decision making: A multiple-criteria quadratic programming approach, *International Journal of Information Technology & Decision Making* 4:4, 581–599.

about 20. Using 50% of the data for training, See5 selected 3,000 observations at random as the training set, which was then tested on the remaining 3,000 observations in the test set. Minimum support on See5 was varied over the settings of 10, 20, and 30 cases. Pruning confidence factors were also varied, from 10% (greater pruning), 20%, 30%, and 40% (less pruning). Data was locked within nominal data runs, so that each treatment of pruning and minimum case settings was applied to the same data within each repetition. Five repetitions were conducted (thus there were 12 combinations repeated five times, or 60 runs). Each run was replicated for original crisp data, original data using fuzzy settings, ordinal crisp data, ordinal data using fuzzy settings, and categorical data (See5 would have no difference between crisp and fuzzy settings). Rules obtained were identical across crisp and fuzzy models, except fuzzy models had adjusted rule limits. For instance, in the first run, the following rules were obtained.

```
CRISP MODEL: RULE 1:  IF RevtoPayNov ≤ 11.441,      then GOOD
                      RULE 2:  IF RevtoPayNov > 11.441 AND
                                        IF CoverBal3 = 1                 then GOOD
                      RULE 3:  IF RevtoPayNov > 11.441 AND
                                        IF CoverBal3 = 0 AND
                                        IF OpentoBuyDec > 5.35129  then GOOD
                      RULE 4:  IF RevtoPayNov > 11.441 AND
                                        IF CoverBal3 = 0 AND
                                        IF OpentoBuyDec ≤ 5.35129 AND
                                        IF NumPurchDec ≤ 2.30259   then BAD
              RULE 5  ELSE                                                GOOD
```

The fuzzy model for this data set:

```
FUZZY MODEL: RULE 1:  IF RevtoPayNov ≤ 11.50565   then GOOD
                      RULE 2:  IF RevtoPayNov > 11.50565 AND
                                        IF CoverBal3 = 1                 then GOOD
                      RULE 3:  IF RevtoPayNov > 11.50565 AND
                                        IF CoverBal3 = 0 AND
                                        IF OpentoBuyDec > 5.351905 then GOOD
                      RULE 4:  IF RevtoPayNov > 11.50565 AND
                                        IF CoverBal3 = 0 AND
                                        IF OpentoBuyDec ≤ 5.351905 AND
                                        IF NumPurchDec ≤ 2.64916   then BAD
              RULE 5  ELSE                                                GOOD
```

Binary and categorical data are not affected by the fuzzy option in See5. They are considered already "fuzzified" with several possible values and corresponding membership function of 1 and 0.

Table 5.1 shows overall numbers of rules obtained and error rates for models run with initial data (numeric and categorical scales) in original

Table 5.1. See5 decision tree results-continuous and categorical data

Prune %	Minimum cases	Crisp rules	Crisp bad error	Crisp cheap error	Crisp total error	Fuzzy bad error	Fuzzy cheap error	Fuzzy total error
10		5.57	390.93	98.50	489.43	444.50	46.29	490.79
20		10.14	390.00	98.57	488.57	431.71	54.29	486.00
30		10.14	379.71	109.14	488.86	431.29	55.93	487.21
40		12.00	388.29	99.93	488.21	430.14	55.43	485.57
	10	15.00	379.55	107.70	487.25	424.30	59.80	484.10
	20	7.95	385.25	107.30	492.55	432.10	58.45	490.55
	30	5.35	394.25	92.30	486.55	440.75	46.10	486.85
Overall		9.43	386.35	102.43	488.78	432.38	54.78	487.17

crisp form, and using See5's fuzzification. In Table 5.1 there were 15 runs averaged for each pruning level, and 20 runs averaged for each minimum case level. The overall line is based on all 60 runs.

The number of rules responded to changes in pruning rates and minimum case settings as expected (the tie between 20% and 30% pruning rates can be attributed to data sampling chance). There were no clear patterns in error rates by treatment. Fuzzy models were noticeably different from crisp models in that they had higher error rates for bad cases, with corresponding improvement in error in the good cases. The overall error was tested by t-test, and the only significant differences found were that the fuzzy models had significantly greater bad error than the crisp models, and significantly less cheap error. The fuzzy models had slightly less overall average error, but given the context of credit cards, bad error is much more important. For data fit, here the models were not significantly different. For application context, the crisp models would clearly be preferred. The generalizable conclusion is not that crisp models are better, only that in this case the fuzzy models were worse, and in general one cannot count on the same results across crisp and fuzzified models.

In this case introducing fuzzy thresholds in the rules did not lead to any significant results. The usage of small fuzzy intervals instead of crispy thresholds did not significantly improve the accuracy of the model and did not provide better interpretation of the introduced "interval rules." On the other hand, crisp data was not significantly better than the fuzzy data.

The same tests were conducted with presenting relevant binary and categorical variables in an ordinal form. See5 allows stating that the categorical

scale is "[ordered]" with the presented order of attribute values corresponding to the order of final classes. The order is not derived from the data but is introduced by the user as a pre-processing step in rules/tree formation.

See5 would not allow locking across data sets, and required different setup for ordinal specification, so we could not control for data set sampling across the tests. Some categorical and/or binary variables such as "Months late" were clearly ordinal and were marked as ordinal for this experiment. Categorical variables with no clear ordinal qualities such as "State," were left nominal. Test results are given in Table 5.2.

The number of rules clearly dropped. Expected response of number of rules to pruning and minimum case settings behaved as expected, with the one anomaly at 20% pruning, again explainable by the small sample size. Total error rates within ordinal model were similar to the nominal case given in Table 5.1, with fuzzy model total error rates showing up as slightly significant (0.086 error probability) in the ordinal models.

Comparing nominal and ordinal models, the number of rules was significantly lower for ordinal models (0.010 error probability). There were no significances in errors across the two sets except for total error (ordinal models had slightly significantly lower total errors, with 0.087 error probability).

The data was categorized into three categories for each continuous variable. This in itself is another form of fuzzification. Twenty five variables were selected based upon the typical winnowing results of the original data. The same tests were conducted, although fuzzification was not used since

Table 5.2. See5 decision tree results with appropriate categorical and binary variables marked as ordinal

Prune %	Minimum cases	Crisp rules	Crisp bad error	Crisp cheap error	Crisp total error	Fuzzy bad error	Fuzzy cheap error	Fuzzy total error
10		5.86	400.21	85.93	486.14	425.64	56.57	482.21
20		5.71	403.29	86.14	489.43	427.86	56.43	484.29
30		7.93	382.79	104.14	486.93	417.93	66.14	484.07
40		9.00	395.00	94.29	489.29	427.86	58.21	486.07
	10	10.00	398.30	90.90	489.20	424.85	57.85	482.70
	20	6.50	377.10	112.55	489.65	414.60	67.55	482.15
	30	4.50	403.55	78.90	482.45	428.90	53.65	482.55
Overall		7.00	392.98	94.12	487.10	422.78	59.68	482.47

Table 5.3. See5 model results for categorical models

Prune %	Minimum cases	Rules	Bad error	Cheap error	Total error
10		5.86	400.21	85.93	486.14
20		5.71	403.29	86.14	489.43
30		7.93	382.79	104.14	486.93
40		9.00	395.00	94.29	489.29
	10	10.00	398.30	90.90	489.20
	20	6.50	377.10	112.55	489.65
	30	4.50	403.55	78.90	482.45
Overall		7.00	392.98	94.12	487.10

See5 would have no difference in applying its fuzzification routine (we did run as a check, but results were always identical). Table 5.3 provides results.

These are much worse than the results obtained in Tables 5.1 and 5.2. That is clearly because in Table 5.3, data was categorized manually, while in Tables 5.1 and 5.2 See5 software set the categorical limits. Table 5.2 involved data converted to fuzzy form by the See5 software. These are two different ways to obtain fuzzy categories. Clearly the software can select cutoff limits that will outperform ad hoc manual cutoffs.

Fuzzy Sets and Ordinal Classification

Previous experiments showed very modest improvements in the rule set derived from introducing of fuzzy intervals instead of crisp thresholds for continuous scales using See5. Interpretation of the modified rules was not "more friendly" or "more logical." Using stable data intervals was in general slightly more robust than using crisp thresholds. Considering ordinal properties of some categorical/binary attributes led to a better rule set although this did not change the fuzzy intervals for the continuous scales.

One of the more useful aspects of fuzzy logic may be the orientation on the partition of continuous scales into a pre-set number of "linguistic summaries."[13] In Au and Chan (2001)[14] this approach is used to form fuzzy rules in a classification task. The main idea of the method is to use a set of

[13] R.R. Yager (1991). On linguistic summaries of data, in G. Piatetsky-Shapiro and W.J. Frawley, Eds. *Knowledge Discovery in Databases,* Menlo Park, CA: AAAI/MIT Press, 347–363.

[14] W.-H. Au, K.C.C. Chan (2001). Classification with degree of membership: A fuzzy approach. *ICDM* 35–42.

pre-defined linguistic terms for attributes with continuous scales (e.g., "Young", "Middle", "Old" for an attribute "Age" measured continuously). In this approach, the traditional triangular fuzzy number is calculated for each instance of age in the training data set, e.g. age 23 is presented in "Young" with a 0.85 membership function and in "Middle" with a 0.15 membership function (0 in "Old"). Thus the rewritten data set is used to mine interesting IF–THEN rules using linguistic terms.

The method was used for data mining of several datasets. All continuous scales were divided into uniform linguistic terms (without corresponding expert/contextual input described in the method itself). The resulting accuracy of the models was compared with results of three other techniques including C4.5 (which is implemented in See5). It performed slightly better in accuracy (by about 5%) than See5 for two of the data sets which could be characterized as a classification task with two ordered classes (loan application and diabetes presence). The accuracy of both approaches was similar for the last data set in which classification of SALARY assumed a continuous testing variable.

One of the advantages of the proposed approach stressed by the authors is the ability of the mining method to produce rules useful for the user. The method was used to mine a database for direct marketing campaign of a charitable organization.[15] In this case the domain expert defined appropriate uniform linguistic terms for quantitative attributes. For example, an attribute reflecting the average amount of donation (AVGEVER) was fuzzified into "Very low" (0–$300), "Low" ($100–$500), "Medium" ($300–$700), "High" ($500–$900) and "Very High" (over $700). The analogous scale for frequency of donations (FREQEVER) was presented as follows: "Very low" (0–3), "Low" (1–5), "Medium" (3–7), "High" (5–9) and "Very High" (over 7). Triangular fuzzy numbers were derived from these settings for rule mining. The attribute to be predicted was called "Response to the direct mailing" and included two possible values "Yes" and "No." The database included 93 attributes, 44 having continuous scales.

Although the application of the method produced a huge number of rules (31,865) with relatively low classification accuracy (about 64%), the authors argued that for a task of such complexity the selection of several useful rules by the user of the results was enough to prove the usefulness of the process. The presented rules found useful by the user were presented as follows.

[15] K.C.C. Chan, W.-H. Au, B. Choi (2002). Mining fuzzy rules in a donor database for direct marketing by a charitable organization, *IEEE ICCI* 239–246

Rule 1: IF a donor was enrolled in any donor activity in the past (ENROLL=*YES*), THEN he/she will have RESPONSE=*YES*
Rule 2: IF a donor was enrolled in any donor activity in the past (ENROLL=*YES*) AND did not attend it (ATTENDED=*NO*), THEN he/she will have RESPONSE=*YES*
Rule 3: IF FREQEVER = *MEDIUM*, THEN RESPONSE=*YES*
Rule 4: IF FREQEVER = *HIGH*, THEN RESPONSE=*YES*
Rule 5: IF FREQEVER = *VERY HIGH*, THEN RESPONSE=*YES*

We draw two conclusions based on these results. First, if obvious ordinal dependences between final classes of RESPONSE (YES/NO) and such attributes as ENROLL, ATTENDED, and FREQEVER were taken into account the five rules could be collapsed into two without any loss of accuracy and with higher levels for measures of support and confidence: rule 1 and a modified rule 3 in the following format "IF FREQEVER is at least MEDIUM, THEN RESPONSE=YES." Second, although presented rules are "user friendly" and easily understandable, they are not as easily applicable. Overlapping scales for FREQEVER makes it difficult for the user to apply the rules directly. It is necessary to carry out one more step–agree on the number where "medium" frequency starts (if we use initial database) or a level of a membership function to use in selecting "medium" frequency if we use the "rewritten" dataset. The assigned interval of 3–5 evidently includes "High" frequency (which does not bother us) but also includes "Low" frequency which we possibly would not like to include into our mailing list. As a result a convenient approach for expressing continuous scales with overlapping intervals at the preprocessing stage may be not so convenient in applying simple rules.

If the assigning of the linguistic terms was done in a crisp way initially, the problem would be a traditional classification problem with nominal scales. Ordered classes would suggest that the quality of the result may be improved if appropriate ordinal scales are used for some attributes.

The ordinal classification assumes that "cases with better characteristics should be placed in a better class". Formally the ordinal classification problem can be presented as follows. Let U (universe) present all possible objects in the task. $X \subseteq U$ is any subset of these objects (data items in a data set). Objects from X are distributed among k classes: C_1, C_2,… C_k, indicating the degree in which object satisfies the overall output quality (from the lowest to the highest). This means that if $x, y \in X$, and $C(x) > C(y)$, object x has higher overall output quality than object y. Each object is described by a set of attributes $A = \{A_1, A_2, \ldots A_p\}$. Each attribute A_i has an ordinal scale S_i (possible values are rank ordered from the lowest to the highest in quality against this attribute). $A_i(x) \in S_i$, $A_i(y) \in S_i$, and $A_i(x) > A_i(y)$ means object x has higher quality on attribute A_i than object y.

In the loan application case, ordinal classification may tell us, for example, that a middle-aged applicant with low risk and high income has a better chance to pay the loan on-time than a young applicant with low risk and high income. For the mail campaign we can expect that more frequent donations in the past should lead to a better chance of positive response or that if those who enroll in the donor events but don't come often have a positive response, it would be even more probable for those who enroll and attend donor events.

This presentation of the ordinal classification task allows use of this knowledge to make some additional conclusions about the quality of the training set of objects. Ordinal classification allows introduction of the notion of the consistency of the training set as well as completeness of the training set. In the case of the ordinal classification task quality of consistency in a classification (the same quality objects should belong to the same class) can be essentially extended: all objects with higher quality among attributes should belong to a class at least as good as objects with lower quality. This condition can be easily expressed as follows: if $A_i(x) \geq A_i(y)$ for each $i = 1, 2, \ldots, p$, then $C(x) \geq C(y)$.

We can also try to evaluate representativeness of the training set by forming all possible objects in U (we can do that as we have a finite number of attributes with a small finite number of values in their scales) and check on the proportion of them presented in the training set. It is evident that the smaller this proportion the less discriminating power we'll have for the new cases. We can also express the resulting rules in a more summarized form by lower and upper border instances for each class.

Advantages of using ordinal scales in an ordinal classification task do not lessen advantages of appropriate fuzzy set techniques. Fuzzy approaches allow softening strict limitations of ordinal scales in some cases and provides a richer environment for data mining techniques. On the other hand, ordinal dependences represent essential domain knowledge which should be incorporated as much as possible into the mining process. In some cases the overlapping areas of attribute scales may be resolved by introducing additional linguistic ordinal levels. For example, we can introduce an ordinal scale for age with the following levels: "Young" (less than 30), "Between young and medium" (30–40), "Medium" (40–50), "Between medium and old" (50–60) and "Old" (over 60). Though it will increase the dimensionality of the problem, it would provide crisp intervals for the resulting rules.

Ordinal scales and ordinal dependences are easily understood by humans and are attractive in rules and explanations. These qualities should be especially beneficial in fuzzy approaches to classification problems with ordered classes and "linguistic summaries" in the discretization process.

The importance of ordinal scales for data mining is evidenced by appearance of this option in many established mining techniques. See5 includes the variant of ordinal scales in the problem description (See5).

Fuzzy Association Rules[16]

With the rapid growth of data in enterprise databases, making sense of valuable information becomes more and more difficult. KDD (Knowledge Discovery in Databases) can help to identify effective, coherent, potentially useful and previously unknown patterns in large databases.[17] Data mining plays an important role in the KDD process, applying specific algorithms for extracting desirable knowledge or interesting patterns from existing datasets for specific purposes. Most of the previous studies focused on categorical attributes. Transaction data in real-world applications, however, usually consist of quantitative attributes, so some data mining algorithms for quantitative values also have been proposed, where the algorithm finds association rules by partitioning the attribute domain, combining adjacent partitions and then transforming the problem into a binary state.[18]

Mining fuzzy association rules for quantitative values has long been considered by a number of researchers, most of whom based their methods on the important APriori algorithm.[19] Each of these researchers treated all attributes (or all the linguistic terms) as uniform. However, in real-world applications, the users perhaps have more interest in the rules that contain fashionable items. Decreasing minimum support *minsup* and minimum confidence *minconf* to get rules containing fashionable items is not best, because the efficiency of the algorithm will be reduced and many uninteresting rules will be generated simultaneously.[20] Weighted quantitative

[16] This section based on work by Meiping Xie.

[17] W.J. Frawley, G. Piatetsky-Shapiro, C.J. Matheus (1991). Knowledge discovery in databases: an overview, *The AAAI Workshop on Knowledge Discovery in Databases*, 1–27.

[18] R. Srikant, R. Agrawal (1996). Mining quantitative association rules in large relational tables, *The 1996 ACM SIGMOD International Conference on Management of Data.* Montreal, Canada, 1–12.

[19] C. Kuok, A. Fu, H. Wong (1998). Mining fuzzy association rules in databases, *ACM SIGMOD Record* 27, 41–46; T.-P. Hong, C.-S. Kuo, S.-C. Chi (2001). Trade-off between computation time and numbers of rules for fuzzy mining from quantitative data. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 9, 587–604.

[20] R. Ladner, F.E. Petry, M.A. Cobb (2003). Fuzzy set approaches to spatial data mining of association rules, *Transactions in GIS* 7, 123–138.

association rules mining based on a fuzzy approach has been proposed (by Genesei) using two different definitions of weighted support: with and without normalization.[21] In the non-normalized case, he used the product operator for defining the combined weight and fuzzy value. The combined weight or fuzzy value is very small and even tends to zero when the number of items is large in a candidate itemset, so the support level is very small, this will result in data overflow and make the algorithm terminate unexpectedly when calculating the confidence value. In the normalized case, Gyenesei used the geometric mean of item weights as the combined weight, which avoided the problem that the combined weight is very small when the number of items is large in a candidate itemset. However, the support is then very small because the product operator is included in the calculation of the combined fuzzy value. At the same time the problem that a subset of a large itemset may not be large must be addressed. This increases the computation time complexity of the algorithm. In addition, Because the support is small and difficult to predict for both of the two proposed methods, the suitable *minsup* is also hard to provide. Moreover, the proposed two algorithms use categorical regions in the data mining procedure for each categorical attribute, the corresponding membership values is 0 or 1, which loses the flexibility of fuzzy representations of real data, especially when the number of categorical regions is large.

Demonstration Model

A simple example is given to illustrate a fuzzy data mining algorithm to generate association rules. The example is based on 10 data points of applicants for appliance loans, whose attributes are *age, income, assets, debts, credit* (with red for bad credit, amber for some credit problems, and green for clean credit record), *Want* (the amount requested in the appliance loan applications) and *Result* {it is 1 if all payments were received on time and 0 if not (late or default)}. A new attribute *Risk* is generated based on *Assets*, *Debts* and *Want* using the formula *Risk = Assets- Debts - Want*. We will demonstrate with the following data in Table 5.4, which has both quantitative and categorical attributes.

The attributes *Age*, *Income* and *Risk* have three fuzzy regions each. *Credit* and *Result* have two fuzzy regions. Weights for each attribute

[21] A. Gyenesei (2000). Mining weighted association rules for fuzzy quantitative items, *Proceedings of PKDD Conference, September 13–16, 2000,* Lyon, France, 416–423.

Table 5.4. Initial data set

Case	Age	Income	Risk	Credit	Result
1	20	52,623	–38,954	red	0
2	26	23,047	–23,636	green	1
3	46	56,810	45,669	green	1
4	31	38,388	–7,968	amber	1
5	28	80,019	–35,125	green	1
6	21	74,561	–47,592	green	1
7	46	65,341	58,119	green	1
8	25	46,504	–30,022	green	1
9	38	65,735	30,571	green	1
10	27	26,047	–6	red	1

which could be based on some multiple criteria method, or based on direct expert opinion expressing relative importance. Here weights are assumed to be *Credit* (0.8) ≥ *Risk* (0.7) ≥ *Income* (0.55) ≥ *Age* (0.45). You can assign different weights to different fuzzy regions of the same attribute, such as assigning fuzzy region-young of the attribute *Age* a larger weight value when users are just interested in young people's loan applications, so more association rules involving young applicants will be obtained.

Step 1: Transform quantitative or categorical values into fuzzy values. Category limits would need to be established, based on past experience or on some calculations such as dividing the data into equal sized groups. Here attribute *Age* is transformed into fuzzy categories Young (R_{11}), Middle (R_{12}), and Old (R_{13}); attribute *Income* into High (R_{21}), Middle (R_{22}), and Low (R_{23}); attribute *Risk* into High (R_{31}), Middle (R_{32}), and Low (R_{33}); attribute *Credit* into Good (R_{41}) and Bad (R_{42}); and outcome into On Time (R_{51}) and Default (R_{52}).

Step 2: Calculate weighted support $Sup(R_{jk})$, values given in Table 5.5.

Table 5.5. Support values by category

Category		Weight	$Sup(R_{jk})$
Age Young	R_{11}	0.45	0.261
Age Middle	R_{12}	0.45	0.135
Age Old	R_{13}	0.45	0.059
Income High	R_{21}	0.55	0.000
Income Middle	R_{22}	0.55	0.490
Income Low	R_{23}	0.55	0.060
Risk High	R_{31}	0.70	0.320
Risk Middle	R_{32}	0.70	0.146

(Continued)

Table 5.5. (Continued)

Category		Weight	Sup(R_{ik})
Risk Low	R_{33}	0.70	0.233
Credit Good	R_{41}	0.80	0.576
Credit Bad	R_{42}	0.80	0.224

$Sup(R_{jk})$ is the proportion of cases with the given condition, multipled by the weight for the attribute in question.

The complete list of categories is:

$C_1 = \{(R_{11}), (R_{12}), (R_{13}), (R_{21}), (R_{22}), (R_{23}), (R_{31}), (R_{32}), (R_{33}), (R_{41}), (R_{51}), (R_{52})\}$.

Step 3: Let *minsup* = 0.25, If $Sup(R_{jk}) \geq minsup$, then put R_{jk} in the set of large 1-itemsets (L_1). Here $L_1 = \{(R_{11}), (R_{22}), (R_{31}), (R_{41}), (R_{42}), (R_{51})\}$.

Step 4: If L_1 is null, stop.

Step 5: Join itemsets in L_1 to generate the candidate set C_2 of category pairs.

$C_2 = \{(R_{11}, R_{22}), (R_{11}, R_{31}), (R_{11}, R_{41}), (R_{11}, R_{51}), (R_{22}, R_{31}), (R_{22}, R_{41}), (R_{22}, R_{51}), (R_{31}, R_{41}), (R_{31}, R_{51}), (R_{41}, R_{51})\}$.

Note that itemsets such as (R_{11}, R_{12}) having categorical classes of the same attribute would not be retained in C_2.

Step 6: Perform the following substeps for each newly formed candidate 2-itemset.

(a) Calculate the membership value of each transaction datum. Take (R_{11}, R_{22}) as an example. The derived membership value is calculated for case 1, which has membership function value of 1 for R_{11} and membership function value of 1 for R_{22} is: $\min(0.45 \times 1, 0.55 \times 1) = 0.45$.

(b) Calculate the support value of each candidate 2-itemset in C_2. $Sup((R_{11}, R_{22})) = 0.235$. The results for other 2-itemsets can be derived in similar fashion as in Table 5.6.

Table 5.6. Support values for 2-itemsets

2-itemset	Support	2-itemset	Support
(R_{11}, R_{22})	0.235	(R_{22}, R_{41})	0.419
(R_{11}, R_{31})	0.207	(R_{22}, R_{51})	0.449
(R_{11}, R_{41})	0.212	(R_{31}, R_{41})	0.266
(R_{11}, R_{51})	0.230	(R_{31}, R_{51})	0.264
(R_{22}, R_{31})	0.237	(R_{41}, R_{51})	0.560

(c) Check if the support value of each candidate 2-itemset is larger than or equal to *minsup*, then put it in L_2. Here $L_2 = \{(R_{22}, R_{41}), (R_{22}, R_{51}), (R_{31}, R_{41}), (R_{31}, R_{51}), (R_{41}, R_{51})\}$.

Step 7: Since L_2 is not null, repeat Steps 5–6 to find L_3. C_3 is first generated from L_2, and 3-itemsets in C_3. The four support values are 0.417 for (R_{22}, R_{41}, R_{51}), 0.198 (R_{22}, R_{31}, R_{41}), 0.196 for (R_{22}, R_{51}, R_{31}), and 0.264 for (R_{31}, R_{41}, R_{51}). Thus $L_3 = \{(R_{22}, R_{41}, R_{51}), (R_{31}, R_{41}, R_{51})\}$. The next step is for C_4, with $C_4 = \{(R_{22}, R_{31}, R_{41}, R_5)\}$ but the support value – 0.1957 < *minsup*, so L_4 is null. Then go to *Step 8*.

Step 8: Collect the large itemsets together. Here only L_1, L_2 , and L_3 exist.

Step 9: Construct association rules for each large *2*-itemset and *3*-itemset using the following substeps:

(a) Form 16 possible association rules as follows:

$R_{22} \rightarrow R_{41}$; $R_{41} \rightarrow R_{22}$; $R_{22} \rightarrow R_{51}$; $R_{51} \rightarrow R_{22}$;
$R_{31} \rightarrow R_{41}$; $R_{41} \rightarrow R_{31}$; $R_{31} \rightarrow R_{51}$; $R_{51} \rightarrow R_{31}$;
$R_{41} \rightarrow R_{51}$; $R_{51} \rightarrow R_{41}$;
$R_{41}, R_{51} \rightarrow R_{22}$; $R_{22}, R_{51} \rightarrow R_{41}$; $R_{41}, R_{22} \rightarrow R_{51}$;
$R_{31}, R_{41} \rightarrow R_{51}$; $R_{51}, R_{41} \rightarrow R_{31}$; $R_{31}, R_{51} \rightarrow R_{41}$.

(b) Calculate the confidence value for each association rule. The confidence for the first association rule $R_{22} \rightarrow R_{41}$ for example is minimum support between R_{22} and R_{41} over all ten cases (which equals 0.4187) divided by the support for R_{22} (which equals 0.49), yielding a confidence score of 0.855. Table 5.7 gives confidence scores.

Table 5.7. Confidence scores

L_2	conf	L_3	conf
$R_{22} \rightarrow R_{41}$	0.855	$R_{41}, R_{22} \rightarrow R_{51}$	0.995
$R_{41} \rightarrow R_{22}$	0.727	$R_{41}, R_{51} \rightarrow R_{22}$	0.744
$R_{22} \rightarrow R_{51}$	0.916	$R_{22}, R_{51} \rightarrow R_{41}$	0.928
$R_{51} \rightarrow R_{22}$	0.697	$R_{31}, R_{41} \rightarrow R_{51}$	0.993
$R_{31} \rightarrow R_{41}$	0.831	$R_{31}, R_{51} \rightarrow R_{41}$	1.000
$R_{41} \rightarrow R_{31}$	0.462	$R_{51}, R_{41} \rightarrow R_{31}$	0.472
$R_{31} \rightarrow R_{51}$	0.825		
$R_{51} \rightarrow R_{31}$	0.410		
$R_{41} \rightarrow R_{51}$	0.972		
$R_{51} \rightarrow R_{41}$	0.870		

Step 10: Assume *minconf* = 0.90. Output the relative and interesting association rules with $Conf \geq minconf$. The following rules are output to users, the number in parentheses is confidence value.

If Income is middle, then payment will be received on time $R_{22} \rightarrow R_{51}$; (91.6%)
If Credit is good, then payment will be received on time $R_{41} \rightarrow R_{51}$; (97.2%)
If Income is middle and Credit is good, then payment will be received on time R_{41}, $R_{22} \rightarrow R_{51}$; (99.5%)
If Risk is high and Credit is good, then payment will be received on time R_{31}, $R_{41} \rightarrow R_{51}$; (99.25%)

Two other itemsets passed the confidence limit (R_{22}, $R_{51} \rightarrow R_{41}$; R_{31}, $R_{51} \rightarrow R_{41}$) but were not interesting in that they did not yield outcome as a conclusion.

The rules derive from the data in database, which can provide meaningful information that benefit decision-making of many aspects.

Computational Results

The number of association rules decreases along with an increase in *minsup* (or *minconf*) under a given specific *minconf* (or *minsup*), which shows an appropriate *minsup* (or *minconf*) can constraint the number of association rules and avoid the occurrence of some association rules that cannot yield a decision.

The number of unweighted large itemsets is always larger than that of the corresponding number for the weighted model, whether it is large 1-itemset (L1), large 2-itemset (L2), large 3-itemset (L3) or summation.

The number of large itemsets for a given item increases along with an increase in the item's weight, which is also as we expected. Each large itemset is a potential association rule, so the bigger the weight for an item, the more the relative association rules are under some specific *minconf*. This demonstrates how users' preference can be added into the procedure of data mining through adjusting weights.

Testing

We selected 550 cases randomly used for inducing rules from 650 cases in original data set, the remaining 100 cases are used for testing accuracy of

the induced rules of the proposed method by measuring the average percentage of correct predictions. The procedure used was to randomly assign a set of weights to each fuzzy region of each attribute, let *minsup* >= 0.35 and cut set of the fuzzy set occurring when transforming quantitative value to fuzzy value be 0.7. The accuracy of the proposed method varies along with *minsup* and *minconf*. The accuracy of the proposed method increases on the whole with an increase in *minsup* and *minconf*, which is not counter-intuitive and shows the contributions (functions) of *minsup* and *minconf* to accuracy. But there is some variance or exception at the latter part of the curve, because some statistics features disappear (such as cases with only one or two rules left) when *minsup* is large enough, at the same time, it is just a roughly result of accuracy testing here, because our rule is with linguistic terms and we have to use the cut set technology of fuzzy sets and the average percentage of correct rules predictions for testing accuracy. Although there is some information loss in the processing conducted, some base features of the curves can be seen.

Inferences

Fuzzy expressions of association rules match the way in which human subjects think. Fuzzy expressions thus allow users to naturally assign emphases on items matching their true preferences. We showed a fuzzy weighted association rule mining method for quantitative or categorical data, which is more flexible, natural and understandable. For the proposed method, it is easy to add users' guide to data mining procedure, so it is quicker to find association rules interested by users, and reduce the work burden of choosing interesting rules. The proposed data mining approach satisfies the downward closure property, which decreases computation time, at the same time, uninteresting rules can be pruned early because of assigning weights to items, which also results in the reduction of execution time. Because of the adoption of minimum operator, the problem of data overflowing can be avoided here. In the proposed algorithm, we assign fuzzy regions to each categorical attribute, which can describe categorical attribute in linguistic language more flexible according to situations especially when the number of categorical regions is large.

The construction of membership functions in fuzzy association rules mining has always been a bottleneck. The method presented assumes advance knowledge of membership limits. We could make the construction of membership functions more open or automatic according to distribution of data in data set or users' opinions, based on some existing research on

automatically deriving membership functions[22] or making probability and fuzzy set theory work in concert.[23] In the simple example demonstration in Sect. 4, we know the value of Result is 1 if all payment were received on time and 0 if not (late or default). There can be complications when expressing "late or default" using crisp data – 0, for "default" is worse than "late". But if assigning two fuzzy regions (*Good* and *Bad*) to the attribute *Result* as did in proposed method in this paper, for fuzzy region-Bad, there still exists information missing when expressing the situation that result is 0 using fuzzy data 0.9 (for "late" is always better than "default", which will 0.9 denote?), the situation will be improved if using an interval-value. Attribute *Credit* has the same problem. So we could also extend the proposed method to vague weighted data mining algorithm to find fuzzy association rules for quantitative value.

Conclusions

Fuzzy representation is a very suitable means for humans to express themselves. Many important business applications of data mining are appropriately dealt with by fuzzy representation of uncertainty. We have reviewed a number of ways in which fuzzy sets and related theories have been implemented in data mining. The ways in which these theories are applied to various data mining problems will continue to grow.

Our contention is that fuzzy representations better represent what humans mean. Our brief experiment was focused on how much accuracy was lost by using fuzzy representation in one application – classification rules applied to credit applications. While we expected less accuracy, we found that the fuzzy models (as applied by See5 adjusting rule limits) usually actually were more accurate. Models applied to categorical data as a means of fuzzification turned out less accurate in our small sample. While this obviously cannot be generalized, we think that there is a logical explanation. While fuzzification will not be expected to yield better fit to training data, the models obtained by using fuzzification will likely be more robust, which is reflected in potentially equal if not better fit on test data. The results of these preliminary experiments indicate that implementing various forms of fuzzy analysis will not necessarily lead to reduction in classification accuracy.

---

[22] T.-P. Hong, S.-L. Wang (2000). Determining appropriate membership functions to simplify fuzzy induction, *Intelligent Data Analysis* 4, 51–66.

[23] N.D. Singpurwalla, J.M. Booker (2004). Membership function and probability measures of fuzzy sets, *Journal of the American Statistical Association* 9, 867–877.

6 Rough Sets

Rough set analysis is a mathematical approach that is based on the theory of rough sets first introduced by Pawlak (1982).[1] The purpose of rough sets is to discover knowledge in the form of business rules from imprecise and uncertain data sources. Rough set theory is based on the notion of indiscernibility and the inability to distinguish between objects, and provides an approximation of sets or concepts by means of binary relations, typically constructed from empirical data. Such approximations can be said to form models of our target concepts, and hence in the typical use of rough sets falls under the bottom-up approach to model construction. The intuition behind this approach is the fact that in real life, when dealing with sets, we often have no means of precisely distinguishing individual set elements from each other due to limited resolution (lack of complete and detailed knowledge) and uncertainty associated with their measurable characteristics. As an approach to handling imperfect data, rough set analysis complements other more traditional theories such as probability theory, evidence theory, and fuzzy set theory.

The rough set philosophy is founded on the assumption that we associate some information (data, knowledge) with every object of the universe of discourse. Objects, which are characterized by the same information, are indiscernible in view of the available information about them. The indiscernibility relation generated in this way is the mathematical basis for the rough set theory. Any set of all indiscernible objects is called an elementary set, and forms a basic granule of knowledge about the universe. Any set of objects being a union of some elementary sets is referred to as crisp or precise set otherwise the set is rough (imprecise or vague). Consequently, each rough set has boundary-line cases (i.e., objects), which cannot be classified with complete certainty as members of the set.[2]

---

[1] Z. Pawlak (1982). Rough sets, *International Journal of Information and Computer Sciences* 11, 341–356.

[2] A theoretical derivation of rough sets is provided by Z. Pawlak (1991). *Rough Set: Theoretical Aspects and Reasoning About Data*. Dordrecht, the Netherlands: Kluwer Academic Publishers.

A Brief Theory of Rough Sets

Statistical data analysis faces limitations in dealing with data with high levels of uncertainty or with non-monotonic relationships among the variables. Pawlak provided a rule-based approach using Rough sets theory to address these problems. The original idea behind his Rough sets theory was "… vagueness inherent to the representation of a decision situation. Vagueness may be caused by granularity of the representation. Due to the granularity, the facts describing a situation are either expressed precisely by means of 'granule' of the representation or only approximately."[3] The vagueness and imprecision problems are present in the information that describes most real world applications.

Information System

In rough sets, an information system is a representation of data that describes some object. An information system S is composed of a 4-tuple

$$S = < U, Q, V, f >$$

where U is the closed universe of a N objects $\{x_1, x_2, \ldots, x_N\}$, a nonempty finite set; Q is a nonempty finite set of n attributes $\{q_1, q_2, \ldots, q_n\}$ (that uniquely characterizes the objects); $V = U_{q \in Q} V_q$ where V_q is the value of the attribute q; $f : U \times Q \rightarrow V$ is the total decision function called the information function such that $f(x, q) \in V_q$ for every $q \in Q$, $x \in U$.[4] Pawlak (2000) gave a concrete instance in the form of a universe of 6 stores with four attributes (sales force empowerment {no, medium, or high}, merchandise quality {good or average}, presence of high traffic {yes, no}, and categorization of profit {profit, loss}).[5] The six stores are the universe U, the first three attributes are Q, their possible values V, and the profit category f.

Any pair (q, v) for $q \in Q$, $v \in V_q$ is called the descriptor in an information system S. The information system can be represented as a finite data table, in which the columns represent the attributes, the rows represent the objects and the cells represent the attribute values $f(x, q)$. Thus, each row in the table describes the information about an object in S.

---

[3] Z. Pawlak, R. Słowinski (1994). Decision analysis using rough sets, *International Transactions of Operational Research* 1:1, 107–114.

[4] Pawlak (1991), op cit.

[5] Z. Pawlak (2000). Rough sets and decision analysis, *INFOR* 38:3, 132–144.

Indiscernibility. If we let $S = < U, Q, V, f >$ be an information system, $A \subseteq Q$ be a subset of attributes, and $x, y \in U$ are objects, then x and y are indiscernible by the set of attribute A in S if and only if $f(x, a) = f(y, a)$ for every $a \in A$. Every subset of variables A determines an equivalence relation of the universe U, which is referred to indiscernibility relation. For any given subset of attributes $A \subseteq Q$ the IND(A) is an equivalence relation on universe U and is called an indiscernibility relation. The indiscernibility relation IND(A) can be defined as

$$\text{IND}(A) = \{(x, y) \in U \times U : \text{for all } a \in A, f(x, a) = f(y, a)$$

If the pair of objects (x, y) belongs to the relation IND(A) then objects x and y are called indiscernible with respect to attribute set A. In other words, we cannot distinguish object x from y based on the information contained in the attribute set A.

Decision Table

In rough sets, a special case of information systems is called a decision table (or decision system) if the attribute set Q is divided into two disjoint sets, conditional attribute set C and decision attribute set D, such that $C \cup D = Q$ and $C \cap D = \varnothing$. For instance, for a classification type problem, the set C would represent the characterization of the pattern (the independent variables) such as the financial indicators of a company, and the set D would represent the classification decision (bankruptcy or not). A decision table can be denoted as

$$\text{DT} = < U, C \cup D, V, f >$$

where C is a set of condition attributes (a non-empty set of discrete valued independent variables); D is a set of decision attributes (a non empty set of discrete valued decision variables); $V = U_{q \in C \cup D} V_q$, where V_q is the set of discrete values of the attribute $q \in Q$; $f : U \times (C \cup D) \rightarrow V$ is a total decision function such that $f(x, q) \in V_q$ for every $q \in Q$ and $x \in V$. A decision table can also be represented as $(U, C \cup D)$ or DT_C where the set C denotes the conditional attribute set while the set D in general may represent the decision attribute set.

A decision table is called deterministic if each object's decision attribute values can uniquely be specify by some combination of the conditional attribute values. On the other hand, a decision table is called non-deterministic if a number of decision attribute values may be taken for a given conditional attribute values. Some of the non-deterministic decision

tables may be decomposed into two sub tables, deterministic and totally non-deterministic, where a totally non-deterministic decision table does not contain any deterministic sub table.

(Approximation) Some object cannot be completely distinguished from the rest of the objects (in line with the decision variable values) in terms of the available conditional attributes. Their designation can only be roughly (approximately) defined; leading to the idea of rough sets based approximation. The fundamental underlying of rough sets consists of the approximation of a set of objects by a pair of sets, called lower and upper approximation sets.

Let $S = < U, Q, V, f >$ be an information system. A given subset of attributes $A \subseteq Q$ determines the approximation space AS = (U, IND(A)) in S. For a given $A \subseteq Q$ and $X \subseteq U$ the A-lower approximation $\underline{A}X$ of set X in AS and the A-upper approximation $\overline{A}X$ of X in AS are defined as follows:

$$\underline{A}X = \{ x \in U : [x]_A \cap X \neq \varnothing \} = \cup \{ Y \in A^* : Y \subseteq X \}$$

$$\overline{A}X = \{ x \in U : [x]_A \cap X \neq \varnothing \} = \cup \{ Y \in A^* : Y \cap X \neq \varnothing \}$$

The lower approximation of $\underline{A}X$ of set X is the union of all those elementary sets, each of which contained by X. For any $x \in \underline{A}X$, it is certain that x belongs to X. In other words, the lower approximation of $\underline{A}X$ of set X contains all objects that, based on the information content of attributes A, can be classify as belonging to the concept of X with complete certainty.

The upper approximation $\overline{A}X$ of set X is the union of those elementary sets each of which has non-empty intersection with X. For any $x \in \overline{A}X$, we can only say that x can "possibly" belong to X. In other words, the upper approximation $\overline{A}X$ of set X contains all objects that based on the information contained in A cannot be classified as not belonging to the concept X.

The A-boundary region of a set $X \subseteq U$ in AS (the doubtful region of IND(A)) is defined as follows:

$$BN_A(X) = \overline{A}X - \underline{A}X$$

For any $x \in U$ belonging to $BN_A(X)$, it is impossible to determine that x belongs to X based on the description set of the elementary set of IND(A).

The A-lower approximation of set X is the possibly (the greatest) definable set in A of set X and the A-upper approximation of set X is the certainty (the smallest) definable set in A of set X. The A-boundary is a doubtful region in A of set X.

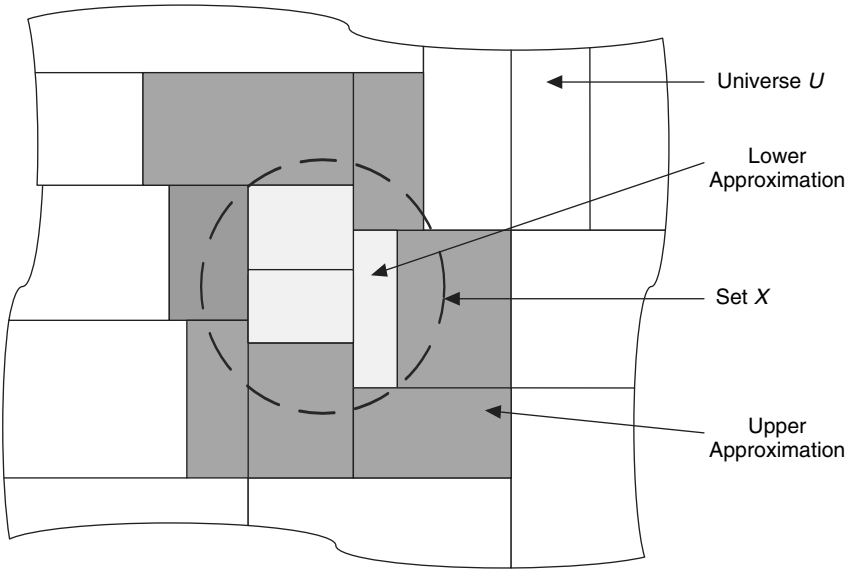


Fig. 6.1. A pictorial representation of a rough set

Given the approximation space AS for $A \in U$ and a subset $X \subseteq U$, one can divide a set U into three regions as follows:

1. $\underline{A}X$: The A-positive region $POS_A(X)$ of X in S.
2. $AX - \underline{A}X$: The A-boundary region $BN_A(X)$ of X in S.
3. U – $\overline{A}X$: The A-negative region $NEG_A(X)$ of X in S.

If $\overline{A}X = \underline{A}X$ then one can say that $X \in U$ is A-exactly approximated in AS. In this case the A-boundary region $BN_A(X) = \varnothing$. If $\overline{A}X \neq \underline{A}X$ then one can say $X \in U$ is A-roughly approximated in AS. In this case the A-boundary region $BN_A(X) \neq \varnothing$.

Some Exemplary Applications of Rough Sets

Most of the successful applications of rough sets are in the field of medicine, more specifically, in medical diagnosis or prediction of outcomes. Rough sets have been applied to analyze a database of patients

with duodenal ulcer treated by highly selective vagotomy1 (HSV).[6] The goal was to predict the long-term success of the operation, as evaluated by a surgeon into four outcome classes. This successful HSV study is still one of few data analysis studies, regardless of the methodology, that has managed to cross the clinical deployment barrier. There have been a steady stream of rough set applications in medicine. Some more recent applications include analysis of breast cancer[7] and other forms of diagnosis,[8] as well as support to triage of abdominal pain[9] and analysis of Medicaid Home Care Waiver programs.[10]

In addition to medicine, Rough Sets have also been applied to a wide range of application areas to include real estate property appraisal,[11] predicting bankruptcy[12] and predicting the gaming ballot outcomes.[13] Rough sets have been applied to identify better stock trading timing,[14] to enhance support vector machine models in manufacturing process document retrieval,[15] and to evaluate safety performance of construction firms.[16] Rough sets have thus been useful in many applications.

---

[6] J. Fibak, Z. Pawlak, K. Słowinski, R. Słowinski (1986). Rough sets based decision algorithm for treatment of duodenal ulcer by HSV, *Biological Sciences* 34: 227–249.

[7] A.E. Hassanien (2003). Intelligent data analysis of breast cancer based on rough set theory, *International Journal on Artificial Intelligence Tools* 12:4, 479–493.

[8] S. Tsumoto (2004). Mining diagnostic rules from clinical databases using rough sets and medical diagnostic model, *Information Sciences* 162:2, 65–80.

[9] W. Michalowski, S. Rubin, R. Slowinski, S. Wilk (2003). Mobile clinical support system for pediatric emergencies, *Decision Support Systems* 36:2, 161–176.

[10] M. Kitchener, M. Beynon, C. Harrington (2004). Explaining the diffusion of Medicaid Home Care Waiver Programs using VPRS decision rules, *Health Care Management Science* 7:3, 237–244.

[11] M. d'Amato (2002). Appraising property with rough set theory, *Journal of Property Investment & Finance* 20:4, 406–418.

[12] R. Slowinski, C. Zopounidis (1995). Application of the rough set approach to evaluation of bankruptcy risk, *International Journal of Intelligent Systems in Accounting, Finance and Management* 4, 27–41; A.I. Dimitras, R. Slowinski, R. Susmaga, C. Zopounidis (1999). Business failure prediction using rough sets, *European Journal of Operational Research*, 114, 263–280.

[13] D. Delen, E. Sirakaya (2006). Determining the efficacy of data mining methods in predicting gaming ballot outcomes, *Journal of Hospitality and Tourism Research* 30:3, 313–332.

[14] L. Shen, H.T. Loh (2004). Applying rough sets to market timing decisions, *Decision Support Systems* 37:4, 583–597.

[15] C.-C. Huang, T.-L. Tseng, H.-F. Chuang, H.-F. Liang (2006). Rough-set-based approach to manufacturing process document retrieval, *International Journal of Production Research* 44:14, 2889–2911.

Rough Sets Software Tools

Even though none of the top commercial data mining toolkits (e.g., SAS Enterprise Miner, SPSS Clementine, Statistica Data Miner) have an implementation of Rough sets in their classification algorithms list, there are several small commercial applications and several free software tools that employ rough sets theory.

- *Datalogic*, professional tool for knowledge acquisition, classification, predictive modeling based on rough sets (commercial). Can be accessed at http://www.reduct.com.
- *Rough set exploration system (RSES)*, contains classification based on rough sets, decision tree, LTF-network, instance based classification and data discretization (free for non-commercial use). This software can be found at http://logic.mimuw.edu.pl/~rses/.
- *ROSETTA* is a toolkit for analyzing tabular data within the framework of rough set theory. It is designed to support the overall data mining and knowledge discovery process: from initial browsing and preprocessing of the data, via computation of minimal attribute sets and generation of if-then rules or descriptive patterns, to validation and analysis of the induced rules or patterns (free for non-commercial use). It can be found at http://rosetta.lcb.uu.se/general/.
- *Grobian*, user-friendly software to analyse data with rough set technology (free). Can be found at http://www.infj.ulst.ac.uk/~cccz23/grobian/grobian.html.

The Process of Conducting Rough Sets Analysis

In order to better understand this relatively new machine learning technique, the main steps in conducting rough set analysis is given in the form of a workflow in Fig. 6.2. In addition to the main processing steps, Fig. 6.2 also pictorially illustrates the flow of artifacts (e.g., processed data, discretization specification, reducts, decision rules, etc.) in developing prediction models using rough sets.

[16] C.M. Tam, T.K.L. Tong, K.K. Chan (2006). Rough set theory for distilling construction safety measures, *Construction Management & Economics* 24:11, 1199–1206.

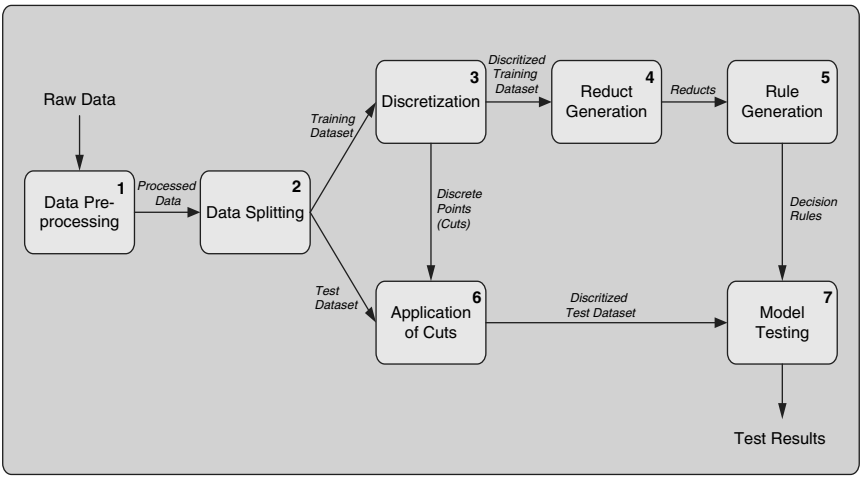


Fig. 6.2. Process map and the main steps of the rough sets analysis

The details about the processing steps are given below.

1. Data Pre-Processing

The first step involves creation of information system or decision systems, depending on what one intends to do with the data. They are generally tables with attributes and a decision variable in case of decision systems. Information systems and decision systems are both represented by the same structure. It is up to the algorithms that operate on such objects to define how the table is to be interpreted. Internally, all value sets for all attributes are represented as integers. Thus, an information system can essentially be perceived as a matrix with integer entries. A data dictionary, associated with each information system, handles the mapping between the internal integer representations and their meanings in the modeling domain. Dictionaries are described in next paragraph. In addition to the attribute values, a decision table object also holds information per attribute of its type (condition or decision) and masking status (enabled or disabled). Attributes that are disabled are “invisible” to the algorithms that operate on the table.

Decision Table may have missing values, which need to be taken care of. Some of the commonly used algorithms for filling these missing values are as follows.

- *Removal completer*: Removes all objects that have one or more missing values.
- *Mean completer*: Substitutes missing values for numerical attributes with the mean value of all observed entries for that attribute. For string attributes, missing values are substituted by the "mode" value, i.e., the most frequently occurring value among the observed entries for that attribute.
- *Conditioned mean completer*: Similar to the algorithm described above, but the computations of the mean and mode values are conditioned to the decision classes.
- *Combinatorial completer*: Expands each missing value for each object into the set of possible values. That is, an object is expanded into several objects covering all possible combinations of the object's missing values.
- *Conditioned combinatorial completer*: Similar to the one above but the sets of values are conditioned to decision class

2. Data Partitioning

Data is partitioned into training and testing sets. This can be as simple as a single split of half and half and as complicated as splitting the data into k disjoint subsection (a.k.a. k-fold cross validation) each of which having one or more variables used as strata (in order to maintain the proportional representations of various sub-grouping) for stratified random sampling, leading to k number of experiments. Regardless of the level of complexity employed, the main idea is to use a representative sample of the data to develop the model (cuts, reducts, and decision rules) and apply this model to the set aside test data to assess the predictive ability of the model. Obtaining compact models is important for interpretability, and thus the idea of reducts is useful. Reducts are rules using a subset of all available attributes that provide the bulk of accurate prediction.

3. Discretization

Discretization is the process of converting continuous valued variables to discrete values where limited numbers of labels are used to represent the original variables. The discrete values can have a limited number of intervals in a continuous spectrum, whereas continuous values can be infinitely many. Many prefer using discrete values as opposed to continuous ones in developing prediction models because (i) discrete values are closer to a

knowledge-level representation; (ii) data can be reduced and simplified through discretization, (iii) for both users and experts, discrete features are easier to understand, use, and explain; and finally and probably the most importantly, some of the machine learning algorithms such as Rough sets can only work with discrete valued variables. There are a number of different algorithms suggested for data discretization. What follows is a sample of these algorithms:

- *Boolean reasoning algorithm*: Its based on combining the cuts found by the Naive algorithm (discussed below) with a Boolean reasoning procedure for discarding all but a small subset of cuts. The remaining subset is a minimal set of cuts that preserves the discernibility inherent in the decision system. Sometimes, the Boolean reasoning approach to discretization may result in no cuts being deemed necessary for some attributes. This means that these attributes are not really needed to preserve the discernibility, if the minimal set of cuts is employed. Rather than simply setting all values for these attributes to, e.g., 0, this algorithm leaves them untouched. The decision on how to deal with these attributes is left to the user. A common fallback is to revert to another discretization scheme for these undiscretized attributes.
- *Manual discretization*: Enables the user to manually specify cuts to be used for discretizing a given attribute.
- *Entropy/MDL algorithm*: Implements the algorithm based on recursively partitioning the value set of each attribute so that a local measure of entropy is optimized. The minimum description length principle defines a stopping criterion for the partitioning process. Missing values are ignored in the search for cuts. If no cuts are found for an attribute, the attribute is left unprocessed.
- *Equal frequency binning*: Implements equal frequency binning, a simple unsupervised and univariate discretization algorithm. Fixing a number of intervals n and examining the histogram of each attribute, n - 1 cuts are determined so that approximately the same number of objects fall into each of the n intervals. This corresponds to assigning n - 1 cut such that the area between two neighboring cuts in the normalized histogram is as close to $1/n$ as possible.
- *Naive algorithm*: Implements a very straightforward and simple heuristic that may result in very many cuts, probably far more than are desired. In the worst case, each observed value is assigned its own interval.
- *Semi-naive algorithm*: Functionally similar to the naive algorithm from above, but has more logic to handle the case where value-neighboring objects belong to different decision classes. Typically results in fewer

cuts than the simpler naive algorithm, but may still produce far more cuts than are desired.

- *From file with cuts*: Given a file with cuts discretizes a table according to the file's contents. The format of the file is a set of tab-delimited (a, c) pairs, one per line. The attribute index a is an integer relative to a masked table, while the cut c is a value relative to the internal integer table representation. Attributes with no cuts are left unprocessed.
- *Enhanced boolean reasoning algorithm*: Functionally similar to the algorithm described in "Boolean Reasoning Algorithm" above, but much faster. Approximate solutions are not supported. If $a(x)$ is missing, object x is not excluded from consideration when processing attribute a, but is instead treated as an "infinitely large" positive value. If no cuts are found for an attribute, all entries for that attribute are set to 0.
- *From file with cuts*: Functionally similar to the algorithm described in From file with cuts above but handles attributes with no cuts in a different manner. All entries for such attributes are set to 0, instead of being left unprocessed.

4. Reduct Generation

Reduct generation is where the approximations of sets are constructed. As mentioned before a rough set can be represented by a pair of crisp sets, called the lower and the upper approximation. The lower approximation consists of all objects, which certainly belong to the set and the upper approximation contains objects, which possibly belong to the set.

An important issue in rough set analysis is that of attribute reduction, which is performed in such a way that the reduced set of attributes provides the same quality of classification as the complete set. Input to a Reducer algorithm is a decision table, and a set of reducts is returned. The returned reduct set may possibly have a set of rules attached to it as a child. Two main types of discernibility methods are:

- *Full*: Computes reducts relative to the system as a whole, i.e., minimal attribute subsets that preserve our ability to discern all relevant objects from each other.
- *Object-based*: Computes reducts relative to a fixed object, i.e., minimal attribute subsets that preserve our ability to discern that object from the other relevant objects.

Modulo Decision: A table can either be interpreted as a decision system or as a general Pawlak information System. If the option to compute

reducts modulo the decision attribute is checked, the table is interpreted as a decision system. If the decision system contains inconsistencies, boundary region thinning should be considered. For consistent systems, there are no boundary regions to thin. In a decision system it may happen that two objects that are indiscernible with respect to attributes A may belong to different decision classes. The decision system is said to be inconsistent with respect to A if this is the case, and consistent otherwise. With boundary region thinning we look at the distribution of decision values within each indiscernibility set, and exclude those decision values from the generalized decision that occur with a frequency below some threshold. Low-probability decision values are thus treated as "noise".

Use of IDG: If the algorithm supports it, a set of IDGs can be supplied that enables the notion of discernibility to be overloaded on a per attribute basis. If no IDG file is specified, strict inequality is used.

- *Genetic algorithm*: Implements a genetic algorithm for computing minimal hitting sets. The algorithm has support for both cost information and approximate solutions. Each reduct in the returned reduct set has a support count associated with it. The support count is a measure of the "strength" of the reduct.
- *Johnson's algorithm*: Invokes a variation of a simple greedy algorithm to compute a single reduct only. The algorithm has a natural bias towards finding a single prime implicant of minimal length. For example Let S = {{cat, dog, fish}, {cat, man}, {dog, man}, {cat, fish}} and for simplicity let w be the constant function that assigns 1 to all sets S in S. Step 2 in the algorithm then amounts to selecting the attribute that occurs in the most sets in S. Initially, B = {empty}. Since cat is the most frequently occurring attribute in S, we update B to include cat. We then remove all sets from S that contain cat, and obtain S = {{dog, man}}. Repeating the process, we arrive at a tie in the occurrence counts of dog and man, and arbitrarily select dog. We add dog to B, and remove all sets from S that contain dog. Now, S = 0, so we're done. Our computed answer is thus B = {cat, dog}.
- *Holte's 1R*: Returns all singleton attribute sets. The set of all 1R rules, i.e., univariate decision rules, are indirectly returned as a child of the returned set of singleton reducts.
- *Manual reducer*: Enables the user to manually specify an attribute subset that can be used as a reduct in subsequent computations.
- *Dynamic reducts (RSES)*: A number of subtables are randomly sampled from the input table, and proper reducts are computed from each of

these using some algorithm. The reducts that occur the most often across subtables are in some sense the most "stable".

- *Exhaustive calculation (RSES)*: Computes all reducts by brute force. No support is provided for IDGs, boundary region thinning or approximate solutions.This algorithm does not scale up well, and is only suitable for tables of moderate size. Computing all reducts is NP-hard.
- *Johnson's algorithm (RSES)*: Invokes the RSES implementation of the greedy algorithm of Johnson
- *Genetic algorithm (RSES)*: Implements a variation of the algorithm described by Wr ´oblewski. Uses a genetic algorithm to search for reducts, either until the search space is exhausted or until a given maximum number of reducts have been found. Three predefined parameter settings can be chosen among that control the thoroughness and speed of the genetic search procedure. No support is provided for IDGs, boundary region thinning or approximate solutions.

Filter Reducts: The next step is to use a set of algorithms that remove elements from reduct sets, according to different evaluation criteria. Unless explicitly stated, algorithms in this family modify their input directly. Following are exemplary algorithms used for this purposes:

- *Basic filtering*: Removes individual reducts from a reduct set. Possible removal criteria can be the length of reduct or the support for that reduct.
- *Cost filtering*: Removes reducts from a reduct set according to their "cost". The function cost specifies the cost of attribute subset B. If a reduct's cost exceeds some specified threshold, the reduct is scheduled for removal.
- *Performance filtering*: Each reduct in the reduct set is evaluated according to the classificatory performance of the rules generated from that reduct alone. The reduct is removed if the performance score does not exceed a specified threshold.

The following process is performed for each reduct.

5. Rule Generation and Rule Filtering

This is where the rules are derived from the generated of the reducts. After you have produced the reducts and filtered them, you need to generate the rules. Conceptually, this is done by overlaying each reduct in the reduct set over the reduct set's parent decision table, and reading off the values. If the

reduct set already has a rule set attached to it, this child rule set is returned. If the reducts are of type "full discernibility", a new set of rules is generated.

Once generated Rules also need to be filtered. An exemplary set of algorithms for doing so are:

1. *Basic filtering*: Removes individual rules or patterns from a rule set. Possible removal criteria can be RHS Support, RHS Accuracy, RHS Coverage, RHS Stability, Decision, LHS Length etc.
2. *Quality filtering*: Filters away rules according to various measures of rule quality.
3. *Quality filtering loop*: Couples the filtering scheme from above together with ROC analysis. Enables the classificatory performance of a set of rules to be monitored as a function of the quality threshold.

6. Apply the Discretization Cuts to Test Dataset

The discretization schema developed using the training data set is not applied to the test dataset in order to develop a decision table that has nothing but discrete values variables.

7. Score the Test Dataset on Generated Rule set (and measuring the prediction accuracy)

This is where the discretized test dataset is scored using the generated rule set. Since there may be a large number of rules classifying a single object differently, there are a number of proposed alternative algorithms to resolve such classification phenomenon. These algorithms often implement some sort of voting mechanism on the outputs of a specified rule set. Specifically, in the firing step, a rule fires if its antecedent is not in conflict with the presented object, and if the percentage of verifiable terms in the antecedent is above a certain threshold value.

In some cases, and especially if the rules are generated as a result of dynamic reduct computation across sub tables of varying sizes, it may happen that some rules are generalizations of other rules. If two or more of the rules in the rule set form a generalization hierarchy, the algorithm has an option to only let the most specific rule fire, i.e., exclude the generalizations from the set of firing rules.

In the election process among the firing rules, each rule gets to cast a certain number of votes in favor of the decision value it indicates, according to a selected voting strategy. The certainty coefficient for each possible decision value is computed by dividing the total number of votes for each

decision value by a normalization factor. Here are some of the alternative voting methods:

- *Standard voting*: *Voting* is an ad hoc technique for rule-based classification that works reasonably well in practice. In the following, the rule set will be assumed to be *unordered*, i.e., all rules in the rule set will participate in the classification process. (more on voting can be found in thesis). Let RUL denote an unordered set of decision rules. The process of voting is one way of employing RUL to assign a numerical certainty factor to each decision class for each object. Typically, but not necessarily, an object is classified to the decision class for which the certainty factor is maximized. Presented with a given object x to classify, the voting process goes as follows:

  1. The set RUL is scanned for rules that fire, i.e., rules that have an antecedent that matches x. Let RUL(x) *of* RUL denote the set of rules that fire for object x.
  2. If RUL(x) = *0* then no classification can be made. Typically, a predetermined fallback classification is then invoked. Alternatives include reverting to a nearest neighbor method and instead consider a collection of rules "close" to object x in some sense
  3. An election process among the rules in RUL(x) is performed in order to resolve conflicts and rank the decisions. The election process is performed as follows:
     (a) Let each rule *r in* RUL(x) cast a number in votes votes(r) in favor of the decision class the rule indicates. The number of votes a rule gets to cast may vary. Typically, votes(r) is based on the support of the rule, but more complex quality-based measures are also possible.
     (b) Compute a normalization factor norm(x). The normalization factor can be computed in different ways. Typically, norm(x) is simply the sum of all cast votes, and only serves as a scaling factor.
     (c) Divide the accumulated number of votes for each possible decision class β by the normalization factor norm(x) in order to arrive at a certainty coefficient certainty(x, β) for each decision class.

- *Voting with object tracking*: Even though standard voting is simple and often works adequately, an objection one might make is that the procedure does not take into account which objects the rules in RUL(x) were originally derived from. It may very well be the case that the sets of objects that two or more firing rules were originally derived from partially

overlap. Hence, the objects that are members of such intersections are given a possibly unfair amount of weight in the voting process, since they effectively participate in the voting process through more than one rule.

Let $\alpha \rightarrow \beta$ denote a rule in RUL(x) and let $\alpha \rightarrow \beta$ have been induced from a decision system A. We define the set of *tracked objects* for the rule as the set of objects that are included in the support basis of the rule's antecedent. The union of all tracked objects for all firing rules then constitutes a natural basis for assigning a "fair" certainty factor to x.

Object tracking can also be viewed as a kind of voting. The participants in the election process are then not the firing rules themselves, but rather the union of all the tracked objects defined through the firing rules. Each tracked object in this union gets to cast one vote, and the results are then normalized to sum to unity.

- *Naive bayes*: For each decision class, computes the conditional probability that decision class is the correct one, given an object's information vector. The algorithm assumes that the object's attributes are independent. The probabilities involved in producing the final estimate are computed as frequency counts from a "master" decision table. The naive Bayes classifier often works very well in practice, and excellent classification results may be obtained even when the probability estimates contain large errors.
- *Standard/tuned voting (RSES)*: Offers rule-based classification based on voting, similar in spirit to the algorithm described above as Standard Voting

  Two main options are implemented:
  - Majority: Similar to the algorithm described above as standard voting with support-based voting, but with no tolerance for missing values. If any rules fire, the decision class that achieves the highest certainty factor is returned.
  - Tuned: Allows the voting to incorporate user-defined "distance" values between decision classes. If x denotes the object to classify and there are different groups of decision rules R_i that recognize x and indicate decision value i, x will be assigned decision value k, where k satisfies some condition.

8. Deploying the Rules in a Production System

Now that the rules are tested and confirmed to have value, they can easily be made a part of the decision making system by simply encoding them as series of conditional statement sequenced in a logical manner in the language being used.

A Representative Example[17]

Predicting the outcome of a ballot to legalize gaming in a state or county is an interesting and challenging problem. From the analytics standpoint, given that data from the past experiences are available, one might approach to solve this problem using knowledge discovery tools and techniques that include machine learning methods such as artificial neural networks, decision trees and rough sets. This study compares three popular machine-learning methods (i.e., artificial neural networks, decision trees and rough sets) based on their prediction accuracy of the gaming ballot outcomes. It uses a stratified *ten*-fold cross validation method in order to minimize the bias coming from the partitioning of the data into training and validation sets, and hence producing an unbiased comparison of the three methods.

Variables used in this study stem from a recently developed model by Sirakaya et al. (2005). From a through literature review, the authors developed a conceptual model by synthesizing a series of determinants affecting voting behavior. According to the model, voting behavior is a function of a person's political socialization, exposure and evaluation of competing issues and ideas, moderating impact of time on voters' socio-demographic and political background, and ad hoc determinants stemming from in gaming research such as expectations of economic, social or overall personal benefits from gambling, perceived changes in crowding and congestion, safety and security.

Both primary and secondary data collection techniques were used to collect data for this study. An e-mail survey was sent to all 50 state and county-election offices that solicited information about the results of gambling ballot outcomes ('yea' and 'nay' votes for propositions). All contacted officials responded by submitting historical or current data regarding all gaming and wagering related ballot outcomes (e.g., horse race ballots, grey-hound race ballots, lotteries, casino gambling etc.) in the history of their states and counties. In cases where gaming was never voted on or not currently on the ballots, the election offices a issued a statement by regarding that fact in their response. If the ballot was a state-wise issue, we included all county results; if the ballots were put on specific counties, we included only those counties in our data set. Thus the data set contains both statewide- as well as county-data. Gambling ballot are not put on ballot every election in every community; therefore, creating a data set with sufficient observations on all variables requires the use of cross-sectional data. For this analysis, only the results of gambling related ballots proposed

[17] This case is based on E. Sirakaya, D. Delen, H. Choi (2005). Forecasting Gaming Referenda, *Annals of Tourism Research* 32:1, 127–149.

during 1990s were used (14 states with 24 ballot results: California, Colorado, Florida, Georgia, Louisiana, Nebraska, New Jersey, North Dakota, Oklahoma, Ohio, South Dakota, Texas, Utah, and Washington) because of the restriction placed on the availability of the nationwide religious-affiliation data. In other words, religious affiliation data put a restriction on how we could use the rest of the variables at hand. To construct the final cross-sectional data containing all relevant variables, a variety of sources were consulted: American Religious Archive for county level religious data (Sirakaya et al. 2005), the U.S. Census Bureau for socio-demographic data, Bureau of Economic Analysis (BEA) (1999) for per capita income data, Bureau of Labor Statistics (BLS) (2000) for employment data. To obtain an estimate of population density (population per square mile), the size of a county was divided by its population. As the reader will notice, the variables are chosen based on the review of literature and the availability of national data on this issue. There is no one complete data set regarding this issue. Although, the literature points out to many other variables as explained in the previous review of variables and models, the large number of studies showed that the variables included in this study do explain much of the error variation in predictive models. The final variable list used in this study is depicted in Table 6.1. The final data set contained 1287 records (each of which was representing the results of a gaming referenda) of which 589 records were ballot type I (casino style gambling) and remaining 698 records were ballot Type II (wagering or non-casino style gaming). Although, the rate of change in demographic data seemed to be stable over a period of a decade, when interpreting the findings, the limitations placed on this cross-sectional data must be considered.

Table 6.1. Variables used in the study

1.	Ballot Type I (casino style gambling)
2.	Ballot Type II (wagering or non-casino style gaming)
3.	Percent population voted
4.	Medium family income
5.	Percent population church members
6.	Percent population male
7.	Poverty level
8.	Unemployment rate
9.	Percent population minority (non-white)
10.	Percent population older than 45
11.	Metropolitan statistical area (MSA) – Unary encoding (UE)

Note: Dependent variable is the ballot outcome (‘nay’ or ‘yea’).
The data set includes 1287 records from 1287 counties.

The complete sets of empirical results are presented in Tables 6.2, 6.3 and 6.4. For each method (i.e., artificial neural networks, decision trees and rough sets) and ballot type (i.e., ballot type 1 and ballot type 2) the results are presented in detail for all ten folds. Each fold can be associated with an instance of a prediction model. For each fold, the results obtained from the holdout sample are presented separately for "Yea" votes and "Nay" votes. This way one might perceive what the Type 1-casino gambling- (false positives) and Type 2-wagering-(false negative) error rates are for those experiments (readers will notice that Type 1 and 2 refer to the nature of the political proposition and not hypothesis testing. The aggregated results of all ten folds for each model and ballot type combination are presented in the form of mean, standard deviation, minimum and maximum.

Table 6.2. Decision tree cross-validation prediction results

Ballot Type 1				Ballot Type 2			
	Accuracy (%)				Accuracy (%)		
Fold No.	Yes	No	Overall	Fold No.	Yes	No	Overall
1	75.00	74.07	74.58	1	89.29	61.90	72.86
2	76.92	75.76	76.27	2	62.96	81.40	74.29
3	88.24	76.00	83.05	3	71.43	71.43	71.43
4	85.29	72.00	79.66	4	85.29	66.67	75.71
5	80.65	67.86	74.58	5	75.76	72.97	74.29
6	93.55	64.29	79.66	6	65.52	85.37	77.14
7	84.38	74.07	79.66	7	73.33	82.50	78.57
8	87.50	60.71	75.00	8	70.37	79.07	75.71
9	79.31	73.33	76.27	9	69.70	78.38	74.29
10	87.88	73.08	81.36	10	83.33	73.53	78.57
Mean	83.87	71.12	78.01	Mean	74.70	75.32	75.29
St. Dev.	5.50	4.88	2.89	St. Dev.	8.24	7.01	2.22
Min	75.00	60.71	74.58	Min	62.96	61.90	71.43
Max	93.55	76.00	83.05	Max	89.29	85.37	78.57

Table 6.3. ANN Cross-Validation prediction results

Ballot Type 1				Ballot Type 2			
	Accuracy (%)				Accuracy (%)		
Fold No.	Yes	No	Overall	Fold No.	Yes	No	Overall
1	76.31	91.06	83.87	1	86.06	63.41	78.08
2	73.52	93.05	83.53	2	90.49	63.01	80.80
3	73.52	91.72	82.85	3	86.06	67.89	79.66
4	72.82	91.06	82.17	4	85.62	70.73	80.37
5	80.14	81.79	80.98	5	84.96	65.85	78.22
6	77.35	81.79	79.63	6	78.32	77.24	77.94

(Continued)

Table 6.3. (Continued)

Ballot Type 1				Ballot Type 2			
	Accuracy (%)				Accuracy (%)		
Fold No.	Yes	No	Overall	Fold No.	Yes	No	Overall
7	70.38	85.10	77.93	7	86.73	63.41	78.51
8	68.64	86.75	77.93	8	88.94	59.35	78.51
9	77.35	66.23	71.65	9	84.07	67.48	78.22
10	77.35	81.79	79.63	10	83.12	66.26	77.26
Mean	74.74	85.03	80.02	Mean	85.44	66.46	78.76
St. Dev.	3.39	7.55	3.46	St. Dev.	3.14	4.68	1.08
Min	68.64	66.23	71.65	Min	78.32	59.35	77.26
Max	80.14	93.05	83.87	Max	90.49	77.24	80.80

In Table 6.4, the left half of the table lists the prediction results of decision tree models for ballot Type 1 whereas the right side of the table lists the prediction results of decision tree models for ballot Type 2. The first column for each of the two ballot types identifies the fold number, the second column shows the percent classification accuracy for "Yea" votes, the third column shows the percent classification accuracy for "Nay" votes and the fourth column shows the percent classification accuracy for all records. Let TP (true positive) be the number of

Table 6.4. Rough sets cross-Validation prediction results

Ballot Type 1				Ballot Type 1			
	Accuracy (%)				Accuracy (%)		
Fold No.	Yes	No	Overall	Fold No.	Yes	No	Overall
1	83.33	90.91	86.21	1	90.48	96.15	92.65
2	88.46	81.82	84.75	2	85.71	100.00	82.86
3	70.97	85.19	76.27	3	82.22	78.95	74.29
4	87.18	95.00	89.83	4	80.00	94.12	80.00
5	100.00	80.00	86.44	5	91.67	80.95	87.14
6	96.43	77.42	86.44	6	85.11	76.19	80.00
7	86.96	86.11	86.44	7	77.08	90.91	81.43
8	95.83	77.14	84.75	8	88.00	75.00	84.29
9	81.82	91.89	88.14	9	86.54	92.86	82.86
10	70.97	82.14	76.27	10	93.75	78.95	85.71
Mean	86.19	84.76	84.55	Mean	86.06	86.41	83.12
St. Dev.	9.41	5.88	4.37	St. Dev.	4.97	8.81	4.64
Min	70.97	77.14	76.27	Min	77.08	75.00	74.29
Max	100.00	95.00	89.83	Max	93.75	100.00	92.65

"Yes" records classified as "Yes", TN (true negative) be the number of "No" records classified as "No", FP (false positive – Type 1 error) be the number of "No" records classified as "Yes", and FN (false negative – Type 2 error) be the number of "Yes" records classified as "No". Accordingly the second, third and fourth columns can algebraically be written as Eq. (6.1) through (6.3) respectively.

$$\text{Percent_Yes_Accuracy} = \frac{TP}{TP + FN} \tag{6.1}$$

$$\text{Percent_No_Accuracy} = \frac{TN}{TN + FP} \tag{6.2}$$

$$\text{Percent_Overall_Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.3}$$

After the list of accuracy measures for each of the ten folds, also are given the descriptive statistics (mean, standard deviation, minimum and maximum) for those three columns. These calculations were applied exactly similar to both ballot types and to all three prediction model types. There is, however, a slight difference between rough set analysis and the other two model types in the way they calculate the percent prediction accuracies. While classifying the records in the holdout sample, artificial neural network models and decision tree models forces every record to go into either "Yes" prediction category or into "No" prediction category, correctly or incorrectly. In contrast, rough sets model has another category called "undecided" which is used to place a record during the model evaluation. In other words, if the rules generated by rough sets do not favor (i.e., do not have enough evidence to) either "Yes" or "No" for a specific record, the record will be classified as "undecided". This requires the Eqs. (6.1) through (6.3) to be modified in order to take into account this discrepancy. Let UP (undecided positive) be the number "Yes" records that did not get to be classified and UN (undecided negative) be the number "Yes" records that did not get to be classified as either "Yes" or "No" by a rough set analysis model (see Eqs. (6.4) through (6.6)).

$$\text{Percent_Yes_Accuracy}_{\text{RoughSets}} = \frac{TP}{TP + FN + UDP} \tag{6.4}$$

$$\text{Percent_No_Accuracy}_{\text{RoughSets}} = \frac{TN}{TN + FP + UDN} \tag{6.5}$$

$$\text{Percent_Overall_Accuracy}_{\text{RoughSets}} = \frac{TP + TN}{TP + TN + FP + FN + UDP + UDN} \tag{6.6}$$

This small difference which potentially works against rough sets analysis' prediction accuracy actually makes a stronger case for rough sets being more model/knowledge driven and less probabilistic, therefore more reliable and understandable.

According to a nominal comparison of performance differences among the three methods show that for ballot Type 1 (casino gambling), rough sets prediction results are better than decision tress and artificial neural networks. The performances of both decision trees and artificial neural networks seem to be similar for ballot Type 1. However, for the ballot Type 2 (wagering), artificial neural networks seem to outperform decision trees. Rough sets, once again for the ballot Type 2, outperformed both of the other two models (versus decision trees, and artificial neural networks). Based on these results, we can conclude that for the dataset used in this study, rough sets seem to be the best analysis technique predicting cross-validation sets significantly better than the other two. The artificial neural networks are the second best performing better than decision trees for the ballot Type 2; the decision trees performed least in predicting ballot outcomes.

This study showed that advanced data mining methods could successfully be used to develop models for predicting the ballot outcomes for legalizing gaming with a relatively high degree of accuracy. However, in the context of predictive modeling, one should be aware of several issues that delimit the applicability and predictive accuracy of data mining models: the nature of (i) the data (including the richness, correctness, completeness and representation of the data itself), (ii) the data mining methods (including their capabilities and limitations to handle different types and combination of data as well as their ability to capture non-linear relationships between independent and dependent variables) and (iii) the application domain (including understandability and availability of related factors needed to develop accurate predictive models). The models are only as good and as predictive as the data used to build them. One of the key determinants of predictive accuracy is the amount and quality of the data used for a study. Although all of the models built in this study provided an acceptable level of prediction accuracy based on variables at hand, their performance levels could be improved by including more relevant variables. Currently availability of such secondary data puts limits on how far the prediction models can go. It is recommended that more (and relevant) variables be added to the model when possible to increase the accuracy levels. Another issue is the lack of such studies in the forecasting literature that would help establish a comparative base. One could only hope that the number of empirical data mining studies increases in tourism and travel research, leading to establishing a comparative base for the evaluation of efficacy of such models.

Conclusion

In evaluating the performance of data mining techniques, in addition to predictive accuracy, some researchers have emphasized the importance of the explanatory nature of the models and the need to reveal patterns that are valid, novel, useful and may be most importantly understandable and explainable (Matheus et al. 1993). Despite their modeling power and wide spread use in complex prediction modeling tasks, artificial neural networks have been criticized for their lack of explanatory power. In other words, it is difficult to trace and explain the way the reasoning is derived from the input variables due to the complexity and non-linear nature of data transformation conducted within the algorithm. Consequently, researchers in forecasting domain tend to call it a black-box approach to predictive modeling. In response to this criticism, the research community has developed ways to extract explicit knowledge from the neural networks (Sentiono and Liu 1996; Lu et al. 1996). Though somewhat successful in developing these mostly heuristic methods, the major criticism of neural networks lacking theoretical base still holds. In contrast to neural networks, decision trees and rough sets present their models in the form of business rules, which are intuitively explainable. They connect the input variables (factors) to output variables (conclusions) using IF <condition(s)> THEN <action(s)> structure.

The major criticism of rough sets is that it requires all of the variables to be in nominal format. Rough sets cannot work with numerical continuous valued variables. In order to use such variables, one should perform discretization. Discretization, a process of converting continuous numerical variables into range labels, can be done by using different methods ranging from simple sorting algorithms such as equal width, equal bin/frequency to highly sophisticated search algorithms such as genetic algorithms and simulated annealing. Regardless of the sophistication level employed, there is no single best method to discretize for any data set; making it an experiment based lengthy learning process. Some also may argue that converting continuous valued numerical values to a few range labels would lessen the information content of the variable. Compared to rough sets in this issue, neural networks and decision trees can process any combination of continuous and nominal values variables.

7 Support Vector Machines

Support vector machines (SVMs) are supervised learning methods that generate input-output mapping functions from a set of labeled training data. The mapping function can be either a classification function (used to categorize the input data) or a regression function (used to estimation of the desired output). For classification, nonlinear kernel functions are often used to transform the input data (inherently representing highly complex nonlinear relationships) to a high dimensional feature space in which the input data becomes more separable (i.e., linearly separable) compared to the original input space. Then, the maximum-margin hyperplanes are constructed to optimally separate the classes in the training data. Two parallel hyperplanes are constructed on each side of the hyperplane that separates the data by maximizing the distance between the two parallel hyperplanes. An assumption is made that the larger the margin or distance between these parallel hyperplanes the better the generalization error of the classifier will be.

SVMs belong to a family of generalized linear models which achieves a classification or regression decision based on the value of the linear combination of features. They are also said to belong to "kernel methods".

In addition to its solid mathematical foundation in statistical learning theory, SVMs have demonstrated highly competitive performance in numerous real-world applications, such as medical diagnosis, bioinformatics, face recognition, image processing and text mining, which has established SVMs as one of the most popular, state-of-the-art tools for knowledge discovery and data mining. Similar to artificial neural networks, SVMs possess the well-known ability of being universal approximators of any multivariate function to any desired degree of accuracy. Therefore, they are of particular interest to modeling highly nonlinear, complex systems and processes.

Generally, many linear classifiers (hyperplanes) are able to separate data into multiple classes. However, only one hyperplane achieves maximum separation. SVMs classify data as a part of a machine-learning process, which "learns" from the historic cases represented as data points. These data points may have more than two dimensions. Ultimately we are interested in whether we can separate data by an n–1 dimensional hyperplane.

This may be seen as a typical form of linear classifier. We are interested in finding if we can achieve maximum separation (margin) between the two (or more) classes. By this we mean that we pick the hyperplane so that the distance from the hyperplane to the nearest data point is maximized. Now, if such a hyperplane exists, the hyperplane is clearly of interest and is known as the maximum-margin hyperplane and such a linear classifier is known as a maximum margin classifier.

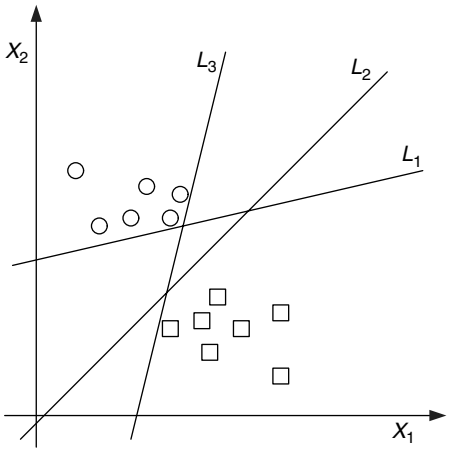


Fig. 7.1. Many linear classifiers (hyperplanes) may separate the data

Formal Explanation of SVM

Consider data points in the training dataset of the form:

$$\{(x_1, c_1), (x_2, c_2), \ldots, (x_n, c_n)\},$$

where the c_i is either 1 (i.e., "yes") or − 0 (i.e., "no") – this constant denotes the class to which the data point belongs. Each data point is an n dimensional real vector, usually of scaled [0, 1] or [–1, 1] values. The scaling is important to guard against variables (attributes) with larger variance that might otherwise dominate the classification. We can view this as training data, which denotes the correct classification which we would like the SVM to eventually distinguish, by means of the dividing hyperplane, which takes the form

$$w \bullet x - b = 0.$$

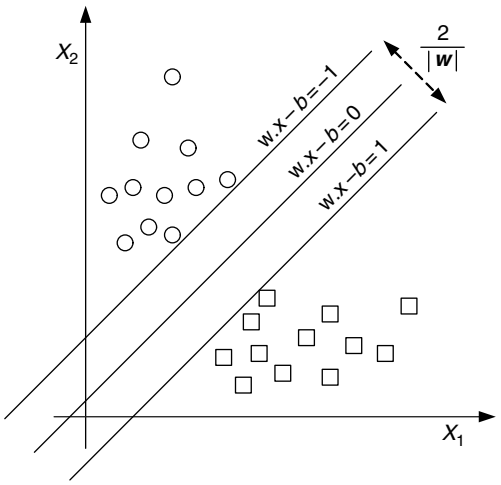


Fig. 7.2. Maximum separation hyperplanes

The vector $\boldsymbol{w}$ points perpendicular to the separating hyperplane. Adding the offset parameter b allows us to increase the margin. In its absence, the hyperplane is forced to pass through the origin, restricting the solution.

As we are interested in the maximum margin, we are interested in the support vectors and the parallel hyperplanes (to the optimal hyperplane) closest to these support vectors in either class. It can be shown that these parallel hyperplanes can be described by equations

$$w \bullet x - b = 1,$$

$$w \bullet x - b = -1.$$

If the training data are linearly separable, we can select these hyperplanes so that there are no points between them and then try to maximize their distance. By using geometry, we find the distance between the hyperplanes is $2/|\boldsymbol{w}|$, so we want to minimize $|\boldsymbol{w}|$. To exclude data points, we need to ensure that for all i either

$$w \bullet x_i - b \geq 1 \quad \text{or}$$
$$w \bullet x_i - b \leq -1.$$

This can be rewritten as:

$$c_i\left(w \bullet x_i - b\right) \geq 1, \quad 1 \leq i \leq n.$$

Primal Form

The problem now is to minimize $|\boldsymbol{w}|$ subject to the constraint (1). This is a quadratic programming (QP) optimization problem. More clearly,

$$\text{Minimize} \qquad (1/2)\|w\|^2$$

$$\text{Subject to} \qquad c_i(w \bullet x_i - b) \geq 1, \quad 1 \leq i \leq n.$$

The factor of 1/2 is used for mathematical convenience.

Dual Form

Writing the classification rule in its dual form reveals that classification is only a function of the support vectors, i.e., the training data that lie on the margin. The dual of the SVM can be shown to be:

$$\max \sum_{i=1}^{n} \alpha_i - \sum_{i,j} \alpha_i \alpha_j c_i c_j x_i^T x_j,$$

where the α terms constitute a dual representation for the weight vector in terms of the training set:

$$w = \sum_i \alpha_i c_i x_i.$$

Soft Margin

In 1995, Cortes and Vapnik suggested a modified maximum margin idea that allows for mislabeled examples.[1] If there exists no hyperplane that can split the "yes" and "no" examples, the Soft Margin method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. This work popularized the expression Support Vector Machine or SVM. The method introduces slack variables, ξ_i, which measure the degree of misclassification of the datum.

$$c_i(w \bullet x_i - b) \geq 1 - \xi_i \quad 1 \leq i \leq n.$$

The objective function is then increased by a function which penalizes non-zero ξ_i, and the optimization becomes a tradeoff between a large

[1] C. Cortes, V. Vapnik (1995). Support-vector networks, *Machine Learning*, 20.

margin, and a small error penalty. If the penalty function is linear, the primal form of the objective function becomes

$$\min\|w\|^2 + C\sum_i \xi_i \quad \text{such that } c_i(w \bullet x_i - b) \geq 1 - \xi_i \quad 1 \leq i \leq n.$$

This constraint in (2) along with the objective of minimizing |w| can be solved using Lagrange multipliers. The key advantage of a linear penalty function is that the slack variables vanish from the dual problem, with the constant C appearing only as an additional constraint on the Lagrange multipliers. Non-linear penalty functions have been used, particularly to reduce the effect of outliers on the classifier, but unless care is taken, the problem becomes non-convex, and thus it is considerably more difficult to find a global solution.

Non-linear Classification

The original optimal hyperplane algorithm proposed by Vladimir Vapnik in 1963, while he was a doctoral student at the Institute of Control Science in Moscow, was a linear classifier. However, in 1992, Bernhard Boser, Isabelle Guyon and Vapnik[2] suggested a way to create non-linear classifiers by applying the kernel trick (originally proposed by Aizerman et al.[3]) to maximum-margin hyperplanes. The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in the transformed feature space. The transformation may be non-linear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space it may be non-linear in the original input space.

If the kernel used is a Gaussian radial basis function, the corresponding feature space is a Hilbert space of infinite dimension. Maximum margin classifiers are well regularized, so the infinite dimension does not spoil the results. Some common kernels include,

---

[2] B.E. Boser, I.M. Guyon, V.N. Vapnik (1992). A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, 144–152, Pittsburgh, PA: ACM Press.

[3] M. Aizerman, E. Braverman, L. Rozonoer (1964). Theoretical foundations of the potential function method in pattern recognition learning, *Automation and Remote Control* 25, 821–837.

- Polynomial (homogeneous): $k(x, x') = (x \bullet x')^d$
- Polynomial (inhomogeneous): $k(x, x') = (x \bullet x' + 1)^d$
- Radial Basis Function: $k(x, x') = \exp\left(-\gamma \|x - x'\|^2\right)$, for $\gamma > 0$
- Gaussian Radial basis function: $k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$
- Sigmoid: $k(x, x') = \tanh(kx \bullet x' + c)$ for some $k > 0$ and $c < 0$

Regression

A version of a SVM for regression was proposed called support vector regression (SVR).[4] The model produced by support vector classification (as described above) only depends on a subset of the training data, because the cost function for building the model does not care about training points that lie beyond the margin. Analogously, the model produced by SVR only depends on a subset of the training data, because the cost function for building the model ignores any training data that are close (within a threshold ε) to the model prediction.

Implementation

The parameters of the maximum-margin hyperplane are derived by solving the optimization. There exist several specialized algorithms for quickly solving the QP problem that arises from SVMs, mostly reliant on heuristics for breaking the problem down into smaller, more-manageable chunks. A common method for solving the QP problem is Platt's SMO algorithm,[5] which breaks the problem down into two-dimensional sub-problems that may be solved analytically, eliminating the need for a numerical optimization algorithm such as conjugate gradient methods.

---

[4] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, V. Vapnik (1997). Support vector regression machines, *Advances in Neural Information Processing Systems* 9, NIPS, 155–161, Cambridge, MA: MIT Press.

[5] J. Platt (1998). Sequential minimal optimization: A fast algorithm for training support vector machines, *Microsoft Research Technical Report MSR-TR-98-14.*

Kernel Trick

In machine learning, the kernel trick is a method for converting a linear classifier algorithm into a non-linear one by using a non-linear function to map the original observations into a higher-dimensional space; this makes a linear classification in the new space equivalent to non-linear classification in the original space.

This is done using Mercer's theorem, which states that any continuous, symmetric, positive semi-definite kernel function $K(x, y)$ can be expressed as a dot product in a high-dimensional space.

More specifically, if the arguments to the kernel are in a measurable space X, and if the kernel is positive semi-definite – i.e.

$$\sum_{i,j} K\left(x_i, x_j\right) c_i c_j \geq 0,$$

for any finite subset $\{x_1, \ldots x_n\}$ of X and subset $\{c_1, \ldots, c_n\}$ of objects (typically real numbers or even molecules) – then there exists a function $\varphi(x)$ whose range is in an inner product space of possibly high dimension, such that

$$K\left(x, y\right) = \varphi\left(x\right) \bullet \varphi\left(y\right).$$

The kernel trick transforms any algorithm that solely depends on the dot product between two vectors. Wherever a dot product is used, it is replaced with the kernel function. Thus, a linear algorithm can easily be transformed into a non-linear algorithm. This non-linear algorithm is equivalent to the linear algorithm operating in the range space of φ. However, because kernels are used, the φ function is never explicitly computed. This is desirable, because the high-dimensional space may be infinite-dimensional (as is the case when the kernel is a Gaussian).

Although the origin of the term kernel trick is not known, the kernel trick was first published by Aizerman et al.[6] It has been applied to several kinds of algorithm in machine learning and statistics, including:

- Perceptrons
- Support vector machines
- Principal components analysis
- Fisher's linear discriminant analysis
- Clustering

[6] M. Aizerman et al. (1964), op cit.

Use of SVM – A Process-Based Approach

Due largely to the better classification results, recently support vector machines (SVMs) have become a popular technique for classification type problems. Even though people consider them as easier to use than artificial neural networks, users who are not familiar with the intricacies of SVMs often get unsatisfactory results. In this section we provide a process-based approach to the use of SVM which is more likely to produce better results.

- Preprocess the data
  - Scrub the data
    - Deal with the missing values
    - Deal with the presumably incorrect values
    - Deal with the noise in the data
  - Transform the data
    - Numerisize the data
    - Normalize the data
- Develop the model(s)
  - Select kernel type (RBF is a natural choice)
  - Determine kernel parameters based on the selected kernel type (e.g., C and γ for RBF) – A hard problem. One should consider using cross-validation and experimentation to determine the appropriate values for these parameters.
  - If the results are satisfactory, finalize the model, otherwise change the kernel type and/or kernel parameters to achieve the desired accuracy level.
- Extract and deploy the model.

For clarity purposes, a pictorial representation of the process model is depicted in Fig. 7.3. A short description for some of the important steps in the above listed process is given below.

Numerisizing the Data: SVMs require that each data instance is represented as a vector of real numbers. Hence, if there are categorical attributes, we first have to convert them into numeric data. We recommend using m numbers to represent an m-category attribute. Only one of the m numbers is one, and others are zero. For example, a three-category attribute such as {red, green, blue} can be represented as (0,0,1), (0,1,0), and (1,0,0).

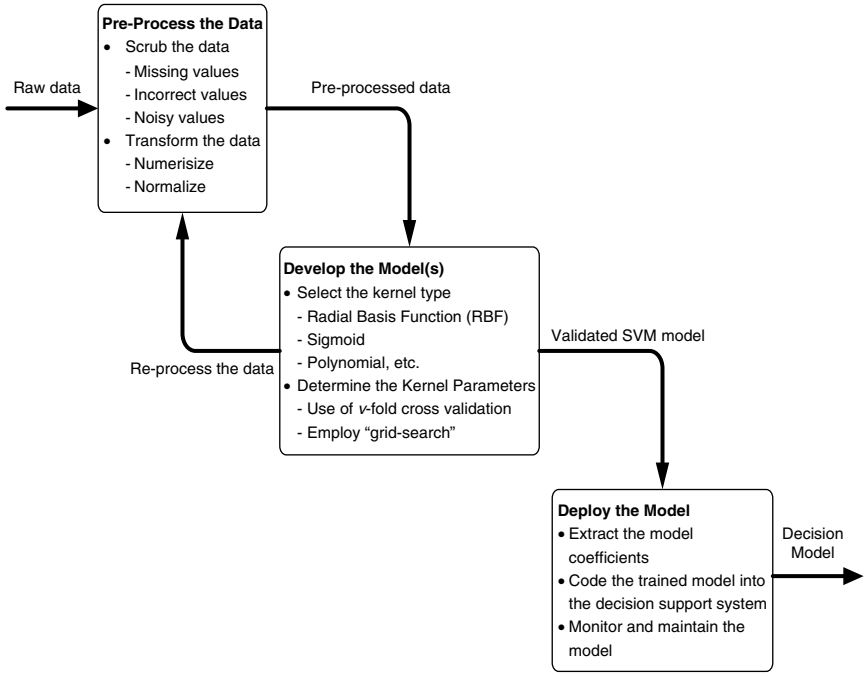


Fig. 7.3. A process description for SVM model development

Normalizing the Data (as is the case for ANN): Scaling them before applying SVM is very important. Sarle explained why we scale data while using Neural Networks, and most of considerations also apply to SVM.[7] The main advantage is to avoid attributes in greater numeric ranges dominate those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems. We recommend linearly scaling each attribute to the range [−1, +1] or [0, 1]. Of course we have to use the same method to scale testing data before testing. For example, suppose that we scaled the first attribute of training data from [−10, +10] to [−1, +1]. If the first attribute of testing data is lying in the range [−11, +8], we must scale the testing data to [−1.1, +0.8].

[7] W.S. Sarle, editor (1997), *Neural Network FAQ*, part 1 of 7: Introduction, periodic posting to the Usenet newsgroup comp.ai.neural-nets, URL: ftp://ftp.sas.com/pub/neural/FAQ.html.

Select the Kernel Model: Though there are only four common kernels, we must decide which one to try first. Then the penalty parameter C and kernel parameters are chosen. We suggest that in general RBF is a reasonable first choice. The RBF kernel nonlinearly maps samples into a higher dimensional space, so it, unlike the linear kernel, can handle the case when the relation between class labels and attributes is nonlinear. Furthermore, the linear kernel is a special case of RBF. Keerthi and Lin show that the linear kernel with a penalty parameter C has the same performance as the RBF kernel with some parameters (C, γ).[8] In addition, the sigmoid kernel behaves like RBF for certain parameters.[9] The second reason is the number of hyperparameters which influences the complexity of model selection. The polynomial kernel has more hyperparameters than the RBF kernel. Finally, the RBF kernel has less numerical difficulties. Cross Validation and Grid Search: Cross-validation and Grid-search. There are two parameters while using RBF kernels: C and γ. It is not known beforehand which C and γ are the best for one problem; consequently some kind of model selection (parameter search) must be done. The goal is to identify good (C, γ) so that the classifier can accurately predict unknown data (i.e., testing data). Note that it may not be useful to achieve high training accuracy (i.e., classifiers accurately predict training data whose class labels are indeed known). Therefore, a common way is to separate training data to two parts of which one is considered unknown in training the classifier. Then the prediction accuracy on this set can more precisely reflect the performance on classifying unknown data. An improved version of this procedure is cross-validation. In v-fold cross-validation, we first divide the training set into v subsets of equal size. Sequentially one subset is tested using the classifier trained on the remaining $v-1$ subsets. Thus, each instance of the whole training set is predicted once so the cross-validation accuracy is the percentage of data which are correctly classified. The cross-validation procedure can prevent the overfitting problem. Another recommendation for dealing with the overfitting problem is a "grid-search" on C and γ using cross-validation. Basically pairs of (C, γ) are tried and the one with the best cross-validation accuracy is picked. We found that trying exponentially growing sequences of C and γ is a practical method to identify good parameters (for example, $C = 2^{-5}, 2^{-3}, \ldots, 215, \gamma = 2^{-15}, 2^{-13}, \ldots, 23$). The grid-search is straightforward but seems stupid. In fact, there are several advanced methods which can save computational cost by, for example,

[8] S.S. Keerthi, C.-J. Lin (2003). Asymptotic behaviors of support vector machines with Gaussian kernel, *Neural Computation* 15:7, 1667–1689.

[9] K.-M. Lin, C.-J. Lin (2003). A study on reduced support vector machines, *IEEE Transactions on Neural Networks* 14:6, 1449–1559.

approximating the cross-validation rate. However, there are two motivations why we prefer the simple grid-search approach.

In some situations, the proposed procedure is not good enough, so other techniques such as feature selection may be needed. Such issues are beyond our consideration here. Our experience indicates that the procedure works well for data which do not have many features. If there are thousands of attributes, there may be a need to choose a subset of them before giving the data to SVM.

Support Vector Machines versus Artificial Neural Networks

The development of ANNs followed a heuristic path, with applications and extensive experimentation preceding theory. In contrast, the development of SVMs involved sound theory first, then implementation and experiments. A significant advantage of SVMs is that while ANNs can suffer from multiple local minima, the solution to an SVM is global and unique. Two more advantages of SVMs are that that have a simple geometric interpretation and give a sparse solution. Unlike ANNs, the computational complexity of SVMs does not depend on the dimensionality of the input space. ANNs use empirical risk minimization, whilst SVMs use structural risk minimization. The reason that SVMs often outperform ANNs in practice is that they deal with the biggest problem with ANNs, SVMs are less prone to over fitting.

- They differ radically from comparable approaches such as neural networks: SVM training always finds a global minimum, and their simple geometric interpretation provides fertile ground for further investigation.
- Most often Gaussian kernels are used, when the resulted SVM corresponds to an RBF network with Gaussian radial basis functions. As the SVM approach “automatically” solves the network complexity problem, the size of the hidden layer is obtained as the result of the QP procedure. Hidden neurons and support vectors correspond to each other, so the center problems of the RBF network is also solved, as the support vectors serve as the basis function centers.
- In problems when linear decision hyperplanes are no longer feasible, an input space is mapped into a feature space (the hidden layer in NN models), resulting in a nonlinear classifier.

- SVMs, after the learning stage, create the same type of decision hypersurfaces as do some well-developed and popular NN classifiers. Note that the training of these diverse models is different. However, after the successful learning stage, the resulting decision surfaces are identical.
- Unlike conventional statistical and neural network methods, the SVM approach does not attempt to control model complexity by keeping the number of features small.
- Classical learning systems like neural networks suffer from their theoretical weakness, e.g. back-propagation usually converges only to locally optimal solutions. Here SVMs can provide a significant improvement.
- In contrast to neural networks SVMs automatically select their model size (by selecting the Support vectors).
- The absence of local minima from the above algorithms marks a major departure from traditional systems such as neural networks.
- While the weight decay term is an important aspect for obtaining good generalization in the context of neural networks for regression, the margin plays a somewhat similar role in classification problems.
- In comparison with traditional multilayer perceptron neural networks that suffer from the existence of multiple local minima solutions, convexity is an important and interesting property of nonlinear SVM classifiers.
- SVMs have been developed in the reverse order to the development of neural networks (NNs). SVMs evolved from the sound theory to the implementation and experiments, while the NNs followed more heuristic path, from applications and extensive experimentation to the theory.

Disadvantages of Support Vector Machines

- Besides the advantages of SVMs (from a practical point of view) they have some limitation. An important practical question that is not entirely solved, is the selection of the kernel function parameters – for Gaussian kernels the width parameter (Σ) – and the value of (ε) in the (ε)-insensitive loss function.
- A second limitation is the speed and size, both in training and testing. It involves complex and time demanding calculations. From a practical point of view perhaps the most serious problem with SVMs is the high algorithmic complexity and extensive memory requirements of the

required quadratic programming in large-scale tasks. Shi et al. have conducted comparative testing of SVM with other algorithms on real credit card data.[10]

- Processing of discrete data presents another problem.

Despite these limitations, because SVMs are based on sound theoretical foundation and the solution it produces are global and unique in nature (as opposed to getting stuck in local minima), nowadays they are the most popular prediction modeling techniques in the data mining arena. Their use and popularity will only increase as the popular commercial data mining tools start to incorporate them into their modeling arsenal.

[10] Y. Shi, Y. Peng, G. Kou, Z. Chen (2005). Classifying credit card accounts for business intelligence and decision making: A multiple-criteria quadratic programming approach, *International Journal of Information Technology & Decision Making* 4:4, 581–599.

8 Genetic Algorithm Support to Data Mining

Genetic algorithms are mathematical procedures utilizing the process of genetic inheritance. They have been usefully applied to a wide variety of analytic problems. Data mining can combine human understanding with automatic analysis of data to detect patterns or key relationships. Given a large database defined over a number of variables, the goal is to efficiently find the most interesting patterns in the database.[1] Genetic algorithms have been applied to identify interesting patterns in some applications.[2] They usually are used in data mining to improve the performance of other algorithms, one example being decision tree algorithms,[3] another association rules.[4]

Genetic algorithms require certain data structure. They operate on a population with characteristics expressed in categorical form. The analogy with genetics is that the population (genes) consist of characteristics (alleles).[5] One way to implement genetic algorithms is to apply operators (reproduction, crossover, selection) with the feature of mutation to enhance generation of potentially better combinations. The genetic algorithm process is thus:

---

[1] H.K. Bhargava (1999). Data mining by decomposition: Adaptive search for hypothesis generation, *INFORMS Journal on Computing* 11:3, 239–247.

[2] A. Kamrani, W. Rong, R. Gonzalez (2001). A genetic algorithm methodology for data mining and intelligent knowledge acquisition, *Computers & Industrial Engineering* 40:4, 361–377.

[3] K. Sorensen, G.K. Janssens (2003). Data mining with genetic algorithms on binary trees, *European Journal of Operational Research* 151, 253–264; Z. Fu, G.L. Golden, S. Lele, S. Raghavan, E. Wasil (2006). Diversification for better classification trees, *Computers & Operations Research* 33:11, 3185–3202.

[4] S.P. Deepa, K.G. Srinivasa, K.R. Venugopal, L.M. Patnaik (2005). Dynamic association rule mining using genetic algorithms, *Intelligent Data Analysis* 9:5, 439–453.

[5] J.H. Holland (1992). *Adaptation in Natural and Artificial Systems* Cambridge, MA: MIT Press.

1. Randomly *select parents*.
2. *Reproduce* through *crossover*. Reproduction is the operator choosing which individual entities will survive. In other words, some objective function or selection characteristic is needed to determine survival. Crossover relates to changes in future generations of entities.
3. *Select survivors* for the next generation through a *fitness function*.
4. *Mutation* is the operation by which randomly selected attributes of randomly selected entities in subsequent operations are changed.
5. Iterate until either a given *fitness level* is attained, or the preset number of iterations is reached.

Genetic algorithm parameters include population size, crossover rate (the probability that individuals will crossover), and the mutation rate (the probability that a certain entity mutates).

Genetic Algorithm Advantages: Genetic algorithms are very easy to develop and to validate, which makes them highly attractive if they apply. The algorithm is parallel, meaning that it can applied to large populations efficiently. The algorithm is also efficient in that if it begins with a poor original solution, it can rapidly progress to good solutions. Use of mutation makes the method capable of identifying global optima even in very nonlinear problem domains. The method does not require knowledge about the distribution of the data.

Genetic Algorithm Disadvantages: Genetic algorithms require mapping data sets to a form where attributes have discrete values for the genetic algorithm to work with. This is usually possible, but can lose a great deal of detail information when dealing with continuous variables. Coding the data into categorical form can unintentionally lead to biases in the data.

There are also limits to the size of data set that can be analyzed with genetic algorithms. For very large data sets, sampling will be necessary, which leads to different results across different runs over the same data set.

Demonstration of Genetic Algorithm

The value of genetic algorithms is in their ability to deal with complex sets of data, where there are an unreasonably large number of variable combinations. One of the commonly applied applications of data mining is to loan applications. We use a loan application dataset with representative observations given in the Appendix. The purpose of the genetic algorithm in this case is simply to identify a set of loan applicant characteristics with the optimal fitness function. The example is very small, so you will know

the answer. The purpose is to show how the genetic algorithm would work, no matter how large the data set.

The steps of a genetic algorithm application begin with *discretizing the data*. In this case, the data has already been discretized, into three variables (age, income, and risk) each with three possible values. This provides a relatively simple case in that there are only 27 combinations of all of these variables. The Japanese credit example below shows that real problems often involve more variables, which would yield too many combinations to realistically enumerate. Therefore, while our example is simple, the intent is to demonstrate how a genetic algorithm would work.

The training set of 20 cases from Table 8.A1 can be organized in discretized form as shown in Table 8.1. Variable Age has three values (young, middle, and old), which are nominal. Variable Income has three values as well (low, average, and high), which are ordinal. Variable Risk's three values (high, average, low) are ordinal as well. Outcomes are OK and not (late or default).

Table 8.1. Discretized data for loan application building set

Case	Age	Income	Risk	Outcome
1	1	1	1	1
2	1	1	1	1
3	1	1	1	0
4	1	1	1	0
5	1	1	3	1
6	1	2	1	0
7	1	2	1	1
8	1	2	1	1
9	1	2	1	1
10	1	2	3	1
11	1	2	3	1
12	1	3	1	0
13	2	2	2	0
14	2	2	1	1
15	2	2	1	1
16	2	3	1	1
17	2	3	1	1
18	3	3	1	1
19	3	3	1	1
20	3	3	1	1

Table 8.2. Discretized data for loan application building set

Age	Income	Risk	Not OK	OK	Function
1	1	1	2	2	0.50
1	1	3	0	1	1.00
1	2	1	1	3	0.75
1	2	3	0	2	1.00
1	3	1	1	0	0.00
2	2	1	0	2	1.00
2	2	2	1	0	0.00
2	3	1	0	2	1.00
3	3	1	0	3	1.00

This data can be aggregated to obtain probabilistic outputs as in Table 8.2. (Note that we should have more observations, but again, our purpose is demonstration.) "Function" in the tables to follow is the probability of repayment on time, based upon the probabilities calculated from Table 8.1. This function will be the basis for selecting surviving genes.

Three steps are involved in the genetic algorithm:

1. Reproduction
2. Selection
3. Mutation

The algorithm is initialized by randomly generate parents from the building set. Here we will arbitrarily select four parents. The more parents selected, the longer each iteration will take, but the more thorough the algorithm will be as shown in Table 8.3.

We will pair these random selections as parents to generate the next generation. Reproduction is accomplished by crossing over randomly selected

Table 8.3. Randomly selected parents

Case	Age	Income	Risk	Function
3	1	1	1	0.50
13	2	2	2	0.00
12	1	3	1	0.00
7	1	2	1	0.75

Table 8.4. Generation of offspring – first case

	Case	Age	Income	Risk	Function
Parent 1	3	1	1	1	0.50
Parent 2	13	2	2	2	0.00
Offspring 1		1	2	1	0.75
Offspring 2		2	1	2	Not observed

variable values among parents. For instance, Table 8.4 shows cases 3 and 13 vary on all three variables. If the algorithm randomly selected the second variable (income) for crossover, the two generated cases would be (age 1, income 2, risk 1) and (age 2, income 1, risk 2). Outcomes for new cases can be obtained by the algorithm. Crossover may repeat one of the parents, and in fact if both parents have identical characteristics, repetition is guaranteed. However, the third step of mutation can break such deadlocks.

Selection can follow a number of rules, but we will select the two cases with the best function values. In this case, we generate one new solution, which is superior to both parents. This strain will carry forward the two alternatives with the best functional values (Parent 1, Offspring 1).

For the next pair of randomly selected parents, we can randomly select the variable to crossover. If Income were selected, the results would be as shown in Table 8.5:

Here reproduction generated nothing new. But if either of the other two variables were selected, the offspring would have been clones of the parents, again with nothing new. Instances like this make mutation useful. Mutation involves the random change of one of the selected cases. For instance, Offspring 3 could mutate by changing the value for age to some other value, such as 2. Algorithmically this would be accomplished randomly, both as to which variable was to change, as well as to what value it would change to.

Table 8.5. Generation of offspring – second case

	Case	Age	Income	Risk	Function
Parent 3	12	1	3	1	0.00
Parent 4	7	1	2	1	0.75
Offspring 3		1	2	1	0.75
Offspring 4		1	3	1	0.00

Table 8.6. Parents for second generation

Case	Age	Income	Risk	Function
Parent 1	1	1	1	0.50
Offspring 1	1	2	1	0.75
Parent 4	1	2	1	0.75
Mutation	2	2	1	1.00

At the end of the first iteration, we now have two sets of surviving parents for the second generation, shown in Table 8.6:

The second set of active parents generate new cases, randomly selecting variable Age for crossover, shown in Tables 8.7 and 8.8:

Here we have reached an impasse, with the offsprings reversing the characteristics of the parents. Thus, this strain has reached a dead end with respect to generating new solutions (unless mutation is applied).

Here we obtain two cases with strong survival characteristics (mutation, Offspring 5). Thus the algorithm succeeded in identifying three very good solutions (the mutation, and Offspring 5). There are other good solutions as well, but the purpose of the genetic algorithm was to identify any good solution. The implication is that loan applicants with middle age, middle income, and good scores on risk are highly likely to repay without complications. Note that we are not suggesting genetic algorithms as a means for data mining to predict repayment. We are trying to demonstrate how genetic algorithms can yield improved solutions in domains involving high degrees of nonlinearity, where traditional approaches such as linear

Table 8.7. Offspring in second generation in strain 1

Case	Age	Income	Risk	Function
Parent 1	1	1	1	0.50
Offspring 1	1	2	1	0.75
Offspring 5	1	2	1	0.75
Offspring 6	1	1	1	0.50

Table 8.8. Offspring in second generation in strain 1

Case	Age	Income	Risk	Function
Parent 4	1	2	1	0.75
Mutation	2	2	1	1.00
Offspring 5	2	2	1	1.00
Offspring 6	1	2	1	0.75

regression may have difficulties. Genetic algorithms are useful in data mining as a supplemental tool.

Application of Genetic Algorithms in Data Mining

Genetic algorithms have been applied to data mining in two ways.[6] External support is through evaluation or optimization of some parameter for another learning system, often hybrid systems using other data mining tools such as clustering or decision trees. In this sense, genetic algorithms help other data mining tools operate more efficiently. Genetic algorithms can also be directly applied to analysis, where the genetic algorithm generates descriptions, usually as decision rules or decision trees. Many applications of genetic algorithms within data mining have been applied outside of business. Specific examples include medical data mining[7] and computer network intrusion detection.[8] In business, genetic algorithms have been applied to customer segmentation,[9] credit scoring,[10] and financial security selection.[11]

Genetic algorithms can be very useful within a data mining analysis dealing with more attributes and many more observations. It saves the brute force checking of all combinations of variable values, which can make some data mining algorithms more effective. However, application of genetic algorithms requires expression of the data into discrete

---

[6] I. Bruha, P. Karlik, P. Berka (2000). Genetic learner: Discretization and fuzzification of numerical attributes, *Intelligent Data Analysis* 4, 445–460.

[7] Bhargava (1999), op. cit.

[8] T. Öyzer, R. Alhajj, K. Barker (2007). Intrusion detection by integrating boosting genetic fuzzy classifier and data mining criteria for rule pre-screening, *Journal of Network & Computer Applications* 30:1, 99–113.

[9] E. Kim, W. Kim, Y. Lee (2003). Combination of multiple classifiers for the customer's purchase behavior prediction, *Decision Support Systems* 34:2, 167–175; Y.-S. Kim, W.N. Street (2004). An intelligent system for customer targeting: A data mining approach, *Decision Support Systems* 37:2, 215–228.

[10] F. Hoffmann, B. Baesens, C. Mues, T. Van Gester, J. Vanthienen (2007). Inferring descriptive and approximate fuzzy rules for credit scoring using evolutionary algorithms, *European Journal of Operational Research* 177:1, 540–555; H. Zhao (2007). A multi-objective genetic programming approach to developing Pareto optimal decision trees, *Decision Support Systems* 43:3, 809–826.

[11] K. Mehta, S. Bhattacharyya (2004). Adequacy of training data for evolutionary mining of trading rules, *Decision Support Systems* 37:4, 461–474; D. Zhang, L. Zhou (2004). Discovering golden nuggets: Data mining in financial application, *IEEE Transactions on Systems, Man & Cybernetics: Part C* 34:4, 513–522.

outcomes, with a calculable functional value upon which to base selection. This does not fit all data mining applications. Genetic algorithms are useful because sometimes it does fit. We review an application to demonstrate some of the aspects of genetic algorithms.

Summary

Genetic algorithms operate on discretized data by systematically searching for better (or possibly optimal) combinations of variable values. They can be very efficient in problems with complex interactions, especially when nonlinear functions are involved. Genetic algorithms are valuable in data mining because of their ability to deal with imprecise and inconsistent information.

Genetic algorithms are usually applied in conjunction with other data mining techniques. They can be used to enhance the efficiency of other methods, or can be more directly applied.

Appendix: Loan Application Data Set

This data set (Table 8.A1) consists of information on applicants for appliance loans. The full data set involves 650 past observations. Applicant information on age, income, assets, debts, and credit rating (from a credit bureau, with red for bad credit, yellow for some credit problems, and green for clean credit record) is assumed available from loan applications. Variable Want is the amount requested in the appliance loan application. For past observations, variable OK is 1 if all payments were received on time, and 0 if not (Late or Default). The majority of past loans were paid on time. Data was transformed to obtain categorical data for some of the techniques. Age was groups by less than 30 (young), 60 and over (old), and in between (middle aged). Income was grouped as less than or equal to $30,000 per year and lower (low income), $80,000 per year or more (high income), and average in between. Asset, debt, and loan amount (variable Want) are used by rule to generate categorical variable risk. Risk was categorized as high if debts exceeded assets, as low if assets exceeded the sum of debts plus the borrowing amount requested, and average in between.

Table 8.A1. Loan application training data set

Age	Income	Assets	Debts	Want	Risk	Credit	Result
20 (young)	17,152 (low)	11,090	20,455	400	high	Green	OK
23 (young)	25,862 (low)	24,756	30,083	2,300	high	Green	OK
28 (young)	26,169 (low)	47,355	49,341	3,100	high	Yellow	Late
23 (young)	21,117 (low)	21,242	30,278	300	high	Red	Default
22 (young)	7,127 (low)	23,903	17,231	900	low	Yellow	OK
26 (young)	42,083 (average)	35,726	41,421	300	high	Red	Late
24 (young)	55,557 (average)	27,040	48,191	1,500	high	Green	OK
27 (young)	34,843 (average)	0	21,031	2,100	high	Red	OK
29 (young)	74,295 (average)	88,827	100,599	100	high	Yellow	OK
23 (young)	38,887 (average)	6,260	33,635	9,400	low	Green	OK
28 (young)	31,758 (average)	58,492	49,268	1,000	low	Green	OK
25 (young)	80,180 (high)	31,696	69,529	1,000	high	Green	Late
33 (middle)	40,921 (average)	91,111	90,076	2,900	Avg	Yellow	Late
36 (middle)	63,124 (average)	164,631	144,697	300	Low	Green	OK
39 (middle)	59,006 (average)	195,759	161,750	600	Low	Green	OK
39 (middle)	125,713 (high)	382,180	315,396	5,200	Low	Yellow	OK
55 (middle)	80,149 (high)	511,937	21,923	1,000	Low	Green	OK
62 (old)	101,291 (high)	783,164	23,052	1,800	Low	Green	OK
71 (old)	81,723 (high)	776,344	20,277	900	Low	Green	OK
63 (old)	99,522 (high)	783,491	24,643	200	Low	Green	OK

Table 8.A2 gives a test set of data.

Table 8.A2. Loan application test data

Age	Income	Assets	Debts	Want	Risk	Credit	Result
37 (middle)	37,214 (average)	123,420	106,241	4,100	Low	Green	OK
45 (middle)	57,391 (average)	250,410	191,879	5,800	Low	Green	OK
45 (middle)	36,692 (average)	175,037	137,800	3,400	Low	Green	OK
25 (young)	67,808 (average)	25,174	61,271	3,100	High	Yellow	OK
36 (middle)	102,143 (high)	246,148	231,334	600	Low	Green	OK
29 (young)	34,579 (average)	49,387	59,412	4,600	High	Red	OK
26 (young)	22,958 (low)	29,878	36,508	400	High	Yellow	Late
34 (middle)	42,526 (average)	109,934	92,494	3,700	Low	Green	OK
28 (young)	80,019 (high)	78,632	100,957	12,800	High	Green	OK
32 (middle)	57,407 (average)	117,062	101,967	100	Low	Green	OK

The model can be applied to the new applicants given in Table 8.A3:

Table 8.A3. New appliance loan applications

Age	Income	Assets	Debts	Want	Credit
25	28,650	9,824	2,000	10,000	Green
30	35,760	12,974	32,634	4,000	Yellow
32	41,862	625,321	428,643	3,000	Red
36	36,843	80,431	120,643	12,006	Green
37	62,743	421,753	321,845	5,000	Yellow
37	53,869	286,375	302,958	4,380	Green
37	70,120	484,264	303,958	6,000	Green
38	60,429	296,843	185,769	5,250	Green
39	65,826	321,959	392,817	12,070	Green
40	90,426	142,098	25,426	1,280	Yellow
40	70,256	528,493	283,745	3,280	Green
42	58,326	328,457	120,849	4,870	Green
42	61,242	525,673	184,762	3,300	Green
42	39,676	326,346	421,094	1,290	Red
43	102,496	823,532	175,932	3,370	Green
43	80,376	753,256	239,845	5,150	Yellow
44	74,623	584,234	398,456	1,525	Green
45	91,672	436,854	275,632	5,800	Green
52	120,721	921,482	128,573	2,500	Yellow
63	86,521	241,689	5,326	30,000	Green

9 Performance Evaluation for Predictive Modeling

Once a predictive model is developed using the historical data, one would be curious as to how the model will perform for the future (on the data that it has not seen during the model building process). One might even try multiple model types for the same prediction problem, and then, would like to know which model is the one to use for the real-world decision making situation, simply by comparing them on their prediction performance (e.g., accuracy). But, how do you measure the performance of a predictor? What are the commonly used performance metrics? What is accuracy? How can we accurately estimate the performance measures? Are there methodologies that are better in doing so in an unbiased manner? These questions are answered in the following sub-sections. First, the most commonly used performance metrics will be described, then a wide range of estimation methodologies are explained and compared to each other.

Performance Metrics for Predictive Modeling

In classification problems, the primary source of performance measurements is a coincidence matrix (a.k.a. classification matrix or a contingency table). Figure 9.1 shows a coincidence matrix for a two-class classification problem. The equations of most commonly used metrics that can be calculated from the coincidence matrix is also given below.

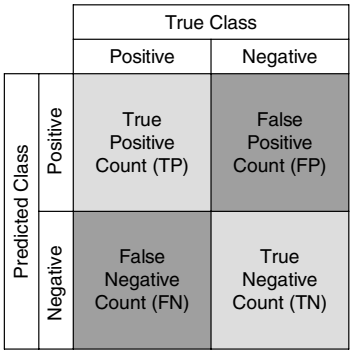


Fig. 9.1. A simple coincidence matrix

The numbers along the diagonal from upper-left to lower-right represent the correct decisions made, and the numbers outside this diagonal represent the errors. The true positive rate (also called hit rate or recall) of a classifier is estimated by dividing the correctly classified positives (the true positive count) by the total positive count. The false positive rate (also called false alarm rate) of the classifier is estimated by dividing the incorrectly classified negatives (the false negative count) by the total negatives. The overall accuracy of a classifier is estimated by dividing the total correctly classified positives and negatives by the total number of samples. Other performance measures, such as recall (a.k.a. sensitivity), specificity and F-measure are also used for calculating other aggregated performance measures (e.g., area under the ROC curves).

$$\text{True Positive Rate} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{9.1}$$

$$\text{True Negative Rate} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{9.2}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{9.3}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{9.4}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{9.5}$$

$$F - \text{measure} = \frac{2}{\frac{1}{\text{Presition}} + \frac{1}{\text{Recall}}} \tag{9.6}$$

Figure 9.2 illustrates an example for a classification matrix for a two-class classification problem. In this example the true positive rate is calculated (as per Eq. 9.1) as 88%, while the true negative rate is calculated (as per Eq. 9.2) as 62%. That is the classifier is doing a better job in correctly predicting the positives then predicting the negatives. The overall accuracy of the classifier is also calculated (as per Eq. 9.3) as 82%. Notice that the sample count for true positives and true negatives are not the same. Therefore one should reconsider the prior judgment of having better classification results for positive cases.

		Actual Classification of Classes in the Dataset		
		Class 1	*Class 2*	
Model Classification	*Class 1*	674	91	
	Class 2	89	146	
	Sum	30	30	
	Probability	0.76	0.24	
	Accuracy	0.88	0.62	0.82

Fig. 9.2. A sample coincidence matrix for a two class classifier

When the classification problem is not binary, the coincidence matrix gets a little more complicated (see Fig. 9.3). In this case the terminology of the performance metrics becomes limited to "per class accuracy rates" and the "overall classifier accuracy". These formulas are given below.

$$(\text{True Classification Rate})_i = \frac{(\text{True Classification})_i}{\sum_{i=1}^{n}(\text{False Classification})_i} \tag{9.7}$$

$$(\text{Overall Classifier Accuracy})_i = \frac{\sum_{i=1}^{n}(\text{True Classification})_i}{\text{Total Number of Cases}} \tag{9.8}$$

where i is the class number, n is the total number of classes.

		Actual Classification of Classes in the Dataset			
		Class 1	*Class 2*	*Class 3*	
Model Classification	*Class 1*	22	7	2	
	Class 2	5	18	7	
	Class 3	3	5	21	
	Sum	30	30	30	
	Probability	0.33	0.33	0.33	
	Accuracy	0.73	0.60	0.70	0.68

Fig. 9.3. A sample coincidence matrix for a three class classifier

Estimation Methodology for Classification Models

Estimating the accuracy of a classifier induced by some supervised learning algorithms is important for the following reasons. First, it can be used to estimate its future prediction accuracy which could imply the level of confidence one should have in the classifier's output in the prediction system. Second, it can be used for choosing a classifier from a given set (selecting the "best" model from two or more qualification models). Lastly, it can be used to assign confidence levels to multiple classifiers so that the outcome of a combining classifier can be optimized. Combined classifiers are increasingly becoming more popular due to the empirical results that suggest them producing more robust and more accurate predictions as they are compared to the individual predictors. For estimating the final accuracy of a classifier one would like an estimation method with low bias and low variance. In some application domains, to choose a classifier or to combine classifiers the absolute accuracies may be less important and one might be willing to trade off bias for low variance. Following subsections provides a high level coverage for the most popular estimation methodologies used for classification type data mining models.

Simple Split (Holdout)

The simple split (a.k.a. holdout or test sample estimation) partitions the data into two mutually exclusive subsets called a training set and a test set (or holdout set). It is common to designate 2/3 of the data as the training set and the remaining 1/3 as the test set. The training set is used by the inducer (model builder) and the built classifier is then tested on the test set. There is an exception to this rule where the classifier is an artificial neural network. There the data is partition into three mutually exclusive subsets; training, validation, testing. The validation set is used during the model building to prevent the over-fitting. Figure 9.4 shows the simple split estimation methodology.

The main criticism of this method is the fact that it makes the assumption that the data in the two subset are of the same kind (has the exact same properties). Since this is a simple random partitioning, in most realistic datasets where the data is skewed on the classification variable, such an assumption may not hold true. In order to improve this situation, stratified sampling is suggested, where the strata becomes the output variable. Even though this is an improvement over the simple split, it still has a bias associated from the single random partitioning.

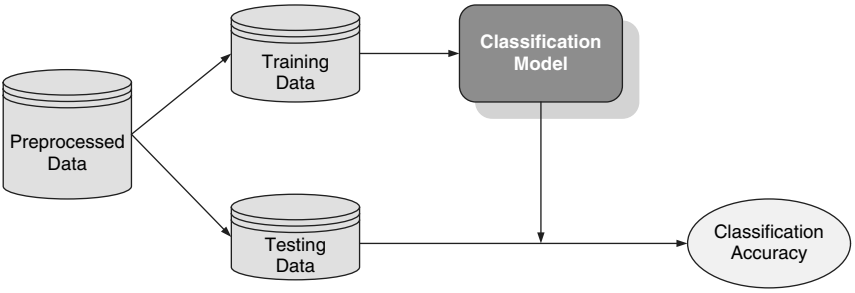


Fig. 9.4. Simple random data splitting

The *k*-Fold Cross Validation

In order to minimize the bias associated with the random sampling of the training and holdout data samples in comparing the predictive accuracy of two or more methods, one can use a methodology called *k*-fold cross validation. In *k*-fold cross validation, also called rotation estimation, the complete data set is randomly split into *k* mutually exclusive subsets of approximately equal size. The classification model is trained and tested *k* times. Each time it is trained on all but one folds and tested on the remaining single fold. The cross validation estimate of the overall accuracy of a model is calculated by simply averaging the *k* individual accuracy measures (Eq. 9.9).

$$CVA = \frac{1}{k}\sum_{i=1}^{k} A_i \tag{9.9}$$

where CVA stands for cross validation accuracy, *k* is the number of folds used, and *A* is the accuracy measure (e.g., hit-rate, sensitivity, specificity, etc.) of each folds.

Since the cross-validation accuracy would depend on the random assignment of the individual cases into *k* distinct folds, a common practice is to stratify the folds themselves. In stratified *k*-fold cross validation, the folds are created in a way that they contain approximately the same proportion of predictor labels (i.e., classes) as the original dataset. Empirical studies showed that stratified cross validation tend to generate comparison

results with lower bias and lower variance when compared to regular cross-validation.[1]

The *k*-fold cross validation is also called 10-fold cross validation, because the *k* taking the value of 10 has been the most common practice. In fact, empirical studies showed that *ten* seem to be an optimal number of folds (that optimizes the time it takes to complete the test and the bias and variance associated with the validation process)[2]. A pictorial representation of the *k*-fold cross validation where $k = 10$ is given in Fig. 9.5.

The methodology (step-by-step process) that one should follow in performing k-fold cross validation is as follows:

Step 1: The complete dataset is randomly divided into *k* disjoint subsets (i.e., folds) with each containing approximately the same number of records. Sampling is stratified by the class labels to ensure that the proportional representation of the classes is roughly the same as those in the original dataset.

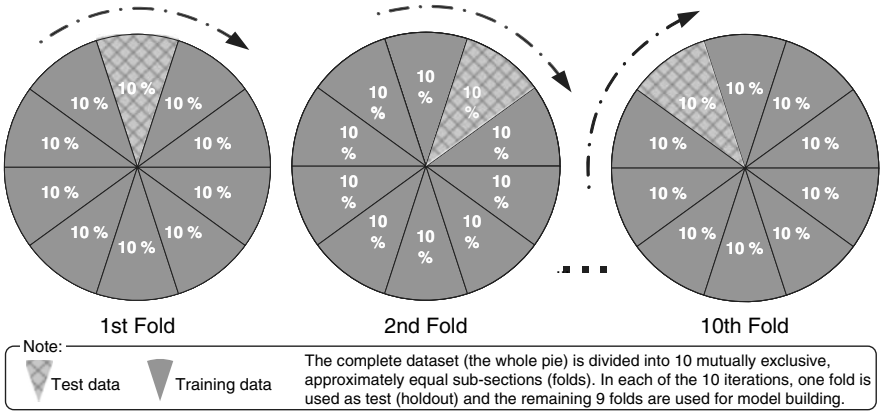


Fig. 9.5. Pictorial representation of 10-fold cross validation

[1] R. Kohavi (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, The Fourth International Joint Conference on Artificial Intelligence. Montreal, Quebec, Canada. American Association for Artificial Intelligence (AAAI): 202–209.

[2] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software; Kohavi (1995)., op cit.

Step 2: For each fold, a classifier is constructed using all records except the ones in the current fold. Then the classifier is tested on the current fold to obtain a cross-validation estimate of its error rate. The result is recorded.

Step 3: After repeating the step 2 for all 10 folds, the ten cross-validation estimates are averaged to provide the aggregated classification accuracy estimate of each model type.

Be reminded that 10-fold cross validation does not require more data compared to the traditional single split (2/3 training, 1/3 testing) experimentation. In fact, in data mining community, for methods-comparison studies with relatively smaller datasets, *k*-fold type of experimentation methods are recommended. In essence, the main advantage of 10-fold (or any number of folds) cross validation is to reduce the bias associated with the random sampling of the training and holdout data samples by repeating the experiment 10 times, each time using a separate portion of the data as holdout sample. The down side of this methodology is that, one needs to do the training and testing for k times ($k = 10$ in this study) as opposed to only once.

The *leave-one-out* methodology is very similar to the k-Fold cross validation where the value of k is set to 1. That is, every single data point is used for testing at least once on as many models developed as there are number of data pints. Even though this methodology is rather time consuming, for some small datasets it is a viable option.

Bootstrapping and Jackknifing

Especially for smaller datasets, it is necessary to creatively sample from the original data for performance estimation purposes. Bootstrapping, first introduced by Efron,[3] is a statistical method for estimating the sampling distribution of an estimator by sampling with replacement from the original sample, most often with the purpose of deriving robust estimates of standard errors and confidence intervals of a population parameter like a mean, odds ratio, correlation coefficient or regression coefficient. In estimating the performance of classification methods, bootstrapping is used to saple a fix number of instances from the original data to use for training and use the rest of the dataset for testing. This process can be repeated as many times as the experimenter desires. It can be repeated for instance 10 times much the same way the *k*-fold cross-validation is performed. Though

[3] B. Efron (1979). Bootstrap methods: Another look at the jackknife, *The Annals of Statistics* 7, 1–26.

they are seemingly similar, there is a difference between *k*-fold cross-validation and bootstrapping which is bootstrapping randomly samples from the original dataset with replacement (allowing the same instance to appear in the bootstrap sample more than once) while *k*-fold cross-validation randomly splits the original dataset into *k* mutually exclusive sub sections (not allowing the same instance to appear in any of training set more than once).

Jackknifing, which is similar to bootstrapping, is originally used in statistical inferencing to estimate the bias and standard error in a statistic, when a random sample of observations is used to calculate it. The basic idea behind the jackknife estimator lies in systematically recomputing the statistic estimate leaving out one observation at a time from the sample. From this new set of "observations" for the statistic an estimate for the bias can be calculated and an estimate for the variance of the statistic. As it sounds, it is a similar method to leave-one-out experimentation methodology.

Both bootstrapping and jackknifing methods estimate the variability of a statistic (e.g., percent-hit rate) from the variability of that statistic between subsamples, rather than from parametric assumptions. The jackknife is a less general technique than the bootstrap, and explores the sample variation differently. However the jackknife is easier to apply to complex sampling schemes for complex prediction problems, such as multi-stage sampling with varying sampling weights, than the bootstrap. The jackknife and bootstrap may in many situations yield similar results. But when used to estimate the standard error of a statistic, bootstrap gives slightly different results when repeated on the same data, whereas the jackknife gives exactly the same result each time (assuming the subsets to be removed are the same).

Area Under the ROC Curve

A Receiver Operating Characteristics (ROC) curve is a technique for visualizing, organizing and selecting classifiers based on their performance. In essence, it is another performance evaluation technique for classification models. ROC curves have long been used in signal detection theory to depict the tradeoff between hit rates and false alarm rates of classifiers.[4] The use of ROC analysis has been extended into visualizing and analyzing the

[4] J.P. Egan (1975). *Signal Detection Theory and ROC Analysis, Series in Cognition and Perception*. New York: Academic Press.

behavior of diagnostic systems.[5] Recently, the medical decision making community has developed an extensive literature on the use of ROC curves as one of the primary methods for diagnostic testing.[6]

Given a classifier and an instance, there are four possible prediction outcomes. If the instance is positive and it is classified as positive, it is counted as a true positive; if it is classified as negative, it is counted as a false negative. If the instance is negative and it is classified as negative, it is counted as a true negative; if it is classified as positive, it is counted as a false positive. Given a classifier and a set of instances (the test set), a two-by-two coincidence matrix (also called a contingency table) can be constructed representing the dispositions of the set of instances (see Fig. 10.1). This matrix forms the basis for many common metrics including the ROC curves.

ROC graphs are two-dimensional graphs in which true positive (TP) rate is plotted on the *Y* axis and false positive (FP) rate is plotted on the *X* axis (see Fig. 9.6). In essence, an ROC graph depicts relative trade-off between benefits (true positives) and costs (false positives). Several points in ROC space are important to note. The lower left point (0; 0) represents the strategy of never issuing a positive classification; such a classifier commits no false positive errors but also gains no true positives. The opposite strategy, of unconditionally issuing positive classifications, is represented by the upper right point (1; 1). The point (0; 1) represents perfect classification. Informally, one point in ROC space is better than another if it is to the northwest (TP rate is higher, FP rate is lower, or both) of the first. Classifiers appearing on the left hand-side of an ROC graph, near the *X* axis, may be thought of as "conservative": they make positive classifications only with strong evidence so they make fewer false positive errors, but they often have low true positive rates as well. Classifiers on the upper right-hand side of an ROC graph may be thought of as "liberal": they make positive classifications with weak evidence so they classify nearly all positives correctly, but they often have high false positive rates. Many real world domains are dominated by large numbers of negative instances, so performance in the far left-hand side of the ROC graph becomes more interesting.

An ROC curve is basically a two-dimensional depiction of a classifier's performance. To compare classifiers or to judge the fitness of a single classifier one may want to reduce the ROC measures to a single scalar value representing the expected performance. A common method to perform such

---

[5] J. Swets (1988). Measuring the accuracy of diagnostic systems, *Science* 240, 1285–1293.

[6] K.H. Zou (2002). Receiver operating characteristic (ROC) literature research, Online bibliography at http://splweb.bwh.harvard.edu.

task is to calculate the area under the ROC curve, abbreviated as AUC. Since the AUC is a portion of the area of the unit square, its value will always be between 0 and 1.0. A perfect accuracy gets a value of 1.0. The diagonal line $y = x$ represents the strategy of randomly guessing a class. For example, if a classifier randomly guesses the positive class half the time (much like flipping a coin), it can be expected to get half the positives and half the negatives correct; this yields the point (0:5; 0:5) in ROC space, which in turn translates into area under the ROC curve value of 0.5. No classifier that has any classification power should have an AUC less than 0.5.

In Fig. 9.6 classification performance of three classifiers (A, B and C) are shown in a single ROC graph. Since the AUC is the commonly used metric for performance comparison of prediction models, one can easily tell that the best performing classifier (out of the three that is being compared to each other) is A, followed by B. The classifier C is not shoving any predictive power; staying at the same level as random chance.

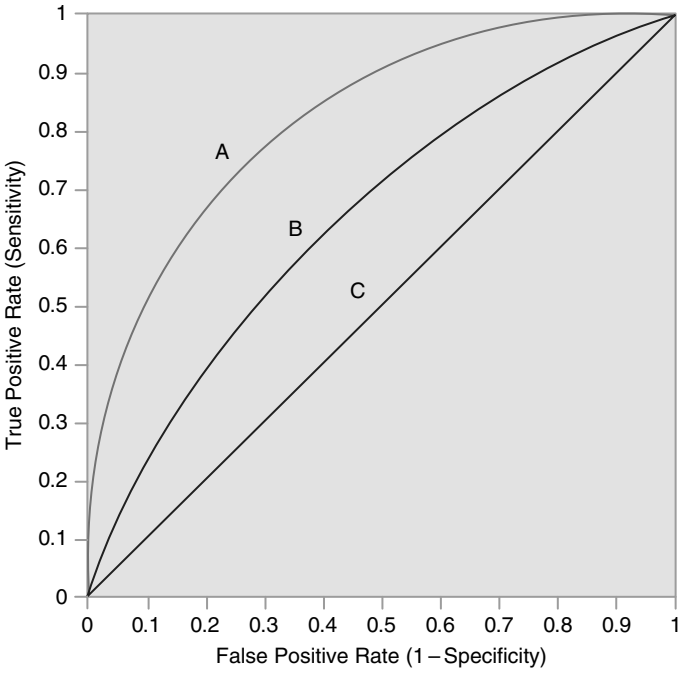


Fig. 9.6. A sample ROC curve

Summary

Above listed estimation methods illustrates a wide range of options for the practitioner. Some of them are more statistically sound (based on sound theories) while others have empirically showed to produce better results. Additionally, some of these methods are computationally more costly then the others. It is not always the case that increasing the computational cost is beneficial especially if the relative accuracies are more important than the exact values. For example leave-one-out is almost unbiased but it has high variance leading to unreliable estimates, not to mention the computational cost that it bring into the picture while dealing with relatively large datasets. For linear models using leave-one-out cross-validation for model selection is asymptotically inconsistent in the sense that the probability of selecting the model with the best predictive power does not converge to one as the total number of observations approaches infinity.[7] Based on the recent trends in data mining, two of the most commonly used estimation techniques are area under the ROC curve (AUC) and the stratified *k*-fold cross validation. That said, most of the commercial data mining tools are still promoting and using simple split as the estimation technique.

[7] P. Zhang (1992). On the distributional properties of model selection criteria, *Journal of the American Statistical Association* 87:419, 732–737.

Part III
APPLICATIONS

10 Applications of Methods

This chapter present some reported applications of the methods presented in the prior chapters. The methods were selected because they were relatively specialized, so there are fewer reported applications of these methods than of the standard methods such as neural networks, logistic regression, or decision trees. Furthermore, a common practice is to run multiple models, so some of these reports compare results of a variety of models, or even combine them in ensemble models. The main point is that the methods covered in this book expand the toolbox available to data miners in their efforts to discover knowledge.

Memory-Based Application

This application involves the study of work-related lower back disorders, which have been the source of roughly one-fourth of worker compensation claims in the United States.[1] The top three causes of injury were overexertion, falls, and bodily reaction were the fastest growing injury causes, and were the source of about half of insurance claims.

The data mining study reported involved binary classification. Industry data on 403 industrial lifting jobs from 48 manufacturers was collected, and divided into high risk and low risk cases relative to lower back disorder. High risk jobs were defined as those with at least 12 incidents reported for 200,000 hours of exposure, involving 111 cases. Low risk was defined as jobs with at least 3 years of records with no reported injuries and no turnover (turnover defined as the average number of workers leaving that job per year). There were 124 low risk cases. The other 168 cases in the database were not used in the study. This data treatment provided two distinctly different categories of outcome. There were five independent variables, given in Table 10.1:

---

[1] J. Zurada, W. Karwowski, W. Marras (2004). Classification of jobs with risk of low back disorders by applying data mining techniques, *Occupational Ergonomics* 4:4, 291–305.

Table 10.1. Lower back risk study variables

Variable Type	Acronym	Definition	Count
Dependent	RISK	Risk of lower back disorder	Low (0) – 124; High (1) – 111
Independent	LIFTR	Number of lifts per hour	
Independent	PTVAVG	Peak twist velocity average	
Independent	PMOMENT	Peak moment	
Independent	PSUP	Peak sagittal angle	
Independent	PLVMAX	Peak lateral velocity maximum	

The analysts applied neural networks, logistic regression, and decision tree models, the standard binary classification tools. They also applied memory-based reasoning and an ensemble model. The memory-based reasoning model was a *k*-nearest neighbor algorithm, which classified cases by the lowest Euclidean distance to the targets representing high risk and low risk. The parameter *k* was the number of cases considered in classifying each observation. The *k* closest cases to a given observation were identified, using the five independent variables in the distance calculation. Posterior probability was simply the proportion of these *k* cases that with the given classification outcome. For instance, if *k* were 3, the squared distance to each of the training observations would be calculated, and the three closest neighbors identified. If two of these closest neighbors were high, and one low, the probability assigned to the target case would be 0.67 high and 0.33 low. No model needed to be fitted, but all observations had to be retained in memory. The study utilized a *k* of 14.

The ensemble model created a new model by averaging posterior probabilities for class targets based upon neural network, regression, and decision tree models, using SAS Enterprise Miner. The ensemble model may be more accurate than individual models if the individual models have divergent predictions. Data was stratified into three groups, one for training (94 cases), one for validation to fine tune models (71 cases), and the last for testing models (70 cases). To counter the problems created by reducing data size, 10-fold cross-validation was applied, meaning a computer simulation for ten random generations of the three data sets was used and classification results were averaged.

All five of the models yielded similar results, averaging between 70.6% and 75.6% accuracy. The memory-based reasoning model turned out to have the best average accuracy, while logistic regression yielded the best accuracy for the low classification and the decision tree model the best accuracy for the high classification. The 1991 National Institute

of Occupational Safety and Health guide was in the same range for the high classification, but had much lower accuracy for the low classification.

The reported application demonstrated that memory-based reasoning models can be effective in classification applications. They are additional tools expanding the options available. They are relatively simple models, but face the limitation that most commercial data mining software does not include them for classification.

Association Rule Application

Warranty costs are important factors in the convergence of products and services. There is more focus on the value provided to customers as opposed to simply making products with which to stock shelves. Monitoring warranty claims is important for a variety of reasons. Warranty claims are indications of product quality problems, and can lead to improved product development processes or better product designs.

In the automotive industry, there are millions of transactions of warranties each year, providing massive databases of warranty information. This warranty data is typically confidential as it can affect the manufacturer's reputation. Machine learning methods to detect significant changes in warranty cost performance are very useful.[2] This study obtained data from multiple sources. Production data involved where each auto was manufactured, the parts that went into it, and so forth. Sales data was by dealer, to include date of sale, repair experience, and warranty claims. Some repair information came from independent repair shops. Integrating this data was a challenge, although the vehicle identification number served as a common key. A major problem source was the massive variety of ways in which different dealerships and repair shops collected and stored data. Data types included temporal (dates), numerical (costs, mileage), and categorical (labor codes, repair types, automobile types). Automobiles of course include hundreds of attributes in large volumes, leading to unified warranty databases of several millions of records.

Association rule-mining was used to find interesting relationships. Example outcomes include relationships between particular product sources and defects in the form of IF–THEN relationships. This approach is especially useful given the massive database size involved. An improved algorithm was developed by Buddhakulsomsiri et al. using automotive warranty data

[2] J. Buddhakulsomsiri, Y. Siradeghyan, A. Zakarian, X. Li (2006). Association rule-generation algorithm for mining automotive warranty data, *International Journal of Production Research* 44:14, 2749–2770.

Table 10.2. Automotive warranty dataset variables

Variable	Possible Values
Labor code	2,238
Fuel economy code	2,213
Production month-year	35
Engine family	28
Merchandising model code	20
Mileage-at-repair	14
Vehicle series	10
Engine	6
Chassis package	5
Transmission	4

collected over 2 years, containing over 680,000 records of warranty claims for one specific vehicle model. Each claim had 88 attributes involving 2,238 different labor codes. The first step was data cleansing, deleting observations with missing or irrelevant data. Continuous attributes were clustered using a *k*-means clustering using analyst expertise to set *k*. This reduced the data set down to ten variables, given in Table 10.2:

Labor code was used as the decision attribute, and the other nine variables condition attributes. All possible association rules were considered (1 to 1 up through 9 to 1), making for a very large number of possible combinations. To focus on the most significant rules, the total warranty cost by labor code was identified. Association rule parameters for minimum support and confidence. For the top ten labor codes by warranty cost, 2,896 rules were generated at the loosest set of parameter settings. By tightening these parameters, the number of rules were reduced, down to 13 using the strictest set of parameters.

An example of rules obtained included:

Rule A: IF Engine = E02 THEN Labor Code = D7510

Rule B: IF Production month = March AND Production year = 2002
AND Transmission = T05
THEN Labor code = K2800

Rules thus provided guidance as to the type of repair work typically involved with various combinations of attribute values. For each rule, the support, confidence, number of cases, and total warranty cost was identified.

The procedure proposed in Buddhakulsomsiri et al. reduced computation time from over 10 hours to 35 seconds for 300,000 objects.

Fuzzy Data Mining

Fuzzy analysis can be applied to pretty much any data mining technique. Commercial software usually includes fuzzy features for decision tree models involving continuous numbers, for instance. The way these work in See5 (which is used in Clementine) is to identify rules using the training set, then applying the resulting model on the test set, off-setting numerical limits so that the same results are attained, but broadening limits for future observations. Thus in those applications it only applies to continuous numerical data. Most software doesn't specify precisely what they do (e.g., Polyanalyst).

Fuzzy modeling has been applied to many other algorithms, to include the *k*-means algorithm.[3] Chang and Lu applied the fuzzy c-means method to analysis of customer demand for electricity.[4] In this clustering approach, each data point belongs to a cluster to some degree rather than crisp yes/no (1/0). The method starts with an initial guess to the current mean for each cluster, and iteratively updates cluster centers and membership functions. In principle, fuzzy analysis is not a distinct data mining technique, but can be applied within a number of data mining techniques.

Rough Set Models

Like fuzzy numbers, rough sets are a way of viewing data that can be useful in reflecting reality. Rough sets apply to categorical data. Hassanien reported application of rough sets to breast cancer data.[5] The data set consisted of 360 samples of needle aspirates from human breast tissue with nine measures (condition attributes) with values on a 1–10 integer scale:

---

[3] G. Guo, D. Neagu (2005). Fuzzy *k*NNmodel applied to predictive toxicology data mining, *International Journal of Computational Intelligence and Applications* 5:3, 321–333.

[4] R.F. Chang, C.N. Lu (2003). Load profile assignment of low voltage customers for power retail market applications, *IEEE Proceedings – Generation, Transmission, Distribution* 150:3, 263–267.

[5] A.E. Hassanien (2003). Intelligent data analysis of breast cancer based on rough set theory, *International Journal on Artificial Intelligence Tools* 12:4, 465–479.

A1. Clump thickness
A2. Uniformity of cell size
A3. Uniformity of cell shape
A4. Marginal adhesion
A5. Single epithelial cell size
A6. Bare nuclei
A7. Bland chromatin
A8. Normal nucleoli
A9. Mitoses

The outcome variable was binary (benign or malignant). Rough set theory was used to obtain lower and upper approximations. Some condition attributes did not provide additional information about outcomes, so the rough set process includes attribute reduction by removing such attributes. Decision rules were generated on the reduct system over those attributes best distinguishing benign and malignant cases. The set of rules obtained can still be quite large, so significance of rules was assessed. The more rigorous this significance test, the fewer rules retained. The process included six steps:

1. Check for relevant and irrelevant attributes,
2. Check dependencies between a single attribute and a set of attributes,
3. Select system reducts,
4. Extract hidden rules,
5. Generalize rules for better understanding,
6. Test the significance of each rule.

In the application, of 360 patients 27 reducts were identified. This data had high dependency among attributes, and there were many possibilities for identifying cases with substitute rules. For instance, the reduct A2, A3, A5 generated 25 rules, while the reduct A3, A7, A9 generated 18 rules. By looking at approximation qualities, the reduct consisting of A7 (bland chromatin) and A8 (normal nuclei) was found to be the best choice for classification, generating 23 rules. The generalization step reduced 428 rules generated in step 4 down to 30 rules. Example rules included:

IF A7 = 1 AND A8 = 2 THEN benign	based on 69 instances
IF A7 = 1 AND A8 = 1 THEN benign	based on 87 instances
IF A8 = 8 THEN malignant	based on 9 instances

The method was compared with a decision tree algorithm, which generated 1,022 rules prior to pruning, and 76 rules after pruning. The rough set

rules were found to be 98% accurate, while the decision tree pruned rules were 85% accurate. Thus in this one case, the rough set method was more accurate and more compact (had fewer rules).

Support Vector Machine Application

Stock price forecasting is a topic of major interest among business students and practitioners. It also has proven challenging in a rigorous real environment. Technical analysis (charting) uses technical indicators of stock prices and volumes as guides for investment. Over 100 indicators have been developed, and selecting which indicators to use is challenging. Technical indicators vary in that some model market fluctuations and other focus on buy/sell decisions. They tend to be highly dependent on asset price, and are usually specific in that they may not work well for all stocks. Technical indicators can be used for both long-term and short-term decisions.

Statistical techniques include kernel Principal Component Analysis (kPCA) and factor analysis. Many heuristics (rules of thumb; simplified rules) have been developed based on experience. Ince and Trafalis used these four approaches to select inputs, and then applied two machine learning techniques (support vector machines and neural network models) for forecasting.[6]

The study analyzed short-term stock price forecasting, defined as a 2–3 weeks time horizon. Daily stock prices of ten companies traded on NASDAQ were used. The average training set consisted of 2,500 observations, with 200 test observations. In the first stage of the experiment, the objective was to identify indicators to explain upside and downside stock price movement. Kernel PCA and factor analysis were used to reduce input size. Kernel PCA selected ten technical indicators, while factor analysis selected 12. The relationships between stock price and these indicators was highly nonlinear and highly correlated. Thus support vector machines and neural network models were expected to work well as they are nonparametric. The support vector machine algorithm used a radial basis kernel function. Once this kernel function was selected, ten-fold cross-validation was used, guiding parameter selection. Overall, the heuristic models produced better results when support vector machines or neural network models were used as training methods. The neural network model yielded slightly better results for mean-squared-error on average, but not significantly better than the support vector machine-trained models.

---

[6] H. Ince, T.B. Trafalis (2007). Kernel principal component analysis and support vector machines for stock price prediction, *IIE Transactions* 39, 629–637.

Genetic Algorithm Applications

Three business applications using genetic algorithms in data mining are briefly described here. The first is an example of a widely used application, evaluating credit risk of loan applicants. The second business example involves use of genetic algorithms in a hybrid model involving rough set theory in design of product quality testing plans. The third example uses genetic algorithms to improve neural network performance on the classical data mining problem of customer segmentation. Finally, a fourth example (from the medical field) is presented to show how decision problems can be transformed to apply genetic algorithms.

Japanese Credit Screening

Bruha et al. (2000) tested a hybrid model involving fuzzy set theory and genetic algorithms on a number of data sets.[7] One of these data sets involve credit screening in a Japanese environment. This database consisted of 125 cases with ten variables:

1. Jobless state
2. What loan request was for
3. Sex of applicant
4. Marital status
5. Problematic region
6. Age of applicant
7. Amount applicant held on deposit at the bank
8. Monthly loan payment amount
9. Number of months expected for loan payoff
10. Number of years working at current company

Output classes were credit was granted (85 cases) or not (40 cases). Thus, the data mining analysis was focused on replicating the loan approval process of the expert bankers. Specific variable data, such as item 10 (number of years working with the company) were analyzed to determine useful cutoffs to classify this variable. Those with lower years at their current company had higher frequencies of loan denial. The cutoff was determined by using the building data set of 125 cases and finding the specific value of number of years with current company that minimized classification error. This enabled

[7] I. Bruha, P. Karlik, P. Berka (2000). Genetic learner: Discretization and fuzzification of numerical attributes, *Intelligent Data Analysis* 4, 445–460.

efficient transformation of this variable into a categorical form with two possible values. The genetic algorithm supported a hybrid data mining algorithm by optimally categorizing a variable.

Product Quality Testing Design

Rough set theory and genetic algorithms were combined in a hybrid data mining analysis to determine the best product quality test plan for a manufacturing firm.[8] Rough set theory deals with inconsistent information using upper and lower approximations for specific model components. In that application involving an electronic device, there were critical product components for which assurance of performance was required prior to assembly. Stringent acceptance tests were usually conducted to confirm conformance to specifications. These acceptance tests were tedious and costly, and could slow the entire production process.

The technique was demonstrated on a case involving two critical attributes, one with three settings, and the other with four settings. Acceptance tests were conducted in situ on the production machine, thus shutting down the production line. Past acceptance test data was available, and it was desired that decision rules be identified that would expedite this part acceptance process.

A total of 170 sets of historical data were available, including 158 good units and 12 bad units. There were two output classes (good and bad), and thus only one cutting point for the function developed by the genetic algorithm was needed. The two continuous variable inputs were discretized. Rough set theory was used to overcome inconsistencies introduced by this discretization step. The fitness function was defined as the square of the ratio of correct classifications over total number of opportunities by category. The genetic algorithm generated four rules providing the highest reliability in terms of probability of correct classification. This would enable the manufacturing firm to reduce testing time tremendously, and thus increase production output without diminishing output quality.

Customer Targeting

This application used genetic algorithms with neural networks to develop a predictive model to select an optimal set of potential customers to send

[8] L.-Y. Zhai, L.-P. Khoo, S.-C. Fok (2002). Feature extraction using rough set theory and genetic algorithms – an application for the simplification of product quality evaluation, *Computers & Industrial Engineering* 43, 661–676.

marketing materials.[9] This is a classical customer segmentation application of data mining. The traditional model seeks to identify as many customers as possible who would respond to a specific solicitation campaign letter, utilizing the estimated probability of response.

Genetic algorithms were used in this application to avoid local optima in a search over a complex response space. The genetic algorithm was used to select combinations of available variables for use in the neural network model. Each neural network model was tested on an evaluation set for both cumulative hit rate (maximizing number of responding targets) and complexity (minimize the number of variables used in the model). Cumulative hit rate was defined as the ratio of the number of actual customers identified divided by the total number of actual customers in the data set. Complexity was measured as:

$$1.0 - (\text{variables in the model} - 1) / (\text{total variables available} - 1)$$

Data was taken from 9,822 European households contacted about recreational vehicle insurance. A training set of 5,822 households was selected, using the remaining 4,000 cases as a test set. Of the 5,822 cases in the training set, 348 purchased the insurance, yielding a hit rate of 0.0597. The predictive model intended to identify the most likely purchasers. Each case included outcome (a binary variable of whether or not insurance was purchased) and 93 other variables (51 demographic, 42 relating to ownership of various types of insurance).

The model was run repeatedly over different target points, obtaining local solutions (having the highest fitness value at a specific target point). Once local solutions were obtained, they were combined into a global solution used to select the best target point (a procedure referred to as Ensemble). A lift curve was used to show the relationship between target points and the corresponding hit rate. Target proportions were set as all nine multiples of ten percent. The Ensemble procedure had the best performance at five of these nine target points, having the greatest superiority at small target points. The Ensemble solution was credited with returning the highest expected profit.

Medical Analysis

Exploratory data analysis involves gaining statistical insight and development of models driven by data. Often data mining studies involve large

[9] Y.S. Kim, W.N. Street (2004). An intelligent system for customer targeting: A data mining approach. *Decision Support Systems* 37:2, 215–228.

sets of data over many attributes, leading to a very large number of variable/category combinations.[10]

This study involved analysis of Persian Gulf War illnesses. Medical research was inconclusive as to the existence of specific illnesses related to the conflict, but there were large amounts of medical data generated in the effort to find out. The data mining study was based on the premise that is a syndrome existed; it would appear through higher proportions of occurrence in groups with similar attribute characteristics.

Problem Definition: There were over 20,000 observations, with measurements on over 150 attributes. These attributes involved the categories of demographic data, exposure data (by type of pathogen), reported symptoms, and diagnoses. The goal of the study was to efficiently find patterns in the data.

Problem Decomposition: At least 80 of the 150 attributes seemed relevant to the study. There were no prior hypotheses. The complexity of the database made decoupling the pattern search problem (identifying fit of a model based on a set of attributes) from the attribute selection model. Therefore, a genetic algorithm was used to select a set of attributes, which was then run over the training set of data (randomly selected from the full data set) to build a model. The proportion of training cases correctly classified was used as a fitness function for this attribute combination. This was fed back to the genetic algorithm for further genetic development.

Attribute Search: The genetic search process was iterative, with each iteration selecting a subset of attributes for use in the pattern search subproblem. Attributes were expressed as a string of 0/1 values representing *alleles*. The initial set of attribute values was randomly generated. At each iteration, reproduction was accomplished by crossover and mutation. Fitness of prior alleles was used to guide reproduction, making for a more efficient genetic algorithm.

Three studies were conducted. There was a study of exposures versus diagnoses, based on 32 independent variables, 29 binary variables, 21 binary dependent variables, and 21 diagnoses. A similar set of variables was used for a study of exposures relative to symptoms. A third study based upon a chemical dump explosion was conducted with a larger set of variables.

---

[10] H.K. Bhargava (1999). Data mining by decomposition: Adaptive search for hypothesis generation. *INFORMS Journal on Computing* 11:3, 239–247.

The size of the problem led to extensive computer run time. Results were excellent in terms of hypothesis quality improvement. Analysis times varied from 30 to 45 hours (about 130 iterations) on a 90 megahertz Pentium computer. The average population fitness score increased about 60 times in 30 generations.

The study revealed the following outcomes that were not previously evident:

- Of the cases diagnosed with lower back pain and major depression, 94% were in the Army, with a rate four times that of any other service. This was considered to be not surprising, and a conclusion that made sense.
- Participants within 10 kilometers of the chemical dump explosion were over five times as likely to suffer from muscle pain, rash, but not chest pain. This was considered a useful hypothesis for further study.
- Participants reporting exposure to pesticides and food outside of the military system were almost four times as likely to have osteoarthritis and sleep apnea. This combination of factors was considered surprising, and worthy of further research.

Data mining studies can involve highly complex combinations of attributes and possible attribute values. If the data space can be converted to genetic form, genetic algorithms can provide a useful means to efficiently generate better models. An interesting feature of this study was the movement between genetic algorithm to generate a set of variables for a parametric model that was used to generate a fitness function to guide generation of the next iteration's genetic algorithm.

Predicting the Financial Success of Hollywood Movies

In this section we describe a data mining based decision support system aimed at helping Hollywood managers make better decisions on important movie characteristics (e.g., genre, super stars, technical effects, release time, etc.) in order to optimize the financial return on their investments. In the motion picture industry, where the results of managerial decisions are measured in millions of dollars, managers are expected to make the right decisions in a timely manner to take advantage of the opportunities. In order to succeed, they need decision support systems powered by data mining driven prediction models. A set of parameters characterizing motion pictures are used to build range of prediction models to classify a movie in one of nine success categories, ranging from a "flop" to a "blockbuster".

Problem and Data Description

Prediction of financial success of a movie is arguably the most important piece of information needed by decision makers in the motion picture industry. Knowledge about the main factors affecting the financial success of a movie (and their level of influence) would be of great value in making investment and production related decisions. However, forecasting financial success (box-office receipts) of a particular motion picture has always been considered as a rather difficult and challenging problem. To some Hollywood is the land of hunches and the wild guesses due to the uncertainty associated with predicting the product demand. In support of such observations, Jack Valenti, long-time president and CEO of the Motion Picture Association of America, once stated that "…No one can tell you how a movie is going to do in the marketplace … not until the film opens in darkened theatre and sparks fly up between the screen and the audience". Trade journals and magazines of the motion picture industry have been full of examples, statements, and experiences that support such a claim.

The difficulty associated with the unpredictable nature of the problem domain has intrigued researchers to develop models for understanding and potentially forecasting the financial success of motion pictures. The major studies on predicting financial success of motion pictures were summarized by Litman and Ahn.[11] Most analysts have tried to predict the box-office receipts of motion pictures after a movie's initial theatrical release. Because they attempt to determine how a movie is going to do based on the early financial figures, the results are not usable to make investment and have had production related decisions, which are to be made during the planning phase. Some studies have attempted to forecast the performance of a movie before it is released but had only limited success. These studies, which are either good for predicting the financial success of a movie after its initial theatrical release or are not accurate enough predictors for decision support, leave us with an unsatisfied need for a forecasting system capable of making a prediction prior to a movie's theatrical release. Research by Delen et al. aims to fill this need by developing and embedding an information fusion-based forecasting engine into a Web-based decision support system that could be used by Hollywood managers.[12]

---

[11] B.R. Litman, H. Ahn (1998). Predicting financial success of motion pictures, in B.R. Litman, *The Motion Picture Mega-Industry* Boston: Allyn & Bacon Publishing, Inc.

[12] D. Delen, R. Sharda, P. Kumar (2007). Movie forecast Guru: A web-based DSS for Hollywood managers, *Decision Support Systems* 43:4, 1151–1170.

The initial launch of this study was in 1998. For each year since 1998, the available data has been collected, organized, and analyzed on a year-by-year basis using 834 movies released between 1998 and 2002. The sample data was drawn (and partially purchased) from ShowBiz Data Inc. In order to predict the financial success (e.g., box-office receipts) of a particular motion picture, Delen et al. used seven different types of independent variables. The choice of independent variables was based on the availability/accessibility of these variables and the insight gained from the previous studies conducted in this application domain. The list of variables along with their definitions and possible values is listed in Table 10.3.

The dependent variable in this study was the box-office gross revenues. A movie was classified based on its box-office receipts in one of nine categories, ranging from a "flop" to a "blockbuster." This process of converting a continuous variable in a limited number of classes is commonly called as "discretization". Many prefer using discrete values as opposed to continuous

Table 10.3. Summary of independent variables

Independent variable name	Definition	Range of Possible Values
MPAA Rating	The rating assigned by the Motion Picture Association of America (MPAA).	G, PG, PG-13, R, NR
Competition	Indicates the level at which each movie competes for the same pool of entertainment dollars against movies released at the same time.	High, Medium, Low
Star value	Signifies the presence of any box office superstars in the cast. A superstar actor/actress can be defined as one who contributes significantly to the up-front sale of the movie.	High, Medium, Low
Genre	Specifies the content category the movie belongs to. Unlike the other categorical variables, a movie can be classified in more than one content category at the same time (e.g., action as well as comedy). Therefore, each content category is represented with a binary variable.	Sci-Fi, Epic Drama, Modern Drama, Thriller, Horror, Comedy, Cartoon, Action, Documentary
Special effects	Signifies the level of technical content and special effects (animations, sound, visual effects) used in the movie.	High, Medium, Low
Sequel	Specifies whether a movie is a sequel (value of 1) or not (value of 0).	Yes, No
Number of screens	Indicates the number of screens on which the movie is planned to be shown during its initial launch.	A positive integer between 1 and 3876

ones in prediction modeling because (i) discrete values are closer to a knowledge-level representation, (ii) data can be reduced and simplified through discretization, (iii) for both users and experts, discrete features are easier to understand, use, and explain; and finally, (iv) discretization makes many learning algorithms faster and more accurate. The dependent variable is discretized into nine categories using the following breakpoints:

Class No.	1	2	3	4	5	6	7	8	9
Range (Millions)	< 1 (Flop)	> 1 < 10	> 10 < 20	> 20 < 40	> 40 < 65	> 65 < 100	> 100 < 150	> 150 < 200	> 200 (Blockbuster)

Comparative Analysis of the Data Mining Methods

The researchers used a wide range of data mining methods including multi-layered perceptron (MLP) type neural networks, classification and regression type decision trees, multinomial logistic regression and discriminant analysis. The preliminary experiments showed that for this problem domain, two hidden layered MLP type neural networks consistently generated better prediction results than the other data mining methods. A graphical depiction of the aggregated comparison results are given in Fig. 10.1. For all model

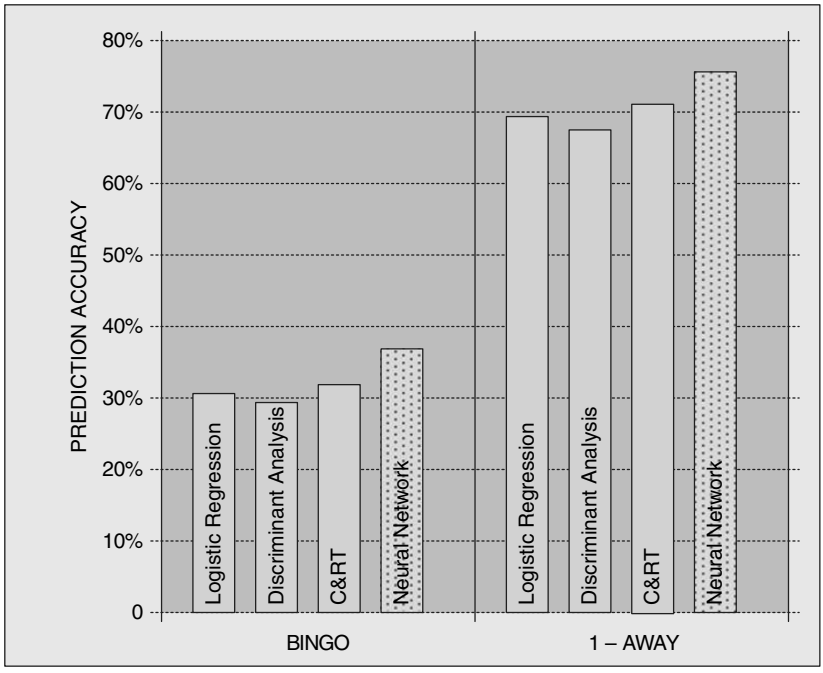
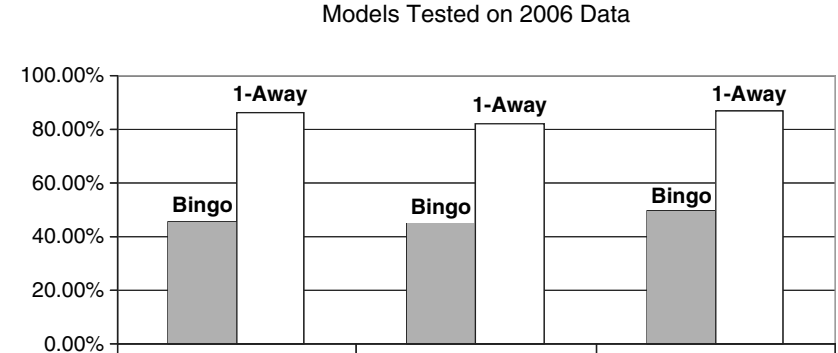


Fig. 10.1. Aggregated results for model comparison

development and testing experiments, the ten-fold cross validation methodology was used. Accuracy of a model is measured using two accuracy metrics. The first metric is the percent correct classification rate, which was also called "bingo". This metric was a reasonably conservative measure for accuracy in this scenario, and therefore the classification results within one category is also reported (this was called 1-away correct classification rate).

The results showed that the data mining methods can accurately predict the success category of a motion picture before its theatrical release with pinpoint accuracy (i.e., Bingo) with 36.9% and within one category with 75.2% accuracy. Recent experiments by Delen and Sharda showed much improved prediction results (see Fig. 10.2). In these latest results, they developed the prediction models using data from three different time windows (1998–2002, 2003–2005, and 1998–2005) and tested those models on 2006 data. The test results for all three data sets are given in Fig. 10.2. As shown, for the models developed using the largest dataset (all movies from 1998 through 2005) gave the best test results on 2006 movies, with close to 50% on the bingo and close to 87% on the within one category.

Compared to the limited number of previously reported studies, these prediction accuracies are significantly better. Compared to the other model types (i.e., logistic regression, discriminant analysis and classification and regression trees), using the exact same experimental conditions, neural networks performed significantly better. Because the neural network models built as part of this study are designed to predict the financial success of a movie before its theatrical release, they can be used as a powerful



	1998–2002	2003–2005	1998–2005
Bingo	45.82%	44.67%	49.86%
1-Away	86.46%	82.13%	86.74%

Fig. 10.2. Latest test results reported for the movie prediction project

decision aid by studios, distributors, and exhibitors. In any case, the results of this study are very intriguing, and once more, prove the continued value of data mining methods in addressing difficult prediction problems.

There are several directions in which this work was projected to evolve. The overall prediction accuracy can be enhanced by adding more relevant variables and by polling human expert opinions into a fused prediction. For instance, if a significant number of users indicate that a model can be improved by including an additional parameter to capture a specific characteristic of the problem (e.g., level of correlation between the movie script and the current social/political affairs), then the system should be receptive to such a proposal. Once in use by the decision makers, the forecasted results obtained over time can be stored and matched (synchronized/compared) with the actual box-office data (as soon as they become available) to check how good the forecasts (both individual model predictions and the fused ones) had been. Based on the results, the new data can be used to update the parameters of the predictions models, moving towards realizing the concept of living models. That is, data mining driven models and subsequent prediction systems should be deployed in environments where the models are constantly tested, validated and (as needed) modified (recalibrated). Such a dynamic (living and evolving) prediction systems are what the decision makers need, and what the data mining technology can provide.

Conclusions

We have discussed a variety of specialized tools available for data mining, and in this chapter described applications using these methods. The nature of data mining is to apply a variety of techniques, and the methods covered in this book describe some of the supplemental or alternative approaches to aid in the process of knowledge discovery. These examples provide a very brief taste for the variety of problems treated by data mining. They include business (workman's compensation; product warranty; credit scoring; product quality; customer segmentation), medical (breast cancer; Persian Gulf War syndrome), and financial (stock trading). Many other applications include scientific and engineering studies. Wherever there are masses of data, data mining is potentially useful.

The brief descriptions given here are meant to provide a taste for possibilities. The original sources are given, which can provide details for more in-depth study for interested readers. However, even this brief taste demonstrates the great potential for application of specialized techniques in data mining.

Bibliography

R. Agrawal, T. Izmielinski, A. Swami (1993). Database mining: A performance perspective, *IEEE Transactions on Knowledge and Data Engineering* 5:6, 914–925.

R. Agrawal, R. Srikant (1994). Fast algorithms for mining association rules, *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94),* Santiago, Chile, 487–499.

D. Aha, D. Kibler, M. Albert (1991). Instance-based learning algorithms, *Machine Learning* 6, 37–66.

M. Aizerman, E. Braverman, L. Rozonoer (1964). Theoretical foundations of the potential function method in pattern recognition learning*, Automation and Remote Control* 25, 821–837.

C. Apte, B. Liu, E.P.D. Pednault, P. Smyth (2002). Business applications of data mining, *Communications of the ACM* 45:8, 49–53.

W.-H. Au, K.C.C. Chan (2001). Classification with degree of membership: A fuzzy approach, *ICDM*, San Jose CA, IEEE, 35–42.

A. Ben David (1992). Automated generation of symbolic multiattribute ordinal knowledge-based DSSs: Methodology and applications, *Decision Sciences* 23:6, 157–1372.

H.K. Bhargava (1999). Data mining by decomposition: Adaptive search for hypothesis generation, *INFORMS Journal on Computing* 11:3, 239–247.

B.E. Boser, I.M. Guyon, V.N. Vapnik (1992). A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*. Pittsburgh, PA: ACM Press, 144–152.

L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software.

I. Bruha, P. Karlik, P. Berka (2000). Genetic learner: Discretization and fuzzification of numerical attributes, *Intelligent Data Analysis* 4, 445–460.

J. Buddhakulsomsiri, Y. Siradeghyan, A. Zakarian, X. Li (2006). Association rule-generation algorithm for mining automotive warranty data, *International Journal of Production Research* 44:14, 2749–2770.

C.J.C. Burges (1988). A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery* 2, 121–167.

C.H. Cai, W.-C. Fu, C.H. Cheng, W.W. Kwong (1998). Mining association rules with weighted items, *Proceedings of 1998 International Database Engineering and Applications Symposium*, Cardiff, Wales, Los Alamitos, CA, IEEE, 68–77.

K.C.C. Chan, W.-H. Au, B. Choi (2002). Mining fuzzy rules in a donor database for direct marketing by a charitable organization, *IEEE ICCI*, Los Alamitos CA, IEEE, 239–246.

R.F. Chang, C.N. Lu (2003). Load profile assignment of low voltage customers for power retail market applications, *IEEE Proceedings – Generation, Transmission, Distribution* 150:3, 263–267.

Y.-L. Chen, K. Tang, R.-J. Shen, Y.-H. Hu (2005). Market basket analysis in a multiple store environment, *Decision Support Systems* 40:2, 339–354.

C. Cortes, V. Vapnik (1995). Support-Vector networks, *Machine Learning* 20:3, 273–297.

N. Cristianini, J. Shawe-Taylor (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge, England: Cambridge University Press.

C. Da Cunha, B. Agard, A. Kusiak (2006). Data mining for improvement of product quality, *International Journal of Production Research* 44:18/19, 4041–4054.

M. d'Amato (2002). Appraising property with rough set theory, *Journal of Property Investment & Finance* 20:4, 406–418.

S. Daskalaki, I. Kopanas, M. Goudara, N. Avouris (2003). Data mining for decision support on customer insolvency in the telecommunications business, *European Journal of Operational Research* 145, 239–255.

R. Decker, K. Monien (2003). Market basket analysis with neural gas networks and self-organising maps, *Journal of Targeting, Measurement & Analysis for Marketing* 11:4, 373–386.

S.P. Deepa, K.G. Srinivasa, K.R. Venugopal, L.M. Patnaik (2005). Dynamic association rule mining using genetic algorithms, *Intelligent Data Analysis* 9:5, 439–453.

D. Delen, E. Sirakaya (2006). Determining the efficacy of data mining methods in predicting gaming ballot outcomes, *Journal of Hospitality and Tourism Research* 30:3, 313–332.

A.I. Dimitras, R. Slowinski, R. Susmaga, C. Zopounidis (1999). Business failure prediction using rough sets, *European Journal of Operational Research* 114, 263–280.

H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, V. Vapnik (1997). Support vector regression machines, *Advances in Neural Information Processing Systems* 9, NIPS, 155–161, Cambridge MA, MIT Press.

D. Dubois, E. Hullermeier, H. Prade (2006). A systematic approach to the assessment of fuzzy association rules, *Data Mining and Knowledge Discovery*, July, 1–26.

B. Efron (1979). Bootstrap methods: Another look at the jackknife, *The Annals of Statistics* 7:1, 1–26.

J.P. Egan (1975). *Signal Detection Theory and ROC Analysis, Series in Cognition and Perception*. New York: Academic Press.

T.P. Exarchos, C. Papaloukas, D.I. Fotiadis, L.K. Michalis (2006). An association rule mining-based methodology for automated detection of ischemic ECG beats, *IEEE Transactions on Biomedical Engineering* 53:8, 1531–1540.

J. Fibak, Z. Pawlak, K. Słowinski, R. Słowinski (1986). Rough sets based decision algorithm for treatment of duodenal ulcer by HSV, *Biological Sciences* 34, 227– 249.

W.J. Frawley, G. Piatetsky-Shapiro, C.J. Matheus (1991). Knowledge discovery in databases: An overview, *The AAAI Workshop on Knowledge Discovery in Databases*, 1–27.

Z. Fu, G.L. Golden, S. Lele, S. Raghavan, E. Wasil (2006). Diversification for better classification trees, *Computers & Operations Research* 33:11, 3185–3202.

R.L. Grossman, M.F. Hornick, G. Meyer (2002). Data mining standards initiatives, *Communications of the ACM* 45:8, 59–61.

G. Guo, D. Neagu (2005). Fuzzy *k*NNmodel applied to predictive toxicology data mining, *International Journal of Computational Intelligence and Applications* 5:3, 321–333.

A. Gyenesei (2000). Mining weighted association rules for fuzzy quantitative items, *Proceedings of PKDD Conference, September 13–16, 2000,* Lyon, France, Springer-Verlag, NY, 416–423.

J. Han, J. Pei, Y. Yin (2000). Mining frequent patterns without candidate generation, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data,* Dallas, Texas, United States, 1–12.

A.E. Hassanien (2003). Intelligent data analysis of breast cancer based on rough set theory, *International Journal on Artificial Intelligence Tools* 12:4, 479–493.

H. Havenstein (2006). IT efforts to help determine election successes, failures: Dems deploy data tools; GOP expands microtargeting use, *Computerworld* 40:45, 1, 16.

R. Hendler, F. Hendler (2004). Revenue management in fabulous Las Vegas: Combining customer relationship management and revenue management to maximize profitability, *Journal of Revenue & Pricing Management* 3:1, 73–79.

J. Hipp, U. Güntzer, G. Nakhaeizadeh (2000). Algorithms for association rule mining – a general survey and comparison, *SIGKDD Explorations* 2:1, 58–64.

F. Hoffmann, B. Baesens, C. Mues, T. Van Gester, J. Vanthienen (2007). Inferring descriptive and approximate fuzzy rules for credit scoring using evolutionary algorithms, *European Journal of Operational Research* 177:1, 540–555.

J.H. Holland (1992). *Adaptation in Natural and Artificial Systems*. Cambridge, MA: MIT Press.

T.-P. Hong, C.-S. Kuo, S.-C. Chi (2001). Trade-off between computation time and numbers of rules for fuzzy mining from quantitative data, *International Journal of Uncertainty, Fuzziness and Knowledge-based Systems* 9, 587–604.

T.-P. Hong, S.-L. Wang (2000). Determining appropriate membership functions to simplify fuzzy induction, *Intelligent Data Analysis* 4, 51–66.

T.-M. Huang, V. Kecman, I. Kopriva (2006). Kernel based algorithms for mining huge data sets, *Supervised, Semi-supervised, and Unsupervised Learning*. Heidelberg: Springer-Verlag.

C.-C. Huang, T.-L. Tseng, H.-F. Chuang, H.-F. Liang (2006). Rough-set-based approach to manufacturing process document retrieval, *International Journal of Production Research* 44:14, 2889–2911.

T. Imielinski, H. Mannila (1996). A database perspective on knowledge discovery, *Communications of the ACM* 39:11, 58–64.

H. Ince, T.B. Trafalis (2007). Kernel principal component analysis and support vector machines for stock price prediction, *IIE Transactions* 39, 629–637.

A. Kamrani, W. Rong, R. Gonzalez (2001). A genetic algorithm methodology for data mining and intelligent knowledge acquisition, *Computers & Industrial Engineering* 40:4, 361–377.

V. Kecman (2001). Learning and soft computing, *Support Vector Machines, Neural Networks, Fuzzy Logic Systems*. Cambridge, MA: The MIT Press.

E. Kim, W. Kim, Y. Lee (2003). Combination of multiple classifiers for the customer's purchase behavior prediction, *Decision Support Systems* 34:2, 167–175.

Y.-S. Kim, W.N. Street (2004). An intelligent system for customer targeting: A data mining approach, *Decision Support Systems* 37:2, 215–228.

M. Kitchener, M. Beynon, C. Harrington (2004). Explaining the diffusion of Medicaid Home Care Waiver Programs using VPRS decision rules, *Health Care Management Science* 7:3, 237–244.

R. Kohavi (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection, *The Fourth International Joint Conference on Artificial Intelligence*. Montreal, Quebec, Canada: American Association for Artificial Intelligence (AAAI), 202–209.

C. Kuok, A. Fu, H. Wong (1998). Mining fuzzy association rules in databases, *ACM SIGMOD Record* 27, 41–46.

R. Ladner, F.E. Petry, M.A. Cobb (2003). Fuzzy set approaches to spatial data mining of association rules, *Transactions in GIS* 7, 123–138.

O.I. Larichev, H.M. Moshkovich (1994). An approach to ordinal classification problems, *International Transactions on Operations Research* 82, 503–521.

W.-J. Lee, S.-J. Lee (2004). Discovery of fuzzy temporal association rules, *IEEE Transactions on Systems, Man, and Cybernetics – Part B* 34:6, 2330–2342.

R.S.T. Lee, J.N.K. Liu (2004). IJADE Web-Miner: An intelligent agent framework for internet shopping, *IEEE Transactions on Knowledge & Data Engineering* 16:4, 461–473.

G.S. Linoff (2004). Survival data mining for customer insight, *Intelligent Enterprise* 7:12, 28–33.

B. Liu, H.-Y. Lee (1999). Finding interesting patterns using user expectations, *IEEE Transactions on Knowledge & Data Engineering* 11:6, 817–832.

S. Lopes, J.-M. Petit, L. Lakhal (2002). Functional and approximate dependency mining: Database and FCA points of view, *Journal of Experimental Theoretical Artificial Intelligence* 14, 93–114.

G. Loveman (2003). Diamonds in the data mine, *Harvard Business Review* 81:5, 109–113.

C. Lucchese, S. Orlando, R. Perego (2006). Fast and memory efficient mining of frequent closed itemsets, *IEEE Transactions on Knowledge & Data Engineering* 18:1, 21–36.

H. Lu, R. Sentiono, H. Liu (1996). "Effective Data Mining Using Neural Networks." *IEEE Transactions on Knowledge and Data Engineering*, 8(6): 957–961.

S.-F. Lu, H. Hu, F. Li (2001). Mining weighted association rules, *Intelligent Data Analysis* 5, 211–225.

C.J. Matheus, P.K. Chan, G. Piatetsky-Shapiro (1993). "Systems for Knowledge Discovery in Databases." *IEEE Transactions on Knowledge and Data Engineering*, 5(6): 903–913.

K. Mehta, S. Bhattacharyya (2004). Adequacy of training data for evolutionary mining of trading rules, *Decision Support Systems* 37:4, 461–474.

C.J. Merz, P.M. Murphy. *UCI Repository of Machine Learning Databases.* Irvine, CA: University of California, Department of Information and Computer Science, http://www.ics.uci.edu/~mlearn/MLOther.html.

W. Michalowski, S. Rubin, R. Slowinski, S. Wilk (2003). Mobile clinical support system for pediatric emergencies, *Decision Support Systems* 36:2, 161–176.

A. Mild, T. Reutterer (2003). An improved collaborative filtering approach for predicting cross-category purchases based on binary market basket data, *Journal of Retailing & Consumer Services* 10:3, 123–133.

H.M. Moshkovich, A.I. Mechitov, D. Olson (2002). Rule induction in data mining: Effect of ordinal scales, *Expert Systems with Applications* 22, 303–311.

R. Nayak, T. Qiu (2005). A data mining application: Analysis of problems occurring during a software project development process, *International Journal of Software Engineering* 15:4, 647–663.

M. O'Connell (2006). Drug safety, the U.S. Food and Drug Administration and statistical data mining, *Scientific Computing* 23:7, 32–33.

D.L. Olson, H.M. Moshkovich, A.I. Mechitov (2007). An experiment with Fuzzy sets in data mining, *International Conference on Computational Science*. Beijing, LNCS 4488, Heidelberg, Springer, 27–29.

D.L. Olson, Y. Shi (2007). *Introduction to Business Data Mining*. Boston: McGraw-Hill/Irwin.

T. Öyzer, R. Alhajj, K. Barker (2007). Intrusion detection by integrating boosting genetic fuzzy classifier and data mining criteria for rule pre-screening, *Journal of Network & Computer Applications* 30:1, 99–113.

Z. Pawlak (1982). Rough sets, *International Journal of Information and Computer Sciences* 11, 341–356.

Z. Pawlak, (1991). *Rough Set: Theoretical Aspects and Reasoning About Data*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Z. Pawlak (2000). Rough sets and decision analysis, *INFOR* 38:3, 132–144.

Z. Pawlak, R. Słowinski (1994). Decision analysis using rough sets, *International Transactions of Operational Research* 1:1, 107–114.

J. Pearl (1988). Probabilistic reasoning in intelligent systems, *Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann.

R. Reed (1995). Household ethnicity, household consumption: Commodities and the Guarani, *Economic Development & Cultural Change* 44:1, 129–145.

R.M. Rejesus, B.B. Little, A.C. Lovell (2004). Using data mining to detect crop insurance fraud: Is there a role for social scientists? *Journal of Financial Crime* 12:1, 24–32.

T.G. Roche (2006). Expect increased adoption rates of certain types of EHRs, EMRs, *Managed Healthcare Executive* 16:4, 58.

W. Rodriguez, M. Last, A. Kandel, H. Bunke (2001). Geometric approach to data mining, *International Journal of Image & Graphics* 2, 363–386.

G.J. Russell, A. Petersen (2000). Analysis of cross category dependence in market basket selection, *Journal of Retailing* 78:3, 367–392.

S. Sarawagi, S. Thomas, R. Agrawal (1998). Integrating association rule mining with relational database systems: Alternatives and implications, *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data,* Seattle, WA, United States, ACM, New York, 343–354.

B. Schölkopf, C.J.C. Burges, A.J. Smola (1999). *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA: MIT Press.

B. Schölkopf, A.J. Smola (2002). *Learning with Kernelss.* Cambridge, MA: MIT Press.

R. Sentiono, H. Liu (1996). "Symbolic Representation of Neural Networks. *IEEE Computer*, 29(3): 71–77.

J. Shawe-Taylor, N. Cristianini (2004). *Kernel Methods for Pattern Analysis.* Cambridge, England, Cambridge University Press.

L. Shen, H.T. Loh (2004). Applying rough sets to market timing decisions, *Decision Support Systems* 37:4, 583–597.

Y. Shi, Y. Peng, G. Kou, Z. Chen (2005). Classifying credit card accounts for business intelligence and decision making: A multiple-criteria quadratic programming approach, *International Journal of Information Technology & Decision Making* 4:4, 581–599.

W. Shitong, K.F.L. Chung, S. Hongbin (2005). Fuzzy taxonomy, quantitative database and mining generalized association rules, *Intelligent Data Analysis* 9, 207–217.

E. Simoudis (1996). Reality check for data mining, *IEEE Expert Intelligent Systems & Their Applications* 11:5, 26–33.

N.D.J. Singpurwalla, M. Booker (2004). Membership function and probability measures of fuzzy sets, *Journal of the American Statistical Association* 9, 867–877.

E. Sirakaya, D. Delen, H. Choi (2005). Forecasting gaming referenda, *Annals of Tourism Research* 32:1, 127–149.

R. Slowinski (1995). Rough set approach to decision analysis, *AI Expert*, 10:3, 19–25.

R. Slowinski, C. Zopounidis (1995). Application of the rough set approach to evaluation of bankruptcy risk, *International Journal of Intelligent Systems in Accounting, Finance and Management* 4, 27–41.

K. Sorensen, G.K. Janssens (2003). Data mining with genetic algorithms on binary trees, *European Journal of Operational Research* 151, 253–264.

D. Souliou, A. Pagourtzis, N. Drosinos, P. Tsanakas (2006). Computing frequent itemsets in parallel using partial support trees, *Journal of Systems & Software* 79:12, 1735–1743.

R. Srikant, R. Agrawal (1996). Mining quantitative association rules in large relational tables, *The 1996 ACM SIGMOD International Conference on Management of Data.* Montreal, Canada, ACM, New York, 1–12.

C. Stanfill, D.L. Waltz (1986). Toward memory-based reasoning, *Communications of the ACM* 29:12, 1213–1223.

N. Swartz (2004). IBM, Mayo clinic to mine medical data, *The Information Management Journal* 38:6, 8.

J. Swets (1988). Measuring the accuracy of diagnostic systems, *Science* 240, 1285–1293.

C.M. Tam, T.K.L. Tong, K.K. Chan (2006). Rough set theory for distilling construction safety measures, *Construction Management & Economics* 24:11, 1199–1206.

P.J. Tan, D.L. Dowe (2004). MML Inference of oblique decision trees, *Lecture Notes in Artificial Intelligence (LNAI) 3339*. Heidelberg, Springer-Verlag, 1082–1088.

S. Thelen, S. Mottner, B. Berman (2004). Data mining: On the trail to marketing gold, *Business Horizons* 47:6, 25–32.

A.-C. Tsoi, S. Zhang, M. Hagenbuchner (2005). Pattern discovery on Australian medical claims data – a systematic approach, *IEEE Transactions on Knowledge & Data Engineering* 17:10, 1420–1435.

S. Tsumoto (2004). Mining diagnostic rules from clinical databases using rough sets and medical diagnostic model, *Information Sciences* 162:2, 65–80.

P. Vaitchev, R. Missaoui, R. Godin, M. Meridji (2002). Generating frequent itemsets incrementally: Two novel approaches based on Galois lattice theory, *Journal of Experimental & Theoretical Artificial Intelligence* 14:2/3, 115–142.

V. Vapnik (1999). *The Nature of Statistical Learning Theory*. Heidelberg: Springer-Verlag.

C.M. van der Walt, E. Barnard (2006). Data characteristics that determine classifier performance, In *Proceedings of the Sixteenth Annual Symposium of the Pattern Recognition Association of South Africa*, 160–165. Available [Online] http://www.patternrecognition.co.za

D. Waltz, S. Kasif (1995). On reasoning from data, *ACM Computing Surveys* 27:3, 356–359.

S.M. Weiss, B.F. White, C.V. Apte, F.J. Damerau (2000). Lightweight document matching for help-desk applications, *IEEE Expert Intelligent Systems & Their Applications* 15:2, 57–61.

S.-S. Weng, R.-K. Chiu, B.-J. Wang, S.-H. Su (2006/2007). The study and verification of mathematical modeling for customer purchasing behavior, *Journal of Computer Information Systems* 47:2, 46–57.

R. Whiting (1999). Oracle boosts data analysis, *Informationweek* 738, 14 June, 30.

R.R. Yager (1991). On linguistic summaries of data. In G. Piatetsky-Shapiro, W.J. Frawley, editors, *Knowledge Discovery in Databases.* Menlo Park, CA: AAAI/MIT Press, 347–363.

X. Yan, S. Zhang, C. Zhangt (2007). On data structures for association rule discovery, *Applied Artificial Intelligence* 21:2, 57–79.

L.A. Zadeh (1965). Fuzzy sets. *Information and Control* 8, 338–356.

L.-Y. Zhai, L.-P. Khoo, S.-C. Fok (2002). Feature extraction using rough set theory and genetic algorithms – an application for the simplification of product quality evaluation, *Computers & Industrial Engineering* 43, 661–676.

P. Zhang (1992). On the distributional properties of model selection criteria, *Journal of the American Statistical Association* 87:419, 732–737.

D. Zhang, L. Zhou (2004). Discovering golden nuggets: Data mining in financial application. *IEEE Transactions on Systems, Man & Cybernetics: Part C* 34:4, 513–522.

K.-G. Zhao (1989). Set pair analysis – a new concept and new systematic approach, *Proceedings of National System Theory and Regional Analysis Conference*. Baotou (In Chinese).

K.-G. Zhao (2000). *Set Pair Analysis and its Preliminary Application*. Zhejiang: Science and Technology Press (In Chinese).

H. Zhao (2007). A multi-objective genetic programming approach to developing Pareto optimal decision trees. *Decision Support Systems* 43:3, 809–826.

J. Zurada, W. Karwowski, W. Marras (2004). Classification of jobs with risk of low back disorders by applying data mining techniques, *Occupational Ergonomics* 4:4, 291–305.

J.M. Zurada, A.S. Levitan, J. Guan (2006). Non-conventional approaches to property value assessment, *Journal of Applied Business Research* 22:3, 1–14.

J. Zurada, S. Lonial (2005). Comparison of the performance of several data mining methods for bad debt recovery in the healthcare industry, *Journal of Applied Business Research* 21:2, 37–54.

___ (2007). Data mining: Early attention to privacy in developing a key DHS program could reduce risks, *GAO Report 07-293*, 3/21/2007.

Index

Accuracy, 16, 17, 19, 20, 23, 26, 28, 32, 33, 64, 65, 68, 71, 73, 76, 77, 84, 86, 100, 103, 105–109, 111, 118, 120, 137–143, 145, 146, 152, 153, 166, 167
Actionable, 7
Affinity positioning, 11
AI, 5, 77
Allele, 125
Analysis of results, 23, 26–28
Approximation, 87, 90, 91, 97, 156, 159
A priori algorithm, 53
Artificial intelligence, *see* AI
Association, 16, 53, 54, 57, 61
Association rule, 5, 10, 16, 25, 26, 28, 50, 53–69, 79–81, 83–86, 125
Association rule mining., 26, 55, 68, 85, 153–155
Automotive industry, 153

Banking, 3, 7
Bellagio, 6
Binary, 13–16, 45, 54, 61, 71–75, 79, 87, 125, 139, 151, 152, 156, 160, 161, 164
Binary classification, 151, 152
Binning, 96
Boolean reasoning, 96, 97
Bootstrapping, 143, 144
Brio Technology Inc, 8
Business understanding, 9–11, 27, 28, 34

CART, 1
Case-based reasoning, 40
Casino, 6, 7, 103–105, 108
Category, 13, 14, 29–31, 50, 81, 82, 88, 107, 118, 159, 161, 164, 166
Churn, 4, 7
Classification, 5, 16, 17, 25, 34, 39, 48, 70–86, 89, 93, 97, 100–102, 106, 111, 112, 114–118, 122, 125, 137–146, 151–153, 156, 158, 159, 165, 166
Classification error, 31, 158
CLEMENTINE, 8, 13, 93, 155
Cluster analysis, 9, 10, 14, 15, 17, 68
Clustering, 16, 17, 20, 117, 131, 154, 155
Coincidence matrix, 31–34, 42, 47, 137, 139, 145
Confidence, 16, 23, 25–28, 55, 59, 62, 72, 77, 79, 80, 83, 84, 140, 143, 154
Confusion matrice, *see* Coincidence matrix
Continuous variables, 40, 50, 126
Correlation analysis, 6
Credit card, 3, 4, 7, 12, 13, 71, 73, 123
Credit screening, 158
CRISP, 9–18, 27, 28, 34
CRM, *see* Customer relationship management
Cross-Industry Standard Process for Data Mining, 9
Crossover, 125, 126, 129, 130, 161
Cross-selling, 4, 7
Cross validation, 26, 95, 103, 105, 106, 108, 119–121, 141–143, 147, 152, 166

Customer relationship management, 6, 11, 131, 158, 160, 167
Customer targeting, 159, 160

Data collection, 9, 23, 103
Data exploration, 5, 9, 53, 68
Data Miner, 93, 151
Data mining techniques, 16, 50, 71, 78, 109, 132, 151, 155
Data modeling, 9, 15, 21, 23, 25, 27
Data preparation, 9, 10, 12, 14, 27–29, 34
Data processing, 9, 14
Data structure, 54, 125
Data transformation, 6, 9, 14, 23, 24, 27, 28, 34, 109
Data treatment, 16, 151
Data understanding, 9–11, 27, 29, 34
Data warehouse, 6, 8, 29, 56
Decision support system, 119, 162, 163
Decision tree, 4, 5, 14–17, 21, 31, 33, 39, 69, 71–74, 93, 103–109, 125, 131, 151, 152, 155–157, 165
Demographic data, 12, 104, 161
Deployment, 10, 18, 19, 27, 28, 34, 92
Discretization, 78, 93, 94, 96, 100, 109, 131, 158, 159, 164, 165
Discriminant analysis, 15, 31–33, 117, 165, 166
Discriminant regression, 17
Dynamic reduct, 98, 100

Ensemble model, 151, 152
Enterprise Miner, 8, 93, 152
Entropy, 17, 96
Evaluation, 10, 16, 18, 25, 27, 28, 31, 34, 99, 103, 107, 108, 131
Evaluation set, *see* Test set
Evidence theory *see* Knowledge discovery

False positive, 105, 107, 137, 138, 145, 146
Filtering, 12, 60, 99, 100
Fraud, 4, 7, 13, 29, 53
Fuzzy association rules, 69, 79–86
Fuzzy data mining, 5, 80, 155
Fuzzy sets, 69–86

Gaussian kernel, 120–122
Gaussian radial basis function, 115, 116, 121
Genetic algorithm, 5, 98, 99, 109, 125–132, 158–162
Granularity, 88
Grobian, 93

Hamming distance, 40, 41
Harrah's Entertainment Inc., 6
Holdout, 105, 107, 140–143
Hollywood, 162–165
Hyperplane, 111–116, 121
Hypothesis development, 4
Hypothesis testing, 105

IBM, 3, 8
Imprecision, 69, 88
Improvement, 33, 59, 62, 73, 75, 122, 140, 162
Imputation, 12
Indiscernibility, 87, 89, 98
Information system, 8, 41, 42, 45, 48, 51, 88–90, 94, 97
Insurance, 3, 4, 7, 13, 28, 53, 160, 161

Jackknifing, 143, 144

KDNuggets, 8, 21, 22, 50
Kernel, 111, 115, 118–122, 157
Kernel trick, 117
K-fold cross validation, 95, 141–143, 147
Knowledge discovery, 4, 10, 22, 28, 53–68, 79, 93, 103, 111, 167

Leave-one-out methodology, 143
Lift, 7, 151, 152, 160
Link analysis, 25
Logistic regression, 5, 15, 151, 152, 165, 166
Lower back disorder, 151, 152

Machine learning, 8, 93, 96, 103, 111, 114, 117, 153, 157
Mandalay Bay, 6
Market-basket analysis, 4, 55–59
Maximum margin classifier, 112, 115
Mayo Clinic, 3
Medical research, 161
Megaputer Intelligence, 50
Memory-based application, 151–153
Memory-based reasoning, 21, 39–50, 52, 152, 153
Microsoft, 8
Minimum confidence, 55, 59, 62, 79
Minimum improvement, 59, 62
Minimum support, 26, 28, 55, 58, 60–62, 72, 79, 83, 154
Misclassification costs., 32, 33
Motion picture industry, 162, 163
Mutation, 125, 126, 128–130, 161

Naïve Bayes, 102
NASDAQ, 157
National Institute of Occupational Safety and Health, 152, 153
Neural network, 5, 8, 15, 17, 21, 31, 33, 39, 103, 105, 107–109, 111, 119, 121, 122, 140, 151, 152, 157–160, 165, 166
Nominal data, 12, 45, 72

Oracle, 8, 50
Ordinal data, 12, 44, 71, 72
Overfitting, 120

Performance evaluation, 137–146
Persian Gulf War illness, 161
PolyAnalyst, 8, 14, 50, 61–66, 68, 71, 155
Prediction, 5, 11, 25, 29, 30, 42, 50, 85, 91, 95, 100, 103, 105–109, 116, 117, 120, 123
Predictive model, 55, 93, 104, 108, 109, 159, 160
Predictive modeling, 93, 108, 109, 137–146
Principal components analysis, 117
Product quality testing, 158, 159

Qualitative data, 12
Quality testing, 158, 159
Quantitative data, 12

Radial basis function, 115, 116, 119, 121
Receiver operating characteristics, *see* ROC
Reduct, 93, 94, 97–100, 156
Regression, 5, 14–17, 39, 111, 116, 122, 131, 142, 143, 151, 152, 165, 166
Resolution, 87
Retailing, 7, 55, 56
ROC, 100, 138, 144–147
ROSETTA, 93
Rough set, 5, 21, 69, 70, 87–109, 155–159
Rough set exploration system, 93
Rough set information system, 88–91, 93, 94
Rough set theory, 69, 70, 87, 93, 156, 158, 159

SAS Institute Inc., 8
Scalable, 6
See5, 71–76, 79, 86, 155
SEMMA, 9, 19–22, 27
Sigmoid, 116, 119, 120
SQL, 54–56, 60–63, 68
Standard voting, 101, 102

Stock price forecasting, 157
Structured query language, *see* SQL
Supervised learning, 111, 140
Support, 5, 8, 11, 16, 21–28, 39, 50, 55, 58–62, 68, 72, 77, 79–83, 92, 93, 97–102, 125–131, 134, 154, 162, 163
Support vector machine(SVM) 111–123, 157–159

Targeting, 4
Telecommunication, 3, 7, 22, 28
Telemarketing, 74
Telephone, 4, 30, 32–34
Test set, 10, 16, 27, 28, 31–34, 42, 46, 47, 72, 134, 140, 145, 154, 155, 160
Text analyst, 25–28
Text mining., 24, 25, 27, 50, 111
Textual data, 39
Training observations, 42, 152
Training set, 16, 26, 31, 33, 41–43, 46, 49, 72, 78, 114, 120, 127, 140, 144, 155, 157, 160, 161
Transactional data, 12
True positive rate, 138, 145, 146
Trump's, 6

U.S. election, 3
UCI Machine Learning Repository, *see* Knowledge discovery
Uncertainty, 5, 69, 79, 86, 88, 163

Vagueness, 69, 88
Validation set, 16, 103, 108, 140

Warranties, 153
Warranty claims, 153, 154
WEKA, 8