Chapman & Hall/CRC
Data Mining and Knowledge Discovery Series

RapidMiner

Data Mining Use Cases and Business Analytics Applications

Edited by

Markus Hofmann
Ralf Klinkenberg

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Contents

Foreword

Case Studies Are for Communication and Collaboration

Data mining or data analysis in general has become more and more important, since large amounts of data are available and open up new opportunities for enhanced empirical sciences, planning and control, and targeted marketing and information services. Fortunately, theoretically well-based methods of data analysis and their algorithmic implementations are available. Experienced analysts put these programs to good use in a broad variety of applications. However, the successful application of algorithms is an art! There is no mapping from application tasks to algorithms, which could determine the appropriate chain of operators leading from the given data to the desired result of the analysis—but there are examples of such processes. Case studies are an easy way of communicating smart application design. This book is about such case studies in the field of data analysis.

Analysts are interested in the work of others and curiously inspect new solutions in order to stay up to date. Case studies are an excellent means to communicate best-practice compositions of methods to form a successful process. Cases are also well-suited to storing results for further use. A case is then used as a blueprint for further applications[1]. This eases the development of an application to quite some extent.

Another good use of case studies is to ease the communication between application domain experts and data mining specialists. The case shows what could already be achieved and inspire future cases[2]. This allows to frame new applications and to illustrate possible results. For those experts who want to set up a data mining project, it is a valuable justification.

Finally, teaching means communication. Teaching data mining is not complete without reference to case studies, either. Rapid-I offers at their website, `http://rapid-i.com`, video tutorials (webinars), white papers, manuals—a large variety of documentation with many illustrations. Offering case studies is now a further step into communicating not only the facilities of the system, but its use in real-world applications. The details of complex data analysis processes help those who want to become data analysts.

In summary, case studies support the collaboration of data analysts among themselves, the communication of data analysts and application experts, the interaction between experienced and beginners. Now, how can complex data mining processes be communicated, exchanged, and illustrated? An easy-to-understand view of the process is to abstract away the programming details. As is explained in the following, RapidMiner offers this.

[1] T. Euler. Publishing operational models of data mining case studies. In B. Kitts, G. Melli, and K. Rexer, editors, *Procs. of the 1st Workshop on Data Mining Case Studies*, held at IEEE ICDM, pages 99-106, 2005.

[2] G. Melli, X Wu, P. Beinat, F. Bonchi, L. Cao, Rong Dan, C. Faloutsos, B. Kitts, B. Goethals, G. McLachlan, J. Pei, A. Srivastava, and O. Zaiane. Top10 data mining case studies. *Int. J. Information Technology and Decision Making*, 11(2):389-400, 2012.

RapidMiner

RapidMiner is a system which supports the design and documentation of an overall data mining process. It offers not only an almost comprehensive set of operators, but also structures that express the control flow of the process.

- Nesting operator chains were characteristic of RapidMiner (Yale) from the very beginning. This allows us to have a small number of operators at one level, each being expanded at the level below by simply clicking on the lower right area of the operator box.
- An example set can be multiplied for different processes that are executed in parallel and then be unified again. Sets of rows of different example sets of the same set of attributes can be appended. Hence, the example set that is used by some learning method can flexibly be modified.
- The cross validation is one of the most popular nested operators. The training set is split into n parts and, in a loop, $n-1$ parts are put to training and the remaining part to testing, so that the performance on the test set can be averaged over a range of different example sets from the same domain. The operator X-Validation is used in most of the case studies in order to achieve sensible performance evaluations.
- Several loop operators can be specified for an application. The Loop Parameters operator repeatedly executes some other operators. The parameters of the inner operators as well as conditions controlling the loop itself tailor the operator to the desired control flow.
- Wrapper approaches wrap a performance-based selection around a learning operator. For instance, those feature sets are selected for which a learner's performance is best. The wrapper must implement some search strategy for iterating over sets and for each set call a learning algorithm and its evaluation in terms of some performance measure.

These structures are similar to notions of programming languages, but no programming is necessary – it is just drag, drop, and click! Visually, the structures are shown by boxes which are linked or nested. This presentation is easy to understand.

Only a small (though decisive) part of an overall data mining process is about model building. Evaluating and visualizing the results is the concluding part. The largest part is the pre-processing.

- It starts with reading in the data and declaring the meta data. RapidMiner supports many data formats and offers operators for assigning not only value domains of attributes (attribute type), but also their role in the learning process.
- The inspection of the data through diverse plots is an important step in developing the case at hand. In many case studies, this step is not recorded, since after the exploration it is no longer necessary. The understanding of the analysis task and the data leads to the successful process.
- Operators that change the given representation are important to bridge the gap between the given data and the input format that a learner needs. Most analysts have a favorite learning method and tweak the given data until they suit this algorithm well. If frequent set mining is the favorite, the analyst will transform m nominal values of

one attribute into m binomial attributes so that frequent set mining can be applied. If the attribute type requirements of a learner are not yet fulfilled, RapidMiner proposes fixes.

- The discretization of real-valued attributes into nominal- or binomial-valued attributes is more complex and, hence, RapidMiner offers a variety of operators for this task.
- Beyond type requirements, features extraction and construction allow learners to find interesting information in data which otherwise would be hidden. A very large collection of operators offers the transformation of representations. The text processing plug-in, the value series plug-in, and the image processing plug-in are specifically made for the pre-processing of texts, time series or value series in general, and images.
- The feature selection plug-in automatically applies user-specified criteria to design the best feature set for a learning task. Moreover, it evaluates the selected features with respect to stability. For real-world applications, it is important that good performance is achieved at any sample of data. It is not sufficient that the selected features allow a good performance on average in the cross-validation runs, but it must be guaranteed that the features allow a sufficiently good performance on every data sample.

Given the long operator chains and nested processes in data mining, the aspect of documentation becomes indispensable. The chosen parameters of, e.g., discretization, the particular feature transformations, and the criteria of feature selection are stored with the RapidMiner process. The metadata characterize the data at any state in the process. Hence, its result is explainable and reproducible.

In this book, case studies communicate how to analyze databases, text collections, and image data. The favorite learning tasks are classification and regression with the favorite learning method being support vector machines followed by decision trees. How the given data are transformed to meet the requirements of the method is illustrated by pictures of RapidMiner. The RapidMiner processes and datasets described in the case studies are published on the companion web page of this book. The inspiring applications may be used as a blueprint and a justification of future applications.

Prof. Dr. Katharina Morik *(TU Dortmund, Germany)*

Preface

Data and the ability to make the best use of it are becoming more and more crucial for today's and tomorrow's companies, organizations, governments, scientists, and societies to tackle everyday challenges as well as complex problems and to stay competitive. *Data mining*, *predictive analytics*, and *business analytics* leverage these data, provide unprecedented insights, enable better-informed decisions, deliver forecasts, and help to solve increasingly complex problems. Companies and organizations collect growing amounts of data from all kinds of internal and external sources and become more and more data-driven. Powerful tools for mastering data analytics and the know-how to use them are essential to not fall behind, but to gain competitive advantages, and to increase insights, effectiveness, efficiency, growth, and profitability.

This book provides an *introduction to data mining and business analytics*, to the most powerful and flexible open source software solutions for data mining and business analytics, namely *RapidMiner and RapidAnalytics*, and to many application *use cases* in scientific research, medicine, industry, commerce, and diverse other sectors. RapidMiner and RapidAnalytics and their extensions used in this book are all freely available as open source software community editions and can be downloaded from
`http://www.RapidMiner.com`

Each chapter of this book describes an application *use case*, how to approach it with *data mining methods*, and how to implement it with RapidMiner and RapidAnalytics. These application-oriented chapters do not only provide you with the necessary analytics know-how to solve these problems and tasks, but also with easy-to-follow reproducible step-by-step descriptions for accomplishing this with RapidMiner and RapidAnalytics. The datasets and RapidMiner processes used in this book are available from the companion web page of this book:
`http://www.RapidMinerBook.com`

This application-oriented analytics use case collection will quickly enable you to solve similar problems effectively yourself. The case studies can serve as blue prints for your own data mining applications.

What Is Data Mining? What Is It Good for, What Are Its Applications, and What Does It Enable Me to Do?

While technology enables us to capture and store ever larger quantities of *data*, finding relevant *information* like underlying patterns, trends, anomalies, and outliers in the data and summarizing them with simple understandable and robust quantitative and qualitative models is a grand challenge. Data mining helps to discover underlying structures in the data, to turn *data* into *information*, and information into *knowledge*. Emerged from mathematics,

statistics, logic, computer science, and information theory, data mining and machine learning and statistical learning theory now provide a solid theoretical foundation and powerful methods to master this challenge. *Data mining* is the extraction of implicit, previously unknown, and potentially useful information from data. The automatically extracted models provide insight into customer behavior and into processes generating data, but can also be applied to, for example, automatically classify objects or documents or images into given categories, to estimate numerical target variables, to predict future values of observed time series data, to recommend products, to prevent customer churn, to optimize direct marketing campaigns, to forecast and reduce credit risk, to predict and prevent machine failures before they occur, to automatically route e-mail messages based on their content and to automatically detect e-mail spam, and to many other tasks where data helps to make better decisions or even to automate decisions and processes. Data mining can be applied not only to structured data from files and databases, but text mining extends the applicability of these techniques to unstructured data like texts from documents, news, customer feedback, e-mails, web pages, Internet discussion groups, and social media. Image mining, audio mining, and video mining apply these techniques to even further types of data.

Why Should I read This Book? Why Case Studies? What Will I learn? What Will I Be Able to Achieve?

This book introduces the most important machine learning algorithms and data mining techniques and enables you to use them in real-world applications. The open source software solutions RapidMiner and RapidAnalytics provide implementations for all of these algorithms and a powerful and flexible framework for their application to all kinds analytics tasks. The book and these software tools cover all relevant steps of the data mining process from data loading, transformation, integration, aggregation, and visualization via modeling, model validation, performance evaluation, model application and deployment, to automated feature selection, automated parameter and process optimization, and integration with other tools like, for example, the open source statistics package R or into your IT infrastructure via web services. The book and the tools also extensively cover the analysis of unstructured data including text mining and image mining.

The application-oriented focus of this book and the included use cases provide you with the know-how and blueprints to quickly learn how to efficiently design data mining processes and how to successfully apply them to real-world tasks. The book not only introduces you to important machine learning methods for tasks like clustering, classification, regression, association and recommendation rule generation, outlier and anomaly detection, but also to the data preprocessing and transformation techniques, which often are at least as crucial for success in real-world applications with customer data, product data, sales data, transactional data, medical data, chemical molecule structure data, textual data, web page data, image data, etc. The use cases in this book cover domains like retail, banking, marketing, communication, education, security, medicine, physics, and chemistry and tasks like direct marketing campaign optimization, product affinity scoring, customer churn prediction and prevention, automated product recommendation, increasing sales volume and profits by cross-selling, automated video lecture recommendation, intrusion detection, fraud detection, credit approval, automated text classification, e-mail and mobile phone text message spam detection, automated language identification, customer feedback and hotel review analysis,

image classification, image feature extraction, automated feature selection, clustering students in higher education and automated study program recommendation, ranking school applicants, teaching assistant evaluation, pharmaceutical molecule activity prediction, medical research, biochemical research, neutrino physics research, and data mining research.

What Are the Advantages of the Open Source Solutions RapidMiner and RapidAnalytics Used in This Book?

RapidMiner and RapidAnalytics provide an integrated environment for all steps of the data mining process, an easy-to-use graphical user interface (GUI) for the interactive data mining process design, data and results visualization, validation and optimization of these processes, and for their automated deployment and possible integration into more complex systems. RapidMiner enables one to design data mining processes by simple drag and drop of boxes representing functional modules called operators into the process, to define data flows by simply connecting these boxes, to define even complex and nested control flows, and all without programming. While you can seamlessly integrate, for example, R scripts or Groovy scripts into these processes, you do not need to write any scripts, if you do not want to. RapidMiner stores the data mining processes in a machine-readable XML format, which is directly executable in RapidMiner with the click of a button, and which along with the graphical visualization of the data mining process and the data flow serves as an automatically generated documentation of the data mining process, makes it easy to execute, to validate, to automatically optimize, to reproduce, and to automate.

Their broad functionality for all steps of the data mining process and their flexibility make RapidMiner and RapidAnalytics the tools of choice. They optimally support all steps of the overall data mining process and the flexible deployment of the processes and results within their framework and also integrated into other solutions via web services, Java API, or command-line calls. The process view of data mining eases the application to complex real-world tasks and the structuring and automation of even complex highly nested data mining processes. The processes also serve as documentation and for the reproducibility and reusability of scientific results as well as business applications. The open source nature of RapidMiner and RapidAnalytics, their numerous import and export and web service interfaces, and their openness, flexibility, and extendibility by custom extensions, operators, and scripts make them the ideal solutions for scientific, industrial, and business applications. Being able to reproduce earlier results, to reuse previous processes, to modify and adapt them or to extend them with customized or self-developed extensions means a high value for research, educational training, and industrial and business applications. RapidMiner allows you to quickly build working prototypes and also quickly deploy them on real data of all types including files, databases, time series data, text data, web pages, social media, image data, audio data, web services, and many other data sources.

What Is the Structure of This Book and Which Chapters Should I Read?

The *first chapter* of this book introduces the *basic concepts of data mining and machine learning*, common terms used in the field and throughout this book, and the decision tree modeling technique as a machine learning technique for classification tasks. The *second chapter* gives you an introductory tour through the *RapidMiner graphical user interface (GUI)* and how to use it to define data mining processes. In case you are already familiar with data mining and RapidMiner, you can skip these two chapters. However, if you are a novice in the field or regarding the software, these first two chapters are highly recommended and will give you a quick start in both data mining and RapidMiner. All following chapters provide a use case each and introduce additional data mining concepts and RapidMiner operators needed to solve the task at hand.

The *Chapters 3 to 6* describe *classification use cases* and introduce the *k-nearest neighbors (k-NN)* and *Naïve Bayes learning algorithms*. *Chapter 3* applies k-NN for the *evaluation of teaching assistants*. In *Chapter 4* k-NN is used to *classify different glass types based on chemical components* and the RapidMiner process is extended by Principal Component Analysis (PCA) to better preprocess the data and to improve the classification accuracy. *Chapter 5* explains Naïve Bayes as an algorithm for generating *classification* models and uses this modeling technique to generate a *credit approval* model to decide whether a credit loan for which a potential or existing customer applies should be approved or not, i.e. whether it is likely that the customer will pay back the credit loan as desired or not. *Chapter 6* uses Naïve Bayes to *rank applications* for nursery schools, introduces the RapidMiner operator for importing Excel sheets, and provides further explanations of Naïve Bayes.

Chapter 7 addresses the task of *product affinity-based marketing* and *optimizing a direct marketing campaign*. A bank has introduced a new financial product, a new type of current (checking) account, and some of its customers have already opened accounts of the new type, but many others have not done so yet. The bank's marketing department wants to push sales of the new account by sending direct mail to customers who have not yet opted for it. However, in order not to waste efforts on customers who are unlikely to buy, they would like to address only those customers with the highest affinity for the new product. Binary classification is used to predict for each customer, whether they will buy the product, along with a confidence value indicating how likely each of them is to buy the new product. Customers are then ranked by this confidence value and the 20% with the highest expected probability to buy the product are chosen for the campaign.

Following the *CRoss-Industry Standard Process for Data Mining (CRISP-DM)* covering all steps from business understanding and data understanding via data preprocessing and modeling to performance evaluation and deployment, this chapter first describes the task, the available data, how to extract characteristic customer properties from the customer data, their products and accounts data and their transactions, which data preprocessing to apply to balance classes and aggregate information from a customer's accounts and transactions into attributes for comparing customers, modeling with binary classification, evaluating the predictive accuracy of the model, visualizing the performance of the model using Lift charts and ROC charts, and finally ranking customers by the predicted confidence for a purchase to select the best candidates for the campaign. The predictive accuracy of several learning algorithms including Decision Trees, Linear Regression, and Logistic Regression is compared and visualized comparing their ROC charts. Automated attribute weight and parameter optimizations are deployed to maximize the prediction accuracy and thereby the customer response, sales volume, and profitability of the campaign. Similar processes can

be used for *customer churn prediction* and addressing the customers predicted to churn in a campaign with special offers trying *to prevent them from churning*.

Chapters 8 to 10 describe three different approaches to *building recommender systems*. *Product recommendations in online shops* like Amazon increase the sales volume per customer by *cross-selling*, i.e., by selling more products per customer by recommending products that the customer may also like and buy.

The recommendations can be based on product combinations frequently observed in market baskets in the past. Products that co-occurred in many purchases in the past are assumed to be also bought together frequently in the future. *Chapter 8* describes how to generate such *association rules* for product recommendations from shopping cart data using the FP-Growth algorithm. Along the way, this chapter also explains how to import product sales data from CSV files and from retailers' databases and how to handle data quality issues and missing values.

Chapter 9 introduces the *RapidMiner Extension for Recommender Systems*. This extension allows building more sophisticated recommendation systems than described in the previous chapter. The application task in this chapter is to *recommend appropriate video lectures* to potential viewers. The recommendations can be based on the content of the lectures or on the viewing behavior or on both. The corresponding approaches are called content-based, collaborative, and hybrid recommendation, respectively. Content-based recommendations can be based on attributes or similarity and collaborative recommendation systems deploy neighborhoods or factorization. This chapter explains, evaluates, and compares these approaches. It also demonstrates how to make RapidMiner processes available as *RapidAnalytics web services*, i.e., how to build a *recommendation engine* and make it available for *real-time recommendations* and easy *integration* into web sites, online shops, and other systems via web services.

A third way of building recommender systems in RapidMiner is shown in *Chapter 10*, where *classification algorithms* are used to *recommend the best-fitting study program for higher-education students* based on their predicted success for different study programs at a particular department of a particular university. The idea is an early analysis of students' success on each study program and the recommendation of a study program where a student will likely succeed. At this university department, the first year of study is common for all students. In the second year, the students select their preferred study program among several available programs. The attributes captured for each graduate student describe their success in the first-year exams, their number of points in the entrance examination, their sex, and their region of origin. The target variable is the average grade of the student at graduation, which is discretized into several categories. The prediction accuracy of several classification learning algorithms, including Naïve Bayes, Decision Trees, Linear Model Tree (LMT), and CART (Classification and Regression Trees), is compared for the prediction of the student's success as measured by the discretized average grade. For each student, the expected success classes for each study program is predicted and the study program with the highest predicted success class is recommended to the student. An optimization loop is used to determine the best learning algorithm and automated feature selection is used to find the best set of attributes for the most accurate prediction. The RapidMiner processes seamlessly integrate and compare learning techniques implemented in RapidMiner with learning techniques implemented in the open source data mining library Weka, thanks to the *Weka extension for RapidMiner* that seamlessly integrates all Weka learners into RapidMiner.

Chapter 11 provides an introduction to *clustering*, to the *k-Means* clustering algorithm, to several *cluster validity measures*, and to their *visualizations*. Clustering algorithms group cases into groups of similar cases. While for classification, a training set with examples with predefined categories is necessary for training a classifier to automatically classify new cases

into one of the predefined categories, clustering algorithms need no labeled training examples with predefined categories, but automatically group unlabeled examples into clusters of similar cases. While the predictive accuracy of classification algorithms can be easily measured by comparing known category labels of known examples to the categories predicted by the algorithm, there are no labels known in advance in the case of clustering. Hence it is more difficult to achieve an objective evaluation of a clustering result. Visualizing cluster validity measures can help humans to evaluate the quality of a set of clusters. This chapter uses k-Means clustering on a *medical dataset to find groups of similar E-Coli bacteria with regards to where protein localization occurs in them* and explains how to judge the quality of the clusters found using visualized cluster validity metrics. Cluster validity measures implemented in the open source statistics package R are seamlessly integrated and used within RapidMiner processes, thanks to the *R extension for RapidMiner*.

Chapter 12 applies *clustering* to automatically *group higher education students*. The dataset corresponds to the one already described in Chapter 10, but now the task is to find groups of similarly performing students, which is achieved with automated clustering techniques. The attributes describing the students may have missing values and different scales. Hence data preprocessing steps are used to replace missing values and to normalize the attribute values to identical value ranges. A parameter loop automatically selects and evaluates the performance of several clustering techniques including k-Means, k-Medoids, Support Vector Clustering (SVC), and DBSCAN.

Chapters 13 to 15 are about *text mining* applications. *Chapter 13* gives an *introduction to text mining*, i.e., the application of data mining techniques like classification to text documents like e-mail messages, mobile phone text messages (SMS = Short Message Service) or web pages collected from the World-Wide Web. In order to detect text message spam, preprocessing steps using the *RapidMiner text processing extension* transform the unstructured texts into document vectors of equal length, which make the data applicable to standard classification techniques like Naïve Bayes, which is then trained to *automatically separate legitimate mobile phone text messages from spam messages*.

The second text mining use case uses *classification to automatically identify the language of a text* based on its characters, character sequences, and/or words. *Chapter 14* discusses character encodings of different European, Arabic, and Asian languages. The chapter describes different text representations by characters, by tokens like words, and by character sequences of a certain length also called n-grams. The transformation of document texts into document vectors also involves the weighting of the attributes by term frequency and document frequency-based metrics like TF/IDF, which is also described here. The classification techniques Naïve Bayes and Support Vector Machines (SVM) are then trained and evaluated on four different multi-lingual text corpora including for example dictionary texts from Wikipedia and book texts from the Gutenberg project. Finally, the chapter shows how to make the RapidMiner *language detection* available as *web service for the automated language identification* of web pages via *RapidAnalytics web services*.

Chapter 15 analyses *hotel review texts* and ratings by customers collected from the TripAdvisor web page. Frequently co-occurring words in the review texts are found using FP-Growth and association rule generation and visualized in a *word-association graph*. In a second analysis, the review texts are *clustered* with k-Means, which reveals *groups of similar texts*. Both approaches provide insights about the hotels and their customers, i.e., about *topics of interest and of complaints*, quality and service issues, likes, dislikes, and preferences, and could similarly be applied to all kinds of *textual reviews and customer feedback*.

Chapter 16 describes a data mining use case in *astroparticle physics*, the application of automated *classification* and automated *feature selection* in *neutrino astronomy* to separate a small number of neutrinos from a large number of background noise particles or signals

(muons). One of the main scientific goals of neutrino telescopes is the detection of neutrinos originating from astrophysical sources as well as a precise measurement of the energy spectrum of neutrinos produced in cosmic ray air showers in the Earth's atmosphere. These so-called atmospheric neutrinos, however, are hidden in a noisy background of atmospheric muons produced in air showers as well. The first task in rejecting this background is the selection of upward-going tracks since the Earth is opaque to muons but can be traversed by neutrinos up to very high energies. This procedure reduces the background by roughly three orders of magnitude. For a detailed analysis of atmospheric neutrinos, however, a very clean sample with purity larger than 95% is required. The main source of remaining background at this stage are muon tracks, falsely reconstructed as upward going. These falsely reconstructed muon tracks still dominate the signal by three orders of magnitude and have to be rejected by the use of straight cuts or multivariate methods. Due to the ratio of noise (muons) and signal (neutrinos), about 10,000 particles need to be recorded in order to catch about 10 neutrinos. Hence, the amount of data delivered by these experiments is enormous and it must be processed and analyzed within a proper amount of time. Moreover, data in these experiments are delivered in a format that contains more than 2000 attributes originating from various reconstruction algorithms. Most of these attributes have been reconstructed from only a few physical quantities. The direction of a neutrino event penetrating the detector at a certain angle can, for example, be reconstructed from a pattern of light that is initiated by particles produced by an interaction of the neutrino close to or even in the detector. Due to the fact that all of the 2000 reconstructed attributes are not equally well suited for classification, the first task in applying data mining techniques in neutrino astronomy lies in finding a good and reliable representation of the dataset in fewer dimensions. This is a task which very often determines the quality of the overall data analysis. The second task is the training and evaluation of a stable learning algorithm with a very high performance in order to separate signal and background events. Here, the challenge lies in the biased distribution of many more background noise (negative) examples than there are signals (positive) examples. Handling such skewed distributions is necessary in many real-world problems. The application of RapidMiner in neutrino astronomy models the separation of neutrinos from background as a two-step process, accordingly. In this chapter, the *feature or attribute selection* is explained in the first part and the *training of selecting relevant events* from the masses of incoming data is explained in the second part. For the feature selection, the *Feature Selection Extension for RapidMiner* is used and a wrapper cross-validation to evaluate the performance of the feature selection methods. For the selection of the relevant events, *Random Forests* are used as classification learner.

Chapter 17 provides an introduction to *medical data mining*, an overview of methods often used for classification, regression, clustering, and association rules generation in this domain, and two application use cases with data about patients suffering from *carpal tunnel syndrome* and *diabetes*, respectively.

In the study of the *carpal tunnel syndrome (CTS)*, thermographic images of hands were collected for constructing a predictive classification model for CTS, which could be helpful when looking for a non-invasive diagnostic method. The temperatures of different areas of a patient's hand were extracted from the image and saved in the dataset. Using a RapidMiner preprocessing operator for aggregation, the temperatures were averaged for all segments of the thermal images. Different machine learning algorithms including *Artificial Neural Network* and *Support Vector Machines (SVM)* were evaluated for generating a classification model capable of diagnosing CTS on the basis of very discrete temperature differences that are invisible to the human eye in a thermographic image.

In the study of *diabetes*, various research questions were posed to evaluate the level of knowledge and overall perceptions of diabetes mellitus type 2 (DM) within the older population in North-East Slovenia. As a chronic disease, diabetes represents a substantial

burden for the patient. In order to accomplish good self-care, patients need to be qualified and able to accept decisions about managing the disease on a daily basis. Therefore, a high level of knowledge about the disease is necessary for the patient to act as a partner in managing the disease. Various research questions were posed to determine what the general knowledge about diabetes is among diabetic patients 65 years and older, and what the difference in knowledge about diabetes is with regard to the education and place of living on (1) diet, (2) HbA1c, (3) hypoglycemia management, (4) activity, (5) effect of illness and infection on blood sugar levels, and (6) foot care. A hypothesis about the level of general knowledge of diabetes in older populations living in urban and rural areas was predicted and verified through the study. A cross-sectional study of older (age >65 years), non-insulin dependent patients with diabetes mellitus type 2 who visited a family physician, DM outpatient clinic, a private specialist practice, or were living in a nursing home was implemented. The Slovenian version of the Michigan Diabetes Knowledge test was then used for data collection. In the data preprocessing, missing values in the data were replaced, before *k-means clustering* was used to find groups of similar patients, for which then a *decision tree* learner was used to find attributes discriminating the clusters and generate a classification model for the clusters. A grouped *ANOVA (ANalysis Of VAriances) statistical test* verified the hypothesis that there are differences in the level of knowledge about diabetes in rural populations and city populations in the age group of 65 years and older.

Chapter 18 covers a use case relevant in *chemistry* and the *pharmaceutical industry*. The RapidMiner Extension *PaDEL (Pharmaceutical Data Exploration Laboratory)* developed at the University of Singapore is deployed to *calculate a variety of molecular properties from the 2-D or 3-D molecular structures of chemical compounds*. Based on these molecular property vectors, RapidMiner can then generate predictive models for *predicting chemical, biochemical, or biological properties based on molecular properties*, which is a frequently encountered task in theoretical chemistry and the pharmaceutical industry. The combination of RapidMiner and PaDEL provides an open source solution to generate prediction systems for a broad range of biological properties and effects.

One application example in *drug design* is the *prediction of effects and side effects* of a new drug candidate before even producing it, which can help to avoid testing many drug candidates that probably are not helpful or possibly even harmful and thereby help to focus research resources on more promising drug candidates. With PaDEL and RapidMiner, properties can be calculated for any molecular structure, even if the compound is not physically accessible. Since both tools are open source and can compute the properties of a molecular structure quickly, this allows significant reduction in cost and an increase in speed of the development of new chemical compounds and drugs with desired properties, because more candidate molecules can be considered automatically and fewer of them need to be actually generated and physically, chemically, or biologically tested.

The combination of data mining (RapidMiner) and a tool to handle molecules (PaDEL) provides a convenient and user-friendly way to generate accurate relationships between chemical structures and any property that is supposed to be predicted, mostly biological activities. Relationships can be formulated as qualitative structure-property relationships (SPRs), qualitative structure-activity relationships (SARs) or quantitative structure-activity relationships (QSARs). SPR models aim to highlight associations between molecular structures and a target property, such as lipophilicity. SAR models correlate an activity with structural properties and QSAR models quantitatively predict an activity. Models are typically developed to predict properties that are difficult to obtain, impossible to measure, require time-consuming experiments, or are based on a variety of other complex properties. They may also be useful to predict complicated properties using several simple properties. The PaDEL extension enables RapidMiner to directly read and handle molecular structures, calculate their molecular properties, and to then correlate them to and generate predictive

models for chemical, biochemical, or biological properties of these molecular structures. In this chapter *linear regression* is used as a *QSAR modeling* technique to *predict chemical properties* with RapidMiner *based on molecular properties* computed by PaDEL.

Chapter 19 describes a second *Quantitative Structure-Activity Relationship (QSAR)* use case relevant in *chemistry* and the *pharmaceutical industry, the identification of novel functional inhibitors of acid sphingomyelinase (ASM).* The use case in this chapter is based on the previous chapter and hence you should first read Chapter 18 before reading this chapter. In the data preprocessing step, the *PaDEL (Pharmaceutical Data Exploration Laboratory)* extension for RapidMiner described in the previous chapter is again used to compute molecular properties from given molecular 2-D or 3-D structures. These properties are then used to *predict ASM inhibition.* Automated *feature selection* with *backward elimination* is used to reduce the number of properties to a relevant set for the prediction task, for which a *classification* learner, namely *Random Forests*, generates the predictive model that captures the *structure- and property-activity relationships.*

The process of drug design from the biological target to the drug candidate and, subsequently, the approved drug has become increasingly expensive. Therefore, strategies and tools that reduce costs have been investigated to improve the effectiveness of drug design. Among them, the most time-consuming and cost-intensive steps are the selection, synthesis, and experimental testing of the drug candidates. Therefore, numerous attempts have been made to reduce the number of potential drug candidates for experimental testing. Several methods that rank compounds with respect to their likelihood to act as an active drug have been developed and applied with variable success. In silico methods that support the drug design process by reducing the number of promising drug candidates are collectively known as virtual screening methods. Their common goal is to reduce the number of drug candidates subjected to biological testing and to thereby increase the efficacy of the drug design process.

This chapter demonstrates an in silico method to predict biological activity based on RapidMiner data mining work flows. This chapter is based on the type of chemoinformatic predictions described in the previous chapter based on chemoinformatic descriptors computed by PaDEL. *Random Forests* are used as a predictive model for predicting the molecular activity of a molecule of a given structure, for which PaDEL is used to compute molecular structural properties, which are first reduced to a smaller set by automated *attribute weighting* and *selecting the attributes* with the highest weights according to several weighting criteria and which are reduced to an even smaller set of attributes by automated *attribute selection* using a *Backward Elimination* wrapper. Starting with a large number of properties for the example set, a *feature selection* vastly reduces the number of attributes before the systematic backward elimination search finds the most predictive model for the feature generation. Finally, a *validation* is performed to avoid over-fitting and the benefits of *Y-randomization* are shown.

Chapter 20 introduces the *RapidMiner IMage Mining (IMMI) Extension* and presents some introductory *image processing* and *image mining* use cases. *Chapter 21* provides more advanced image mining applications.

Given a set of images in a file folder, the image processing task in the first use case in *Chapter 20* is to adjust the contrast in all images in the given folder and to store the transformed images in another folder. The IMMI extension provides RapidMiner operators for *reading and writing images*, which can be used within a RapidMiner loop iterating over all files in the given directory, *adjusting the contrast* of each of these images, for example, using a histogram equalization method. Then the chapter describes image *conversions between color and gray-scale images* and different *feature extraction* methods, which convert image data in unstructured form into a tabular form. Feature extraction algorithms for images can

be divided into three basic categories: local-level, segment-level, and global-level feature extraction.

The term *local-level* denotes that information is mined from given points (locations) in the image. *Local-level feature extraction* is suitable for *segmentation*, *object detection* or *area detection*. From each point in the image, it is possible to extract information like pixel gray value, minimal or maximal gray value in a specified radius, value after applying kernel function (blurring, edge enhancements). Examples of utilization of such data are the *trainable segmentation of an image*, *point of interest detection*, and *object detection*.

The term *segment-level* denotes feature extraction from segments. Many different segmentation algorithms exist, such as k-means, watershed, or statistical region merging. Segment level feature extraction algorithms extract information from the whole segments. Examples of such features are mean, median, lowest and highest gray value, circularity, and eccentricity. In contrast to local-level features, it does not take into consideration only a single point and its neighborhood, however, it considers the whole segment and its properties like shape, size, and roundness. With the use of knowledge about the size or shape of target objects, it is for example possible to *select or remove objects according to their size or shape*.

The *global-level* denotes *feature extraction from the whole image*, for example, mean color, dominant color, maximal gray value, minimal gray value, variance of pixels, number of edges etc. Unlike the local or segment level, the global level segmentation is not suitable for points or areas identification or segmentation. Rather, it is suitable *for classification of images* and determining *properties of the image as a whole*.

Chapter 20 provides examples demonstrating the use of local-level, segment-level, and global-level feature extraction. *Local-level feature extraction* is used for *trainable image segmentation* with radial-basis function (RBF) Support Vector Machines (SVM). *Segment-level feature extraction* and trainable segment selection reveal interesting segment properties like size and shape for image analysis. With the help of *global-level feature extraction*, images are classified into pre-defined classes. In the presented use case, two *classes of images* are *distinguished automatically*: images containing birds and images containing sunsets. To achieve this, global features like dominant color, minimal intensity, maximal intensity, percent of edges, etc. are extracted and based on those, an image classifier is trained.

Chapter 21 presents *advanced image mining applications* using the *RapidMiner IMage Mining (IMMI) Extension* introduced in the previous chapter. This chapter demonstrates several examples of the use of the IMMI extension for image processing, image segmentation, feature extraction, pattern detection, and image classification. The first application *extracts global features* from multiple images to enable *automated image classification*. The second application demonstrates the *Viola-Jones algorithm for pattern detection*. And the third process illustrates the *image segmentation* and *mask processing*.

The *classification of an image* is used to identify which group of images a particular image belongs to. An automated image classifier could, for example, be used to distinguish different scene types like nature versus urban environment, exterior versus interior, images with and without people, etc. *Global features* are usually used for this purpose. These features are calculated from the whole image. The key to a correct classification is to find the features that differentiate one class from other classes. Such a feature can be, for example, the dominant color in the image. These features can be calculated from the original image or from an image after pre-processing like Gaussian blur or edge detection.

Pattern detection searches known patterns in images in the images, where approximate fits of the patterns may be sufficient. A good algorithm for detection should not be sensitive to the size of the pattern in the image or its position or rotation. One possible approach is to use a histogram. This approach compares the histogram of the pattern with the histogram of a selected area in the image. In this way, the algorithm passes step by step through

the whole image, and if the match of histograms is larger than a certain threshold, the area is declared to be the sought pattern. Another algorithm, which is described in this chapter, is the Viola-Jones algorithm. The classifier is trained with positive and negative image examples. Appropriate features are selected using the AdaBoost algorithm. An image is iterated during pattern detection using a window with increasing size. Positive detections are then marked with a square area of the same size as the window. The provided example application uses this process to detect the cross-sectional artery in an ultrasound image. After detection, the images can be used to measure the patient's pulse if taken from a video or stream of time-stamped images.

The third example application demonstrates *image segmentation* and *feature extraction*: Image segmentation is often used for the *detection of different objects* in the image. Its task is to split the image into parts so that the individual segments correspond to objects in the image. In this example, the identified segments are combined with *masks* to remove the background and focus on the object found.

Chapter 22 introduces the *RapidMiner Extension for Instance Selection and Prototype-based Rule (ISPR) induction*. It describes the *instance selection* and *prototype construction* methods implemented in this extension and applies them *to accelerate* 1-NN classification on large datasets and to perform *outlier elimination* and *noise reduction*. The datasets analyzed in this chapter include several *medical datasets* for classifying patients with respect to certain medical conditions, i.e., *diabetes, heart diseases, and breast cancer*, as well as an *e-mail spam detection* dataset. The chapter describes a variety of *prototype selection algorithms* including k- Nearest-Neighbors (k-NN), Monte-Carlo (MC) algorithm, Random Mutation Hill Climbing (RMHC) algorithm, Condensed Nearest-Neighbor (CNN), Edited Nearest-Neighbor (ENN), Repeated ENN (RENN), Gabriel Editing proximity graph-based algorithm (GE selection), Relative Neighbor Graph algorithm (RNG selection), Instance-Based Learning (IBL) algorithm (IB3 selection), Encoding Length Heuristic (ELH selection), and combinations of them and compares their performance on the datasets mentioned above. *Prototype construction methods* include all algorithms that produce a set of instances at the output. The family contains all prototype-based clustering methods like k-Means, Fuzzy C-Means (FCM), and Vector Quantization (VQ) as well as the Learning Vector Quantization (LVQ) set of algorithms. The price for the speed-up of 1-NN by instance selection is visualized by the drop in predictive accuracy with decreasing sample size.

Chapter 23 gives an overview of a large range of *anomaly detection* methods and introduces the *RapidMiner Anomaly Detection Extension*. Anomaly detection is the process of finding patterns in a given dataset which deviate from the characteristics of the majority. These outstanding patterns are also known as anomalies, outliers, intrusions, exceptions, misuses, or fraud. Anomaly detection identifies single records in datasets which significantly deviate from the normal data. Application domains among others include *network security*, *intrusion detection, computer virus detection*, *fraud detection*, *misuse detection*, *complex system supervision*, and *finding suspicious records in medical data*. Anomaly detection for fraud detection is used to *detect fraudulent credit card transactions* caused by stolen credit cards, *fraud in Internet payments*, and *suspicious transactions in financial accounting data*. In the medical domain, anomaly detection is also used, for example, for *detecting tumors in medical images* or *monitoring patient data* (electrocardiogram) to get early warnings in case of life-threatening situations. Furthermore, a variety of other specific applications exists such as *anomaly detection in surveillance camera data*, *fault detection in complex systems* or *detecting forgeries in the document forensics*. Despite the differences of the various application domains, the basic principle remains the same. Multivariate normal data needs to be modeled and the few deviations need to be detected, preferably with a *score* indicating their "outlierness", i.e., a score indicating their extent of being an outlier. In case

of a univariate data, such an *outlier factor* could for example be the number of standard deviations by which an outlier differs from the mean of this variable.

The overview of anomaly detection method provided in this chapter distinguishes three different types of anomalies, namely (1) *point anomalies*, which are single data records deviating from others, (2) *contextual anomalies*, which occur with respect to their context only, for example, with respect to time, and (3) *collective anomalies*, where a bunch of data points causes the anomaly. Most anomaly detection algorithms detect point anomalies only, which leads to the requirement of transforming contextual and collective anomalies to point anomaly problems using an appropriate pre-processing and thus generating processable data views. Furthermore, anomaly detection algorithms can be categorized with respect to their operation mode, namely (1) *supervised algorithms* with training and test data as used in traditional machine learning, (2) *semi-supervised algorithms* with the need of anomaly-free training data for one-class learning, and (3) *unsupervised approaches* without the requirement of any labeled data. Anomaly detection is, in most cases, associated with an unsupervised setup, which is also the focus of this chapter. In this context, all available unsupervised algorithms from the RapidMiner anomaly detection extension are described and the most well-known algorithm, the *Local Outlier Factor (LOF)* is explained in detail in order to get a deeper understanding of the approaches themselves. The *unsupervised anomaly detection algorithms* covered in this chapter include Grubbs' outlier test and noise removal procedure, k-NN Global Anomaly Score, Local Outlier Factor (LOF), Connectivity-Based Outlier Factor (COF), Influenced Outlierness (INFLO), Local Outlier Probability (LoOP), Local Correlation Integral (LOCI) and aLOCI, Cluster-Based Local Outlier Factor (CBLOF), and Local Density Cluster-Based Outlier Factor (LDCOF). The *semi-supervised anomaly detection algorithms* covered in this chapter include a one-class Support Vector Machine (SVM) and a two-step approach with clustering and distance computations for detecting anomalies.

Besides a simple example consisting of a two-dimensional mixture of Gaussians, which is ideal for first experiments, two real-world datasets are analyzed. For the unsupervised anomaly detection the player statistics of the NBA, i.e., a dataset with the *NBA regular-season basketball player statistics from 1946 to 2009*, are analyzed for outstanding players, including all necessary pre-processing. The *UCI NASA shuttle dataset* is used for illustrating how semi-supervised anomaly detection can be performed in RapidMiner to find suspicious states during a NASA shuttle mission. In this context, a *Groovy script* is implemented for a simple *semi-supervised cluster-distance-based anomaly detection* approach, showing how to easily extend RapidMiner by your own operators or scripts.

Chapter 24 features a complex *data mining research use case*, the *performance evaluation and comparison of several classification learning algorithms* including Naïve Bayes, k-NN, Decision Trees, Random Forests, and Support Vector Machines (SVM*) across many different datasets.* Nested process control structures for loops over datasets, loops over different learning algorithms, and cross validation allow an automated validation and the selection of the best model for each application dataset. Statistical tests like t-test and ANOVA test (ANalysis Of VAriance) determine whether performance differences between different learning techniques are statistically significant or whether they may be simply due to chance. Using a custom-built Groovy script within RapidMiner, meta-attributes about the datasets are extracted, which can then be used for *meta-learning*, i.e., for learning to predict the performance of each learner from a given set of learners for a given new dataset, which then allows the selection of the learner with the best expected accuracy for the given dataset. The performance of fast learners called landmarkers on a given new dataset and the meta-data extracted from the dataset can be used for meta-learning to predict the performance of another learner on this dataset. The RapidMiner Extension for Pattern Recognition Engineering (PaREn) and its Automatic System Construction Wizard perform this kind of

meta-learning for automated learner selection and a parameter optimization for a given dataset.

The *index at the end of the book* helps you to find explanations of data mining concepts and terms you would like to learn more about, use case applications you may be interested in, or reference use cases for certain modeling techniques or RapidMiner operators you are looking for. The *companion web page for this book* provides the RapidMiner processes and datasets deployed in the use cases:
`http://www.RapidMinerBook.com`

About the Editors

Markus Hofmann

Dr. Markus Hofmann is currently a lecturer at the Institute of Technology Blanchardstown in Ireland where he focuses on the areas of data mining, text mining, data exploration and visualisation as well as business intelligence. He holds a PhD degree from Trinity College Dublin, an MSc in Computing (Information Technology for Strategic Management) from the Dublin Institute of Technology and a BA in Information Management Systems. He has taught extensively at undergraduate and postgraduate level in the fields of Data Mining, Information Retrieval, Text/Web Mining, Data Mining Applications, Data Pre-processing and Exploration and Databases. Dr. Hofmann published widely at national as well as international level and specialised in recent years in the areas of Data Mining, learning object creation, and virtual learning environments. Further he has strong connections to the Business Intelligence and Data Mining sector both on an academic as well as industry level. Dr. Hofmann has worked as technology expert together with 20 different organisations in recent years including companies such as Intel. Most of his involvement was on the innovation side of technology services and products where his contributions had significant impact on the success of such projects. He is a member of the Register of Expert Panellists of the Irish Higher Education and Training Awards council, external examiner to two other third level institutes and a specialist in undergraduate and post graduate course development. He has been internal as well as external examiner of postgraduate thesis submissions. He was also local and technical chair of national and international conferences.

Ralf Klinkenberg

Ralf Klinkenberg holds Master of Science degrees in computer science with focus on machine learning, data mining, text mining, and predictive analytics from the Technical University of Dortmund in Germany and Missouri University of Science and Technology in the USA. He performed several years of research in these fields at both universities before initiating the RapidMiner open source data mining project in 2001, whose first version was called Yet Another Learning Environment (YALE). Ralf Klinkenberg founded this software project together with Dr. Ingo Mierswa and Dr. Simon Fischer. In 2006 he founded the company Rapid-I together with Ingo Mierswa. Rapid-I now is the company behind the open source software solution RapidMiner and its server version RapidAnalytics, providing these and further data analysis solutions, consulting, training, projects, implementations, support, and all kinds of related services. Ralf Klinkenberg has more than 15 years of experience in consulting and training large and small corporations and organizations in many different sectors how to best leverage data mining and RapidMiner based solutions for their needs. He performed data mining, text mining, web mining, and business analytics projects for companies like telecoms, banks, insurances, manufacturers, retailers, pharmaceutical com-

panies, healthcare, IT, aviation, automotive, and market research companies, utility and energy providers, as well as government organizations in many European and North American countries. He provided solutions for tasks like automated direct marketing campaign optimization, churn prediction and prevention, sales volume forecasting, automated online media monitoring and sentiment analysis to generate customer insights, market insights, and competitive intelligence, customer feedback analysis for product and service optimization, automated e-mail routing, fraud detection, preventive maintenance, machine failure prediction and prevention, manufacturing process optimization, quality and cost optimization, profit maximization, time series analysis and forecasting, critical event detection and prediction, and many other data mining and predictive analytics applications.

List of Contributors

Editors

- **Markus Hofmann**, Institute of Technology Blanchardstown, Ireland
- **Ralf Klinkenberg**, Rapid-I, Germany

Chapter Authors

- **Ingo Mierswa**, Rapid-I, Germany
- **M. Fareed Akhtar**, Fastonish, Australia
- **Timm Euler**, viadee IT-Consultancy, Münster/Köln (Cologne), Germany
- **Matthew A. North**, The College of Idaho, Caldwell, Idaho, USA
- **Matej Mihelčić**, Electrical Engineering, Mathematics and Computer Science, University of Twente, Netherlands; Rudjer Boskovic Institute, Zagreb, Croatia
- **Matko Bošnjak**, University of Porto, Porto, Portugal; Rudjer Boskovic Institute, Zagreb, Croatia
- **Nino Antulov-Fantulin**, Rudjer Boskovic Institute, Zagreb, Croatia
- **Tomislav Šmuc**, Rudjer Boskovic Institute, Zagreb, Croatia
- **Milan Vukićević**, Faculty of Organizational Sciences, University of Belgrade, Belgrade, Serbia
- **Miloš Jovanović**, Faculty of Organizational Sciences, University of Belgrade, Belgrade, Serbia
- **Boris Delibašić**, Faculty of Organizational Sciences, University of Belgrade, Belgrade, Serbia
- **Milija Suknović**, Faculty of Organizational Sciences, University of Belgrade, Belgrade, Serbia
- **Andrew Chisholm**, Institute of Technology, Blanchardstown, Dublin, Ireland
- **Neil McGuigan**, University of British Columbia, Sauder School of Business, Canada
- **Eduarda Mendes Rodrigues**, University of Porto, Porto, Portugal
- **Luis Sarmento**, Sapo.pt - Portugal Telecom, Lisbon, Portugal
- **Gurdal Ertek**, Sabancı University, Istanbul, Turkey
- **Dilek Tapucu**, Sabancı University, Istanbul, Turkey

- **Inanc Arin**, Sabancı University, Istanbul, Turkey
- **Tim Ruhe**, TU Dortmund, Dortmund, Germany
- **Katharina Morik**, TU Dortmund, Dortmund, Germany
- **Wolfgang Rhode**, TU Dortmund, Dortmund, Germany
- **Mertik Matej**, Faculty of information study Novo mesto, Slovenia
- **Palfy Miroslav**, University Medical Centre Maribor, Slovenia
- **Markus Muehlbacher**, Department of Psychiatry and Psychotherapy, University Hospital of Erlangen-Nuremberg, Friedrich-Alexander-University Erlangen, Germany; Computer Chemistry Center, Friedrich-Alexander-University Erlangen, Germany
- **Johannes Kornhuber**, Department of Psychiatry and Psychotherapy, University Hospital of Erlangen-Nuremberg, Friedrich-Alexander-University Erlangen, Germany
- **Radim Burget**, Brno University of Technology, Czech Republic
- **Václav Uher**, Brno University of Technology, Czech Republic
- **Jan Masek**, Brno University of Technology, Czech Republic
- **Marcin Blachnik**, Silesian University of Technology, Department of Management and Informatics, Poland
- **Miroslaw Kordos**, University of Bielsko-Biala, Department of Mathematics and Computer Science, Bielsko-Biala, Poland
- **Markus Goldstein**, German Research Center for Artificial Intelligence, Kaiserslautern, Germany

Acknowledgments

A lot of people have contributed to make this book and the underlying open source software solutions RapidMiner and RapidAnalytics happen. We are thankful to all of you.

We would like to thank the contributing authors of this book, who shared their experience in the chapters and who thereby enable others to have a quick and successful data mining start with RapidMiner providing successful application examples and blueprints for the readers to tackle their data mining tasks and benefit from the strength of using RapidMiner and RapidAnalytics.

Many thanks to Dr. Brian Nolan, Head of Department of Informatics, Institute of Technology Blanchardstwon (ITB), for continuously supporting the relationship between the institute and Rapid-I.

The entire team of the Taylor & Francis Group were very professional, responsive and always helpful in guiding us through this project. Should any of you readers consider publishing a book then we can highly recommend this publisher.

Before there could be any thought about a book like this one, there needed to be the open source data mining software RapidMiner to whose success many contributed.

A special thanks goes to Prof. Dr. Katharina Morik, Head of the Artificial Intelligence Unit at the Technical University of Dortmund, Germany, for providing an introduction to and deep insights into machine learning (ML), data mining, text mining, artificial intelligence (AI), and natural language processing (NLP), for providing an environment that enabled the initiation of an open source data mining software project named YALE (Yet Another Learning Environment), which was later improved and renamed to RapidMiner. She supports the open source project RapidMiner, the company Rapid-I, which is behind the project, and its founders until today, long after they left the university. We appreciate the good cooperation and exchange of ideas with her research and teaching unit.

Another big thank you goes to Dr. Ingo Mierswa and Dr. Simon Fischer who started the open source project YALE and later RapidMiner together with Ralf Klinkenberg and who took turns in supervising the international development team of the project. Without their ideas, passion, commitment, and enormous development work, we would not have such a powerful and flexible open source data mining framework and solution today, available for everyone to use and gain from.

Dr. Ingo Mierswa and Ralf Klinkenberg are also the co-founders of Rapid-I, the data mining and business analytics company behind the open source software RapidMiner.

We are grateful to all who joined the RapidMiner and Rapid-I teams, especially to Sebastian Land, Helge Homburg, and Tobias Malbrecht who joined the RapidMiner team in its early days and contributed a lot to its development and who are strong members of the team until today. We are also thankful to all contributors from the early days, like Michael Wurst, Martin Scholz, and Timm Euler, as well as to the newer team members like Marius Helf, Nils-Christian Whler, Marcin Skirzynski, Venkatesh Umaashankar, Marco Bck, Dominik Halfkann, and to those who support the team in other roles, like Nadja Mierswa, Simone Horstmann, Christian Brabandt, Edin Klapic, Balzs Brny, Dietrich Niederlintner, Caroline Hahne, Miguel Bscher, Thilo Kamradt, Jannik Zappe, Kyle Goslin, and Assumpta Harvey.

Open source projects grow strong with their community. We are thankful to all contributors to RapidMiner and RapidAnalyitcs and to all supporters of these open source projects. We are grateful not only for source code contributions, community support in the forum, bug reports and fixes, but also to those who spread the word with their blogs, videos, and words of mouth, especially to Thomas Ott,[3] Neil McGuigan,[4] Dr. Bala Deshpande,[5] Prof. Dr. Bonnie Holub,[6] Prof. Dr. Matthew North,[7] Sheamus McGovern,[8] Prof. Dr. David Wheismann, and many more.

Many bright minds have influenced our thoughts and inspired us with their ideas and valuable discussion. We would like to thank Prof. Dr. Thorsten Joachims, Prof. Dr. Hans-Ulrich Simon, Prof. Dr. Daniel St. Clair, Prof. Dr. Cihan Dagli, Prof. Dr. Tom Mitchell, and many others for widening our horizon and for many deep insights into the theoretical limits, enormous capabilities, and many practical applications of machine learning, data mining, data mining, text mining, statistical learning theory, and predictive analytics.

We would also like the many companies and organizations supporting the development of RapidMiner and RapidAnalytics by becoming customers of Rapid-I, including, for example, Sanofi, Daimler, Honda, Volkswagen, Miele, Siemens, Telekom Deutschland, T-Mobile International, mobilkom austria, Telenor, Nokia, Philips, Lufthansa, EADS, Salzgitter Mannesmann, ThyssenKrupp, Libri, Tchibo, KNV, PayPal, Intel, PepsiCo, GfK, Landesbank Berlin, E.ON, RWE, Axel Springer, 1&1, Schober, Schweizer Bundesbahn, FC Chelsea, The Cleveland Indians, and many more[9].

With best regards and appreciation to all contributors,

Dr. Markus Hofmann, Institute of Technology Blanchardstown, Dublin, Ireland

Ralf Klinkenberg, Co-Founder of RapidMiner and Rapid-I, CBDO, Rapid-I, Dortmund, Germany

[3] http://www.NeuralMarketTrends.com/

[4] http://vancouverdata.blogspot.com/

[5] http://www.SimaFore.com/

[6] http://www.arclight.biz/uncategorized/rapidminer-introductory-tutorial-videos/

[7] http://docs.rapid-i.com/r/data-mining-for-the-masses

[8] http://www.cambridgecodeworks.com/wordpress/?author=2, http://www.meetup.com/Boston-Predictive-Analytics/

[9] http://rapid-i.com/content/view/8/56/

List of Figures

List of Tables

Part I

Introduction to Data Mining and RapidMiner

Chapter 1

What This Book is About and What It is Not

Ingo Mierswa

Rapid-I, Dortmund, Germany

1.1 Introduction

Today, analytics is a very important topic and affects practically all levels in modern organizations. Analytics is also used by many data-driven researchers. Data is collected and analyzed, and the results of this analytical work either prove our hypothesis or delivers new insights.

When we talk about analytics in this book we are referring to what many call "advanced analytics". This field includes technologies known from statistics as well as from computer science. Isn't this just statistics done by computers then? By far not! Statistics often deal with the question if a hypothesis can be proven by a statistical test using a small but representative data sample. Although this is of biggest importance, it is even more useful to mix these ideas with algorithms from computer science to make sure that the methods we are talking about will be able to scale up and analyze even the largest datasets.

And I see another distinction: traditional statistics often requires that the analyst is creating a model or hypothesis right at the start of the analysis process. After creating such a model, parameters of the model are estimated or the applicability of this model is proven by means of a statistical test. Maybe this is because I am lazy, but I actually don't like this idea too much: Why should I manually do the work a computer is perfectly able to deal with itself? In this sense of manual analysis, statistical analysis is much more connected to online analytical processing (OLAP) than with "advanced analytics": In the OLAP world, people try to drill through their data to find the interesting patterns and the reasons in deeper data levels themselves. Fine, but again I think this is the wrong approach for solving

the underlying problem for mainly two reasons: Firstly, people tend to see only what they are looking for. Most analysts have some expectations before they start and try to work as long as necessary on the data to prove their point. Secondly, OLAP is again a pretty tedious work I personally believe a computer is much better suited for. Did I already mention that I am pretty lazy in this respect? I prefer actually to describe myself as "unbiased" and "efficient".

Don't get me wrong: statistics and OLAP offer very important methods necessary for many day-to-day business cases and I myself am half computer scientist and half statistician. However, if you mix the methods described above with algorithms from computer science to scale those methods up to larger datasets and also throw in some ideas from artificial intelligence, especially from the field of machine learning, I personally think that much more interesting possibilities can arise. This is actually nothing new and has been an important field for research during the last decades. The methods and algorithms which have been developed over this time have actually formed complete new research fields under the names data mining, predictive analytics, or pattern detection. And one of the most amazing developments is a collection of methods which can not only be used on structured data, i.e., on tables, but also on unstructured data like texts or images. This has been the underlying motivation for fields like text mining, image mining, or audio mining.

Most recently a new buzzword has been used a lot: Big Data. Well, most recently means in the years 2012 and following, so if you are reading this book in 2092 you might want to use this as a historical lecture. OK, back to big data: What is so special about that? If you ask me, not that much from the point of view of a data miner. Big data is an umbrella term for many ideas and technologies, but the underlying point of all those things is that big data should be about infrastructure and methods for collecting, retrieving, and analyzing very large datasets which might be of structured, unstructured, or polystructured nature. Well, if you have read the paragraphs above you will certainly agree that this sounds actually like a perfect description of "data mining". As of 2013, the big data market is still in its early days and most people are worrying about data infrastructures. But this will change as soon as organizations do understand that the mere collection of data is worth nothing. It is the analysis of data which delivers new insights, explains underlying patterns, or creates models which can be extrapolated in order to predict the most likely future.

Reading this book might therefore be a very good idea for learning more about where and how data mining can be used to deliver those insights. It might also just serve you for your personal career—remember, the big data market will slowly but surely move into the direction of analytics in the future. Whatever the reason is, I hope that you will learn more about the use cases discussed in this book so that you are able to transfer them to your own business problems. RapidMiner is an excellent and very flexible tool for re-using those use cases and adapt them to your concrete problems. Have fun using it!

In the following, I would like to discuss what data mining actually is and to which types of problems it can be applied. At the same time I will give you an introduction to the most important terms. Whether you are already an experienced data mining expert or not, this chapter is worth reading in order for you to know and have a command of the terms used both here in this book as well as in RapidMiner.

1.2 Coincidence or Not?

Before we get properly started, let us try a small experiment:

1. Think of a number between 1 and 10.
2. Multiply this number by 9.
3. Work out the checksum of the result, i.e., the sum of the numbers.
4. Multiply the result by 4.
5. Divide the result by 3.
6. Deduct 10.

The result is 2.

Do you believe in coincidence? As an analyst you will probably learn to answer this question in the negative or even do so already. Let us take for example what is probably the simplest random event you could imagine, i.e., the toss of a coin. "Ah", you may think, "but that is a random event and nobody can predict which side of the coin will be showing after it is tossed". That may be correct, but the fact that nobody can predict it does in no way mean that it is impossible in principle. If all influence factors such as the throwing speed and rotation angle, material properties of the coin and those of the ground, mass distributions and even the strength and direction of the wind were all known exactly, then we would be quite able, with some time and effort, to predict the result of such a coin toss. The physical formulas for this are all known in any case.

We shall now look at another scenario, only this time we can predict the outcome of the situation: A glass will break if it falls from a certain height onto a certain type of ground. We even know in the fractions of a second when the glass is falling that there will be broken glass. How are we able to achieve this rather amazing feat? We have never seen the glass which is falling in this instant break before and the physical formulas that describe the breakage of glass are a complete mystery for most of us at least. Of course, the glass may stay intact "by chance" in individual cases, but this is not likely. For what it's worth, the glass not breaking would be just as non-coincidental, since this result also follows physical laws. For example, the energy of the impact is transferred to the ground better in this case. So how do we humans know what exactly will happen next in some cases and in other cases, for example that of the toss of a coin, what will not?

The most frequent explanation used by laymen in this case is the description of the one scenario as "coincidental" and the other as "non-coincidental". We shall not go into the interesting yet nonetheless rather philosophical discussions on this topic, but we are putting forward the following thesis:

The vast majority of processes in our perceptible environment are not a result of coincidences. The reason for our inability to describe and extrapolate the processes precisely is rather down to the fact that we are not able to recognize or measure the necessary influence factors or correlate them.

In the case of the falling glass, we quickly recognized the most important characteristics, such as the material, falling height, and nature of the ground, and can already estimate, in the shortest time, the probability of the glass breaking by analogy reasoning from similar experiences. However, it is just that we cannot do with the toss of a coin. We can watch as many tosses of a coin as we like; we will never manage to recognize the necessary factors fast enough and extrapolate them accordingly in the case of a random throw.

So what were we doing in our heads when we made the prediction for the state of the glass after the impact? We measured the characteristics of this event. You could also say that we collected data describing the fall of the glass. We then reasoned very quickly by analogy, i.e., we made a comparison with earlier falling glasses, cups, porcelain figurines,

or similar articles based on a similarity measure. Two things are necessary for this: firstly, we need to also have the data of earlier events available and secondly, we need to be aware of how a similarity between the current and past data is defined at all. Ultimately we are able to make an estimation or prediction by having looked at the most similar events that have already taken place, for example. Did the falling article break in these cases or not? We must first find the events with the greatest similarity, which represents a kind of optimization. We use the term "optimization" here, since it is actually unimportant whether we are now maximizing a similarity or the sales figures of one enterprise or any other—the measurement concerned, in this case similarity, is always optimized. The analogy reasoning described then tells us that the majority of glasses we have already looked at broke and this very estimation then becomes our prediction. This may sound complicated, but this kind of analogy reasoning is basically the foundation for almost every human learning process and is done at a staggering speed.

The interesting thing about this is that we have just been acting as a human data mining method, since data analysis usually involves matters such as the representation of events or conditions and the data resulting from this, the definition of events' similarities and of the optimization of these similarities.

However, the described procedure of analogy reasoning is not possible with the toss of a coin: It is usually insufficient at the first step and the data for factors such as material properties or ground unevenness cannot be recorded. Therefore, we cannot have these ready for later analogy reasoning. This does in no way mean, however, that the event of a coin toss is coincidental, but merely shows that we humans are not able to measure these influence factors and describe the process. In other cases we may be quite able to measure the influence factors, but we are not able to correlate these purposefully, meaning that computing similarity or even describing the processes is impossible for us.

It is by no means the case that analogy reasoning is the only way of deducing forecasts for new situations from already known information. If the observer of a falling glass is asked how he knows that the glass will break, then the answer will often include things like "every time I have seen a glass fall from a height of more than 1.5 meters it has broken". There are two interesting points here: The relation to past experiences using the term "always" as well as the deduction of a rule from these experiences:

If the falling article is made of glass and the falling height is more than 1.5 meters, then the glass will break.

The introduction of a threshold value like 1.5 meters is a fascinating aspect of this rule formation. For although not every glass will break immediately if greater heights are used and will not necessarily remain intact in the case of lower heights, introducing this threshold value transforms the rule into a rule of thumb, which may not always, but will mostly lead to a correct estimate of the situation. Instead of therefore reasoning by analogy straight away, one could now use this rule of thumb and would soon reach a decision as to the most probable future of the falling article. Analogy reasoning and the creation of rules are two first examples of how humans, and also data mining methods, are able to anticipate the outcome of new and unknown situations.

Our description of what goes on in our brains and also in most data mining methods on the computer reveals yet another interesting insight: The analogy reasoning described does at no time require the knowledge of any physical formula to say why the glass will now break. The same applies for the rule of thumb described above. So even without knowing the complete (physical) description of a process, we and the data mining method are equally able to generate an estimation of situations or even predictions. Not only was the causal relationship itself not described here, but even the data acquisition was merely superficial and rough and only a few factors such as the material of the falling article (glass) and the falling height (approx. 2m) were indicated, and relatively inaccurately at that.

Causal chains therefore exist whether we know them or not. In the latter case, we are often inclined to refer to them as coincidental. And it is equally amazing that describing the further course is possible even for an unknown causal chain, and even in situations where the past facts are incomplete and only described inaccurately.

This section has given you an idea of the kind of problems we wish to address in this book. We will be dealing with numerous influence factors, some of which can only be measured insufficiently or not at all. At the same time, there are often so many of these factors that we risk losing track. In addition, we also have to deal with the events which have already taken place, which we wish to use for modeling and the number of which easily goes into millions or billions. Last but not least, we must ask ourselves whether describing the process is the goal or whether analogy reasoning is already sufficient to make a prediction. And in addition, this must all take place in a dynamic environment under constantly changing conditions—and preferably as soon as possible. Impossible for humans? Correct. But not impossible for data mining methods.

1.3 Applications of Data Mining

indexData Mining Applications Before we start to discuss the fundamental terms of data mining and predictive analytics, I would like to give you a feeling about the possible application fields for those techniques. There are literally hundreds of possible applications for data mining and predictive analytics across basically all verticals and horizontals you can think of. The following selection is therefore by no means complete; it should only give you some ideas where these technologies already have been successfully applied. The applications described in the following are grouped along selected verticals where they are most often used, but make sure to check them all since there is no reason why you should not use "fraud detection" also in retail instead of financial services.

1.3.1 Financial Services

- ***Fraud detection:*** Fraud detection is often used in the financial service industry but not only there. The basic idea here is to detect fraudulent activities among a very large set of transactions with methods from predictive analytics. Possible approaches include the detection of outliers or the modeling of "normal" cases against "fraudulent" cases and using this model for new transactions to check if they fall into the fraud segment.

- ***Churn prevention:*** Assume your insurance company has a contract with one of your customers. The optimal case for your insurance would that this contract stays and you keep an ongoing relationship with your customer. Sadly, some customers decide to quit the contract for a wide variety of reasons and you would like to know in advance for which customers this will happen with the highest likelihood in the near future. This is exactly the idea behind churn prevention: create predictive models calculating the probability that customers are likely to quit contracts soon so you can be pro-active, engage with them, offer incentives, etc. Besides the financial industry, churn prevention can also be often found in the retail industry, e-commerce, or telecommunications among others.

- ***Sentiment analysis:*** Sentiment analysis is not at all connected to the financial industry but we have seen it very often here lately. You can also find sentiment analysis

in industries like consumer goods, retail, telecommunications, and life sciences a lot. The idea behind sentiment analysis is to connect to thousands of online sources in the web, collect statements about your brands or products, and analyze them by means of text analytics with respect to their tonality or sentiment. You can identify how sentiment changes over time and measure the success of marketing or PR by this or you can get new insights about how to improve your products. If you combine this with network analysis you can even detect key opinion leaders and see how they influence their peer groups.

- ***Trading analytics:*** If you are trading, building portfolios, or preparing deals, the natural idea would be to calculate the success rate of your decisions with help of predictive analytics. In general, you could analyze potential trade opportunities by looking at market data or inspect markets for trading activity which showing an emerging trend. Lately, many analysts combine more traditional methods like time series analysis with behavioral trading algorithms or even sentiment analysis.
- ***Risk management:*** This is another example done in the financial services industry which can also be applied to many other industries as well, especially when it comes to supply chain management, for example in manufacturing, or to logistics and transportation. Data mining and predictive analytics can be used for solving multiple problems connected to risk management, including error detection and quantification, unit reviews or internal audits, detecting fraud (see above), identifying suppliers with highest probability of failure, quantifying payment risks, and credit scoring.

1.3.2 Retail and Consumer Products

- ***Customer segmentation, channel selection, and next best action:*** In retail you can easily find most applications of data mining for sales and marketing. A typical approach is to use data about customers, including descriptions of the customers' purchase and transaction history, for creating segments of customers, for example based on classification but more often based on clustering techniques. Those clusters can build data-driven segments much more optimized than the gut-driven A-B-C segments which even today can often be seen in practice. The assignment of customers to segments is an important prerequisite for further analysis, for example, for selecting customers for specific sales or marketing channels or for predicting which is the optimal next best action to approach those customers or leads.
- ***Direct marketing:*** The bread-and-butter business for data mining and one of the stories how it all has begun. The idea behind direct marketing is to assign costs for different types of actions: If I contact a lead and he does not react, this costs A (for the contact efforts, etc.). If he does, I will get some gain B (gains are essentially negative costs). If I don't contact the lead and he would not have reacted anyway, this saves me cost A above. But if I decide to not contact the lead and he or she would have purchases, this might cause a huge loss C. So the whole point is to identify those leads with the highest probabilities for conversion for a certain marketing campaign so that you only would contact the most likely cases up to a limit where the contacting costs would no longer lead to revenue high enough to compensate the costs. Even with the advent of e-mail, direct marketing is necessary, the "cost" here might be that recipients are tired of getting too much spam and might opt out instead so we are using the lead.
- ***Recommendations, cross-selling and up-selling:*** Another one of the big success

stories of data mining. Everybody who has already purchased a book at Amazon already came across the results of so-called recommender systems: "People who bought THIS book also purchased THAT one." At first sight, this problem type might not look too complicated: for each item, just search for those which have been purchased frequently together with the first one. The problem comes with the high number of available items and the high number of transactions typically available in retail and e-commerce. It is literally impossible to make those calculations for all combinations which might happen and so we need algorithms guiding our search to the most promising directions. We call those promising combinations "frequent item sets" and after we found those sets, we might want to recommend other items from those sets if the first one is added to the cart. This approach is called *cross-selling* and might complement or even replace traditional cross-selling approaches based on manual rules. Loosely connected to this is *up-selling* where we try to identify customers who are likely to purchase a higher-valued product or a larger quantity.

- ***Customer lifetime value***: Traditional systems for business intelligence based on OLAP approaches are great for answering questions like "who are the customers who bought most so far" or "what are the revenues created with my top 10 customers in the past". Although this is with no doubt important information, it unfortunately is also only reflecting the past. Previously good customers might change to another supplier or drop out for other reasons, and the fact that a customer has created much revenue so far does not mean this will continue also in the future. Instead of analyzing the historical customer value, many organizations now turn to predict how customers will develop in the future and what their total customer lifetime value will be to guide their sales and marketing efforts. Predictive analytics methods help to identify typical customer value cycles as well as to assign customers and leads to those cycles and determining at which state within their cycle they are.

1.3.3 Telecommunications and Media

- ***Network analysis:*** The telecommunications industry actually already is a driver for many of the application fields described above including churn prevention, channel selection, direct marketing, customer lifetime value, and sentiment analysis. One interesting finding is that the telecommunication industry in addition can make use of another data source describing social interactions between their customers. For example, the calls between customers might describe their social connections and describing those connections in addition to their usage behavior and transactional data can easily help to improve models for churn prevention and others. If a key opinion leader decides to change to another provider, other people influenced by this person might show a higher probability to also quit their contracts as well.
- ***Customer service process automation:*** Many companies spend a lot of efforts to improve customer support and they are right to do so. If a customer is happy with the support in case of a previously bad experience, this person might be turned to a more loyal customer than before. One of the most important influence factors for the happiness with customer service is the amount of time between when a request is sent and when the answer is delivered. Text analytics can help to either answer a question with an automatically selected text block or at least by assigning the request to the right employee without any further delay. This is another example not only applicable to the telecommunications industry but to all customer-centric businesses with many customer contacts.

1.3.4 Manufacturing, Construction, and Electronics

- ***Predictive maintenance:*** Predictive analytics can analyze sensor data right from the production processes and machines to determine if there is an upcoming problem with a machine which is likely to lead to a problem soon. Many problems manifest themselves at early stages already with certain types of error messages or changed behavior ultimately leading to changed sensor data. Analysts can create predictive models based on the failure events in the past and the historical data before those failures. Such a model can then be used to predict if a new failure is likely to happen before the next maintenance interval and should be addressed better now than later. Those models could also deliver insights into the reasons of those failures and hence deliver a root cause analysis.

- ***Patent text analysis:*** Another of those examples which is actually also applicable to other industries is the analysis of patent texts. This is most often done by methods derived from text analysis and text similarities for one of two reasons: either a company would like to detect emerging trends as soon as possible or it would like to be prepared for cases in which own patents are attacked.

- ***Supply chain management:*** There are multiple application fields for predictive analytics around supply chain management. We already have discussed risk management above, for supply chain management this could mean determining which suppliers have the highest risks for failure and what would be the expected impact in case of failure. Predictive analytics can also be used for demand forecasting and hence for improving logistics but also timely negotiations with suppliers. And finally those techniques can be used to predict prices and their changes in the supply chain, again allowing for pro-active and well-informed decisions.

- ***Optimizing throughput rates:*** The manufacturing industry even has started to connect data mining and predictive analytics with the control centers of their factories. Based on the sensor data describing the production process itself and also the input to this process, those models find the optimal settings for the process in real time in order to optimize for quality, or higher throughput rates, or even both at the same time. In some of those cases it is important not to leave certain parameter ranges in order to not damage the involved machines and even this is possible to do with the help of advanced analytics.

- ***Quality assurance:*** Another application field is the prediction of the quality of the outcome of a process even before the full process has been finished. Those predictive models use the data describing the process and combine it with sensor data describing the current state of an item in order to predict the quality of the final outcome. We even have seen cases where the item was taken out of further refinement, which would just have induced additional costs for a product which will not be sold due to quality restrictions anyway. Closely connected to this are questions like anomaly detection and why a certain item has a lower quality, hence a root cause analysis.

1.4 Fundamental Terms

We managed so far to get the general idea about what are data mining and predictive analytics about. Plus we have a good feeling that those technologies are very useful for many application fields across basically all industries. In the following, we will introduce some fundamental terms which will make dealing with the problems described later in this book easier for us. You will come across these terms again and again in the RapidMiner software too, meaning it is worth becoming acquainted with the terms used even if you are already an experienced data analyst.

First of all, we can see what the two examples looked at in the previous sections, namely the toss of a coin and the falling glass, have in common. In our discussion on whether we are able to predict the end of the respective situation, we realized that knowing the influence factors as accurately as possible, such as material properties or the nature of the ground, is important. And one can even try to find an answer to the question as to whether this book will help you by recording the characteristics of yourself, the reader, and aligning them with the results of a survey of some of the past readers. These measured reader characteristics could be, for example, the educational background of the person concerned, the liking of statistics, preferences for other, possibly similar books and further features, which we could also measure as part of our survey. If we now knew such characteristics of 100 readers and had the indication as to whether you like the book or not in addition, then the further process would be almost trivial. We would also ask you the questions from our survey and measure the same features in this way and then, for example using analogy reasoning as described above, generate a reliable prediction of your personal taste. "Customers who bought this book also bought. . . ." This probably rings a bell.

1.4.1 Attributes and Target Attributes

Whether coins, customers, or production processes, there is, as previously mentioned, the question in all scenarios as to the characteristics or features of the respective situation. We will always speak of **attributes** in the following when we mean such describing factors of a scenario. This is also the term that is always used in the RapidMiner software when such describing features arise. There are many synonyms for this term and depending on your own background you might have already come across different terms instead of "attribute", for example

- characteristic,
- feature,
- influence factor (or just factor),
- indicator,
- variable, or
- signal.

We have seen that description by attributes is possible for processes and also for situations. This is necessary for the description of technical processes for example and the thought of the falling glass is not too far off here. If it is possible to predict the outcome of such a situation, then why not also the quality of a produced component? Or the imminent

failure of a machine? Other processes or situations which have no technical reference can also be described in the same way. How can I predict the success of a sales or marketing promotion? Which article will a customer buy next? How many more accidents will an insurance company probably have to cover for a particular customer or customer group?

We shall use such a customer scenario in order to introduce the remaining important terms. Firstly, because humans are famously better at understanding examples about other humans and secondly, because each enterprise probably has information, i.e., attributes, regarding their customers and most readers can therefore relate to the examples immediately. The attributes available as a minimum, which just about every enterprise keeps about its customers, are for example address data and information as to which products or services the customer has already purchased. You would be surprised what forecasts can be made even from such a small number of attributes.

Let us look at an (admittedly somewhat contrived) example. Let us assume that you work in an enterprise that would like to offer its customers products in the future which are better tailored to their needs. Within a customer study of only 100 of your customers some needs became clear, which 62 of these 100 customers share all the same. Your research and development department got straight to work and developed a new product within the shortest time, which would satisfy these new needs better. Most of the 62 customers with the relevant needs profile are impressed by the prototype in any case, although most of the remaining participants of the study only show a small interest as expected. Still, a total of 54 of the 100 customers in the study said that they found the new product useful. The prototype is therefore evaluated as successful and goes into production—now only the question remains as to how, from your existing customers or even from other potential customers, you are going to pick out exactly the customers with whom the subsequent marketing and sales efforts promise the greatest success. You would therefore like to optimize your efficiency in this area, which means in particular ruling out such efforts from the beginning which are unlikely to lead to a purchase. But how can that be done? The need for alternative solutions and thus the interest in the new product arose within the customer study on a subset of your customers. Performing this study for all your customers is much too costly and so this option is closed to you. And this is exactly where data mining can help. Let us first look at a possible selection of attributes regarding your customers:

- Name
- Address
- Sector
- Subsector
- Number of employees
- Number of purchases in product group 1
- Number of purchases in product group 2

The number of purchases in the different product groups means the transactions in your product groups which you have already made with this customer in the past. There can of course be more or less or even entirely different attributes in your case, but this is irrelevant at this stage. Let us assume that you have the information available regarding these attributes for every one of your customers. Then there is another attribute which we can look at for our concrete scenario: Whether the customer likes the prototype or not. This attribute is of course only available for the 100 customers from the study; the information on

this attribute is simply unknown for the others. Nevertheless, we also include the attribute in the list of our attributes:

- *Prototype positively received?*
- Name
- Address
- Sector
- Subsector
- Number of employees
- Number of purchases in product group 1
- Number of purchases in product group 2

If we assume you have thousands of customers in total, then you can only indicate whether 100 of these evaluated the prototype positively or not. You do not yet know what the others think, but you would like to! The attribute "prototype positively received" thus adopts a special role, since it identifies every one of your customers in relation to the current question. We therefore also call this special attribute a **label**, since it sticks to your customers and identifies them like a brand label on a shirt or even a note on a pin board. You will also find attributes which adopt this special role in RapidMiner under the name "label". The goal of our efforts is to fill out this particular attribute for the total quantity of all customers. We might therefore also speak of the **target attribute** instead of the term "label" since our target is to create a model which predicts this special attribute from the values of all others. You will also frequently discover the term **target variable** in the literature, which means the same thing.

1.4.2 Concepts and Examples

The structuring of your customers' characteristics by attributes, introduced above, already helps us to tackle the problem a bit more analytically. In this way we ensured that every one of your customers is represented in the same way. In a certain sense we defined the type or **concept** "customer", which differs considerably from other concepts such as "falling articles" in that customers will typically have no material properties and falling articles will only rarely buy in product group 1. It is important that, for each of the problems in this book (or even those in your own practice), you first define which concepts you are actually dealing with and which attributes these are defined by.

We implicitly defined above, by indicating the attributes name, address, sector, etc., and in particular the purchase transactions in the individual product groups, that objects of the concept "customer" are described by these attributes. Yet this concept has remained relatively abstract so far and no life has been injected into it yet. Although we now know in what way we can describe customers, we have not yet performed this for specific customers. Let us look at the attributes of the following customer, for example:

- Prototype positively received: yes
- Name: Miller Systems Inc.
- Address: 2210 Massachusetts Avenue, 02140 Cambridge, MA, USA

- Sector: Manufacturing
- Subsector: Pipe bending machines
- Number of employees: >1000
- Number of purchases in product group 1: 5
- Number of purchases in product group 2: 0

We say that this specific customer is an **example** for our concept "customer". Each example can be characterized by its attributes and has concrete values for these attributes which can be compared with those of other examples. In the case described above, Miller Systems Inc. is also an example of a customer who participated in our study. There is therefore a value available for our target attribute "prototype positively received?" Miller Systems Inc. was happy and has "yes" as an attribute value here, thus we also speak of a **positive example**. Logically, there are also negative examples and examples which do not allow us to make any statement about the target attribute.

1.4.3 Attribute Roles

We have now already become acquainted with two different kinds of attributes, i.e., those which simply describe the examples and those which identify the examples separately. Attributes can thus adopt different roles. We have already introduced the role "label" for attributes which identify the examples in any way and which must be predicted for new examples that are not yet characterized in such a manner. In our scenario described above, the label still describes (if present) the characteristic of whether the prototype was received positively.

Likewise, there are for example roles, the associated attribute of which serves for clearly identifying the example concerned. In this case the attribute adopts the role of an identifier and is called **ID** for short. You will find such attributes named ID in the RapidMiner software also. In our customer scenario, the attribute "name" could adopt the role of such an identifier.

There are even more roles, such as those with an attribute that designates the weight of the example with regard to the label. In this case the role has the name **Weight**. Attributes without a special role, i.e., those which simply describe the examples, are also called **regular** attributes and just leave out the role designation in most cases. Apart from that you have the option in RapidMiner of allocating your own roles and of therefore identifying your attributes separately in their meaning. Please note that for most data mining tasks in RapidMiner the regular attributes are used as input to the method, for example, to create a model predicting the attribute with role label.

1.4.4 Value Types

As well as the different roles of an attribute, there is also a second characteristic of attributes which is worth looking at more closely. The example of Miller Systems Inc. above defined the respective values for the different attributes, for example "Miller Systems Inc." for the attribute "Name" and the value "5" for the number of past purchases in product group 1. Regarding the attribute "Name", the concrete value for this example is therefore arbitrary free text to a certain extent; for the attribute "number of purchases in product group 1" on the other hand, the indication of a number must correspond. We call the indication whether the values of an attribute must be in text or numbers the **Value Type** of an attribute.

In later chapters we will become acquainted with many different value types and see how these can also be transformed into other types. For the moment we just need to know that there are different value types for attributes and that we speak of value type **text** in the case of free text, of the value type **numerical** in the case of numbers and of the value type **nominal** in the case of only few values being possible (like with the two possibilities "yes" and "no" for the target attribute).

Please note that in the above example the number of employees, although really of numerical type, would rather be defined as nominal, since a size class, i.e., ">1000" was used instead of an exact indication like 1250 employees.

1.4.5 Data and Meta Data

We want to summarize our initial situation one more time. We have a Concept "customer" available, which we will describe with a set of attributes:

- Prototype positively received? [Label; Nominal]
- Name [Text]
- Address [Text]
- Sector [Nominal]
- Subsector [Nominal]
- Number of employees [Nominal]
- Number of purchases in product group 1 [Numerical]
- Number of purchases in product group 2 [Numerical]

The attribute "Prototype positively received?" has a special **Role** among the attributes; it is our **Target Attribute** or **Label** here. The target attribute has the value type **Nominal**, which means that only relatively few characteristics (in this case "yes" and "no") can be accepted. Strictly speaking it is even binominal, since only two different characteristics are permitted. The remaining attributes all have no special role, i.e., they are **regular** and have either the value type **Numerical** or **Text**. The following definition is very important, since it plays a crucial role in a successful professional data analysis: This volume of information, which describes a concept, is also called meta data, since it represents data about the actual data.

Our fictitious enterprise has a number of Examples for our concept "customer", i.e., the information which the enterprise has stored for the individual attributes in its customer database. The goal is now to generate a prediction instruction from the examples for which information is available concerning the target attribute, which predicts for us whether the remaining customers would be more likely to receive the prototype positively or reject it. The search for such a prediction instruction is one of the tasks which can be performed with data mining.

However, it is important here that the information for the attributes of the individual examples is in an ordered form, so that the data mining method can access it by means of a computer. What would be more obvious here than a table? Each of the attributes defines a column and each example with the different attribute values corresponds to a row of this table. For our scenario this could look like Table 1.1 for example.

We call such a table an **Example Set**, since this table contains the data for all the attributes of our examples. In the following, and also within RapidMiner, we will use the

TABLE 1.1: A set of examples with values for all attributes together with label values.

Prototype positively received?	**Name**	**Address**	**Sector**	**Sub-sector**	**Number of employees**	**No of purchases group 1**	**No of purchases group 2**	**...**
Yes	Miller Systems Inc.	2210 Massachusetts Ave, Cambridge, MA, USA	Manufacturing	Pipe bending machines	>1000	5	0	...
?	Smith Paper	101 Huntington Ave, Boston, MA, USA	IT	Supplies	600-1000	3	7	...
No	Meyer Inc.	1400 Commerce St, Dallas, TX, USA	Retail	Textiles	<100	1	11	...
...	...	...	...	...	...	...	...	...

terms **Data**, **Dataset**, and **Example Set** synonymously. A table with the appropriate entries for the attribute values of the current examples is always meant in this case. It is also such data tables which have taken their name from data analysis or data mining. Note:

Data *describes the objects of a concept,* ***Meta Data*** *describes the characteristics of a concept (and therefore also of the data).*

Most data mining methods expect the examples to be given in such an attribute value table. Fortunately, this is the case here and we can spare ourselves any further data transformations. In practice, however, this is completely different and the majority of work during a data analysis is time spent transferring the data into a format suitable for data mining. These transformations are therefore dealt with in detail in later chapters when we are discussing the different use cases.

1.4.6 Modeling

Once we have the data regarding our customers available in a well-structured format, we can then finally replace the unknown values of our target attribute with the prediction of the most probable value by means of a data mining method. We have numerous methods available here, many of which, just like the analogy reasoning described at the beginning or the generating of rules of thumb, are based on human behavior. We call the use of a data mining method "**to model**" and the result of such a method, i.e., the prediction instruction, is a **model**. Just as data mining can be used for different issues, this also applies for models. They can be easy to understand and explain the underlying processes in a simple manner. Or they can be good to use for prediction in the case of unknown situations. Sometimes both apply, such as with the following model for example, which a data mining method could have supplied for our scenario:

If the customer comes from urban areas, has more than 500 employees and if at least 3 purchases were transacted in product group 1, then the probability of this customer being interested in the new product is high.

Such a model can be easily understood and may provide a deeper insight into the underlying data and decision processes of your customers. And in addition, it is an **operational model**, i.e., a model which can be used directly for making a prediction for further customers. The company "Smith Paper" for example satisfies the conditions of the rule above and is therefore bound to be interested in the new product—at least there is a high probability of this. Your goal would therefore have been reached and by using data mining you

would have generated a model which you could use for increasing your marketing efficiency: Instead of just contacting all existing customers and other candidates without looking, you could now concentrate your marketing efforts on promising customers and would therefore have a substantially higher success rate with less time and effort. Or you could even go a step further and analyze which sales channels would probably produce the best results and for which customers.

In the rest of this book we will focus on further uses of data mining and at the same time practice transferring concepts such as customers, business processes, or products into attributes, examples, and datasets. This will train the eye to detect further possibilities of application tremendously and will make analyst life much easier for you later on. In the next chapter though, we would like to spend a little time on RapidMiner and give a small introduction to its use, so that you can implement the following examples immediately.

Chapter 2

Getting Used to RapidMiner

Ingo Mierswa

Rapid-I, Dortmund, Germany

2.1 Introduction

We will use the open source data mining solution RapidMiner for the practical exercises in this book and for demonstrating the use cases for data mining, predictive analytics, and text mining discussed here. Before we can work with RapidMiner, you of course need to download and install the software first. You will find RapidMiner in the download area of the Rapid-I website: `http://www.rapid-i.com`. If you want to follow the use cases in this book, it is highly recommended to download the appropriate installation package for your operating system and install RapidMiner according to the instructions on the website now. All usual Windows versions are supported as well as Macintosh, Linux or Unix systems. Please note that an up-to-date Java Runtime Environment (JRE) with at least version 7 is needed for all non-Windows systems. You can find such a JRE for example at `http://www.java.com/`.

2.2 First Start

If you are starting RapidMiner for the first time, you will be welcomed by the so-called Welcome Perspective. The lower section shows current news about RapidMiner, if you have an Internet connection. The list in the center shows the analysis processes recently worked on. This is useful if you wish to continue working on or execute one of these processes. You can open a process from this list to work on or execute it simply by double clicking. The upper section shows typical actions which you as an analyst will perform frequently after starting RapidMiner. Here are the details of these:

1. New Process: Starts a new analysis process. This will be the most often used selection

for you in the future. After selecting this, RapidMiner will automatically switch to the Design perspective (explained below).

2. Open Recent Process: Opens the process which is selected in the list below the actions. Alternatively, you can open this process by double-clicking inside the list. Either way, RapidMiner will then automatically switch to the Design Perspective.

3. Open Process: Opens the repository browser and allows you to select a process to be opened within the process Design Perspective.

4. Open Template: Shows a selection of different pre-defined analysis processes, which can be configured in a few clicks.

5. Online Tutorial: Starts a tutorial which can be used directly within RapidMiner and gives an introduction to some data mining concepts using a selection of analysis processes. Recommended if you have a basic knowledge of data mining and are already familiar with the fundamental operation of RapidMiner.

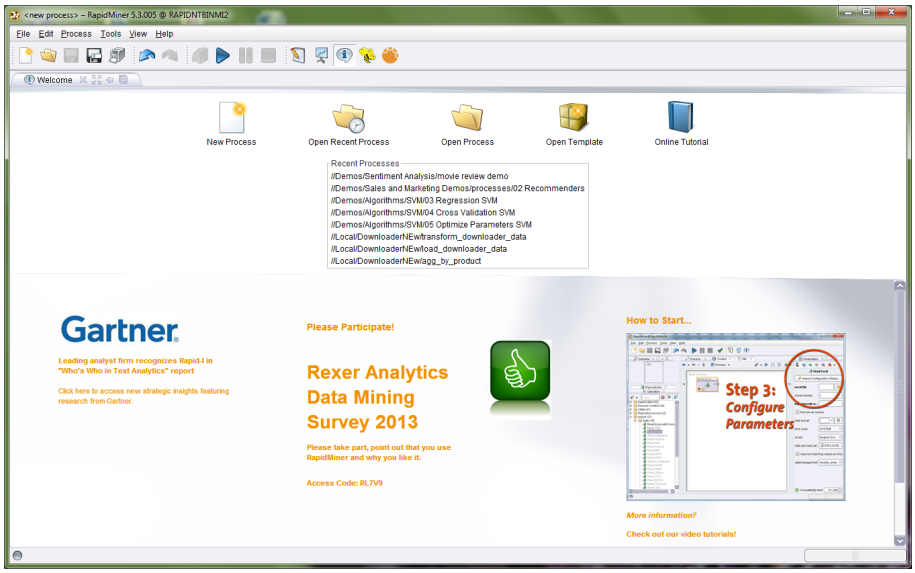

FIGURE 2.1: Welcome Perspective of RapidMiner.

At the right-hand side of the toolbar inside the upper section of RapidMiner, you will find three icons which switch between the individual RapidMiner perspectives. A **perspective** consists of a freely configurable selection of individual user interface elements, the so-called views. These can also be arranged however you like. In the Welcome Perspective there is only one view; one preset at least, namely the welcome screen, which you are looking at now. You can activate further views by accessing the "View" menu. Sometimes you may inadvertently delete a view or the perspective is unintentionally moved into particularly unfavorable positions. In this case the "View" menu can help, because apart from the possibility of reopening closed views via "Show View", the original state can also be recovered at any time via "Restore Default Perspective".

2.3 Design Perspective

You will find an icon for each (pre-defined) perspective within the right-hand area of the toolbar:

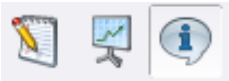

FIGURE 2.2: Toolbar icons for perspectives.

The icons shown here take you to the following perspectives:

1. *Design Perspective:* This is the central RapidMiner perspective where all analysis processes are created and managed.
2. *Result Perspective:* If a process supplies results in the form of data, models, or the like, then RapidMiner takes you to this Result Perspective, where you can look at several results at the same time as normal thanks to the views.
3. *Welcome Perspective:* The Welcome Perspective already described above, in which RapidMiner welcomes you with after starting the program.

You can switch to the desired perspective by clicking inside the toolbar or alternatively via the menu entry "View" – "Perspectives" followed by the selection of the target perspective. RapidMiner will eventually also ask you automatically if switching to another perspective seems a good idea, e.g., to the Result Perspective on completing an analysis process.

Now switch to the Design Perspective by clicking in the toolbar. This is the major working place for us while using RapidMiner. Since the Design Perspective is the central working environment of RapidMiner, we will discuss all parts of the Design Perspective separately in the following and discuss the fundamental functionalities of the associated views. In any case, you should now see the screen on the next page:

All work steps or building blocks for different data transformation or analysis tasks are called **operators** in RapidMiner. Those operators are presented in groups in the Operator View on the left side. You can navigate within the groups in a simple manner and browse in the operators provided to your heart's desire. If RapidMiner has been extended with one of the available extensions, then the additional operators can also be found here. Without extensions you will find at least the following groups of operators in the tree structure:

- *Process Control:* Operators such as loops or conditional branches which can control the process flow.
- *Utility:* Auxiliary operators which, alongside the operator "Subprocess" for grouping subprocesses, also contain the important macro-operators as well as the operators for logging.
- *Repository Access:* Contains the two operators for read and write access in repositories.
- *Import:* Contains a large number of operators in order to read data and objects from external formats such as files, databases, etc.

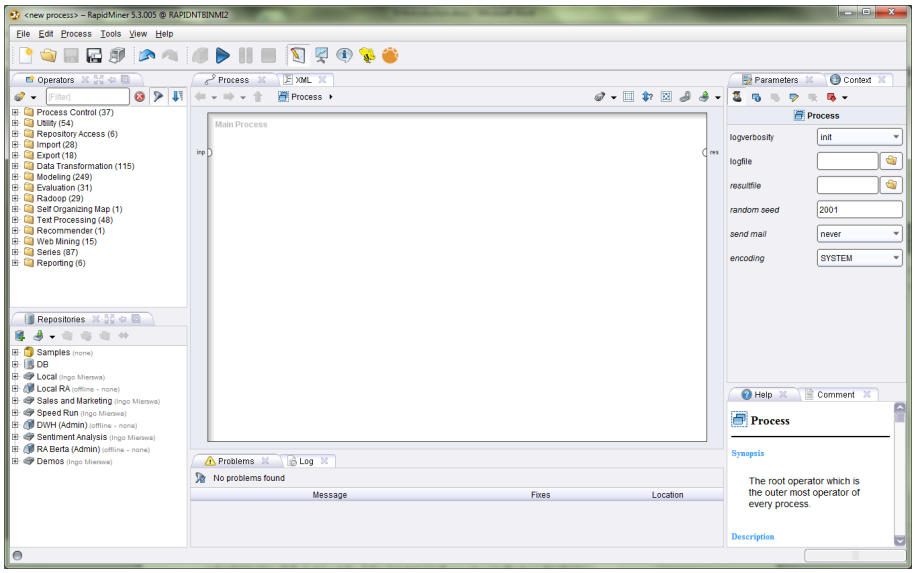

FIGURE 2.3: Design Perspective of RapidMiner.

- *Export:* Contains a large number of operators for writing data and objects into external formats such as files, databases, etc.
- *Data Transformation:* Probably the most important group in the analysis in terms of size and relevance. All operators are located here for transforming both data and meta data.
- *Modeling:* Contains the actual data mining process, such as classification methods, regression methods, clustering, weightings, methods for association rules, correlation and similarity analyses as well as operators, in order to apply the generated models to new datasets.
- *Evaluation:* Operators using which one can compute the quality of a modeling and thus for new data, e.g., cross-validations, bootstrapping, etc.

You can select operators within the Operators View and add them in the desired place in the process by simply dragging them from the Operators View and dropping them into the large white area in the center, the so-called Process View. Every analysis in RapidMiner is a **process**, and every process consists of one or several steps which are the operators.

Depending on your settings, those new operators might be directly connected with existing operators as suitably as possible on the basis of the available meta data information. If this is not happening or the automatically inserted connection is not desired, you can delete the connection by selecting them and pressing the Delete key or by pressing the Alt key while clicking on any of the connection **ports**. Ports are the round bubbles on the sides of the operators and they are used to define the data flow through your analytical processes. You can insert new connections by either clicking on the source port and then clicking again on a target port or by dragging a line between those ports.

Later on, when you have successfully defined your first RapidMiner processes, a typical result might look like in the image on the following page:

You could now simply try and select a few operators from the Operator View and drag them into the Process View. Connect their ports, even if this is probably not leading

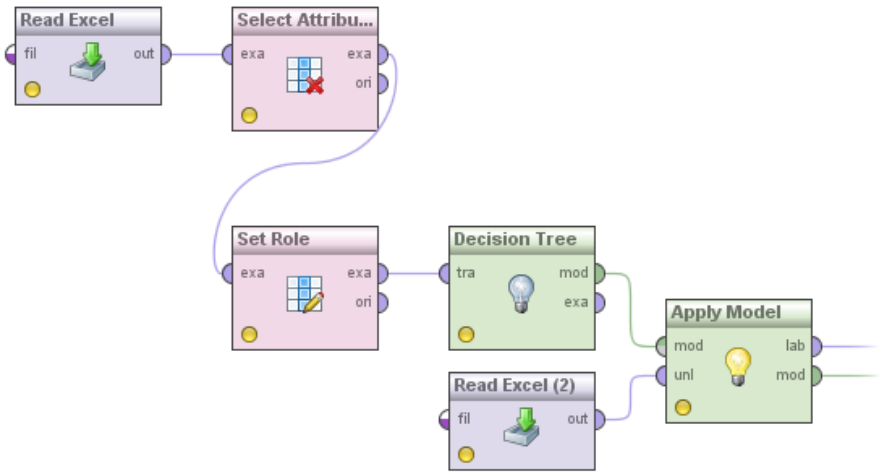

FIGURE 2.4: A typical process in RapidMiner consists of several operators. The data flow is defined by the connected ports, for example, the "Read Excel" operator loads a dataset from an Excel file and delivers it to the succeeding "Select Attributes" operator.

to working processes, and get familiar with the user interface of RapidMiner. In order to edit parameters you must select an individual operator. You will recognize the operator currently selected by its orange frame as well as its shadow. If you wish to perform an action for several operators at the same time, for example moving or deleting, please select the relevant operators by dragging a frame around these. In order to add individual operators to the current selection or exclude individual operators from the current selection, please hold the CTRL key down while you click on the relevant operators or add further operators by dragging a frame. You can also move operators around by selecting them and dragging them in the Process View. You will notice that the parameters in the **Parameter View** on the right side of RapidMiner changes sometimes if you select different operators. As you can see, most operators provide a set of parameters which control the actual working mode of the respective operator. You will find much information about RapidMiner and its user interface in the RapidMiner User Manual available at

`http://docs.rapid-i.com`

2.4 Building a First Process

One of the first steps in a process for data analysis is usually to load some data into the system. RapidMiner supports multiple methods for accessing datasets. It supports more than 40 different file types and of course all major database systems. If the data is not originally stored in a relational database system, the best approach is to import the data first into the RapidMiner repository. Please follow the instructions from the RapidMiner manual for more information or just try to import a dataset, for example an Excel file, with "File" – "Import Data". Later you will also realize that there are dozens of operators for data import in the operator group "Import", which can also be used directly as part of the process.

2.4.1 Loading Data

In the following we assume that you have managed to import the data into the RapidMiner repository and hence we will retrieve the data from there. If you are loading the data from a database or file, your following steps are at least similar to those described below.

It is always recommended to use the repository whenever this is possible instead of files. This will allow RapidMiner to get access to the describing meta data and will ease process design a lot. We will now create the beginning of a data analysis process and will add a first data mining technique using this data. The very first operation in our process should be to load the data from the repository again in order to make it available for the next analysis steps:

1. Go to the Repositories view and open the repository **Samples** delivered with RapidMiner. Click on the small plus sign in front of this repository. You should now see two folders named **data** and **processes**. Open the data folder and you will find a collection of datasets coming together with RapidMiner. Click on the dataset named **Iris** and drag it onto the large white view named **Process** in the center of your frame. After releasing the dataset somewhere on the white area, it should be transformed into an operator named **Retrieve** with a bluish output port on the right side. RapidMiner automatically has transformed the dataset into an operator loading the dataset. If you click on the operator, you can see a parameter in the **Parameters** view pointing to the data location. The Retrieve operator in general, well, retrieves objects from a repository and makes them available in your process.

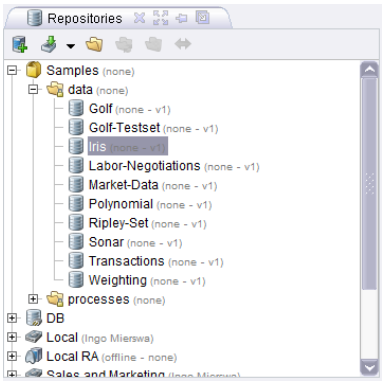

FIGURE 2.5: Drag the Iris dataset into the process view in order to create a new operator loading this dataset during process execution.

Background Information – Iris Data:
You might already be familiar with the "Iris" dataset since it is a well-known dataset among many data analysts. If not, here is the basic idea: the dataset describes 150 Iris plants with four attributes: sepal-length, sepal-width, petal-length, and petal-width. Since only experts for Iris plants actually understand the meaning of those attributes we will refer to those attributes with "a1" to "a4". And there is a fifth column describing to which class of Irises each of the 150 plants belong. There are three options: Iris-setosa, Iris-versicolor, and Iris-virginica. Each of those three classes is represented by 50 plants in the dataset. The goal is now to find a classification model using the measured lengths and widths (a1 to a4) to predict the class of the plant. This would

allow the classification of those plants even by non-experts like myself.

2. Maybe the output was automatically connected to one of the result ports named **res** on the right side of the Process area. If that is the case, you are done already. If not—and this depends on your current program settings—click on the output port of the Retrieve operator and then click on the first res port on the right side. Alternatively, you can also drag a connection line between the two ports. Your process should now look like the following:

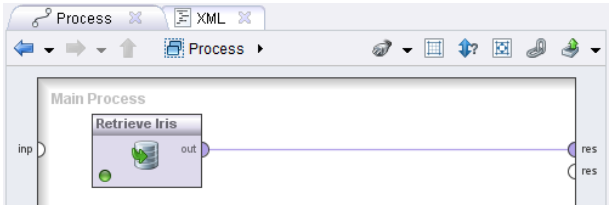

FIGURE 2.6: The probably most simple process which can be created with RapidMiner: It just retrieves data from a repository and delivers it as a result to the user to allow for inspection.

All results which are delivered at a result port named **res** will be delivered as a result to the user (or other systems if the process execution is integrated into other systems). The next step would be to create a decision tree on the Iris dataset and also deliver this model to the user.

2.4.2 Creating a Predictive Model

We have seen above how we create a new process just loading a dataset. The next step will be to create a predictive model using this dataset. This model predicts a categorical or nominal value, hence we could also say the model should describe rules which allow us to assign one of the three classes to new and unseen data describing new plants. We refer to this type of modeling as **classification**.

Adding a modeling technique to your process so that it calculated a predictive model is actually very easy. Just follow the following steps for creating such a model:

Go to the Operators view and open the operator group **Modeling, Classification, and Regression**, and then **Tree Induction**. You should now see an operator named **Decision Tree**. Click on it and drag it to your process, somewhere to the right of your initial retrieve operator.

You now only have to create the necessary connections. The dataset should be delivered to the modeling operator which is delivering a model then. However, you can also deliver the dataset itself to the user if you also connect the data port with one of the result ports. The complete process should look like the following figure.

In the next section we will learn how to execute this process and how to inspect the created results.

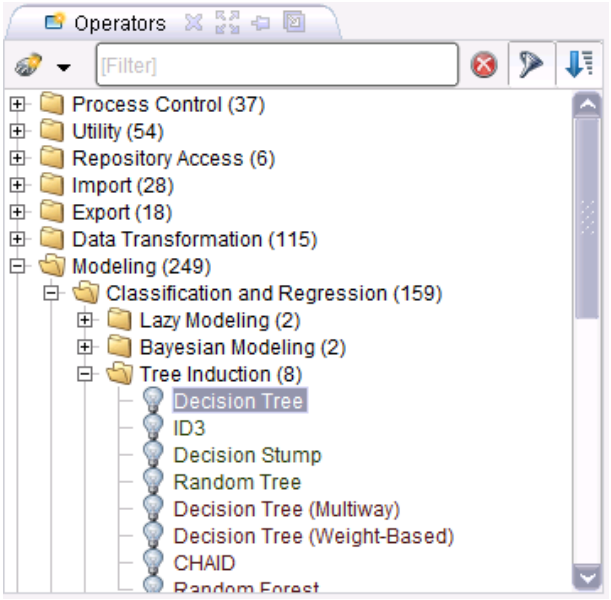

FIGURE 2.7: Drag the operator named "Decision Tree" into your process.

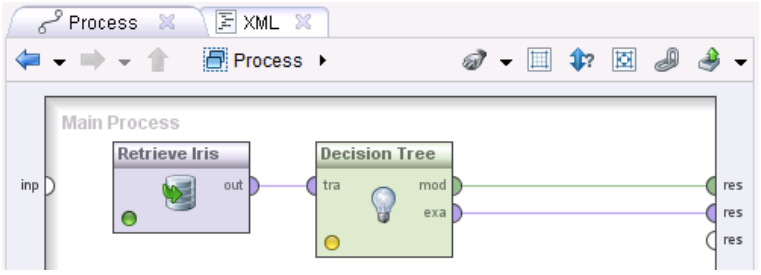

FIGURE 2.8: The complete process consisting of data loading and model creation.

Background Information: Decision Trees

Decision trees are probably one of the most widely used techniques in data mining. One of the biggest advantages is that they are easy to use and, maybe even more important, also easy to understand even by non-experts. The basic idea behind decision trees is a so-called divide-and-conquer approach. In each step the dataset is divided into different parts while each part should better represent one of the possible classes. The final result will be a tree-structure where each inner node represents a test for the value of a particular attribute and each leaf is representing the decision for a particular class. A new and unknown case is then routed down the tree until it reaches one of the leaves.

For each node we have two options depending on the value type of the attribute used at this node. For nominal attributes, the number of children is usually equal to the number of possible values for this attribute. If we are using a nominal attribute for a test in one of the inner nodes, this means that the dataset is at this stage basically divided according to the different values of this attribute. Hence, a nominal attribute will not get tested more than once since all examples further down the tree will have the same value for this particular attribute. This is different for numerical attributes: Here, we usually test if the attribute value is greater or less than a determined constant. The attribute may get tested several times for different constants.

The strategy for the construction of a decision tree is top-down in a recursive divide-and-conquer fashion. The first step is to select one of the attributes for the root node. Then we create a branch for each possible attribute value and split instances into subsets according to the possible values, i.e.,we will get one subset for each branch extending from the node. Finally we will repeat these steps recursively for each branch but only use instances that reach the branch now. We can stop this process if all instances have the same class.

The major question is: How should we select which attribute should be tested next? Our goal is to get the smallest tree possible since we believe that a small tree manages to explain the data best and in a general fashion leading to fewer errors for unseen data compared to a more complex tree. There is not a strict optimal solution for this task but a widely used heuristic: we should choose the attribute that produces the "purest" subsets of data with respect to the label attribute. A very popular choice for this type of measurement is the so-called ***information gain****. The information gain basically increases with the average purity of the subsets and hence we should choose the attribute which gives us the greatest information gain at each level.*

We will now discuss how we can measure information at all in order to be able to calculate which decision delivers the highest gain in information. We will see now that information can be easily measured in bits and that a concept exists for calculating this amount: the ***entropy****. The entropy measures the information required in bits (can also mean fractions of bits) to predict the outcome of an event if the probability distribution of this event is given. The formula is as follows:*

$$\textit{Entropy}\ (p_1,\ldots,p_n) = -p_1\ \textit{ld}\ (p_1) - \ldots - p_n\ \textit{ld}\ (p_n)$$

*p_1 to p_n are the possible probabilities for all outcomes while ld is the logarithm dualis, i.e.,the logarithm with base 2. The formula is easy to understand with a small example. Let's assume we are tossing an unbiased coin where each side can be shown with a probability of 0.5. In this case the entropy is –0.5 * ld(0.5) – 0.5 * ld(0.5) = 1. The necessary information to decide which side of a tossed coin is actually shown is hence 1 bit, which is the perfect amount for a binary decision with only two possible outcomes which are equally likely.*

*Now let's assume that the coin is biased and shows "head" with a probability of 75% and "tails" with 25%. In this case the entropy would be calculated as –0.75 * ld(0.75) – 0.75 **

ld(0.75) = 0.81 and hence we need less information to decide. This is only natural since we already expected "head" with a higher probability.

Algorithms for creating decision trees make use of the notion of entropy for selecting the optimal attribute for each split, i.e.,inner node. The entropy for the class distribution is first calculated for each possible value of a nominal attribute. Let's do this for an example and assume that we have an attribute with two possible values A and B. We have 10 examples with value A and 20 examples with value B. For those with value A we have 3 examples with class X and 7 examples with class Y. And for those with value B we have 15 examples with class X and 5 examples with class Y. We can now calculate the entropy values based on those class distributions in the subsets defined by those examples having values A and B respectively for this attribute:

1. ***Entropy A → entropy of class distribution (3/10, 7/10) = 0.881 bits***
2. ***Entropy B → entropy of class distribution (16/20, 4/20) = 0.322 bits***

For numerical attributes, each possible threshold for a numerical split is tested instead. In both cases the total information needed given a specific attribute is the weighted average of all entropy values; in this case this would be

(10 * Entropy A + 20 * Entropy B) / 30 = 0.508.

In order to calculate the ***information gain*** *by this split, we have to compare the entropy of the class distribution before the split with the entropy after the split, i.e.,in our example:*

gain = info (19/30,11/30) – weighted entropy = 0.948 – 0.508 = 0.44

The information gain for performing this split hence is 0.44. We can now calculate the gain for all possible attributes and compare all gains to find the one delivering the highest value. This attribute would then be used for the next split. The dataset is divided accordingly and the complete process is repeated recursively until all subsets only contain examples for a single class.

The section above has explained the basic ideas behind decision trees, but be assured that there is more about decision trees beyond this basic approach. Instead of the information gain, one could use different measurements including the so-called ***gain ratio*** *which prefers splits with less possible branches. Another important addition is* ***pruning*** *the tree after the growing phase in order to make it more robust for unseen data points. Please refer to the standard literature about data mining for getting more information about additions to the basic decision tree algorithm.*

2.4.3 Executing a Process

Now we are ready and want to execute the process we have just created for the first time. The status indicators of all used operators should now be yellow or green (the small traffic lights in each operator box) in the best case, and there should be no entries in the Problems View before you start executing a process. In such a case it should be possible to execute our process currently consisting of only one operator without any problems. However, the problems in the Problems view and also the red traffic lights only indicate that there might be a potential problem—you might be able to execute a process even if RapidMiner detects a potential problem. Just execute the process and check if it works despite the complaint as described below.

You have the following options for starting the process:

1. Press the large play button in the toolbar of RapidMiner.
2. Select the menu entry "Process" → "Run".
3. Press F11.

FIGURE 2.9: Press the Play icon in order to execute your process.

While a process is running, the status indicator of the operator being executed in each case transforms into a small green play icon. This way you can see what point the process is currently at. After an operator has been successfully executed, the status indicator then changes and stays green—until for example you change a parameter for this operator: Then the status indicator will be yellow. The same applies for all operators that follow. This means you can see very quickly which operators a change could have an effect on.

The process defined above only has a short runtime and so you will hardly have the opportunity to pause the running process. In principle, however, you can briefly stop a running process with the pause symbol, e.g., in order to see an intermediate result. The operator currently being executed is then finished and the process is then stopped. You can recognize a process that is still running but currently paused by the fact that the color of the play icon changes from blue to green.

Press the play button again to continue executing the process further. If you do not wish to merely pause the process but to abort it completely, then you can press the stop button. Just like when pausing, the operator currently being executed is finished and the process fully aborted immediately after. Please note that you can switch to the Design Perspective immediately after aborting the process and make changes to processes, even if the execution of the current operator is being finished in the background. You can even start further processes and do not need to wait for the first process to be completed.

Note: The operator being executed is always completed if you abort. This is necessary to ensure a sound execution of operators. However, completing an operator may need much more time in individual cases and also require other resources such as storage space. So if when aborting very complex operators you can see this taking hours and requiring additional resources, then your only option is to restart the application.

2.4.4 Looking at Results

After the process was terminated, RapidMiner should automatically have switched to the Result Perspective. If this was not the case, then you probably did not connect the output port of the last operator with one of the result ports of the process on the right-hand side. Check this and also check for other possible errors, taking the notes in the Problems View into consideration.

Feel free to spend a little time with the results. The process above should have delivered a dataset and a decision tree used to predict the label of the dataset based on the attributes' values. You can inspect the data itself as well as the meta data of this dataset and try out some of the visualizations in the plot view. You can also inspect the decision tree and try to understand if this makes sense to you. If you wish to return to the Design Perspective, then you can do this at any time using the switching icons at the right of the toolbar.

Tip: After some time you will want to switch frequently between the Design Perspective

and the Result Perspective. Instead of using the icon or the menu entries, you can also use keyboard commands F8 to switch to the Design Perspective and F9 to switch to the Result Perspective.

What does that result mean to us? We now have managed to load a dataset from the RapidMiner repository and then we have built the first predictive model based on this data. Furthermore, we got a first feeling about how to build RapidMiner processes. You are now ready to learn more about the use cases for data mining and how to build corresponding processes with RapidMiner. Each of the following chapters will describe a use case together with the data which should be analyzed. At the same time each chapter will introduce new RapidMiner operators to you which are necessary to successfully solve the tasks at hand.

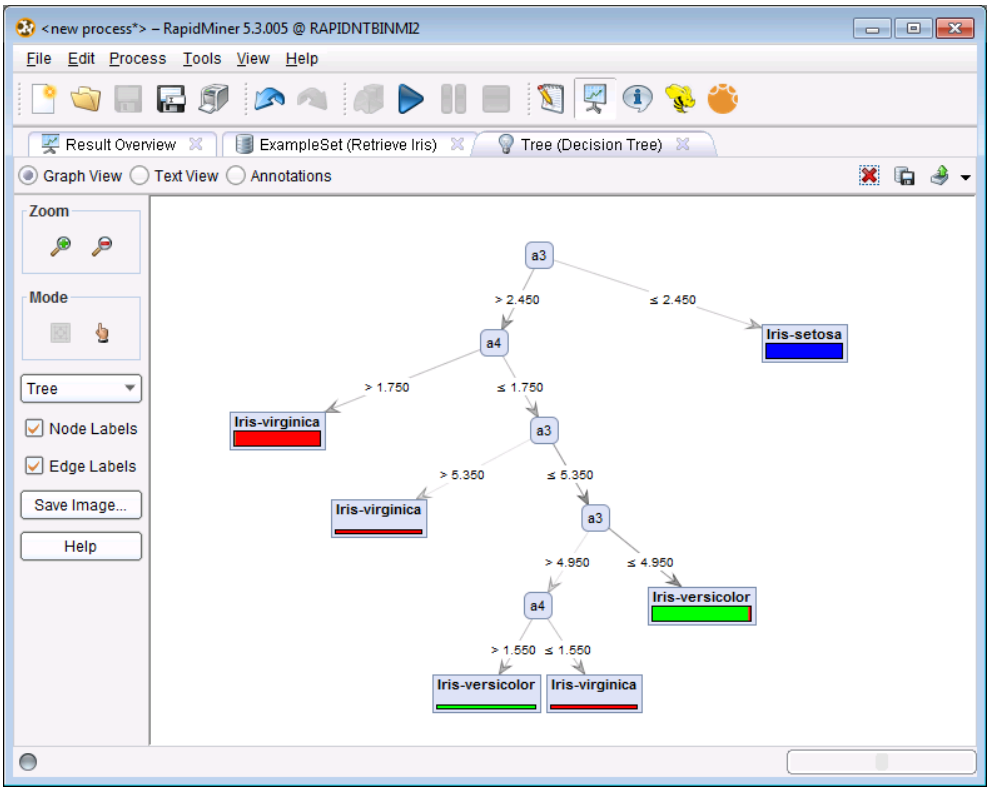

FIGURE 2.10: The decision tree described the rules how to assign the different classes to a new plant. First we check the value of a3 and if it small enough, we assign the blue class Iris-setosa to this plant. If not, we additionally check the value for a4 and if this is high we assign the red class Iris-virginica to it. Finally, we will check for attribute a3 again and decide for Iris-versicolor if the value is small enough.

Part II

Basic Classification Use Cases for Credit Approval and in Education

Chapter 6

Naïve Bayes Classificaton II

M. Fareed Akhtar

Fastonish, Australia

The use case of this chapter applies the Naïve Bayes operator on the Nursery dataset (overview of the Naïve Bayes algorithm has been discussed in the previous chapter). The working of the Naïve Bayes operator is also discussed in detail. The purpose of this use case is to rank applications for nursery schools using the Naïve Bayes algorithm. The operators explained in this chapter are: Read Excel and Select Attributes operators.

6.1 Dataset

6.1.1 Nursery Dataset

This dataset has been taken from the UCI repositories. This dataset can be accessed through this link: `http://archive.ics.uci.edu/ml/datasets/Nursery`.

6.1.2 Basic Information

Nursery Database was derived from a hierarchical decision model originally developed to rank applications for nursery schools. It was used during several years in the 1980s when there was excessive enrollment in these schools in Ljubljana, Slovenia, and the rejected applications frequently needed an objective explanation.

6.1.3 Examples

This dataset has 12,960 examples and there are no missing values in this dataset.

6.1.4 Attributes

This dataset has 9 attributes (including the label attribute). The dataset comes with some basic information, but the type and role of attributes is set by the user of the dataset. Even the attribute names are specified by the user. Here is an explanation of the attributes of this dataset:

1. **Parents:** This attribute has information about the parents of the child. It has three possible values: usual, pretentious, and great_pret. As this attribute can have three possible values, the type of this attribute should be set to polynominal in RapidMiner.
2. **Has_nur:** This attribute has information about the nursery of the child. It has five possible values: proper, less_proper, improper, critical, and very_crit. As this attribute can have five possible values, the type of this attribute should be set to polynominal in RapidMiner.
3. **Form:** This attribute has information about the form filled out by the applicants. It has four possible values: complete, completed, incomplete, and foster. As this attribute can have four possible values, the type of this attribute should be set to polynominal in RapidMiner.
4. **Children:** This attribute has information about the number of children of the applicant. It has four possible values: {1, 2, 3, more}. As this attribute can have four possible values, the type of this attribute should be set to polynominal in RapidMiner.
5. **Housing**: This attribute has information about the housing standard of the applicant. It has three possible values: convenient, less_conv, and critical. As this attribute can have three possible values, the type of this attribute should be set to polynominal in RapidMiner.
6. **Finance:** This attribute has information about the financial standing of the applicant. It has two possible values: convenient, inconv. As this attribute can have only two possible values, the type of this attribute should be set to binominal in RapidMiner.
7. **Social:** This attribute has information about the social structure of the family. It has three possible values: nonprob, slightly_prob, and problematic. As this attribute can have three possible values, the type of this attribute should be set to polynominal in RapidMiner.
8. **Health:** This attribute has information about the health picture of the family. It has three possible values: recommended, priority, and not_recom. As this attribute can have three possible values, the type of this attribute should be set to polynominal in RapidMiner.
9. **Rank** (label attribute): This attribute specifies the rank of the application. It has five possible values: not_recom, recommend, very_recom, priority, and spec_prior. As this attribute has five possible values its type should be set to polynominal in RapidMiner. The role of this attribute should be set to label because this is the target attribute or the attribute whose value will be predicted by the classification algorithms. The role of all other attributes should be set to regular.

More information about this dataset can be obtained from UCI repositories.

6.2 Operators in this Use Case

6.2.1 Read Excel Operator

For applying any algorithm on a dataset the first step is to import the dataset. Importing the dataset means loading the dataset into RapidMiner. RapidMiner provides numerous operators for loading datasets. These operators can be found at the 'Import/Data' section in the Operators window. Data is available in different formats. Therefore, RapidMiner provides different operators for importing data in these different formats.

Mostly data is stored in CSV files, Excel files, or in databases. To access such datasets, RapidMiner provides operators like Read CSV, Read Excel, and Read Database.

The Read Excel operator imports data from Microsoft Excel spreadsheets. The user has to define which of the spreadsheets in the workbook should be used as data table. The table must have a format so that each row is an example and each column represents an attribute. The data table can be placed anywhere on the sheet.

The easiest and shortest way to import an Excel file is to use *import configuration wizard* from the *Parameters* View. The import configuration wizard, Excel file, first row as names, and dataset meta data information parameters of the Read Excel operator are very similar to corresponding parameters in the Read CSV operator. Other important parameters of the Read Excel operator include the *sheet number* parameter which specifies the number of the sheet which contains the required dataset. The *imported cell range* parameter specifies the range of cells to be imported. It is specified in "xm:yn" format where "x" is the column of the first cell of range, "m" is the row of the first cell of range, "y" is the column of the last cell of range, "n" is the row of the last cell of range. For example, "A1:E10" will select all cells of the first five columns from row 1 to 10.

6.2.2 Select Attributes Operator

This is a very powerful operator for selecting required attributes of the given dataset because it gives many different options for selecting attributes. Attributes can be selected by simply mentioning their names. Attributes of a particular type or block type can also be selected. Attributes can be selected on the basis of missing values or even attribute value filters. The most powerful option for selecting attributes is through regular expression. Moreover, exceptions can be provided for any method of attribute selection.

6.3 Use Case

The main goal of this process is to have a deeper look at the working of the Naïve Bayes algorithm. The Read Excel operator is used for importing the Nursery dataset. The Split Data operator is used for splitting the data into training and testing datasets. Finally, the Performance (Classification) operator is used for evaluating the performance of the model.

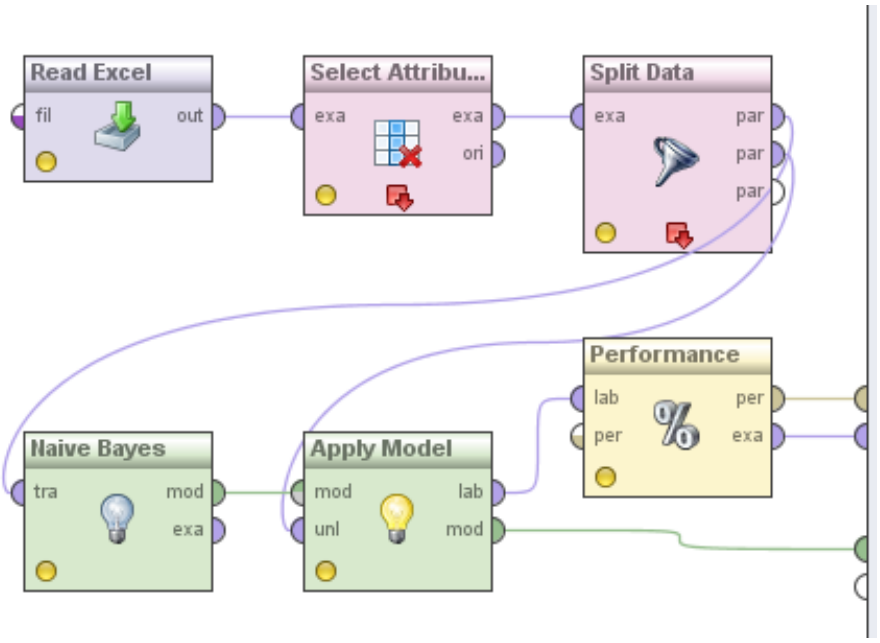

FIGURE 6.1: Workflow of the process.

6.3.1 Data Import

The Read Excel operator is used for importing data in this process. The data at the following url is stored in an Excel file. `http://archive.ics.uci.edu/ml/machine-learning-databases/nursery/nursery.data`.

This Excel file is then used as the data source in RapidMiner. The Read Excel operator is used for loading data from an Excel file. The *Import Configuration Wizard* is used for loading the data in this process. The steps of this configuration wizard are explained below.

Step 1: Select Excel file.

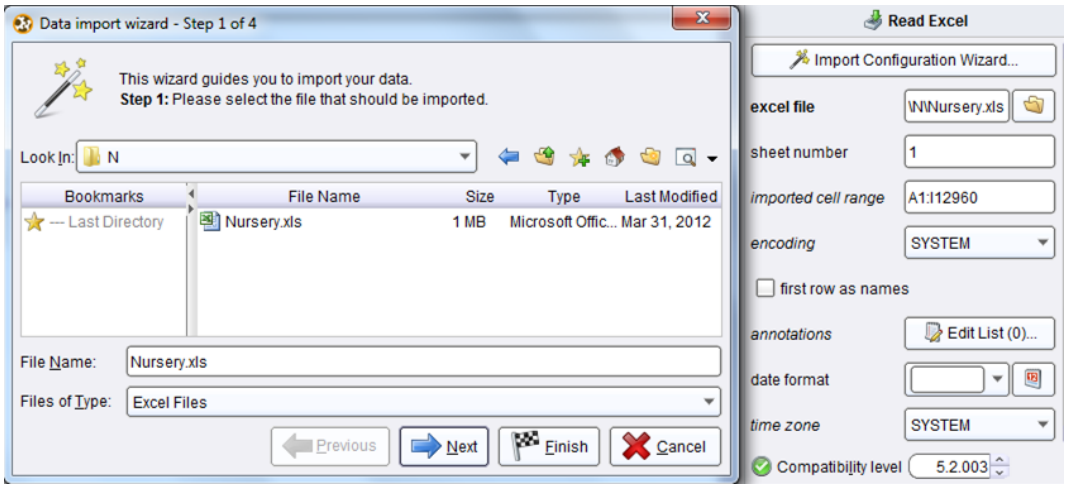

FIGURE 6.2: Step 1 of import Configuration Wizard of the Read Excel operator.

The first step is choosing the Excel file that contains the required data.

Step 2: Select the required sheet and cell range.

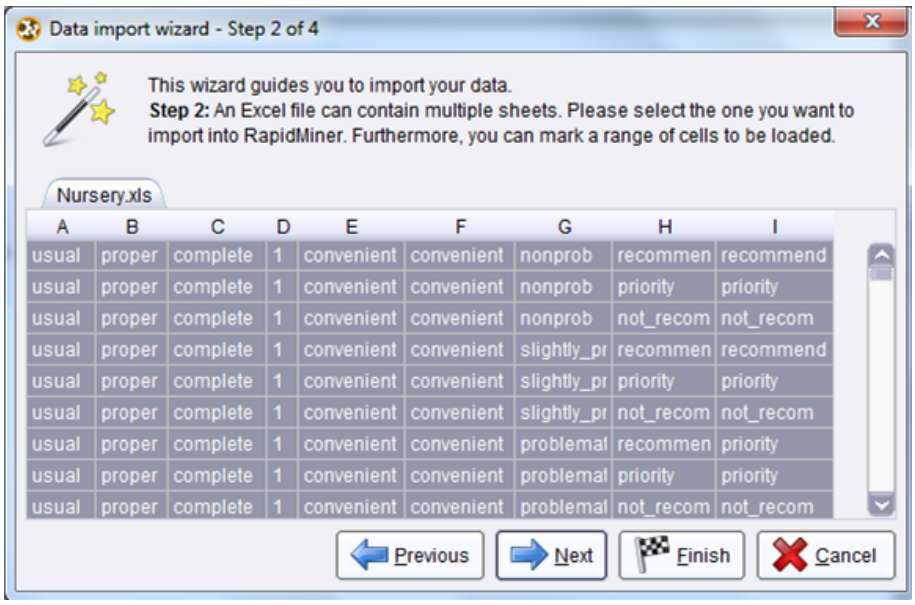

FIGURE 6.3: Step 2 of import Configuration Wizard of the Read Excel operator.

RapidMiner retrieves the Excel file and displays it in this step. The desired Excel sheet and cells are selected. By default, all cells are selected. In this process, there is no need to change anything at this step because there is only one sheet and all the required cells are already selected.

Step 3: Annotations.

This step enables the annotations to be specified. By default, the annotation of the first row is set to 'name'. If the first row of data does not contain the names of attributes, its annotation should be unchecked (as shown in Figure 6.4).

Step 4: Set the name, type, and role of attributes.

This is the most crucial step of the import wizard. The name, type, and role of attributes are specified in this step. Figure 6.5 shows the name, type, and role of attributes that are specified for the Nursery dataset. A breakpoint is inserted after this operator so that the output of the Read Excel operator can be seen in the Results Workspace.

6.3.2 Pre-processing

The Select attributes operator is used for selecting only the Form and Parents attributes. Only two attributes are selected to simplify this process (in order to make the explanation of working of Naïve Bayes operator simple).

6.3.3 Model Training, Testing, and Performance Evaluation

The Split Data operator is used in this process for splitting the dataset into two partitions. One partition is used for training the model and the other partition is used for testing the model. The *partitions* parameter of the Split Data operator performs almost the same task that the *split ratio* parameter does in the Split Validation operator. The *partitions*

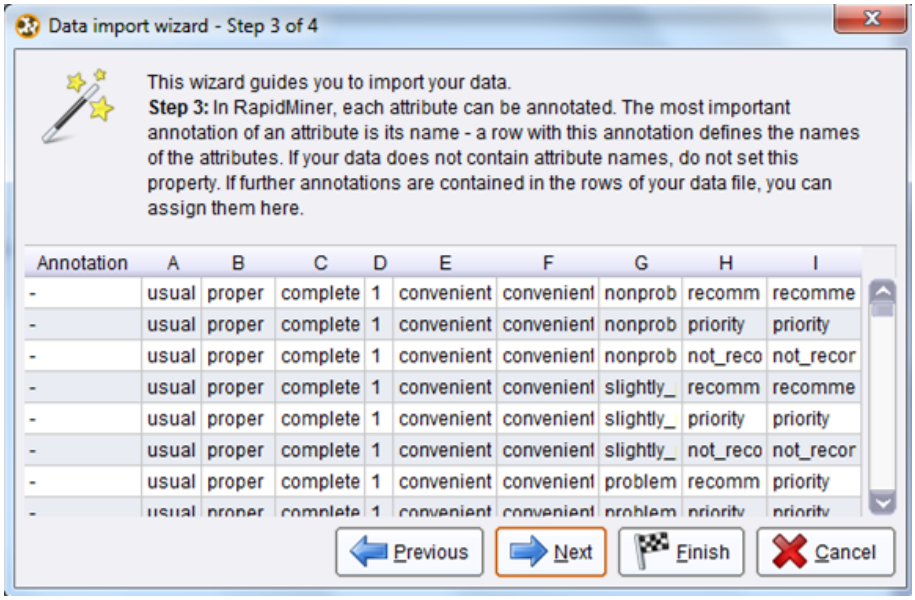

FIGURE 6.4: Step 3 of Import Configuration Wizard of the Read Excel operator.

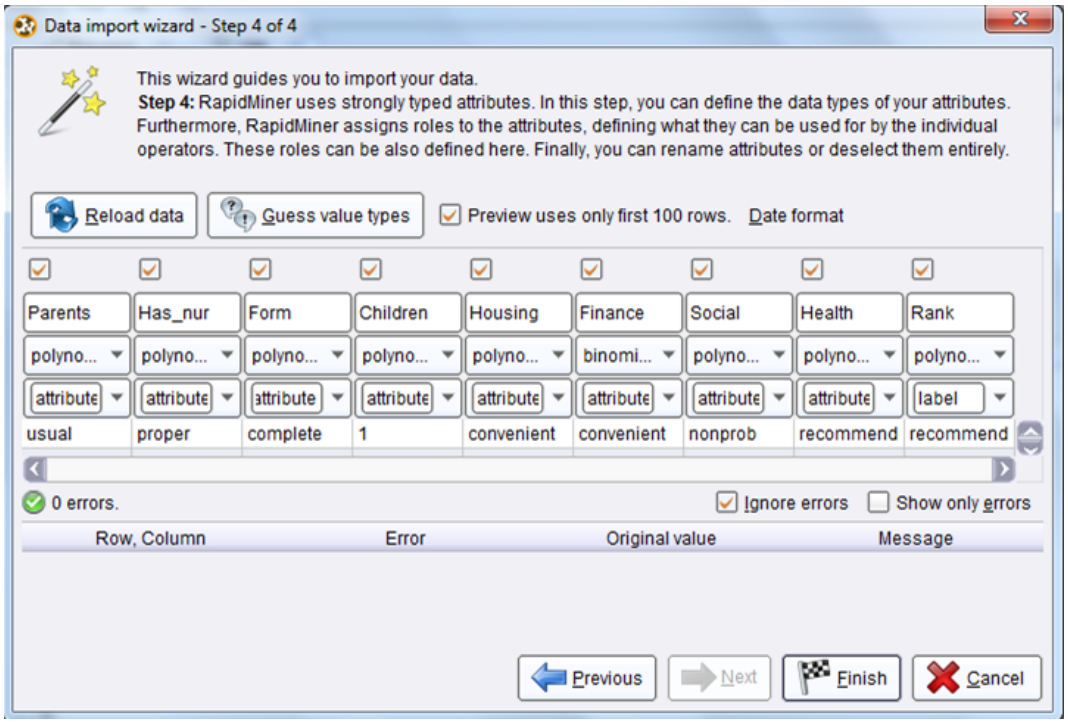

FIGURE 6.5: Step 4 of import configuration Wizard of the Read Excel operator.

parameter defines the number of partitions and ratio of each partition. The *sampling type* parameter of the Split Data operator behaves exactly the same as the *sampling type* parameter in the Split Validation operator. The parameters of the Split Data operator are shown in Figure 6.6.

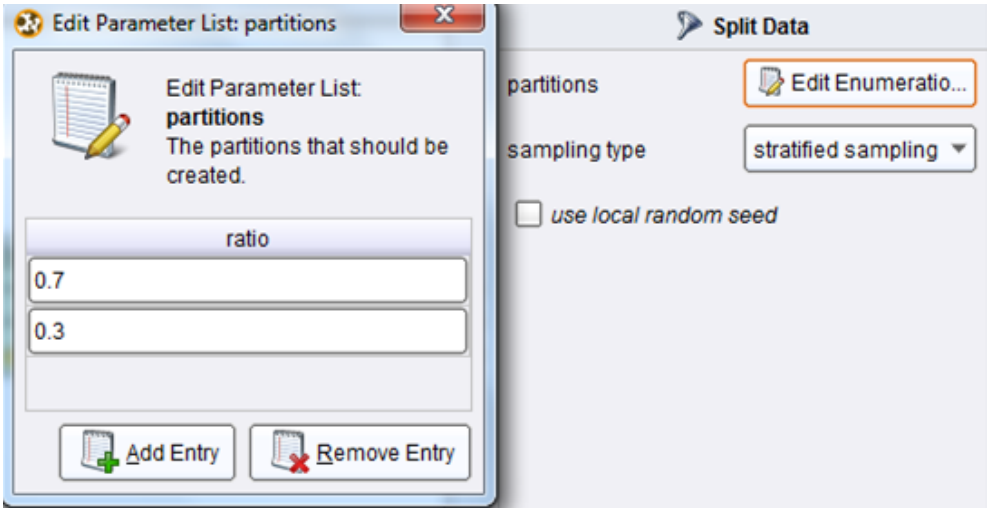

FIGURE 6.6: The parameters of the Split Data operator.

Training the Model The Split Data operator provides the first partition (ratio = 0.7) through its first output port. This partition is used for training a Naïve Bayes classification model through the Naïve Bayes operator. Initially, all parameters of the Naïve Bayes operator are used with default values. The Naïve Bayes operator provides the Naïve Bayes classification model as its output. This model is given as input to the Apply Model operator.

Testing the Model The Split Data operator provides the second partition (ratio = 0.3) through its second output port. This partition is used for testing the Naïve Bayes classification model through the Apply Model operator. The Apply Model operator applies the Naïve Bayes classification model (that was provided at its *model* input port) on the testing dataset (that was provided at its *unlabeled data* port). The resultant labeled dataset is delivered as output by the Apply Model operator. This labeled dataset is provided as input to the Performance (Classification) operator.

Performance Evaluation The Performance (Classification) operator should be used for performance evaluation of classification models because it provides many different performance measures in its performance vector. The performance vector generated by the Performance (Classification) operator and the labeled data set are connected to the *result* port of the process.

6.3.4 A Deeper Look into the Naïve Bayes Algorithm

The results of this process show three outputs:

1. Performance Vector
2. Labeled dataset (obtained by application of model on testing dataset)
3. Naïve Bayes classification model

Table 6.1 shows the posterior probabilities of the label values. Posterior probability is calculated by dividing the number of examples with that label value by the total number of examples.

TABLE 6.1: Posterior probabilities.

Rank (label)	No. of Examples	Posterior Probability
recommend	1	1/9072
priority	2986	2986/9072
not_recom	3024	3024/9072
very_recom	230	230/9072
spec_prior	2831	2831/9072

Figure 6.7 shows the first row of the labeled dataset. There is a confidence attribute for each possible value of the label. The label value with the highest confidence is assigned as predicted value for the example. In this example, the highest confidence (i.e., 0.450) is for label value = priority. Therefore, this example is predicted as priority.

The following steps explain how the confidences for each label value are calculated. They are calculated using the posterior probabilities and the distribution table (shown in Figure 6.8). The following calculations are done for finding the confidence of a label value:

1. Multiply the probabilities of all the attributes for that label value.
2. Multiply this product with the posterior probability of that label value.
3. Divide the resultant value by sum of all the confidences.

Here is an explanation of how the confidences for the first example were calculated. The rows where "Parents = usual" and "Form = complete" are highlighted in Figure 6.8 because these are the only rows that will be used for calculations for confidences in this example (because in this example, Parents and Form attributes have values "usual" and "complete" respectively). Firstly, confidences of all the label values are calculated without normalization. These values will not match the values in Figure 6.7. After calculating all the confidences, these values will be normalized. These normalized values will be the same as the values in Figure 6.7.

Rank	confidence(recommend)	confidence(priority)	confidence(not_recom)	confidence(very_recom)	confidence(spec_prior)	prediction(Rank)	Parents	Form
priority	0.001	0.450	0.339	0.062	0.148	priority	usual	complete

FIGURE 6.7: First row of labeled ExampleSet.

confidence (recommend) confidence (recommend) = P(Parents = usual |Rank = recommend) * P(Form = complete |Rank = recommend) * Posterior(recommend)

$$= 1.000 \quad * \quad 1.000 \quad * \quad 1/9072 = \quad 0.0001$$

confidence (priority) confidence (priority) = P(Parents = usual |Rank = priority) * P(Form = complete |Rank = priority) * Posterior(priority)

$$= 0.449 \quad * \quad 0.262 \quad * \quad 2986/9072 = \quad 0.0387$$

confidence (not_recom) confidence (not_recom) = P(Parents = usual |Rank = not_recom) * P(Form = complete |Rank = not_recom) * Posterior(not_recom)

$$= 0.344 \quad * \quad 0.254 \quad * \quad 3024/9072 = \quad 0.0291$$

Text View | Plot View | Distribution Table | Annotations

Attribute	Parameter	recommend	priority	not_recom	very_recom	spec_prior
Parents	value=usual	1.000	0.449	0.344	0.587	0.191
Parents	value=pretentious	0.000	0.348	0.329	0.413	0.314
Parents	value=great_pret	0.000	0.203	0.326	0.000	0.495
Parents	value=unknown	0.000	0.000	0.000	0.000	0.000
Form	value=complete	1.000	0.262	0.254	0.357	0.214
Form	value=completed	0.000	0.256	0.244	0.300	0.245
Form	value=incomplete	0.000	0.247	0.252	0.230	0.267
Form	value=foster	0.000	0.235	0.250	0.113	0.274
Form	value=unknown	0.000	0.000	0.000	0.000	0.000

FIGURE 6.8: Distribution table.

confidence (very_recom) confidence (very_recom) = P(Parents = usual |Rank = very_recom) * P(Form = complete |Rank = very_recom) * Posterior(very_recom)

$$= 0.587 \quad * \quad 0.357 \quad * \quad 230/9072 = \quad 0.0053$$

confidence (spec_prior) confidence (spec_prior) = P(Parents = usual |Rank = spec_prior) * P(Form = complete |Rank = spec_prior) * Posterior(spec_prior)

$$= 0.191 \quad * \quad 0.214 \quad * \quad 2831/9072 = \quad 0.0127$$

Normalization All the confidences are divided by the sum of all the confidence values to get normalized confidence values. The sum of confidence values is 0.0859. The normalized confidence values are shown in the table below.

TABLE 6.2: Normalized confidences.

Rank (label)	**Confidence**
Recommend	0.0001/0.0859 = 0.001
Priority	0.0387/0.0859 = 0.450
not_recom	0.0291/0.0859 = 0.339
very_recom	0.0053/0.0859 = 0.062
spec_prior	0.0127/0.0859 = 0.148

As the confidence of the "priority" label value is the highest, the rank of this example is predicted as "priority".

Chapter 14

Robust Language Identification with RapidMiner: A Text Mining Use Case

Matko Bošnjak

University of Porto, Porto, Portugal; Rudjer Boskovic Institute, Zagreb, Croatia

Eduarda Mendes Rodrigues

University of Porto, Porto, Portugal

Luis Sarmento

Sapo.pt - Portugal Telecom, Lisbon, Portugal

Acronyms

API - Application Programming Interface

ETL - Extract, Transform and Load

HTTP - HyperText Transfer Protocol

k-NN - k Nearest Neighbours

NLP - Natural Language Processing

SVM - Support Vector Machines

TF-IDF - Term Frequency - Inverse Document Frequency

UTF-8 - Unicode Transformation Format – 8-bit

XML - eXtensible Markup Language

14.1 Introduction

Language identification, the process of determining the language of machine-readable text, is an important pre-processing step in many information retrieval and web mining tasks. For example, the application of natural language processing (NLP) methods may require prior language identification, if the language of the text at hand is unknown. In order to properly execute stemming, sentence tokenization or named entity recognition, we need to identify the language of the text to successfully apply appropriate language technologies. In addition, language identification is essential in machine translation tasks. Some text classification tasks, such as sentiment analysis in social media, may also require language identification for filtering content written in a specific language.

There are numerous proprietary and open-source solutions for language identification, including stand-alone solutions and APIs. However, proprietary solutions are usually costly, and APIs may cease to exist, while open-source solutions may require deeper understanding of the code for further modification and implementation. Moreover, most solutions are usually fixed on a pre-defined set of languages.

This chapter aims to provide a walk-through of language identification fundamentals by first introducing the theoretical background, followed by a step-by-step guide on model building, according to the standard practice of data mining. We will put emphasis on understanding the problem, and solving it in an effective and costless manner within the open source environment RapidMiner. We will become familiar with RapidMiner's "Text Mining Extension", and learn how to create several simple and fast workflows for language identification. The methods we will use are generic, and applicable to any dataset, with the appropriate pre-processing. In addition, we will learn how to use the implemented models in custom projects, by exporting them as web services using RapidAnalytics, and deploying them in custom applications. We will go through a case study of a web application for language identification of web pages, using RapidMiner's "Web Mining Extension". Special emphasis will be given to the optimization of the exported workflow, to enable faster execution, and harmonious integration in other applications.

14.2 The Problem of Language Identification

The task of language identification can be simply described as discerning the language of a given text segment. For example, given a set of six sentences in different languages:

How many languages do you speak well?

Wie viele Sprachen können Sie gut sprechen?

Combien de langues parles-tu bien?

Πόσες γλώσσες μιλάς καλά;

¿Cuántas lenguas hablas bien?

Koliko jezika govorite dobro?

Quantas línguas você fala bem?[1]

the goal is to identify the language of each sentence. This task may be somewhat trivial for humans. We might be able to easily discern the language and possibly identify some of them, even though we might not speak any of those languages. The first sentence is easy, since we are reading this book written in English. We may say that the second sentence "feels" German, or that the third one "sounds" French. The alphabet of the fourth sentence looks completely different from the others, so we might guess it "looks like" Greek, while the fifth sentence might "lead to" Spanish. We might find the sixth sentence somewhat tricky since it looks different from the previous ones. If we are vaguely familiar with the Slavic family of languages we might say it "relates to" that family, since the sentence is written in Croatian. Finally, although we might find the last sentence "similar to" Spanish, it is, in fact, written in Portuguese.

When identifying languages, we use our knowledge of languages, acquired either actively or passively. To design algorithms for achieving the same task, we first need to systematize the knowledge needed for language identification. There are several indicators we can rely on when identifying languages, without knowing those languages at all. These indicators are the following:

Alphabet Differences in symbols a language is written in are the most obvious feature for language identification. Even a person unfamiliar with languages at hand can discern different alphabets, and can easily be taught to identify different scripts due to their specificity. This, of course, does not hold for languages written in the same alphabet. Examples of several different alphabets are given in Figure 14.1.

Typical words Words can be a strong language indicator, whether those words are unique for the language, or the most frequent ones. Unique words, such as *fika*[2] in Swedish, or *saudade*[3] in Portuguese, are an excellent indicator of their corresponding languages. Nevertheless, they may occur infrequently in text. The most frequent words, like *niet*

[1] Source: Tatoeba Project: Open, collaborative, multilingual dictionary of sentences: `http://tatoeba.org`, Sentence $n°682402$.

[2] "Coffee-break"

[3] "A deep emotional state of nostalgic longing for an absent something or someone that one loves", source: Wikipedia

Latin	data mining
Greek	εξόρυξη δεδομένων
Cyrillic	добыча данных
Hebrew	כריית נתונים
Arabic	استخراج البيانات
Japanese	データマイニング
Chinese	数据挖掘

FIGURE 14.1: Examples of different alphabets.

in Dutch, *ne* in Slovene, or *ikke* in Norwegian[4] are the next obvious choice, and we will discuss them further in the chapter.

Accented letters Languages in Latin alphabets often contain various accented letters, which can be present in only several languages, or can even be unique to a language. These letters can be used for narrowing down the choice of languages in question. The downside of this approach is that accented letters may not at all be present in the text, depending on the encoding used or the style of communication. For example, in informal communication, people frequently omit accents. Examples of accented letters are *Ě* in Czech, *Ö* in German, Swedish, Icelandic, Turkish, etc., and *Š* in Slavic and Baltic languages.

Special symbols If a language uses special symbols, rarely present or not present at all in other languages, they can be used as a strong language indicator. A weakness of this indicator is that, similar to accented letters, these symbols can be used rarely, or not used at all. Examples of such symbols are the inverted question mark *¿*, and exclamation mark *¡*, used only in Spanish, the "Scharfes S", *ß* used in German, *ħ* used in Maltese, or semicolon *;* used as a question mark in Greek, among others.

Letter combinations Combinations of n consecutive letters found in substrings of words are called character n-grams, and are powerful language indicators. By finding these combinations we find digraphs, word beginnings and endings, prefixes, suffixes, or any other frequent letter combinations typical for a language. Examples of these combinations are the digraphs *Lj*, *Nj*, *Dž* in Croatian; *Zs*, *Gy*, *Ny* in Hungarian; typical word beginnings like *be-*, *re-*, *sh-* in English, and *des-*, *pr-*, *da-* in Portuguese; and typical word endings like *-ns*, *-ur*, *-ont* in French, and *-ung*, *-eit* in German.

Word length Some languages tend to form long words by concatenating distinct words, suffixes, and prefixes into single words with a compound meaning. Such words can frequently be found in Finnish, German, and Hungarian, among others. An example of such a composite word is *Geschwindigkeitsbegrenzung*[5] in German.

N-gram distribution Distribution of character n-grams per language is one of the most powerful indicators used for language identification. Different languages exhibit different character n-gram distributions, starting with single letters, also called 1-grams or

[4] All of them mean "no".
[5] "Speed limit".

unigrams. An example of the distribution of unigrams is given in Figure 14.2. In this example we observe a much bigger usage of letter a in Finnish than in the other two languages, an almost non-existent usage of letter j in Italian, and far greater usage of letter z in Czech than in both Finnish and Italian.

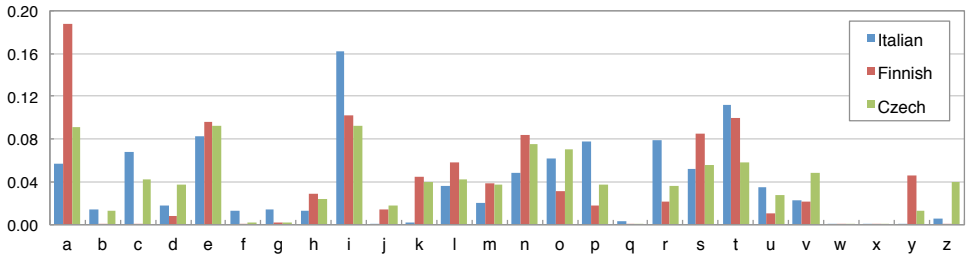

FIGURE 14.2: Distributions of unigrams in Italian, Czech, and Finnish show clear differences in the use of distinct letters per language. In this example, accented letters are stripped of their accents, and non-alphabet symbols are ignored.

In the remainder of this chapter, we show how to implement several language identification methods using the most popular techniques and features, both in literature and in practice. Namely, we implement the frequent words, the character n-gram, and the similarity-based techniques. Although some researchers [1] argue that the problem of language identification has been resolved, issues are still ongoing [2, 3], from the language identification of web pages [4, 5], short queries [6, 7], and micro-blog messages [8], to differentiation of similar languages [8, 9]. However, the extent of these issues go beyond the scope and the intention of this chapter. We continue by presenting the fundamentals of text representation and classification models.

14.3 Text Representation

In order to start the text analysis process, we first need to represent the text as a vector of numbers in a consistent way. Prior to explaining the representation format, we need a word of notice regarding digital alphabet representations — encodings.

14.3.1 Encoding

Simply put, character encoding is a convention that maps a number to a symbol, which enables numerical storage of characters on a digital media. Due to historical reasons, there are many different ways to represent different characters. There are two main ways of encoding characters: single- and multi-byte character encodings.

Single-byte character encodings represent symbols in a single-byte of information. The most known such encoding is ASCII, which uses only 7 bits to encode up to 128 different symbols. The inclusion of the 8^{th} bit already brings many troubles since there are many different character encoding tables which will map numbers higher than 127 to different symbols. The most widespread single-byte character encodings are the ISO 8859, and Microsoft Windows sets of encodings. Due to single-byte encodings' limited capacity of only 256 symbols, multi-byte encodings are more frequent in use today. Multi-byte character en-

codings utilize several bytes of data to represent a symbol, and are therefore able to encode much larger spectra of symbols. The most widespread multi-byte encoding is UTF-8.[6]

The choice of encoding is very important since a badly chosen encoding might make the text illegible. For example, the Greek word for data, $\delta\varepsilon\delta o\mu\acute{\varepsilon}\nu\alpha$ encoded in UTF-8, if mistakenly shown with the ISO 8859-1 encoding, will appear as $\hat{I}''\hat{I}\mu\hat{I}\hat{I}?`\hat{I}^1/_4\hat{I}\hat{I}^1/_2\hat{I}\pm$.

Throughout the rest of this chapter, we use the UTF-8 encoding, due to its widespread usage and the availability of major language scripts in it. While some language identification methods operate on bytes of data rather than characters, they cannot cope with code overlap in different encodings, and need to employ encoding detection. More information about encoding detection for language identification can be found in [10].

In addition, we limit our language identification to languages written in the Latin alphabet. Other alphabets can be detected by checking the availability of letters in certain code ranges, under the assumption of single encoding like the UTF-8. For example, we can detect the Greek alphabet by executing the regular expression `[\u0370-\u03FF]*`. This regular expression will include all the symbols ranging from the code point 0370 to the code point 03FF, expressed in hexadecimal, which in UTF-8 represents all the new and old Greek alphabet symbols. In RapidMiner, this regular expression can be applied to text with the *Keep Document Parts* operator. Note that this regular expression will not include Latin letters nor numerals. To include them and the standard punctuation, `[\u0370-\u03FF\u0020-\u007E]*` should be used.

With the encoding explained, and set to UTF-8, we continue on the level of text representation for data mining tasks.

14.3.2 Token-based Representation

The token-based representation is built on top of basic meaningful elements of text, called tokens. Tokens can be words separated by delimiters; logograms—signs and characters representing words or phrases, such as Chinese letters; idioms, or fixed expressions, such as named entities—names of places, people, companies, organisms, etc.

The process of token extraction is called tokenization. For most of the languages and uses, tokenization is done by splitting sentences over whitespaces and punctuation characters, sometimes ignoring numbers, or allowing specific punctuation in a word, e.g., *wasn't*, *off-line*. For example, the following sentence:

```
El mundo de hoy no tiene sentido, así que ¿por qué debería pintar
cuadros que lo tuvieran?[7]
```

separated by non-alphabet characters, results in the following set of tokens:

```
El mundo de hoy no tiene sentido así que por qué debería pintar
cuadros que lo tuvieran.
```

In this chapter we use this simple tokenization over non-letter characters. However, in languages in which words are not delimited by space, such as Chinese, Japanese, Thai, Khmer, etc., tokenization is a more complicated process, and it requires complex machine learning models outside of the scope of this chapter.

Note that when tokenizing over non-letter characters, apostrophes are used as points of separation, which results in *wasn't* tokenized as *wasn* and *t*. We can argue that this is not

[6] Usage Statistics of Character Encodings for Websites `http://w3techs.com/technologies/overview/character_encoding/all/`.

[7] "Today's world doesn't make any sense, so why should I paint pictures that do?" ---Pablo Picasso

a desired behaviour in our process, since *wasn't* should be one token. However, due to the simplicity of setting parameters in RapidMiner, we opted for this solution. Easy tweaking of RapidMiner options allows inclusion of any other symbol, including the apostrophe. Later in the chapter, we explain how to set these parameters.

When dealing with specialized forms of text, like blog or micro-blog messages, for specific purposes like sentiment analysis, tokenization becomes a more complicated process in which non-letter tokens, such as smileys and interjections,[8] are retained. For more information on tokenization, consult further literature [11, 12, 13].

Other than splitting the text in tokens, and continuing to build a data structure for text mining, we can also split the text on the character basis. This results in a character-based representation.

14.3.3 Character-Based Representation

Character-based representation is a text representation based on top-of-character n-grams, previously defined as word substrings consisting of n consecutive characters. In general NLP usage, n-grams can also denote an ordered set of words, however is this chapter when mentioning the term n-gram, we specifically refer to character n-grams. For example, the following sentence:

```
Ce n'est pas assez d'avoir l'esprit bon, mais le principal
est de l'appliquer bien.[9]
```

cut into 2-grams, also called bigrams, after tokenization results in the following set:

```
_a(3) _b(2) _c  _d(2) _e(3) _l(3) _m  _n  _p(2) ai al  ap as(2) av
bi  bo  ce  ci  d_  de e_(3) en  er  es(3) ez ie  in  ip  iq
ir is  it  l_(3) le  li  ma  n_(3) nc  oi on pa(2) pl  pp  pr(2)
 qu  r_(2) ri(2) s_(2) se  sp ss st(2) t_(3) ue  vo  z_[10]
```

Extraction of n-grams can be executed after tokenized, or even non-tokenized text. The difference is that when extracting n-grams from non-tokenized texts, depending on the number n, n-grams can catch both the ending of one, and the beginning of another word. For example, when extracting 3-grams from the term *data mining*, 3-gram "*a m*" will also be extracted. This does not happen if we execute tokenization prior to n-gram extraction.

N-grams are normally used to catch word beginnings and endings, as well as typical combinations of consecutive letters. Whereas the number of words is practically unlimited, the number of n-grams is limited to the power of number of letters n. In real scenarios, the number of observed n-grams is actually much lower than the maximum possible number of n-grams, since not all letter combinations are present in natural languages. This enables compressed format of the text and creates helpful features in language identification.

Whether we choose to represent the text with tokens or n-grams, prior to continuing the text mining process, we convert them to a bag-of-words vector.

14.3.4 Bag-of-Words Representation

The bag-of-words is a representation used in NLP and text mining, in which a text, such as a sentence or a full document, is represented by a set of words—an unordered collection of

[8] Words used to express emotions and sentiments.

[9] "It's not enough to have a good mind; the most important thing is to use it well." ---René Descartes.

[10] Apostrophes replaced by empty space, added underscores to empty spaces to emphasize word beginnings and ends shorter than n, bigram cardinality shown in parenthesis, where greater than one.

words.[11] Note that although this representation completely disregards grammar and word order, it suffices for our needs. For example, the following sentences:

```
The following sentence is true.
The preceding sentence is false.
```

consists of the following words, without repetition:

```
1. the, 2. following, 3. sentence, 4. is, 5. true, 6. preceding, 7. false
```

By using this set of words as an index or dictionary, we create a fixed representation of a sentence, consisting of an occurrence of a specific word, at its appropriate index. This results in creating a word vector. For example, the previous two sentences, using the extracted dictionary, result in these word vectors:

```
(1 1 1 1 1 0 0)
(1 0 1 1 0 1 1)[12]
```

The second vector in our example denotes a sentence containing one element of each of indexes $1, 3, 4, 6, 7$ in the dictionary, corresponding to words *the*, *sentence*, *is*, *preceding*, and *false*. With this approach we lose grammatical structures and word order, but we simplify the resulting model, making it tractable from the analytic point of view, and as we will demonstrate, sufficient for our case studies. Upon obtaining word vectors of the bag-of-words representation, we are ready to use classifiers for language identification.

14.4 Classification Models

Our main goal, developing a system for language identification, requires an applicable classification model. An example of a model might be a simple string matching—if a language contains one or more words, or n-grams of a certain language, it belongs to that language. This type of model leads to rule-based systems. Another approach is to represent the text segment in an n-dimensional space, where each dimension corresponds to a single word metric like the occurrence, TF-IDF score, etc. This leads to vector-space models, that are frequently used with Support Vector Machines (SVMs). In practice, statistical language models, like Naïve Bayes or Markov models, are frequently used since they yield high performance, and in the case of Naïve Bayes, low computational cost. Though the Naïve Bayes classifier is often outperformed by other classifiers, like SVM, in terms of prediction performance, we use it since it is fast, and does not need parameter optimization. Feel free to experiment with other classifiers to find the best performing one.

The first classification model we employ is the Naïve Bayes classifier. The Naïve Bayes classifier is a generative probabilistic model that assigns a probability to a segment of a text, composed of either single or multiple tokens or parts of tokens [14]. This is done through statistical analysis of the text, by translating frequencies of word occurrences in a document into probabilities, and employing a statistical framework for classification [15]. This is one of the models we use in our case studies.

The second model we employ is the similarity-based model. The similarity-based model

[11] The precise term would be multiset, a generalization of set, which allows repeating members.
[12] The example is considered case-insensitive; words are ordered in the sentence order.

relies on creating a distribution of elements, such as tokens or n-grams per language, and using a similarity measure to determine the language with the most similar distribution to the queried example. While the classification approach requires a large number of examples per language, the similarity approach relies on a single large example per language, for example, all the texts of a single language concatenated together. This model, in its essence, is the k Nearest Neighbours (k-NN) model. Having a large text per language enables us to extract a distribution of tokens, or n-grams, that closely approximates the true distribution of the language.

One of the most widely used approaches is the ranking approach proposed by Cavnar and Trenkle [16] in which they implement the ranking similarity between the query text profile and language profiles. Since the ranking similarity is not yet implemented in RapidMiner, we can use other similarity measures frequently utilized, such as the Euclidean distance, dot product, and cosine similarity [10, 17, 18].

Having token and character-based text representation as the ground representation, out-of-bag word vectors, and the Naïve Bayes and k-NN classifiers at hand, we are in the position to start discussing the implementation of possible language identification workflows in RapidMiner.

14.5 Implementation in RapidMiner

In this section, we move on to building RapidMiner workflows by presenting three different models for language identification, built on top of the previously introduced theoretical background. The models we present are the frequent words, character n-gram, and the similarity-based models [19]. Of course, additional operators in RapidMiner can be used to build more complex models, and improve the ones we will present; however, we do not cover all the advanced details in this chapter.

The first step in the implementation process is to obtain a suitable dataset for experimenting with language identification techniques. If you already have a dataset available, and you wish to skip on to model construction, you can proceed to Section 14.5.3. However, if you do not have a dataset prepared, and cannot easily get a hold of one, there are several freely available datasets you can use. In the next subsection we introduce a few such datasets, which you can use to re-create and experiment with RapidMiner workflows.

14.5.1 Datasets

When creating a dataset for developing language identification models, one has to consider several important factors:

Time Language is dynamic, and changes over time. Commonly used words and phrases differ between centuries, decades, and even years. Obtaining datasets from comparable eras ensures similar language properties and stability of used words.

Source Different sources of text exhibit different properties. For example, news articles, web pages, dictionaries, social media like Twitter, and classical writing greatly differ in the vocabulary, punctuation, and even word case. Thus, the choice of dataset should favor texts from languages coming from the same or comparable sources. This ensures that the language has a fairly stable style.

Size Datasets may differ in size. Small datasets might not exhibit a very representative

distribution of words or n-grams, while a large dataset might. However, computations with large datasets are more costly, in terms of time, complexity, and memory requirements. Obtaining comparable sizes of datasets per language is also important to represent all the languages with approximately the same number of sentences, words, or tokens.

Length Length of texts used for training or for querying has an impact on performance. Building classifiers on small sentences for training might decrease performance if the number of sentences is also small. Likewise, identifying language of a short sentence is a difficult task since only a small number of features are observed.

Therefore, when creating a dataset, we should collect texts of comparable sizes, coming from comparable sources, created in near time, and of adequate text length compared to query texts, in order to ensure higher performance of the language identifier.

There are multiple different datasets which can be used for this purpose, that abide by most of the previous recommendations.

Wikipedia Wikipedia[13] is a free Internet encyclopaedia with more than 21 million articles written in 284 languages. The articles can be downloaded and used for language identification. Wikipedia database dumps can be found at `http://dumps.wikimedia.org/.` in various formats and levels of information.[14] For language identification tasks, we recommend abstracts of web pages in XML format.

Leipzig Corpora Collection The Leipzig Corpora Collection[15] is a collection of textual corpora in 155 different languages using the same format and comparable sources [20]. The corpora consist of randomly selected sentences in the corpus language, and are available in sizes varying from tens of thousands up to tens of millions of sentences. The sources of sentences are news articles and texts, randomly collected from the web.

Project Gutenberg Project Gutenberg[16] is the oldest digital library of written cultural work. This web page offers full text of public domain books in various formats, out of which UTF-8 plain-text is the most straightforward for our use. In May 2012, Project Gutenberg claimed to have in excess of 39,000 free ebooks in more than 50 languages, with English being the most predominant language. Note that these books are older belletristic works.

European legislature datasets There are several datasets originating from the official documents of the European Union (EU), published in the official languages of the EU. Download

EuroGOV[17] is the document collection of web documents crawled from European governmental sites, with restricted access.

European Parliament Proceedings Parallel Corpus[18] [21] is a parallel corpus extracted from the proceedings of the European Parliament 1996–2010. It consists of 21 languages and 20 language pairs, English versus the rest. Though conceived as a standard dataset for statistical machine translation systems, it can also be used for language identification.

[13] `http://www.wikipedia.org/.`

[14] Details on database download are available at `http://en.wikipedia.org/wiki/Wikipedia:Database_download.`

[15] `http://corpora.informatik.uni-leipzig.de/download.html.`

[16] `http://www.gutenberg.org/.`

[17] `http://ilps.science.uva.nl/WebCLEF/EuroGOV/.`

[18] `http://www.statmt.org/europarl/.`

The JRC-Acquis Multilingual Parallel Corpus[19] is the corpus representing the total body of European Union law applicable in the EU member states [22]. These texts are available in 22 languages.

In this chapter, we use the Leipzig Corpora Collection dataset of 30.000 sentences in English, German, French, Spanish, and Portuguese. The dataset is easy to download, we just need go to the Leipzig Corpora Collection web page at `http://corpora.informatik.uni-leipzig.de/download.html`, scroll down to "Download Corpora" type in the captcha text, and click on "check". When a table with various corpus sizes and formats appears, we select the corpora we find suitable and download them by clicking on them. The next step after downloading the data is to prepare it and import into RapidMiner.

14.5.2 Importing Data

Upon downloading the corpora for English, German, French, Spanish, and Portuguese, we extract the compressed file contents and isolate files with a suffix `-sentences.txt`. These sentence files are tab delimited files containing a line number and a sentence in each line of the file. After downloading a dataset per language, and extracting its content, we rename the downloaded text file according to the language of the data, in our case *eng*, *deu*, *fra*, *spa* and *por* respectively, and save the files in a single directory. In our repository, these files are located in the `data/corpus` directory.

In order to import the Leipzig Corpora Collection, we create a data importing workflow, presented in the upper part of Figure 14.3. We loop over the files we previously saved using the *Loop Files* operator (1), by setting the *directory* property to the path to data, in our case `data/corpus`. The *Loop Files* operator will iterate over all files in the directory we set it to, and execute its nested process for each iterated file.

The content of the *Loop Files* operator is shown in the lower part of Figure 14.3. Essentially, in every iteration of the *Loop Files* operator, we read the text file, and fit it to specifications for further processing, creating the appropriate label, and finally splitting the data into two datasets: training and test datasets. The training dataset is further split into two forms: i) example per sentence and ii) all the text concatenated per language.

In more detail, the *Read CSV* operator (1) receives the file to open from the nested process of the *Loop Files* operator. We set the *column separator* property to `\t` since the text is tab delimited, uncheck the *use quotes* property since sentences are not delimited by quotes, uncheck the *first row as names* property since there are no row names in the first row of the file, and set the *encoding* property to UTF-8. The *Rename* operator (2) renames the default `att2` attribute to `text`, since the *Read CSV* operator returns default values for row names, when the first row of the file does not convey names of attributes. With the *Select Attributes* operator (3), we select only the `text` attribute, thus ignoring the line number. The *Generate Attributes* operator (4) generates the `language` attribute, and sets its value to the `replace("%{file_name}",".txt","")` expression. This expression essentially uses the name of the file to populate the `language` attribute. This is why we renamed the file names to short names of languages, to automate the whole process. The `replace("%{file_name}",".txt","")` expression relies on the macro *%{file_name}* constructed by the *Loop Files* operator per each iteration, and contains the name of the file in the current iteration. The next step is to set the label role to the newly created `'language'` attribute with the *Set Role* operator (5). The output of the *Set Role* operator is now a dataset containing two attributes: the *text* attribute containing each sentence text, and the *language* attribute containing the label of each of those sentences, fixed to the name of

[19] `http://langtech.jrc.it/JRC-Acquis.html`.

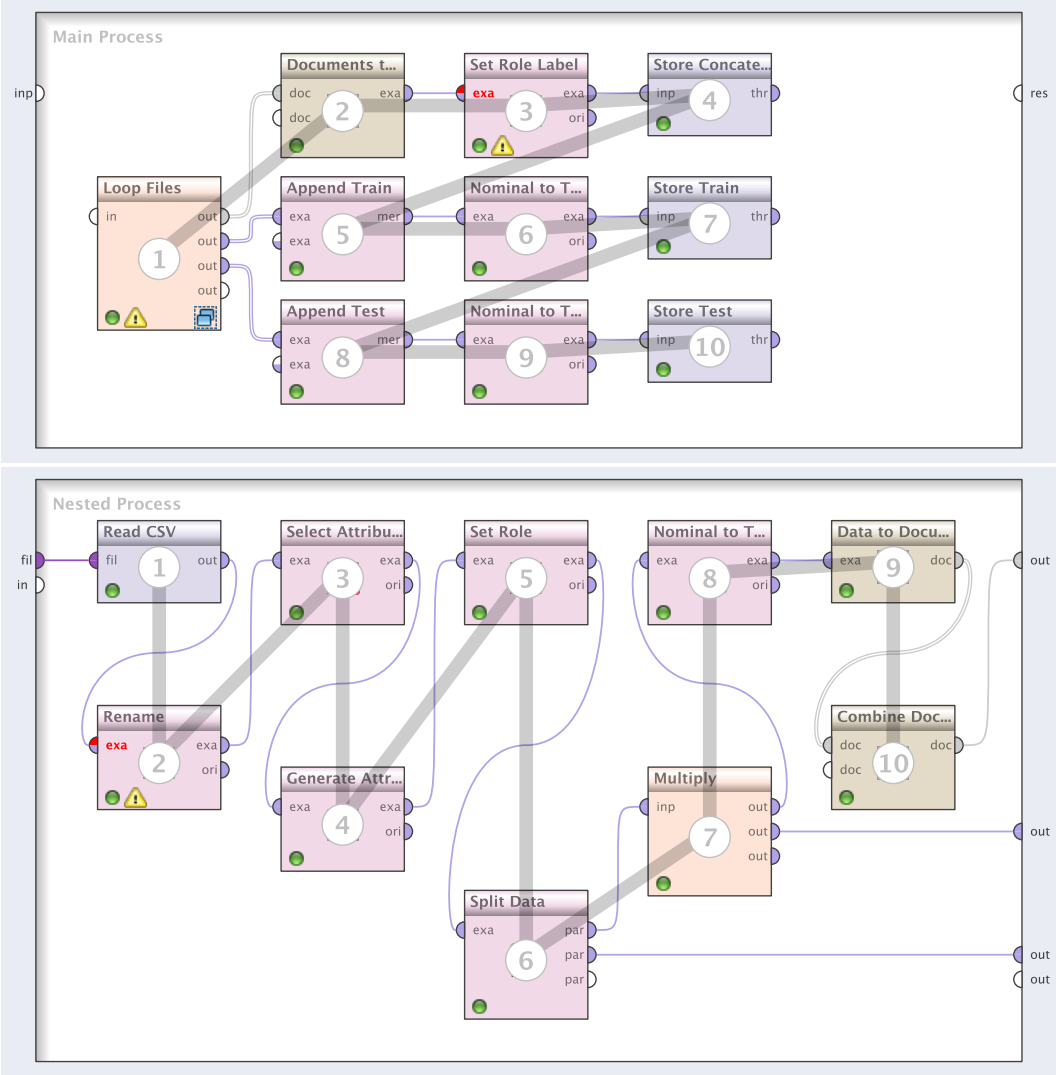

FIGURE 14.3: The workflow for loading the Leipzig Corpora dataset. The upper image depicts the main workflow process, whereas the lower image depicts the content of the *Loop Files* operator (1) in the upper image.

the file which is a short code for the language. With the *Split Data* operator (6) we split the dataset randomly with the ratios of 0.3 and 0.7 into the train and the test datasets, respectively. This kind of split is unusual in real-world setting—we would usually create a bigger training set than the test set. However in this case, we opted for a smaller training set to increase the speed of training. We output the test dataset without any change, while we process the train set into two modes. The first mode is simply the standard sentence-per-example dataset, outputted for processing and saving, just as the test set. The second mode is the concatenated dataset, created by first converting the `text` attribute from nominal to text type, using the *Nominal to Text* operator (8), since the nominal type cannot be processed as a textual value. In the end, the *Data to Documents* (9) and the *Combine Documents* (10) operators convert the dataset first to a collection of documents, and then combine that collection in a single resulting document.

Back to the main process. The datasets are outputted from the *Loop Files* operator (1) in the following order: the collection of documents containing one document of concatenated train data, the collection of train data, and the collection of test data, all three per iterated file. The collection of documents containing documents of concatenated data per language is converted to data with the *Documents to Data* operator (2), the role of the *language* attribute is set to label via the *Set Role* operator (3). The resulting dataset is the concatenated dataset, a dataset containing two columns: a label column denoting the language, and the text column, containing concatenated texts of all the sentences corresponding to each language. This dataset is stored to the repository as the `language_concatenated` dataset, using the *Store* operator (4). The collection of train and test data, on the other hand, both go through the identical process. They are first flattened in a single dataset by the *Append* operator (5), and the `text` attribute is converted to the text type via the *Nominal to Text* operator (6). The resulting datasets contain two columns: a label column denoting the language, and the text column, containing one sentence per line of the dataset. These datasets are stored in the repository as `language_train` and `language_test` with the *Store* operators (7) and (10), respectively.

Thus far, we obtained the data for language modeling, and we are ready to construct the first language identification system. The simplest and the most naïve approach to language identification is the dictionary-based identification. By enumerating all the possible words in every single language, we can try to match the word with its language and identify the language of the text. However, this method is intractable, since the language is a dynamic complex system with no bound on the number of words. In addition to the complexity, the creation and maintenance of a large dictionary set would create another set of problems, from computational load, to the fact that different languages share same words. Alternatively, we can select a fixed number of words and use them for language identification. Choosing the most frequent words for each language is a reasonable option. This is why the next model we introduce is the frequent words model.

14.5.3 Frequent Words Model

The frequent words approach identifies the most frequent words in the text, and uses them for the language identification. The most used words across different languages have similar meanings, and regarding the type of words, they are usually articles, prepositions, pronouns, and some frequently used verbs, and adjectives. Due to their high frequency of occurrence, they are suitable even for shorter texts. For example, it might be easy to see that the following sentences are written in three distinct languages:

```
Det gäller att smida medan järnet är varmt.[20]
Het is niet alles goud wat er blinkt.[21]
Nu masura pe altii cu palma ta.[22]
```

Nevertheless, it is difficult to identify specifically the languages of these sentences without being acquainted with them, especially if those languages are not widespread. Still, given a list of the most frequent words in Dutch,[23] Romanian, [24] and, Swedish[25] as in Table 14.1, it is straightforward to identify the language of each of these sentences as Swedish, Dutch, and Romanian, respectively.

TABLE 14.1: Lists of the 20 most frequent words in Dutch, Romanian, and Swedish.

Dutch	ik, je, het, de, dat, is, een, niet, en, wat, van, we, in, ze, hij, op, te, zijn, er, maar
Romanian	sa, nu, de, o, ca, şi, a, ce, în, e, am, pe, te, la, mai, cu, un, asta, ai, ma
Swedish	i, och, en, av, som, är, att, den, på, till, med, för, det, han, de, ett, var, har, från, under

Building a statistical model for language identification using the frequent words method in RapidMiner is straightforward. First, we must analyze the training set text, extract all the words from it, and select the most frequent ones. Second, we represent the analyzed text by the most frequent words, converting the text to the word-vector representation in which a value of an attribute denotes the occurrence or a frequency measure of a given word in the text. Finally, we feed the word-vector representation of the text to the Naïve Bayes classifier, and build a model that can classify languages by the most frequent words exhibited in the training set.

We create a single RapidMiner workflow for the purpose of building the frequent words language identification, evaluating it, and applying it on a test dataset. This workflow is presented in Figure 14.4.

The first three operators (1), (2), and (3) execute data loading, textual processing, and model training and performance estimation via 10-fold cross-validation. First we load the `language_train` dataset using the *Retrieve* operator (1). The dataset we load contains two attributes, the *text* attribute containing sentences, and the *language* attribute containing the language of the text as the label. After loading the data, we analyze it with the *Process Documents from Data* operator (2). This operator created word vectors based on term frequencies with the property *Vector creation* set to `Term Frequency`. The inner Vector Creation Subprocess of the Process Documents from Data operator, as can be observed in the middle workflow of the Figure 14.4, contains the *Transform Cases* and the *Tokenize* operators. These two operators are set to the default properties; therefore each sentence of our dataset is first transformed to lower-case, and then tokenized over non-letter characters. We want to limit the resulting number of words in the analysis, to speed up the model learning, as well as simplifying the process. We choose to extract the top 0.1% of all the words in our example set by setting the *prune method* parameter to `ranking`, with the

[20]''You should forge while the iron is hot.''
[21]''All that glitters is not gold.''
[22]''Don't measure others with your hand.''
[23]`http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/Dutch_wordlist.`
[24]`http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/Romanian.`
[25]`http://www.lexiteria.com/word_frequency/swedish_word_frequency_list.html`

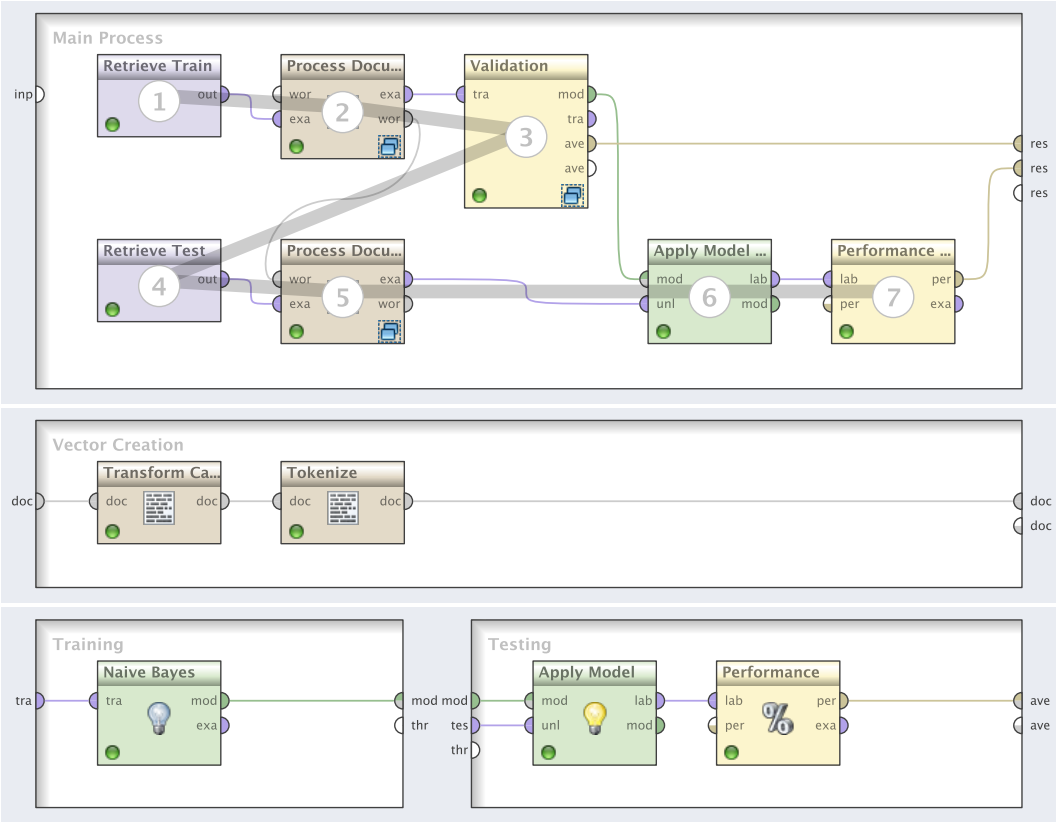

FIGURE 14.4: Workflow for the language identification system based on the most frequent words in languages. The main process is depicted in the top image. The middle image depicts the contents of both *Process Documents from Data* operators (2), (5). The bottom image depicts the content of the *Validation* operator (3).

.

prune below ranking parameter set to `0.001`, and the *prune above ranking* set to `0.0`. The final stage of the modeling phase is the *X-Validation* operator (3), which executes the Naïve Bayes classifier, and estimates its performance through 10 folds.

The next four operators (4), (5), (6), and (7), constitute the application phase of the workflow, meaning that they are responsible for loading the test data, transforming it in the way in which the previously built model was trained, applying the model, and calculating its performance. First, we load the `language_test` dataset using the *Retrieve* operator (4). The test dataset is of the same format as the train dataset. After loading the train dataset, we analyze it in the *Process Documents from Data* operator (5) by passing the WordList from the *Process Documents from Data* operator (2). This ensures that the same words are used for analyzing the test set, and thus ensuring that the dataset is of the same format as the dataset used for model building. The operators inside the *Process Documents from Data* operator (5) are the same as in the model building phase, namely the *Transform Cases* and the *Tokenize* operators. The result of the *Process Documents from Data* operator is a word-vector dataset. The *Apply model* operator (6) then receives the Naïve Bayes model, outputted by the *Validation* operator (3), and applies it on the newly created word-vector dataset. The process ends with the display of the performance.

Note that if we want to separate the model building and the model application into distinct workflows, we should save the resulting WordList, and the model, both built during the model building phase, and load them in the application phase of the workflow of interest.

The cross-validation estimation of the accuracy of the frequent words workflow is equal to $96.19\% \pm 0.35\%$, with the 96.23% accuracy achieved on the test set, as shown in Figure 14.5. It is possible to further improve these results by increasing the percentage of the words included in the model, however this would increase the model complexity. For example, including the top 1% of words instead of the top 0.1%, results in the 99.05% accuracy on the test set.

Table View Plot View

accuracy: 96.19% +/- 0.35% (mikro: 96.19%)

	true deu	true eng	true fra	true por	true spa	class precision
pred. deu	8575	11	16	6	0	99.62%
pred. eng	62	8736	8	3	10	99.06%
pred. fra	13	22	8749	5	35	99.15%
pred. por	10	17	18	8324	53	98.84%
pred. spa	340	214	209	662	8902	86.20%
class recall	95.28%	97.07%	97.21%	92.49%	98.91%	

Table View Plot View

accuracy: 96.23%

	true deu	true eng	true fra	true por	true spa	class precision
pred. deu	20013	28	27	18	3	99.62%
pred. eng	152	20436	11	12	22	99.05%
pred. fra	25	38	20402	30	77	99.17%
pred. por	35	50	39	19403	113	98.79%
pred. spa	775	448	521	1537	20785	86.37%
class recall	95.30%	97.31%	97.15%	92.40%	98.98%	

FIGURE 14.5: Estimated performance of the frequent words method with 10-fold cross-validation (upper image) and performance of the method on the test set (lower image).

Taking a look at the top 10 words extracted from the training set gives us a glimpse of the dataset, and can provide a quick check if everything is in order. This is done by taking a look at the WordList output of the *Process Documents from Data* operator (2), and sorting it over *Total Occurrences*, as presented in Figure 14.6. Most of the top words are, by themselves, bad identifiers of a single language, since they are shared among different

languages. In our example, *a* is shared among all languages analyzed, except German, and *en* is shared among French and Spanish, though *the* is specific only to the English language. We also note that most of the top 10 words do appear in all languages, since it is possible for a sentence in, for example, French to contain foreign named entities such as "El País" or "The New York Times". Though most of the top words are not reliable indicators of a language by themselves, when taken together they achieve a higher accuracy. Note here that the most frequent words are usually discarded in various text mining applications on a single language since those words appear often, and as such do not have the needed discriminating value for various other purposes, though they do in the case of language identification.

Word	Attribute Name	Total Occurences	Document Occurences	deu	eng	fra	por	spa
de	de	27488	15832	30	17	10193	6809	10439
a	a	13553	10592	31	4116	1066	4935	3405
la	la	12868	8542	4	11	5891	39	6923
the	the	9629	5606	13	9525	77	6	8
que	que	9565	7429	0	0	1048	4066	4451
en	en	7372	5738	2	6	2897	1	4466
in	in	5502	4644	2335	3138	18	6	5
el	el	5462	3892	5	6	19	6	5426
o	o	4995	3899	5	19	23	4672	276
le	le	4884	3706	5	1	4605	3	270

FIGURE 14.6: List of the top 10 most frequent words in the language identification dataset, obtained with the frequent words workflow.

The advantages of the frequent words method are its speed, due to relatively small set of words, and high performance, greater than 95%. Higher accuracy can be achieved by including more words, thus increasing the model complexity and its processing time, or by using the two methods we present next. The disadvantages of the frequent words method are problems with similar languages due to similar top words, and bad results on sentences which do not include the top frequency words—usually short words and phrases.

A variation of this technique, short frequent words, uses only words of short length, usually four or five letters long. The rationale of the short frequent words technique is the same, with added shortness criteria which additionally filters out longer words, thus limiting itself to a smaller subset of the most frequent words. This filters out longer words which might be frequent due to linguistic properties of the source of the dataset. Creating a workflow for this technique is simple; it is only necessary to add a *Filter Tokens (by Length)* operator in the *Process Documents from Data* operator in Figure 14.4 (upper image), just after the *Tokenization* operator. The usual length of the token, set for this method, is 5 to 6 characters.

In the following subsection, we present the next method, which differs from the frequent words model on the level of the unit of text used for modeling; it uses n-grams rather than words. This is the character n-gram method.

14.5.4 Character n-Grams Model

We observed that the frequent words method achieves high accuracy, although it performs poorly in cases when the text to be classified does not contain any of the most frequent words. In cases of short sentences, or single words, we want a method capable of achieving high accuracy, without using words as the basis for classification. This is why we turn to character n-grams. As mentioned before, character n-grams are sequences of n consecutive characters, and as such are a powerful feature for language identification. They dissect the

text in its substrings and can catch beginnings and endings of words, as well as some typical character combinations, characteristic to a language. For example, the following sentences:

```
I cannot make out the meaning of this sentence.
Ich kann die Bedeutung dieses Satzes nicht erkennen.
```
[26]

are easily classified into English and German, respectively, knowing that the character n-grams *I*, *can*, *the*, *-ing*, and *of* appear more often in English, and *ich*, *die*, *be-*, *-ung*, *-cht* in German. Notice that n-grams catch not only typical word beginnings and endings, or the combinations of letters, but also the short frequent words of length less than or equal to n.

Building a character n-gram language identification model is essentially an upgrade of the frequent words model, with the resulting workflow differing only slightly from the workflow in Figure 14.4. The conceptual difference is only in employing character n-gram creation after the tokenization, when analyzing the text. The rest of the process is the same: we convert the text to the word-vector representation, and build a classifier to classify it.

The workflow of the character n-gram method, depicted in Figure 14.7, is divided into two parts, i) the model creation consisting of operators (1), (2), and (3) and ii) the model application consisting of operators (4), (5), (6), and (7). As stated, the difference between the character n-gram and the frequent words workflows is minor. In the model building phase, the difference is in the two new operators in the *Process Documents from Data* operator (2), and a modification in properties of two other operators in it. The *Replace Tokens* operator, as seen in the lower image of Figure 14.7, adds underscores around a word, which enables differing cases when an n-gram is found at the beginning or at the end of a word. This is done by replacing the expression `([\w]+)` with the expression `___$1___`. This expression will replace any string of word characters longer than one, with the same string surrounded by three underscores from both the left and the right. The *Generate n-Grams (Characters)* operator creates character n-grams, in our case, n-grams of length 4. Note here that should we tick the *keep terms* option, we would keep the words, in addition to character n-grams, and thus effectively create a hybrid between the frequent words and the character n-gram methods. Two important modifications are apparent in the *Process Documents from Data* operator (2). First, the *prune method* property is set to `by ranking`, pruning all n-grams which occur in more than 10%, and less than 1% of sentences. Second, the *mode* of the tokenization is set to `regular expression` with the *expression* equal to `[^a-zA-Z_]+`. This regular expression describes all the splitting points. In this particular case it defines that the splitting points are everything NOT (`^`) containing strings of all the lower- and upper-case letters, and the underscore (`a-zA-Z_`), of string length equal to at least one character (`[]+`). By adding additional characters or character ranges in this expression, we influence tokens used for the tokenization. For example, by adding `0-9` to it, we enable numbers and words with numbers and tokens, by adding `'` we include apostrophes in words, and by adding various other punctuation characters we can catch smileys and interjections. The rest of the workflow stays the same as the workflow for the most frequent words model.

When executed, the character n-gram workflow, depicted in Figure 14.7, achieves 98.26%±0.14% accuracy of the estimated performance with 98.31% accuracy on the test set, as observed in Figure 14.8. By increasing the number of n-grams included in the analysis, we improve the accuracy of the method.

When compared to the frequent words method, character n-grams exhibit an important property—they create a compact representation of text into a finite number of n-grams. This representation is a priori limited by the number of n-grams, whereas the word count in a language is not limited at all, due to the fact that new words are being produced constantly, i.e., language is dynamic.

[26]Both of these sentences have the same meaning. Source: Tatoeba Project.

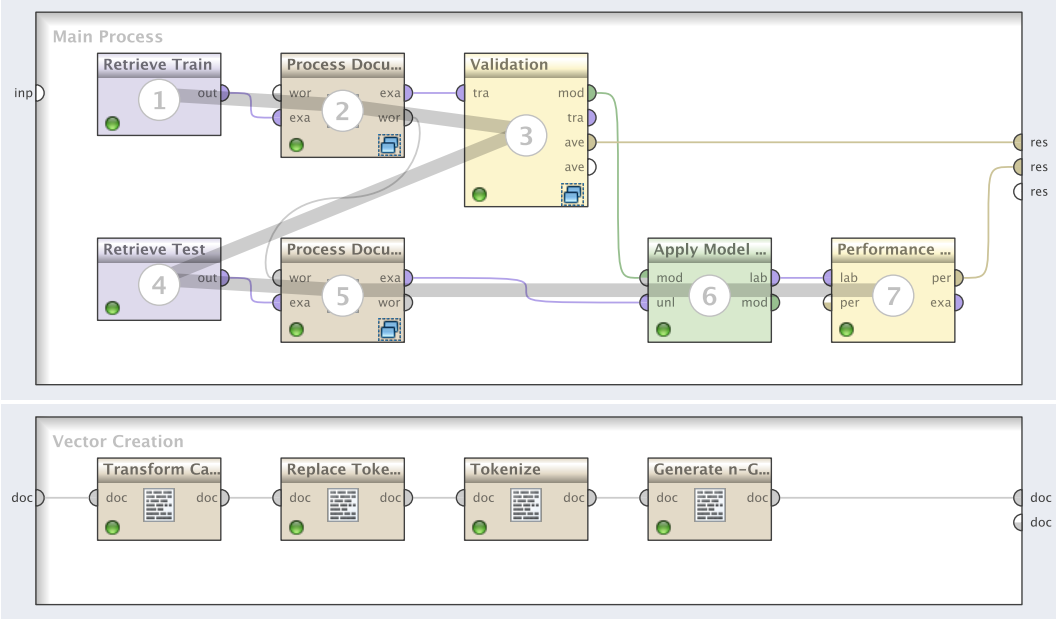

FIGURE 14.7: The workflow for the language identification system based on character n-grams depicted in the upper image. The lower image depicts the content of both *Process Documents from Data* operators (2), (5). The content of the *Validation* operator (3) is the same as in Figure 14.4.

Table View Plot View

accuracy: 98.26% +/- 0.14% (mikro: 98.26%)

	true deu	true eng	true fra	true por	true spa	class precision
pred. deu	8965	14	11	7	32	99.29%
pred. eng	5	8906	18	10	32	99.28%
pred. fra	24	64	8929	32	87	97.73%
pred. por	4	6	15	8764	196	97.54%
pred. spa	2	10	27	187	8653	97.45%
class recall	99.61%	98.96%	99.21%	97.38%	96.14%	

Table View Plot View

accuracy: 98.31%

	true deu	true eng	true fra	true por	true spa	class precision
pred. deu	20923	20	28	14	75	99.35%
pred. eng	18	20823	56	17	88	99.15%
pred. fra	42	118	20832	62	205	97.99%
pred. por	7	19	19	20475	458	97.60%
pred. spa	10	20	65	432	20174	97.45%
class recall	99.63%	99.16%	99.20%	97.50%	96.07%	

FIGURE 14.8: Estimated performance of the n-grams method with 10-fold cross-validation (upper image) and performance of the method on the test set (lower image).

Read more in the complete book, which can be purchased from here:

http://www.crcpress.com/product/isbn/9781482205497

…or most other book stores

Subject Index

Operator Index