FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Performance Analysis of Big Data Tools Based on Benchmarks for Store Sales Forecasting

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Kateryna Zaslavska
Matrikelnummer 1129108

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Priv.-Doz. Dr. Ivona Brandic

Wien, 21.08.2014

(Unterschrift Verfasserin) (Unterschrift Betreuung)

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Performance Analysis of Big Data Tools Based on Benchmarks for Store Sales Forecasting

MASTER THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering & Internet Computing

by

Kateryna Zaslavska
Registration Number 1129108

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Priv.-Doz. Dr. Ivona Brandic

Vienna, 21.08.2014

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Kateryna Zaslavska
Karlsplatz 13, 1040 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Wien, Datum) (Kateryna Zaslavska)

Acknowledgements

I would like to express my sincere gratitude to Univ.Prof.Dr.Full Professor Schahram Dustdar for his lectures and all the efforts he made to share his experience, knowledge and ideas with us.

I will forever be thankful to Univ.Prof. Dipl.-Ing. Dr.techn. Hannes Werthner for the opportunity of studying the Software Engineering and Internet Computing Master program. I highly appreciated for all the knowledge and learning outcomes I received in our university.

I would like to express my deep gratitude to Priv.-Doz. Dr. Ivona Brandic, my supervisor for this master thesis, for her useful advices, remarks, help, encourage through the learning process. I had a great pleasure to work with her. I would like to thank her for introducing me to the topic during interesting and useful lectures as well for the support on the way.

Also, I like to thank MSc Drazen Lucanin, who have willingly shared his precious time during the process of data search and analysis. I would like to acknowledge Renate Weiss for academic and technical support.

And finally, I would like to thank my friend and husband, Dmytro Grygorenko, my parents, Vladimir and Oksana Zaslavska and my sister Olena, who have supported me throughout entire process, for their love and both by keeping me harmonious and helping me putting pieces together. All I made would not be possible without their great belief in me.

Abstract

In the past few years the volume and variety of Big Data significantly increased. With the large growing amount of massive unstructured data the importance of its processing, storing, aggregation, analysis, and derivation of valuable information becomes stronger.

With the increasing amount of Big Data types and sources the number of enterprise and open source applications, techniques and resource usage models for big data analysis constantly rises. Most of them provides basic data mining techniques, however, there is still a significant difference in their possibility to scale, visualization capabilities, performance, extensibility, and processing of various data storages. Although, the large number of materials describing strengths and weaknesses is available, they do not supply business members with the understanding whether a certain application will fit their real problems in a way that provides effective decision-making.

The main goal of this thesis is to investigate and compare such applications and tools for big data analysis as Weka, KNIME, Apache Mahout, and R using several time series real data sets and compare their applicability in the context. For these purposes, we develop and apply suitable scenarios that involve the usage of various data mining techniques and contain data analysis, data transformations, model accuracy assessment, and visualization. Based on achieved results big data analysis applications are evaluated and compared using multiple quantitative and qualitative measurements.

Kurzfassung

In den letzten Jahren sind Volumen und Vielfalt der sogenannten Big Data signifikant angestiegen. Mit der großen und wachsenden Menge an massiv unstrukturierten Daten wird die Bedeutung ihrer Verarbeitung, Speicherung, Aggregation und Analyse sowie der Ableitung von wertvollen Informationen größer. Mit der steigenden Anzahl an unterschiedlichen Arten und Quellen von Big Data wächst auch die Anzahl an Unternehmen und Open-Source-Anwendungen, Techniken und Modellen zur Ressourcennutzung für die Analyse von Big Data stetig an.

Die meisten davon bieten grundlegende Techniken des Data-Minings an. Dennoch besteht ein signifikanter Unterschied in ihren Möglichkeiten der Skalierung, der Visualisierungsfähigkeit, der Leistung, der Erweiterbarkeit und der Verarbeitung von verschiedenen Datenspeichern. Obwohl zahlreiche Unterlagen verfügbar sind, in denen die Stärken und Schwächen beschrieben werden, beantworten diese nicht die Frage der Mitglieder dieses Geschäftszweigs, ob eine bestimmte Anwendung auf tatsächliche Probleme angemessen ist und sie zu einer effektiven Entscheidungsfindung führt.

Das Hauptziel dieser Arbeit ist es, solche Applikationen und Werkzeuge für die Analyse von Big Data, z.B. Weka, KNIME, Apache Mahout und R, zu untersuchen und miteinander zu vergleichen. Dafür verwenden wir mehrere Zeitreihen realer Datensätze und vergleichen ihre Anwendbarkeit im Kontext. Zu diesem Zweck entwickeln und nutzen wir geeignete Szenarien, die verschiedene Techniken des Data Mining anwenden und Datenanalyse, Datenumwandlung, Bewertung der Genauigkeit des Models und Visualisierung beinhalten. Basierend auf den gewonnenen Ergebnissen werden Applikationen für die Analyse von Big Data bewertet und verglichen, indem zahlreiche quantitative und qualitative Messungen verwendet werden.

Contents

List of Figures

List of Tables

Listings

CHAPTER 1

Introduction

In recent years the global marketplace has been grown exponentially in data volume. Every day people generate tremendous amount of unstructured data of various types such as GPS coordinates, payment transactions, web data, emails or smart meter values that are termed as Big Data. The analysis of Big Data is widely used in insurance, medicine for disease prediction and improved health outcomes, industry for sales prediction and customer relationship optimization, and transport.

The ability to store, aggregate, combine data, and derive valuable information from it requires the development of specific tools that are based on techniques as data mining, statistics, and other advanced analytics methods. Data mining is a process of discovering patterns and hidden relations in data [24] with the help of statistical analysis, artificial intelligence, machine learning techniques. Statistical methods are useful for analysis of available data, trend definition, calculation of value frequencies, and estimation of numeric characteristics in data. Machine learning can be applied for such business cases as product recommendation, segmentation of customers, fraud detection, where data silos is used for building and training model.

The main problem nowadays is that the amount of data grows faster than the capabilities for their processing. Therefore it is important to apply all forces for the development of innovative computational models such as distributed computing, efficient usage of shared resources in order to give a possibility for developers to hardly focus on the building of applications than worry about lack of resources.

The software that creates, store or process data can provide a value in different ways. Statistical analysis can be used to obtain research. Real time analysis can help to define inefficiencies in product development. Graphical representation can be useful for streamline decision making [38]. We would like to notice a widely used mistake in the case when people initially start from technology and then concentrate on the problem they try to solve. There are various types of technologies that are available for the final user, but it is necessary to keep in mind the business questions the solution has to answer.

1.1 Problem Description

In spite of large amount of tools, end user who has a certain practical use case to solve should be able to choose definitely on the base of personal knowledge, facilities and OS limitations a certain software application that in an appropriate time will return the expected result. The first steps to start application selection can be seen as a choice of data storage (e.g., relational database or key-value storage), file format (how data are represented, e.g., CSV, RDF, ARFF or JSON format) and data representation style (e.g., charts, dashboards, 2- or 3-dimensional graphs).

Taking into consideration observations described above the problem statement can be summarized as follows: there is no efficient and useful comparison of several big data analysis applications on the base of same real dataset that guides a stakeholder towards desired way of data analysis and decision-making.

Main initial goal of big data tool effort is to propose such application comparison, develop widely used use cases that can simulate real data tasks, analyse data mining steps for each use case and evaluate tools on the base of achieved results.

1.2 Motivation

The amount of paid or open source tools, techniques and resource usage models for big data analytics [47] as following: statistical analysis, online analytical processing (OLAP) tools [44], data warehouses (DWH) [44], distributed programming models (e.g., MapReduce [11]), clouds [35], complex event processing [48] constantly rises. Most of them provides basic data mining techniques, however, there is still a significant difference in their possibility to scale, visualization capabilities, performance, extensibility, and processing of various data storages.

The primary research goal of the proposed research is to investigate and compare applications for big data analysis with focus on their performance, visualization possibilities, database access, preprocessing, modelling, and other related application characteristics. Usage of open-source products in read conditions enables analysis of its strengths and limitations. Although there is a large amount of applications for big data analysis, not all of them can provide such opportunities as scalability and fast queries processing for large data corpuses [47].

The idea is to evaluate different applications for big data analysis using three time series real data sets that contains historical data about customers, products and enterprises with focus on performance, and to compare their applicability in this context. Based on this, two main goals of the thesis can be distinguished.

The first task is to define input data sets for big data analysis tools. To solve it we made a research on available open-source benchmarking applications and data sets and chose several of them. We will solve three general big data challenges with the help of these applications, investigate the benchmarks for the use cases with a focus on volume (e.g., terabytes of data, records in the data files), velocity (e.g., real time data or streams), and variety (e.g., structured, unstructured, and semi-structured data) [39, 47]. For this purposes we will develop use case for each application containing steps as data source analysis, filtering, data partition, file transformations, visualization, and reporting. Since not all tool characteristics and scenarios can be tested in de-

tails, we will identify steps for which we can build a model, apply an algorithm, evaluate, and document achieved results.

The second task is to make evaluation of the chosen applications that support big data analysis on the base of achieved results. The comparison of existing tools will be based on different criteria: possibility of multi format file usage, availability of methods for data transformation, classification and clustering, and visualization capabilities. Feasibility of tools for a specific use case will be measured under the following objectives such as completeness, effort to develop a model, and effort to run it. We will use two methods of evaluation. First one is quantitative that means the measurements of such characteristics as performance and resource consumption. Benchmarking will be the instrument for such method. We will calculate memory usage and record the overall time to model the use case and run procedures. In addition the applications will be evaluated qualitatively that intends comparison of reliability, visualization possibilities, and feasibility. In this case we will estimate the ease of data representation, and record a number of system failures. Such analysis will be a good purpose as an introduction to the application and as an example of how raw data can be transformed into meaningful information.

Two open source applications KNIME [32, 40, 52] and Weka [24, 55] and two open-source software packages R language package [10,43] and Apache Mahout (Apache Software Foundation) [2, 47] will be used for analysis. Each use case will represent one of widely used general task: sales forecasting, classification, and clustering. The detailed description of the task, stepwise analysis of work with each application and result assessment with graphical representation will show the advantages and disadvantages of certain tool for a specific data set and will present applications in more convenient way from the user point. We will categorize these characteristics based on tool specifications in the following way: application characteristics (e.g., data format, store access, filters, data manipulation and transformation, data mining and prediction algorithms), visualization capabilities, and model evaluation parameters (e.g., overall performance, effort on data preparation). Such parameters as file size limits, overall completeness, response time, reporting possibility, converting ability or tool simplicity will be compared as well.

In this work we will categorize applications and make their evaluation according to the proposed use cases. It is hard to understand the differences between big data analysis tools that can be applied for solving analogous tasks and that contains similar characteristics, functionality, without trying them out on benchmarks with real data or reading their documentation and technical reports.

1.3 Contributions of the thesis

Comparison of big data applications is a multiple-aspect problem, where common use case tackling the challenge of collection and storing, cleaning data silos, reviewing the relationships between attributes, analysing evaluation metrics. On the other hand it includes various solutions, frameworks, platforms, databases and visualization tools. They vary from open source and free to paid, from simple to complex, with basic support data mining techniques and with of full support that includes scalability possibilities or data storage, from designed for a single user or stakeholder to programs that support corporations.

Although, the large number of working examples, tutorials, forums and manuals that describing strengths and weaknesses is available, they do not supply business members with the understanding whether a certain application will fit their real problems in a way that enables to unlock the value from the big data silo and provides effective decision-making.

Exactly this reason became a motivation to compare the widely used popular applications such as Weka, Apach Mahout, R and KNIME. We will use benchmarking as a methodology for the further research, because only application of these tools to real data sets and modelling popular data analysis problems enable to simulate them in real conditions. The described evaluation type makes it possible to to assess the relative performance of an application, find out pitfalls, weaknesses and strengths.

It should be noted that the described aspects were not yet considered for chosen applications during investigation of the problem of big data tools comparison. Therefore the results of thesis investigation should become an important contribution into current big data analysis research.

1.4 Structure of the work

Chapter 2 will describe the big data paradigm and provide a research on available application solutions for big data analysis. Each tool chosen for data evaluation different in their features, complexity, performance. Their main characteristics will be described in this Chapter.

In Chapter 3 we will overview three time-series benchmarks that represents real data corpuses, analyse such data properties as quality, and variety.

Chapter 4 will continue with the relevant related works in the area of tools for big data analysis and tools for benchmarking.

In Chapter 5 we will investigate all technical aspects related to building use cases and performing simulations: used technologies, use case workflows, implementation of applications for big data analysis. The appropriate section will describe more deeply the evaluation methodology, results for use cases and tools assessment. All of these tools are different in their possibilities, features and complexity. Therefore, we will define the main characteristics for each group of data analysis and examine each application for compliance with this characteristics.

Possible improvements, final remarks and overview of the future work will be discussed in Chapter 6.

CHAPTER 2

State of the Art Review

This chapter contains the description of main tools for big data analysis and chosen tools for evaluation. We will focus on the following important aspects of software applications such as flow semantics, specification, user interface and principles of working with large data sets.

2.1 Evaluation of Big Data Applications

What is Big Data?

Big Data represents the large data sets that are challenging to store, share, visualize and analyse. To get the value from such data it is necessary to find alternative ways to process them. Company data can contain valuable patterns and hidden information, that for such companies like Amazon, Google or Walmart can be important from the purpose of analytical use, enabling to recognize new patterns or find trends in production and consumer behaviour. Nowadays with a rapid growth of popularity of cloud architectures, distributed frameworks and open source software tools for data mining, big data processing becomes better resourced. Big Data can be characterized by three Vs: volume, variety and velocity [39] (see Figure 2.1).

- *Volume*. The ability to store and process large amount of information is one of the key ideas of big data analysis. The results from running forecast on 100 attributes than on three or defining classification only on 100 of rows can be more accurate. Large data processing requires scalable storages, approaches to query data and distributed computing applications to build models.

- *Velocity*. The importance of fast utilization of streaming information is another crucial challenge in big data processing. The issue is not only the storing of data in proper format but also the frequency of data coming and the time for getting a feedback from the input. If company generates incredible amount of data (e.g., Large Hadron Collider) some part of information should be discarded using certain rules on the level of data collection or

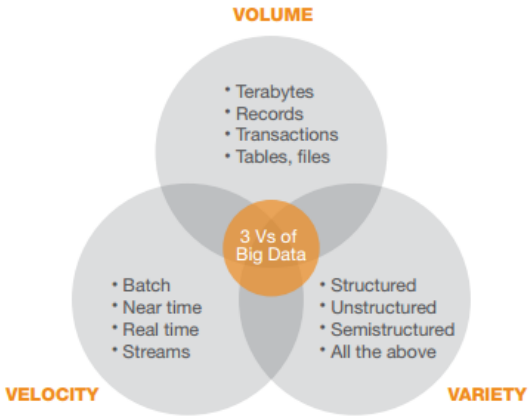


Figure 2.1: The three Vs of Big Data [47]

pushing into a data storage. For the companies like FIFA the feedback loop from on-line games in real time defines greater competitive advantage.

- *Variety*. A common case when deal with a Big Data is a processing of unstructured data from different sources (e.g., text from social networks, images, songs of various format, click streams, logs, geospatial data) with further extraction of the meaning. Data variety requires various types of data storages like semi-structured NoSQL databases. Data can be semi structured (XML and similar standards), complex (hierarchicall sources), unstructured.

There fourth V characteristic of data is a data *value*. Data can be transformed into information useful for marketing strategies, business innovations. New data value can be generated from the interaction of data varieties and can be separated into two categories: analytical use and enabling of the new company products. Data value can contain revealed hidden previously data dependencies and their reasons of these dependencies such as customer product preferences on account of geographical location or social level [39, 53].

Big Data Tools

Solutions for big data analysis are provided in four forms: software-only, cloud-based solutions, appliance and hybrid. All of them are based on the idea of raw data transformation into actionable outcome that involves the following steps: definitions of the objectives, data collection, selection of the model and its further simulation and optimization in order to achieve the objectives (see Figure 2.2).

Newest techniques to support big data analytics and processing contain parallel processing, indexing mechanisms, interactive visualization, and complex SQL.

Term of Big Data is not new. Companies started to analyse their information a lot of time ago. But now they try to do it in more sophisticated way and require for that new technologies [47]. Tools for Big Data can handle with different aspects of data analysis from storage to

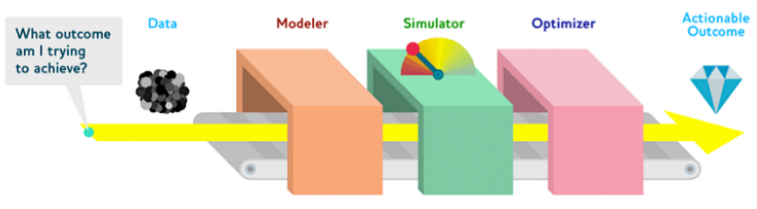


Figure 2.2: Process of raw data transformation into actionable outcome [39]

visualization, and therefore they are inherently dependent on the variety of technologies (e.g., clouds, MapReduce, complex event processing). Frameworks should be able to cope with data diversity, delivered at the various speed, filter and pre-process information, perform tasks like prediction, classification, clustering, query information from storage in comparatively short time frame.

The applications can be divided into the following fundamental categories [6]:

- *Data analysis platforms*: the platforms that support distributed computing techniques and can include storing and analysis tools (e.g., Apache Hadoop [1]).
- *Databases and DWH*: the applications that focus on data storage, archiving, running joins and sorting. They include the following storage types: document store (MongoDB [2]), graph databases (Neo4j [3]), key-value stores, table based stores, object data stores, RDBS (Drizzle [4]) and other databases (Hive [5], Cassandra [6]) and data warehouses (Teradata [7], OLAP tools).
- *Business Intelligence tools*: the applications that includes filtering, indexing, extract, transform, and load (ETL) processes, report generation, aggregation. Commonly used are Birt [8], Talend [9], Pentaho [10], ERP BI Solutions.
- *Data mining tools*: tools that includes machine learning, information extraction, recommendation analysis, behaviour analysis. The examples are Weka, RapidMiner [11], Mahout, KNIME.
- *Programming tools*: platforms that includes programming language and frameworks for statistical analysis and data mining. Commonly used is R statistical platform.

[1] http://hadoop.apache.org/
[2] http://www.mongodb.org/
[3] http://www.neo4j.org/
[4] http://www.drizzle.org/
[5] https://hive.apache.org/
[6] http://cassandra.apache.org/
[7] http://www.teradata.at/
[8] http://www.eclipse.org/birt/
[9] https://www.talend.com/
[10] http://www.pentaho.de/
[11] http://rapidminer.com/

- *Visualization tools*: the applications that focus on visualization of stream data, data from databases or files. The example of such tools are Tableau [12], Gephi [13]. Alternatives of such tools can be based on such characteristics: connectivity of different types of entities, flexible semantics, extensibility.

The commonality between all the tools is the possibility to combine four key features: increase a memory limit, provide data persistence and data storage access, ensure possibility of multi threading and data exchange between nodes [36].

KNIME

Overview. KNIME, the Konstanz Information Miner [32] is an open source data analytics platform for data-driven innovation, machine learning and data mining. KNIME provides a graphical workbench for building workflows and more than 1000 nodes for analysis process: data access, transformations, predictive analytics, visualization, reporting and results evaluation [52]. KNIME products include additional functionalities such as big data extensions for distributed frameworks, cluster execution, community and partner extensions, authentication, shared repositories, SOA integration and security through the enterprise, and remote execution.

The KNIME product suite contains the following parts [33]:

- *Analytics Platform*: an open source platform for data access, data mining, statistics, visualization and reporting. It allows to analyse trends and predict potential results. The platform has a visual workbench.
- *TeamSpace*: the extension to KNIME analytics Platform. It can be used by teams to share workflows and metanodes.
- *Server*: central facility to store and access KNIME workflows.
- *Cluster Execution*: connection layer between KNIME and a cluster. The compute cluster can be used for submission of workflows, splitting large data sets.

KNIME Workflow. KNIME Desktop provides the data analytics routines for such topics as Time Series analysis, Data Mining, Web Analytics, Image Processing, Network Analysis, etc.

A workflow in KNIME is formed from consecutively connected basic processing units called nodes that belong to different categories (e.g., readers, learners, predictors) and are combined via ports. A connection can transfer data (e.g., from database, files) or generate models (e.g., predictors, learners). Visual workbench contains such steps like data access (read data), data transformation (e.g., clean, filter data), initial investigation, analytics (train a model) and visualization. Each node has a number of input/output ports and node status. Node status has three states (Figure 2.3) [31]:

- *not executable (red colour)*: node is not ready for execution, a node is not configured yet or configured incorrectly.

[12] http://www.tableausoftware.com/
[13] http://gephi.github.io/

- *ready for execution (amber colour)*: node is configured correctly and ready for execution.
- *executed (green colour)*: node is executed, results are available at the output port(s).

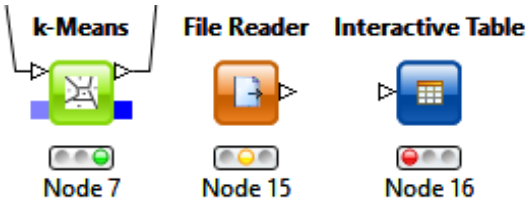


Figure 2.3: Three states of node status in KNIME: executable, ready for execution and not executable

Each node has a configuration dialogue to set a required parameter (e.g., path to the input file, number of clusters, number of cross validation runs).

There are several types of node ports (Figure 2.4):

- *data port*: it is used to transfer flat data tables (white triangle).
- *database port*: it is used to run commands and data transformation inside a database (brown square).
- *PMML port*: it is used to pass data model for the referring predictor (blue square).
- *other ports*: (grey square).

The background of nodes indicates node types (Figure 2.4):

- *source*: green node. It is used to display any kind of data source.
- *sink*: blue node. It is used to display any kind of persisting nodes.
- *manipulator*: brown node. It is used to display transforming nodes.
- *view*: yellow node. It is used to display data.
- *metanodes*: grey node. This are nodes that contain subworkflows of other nodes.

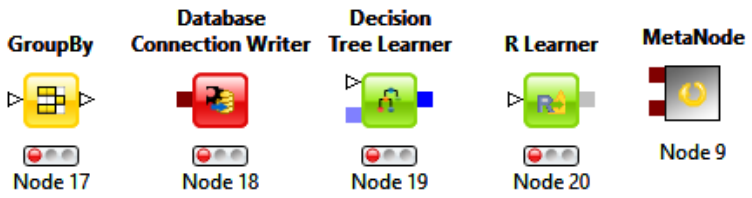


Figure 2.4: Various types of ports and nodes in KNIME

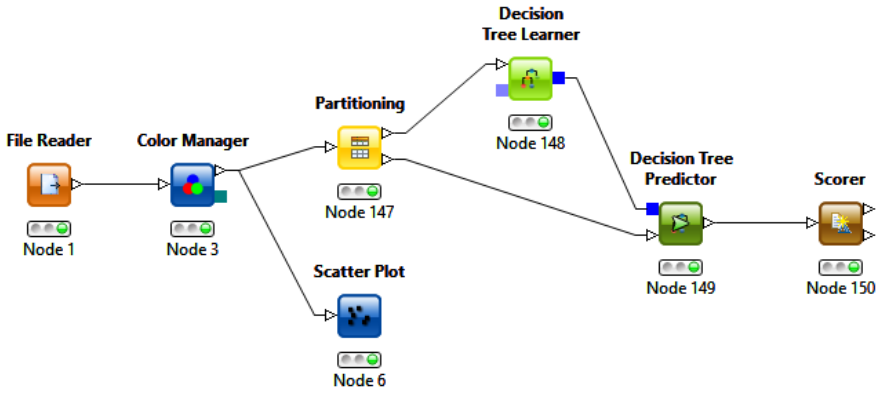


Figure 2.5: Example of the classification workflow in KNIME using *Decision Tree Learner* and *Predictor*

Additional nodes can be installed like plug-ins for KNIME (e.g., R, Weka, Python, Hadoop) or the integration with Chemistry Development Kit with nodes for processing chemical structures. After the workflow is built it is possible to execute the whole workflow or a part of it. This process is automatic.

Data in KNIME are wrapped in a table-based format and consist of columns and rows. In addition, KNIME holds meta information concerning the type of the column. Each node of the workflow processes the whole data from the input port and forwards then the entire results on the output port. This approach is useful for data mining tasks because some algorithms need the whole data and not the stream data.

Figure 2.5 represents the classification use case that applies Decision Trees algorithm. The workflow example contains the following nodes: *File Reader* to read data from file, *Color Manager* to assign colours for each data class, *Scatter Plot* to visualize input data, *Partitioning* to split input data into train and test sets, classification nodes *Decision Tree Learner* and *Decision Tree Predictor* to build model and further predict the class value for the test data, and finally *Scorer* to compare results.

Reporting. KNIME provides the ability for reporting, where the results can be stored in common formats like PDF, XLS, PPT and other. Data from any part of workflow can be combined and used for reporting. The official KNIME website offers additional plug-ins for reporting via BIRT.

Parallel and Distributed Computing. There are several possibilities to parallel a workflow in order to perform tasks faster on Big Data [8]:

- *Parallel execution of independent nodes*: several independent nodes can be executed in parallel. KNIME workflow manager keeps track of queuing and execution of the nodes.

- *Parallel processing of data in a single node*: number of nodes work independently from other nodes with different data. In KNIME they are called Threaded nodes.

- *Parallel processing of sub-workflows*: there is a concept of metanodes, which contains subworkflows or other nodes. This node can be reused (e.g., looper or cross validation). In case of using metanodes, it is possible to split data into different partitions and aggregate results in the end.

The main features of the KNIME Analytics Platform are [13]:

- graphical user interface (GUI): the graphical representation makes it easier for users to keep an overview of the analysis process
- database support
- workflow principle of building data analysis: each analysis algorithm is implemented like a series of calls to functions
- expandability: there is a possibility to extend KNIME with new nodes or views.
- modularity: nodes not depend on each other
- reporting, views and interactive brushing: Each node has a number of views. In KNIME it is possible to highlight rows in tables or different types of plot charts [40].
- big data analysis: KNIME can process large amounts of data, because data tables are not kept completely in memory but can be shifted on the disk.

R Project

Overview. R is an open source software for statistical computing (linear, non linear modelling, time-series analysis, classification) and graphics. R is highly extensible and includes around 4000 packages in CRAN package repository for different tasks. Main tasks that are related to data mining are cluster analysis, machine learning, statistical learning. R commonly referred to as the R language was designed in 1993 [17]. The main features of this software are:

- support of data analysis (missing values, sub-settings, data frames)
- powerful packages for reporting
- strong functional programming facilities, meta programming
- connection to high-performance programming languages like C, Fortran, C++, Python
- graphical facilities
- intuitive syntax
- dynamic typing and evaluation R initialization is lazy. Evaluation of arguments in function is delayed. R is an extremely dynamic programming language. Almost anything can be modified after it is created.

R language core consists of a set of CRAN libraries and packages as well as running engine that can interpret user commands. There is a variety of open source and commercial IDE for R language. The most commonly used is RStudio [14].

One R script example is displayed in Listing 2.1 K-means clustering algorithm takes as input two parameters $data$ refers to dataset and figure 3 refers to cluster number and further visualize of obtained results using additional library qplot. Information illustrated on Figure 2.6 reflects the dependency between car age and age of their owners can be interpreted as follows: 1 is a *red coloured* cluster represents older people that have new cars, *blue coloured* cluster shows people with older car and *green coloured* cluster defines in majority young people who owns old cars.

Listing 2.1: Test example of the K-means clustering algorithm in R

```
1 kc3 <- kmeans({data}, 3)
2 # plot the result
3 qplot(car_age, age_youngest, colour = clusters3)
```

Figure 2.6: Example of K-means clustering algorithm results in R [46]

Data Structures. R contains different data structures that can be organized by their dimensionality into three groups: 1-, 2-, and n-dimensional [46]. Also they are separated by the type of the data that can be stored within them into two groups *homogeneous* and *heterogeneous* (Table 2.1). Homogeneous group includes *atomic vector*, *matrix* and *array* data structures and heterogeneous group includes only 1- and 2-dimensional data structures named *list* and *data frame* respectively.

The most commonly used structure in R is a data frame that is an analogue of the table in relational database and can store lists of values of different type. Listing 2.2 shows the code

[14] http://www.rstudio.com/

	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data frame
nd	Array	

Table 2.1: Data structure in R

example necessary to create a data frame using method *data.frame()* (line 1). Output includes statistical information about structure of data frame (lines $3-5$) and the data frame itself (lines $7-11$), where x and y refers to columns.

Listing 2.2: Example of a data frame in R

```
1  df <- data.frame(x = 14, y = c("a", "b", "c","d"))
2  str(df)
3  #> 'data.frame':        4 obs. of  2 variables:
4  #> $ x: num  14 14 14 14
5  #> $ y: Factor w/ 4 levels "a","b","c","d": 1 2 3 4
6  df
7  #>    x y
8  #> 1 14 a
9  #> 2 14 b
10 #> 3 14 c
11 #> 4 14 d
```

Each structure can be extended with meta information using associated attributes (e.g., length, dimensions, and column names).

R and Large Data Sets. R language was designed for running on single-machine system. Therefore it works slower on large data sets. R is designed to run completely in memory. Such in-memory structure can lead to high-throughput data problems. As long as for 32-bit machines the total available memory for R is 4 GB, one possibility to get round the limitations is to use 64-bit machine. Another possibility is to clean a R garbage collector after processing large data set manually. One more decision for working with large data sets is the usage of the library `bigmemory` that attempts to apply an extra level of memory for data. Another possibility is to use RHadoop project, popular library to connect R to Hadoop and define MapReduce jobs that can run fast. Despite the memory limitation R is flexible and popular tool.

Weka

Overview. Weka is an open source software implemented in Java that contains machine learning algorithms for data mining tasks. Data mining is about using data analysis techniques to define previously undetected relationships or patterns in data [24, 41]. Weka is the product of the University of Waikato (New Zealand) and is distributed under the GNU General Public License; the name stands for Waikato Environment for Knowledge Analysis. Weka is a cross-platform solution that has an API and GUI and all algorithms for data pre-processing, classification,

clustering or data regression can be used directly or called from own Java code or command-line. Weka includes not only a variety of learning algorithms, but a large number of preprocessing tools, provides a large variety of filtering algorithms for instances and attributes separately. User can compare methods and define most appropriate for the given task. Weka product contains three interactive interfaces that encapsulate all basic functionality [14, 24]:

- *Explorer*: Weka's main graphical user interface. It holds every data file in main memory and therefore can be used for solving small- or medium- data size issues. Explorer contains several tabs: preprocess, classify, cluster, associate, select attributes and visualize.
- *Experimenter*: is used when several algorithms will need to be tested with different parameters to find the appropriate one. The Experimenter can queue several data mining approaches for different data sets. At the end of the run Experimenter collects performance statistics (e.g., percent of classified/not classified data, mean error), compares methods and perform significance tests.
- *KnowledgeFlow*: is used to build a visual workflow by dragging boxes that represent data source and learning algorithms. In this alternative way tasks can be performed without loading the entire set into memory, that is more efficient during big data evaluation. It is used for stream data processing.
- *SimpleCLI*: a command-line interface

Figure 2.7 illustrates the implementation of J48 Tree algorithm in KnowledgeFlow. The workflow example contains the following nodes: *CSV Loader* to load data from file and convert automatically in ARFF format, *NumericToNominal* to turning numeric attributes into nominal ones, *Class Assigner* to designate a class column in input data set, *CrossValidationFoldMarker* to split input data into train and test sets and send both outputs to the classifier, *J48* to perform classification on the base of train and test data and finally *BatchClassifier* to compare results.

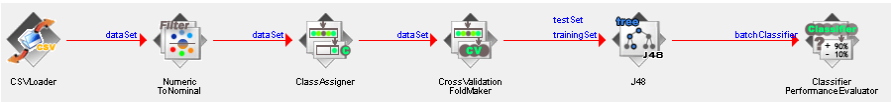


Figure 2.7: Example of data workflow for classification algorithm J48 in Weka KnowledgeFlow

Data Format in Weka. Weka's native method of data storing is the Attribute-Relation File Format (ARFF), that contains two distinct sections. The beginning of the file is a *Header* information section that consists of the list of instances, attributes (that correspond to columns in the data source) presented in a file (e.g., *car_age*, *risk_factor*, *duration_previous*) and their types (see Figure 2.8) [14]. Nominal attributes are followed by the list of values they can take on (e.g., for attribute *homeowner* values are $0, 1$). Numeric values are followed by the keyword `numeric`. The *Data* information section $@DATA$ provides the actual data values. Such declarations as $@DATA$, $@ATTRIBUTE$, and $@RELATION$ are case insensitive. Lines that starts from the $\%$ are comment lines and are used to indicate the stores, meaning or context of the data . The file that is loaded in Weka can be changed using converter to ARFF

```
@attribute homeowner {0,1}
@attribute car_age numeric
@attribute car_value {g,e,c,d,f,h,i,b,a}
@attribute risk_factor {1,2,3,4}
@attribute age_oldest numeric
@attribute age_youngest numeric
@attribute married_couple {0,1}
@attribute c_previous {1,2,3,4}
@attribute duration_previous numeric
@attribute cost numeric
@attribute A {0,1,2}
@attribute B {0,1}
@attribute C {1,2,3,4}
@attribute D {1,2,3}
@attribute E {0,1}
@attribute F {0,1,2,3}
@attribute G {1,2,3,4}

@data
0,0.023529,g,3,0.491228,0.440678,1,1,0.133333,0.644097,1,0,2,2,1,2,1
0,0.117647,e,4,0.175439,0.20339,0,3,0.866667,0.8125,0,0,3,2,0,0,2
0,0.129412,c,?,0.438596,0.457627,0,2,0.266667,0.588542,0,0,1,2,0,0,1
1,0.035294,d,3,0.77193,0.745763,1,3,0.2,0.630208,1,1,3,2,1,1,3
0,0.058824,d,3,0.245614,0.20339,1,1,0.133333,0.609375,1,1,1,1,0,2,2
0,0.070588,d,?,0.105263,0.135593,0,3,0.6,0.579861,2,0,3,3,0,3,3
```

Figure 2.8: Weka ARFF file example

file format automatically. User should provide such information in converter as field separator, missing values symbol, and date format.

Distributed Processing in Weka. Weka can not be applied to large data sets without additional tuning for training data models. One possibility is to increase Java heap size. But sometimes it does not help, because the Explorer uses entire memory and it can incurs overhead. Another possibility is to use Command-Line Interface (CLI), the Knowledge Flow GUI or write scripts on Groovy or Jython. One more decision for dealing with big data sets is to use additional `massiveOnlineAnalysis` library with access to Massive On-line Analysis (MOA) [15] data stream software [56].

Weka version 3.7 contains new packages `distributedWekaBase` for data mining using MapReduce paradigm. Another package `distributedWekaHadoop` provides connection to Hadoop, number of utilities for its configuration, and Hadoop Distributed File System (HDFS).

Advanced users can use a new feature in the Experimenter GUI form Weka and distribute tasks between several processors [24].

In collaboration with Pentaho Weka team developed additional plug-in for Time Series and Forecasting for Weka Explorer then can be installed via the package manager. The key feature of this plug-in is the possibility to investigate trends and seasonality in time-series data [57].

[15] http://moa.cs.waikato.ac.nz/

Apache Mahout

Overview. Apache Mahout is a scalable machine learning library that was started in 2008 as a sub-project of Apache Lucene [2]. Mahout is the set of commonly known algorithms that can be executed in parallel in distributed environments. Currently Apache Mahout utilizes open-source project Apache Hadoop and its its computational capabilities in order to easily scale from several servers to thousands machines and distribute jobs between clusters. As it possible to see from the Figure 2.9 on a small number of training examples Mahout scalable algorithm (number 4) performs worse than non-scalable one (number 3); however when the number of training samples growth Mahout solution becomes more efficient.

Apache Hadoop is an open-source software for scalable, reliable distributed computing tasks. It uses Hadoop Distributed File System (HDFS) and includes the following modules: Pig (high-level data flow language for parallel computation), Hive (DWH infrastructure), HBase (scalable distributed database), Yarn (framework for scheduling and cluster resource management) [3].

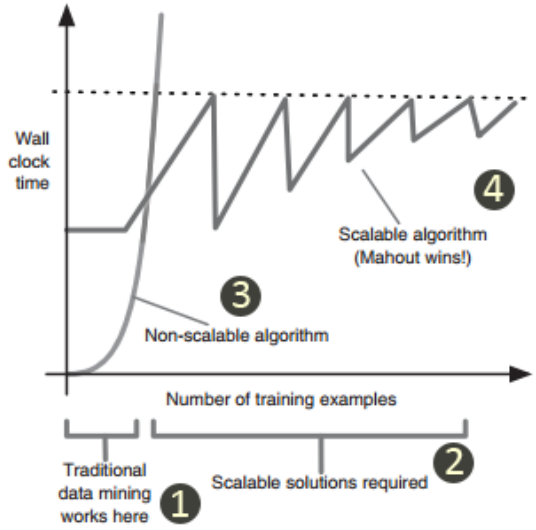


Figure 2.9: Comparison of time usage by scalable and non-scalable solutions [50]

Apache Mahout deals with big data sets effectively by using Hadoop that includes implementation of the MapReduce distributed computing framework. This paradigm defines the input data like a set of key-value pairs. A *map* job splits the input dataset from HDFS into independent chunks that are processed in parallel (Figure 2.10). A *reduce* job will merge all obtained values to produce the output [50]. Hadoop manages store of the data in HDFS, partitioning and data transfer between machines. HDFS allows Hadoop clusters to process data right where they live, without moving them from central location to nodes. It contains such key features as high availability, scale architecture, fault tolerance and load balancing [38].

Mahout is more useful with extremely large data sets (over 1 million of training examples). If data has about 100,000 examples, other solutions can be more efficient. The reason is that

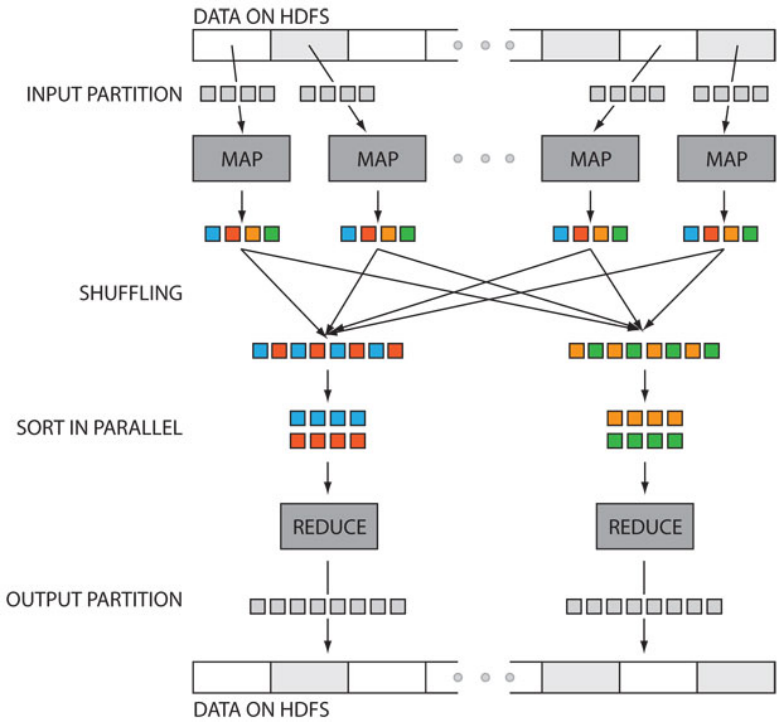


Figure 2.10: MapReduce paradigm [12]

scalable and parallel algorithms can be better with regard to time factor. The jagged line define the case when Hadoop use new clusters to increase performance and scale jobs. Application without possibility of job distribution can not increase time or memory linearly [50].

Algorithms. At the moment Mahout provides an implementation for such algorithms as recommender engine (collaborative filtering), clustering and classification. Library aims to be the tool for choice to process big data.

Three main use cases of Apache Mahout are [50]:

- *Classification*: it is a technique that defines whether the new item belongs or not to a certain category, matches patterns observed during training of the model. For example, popular musical application Shazam can decide to which genre song belongs and then faster find its name and author.

- *Clustering*: it is a technique that attempts to group large number of items that share similar features. Number of clusters can be unknown or predefined. It is a way of discovering hidden hierarchy and relation. For example, Google News use clustering algorithm for grouping its news by the topic.

- *Recommender engine* (collaborative filtering): it is a technique to discover relationships between user and items on the base of the user behaviour. Engine tries to find goods user might like (for example, music and books). Same recommender systems are used widely

by such famous companies like Amazon, Facebook, Netflix and LinkedIn. For example, Amazon recommends items to be interested based on purchase and site activity of the user.

Job design. Mahout does not provide any GUI interface and instead of that user has to prepare the data and submit jobs to it. After a job is submitted, Mahout divides it into small computational chunks using API of Apache Hadoop and further reduces them into the final result returned to end user. The job submission can be done either using Mahout commands or directly from Java code.

Listing 2.3 instantiates a classification task using Random Forest algorithm Listing 2.3, that includes several to copy data in HDFS (lines $1-2$), build a model from the train file (lines $3-7$) and finally test the model (lines $9-11$). Once the model has tested the data set Mahout calculates statistics and deliver summary that includes number of correctly classified instances, confusion matrix and statistics metrics such as Kappa, Accuracy and Reliability (see Figure 2.11).

Listing 2.3: Test example of Random Forest classificatino algorithm in Mahout

```
1  # copy data in HDFS
2  hdfs dfs -copyFromLocal /file-name /destination-path;
3  # build a model from a train file
4  hadoop jar /path/to/mahout-jar-file
5  org.apache.mahout.classifier.df.mapreduce.BuildForest
6   -Dmapred.max.split.size=/split-size
7   -d /train-file -ds /model-file.info /classifier-parameters
8  # test classification model
9  hadoop jar /path/to/mahout-jar-file org.apache.mahout.classifier.
10 df.mapreduce.TestForest -i /train-file -ds /model-file.info -m
11  /output-destination-path /classifier-parameters /output-result-file
```

2.2 Related Work

This section gives an overview over relevant existed works in field of big data analysis. It will show examples of the current big data tools and discuss their possibilities and limitations. It will clarify how it is possible to evaluate the tools for big data analysis and what are the strengths of benchmark generation.

Enterprise and Open Source Analytical Tools

Nowadays there are a lot of different options and applications the user can select for the big data analytic issues. Some of them were developed recently and spread very fast. Authors in [9] provide a comparative study of enterprise (e.g., Pentaho [16], TerraEchos [17], Google BigQuery [18]) and open source tools (e.g., HPCC [19] System). They compared them on the base of the

[16] http://www.pentaho.com/

[17] http://www-01.ibm.com/software/data/infosphere/streams/partners/terraechos.html

[18] https://developers.google.com/bigquery/

[19] http://hpccsystems.com/

```
Summary
-------------------------------------------------------
Correctly Classified Instances          :      287329       65.0244%
Incorrectly Classified Instances        :      148017       34.9756%
Total Classified Instances              :      435346

=======================================================
Confusion Matrix
-------------------------------------------------------
a         b             <--Classified as
184242    50770        |  235012       a     = f
40100     160234       |  200334       b     = t

=======================================================
Statistics
-------------------------------------------------------
Kappa                                       0.1136
Accuracy                                   68.8013%
Reliability                                36.2841%
Reliability (standard deviation)            0.5227
```

Figure 2.11: Example of result output of classification model using Random Forest algorithm in Mahout

following features such as amount of data to proceed, ease of use, energy and time consuming characteristics, extensibility, and maintenance.

The first enterprise applications the authors have investigate is the Pentaho, tool for high volume visualization and analysis that supports entire processing life-cycle of the data. This application support Predictive Model Markup Language (PMML) and is useful for time series forecasting an powerful visualization of predictive models. The only drawback the user can meet is a price, but according to reviews of authors it is compared with other commercial Business Intelligence (BI) tools.

The next commercial application was TerraEchos product from IBM, advanced security solution for big-data statistics, that main features is to handle large volumes of structured and unstructured information on the fly. In parallel the application can evaluate and represent the data. TerraEchos can examine streaming data without storing it, that can be useful for tasks where the result should be immediate. Another tools that is more popular as company product than application for individual research projects is a Cognos Business Analytics's software. Authors analysed as well the Google BigQuery, an application that enables problem solving by queering enormous amount of information (e.g., billions of rows in couple of seconds). One of its advantages is that it works using highly reliable Secure Sockets Layer (SSL) access method. Google BigQuery will be the right choice for solving tasks that are related with data storing and transformations of large amounts of information. Another observed by authors application was Netezza [20], a warehouse engine for run predictive analysis and business intelligence tasks that includes database, server and storage components. Its computational kernels utilize Asymmetric Massively Parallel Processing (AMPP) approach, that allows Netezza to hold exponential growth of data. One of open source tools for big data analysis considered by authors is a High

[20] http://www-01.ibm.com/software/data/netezza/

Performance Computing Cluster (HPCC), developed by LexisNexis Risk Solutions. It is highly scalable parallel processing application that contains highly integrated system environment that includes different methods from raw data processing to queries and data analysis algorithms. It also has a good tolerance for any faults and system reprocessing capabilities in case of various system failures. Another open source application observed by authors is Dremel [21], the ad-hoc query system developed by Google, that is scalable and is often used for analysis of web documents. The next open source big data application is In-Memory Computing Platform GridGrain [22], that was developed in Java and is compatible with HDFS. It is a high-performance, integrated, distributed memory systems to compute and transact on large-scale data sets in real-time.

NASSCOM team [42] presented am overview of trends in databases, software architecture and development of new techniques to facilitate Big Data implementations.

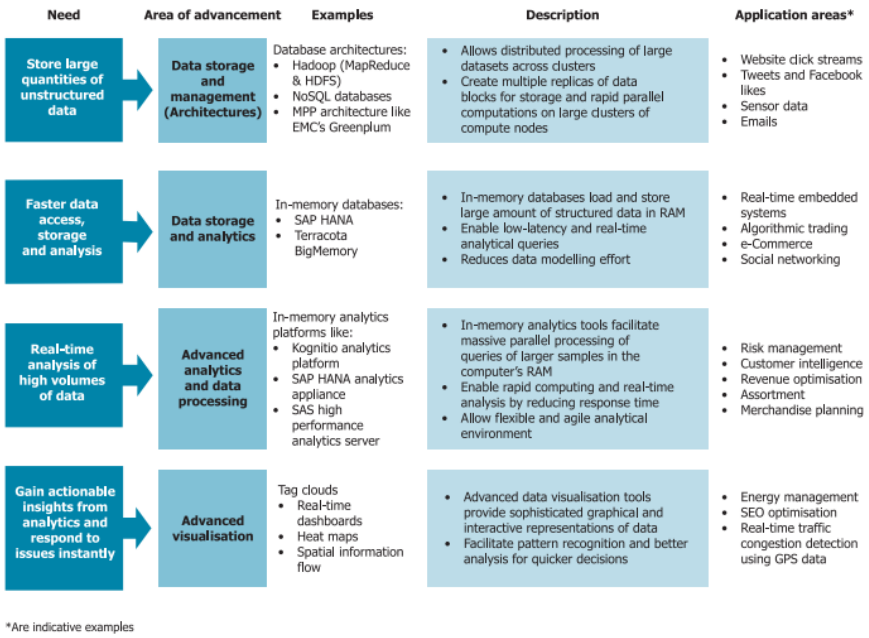


Figure 2.12: Example of results of classification model in Mahout [42]

Big data volumes grows rapidly and software tools should be developed in order to handle new requests. A lot of new organisations emerge as providers of Apache open source Hadoop distributions such as Cloudera and Hortonworks. Non-Hadoop popular vendors are Splunk [23] and Datastax [24]. The positive impact in industry possess cloud computing techniques that try to combine best practices of visualization, grid computing, and database advanced analytic. One of the negative impacts is a talent and knowledge shortage in the area of Big Data. The main needs of Big Data applications covers the following sections [42]: store large unstructured data

[21] http://hpccsystems.com/
[22] http://www.gridgain.com/
[23] http://www.splunk.com/
[24] http://www.datastax.com/

sets using MapReduce paradigm and parallel processing techniques (e.g., Hadoop and NoSQL databases); faster data access and analysis that allows to perform real-time analytical queries and reduce time on model building (e.g., Hadoop, SAP HANA); analyse high volume data in real-time using flexible analytical environment (e.g., Kognitio [25] platform), and provide actionable insight with advanced visualization, sophisticated data representation and facilities of pattern recognition (see Figure 2.12). All applications mentioned above very in privacy and data security capabilities, easy of use, and some of them require additional knowledges.

Ames et al. [1] benchmarked two open-source applications R and Apache Mahout and two enterprise tools SAS Rapid Predictive Modeller [26] for SAS Enterprise Miner [27] and SAS High-Performance Analytics Server. Authors applied in their investigation such classification algorithms as Logistic Regression, Decision trees, and Random Forest. They solved four classification problems and evaluated models on the base of such criteria as model quality, overall completeness, scalability, and overall effort to develop model.

Each data set represented a separate industry sector (e.g., marketing, financial) with various number of observations. There were identified certain software limitation: method Random Forest was not implemented in SAS Rapid Modeler or Decision tree algorithms were available only for R and SAS High-Performance Analytics Server. Data was split into train and test set using percentage split on 60% and 40%, stratified on the target. Authors mentioned that although the Rapid Predictive Modeller simulate data in a specific way finally obtained numeric results can be compared to results of other three applications. The experiment was performed in same operational environment for R and Mahout, however SAS High-Performance Analysis Server was configured on significantly larger massive parallel processing appliance. Therefore scalability measurement can not be compared (see Figure 2.13).

Software	Type	Appliance	Available RAM
SAS High-Performance Analytics Server 12.1 (using Hadoop); SAS Enterprise Miner 12.1 client	SAS product	Greenplum; two master hosts with 60 nodes	96 GB each
Rapid Predictive Modeler for SAS Enterprise Miner and SAS Enterprise Miner 12.1, SAS 9.3	SAS product	Microsoft Windows 2008 R2 server	21 GB
R 2.15.1 "Roasted Marshmallows" version (64-bit)	Open source	Microsoft Windows 2008 R2 server	15 GB
Mahout 0.7	Open source	Five-machine Hadoop cluster	48GB each

Figure 2.13: Operating environment characteristics [1]

Figure 2.14 illustrates measured effort required to do statistical preparations per each applications. Authors compared also the quality of each classified model from various applications by measuring percentage of correctly classified events, event precision.

[25] http://kognitio.com/

[26] http://www.sas.com/en_us/software/analytics/rapid-predictive-modeler.html

[27] http://www.sas.com/en_us/software/analytics/enterprise-miner.html

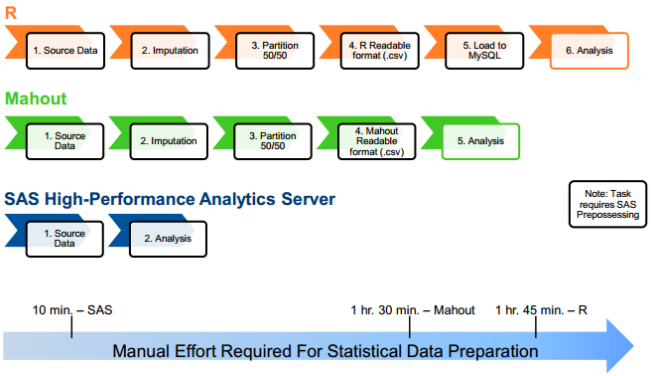


Figure 2.14: Effort required to do statistical preparations per software application [1]

The following results were achieved: SAS Rapid Predictive Modeler produced better model and SAS High-Performance Analytics Server include more methods for computation that are not available in Mahout. It was observed that R is less scalable and with a certain number of calculations can lead to memory shortage. Model effort required to prepare data for actual modelling in two SAS products was significantly smaller.

Data Analysis and Benchmarking

Benchmarking is a widely used technique to compare applications. Han et al. [21] analysed key requirements and challenges (e.g., performance bottleneck, guarantee of the data veracity) for proper benchmark development. They have described a process of data generation with the 4V properties (volume, variety, velocity, and veracity) and compared data generation techniques for existing data benchmarks (e.g., BigBench [28], CloudSuite [29]). *Veracity* is one of the Big Data properties and important characteristic of raw data. This parameter is important during benchmark data generation from synthetic data to guarantee the reality and similarity of data to the raw analogue. Big data benchmark generate *application-specific* workloads and tests that can be further used to produce meaningful evaluation result, compare performance and architecture of applications. Benchmarks can apply the real-world data or generate synthetic data, that can be a key issue during benchmark creation. The obstacle of using real-data benchmarks is a shortage of data variety, because data owners are not willing to share the full information. Therefore the synthetic data generation is the consensus. During benchmark application generation the following challenges can arise: adapting of different type formats, portability to represent software stacks (benchmark applications should be easy portable to test a new software and cover broad spectrum of representative software stacks), extensibility and usability. Big data benchmark includes the following main layers [21]:

- *interface layer*: includes interface to assist user to specify benchmark requirements

[28] https://github.com/intel-hadoop/Big-Bench
[29] http://parsa.epfl.ch/cloudsuite/cloudsuite.html

- *function layer*: includes three main components: data generators to produce data keeping 4V properties, test generators to generate tests with comprehensive workload and metrics to measure final results

- *execution layer*: includes functionality to support benchmark test execution such as data format conversion, result analyzer

Authors compared data generation techniques in existing big data benchmarks using the following characteristics: volume (e.g., scalable, partially scalable), veracity (e.g., unconsidered, partially considered, considered), velocity (e.g., uncontrollable, semi-controllable), variety (e.g., texts, graphs, videos), and type of workloads (e.g., on-line services, off-line analytic, real-time analytic). Large variety of benchmarking techniques is developed to test the performance on Hadoop MapReduce and Database Management System (DBMS) (e.g., PigMix [30], GridMix [31] benchmarks). Some of them are used to evaluate architecture and performance of NoSQL databases (e.g., Yahoo! Cloud Serving Benchmark (YCSB)). Han et al. listed challenges to develop successfully big data benchmarks. They include controllable data velocity (benchmark generators control data generation rates and not the frequency of the data update), metrics to evaluate data veracity (an open question is to measure the conformity of synthetically generated data to the row analogue), and enriching workload (the first key issue is that truly hybrid workload of various data operations is not supported adequately and the second one is benchmarking of multimedia systems and deep-learning platforms). Other challenges such as data redundancy in benchmarks, limitations of representativeness of samples and diversity were proposed in [15]. Authors came up with finding that benchmarking techniques use more artificial data than real world values. Therefore it is not always clear to understand whether such data sets can be used to evaluate big data applications. In article authors analysed both problem of information sharing and the impossibility to download GB of traffic data every time when calculations should be performed.

[30] https://cwiki.apache.org/confluence/display/PIG/PigMix
[31] http://hadoop.apache.org/docs/r1.0.4/gridmix.html

CHAPTER 3

Benchmark Description

In this chapter we will shortly overview the applications that generate benchmarks and describe the main features and attributes of the benchmarks chosen for data analysis. We will focus on the number of important aspects of each benchmark, like the number of fields in the files, data format, and time periods.

3.1 Benchmarking

Today volume of the data increases very fast due to high popularity of different applications and social networks. Big Data is about applying latest innovative techniques to solve existing or future problems, where resources exceed the capabilities of traditional computing frameworks [36]. Data of different types (e.g., structured, unstructured) and volumes provides new opportunities for analytics and business intelligence use cases.

In order to solve numerous problems, open source and commercial companies represent benchmark datasets and tools for benchmark generation in order to rank systems that running Big Data on the base of an etalon. Such benchmark suites includes representative workloads and diversity of data characteristics, enabling to test and compare an applications on various data load, check algorithm efficiency (e.g., sort, k-means, pagerank) and analyse their stability in work with different data types.

Figure 3.1 represents benchmarking process that includes five steps: step 1 is an investigation of application domain, metrics and benchmarking objects. Next step is data and test generation or upload. Next step is execution on the big data analysis tool and final step is the evaluation using metrics [21].

One example of such tool is a BigDataBench benchmark suite that provides 6 real-world data sets (e.g., Wikipedia entries with more than four millions of articles, Amazon movie reviews with 7 millions of instances), two synthetic data sets and 32 big data workloads. Benchmark propose Big Data Generator Suite (BDGS) that overcomes difficulties of obtaining Big Data and can

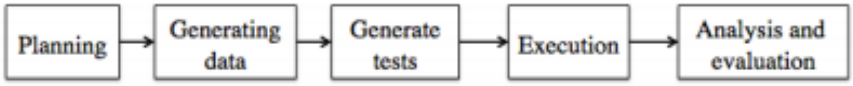


Figure 3.1: Benchmarking process for big data analysis tools [21]

prepare 10 TB of Wiki data in five hours [7]. Another example is a HiBench [1], representative and comprehensive benchmark suite for Hadoop, that contains synthetic and real-world applications and around ten workloads [23]. Other benchmarks are ICTBench [2], Big Data Benchmark [3]. Last one is used to measure response time on a relational queries like joins, aggregations across different data volumes.

Testing and comparison of applications is a process that requires several prerequisites. Firstly, all obtained results should be reproducible with a certain level of confidence. Secondly, the testing use case has to be applicable that supposes modelling of tool behaviour in the real conditions (in our case it is a store sales data for prediction). Therefore, as there is an infinite set of possible parameters and characteristics that can be compared to different types of data, only realistic and representative set of them will release weak and strong aspects of each application. There are four general experimental methodologies of evaluation: in-situ, emulation, benchmarking, and simulation [26].

Emulation and simulation intend usage of model environments that is completely non-applicable in our case. In-situ methodology uses real environment and real data that can cause a reduction of reproducibility level. Taking into consideration the facts described above, we will use benchmarking as the main methodology for our further research. Benchmarking is an experimental methodology that consists in executing application model (benchmark) on a real environment [26]. In our case, a benchmark is a set of certainly structured data with the help of that a researcher tries to assess the relative performance of an application by running a number of standard operations against it.

The datasets that are chosen for analysis of big data applications in this research were uploaded from Kaggle [27], one of the largest communities for data scientists that provide competitions and datasets for them. Selected benchmarks were prepared for a specific challenges in the area of commerce such as store sales forecasting, repeat buyer detection and analysis of tendencies in groups of people who buy car policies. Each of these data sets vary on the nature of the workload and attribute characteristics. Benchmarks are time-series data sets that has a time-oriented chronological sequence of observed values.

We analysed data sets was analysed with the help of Ataccama DQ Analyzer [4] that enables to get data statistics about completeness and quality of the data. For each file was created a profile with a list of types, domain, unique and distinct values per column. Data for benchmarking should be cleaned, complete, accurate (exclusion of misspelling, different types of rows per column) and have same patterns in order to provide applications comparison.

[1] https://github.com/intel-hadoop/HiBench
[2] http://prof.ict.ac.cn/ICTBench/
[3] https://amplab.cs.berkeley.edu/benchmark/

3.2 Benchmark Acquire Buyers

The first dataset is Acquire Buyers benchmark that contains complete, basket-level, pre-offer shopping history data values for a list of shoppers. Different consumer brands provide incentive offers in a form of discounts or coupons to attract new shoppers to buy products. Valued shopper are defined like those who come back after the incented purchase to buy the same item again. The post-incentive behaviour of the shopper and transaction history with all items purchased and those not related to the offer is provided [28]. Only one offer is available per customer. Data set contains 18 attributes that relates to offered product (such as market, brand, product size), 37 offers issued for customer and in summary 349 millions of rows for use case analysis. The associated task that will be performed with this data set is a classification. It will be necessary to predict the repeat buyer among all customers on the base of his/her shopping behaviour. This data set is largest among those used for this research and its decompressed files require about 22 GB of space.

Data Details

Collection contains three files in CSV format: transactions, history, and offers.

History set contains offers for each customer issued before 2013-05-01 and behavioural response to this offer. Set contains all items purchased, not only those related to the offer. The number of instances is 160057 records. These file has seven following columns (see Table 3.1):

- *id*: unique identifier that represents a customer
- *chain*: store chain identifier
- *offer*: certain offer identifier
- *market*: geographical region identifier
- *repeattrips*: number of times when the customer made a repeat purchase
- *repeater*: boolean value defines if customer is a repeat buyer. Example values: true, false.
- *offerdate*: date a customer received his offer

The history file during classification model evaluation will be splitted on the train and test sets without intersection of time periods and the value that defines a class correctness is stored in column *repeater*. Table represents the profile information. Data set contains 160057 customers, 130 chains, 24 offers and 34 markets.

The file 'transactions.csv' contains transaction history of all customers data for a period of one year (from 2012-03 till 2013-07). Structure consists from the following 11 columns:

- *id*: unique identifier that represents a customer
- *chain*: store chain identifier
- *dept*: aggregate grouping identifier of the category (e.g., water)

Expression	Type	Domain	Non-null	Unique	Distinct
id	STRING	long	160057	160057	160057
chain	STRING	integer	160057	6	130
offer	STRING	integer	160057	0	24
market	STRING	integer	160057	0	34
repeattrips	STRING	integer	160057	16	56
repeater	STRING	enum	160057	0	2
offerdate	STRING	day pattern	160057	0	56

Table 3.1: Use case 1: file 'history.csv' overview

- *category*: product category identifier (e.g., sparkling water)
- *company*: identifier of the company that sells items (e.g., Billa)
- *brand*: identifier of the brand to which the item belongs (e.g., Nestle)
- *date*: date of purchase
- *productsize*: size of the product purchase (e.g., 16 oz of water)
- *productmeasure*: units of the product purchase (e.g., ounces)
- *purchasequantity*: number of purchased units
- *purchaseamount*: amount of the purchase in dollars

Table 3.2 represents the profile information. File contains around 340 millions of transactions for 311541 customers. Not all customers got offers, therefore the history file contains only around 160000 instances. Let's say there are 1000 of customers and 600 of them are offered to purchase five bottles of water and only 300 redeem the offer. These 300 buyers are the focus of the classification and data analysis for prediction. Offered product that the customer buys is defined via a triple: brand, category and company. Negative values in fields 'productquantity' and 'purchaseamount' defines the return of the product. The file 'offers.csv' contains information about offers and includes the following 6 columns:

- *offer*: certain offer identifier
- *category*: product category identifier (e.g., sparkling water)
- *quantity*: number of units customer must purchase to get the discount
- *company*: identifier of the company that sells items (e.g., Billa)
- *offervalue*: the value of offer in dollars
- *brand*: identifier of the brand to which the item belongs (e.g., Nestle)

Table 3.3 represents the profile information. File contains 37 offers. It means that one offer can be sent to several customers.

Expression	Type	Domain	Non-null	Unique	Distinct
id	STRING	long	349655789	17	311541
chain	STRING	integer	349655789	0	134
dept	STRING	integer	349655789	0	83
category	STRING	integer	349655789	1	836
company	STRING	long	3496557897	2541	32773
brand	STRING	integer	349655789	3376	35689
date	STRING	day pattern	349655789	0	514
productsize	STRING	float pattern	349655789	109	2726
productmeasure	STRING	enum	349655789	0	11
purchasequantity	STRING	integer pattern	349655789	774	2450
purchaseamount	STRING	float pattern	349655789	25592	66452

Table 3.2: Use case 1: file 'transactions.csv' overview

Expression	Type	Domain	Non-null	Unique	Distinct
offer	STRING	integer	37	37	37
category	STRING	integer enum	37	9	20
quantity	STRING	integer enum	37	1	2
company	STRING	integer enum	37	10	18
offervalue	STRING	float enum pattern	37	2	7
brand	STRING	integer enum	37	10	19

Table 3.3: Use case 1: file 'offers.csv' overview

Data Summary

Data set includes real data, therefore all the fields in data set are anonymized to protect customer and sales information. File does not contain missing values. Mostly all attributes of the dataset (e.g., `chain`, `dept`, `category`) have discrete data type, because values built a countably infinite set. Several values such as `productsize`, `offervalue` have continuous data type. In data set such measurement scales are presented as nominal (e.g., `repeater`) data and quantitative (e.g., `repeattrips`, `offer`) data.

3.3 Benchmark Walmart Store Sales

The second dataset is a Walmart Recruiting [30]. Walmart is an American retail corporation that contains chains of warehouse stores and discount department stores in 27 countries. Data set contains historical sales data for 45 Walmart stores from different regions. Each store contains several departments. Information about department sales covers three years: 2010-02 to 2012-10 and contains information about sales during four largest holidays: the Super Bowl, Labor Day, Thanksgiving, and Christmas. The weeks including these holidays are weighted five times higher in the evaluation than non-holiday weeks. Holiday days are: Super Bowl (12-Feb-

10, 11-Feb-11, 10-Feb-12, 8-Feb-13), Labor Day (10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13), Thanksgiving (26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13), and Christmas (31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13). Data set contains 20 attributes that relates to stores sales (such as markdowns, holiday weeks, consumer price index). Size of the dataset is approximately 18 MB. The associated task that will be performed with this data set is the forecast of store sales for departments.

Data Format and Files

Collection contains three files in CSV format: features, stores, history. File 'stores.csv' includes information about 45 Walmart stores in different locations and has three columns:

- *store*: store identifier
- *type*: store type identifier
- *size*: store size

Table 3.4 represents the profile information. Each store has type: A, B or C and different size. Stores with type A occurred more frequently (in 48% of dataset) and with type C rarely (only 13%).

Expression	Type	Domain	Non-null	Unique	Distinct
store	STRING	integer	45	45	45
type	STRING	enum	45	0	3
size	STRING	integer	45	37	40

Table 3.4: Use case 2: file 'store.csv' overview

File 'features.csv' contains additional information about stores, departments and regional activity in the period of four years from 2010-02 till 2013-07. File contains 8 columns:

- *store*: store identifier
- *date*: week number
- *temperature*: average temperature in the region
- *fuel_price*: cost of the fuel in the region
- *markDown1-5*: data related to promotional markdowns in Walmart store. MarkDown data is available only after November 2011. Infromtion is anonymized. It is not available for all stores all the time.
- *cpi*: consumer price index
- *unemployment*: rate of unemployment in the region

- *isHoliday*: value indicates if the week is a special holiday week (true, false values)

Table 3.5 represents the profile information. File contains approx. 8000 rows. Fuel price vary from 2.4 to 4.4 in different regions,unemployment rate from 9.9 to 10.1.

Expression	Type	Domain	Non-null	Unique	Distinct
store	STRING	integer	8190	0	45
date	STRING	day pattern	8190	0	182
temperature	STRING	float pattern	8190	2009	4178
fuel_price	STRING	float pattern	8190	66	1011
markDown1	STRING	specval pattern	8190	4014	4024
markDown2	STRING	specval pattern	8190	2615	2716
markDown3	STRING	specval pattern	8190	2482	2886
markDown4	STRING	specval pattern	8190	3363	3406
markDown5	STRING	specval pattern	8190	4040	4046
cpi	STRING	float pattern	8190	996	2506
unemployment	STRING	float pattern	8190	0	405
isHoliday	STRING	enum	8190	0	2

Table 3.5: Use case 2: file 'features.csv' overview

Data set contains historical data for the period from 2010-02 to 2012-11 and weekly sales for each store. Files contains the following columns:

- *store*: store identifier
- *dept*: department identifier
- *date*: week number
- *weekly_sales*: number of sales amount for the given department in the given store
- *isHoliday*: value indicates if the week is a special holiday week (true, false values)

Table 3.6 represents the profile information. Each store contains around 80 departments.

Expression	Type	Domain	Non-null	Unique	Distinct
store	STRING	integer	421570	0	45
dept	STRING	integer	421570	0	81
date	STRING	day pattern	421570	0	143
weekly_sales	STRING	float pattern	421570	329075	359464
isHoliday	STRING	enum	421570	0	2

Table 3.6: Use case 2: file 'features.csv' overview

Data Summary

In summary for data preprocessing data set contains around 400,000 rows. Data sets includes real data and all fields are anonymized with numeric values to protect Walmart sales information. File contains missing values for such attributes as 'markdowns', 'cpi', 'unemployemnt rate'. Data set contains discrete attributes type (e.g., `store`, `size`) and continuous `cpi`, `unemployment`. In data set such measurement scales are presented as nominal (e.g., `type`, `isHoliday`) data and quantitative (e.g., `fuel_price`, `temperature`) data.

3.4 Benchmark Allstate Policy

The third benchmark dataset is Allstate Policy [29]. Allstate is a one of the largest personal lines insurer in USA. Data file contains transaction and quote history of customers, options of the car policy they have purchased. Each customer receive a number fo quotes with various coverage options. Dataset contains many plans for customers, that includes customer id, viewing characteristics of the policy and costs. Each shopping point includes information about product options. Customers are formed in groups and policies can cover more than one person. Customer can purchase a product that was not viewed by him before. Customer characteristics can change over time, if the customer got new information concerning car policy. Cost can change depending on customer characteristics as well. If the policy feats customer expectations the quoting process can be shorter and the company less likely will loose a potential buyer. Data set contains 25 attributes. The associated task for this data set is to define groups of customers on the base of corresponding information about them (such as marriage status, car value, age) by dividing them on clusters and analyse which set of option is more attractive for each group. Size of the dataset is approx. 47 MB.

Data Format and Files

Collection contains one file in CSV format. Each product contains 7 different customizable options that customer selects. Each options has 2,3, or 4 ordinal values. It can be assumed that numbers define level of insurance for certain option:

- A: 0,1,2.
- B: 0,1.
- C: 1,2,3,4.
- D: 1,2,3.
- E: 0,1.
- F: 0,1,2,3.
- G: 1,2,3,4.

Data contains the following columns:

- *customer_id*: unique identifier for the customer
- *shopping_pt*: unique identifier for the shopping point of a given customer
- *record_type*: boolean vaue if the point is a shopping point (0) or a purchase point (1).
- *day*: day of the week (0-6, 0=Monday)
- *time*: time of day (HH:MM)
- *state*: state number where shopping point occurred
- *location*: location identifier where shopping point occurred
- *group_size*: how many people will be covered under the policy (1, 2, 3 or 4)
- *homeowner*: boolean value defines whether the customer owns a home or not (0=no, 1=yes)
- *car_age*: age of the customer's car
- *car_value*: relative value of the customer's car when it was new
- *risk_factor*: an ordinal assessment of how risky the customer is (1, 2, 3, 4)
- *age_oldest*: age of the oldest person in customer's group
- *age_youngest*: age of the youngest person in customer's group
- *married_couple*: boolean value, define if the customer group contain a married couple (0=no, 1=yes)
- *c_previous*: boolean value, define if the customer had or has for product option C (0-not, 1,2,3,4)
- *duration_previous*: time in years that define how long customer was covered by their previous issuer
- *A,B,C,D,E,F,G*: coverage option
- *cost*: cost of policy with the set of quoted coverage options

Dataset contains 18 columns, 665249 rows and represents historical information for a period of 26 weeks for 97009 customers. Table 3.7 represents the profile information. Customer age varies from 16 to 75, risk factor from 1 to 4, car value from a to i, where a is the most valuable car.

Expression	Type	Domain	Non-null	Unique	Distinct
customer_id	STRING	integer	665249	0	97009
shopping_pt	STRING	integer enum	665249	0	13
record_type	STRING	integer enum	665249	0	2
day	STRING	integer enum	665249	0	7
time	STRING	pattern	665249	99	1204
state	STRING		665249	0	36
location	STRING	integer	665249	10	6248
group_size	STRING	integer enum	665249	0	4
homeonwer	STRING	integer enum	665249	0	2
car_age	STRING	integer	665249	1	67
car_value	STRING	enum	665249	0	9
risk_factor	STRING	enum	665249	0	5
age_oldest	STRING	integer	665249	0	58
age_youngest	STRING	integer	665249	0	60
married_couple	STRING	integer enum	665249	0	2
c_previous	STRING	integer enum	665249	0	5
duration_previous	STRING	integer enum	665249	0	17
A	STRING	integer enum	665249	0	3
B	STRING	integer enum	665249	0	2
C	STRING	integer enum	665249	0	4
D	STRING	integer enum	665249	0	3
E	STRING	integer enum	665249	0	2
F	STRING	integer enum	665249	0	4
G	STRING	integer enum	665249	0	4
cost	STRING	integer	665249	48	531

Table 3.7: Use case 3: dataset overview

Data Summary

Data set includes real data, therefore all fields concerning customer information and policy specifications are anonymized with numeric values to protect Allstate information. The file contains missing values for such attributes as 'risk factor', 'c_previous', 'duration previous'. Allstate data set includes only discrete (e.g., `state`, `location`, `age_oldest`) data type. In data set such measurement scales are presented as nominal (e.g., `homeowner`) data and quantitative (e.g., `cost`, `duration_previous`) data.

3.5 Benchmark Summary

The data sets described above will be used for classification, regression and clustering tasks. We highlighted the main features and attributes of each data set and describe the meaning of them.

As it described on Table 3.8 and was mentioned above, the data sets differ by volume, number of attributes and instances. Every attribute in each data set is anonymized in order to observe data privacy. File format for all sets is the same, that come up with same data manipulations to insert data into database.

Parameter	**Acquire**	**Walmart**	**Allstate**
Volume	20GB	18MB	47MB
Variety	✓	✓	✓
Velocity	✗	✗	✗
Data privacy	✓	✓	✓
Number of attributes	18	20	25
Number of instances	349 mill.	400000	665249
Missing values	✗	✓	✓
Format	CSV	CSV	CSV

Table 3.8: Comparison of the data sets

CHAPTER 4

Use Case Analysis

Today Big Data is not only the big volume, but also a diversity of types that are delivered at various speeds [47]. Adavanced analytics of such data requires usage of different application types that are based on data mining, statistics, artificial intelligence, predictive analytics and provide suitable graphical techniques [47].

The concept value of Big Data means the result that was achieved after data preparation and application of a suitable model. There is a large difference between what is being said about data and information itself and what is being done with this information [36]. The variety of use cases depend on the area of business and expectations of users. For example for marketing company that sells some goods for customers the example of use cases can be: observe purchasing trends, enable new purchasing patterns, define new customer groups by analyzing shopper's transactions and social data, optimization of customer spendings, define possible improvements in the manufacturing sector to launch a new product, analyse frauds, waste and abuse. For each use case the list of data transformations can vary.

This chapter contains the description of the use cases for each data set and its implementation process, information about used technologies and data workflow for each scenario. We will discuss three data mining methods - classification, clustering and regression and will extract meaningful data, uncover hidden relations from the data sets. Each scenario will be evaluated using the following features such as overall completeness, effort required to build the model. Further obtained results are used during model assessment in order to find the most suitable data mining algorithm.

4.1 Operating Environment

Three big data applications such as R, Weka and KNIME were executed on the same Microsoft Azure virtual machine that was running on Microsoft Windows Server 2012 x64. The virtual machine (VM) has 14 GB RAM and 2,2 GHz virtual processor (Table 4.1). Apache Mahout was utilised like one of a components on HDInsight 3.1 Hadoop cluser. This information is provided,

because time of testing and training each algorithm depend on these system properties. The exact information concerning RAM on VM where Mahout was executed is not available. Since we do not measure scalability, we assume that the virtual machine configuration is comparable among applications.

Software	**Type**	**Appliance**	**RAM**	**Virtual Cores**
Weka 3.7	Open source	Microsoft Azure, VM Extra large (A4)	14 GB	8
KNIME 2.9.2	Open source	Microsoft Azure, VM Extra large (A4)	14 GB	8
R 3.0.3, RStudio-0.98.507	Open source	Microsoft Azure, VM Extra large (A4)	14 GB	8
Mahout 0.9.0, Hadoop 2.1	Open source	Microsoft Azure HDInsight 3.1	NA	25

Table 4.1: Operating Environment information

Application	**Version**
DQ Analyzer	8.0.0.ga-teradata-2012-10-01
MySQL Workbench	6.1.6.11834
Talend	5.4.2

Table 4.2: Additional software used for data analysis

Table 4.2 shows the general set-up for each software platform that was additionally required in order to analyse data set quality and perform data transformations. Data from each benchmark should be filtered and modified in order to correspond to the given task. MySQL Database was used to perform these changes. Although each tool can be tuned to access MySQL database, required data modifications could not be done without additional programming within each system. Since the research does not assume monitoring of the database access from data mining tools, we made a decision to start the application evaluation from the point when the dataset will be prepared for input. In order to run applications with higher efficiency and exclude memory shortage, the following changes were applied to default settings:

- For Weka: *maxheap*=4G
- For KNIME: *Xmx*=2g, XX:*MaxPermSize*=512m
- For MySQL: *innodb_buffer_pool_size* = 11G; *innodb_additional_mem_pool_size* = 20M; *innodb_buffer_pool_instances* = 4; *connect_timeout* = 1000; *DBMS connection keep-alive interval* = 28800

We used Talend Data Integration [1] platform to copy data from CSV files into MySQL database. It provides an open source set of tools to access, transform and integrate data. We built ETL jobs in *Job Designer* using GUI components and added such metadata information as database connection details and file paths. The ETL job starts with the data extraction from operational data source (in our case CSV files), transforms data to fit operational needs and insert into database storage. We use both ETL jobs and SQL queries to join and transform data.

4.2 Benchmark Objectives and Methods

The objectives of the benchmark study of three data sets on application for big data analysis are the following [1]:

- *overall completeness*: define the number of modelling algorithms in software packages for certain chosen task.
- *overall effort to develop model in application*: the effort was calculated by recoding the time required to prepare manually the model in application. Only pre-existing software algorithms were used for each scenario. Theoretically, any algorithm can be programmed in R using R script, in Mahout using Java, in Knime using additional Java Nodes or in Weka using Java language as well. The goal of this research is to compare pre-installed basic application methods, therefore any additional programming was not performed in order to develop extra nodes, classes or instances using API. Additionally, latest version of KNIME and Mahout includes the implementation of new R and Weka plug-ins, that can be additionally installed. Since we are interested in basic functions of the application, the study uses only packages that are currently available in R, command-line interface in Mahout, GUI in Weka and KNIME.
- *overall effort to run the model*: the effort will be calculated by recording the time required for application to upload input data, perform all data transformations and run the model.
- *model quality*: model quality will be evaluated per use case in order to compare methods inside one application (e.g., classification model using Random Forest algorithm). Each tool can be written using specific programming language. Therefore, for the same algorithms the use case results can have different values between applications because of other variable accuracy or programming language facilities. Since the goal of this research is not to compare the implementation of methods in application, but compare the ease of model development and visualization, better model quality results will be represented for one application.

The data mining process can be roughly separated into three activities such as data preprocessing, model execution and results explaining. In this research we concentrate on each of these tasks. Additionally we describe all related algorithms for regression, classification and clustering for each use case.

[1] https://www.talend.com/

4.3 Use cases: Clustering

This section describes scenario that was created by applying several clustering algorithms in KNIME, Weka, Apache Mahout, and R. Clustering allows to build groups of data on the base of certain relations and strong or weak resemblance of items to each other [24]. The dataset that was used for these use case is an Allstate Policy dataset that contains transaction history of customers that buy a car policy.

The algorithms that was used for all applications is a simple k-Means. K-Means algorithm is one of the most popular well-established clustering algorithms, that assigns number of data instances to one of k closest cluster centres accordingly to the chosen distance metric. This method was applied for the first time in 1982 [5]. A disadvantage of using it is that end user have to know ahead or guess the k number of clusters. Clustering algorithm is simple and effective and it usually requires several iterations [24]:

- select number of clusters
- select the value of centroid for each cluster at random
- cluster objects based on a certain distance and assign objects to cluster centers according to the distance metric
- calculate the mean value ot the *centroid* and choose these centroids as new centre values for the respective clusters
- repeat process and count steps
- stop when same points are assigned to each cluster in consecutive rounds

In the implementation of R (package *cluster* [2]), Apache Mahout, KNIME, and Weka this clustering algorithm is presented and has a wide range of different parameters (such as number of clusters, seed, number of iterations, distance function). They were analysed and the most promising parameters were chosen:

- *number of clusters*: 3,4,5
- *distance function*: Euclidean Distance
- *maximal number of iterations*: 500

The dataset was not split into train or test set. The model cluster the same data set the clusterer is trained on.

[2] http://cran.r-project.org/web/packages/cluster/cluster.pdf

Data workflow preparation

Current use case and further two use cases include various data mining steps that consist of applying data analysis techniques and algorithms to produce meaningful result. The data to be mined should be firstly extracted from the initial CSV files into a MySQL database. For each big data analysis tool such as KNIME, Weka, R, and Apache Mahout for efficient working process the initial data should be well-prepared. In this paper we build a database in MySQL, join input files using SQL queries, pre-process data both in database and then in each tool. We clean and process data using various strategies to handle with missing data and remove unrelated variables.

The preparation of the workflow started from its upload in MySQL database with the help of ETL jobs from Talend. Data set contains information about all policy plans the customer viewed and the purchase he finally did. The main question that is relevant for us: 'What kind of people groups are interested in what range of car policy characteristics?'. Since the initial data set contains all historical information about both viewed and purchased policies, the final data set we will use for clustering to define the groups of customers should be narrowed only to the final user decisions that are represented with purchased policies. Therefore we filter data set and utilize values where `record_type` value is 1. This data send tend to cluster around certain age, risk factor, marriage status customer characteristics and car policy cost, allowing us to determine hidden patterns in data. Table 4.3 represents number of observations and attributes of the initial and processed data sets.

Application	**Version**
Number of observations	665249
Number of observations used in model	97010
Number of attributes	25
Number of attributes used in model	17

Table 4.3: Use case 1: Allstate Policy Buyers dataset description

Weka implementation

For the defined use case we use both Weka Explorer and KnowledgeFlow. The data flow includes such main steps: import CSV file, convert data into ARFF format, apply transformations (remove, normalize values, and convert to nominal) and execute model.

Data Import. Weka works with special ARFF format, therefore before upload, Weka automatically converts CSV file in the ARFF format.

Attribute selection. One of the steps during data preparation in application is a feature selection in order to improve the performance of the learning model by eliminating redundant attributes. By remove of irrelevant features the number of model dimensions decreases and learning process can speed up, interpretability of the information in obtained clusters can be improved and relation between data can be more obvious. The records of the data set contains 25 attributes, from which 17 were selected for mode processing. The following attributes like `customer_id`, `shopping_pt`, `record_type`, `day`, `time`, `state`, `location`,

group_size were removed using Weka filter function *Remove* because they are not relevant for the task. Time attributes do not provide information that is useful for clustering. There are 36 different states of USA, that is too big for the number of instances in data set and number of clusters.

Nominal values. K-Means clustering algorithm in Weka accepts nominal data. Such attributes as homeowner, risk_factor, married_couple, and C_previous should be converted into nominal using Weka filter *NumericToNominal* in order to narrow variable tolerance range and reduce the number of possible values to finite series.

Values to normalize. The rest of the numeric values should be normalized with Weka filter *Normalized*. After normalization, the value interval becomes finite and value admitted region infinite. Every numeric value of attribute will be divided by the difference of minimal and maximum value. This process is used to eliminate the the effect of different scales measures in quantitative attributes. For example, for attribute age_oldest the highest value is 75 and the lowest value is 15. Using normalization equation 4.1 age value 45 will be equal to 0.5. All measurements were randomly set within the range $[0, 1]$.

$$f(x) = \frac{x - min}{max - min}, \tag{4.1}$$

where x is a value we would like to normalize, min is a minimal value of the attribute, and max is a maximal values of the attribute.

Missing values. The dataset includes missing values. Weka automatically replaces them with mean values if they are defined as *?* sign, otherwise on the step of file upload it is necessary to specify the format of missing values. To deal with missing values we replaced them with *?* sign.

After all data transformations were finished, we chose to run the whole data set as a training set. Another possibility to run the k-Means clustering in Weka is a KnowledgeFlow. All results and data model can be saved automatically with the help of work flow nodes and be easily repeatable form scratch. In contrast to KnowledgeFlow interface, in Explorer it would be necessary to start the process of clustering from the first step of the file upload.

Results. When model was created and clustering was finished Weka generates the output. Figure 4.1 illustrates the cluster output with number of clustered instances and representative attribute values per each cluster. In general after each clustering process output clusters should be interpreted. In case of Allstate Policy dataset these clusters can be interpreted like different groups of customers, who are interested in certain list of policy characteristics. Clustering algorithm split dataset in the most appropriate way into three clusters:

Cluster 0: this group represents middle age people (normalized age value varies between 0.39 and 0.41). They are single and have no house. Although they had a car policy before (that is indicated with value of the duration_previous attribute equal 3), the price for their insurance is quite high in comparison with other groups, because of higher car value (that is indicated with car_value = e) and short insurance history. Their risk factor is also high.

Cluster 1: this group represents the youngest people (normalized age value varies between 0.28 and 0.29), that have a high risk factor equals 3 and are not married. They had a car policy before but its duration time is in middle among other clusters. They can represent a group of

```
                                Cluster#
Attribute            Full Data          0          1          2
                       (97009)    (34327)    (28598)    (34084)
=============================================================
homeowner                 1          0          0          1
car_age              0.0962     0.1319     0.0743     0.0788
car_value                 e          e          f          e
risk_factor               3          3          3          1
age_oldest           0.4769     0.4162     0.2999     0.6865
age_youngest         0.4522     0.3928     0.2877     0.6501
married_couple            0          0          0          1
c_previous                3          1          3          3
duration_previous    0.4062     0.3372     0.3412     0.5301
cost                 0.6449     0.6399     0.6746     0.6252
A                         1          0          1          1
B                         0          0          1          1
C                         3          1          3          3
D                         3          3          3          3
E                         0          0          1          1
F                         2          0          2          1
G                         2          2          3          3



Time taken to build model (full training data) : 18.72 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      34327 ( 35%)
1      28598 ( 29%)
2      34084 ( 35%)
```

Figure 4.1: Use case 3: cluster output in Weka

students or young employees, who have a car, but not the best one (`car_value` is f) and have no house. The cost of the car policy for them is reasonably very high, because of young age and high risk factor.

Cluster 2: this group contains people of an old age ((normalized age value varies between 0.65 and 0.6). They have a relatively cheap new car. Such people naturally have house and family. They can represent retired or close to that people. The risk factor is low because they had a car before and are good drivers. The `cost` value of their policy is quite low, that can be because they are old clients (`duration_previous` value is 0.5) and have the longest duration of previous car policy. This is a large group of people, that compose 35% of people.

Another useful way to examine a clusters in Weka is to inspect them visually. Weka's GUI enables to easily choose values on X and Y axis and thereby change dependent variables.

Information illustrated on Figure 4.2 reflects the dependency between car age and policy cost and can be interpreted as follows: *red coloured* cluster represents people from cluster 1 : young people with new cars and highest car policy price; *green coloured* cluster represents group of people from cluster 2 : old age people with relatively new cars and very low price for car policy;

blue coloured cluster represents the last group of people from cluster 0: middle age people, their car age value varies from 0 to 0.7. Majority of them have relatively low price for the car policy. On the Figure 4.2 anomalous values outside cluster distribution are observed.

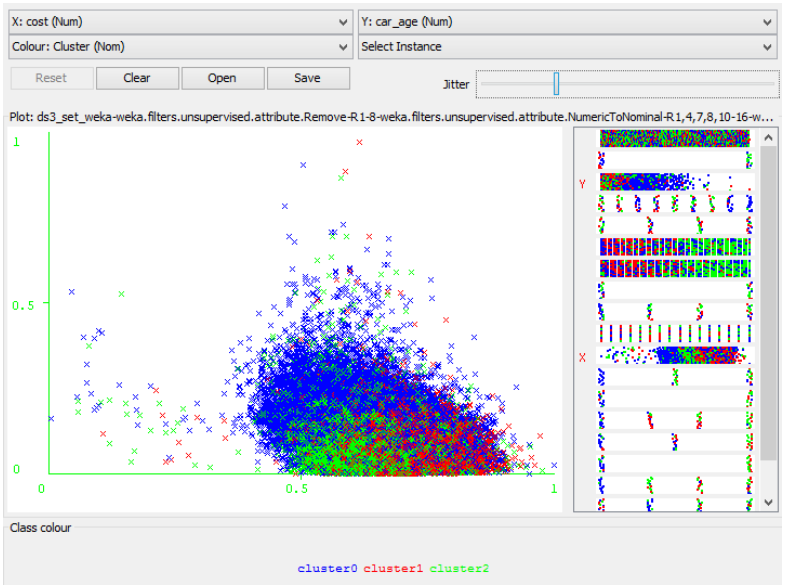


Figure 4.2: Use case 1: relation between policy price and car age value in Weka

Table 4.4 represents squared error, the widely used measure for clustering, where error value reflects the squared Euclidean distance value between instance and its cluster center [37].

No. of clusters	Number of iterations	Squared error
3 clusters	13	438689.04
4 clusters	15	456945.56
5 clusters	16	478085.28

Table 4.4: Use case 1: performance of the clustering algorithm in Weka

KNIME implementation

KNIME is a pipeline application. In order to execute k-Means cluster we build a workflow using nodes, as we discussed in Chapter 3. K-Means method in KNIME does not work with nominal values, therefore we only remove and normalize numeric values in same way as it was done in Weka. For each transformation action on data was added a node such as *ColumnFilter* and *Normalizer*.

Missing values. In KNIME it is necessary to add a special node in order to replace missing values named *MissingValue* (e.g., with mean, min, max, fix values).

Figure 4.3 shows the example of the k-Means data flow that includes six main blocks for data input (includes one node to read a data from the source), preprocessing (includes four main

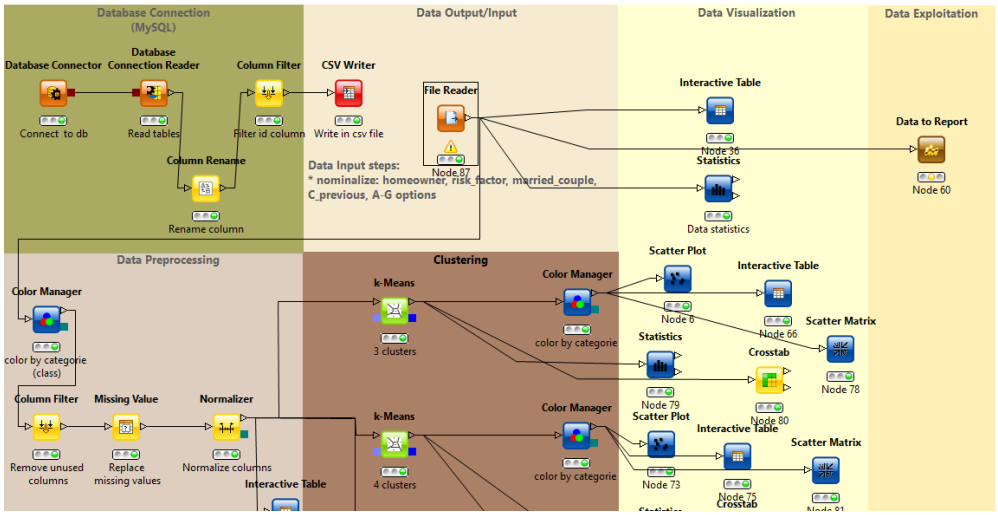


Figure 4.3: Use case 1: implementation of clustering workflow in KNIME

nodes to normalize and filter data set), clustering (includes node *k-Means* for 3, 4, and 5 clusters), visualization and reporting.

KNIME provides a powerful visual possibilities to represent clusters in grid format or chart (see Figure 4.4). If the user selects a data instance in one view, the same instance will be highlighted in another view, that is very useful when working with different types of data representation. It is possible to define colours for clusters as well.

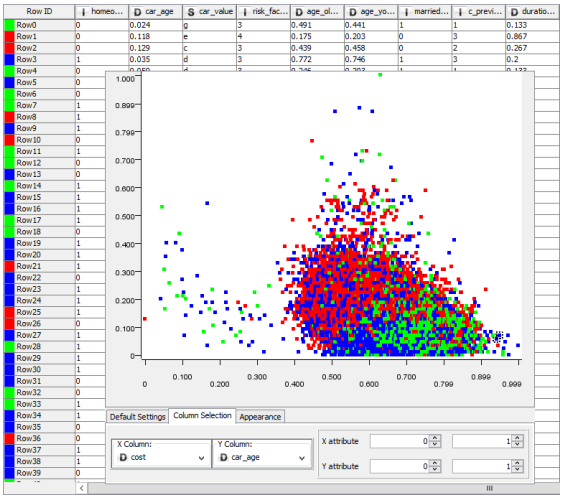


Figure 4.4: Use case 1: relation between policy price and car age value in KNIME

Apache Mahout implementation

In Mahout k-Means cluster algorithm can be implemented as a Java program or script. Algorithm is running using either the `KMeansClusterer` or the `KMeansDriver` classes. The implementation process of k-Means does not required a lot fo time. But unfortunately correct installation of Hadoop and Mahout can be time consuming because of incompatibility of Java versions. Large amount of script examples from various books, tutorials and other sources did not work with initial CSV file, therefore for Mahout implementation we used prepared ARFF file from Weka. The set of steps to perform k-Means clustering in Mahout is the following (Listing 4.1): copy files in HDFS system (lines $3-4$), run Canopy [3] clustering (lines $7-10$) and finally run k-Means clustering algorithm based on one created by Canopy (lines $11-17$). The used variables in the K-Means [4] library in Mahout are: $[--input --clusters --output --distanceMeasure --overwrite --clustering]$, where *-input -i* is a path to input directory; *-clusters -c* are input centroids, *-output -o* is the output directory; *-distanceMeasure -dm* is a class name of the distance measure between centroids, was chosen as Euclidean; *-overwrite -ow* is simply the overwriting of the output directory and *-clustering -cl* is a setting to run clustering after the iterations have taken place.

Listing 4.1: Use case 1: Mahout implementation of clustering k-Means alogithm

```
1  # create hdfs directory
2  hdfs dfs -mkdir /destination-path
3  # copy ds to hdfs
4  hdfs dfs -copyFromLocal /filename /destination-path
5  # convert the arff file into a vector
6  mahout arff.vector -d /destination-path -o /output-pathname -t c:\ds3\dict
7  # run canopy clustering
8  mahout canopy -i /path/to/job/input/directory -o /output-pathname
9  -dm org.apache.mahout.common.distance.EuclideanDistanceMeasure
10 -t1 3 -t2 2 -ow --clustering
11 # run k-Means algorithm based on created by canopy
12 mahout kmeans --input /input-directory
13 --output /output-directory --numClusters 3
14 --clusters /user/hdp/ds3/output/clusters-0-final
15 --maxIter 20 --method mapreduce
16 --distanceMeasure org.apache.mahout.common.distance.TanimotoDistanceMeasure
17 --clustering
```

Since Apache Mahout doesn't provide any GUI for visualization, it is possible to save the results in .graphml [5] file format and run with Gephi, an interactive visualization and exploration platform. Figure 4.5 illustrates three cluster that reflects three groups of people purchasing car policy. The size of clusters represents the quantity of each group. On the base of analysis

[3] https://mahout.apache.org/users/clustering/canopy-clustering.html
[4] http://mahout.apache.org/users/clustering/k-means-commandline.html
[5] http://graphml.graphdrawing.org/

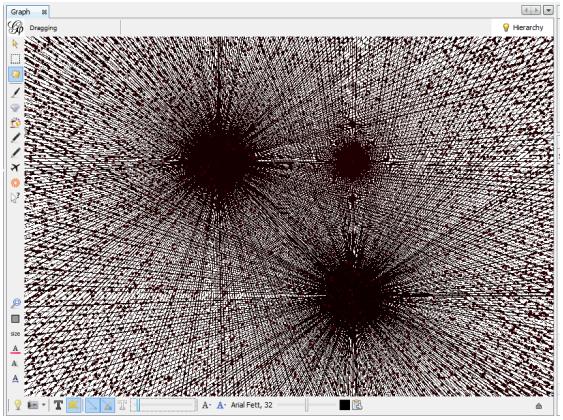


Figure 4.5: Use case 3: visualization of cluster build in Apache Mahout

and calculations from the previous tools we cn conclude that the smallest group reflects young peoples who pay the highest price for car policy, the size of this cluster is approximately 29%.

R implementation

R language implementation for the use case includes creation of a script. All values for the k-Means algorithm implemented in R should be numeric, therefore we only remove same values as it was already done in previous applications, normalize the rest of attributes and replace missing values with mean values in code (Listing 4.2). First of all it is necessary to add required libraries and packages (lines $1-4$). The next step is to import the data set and remove not relative values (lines $5-11$). Since all values should be numeric, we convert `car_value` to numeric (lines $12-13$), replace missing values (lines $14-15$) and normalize (lines $16-21$). Finally, we obtained a data set that can be used for k-Means clustering (lines $23-24$).

Listing 4.2: Use case 1: R implementation of clustering k-Means alogithm

```
1  # used libraries
2  library(ggplot2)
3  library(cluster)
4  library(fpc)
5  # read datase from file
6  ds3 <- read.csv(file="/filename",head=TRUE,sep=",")
7  ds3 <- subset(ds3, select =
8  c("homeowner","car_age","car_value",
9    "risk_factor","age_oldest","age_youngest",
10   "married_couple","c_previous","duration_previous",
11   "A","B","C","D","E","F","G","cost"))
12 # replace characters by numeric
13 ds3$car_value <- as.numeric(ds3$car_value)
14 # replace missing values ("NA") by means for one attribute
```

```
#kc3size
[1] 21152 29007 46850

#kc3centers
  homeowner car_value risk_factor married_couple c_previous        A        B        C        D        E        F
1 0.3924452  4.713597    2.754363      0.1623960   1.933492 0.248298 0.2951494 1.611573 2.156865 0.1051910 0.1628688
2 0.4808150  5.523115    2.682116      0.1810253   1.599409 1.215500 0.5456269 1.446789 2.131279 0.5300445 1.8662392
3 0.6546211  5.772914    2.400828      0.2529349   3.218084 1.092209 0.5147279 3.121729 2.916862 0.5822199 1.1871291
        G     car_age age_oldest age_youngest duration_previous      cost
1 1.941519 0.015808163 0.04989055   0.04687075       0.006108058 0.7350556
2 2.224222 0.008340020 0.05095499   0.04807417       0.006438341 0.7741970
3 2.463757 0.007889837 0.05743370   0.05441308       0.008292808 0.7546449

#kc3aggreg
  cluster homeowner car_value risk_factor married_couple c_previous        A        B        C        D        E
1       1 0.3924452  4.713597    2.754363      0.1623960   1.933492 0.248298 0.2951494 1.611573 2.156865 0.1051910
2       2 0.4808150  5.523115    2.682116      0.1810253   1.599409 1.215500 0.5456269 1.446789 2.131279 0.5300445
3       3 0.6546211  5.772914    2.400828      0.2529349   3.218084 1.092209 0.5147279 3.121729 2.916862 0.5822199
        F        G     car_age age_oldest age_youngest duration_previous      cost
1 0.1628688 1.941519 0.015808163 0.04989055   0.04687075       0.006108058 0.7350556
2 1.8662392 2.224222 0.008340020 0.05095499   0.04807417       0.006438341 0.7741970
3 1.1871291 2.463757 0.007889837 0.05743370   0.05441308       0.008292808 0.7546449
```

Figure 4.6: Use case 3: summary information about centres of clusters in R

```
15 ds3$homeowner[is.na(ds3$homeowner)] <- mean(ds3$homeowner, na.rm=T)
16 # normalize function
17 norm <- function(m){
18    (m-min(m))/(max(m)-min(m))
19 }
20 # normalize: car_age, age_oldest, age_youngest, duration_previous, cost
21 normds3 <- norm(ds3[,c("car_age", "age_oldest",
22 "age_youngest", "duration_previous", "cost")])
23 # clustering, 3 centers
24 kc3 <- kmeans(ds3, 3)
```

In order to get the information about the model (cluster centres, summary information), we have to write the additional script lines Listing 4.3.

Listing 4.3: Use case 1: script lines to get statistical information about clusters in R

```
1 # statistics about cluster
2 kc3summary <- summary(kc3)
3 kc3size <- kc3$size
4 kc3centers <- kc3$centers
5 kc3aggreg <- aggregate(ds3, by=list(cluster=kc3$cluster), mean)
```

The model output will include size and centres of clusters (see Figure 4.6).

In R there are a lot libraries for visualization and one of them is a *qplot* [6]. The following code shows how to draw k-Menas algorithm results using this library.

```
im1 <- qplot(cost, age_youngest, colour = clusters3)
```

Figure 4.7 illustrates the possibilities of the *qplot* library and reflects the dependency between age of the car owners and policy cost, that can be interpreted as follows: *red coloured* cluster represents people from cluster 1; *green coloured* cluster represents group of people from

[6] http://docs.ggplot2.org/0.9.3/qplot.html

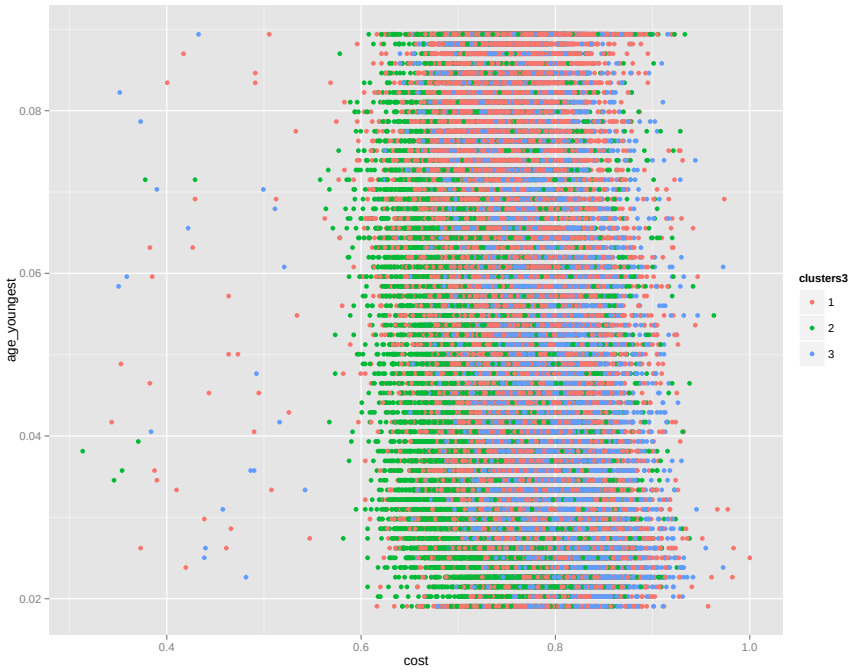


Figure 4.7: Use case 1: relation between `cost` and `car_age` values in R

cluster 2 and *blue coloured* cluster represents the last group of people from cluster 0. Majority of people from clusters 0 and 1 have the highest price value of the car policy. These groups represent younger people with high risk factor value equal 3.

Results

Table 4.5 represents the overall effort in seconds required to prepare and run the data clustering model in Weka, KNIME, Apache Mahout, and R. The obtained results indicates the following findings: model creation and processing in Mahout applications occupies a little bit more time than for other tools that was expected, because Mahout performs better on larger data sets, but with the growth of clustering number the time renames the same. Weka processed the clustering k-Means model faster.

No. of clusters	Weka (s)	Knime (s)	R (s)	Mahout (s)
3 clusters	18.72	144	100	242
4 clusters	22.26	149	102	219
5 clusters	22.4	189	110	218

Table 4.5: Use case 3: effort analysis required for data preparation and model run

Results of the use case study are the following:

- K-Means method was implemented in all applications. However clustering results in Weka were more clear, because of application of conversion numeric values to nominal.

- Another observed feature of Weka is the large number of filtering methods that can be easily applied to the data set. Knime also contains part of them. However filter that normalized data in Weka can be applied to the whole data set only, but for Knime and R it is possible to define exact attributes for example for normalization.

- The effort required to prepare a representative table or chart of results was slightly higher in Apache Mahout, because it doesn't include a GUI and it was necessary to use additional application to visualize the results. Weka includes convenient GUI that was sufficient for describing obtained results. Model in Weka automatically calculates used time and shows detailed description of clusters. Getting this statistics from R requires additional lines of code.

4.4 Use cases: Classification

This section describes scenario that was created by applying several classification algorithms in KNIME, Weka, Apache Mahout, and R. Classification is a process of using applying specific input information to choose a single output, e.g. identify to which class a new element from test data belongs on the base of train set. Computer classification systems are form of machine learning. They are used to teach the system to make decisions on the base of experience [50]. With the help of classification algorithms user can put instance in a single group that belongs to one of already given classes. Classification plays the main role in predictive analysis.

The dataset that was used for this use case is an Acquire Buyers that contains transaction history of customers that buy various products and information about offers issued for them. This research investigates how to get the value out of the large data set and build the model that will predict who from the clients will become potential buyer using weekly and monthly activity summaries of each customer.

The algorithms that were used to model classification problem with real data are k-Nearest Neighbour, Support Vector Machine (SVM), Naive Bayes, Multilayer Perceptron (MLP), Probabilistic neural network (PNN), and Decision trees.

Model	Weka	KNIME	R	Mahout
KNN	✓	✓	✓	✗
SVM	✓	✓	✓	✗
Naive Bayes	✓	✓	✓	✓
Neural networks	✓(MLP)	✓(MLP, PNN)	✓(ANN)	✗
Random Forest	✓	✗	✓	✓
J48	✓	✗	✗	✗

Table 4.6: Use case 2: completeness of applications for classifier modelling

A symbol ✗ indicates that the software does not contain the model implementation in basic installation and ✓ indicates that it includes. Because of the limitations shown in Table 4.6, the following alternative analysis was performed: a model was implemented fully using Weka application and *neural network* group of algorithms was represented with MLP; KNN, SVM, Naive Bayes and both MLP and PNN except of Decision trees were fitted in KNIME; only Naive Bayes and Random Forest methods were performed in Mahout. The R packages we used for classification were *e1071* [7] for the SVM and Naive Bayes algorithms, *kknn* [8] for k-Nearest Neighbour, *nnet* [9] for Artificial neural network (ANN) algorithm and *random forest* [10] for Random Forest classification algorithm.

J48. J48 is a machine learning algorithm based upon Ross Quilan C4.5 Decision tree method, that is often refereed as statistical classifier [51]. C 4.5 performs a binary split in case the selected variable is numeric or categorical split in case there are other variables representing the attributes [54]. J48 algorithm creates a binary tree in two phases: tree construction and tree pruning. Algorithm has wide range of parameters in tools. The default parameters were chosen:

- *confidenceFactor*: 0.25
- *numFolds*: 3
- *minNumObj*: 2

KNN. K-Nearest Neighbour classification algorithm is a type of lazy learning where the function approximated locally and computation is postponed until classification. This algorithm is called lazy because it doesn't have any training phase and all the training data should be accessible during test phase [22]. This algorithm is very simple among other machine learning algorithms and has different parameters. In research it was used with default settings:

- *KNN*: 2
- *crossValidate*: false
- *nearestNeighbourSerachAlogrithm*: LinearNNSearch

SVM. Support Vector Machine is an effective method for both classification and regression that was introduced by Vapnik in 1998 [34]. It is a an effective method because of high generalization performance without the necessity to add a priory knowledge. SVM is based on Structural risk Minimisation (SRM) and the aim of the method is to find the best fitted classification function to distinguish members of the classes in train data set. The best function can be founded by maximization of the margin between these classes [34]. Algorithm has a variety of parameters and in the research it was used with the following default settings:

- *c*: 1.0

[7] http://cran.r-project.org/web/packages/e1071/e1071.pdf
[8] http://cran.at.r-project.org/web/packages/kknn/kknn.pdf
[9] http://cran.r-project.org/web/packages/nnet/nnet.pdf
[10] http://cran.r-project.org/web/packages/randomForest/randomForest.pdf

- *kernel*: PolyKernel
- *numFolds*: -1

Random Forest. Random Forest is a group of un-pruned classification or regression trees that were developed by Leo Breiman [25]. This group is formed on the base of random selection of training data samples and features are selected in the induction process. Each tree in this classification algorithm is grown to the largest possible size without prunning [25]. Random Forest generally shows performance improvement over single tree classifier such as C4.5 or Classification and Regression Trees (CART). The classifier is constructed in a way to minimize the overall error rate and focus on prediction accuracy for the majority classes. Therefore, results of minority classes can have poor accuracy. Classifier has a variety of parameters and for the research the following were chosen:

- *numTrees*: 10
- *maxDepth*: 0 = unlimited

Naive Bayes. Naive Bayes classifier is a probabilistic classifier based on the Bayes' Theorem with strong independence assumptions and the maximum posteriori hypothesis [16]. Bayes Theorem is presented in equation 4.2 and relates the probability $P(H|X)$ of a hypothesis conditional on a given data set to the probability $P(X|H)$ of the data conditional on the hypothesis:

$$P(H|X) = \frac{P(H)P(X|H)}{P(X)} \tag{4.2}$$

where $P(H)$ is the prior distribution of parameter H, $P(H|X)$ is the posterior probability of H given new data X, $P(X|H)$ defines the likelihood function (the probability of X given existing data H), and $P(X)$ is a an evidence. A classifier assumes that the presence of a certain feature of a class is unrelated to the presence of other features. This algorithm is easy to implement and fast to run since the naive assumption of the class conditional independence reduces computational time [16].

MLP. Multi-Layer Perceptron (MLP) is a supervised trained Artificial Neural Network that uses back-propagation to classify variables [49]. This MLP has a set of input and output layers and number of hidden layers. Neurons are connected to each other and one's simulates another neuron connected to that. Adaptation of the Neural Network parameters is performed for all training attributes for the input data set [49]. Classification algorithm has a large variety of attributes. We analysed all of them and chose the most prominent:

- *hiddenLayers*: a
- *learningRate*: 0.3
- *validationThreshold*: 20

MLP was trained with 100 epochs for a likelihood maximization.

PNN. A Probabilistic Neural Network is a network that derived from the Bayesian Network and implements statistical algorithm called Kernel discriminant analysis in which operations are

organized in network with four layers such as input, pattern, summation and output layers [45]. PNN includes a fast training process, inherently parallel structure and learns more quickly than other NN models.

The testing method adopted for this use case was percentage split on train and test set. Eighty percent (80%) of randomly selected values from the data set was used to train each classifier and twenty percent (20%) was used for test evaluation. The classifier accuracy on a given test set is represented as a number of test set class labels that are correctly classified.

Data workflow preparation

The preparation of the workflow started from its upload in MySQL database with the help of ETL jobs from Talend. Dataset contains information about participants, the pre-offered transaction history that contains all items purchased and the offers issued for the customer with post-incentive behaviour. The main question that is relevant for us in this use case is to predict who will be a repeat buyer and return to purchase the same item again. Possible attributes from the data set for this particular task will be those related to the product such as `chain`, `market`, `dept`, `category`, `comp` and `brand`. Also the attribute `repeater` was identified as the predicted attribute. But these information is not enough for the classification. The information concerning offers should be interpreted in a suitable for classifier model way, therefore we defined a groups of offers that the user was granted.

Table 4.7 represents groups, where *1* implies same attribute value as was for most popular user' product before offer was issued and *0* defines the opposite value. User preferences were calculated on the base of `repeattrips` attribute.

Group number	Dept	Category	Company	Brand
Group 1	1	1	1	1
Group 2	1	0	1	1
Group 3	1	0	0	1
Group 4	1	0	1	0
Group 5	1	0	0	0
Group 6	1	1	0	1
Group 7	1	1	1	0
Group 8	1	1	0	0
Group 9	0	0	1	0
Group 10	0	0	0	1
Group 11	0	0	1	1

Table 4.7: Use case 2: groups of offers regarding department, category, company and brand where product relates

Group 1 defines the same product the user bought before he got the offer. Group 2 defines the product of another category, e.g., he used to buy still water, but received an offer for sparkling water. Group 3 defines the product of the same department and brand but competitive company. Group 4 includes offer on a product of same company but another category that can be associated

goods. Group 5 includes offers on product of the same department, e.g., user bought a water in Billa but got an offer from Lidl. Group 6 defines offers on the same products of different company, e.g., user bought sparkling water in Billa, but received an offer to buy it in competitive company Spar. Group 7 represents offers on same products but different brand, e.g., user used to buy Nestle white chocolate bar but received an offer for same chocolate but from Milka production. Group 8 defines offers for goods of the same category and department but another brand and company. Group 9 includes offers on other product of same company. Group 10 defines offers of the same brand, e.g., user received an offer on a milk production of the brand Nöm. Group 11 includes offers on the products of the same company and brand.

After all groups of offers were defined, we developed SQL queries to classify offers according to obtained groups. Final results were stored in CSV file. It is assumed that problems such as inconsistent and duplicate data have been resolved during benchmark creation. Data set does not contain any missing values. Table 4.8 represents number of observations and attributes of initial and processed data sets.

Application	**Version**
Number of observations	236. mill
Number of observations used in model	160058
Number of attributes	18
Number of attributes used in model	14

Table 4.8: Use case 2: Acquire Buyers dataset description

Weka implementation

For the classification use case we used Weka Explorer for testing purposes, KnowledgeFlow to built a visual workflow and Experimenter to run all used algorithms and further compare them. Data flow includes such main steps like import CSV file, convert it into ARFF format suitable for Weka, apply all required transformations and execute model one by one for all classification algorithms. Data import was the same as was described in previous use case.

Attribute selection. Features in the data set are used to better represent the domain, therefore some of them are relevant, other are irrelevant or redundant. The following attributes such as `productsize`, `productmeasure`, `purchasequantity`, `purchaseamount`, and `offervalue` were removed using Weka filter *Remove* because they were not relevant for our classification task. Product size, measures and quantity are various such as *OZ*, *LB*, and *LT* are not representative for us.

Nominal values. All values in this data set are numeric and 11 from them have values $0, 1$, therefore all attributes were converted using Weka filter *NumericToNominal* in order to narrow variable tolerance range. Binary attributes such as `cat1` with values 0,1 were encoded into nominal that defines 0 as false and 1 as true parameter.

Missing values. This data set does not contain any missing values.

After all transformations were finished we chosen 80/20 percentage split and run all models in Weka Experiment Environment.

Results. After training and testing classifiers the output form the Weka represents the summary of the models for each of the methods (Figure 4.8), that includes such measures as precision, recall, and a confusion matrix.

The important numbers to focus on are values for parameters *correctly classified* and *incorrectly classified instances*, 74% and 26% correspondingly. The next values are columns of a confusion matrix that represents the instances of the predicted class and each row reflects the actual class instance.

For this dataset we obtained two classes (repeat buyer or not) therefore we have $2 \cdot 2$ confusion matrix. The number of correctly classified instances is represented as a sum of diagonals in the matrix, other values are incorrectly classified instances. Confusion matrix has the following variables: *True Positive* (TP) if the the outcome from a prediction is p and the actual value is also p, in case if the actual value is m then it is called *False Positive* (FP). In opposite cases the values are *False Negative* (FN) and *True Negative* (TN) [19].

Also such values as precision and recall are observed on the Figure 4.8, where precision (see Equation 4.5) is a mesure of accuracy that a class was predicted and recall (see Equation 4.4) is a measure of model ability to select instances of a class in data set [19].

$$precision = TP/(TP + FP) \tag{4.3}$$

$$recall = TP/(TP + FN) \tag{4.4}$$

Confusion matrix of Figure 4.8 indicates the following: value 1391: is a TP, number of actual *true* instances that were correctly labelled by classifier as *true*; value 7306: *true* values that were incorrectly classified as *false*, is an FP; value 928: is FN, number of *false* instances that were incorrectly labelled as *true* and value 22386 is a TN, are *false* instances that were correctly labelled as *false*. For tree based classification methods it is possible to visualize the pruned tree (Figure 4.9).

In Weka Experimenter we calculated various measures such as Kappa Statistics, Mean Absolute Error, Relative Absolute Error and compare the performance of the predicted model for all classification algorithms [18]:

- *Kappa Statistics*: a degree measure of precision between observes or measurements of the same categorical variable
- *Mean Absolute Error*: the average value of the difference between actual and predicted measurement
- *Time*: value defines the time required to finish training and testing of the model
- *Relative Absolute Error*: a value of total absolute error divided by the error of the random walk

Table 4.9 represents the result of comparison.

The classification results show that J48 logarithm performs better that others based on the number and percent of correct classified instances. However the number of correctly classified

```
=== Summary ===

Correctly Classified Instances       23777               74.2776 %
Incorrectly Classified Instances      8234               25.7224 %
Kappa statistic                          0.156
Mean absolute error                      0.3771
Root mean squared error                  0.4337
Relative absolute error                 95.3243 %
Root relative squared error             97.5012 %
Coverage of cases (0.95 level)         100      %
Mean rel. region size (0.95 level)     100      %
Total Number of Instances            32011

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.160    0.040    0.600      0.160    0.253      0.206    0.585     0.347     t
               0.960    0.840    0.754      0.960    0.845      0.206    0.585     0.765     f
Weighted Avg.  0.743    0.623    0.712      0.743    0.684      0.206    0.585     0.651

=== Confusion Matrix ===

     a     b   <-- classified as
  1391  7306 |     a = t
   928 22386 |     b = f
```

Figure 4.8: Use case 2: classification output results for SVM classifier in Weka

```
J48 pruned tree
------------------

cat1 = 0: f (134812.0/32636.0)
cat1 = 1
|   cat10 = 0
|   |   cat11 = 0
|   |   |   cat8 = 0: f (2587.0/921.0)
|   |   |   cat8 = 1
|   |   |   |   cat4 = 0: t (11646.0/4848.0)
|   |   |   |   cat4 = 1: f (416.0/156.0)
|   |   cat11 = 1: f (5693.0/2200.0)
|   cat10 = 1
|   |   cat3 = 0: f (4807.0/671.0)
|   |   cat3 = 1: t (96.0/40.0)

Number of Leaves  :     7

Size of the tree :  13
```

Figure 4.9: Use case 2: example of classification tree in Weka

instances is similar between the classifiers. Based on achieved results it is possible to make the next summarization:

- all algorithms achieved good enough result (about 73%)
- values of Kappa Statistics and Mean Absolute Error show that the results are relevant
- SVM classifier requires more time for training and KNN for testing the model

Performance Metrics	KNN	SVM	RandomForest	J48	NaiveBayes	MLP
Number Correct	23675.40	23323.70	23684.60	23726.30	23454.30	23625.60
Number Incorrect	8336	8687.70	8326.80	8285.10	8557.10	8385.80
Percent Correct	73.96	72.86	73.99	74.12	73.27	73.80
Percent Incorrect	26.04	27.14	26.01	25.88	26.73	26.20
Percent Unclassified	0.00	0.00	0.00	0.00	0.00	0.00
Relative Absolute Error	90.04	68.62	90.93	95.37	91.43	88.43
Kappa Statistics	0.15	0.00	0.15	0.15	0.18	0.15
Mean Absolute Error	0.36	0.27	0.36	0.38	0.36	0.35
Time to test model (s)	0.05	10477.07	8.45	1	1.00	1.56
Time to train model (s)	1303.42	0.88	0.88	0.23	1.00	5278.4

Table 4.9: Use case 2: performance of the classification algorithms in Weka

KNIME implementation

In order to execute the same use case in KNIME we build a workflow using nodes for each of the classification methods such as KNN, Naive Bayes, and PNN. Since these algorithms do not work with nominal values we only remove attributes from the dataset as it was done for Weka. Since algorithm Decision tree works with nominal values, we separated a data input for this method as well, where required attributes were removed and the whole dataset was converted into nominal values. We used a 80/20 percentage split partition and applied a node *Partitioning* for that. The example of the data workflow for methods Decision tree, KNN, and Naive Bayes, is represented on the Figure 4.10. It consists of three main blocks: data preprocessing that includes nodes *ColumnFilter* and *Partitioning* nodes, classification and scoring part that includes classification nodes and scorer and visualization part.

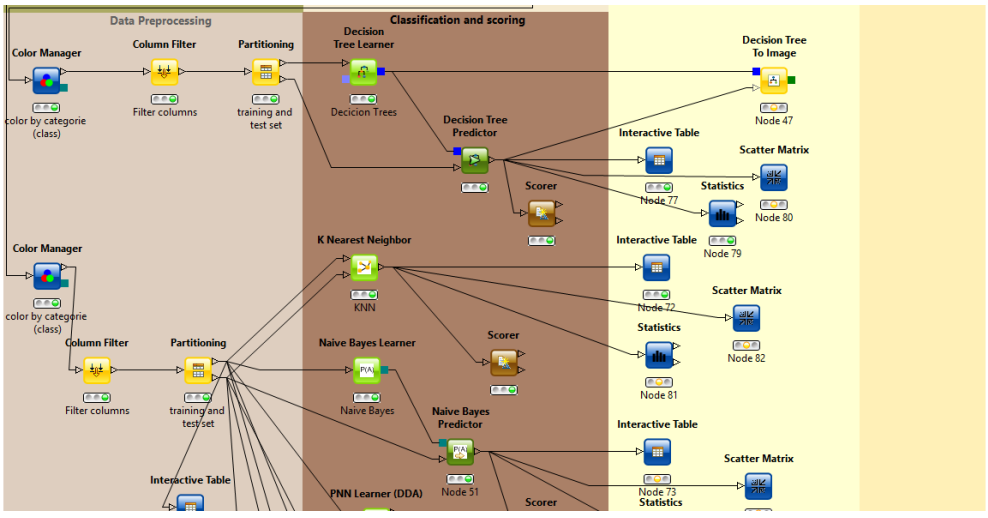


Figure 4.10: Use case 2: implementation workflow in KNIME

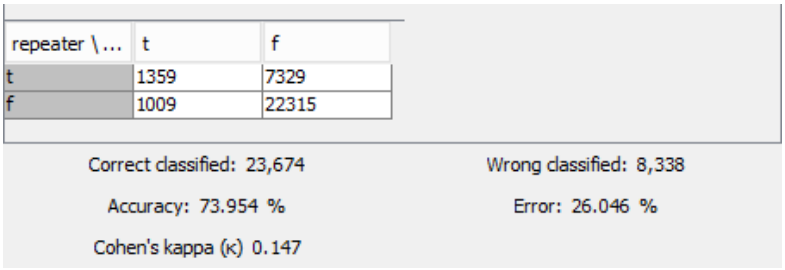


Figure 4.11: Use case 2: confusion matrix and model statistics in KNIME

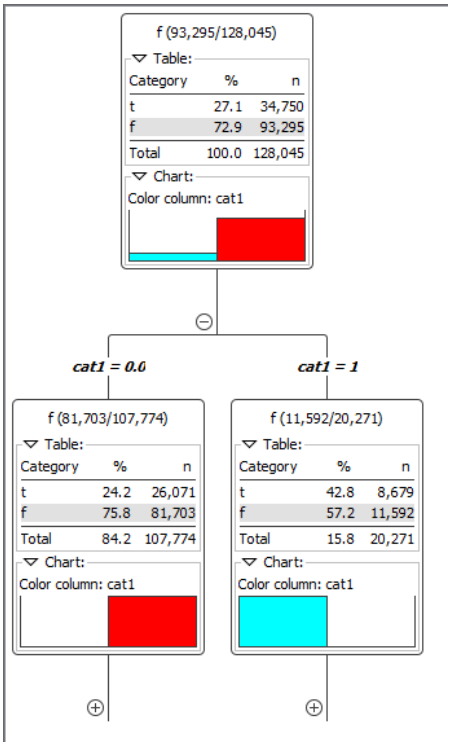


Figure 4.12: Use case 2: decision tree classifier in KNIME

Mostly all classifier algorithms in KNIME consists of two nodes: learner and predictor (e.g. Naive Bayes algorithm has two nodes: Naive Bayes Learner and Naive Bayes Predictor). The input into these two nodes always should be partitioned via node *Partitioning*, that means the training data will be an input for learner node and test data for predictor node. We put node *Scorer* after the implementation of each classification algorithm in order to get the confusion matrix result (see Figure 4.11). The output information contains accuracy value of the classifier, number of correct and wrong classified instances, Kappa statistics value and error percent.

For different classification methods KNIME enables to visualize additional information concerning learner and predictor. Figure 4.12 illustrates such example of the decision tree. We assigned different colours to `cat_1`: 0 for *false* and 1 for *true*. On the Figure the proportion of

the different classified instances is represented in each tree node and leaf.

Apache Mahout implementation

We implemented only one classification algorithms Random Forest in Mahout because of limitation of implemented methods. Same as for clustering use case we used an ARFF file from Weka. The set of steps to perform Random Forest classifier is represented in Listing 4.4 and includes: *step 1* (line $2-4$) to create directory and copy files in HDFS, *step 2* (lines $5-8$) to describe the data types: dataset contains 13 nominal values ($N13$) and one classifier (L), *step 3* (lines $8-14$) to build the classifier model and *step 4* (lines $15-19$) to run it on the test data. The used variables in the BuildForest [11] library in Mahout are the following: $[--selection --partial --nbtrees --output]$, where *-selection -sl* is a number of variables to select randomly for each tree-node, was set 5; *-partial -p* defines the implementation of partial data, *-nbtrees -t* is a number of tree to grow, was set 100; *-output -o* is the output path.

Listing 4.4: Use case 2: Mahout implementation of classification alogithm

```
1  # create hdfs directory
2  hdfs dfs /directory-name /destination-path
3  # copy ds to hdfs
4  hdfs dfs -copyFromLocal /filename /destination-path
5  # describe dataset
6  hadoop jar /path/to/mahout-jar-file
7  org.apache.mahout.classifier.df.tools.Describe
8  -p /train-file  -f /model-file.info -d N 13 L
9  # build a model from the train file
10 hadoop jar /path/to/mahout-jar-file
11 org.apache.mahout.classifier.df.mapreduce.BuildForest
12 -Dmapred.max.split.size=/split-size -d /train-file
13 -ds /model-file.info
14 -sl 5 -p -t 100 -o /output-path
15 # run Random Froest algorithm on test data
16 hadoop jar /path/to/mahout-jar-file
17 org.apache.mahout.classifier.df.mapreduce.
18 TestForest -i /train-file -ds /model-file.info
19 -m /destination-path -a -mr -o /output-path
```

The output from Mahout includes same statistical data as for previous applications such as Kappa Statistic, Accuracy, Reliability and Standard Deviation. It includes summary about classified instances and finally the confusion matrix (See Figure 4.13).

[11] http://mahout.apache.org/users/classification/partial-implementation.html

```
=======================================================
Summary
-------------------------------------------------------
Correctly Classified Instances          :      287329       65.0244%
Incorrectly Classified Instances        :      148017       34.9756%
Total Classified Instances              :      435346

=======================================================
Confusion Matrix
-------------------------------------------------------
a         b             <--Classified as
184242    50770         |  235012       a     = f
40100     160234        |  200334       b     = t

=======================================================
Statistics
-------------------------------------------------------
Kappa                                       0.1136
Accuracy                                   68.8013%
Reliability                                36.2841%
Reliability (standard deviation)            0.5227
```

Figure 4.13: Use case 2: results of classification algorithm Radnom Forest in Apache Mahout

R implementation

The implementation of algorithms in R includes scripts creation. Same as for KNIME, some algorithms such as SVM and KNN can run only on numeric data and others on nominal values. Therefore we remove the required attributes as it was done in previous tools and apply in case of Random Forest, ANN and Naive Bayes the function to convert values to nominal. Below we represent the script examples with required methods. First, we will describe the part of data preprocessing and than the usage of each specific method. The set of steps (see Listing 4.5) to perform a data transformation is same for all used classification algorithms: connect required libraries and packages, import data set (lines $7-11$) as it was shown in previous use case and remove not relevant attributes for data. Then required attributes should be converted to nominal values (lines $11-13$) for certain algorithms (such as Random Forest, Naive Bayes and ANN in our case). We convert all attributes from `market` to `cat11` to nominal. Listing 4.6 shows the 80/20 percentage split into train and test sets (lines $15-35$).

Listing 4.5: Use case 2: data preprocessing for classification alogithms in R

```
1  # used libraries
2  library(e1071)
3  library(nnet)
4  library(kknn)
5  library(randomForest)
6  library(kernlab)
7  # read datase from file
8  ds1data <- read.csv(file="/file/destination",head=TRUE,sep=",")
9  # remove columns
10 ds1data <- ds1data[,-1]
```

```
11 # numeric to nominal for all columns
12 ds1data$market <- as.factor(ds1data$market)
13 #additional columns: repeater, cat1-cat11
```

Listing 4.6: Use case 2: 80/20 percentage split of the data set in R

```
1  ## prepare test and train data
2  # retrieve the class column
3  class <-ds1data[,2]
4  attribs <- ds1data[,c(-2)]
5  # number of row
6  nbrow=nrow(ds1data)
7  # use 80 % of objects as train
8  ntrain <- round(nbrow*0.80)
9  # sample
10 tindex <- sample(nbrow,ntrain) # indices of training samples
11 train <- attribs[tindex,]
12 test <- attribs[-tindex,]
13 trainf <- ds1data[tindex,]
14 testf <- ds1data[-tindex,]
15 classtrain <- class[tindex]
16 classtest <- class[-tindex]
```

Listing B.1 contains methods that will be used in Random Forest classifier to work with trees.

The next step Listing 4.7 includes creation of classification model for all algorithms: KNN (lines $1-2$), Random Forest (lines $10-19$), SVM (lines $5-7$), ANN (lines $8-9$), Naive Bayes (lines $3-4$).

Listing 4.7: Use case 2: classification models for algorithms in R

```
1  # KNN
2  model <- kknn(repeater~., trainf, testf, k = 2, distance = 1,kernel = "triangular")
3  # Naive Bayes
4  model <- naiveBayes(train, classtrain)
5  ## SVM
6  # learn the model, type="C-classification" is set so that
7  model <- svm(train, classtrain,type="C-classification")
8  # ANN
9  model <- nnet(train, classtrainann, size=2)
10 ## Ranodm Forest
11 # fit the randomforest model
12 model <- randomForest(train, classtrain, importance=TRUE,keep.forest=TRUE)
13 tree<-getTree(model, k=1, labelVar=TRUE)
14 # rename the name of the column
```

```
15 colnames(tree)<-sapply(colnames(tree),collapse)
16 rules<-getConds(tree)
17 print(rules)
18 # what are the important variables (via permutation)
19 varImpPlot(model, type=1)
```

After the model was create we perform prediction, that is similar for all tools (lines $1-2$) and execute the additional script in order to get the confusion matrix and other basic statistics (lines $3-7$) (see Listing 4.8).

Listing 4.8: Use case 2: calculation of the prediction results and basic statistics in R

```
1 # prediction
2 prediction<-predict(model, test)
3 # summary of the model
4 summary(model)
5 # confusion matrix
6 tab <- table(pred = prediction, true = classtest)
```

The output includes information about built model and the confusion matrix (See Figure 4.14).

```
> summary(model)
a 12-2-1 network with 29 weights
options were -
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1 i10->h1 i11->h1 i12->h1
  -1.31   -5.93   -0.73   -0.24   -0.61   -0.40   -1.13    0.11    0.14   -1.09    0.40   -0.27   -0.33
  b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2  i9->h2 i10->h2 i11->h2 i12->h2
   6.20   23.25    1.24    1.49   -0.48    0.91    6.17    0.00   -0.28    2.52    1.48    0.46    1.30
   b->o   h1->o   h2->o
1088.16  952.41 1060.46

> tab
    true
pred     1     2
   1 23334  8677
```

Figure 4.14: Use case 2: results of the ANN classification model in R

Results

Results of the use case indicate the following findings: in the research we have compared six classification algorithms KNN, SVM, Random Forest, Neural Networks, Naive Bayes and J48 that are effective techniques for data classification. We used Weka, KNIME, Apache Mahout, and R tools and predict who will be a potential buyer among customers. We have achieved nearly 78% accuracy. The efficiency of results can be further improved by increasing the number of training and test data and incorporating various attributes. Table 4.10 represents the overall effort of model creation and processing for used applications. The obtained results indicates the following findings:

- Apache Mahout classified data set faster then other tools
- SVM itself is very slow, because its time complexity is $O(n * n)$, however with KNIME application we processed the model in 35 hours

Model	Weka (s)	KNIME (s)	R (s)	Mahout (s)
KNN	1303.2	197.4	180	✗
SVM	10800	126000	21600	✗
Naive Bayes	93	5	171	✓
Neural networks	5280 (MLP)	180 , 1800 (MLP, PNN)	600 (ANN)	✗
Random Forest	600	✗	600	81
J48	3.66	✗	✗	✗

Table 4.10: Use case 2: effort analysis required for data preparation and model processing

4.5 Use cases: Regression

This section describes scenario that was created by applying several regression algorithms to the data. Regression is popular method for data mining, that allows to predict variable based on the given set of other input values. Profit, trends, prices, and temperature could be predicted using regression techniques.

For this use case we used the Walmart Store Sales dataset that contains historical sales information about 45 Walmart stores in different regions. The goal is to build model that will predict department sales in the next couple of time slots.

The algorithms that we used for this use case are Linear Regression, MLP and Polynomial Regression.

Model	Weka	KNIME	R	Mahout
LinearRegression	✓	✓	✓	✗
MultilayerPerceptron	✓	✗	✓	✗

Table 4.11: Use case 3: completeness of applications for regression modelling

Because of limitations shown in Table 4.11 the following alternative analysis was performed: a model was implemented fully for Weka with the regression methods Liner Regression and MLP; in KNIME only Liner Regression was fitted. The R package we used for regression algorithms was *stats* [12]. Apache Mahout methods doesn't currently include the implementation of the regression models.

LinearRegression. Linear Regression method enables to find the best line to fit two attributes so that one can be used to predict other [20]. We will use in our research also a Multiple Linear Regression where several attributes are involved in the model creation and data prediction in a multidimensional surface. Equation 4.5 represents the simple linear regression that models single response variable y as a linear function of single predictor variable x [20]:

$$y = b + wx, \tag{4.5}$$

[12]http://stat.ethz.ch/R-manual/R-patched/library/stats/html/00Index.html

where w and b are regression coefficients: w is a slope (angle between a data point and a regression line) of the line and b is an intercept (the point where x crosses the y axis). Multiple Linear Regression involves n predictor variables to model a linear function. Equation 4.6 represents the Multiple Linear Regression based on two predictor attributes A_1 and A_2:

$$y = w_0 + w_1 x_1 + w_2 x_2, \tag{4.6}$$

where w_0, w_1, w_2 are coefficients and x_1, x_2 are values of the attributes A_1 and A_2 respectively [20].

Data workflow Preparation

The preparation of the data workflow started from its upload in MySQL database. Dataset contains information about store sales, departments, and special holiday weeks. The main idea in this use case is to forecast stores sales. The dataset for regression model example will focus on attributes related to stores such as `temperature`, `markdowns`, `cpi`, `unemployment`, and `fuel_price` to predict store sales values. We will create a regression model, that is based on comparable attributes of other department sales in a store. The dataset has a structure presented in Table 4.12, where sales related information is presented for each store consecutively and values in column `Date` repeat for each department of each store. Such type of data representation

Store	Department	Date	Markdown1	Markdown2	Weekly_Sales
1	1	2010-02-05	NA	NA	24924.5
1	1	2010-09-03	NA	NA	16241.78
...	...	...	...	...	...
1	2	2010-02-05	NA	NA	50605.27
...	...	...	...	...	...

Table 4.12: Use case 3: initial data set structure

is not suitable for implementing the regression model. Therefore, we restructured data in the way as it is presented in Table 4.13, where dataset values are grouped by timestamp for each department of each store. Such structure enables to avoid time variable duplication. Columns obtained the following name format: $WS\_\%1\_\%2$, where _$\%1$ refers to number of store and _$\%2$ refers to number of store department.

WS_1_1	WS_1_2	WS_1_n	Date	Markdown1	Markdown2
24924.5	50605.27	10891.37	2010-02-05	NA	NA
46039.49	44682.74	16309.73	2010-09-03	NA	NA
...	...	...	...	...	...

Table 4.13: Use case 3: changes in data set for regression model

Since each store contains around 90 departments, we wrote a Java application to generate an SQL queries for our task. In data set there are 45 stores with approximately 90 departments

each, stores have various type and size. Given the large quantity of stores and departments and the complexity of the data analysis problem it was resolved to limit the number of analysed stores to three and choose only one department per each store. We have compared all of them on various parameters and chosen three different stores of various type and store size (see Table 4.14).

The Java application was applied to those three stores and then we divide obtained queries on several groups in order to run in MySQL, because there was a limit of JOIN values. We wrote another Java application to merge obtained results.

Attributes	**Store 1**	**Store 2**	**Store 3**
store_id	1	18	37
type	A	B	C
number_of_departments	77	78	62
size	151315	120653	39910
max_temperature	91.65	79.75	87.64
min_temperature	35.40	14.84	41.16
max_fuel_price	3.9070	4.1010	3.9070
min_fuel_price	2.5140	2.7160	2.5140
max_cpi	223.44	138.911	222.11
min_cpi	210.34	131.53	209.12
max_unemployment	8.11	9.34	8.55
min_unemployment	6.57	8.08	6.23

Table 4.14: Use case 3: comparison of chosen stores for data regression

We will do a prediction for a period from $2012-08-24$ until $2012-10-26$ with a step of 7 days. Finally CSV files were ready for the input in applications for performing forecast. Table 4.15 represents number of observations and attributes of initial and processed data set.

Application	**Version**
Number of observations	400000
Number of observations used in model	144
Number of attributes	20
Number of attributes used in model	90

Table 4.15: Use case 1: Walmart Store sales dataset description

Weka implementation

Weka model time series data by transforming it into standard algorithms to process. It encode time dependencies (lag values) and relations between attributes. We loaded the CSV file in Weka and converted it in ARFF format. Further step is to built a Linear Regression model. For the current use case we used Weka Explorer and additional tab for Time Series Analysis and

Forecasting developed by Weka and Pentaho team. For a regression task it is important to use numeric not normalized values, therefore we did not perform any changes on data after load it into the Weka tool. Given the examples we would like to compare the predicted values and to learn whether there are a relations between them that can improve the result.

After the data input we chose two basic algorithms Linear Regression and MLP and model several series that can give different results. Since each store contains various departments and additional attributes related to store, we defined three groups of models for each of the algorithms and selected only one department per store for store sales forecast:

- *Group 1*: includes only store sales information for one department
- *Group 2*: includes attributes from the Group 1 and additional attributes such as `cpi`, `unemployment`, `temperature` and `fuel_price`
- *Group 3*: includes attributes from the Group 2 and store sales information for additional three departments

Since in Multiple Linear Regression model all used attributes can influence the result of the prediction, we decided to compare for which of the groups mentioned above results of the prediction will be more accurate. Also we will compare results of model implementation with Linear Regression and MultilayerPerceptron.

Missing values. The dataset includes missing values. Weka automatically replaces missing values with mean values.

Results. When model was created and regression algorithms were finished the Weka output represents the summary of the model for each of the applied methods. In Weka all attributes that were used in case of Multiple Linear Regression model are also predicted apart from the main `store_sales` values and for each of them application generates separate regression model. By default the application generate forecast and perform training evaluation, calculate metrics such as Mean Absolute Error, Root Mean Squared Error (see Figure 4.15).

```
=== Evaluation on training data ===
Target                     1-step-ahead  2-steps-ahead  3-steps-ahead  4-steps-ahead  5-steps-ahead  6-steps-ahead  7-steps-ahead  8-steps-ahead  9-s
==========================================================================================================================================================
ws_18_27
  N                                  81             80             79             78             77             76             75             74
  Mean absolute error          130.4245       137.9259       155.4314       149.6534       150.5512       152.7564       157.1325       159.4988
  Root mean squared error      163.3438       168.2059       198.7125       191.8042       192.8829       192.8691       196.9211         199.01

Total number of instances: 133
```

Figure 4.15: Use case 3: evaluation of the Liner Regression model in Weka

Figure 4.16 shows Linear Regression model equation with parameters that were predicted. The Figures 4.17, 4.18, 4.19 illustrate forecast values for three Groups defined above. The obtained results indicate the finding that prediction results for store department are more accurate when additional attributes related to the store are included in the regression model.

Figure 4.6) illustrates the forecasted values in the last 10 points of the dataset for department 27 of store 18. Large picks on the chart correspond to Super Bowl holidays and smaller one to Labour Day. The slopes correspond to period of low consuming activity periods of the year:

```
ws_18_27:

Linear Regression Model

ws_18_27 =

    242.4467 * Month=may,aug,sep,apr,jan,mar,oct,feb,nov,dec +
   -145.42   * Month=sep,apr,jan,mar,oct,feb,nov,dec +
    -46.1389 * Month=apr,jan,mar,oct,feb,nov,dec +
     78.6421 * Month=jan,mar,oct,feb,nov,dec +
    260.4784 * Month=mar,oct,feb,nov,dec +
     94.6631 * Month=oct,feb,nov,dec +
    370.4824 * Month=feb,nov,dec +
    175.2954 * Month=nov,dec +
    182.8532 * Month=dec +
     78.6421 * Quarter=Q1,Q4 +
   -151.9484 * Quarter=Q4 +
     -5.8787 * date-remapped +
     -0.2374 * Lag_ws_18_27-1 +
     -0.0354 * Lag_ws_18_27-2 +
     -0.0349 * Lag_ws_18_27-3 +
     -0.0592 * Lag_ws_18_27-5 +
     -0.0801 * Lag_ws_18_27-9 +
```

Figure 4.16: Use case 3: evaluation and training results from regression model in Weka

May till July. The same picks and slopes are observed on the charts that forecast store sales with additional attributes.

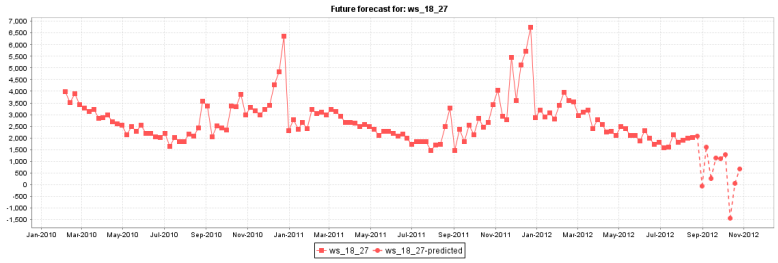


Figure 4.17: Use case 3: forecast of the store sales for store 18 department 27 in Weka

Table 4.16 show the Mean Error value for three groups of two regression methods. The compute error is lower for the Group 3 for LR model.

KNIME implementation

In order to execute the use case in KNIME we build a workflow (see Figure 4.6)). For this use case we did not perform any data transformations. In KNIME it was necessary to convert the time parameters from string values to Date/Time using *StringToDate/Time* node and then order data by time using node *Sorter*. Regression models in KNIME require partitioning of the data

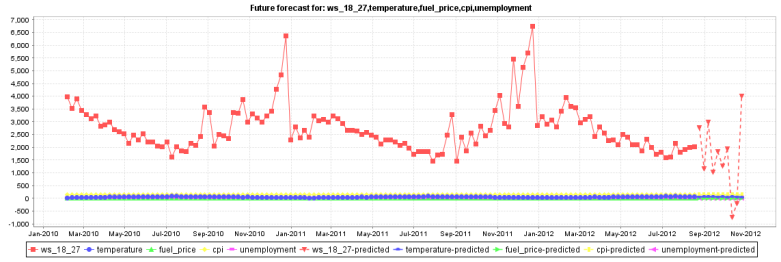


Figure 4.18: Use case 3: forecast of the store sales for store 18 department 27 and additional attributes `cpi`, `unemployment`, `fuel_price` and `temperature` in Weka

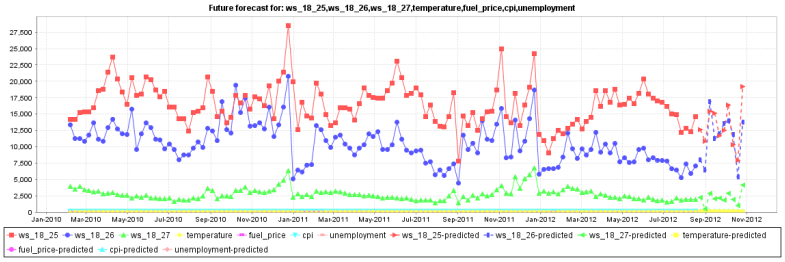


Figure 4.19: Use case 3: forecast of the store sales for store 18 department 27 and additional attributes `cpi`, `unemployment`, `fuel_price` and `temperature` and two additional departments 25 and 26 in Weka

Group	Model	Store 1	Store 2	Store 3
Group 1	LR	18200.57	2066.527	-65.573
	MLP	-1261.715	392.52	35.26
Group 2	LR	-3935.604	1131.375	140.976
	MLP	-1332.686	142.4761	90.383
Group 3	LR	-1284.206	557.041	36.603
	MLP	-11246.13	113.307	353.9272

Table 4.16: Use case 3: performance of the regression algorithms in Weka

set into two parts learner and predictor. In our case it was *Linear Regression Learner* to build the model and *Regression Predictor* to forecast values.

KNIME application give the possibility to approximate but not extrapolate values. Therefore we build auto-regressive model and use past timer series data values to build a learner. First, we read data using `File Reader` node and then sort data values by ascending order to ensure meaningful time sequence. Then we splitted data into train and test sets and after training `Regression Predictor` node predicts the current value. The example of such prediction for store 18, department 27 is presented on Figure.

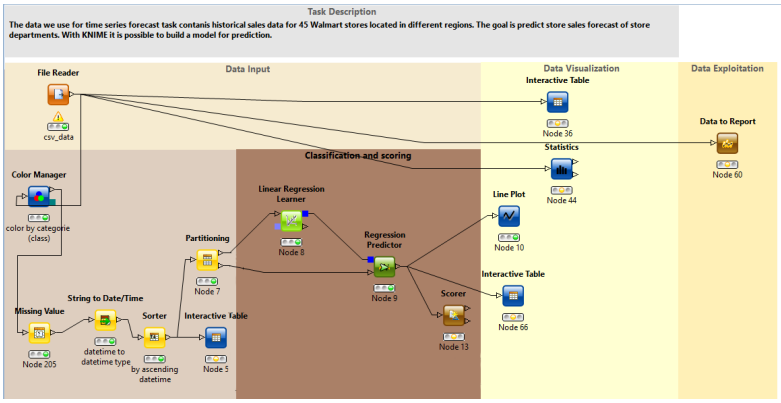


Figure 4.20: Use case 3: implementation of regression workflow in KNIME

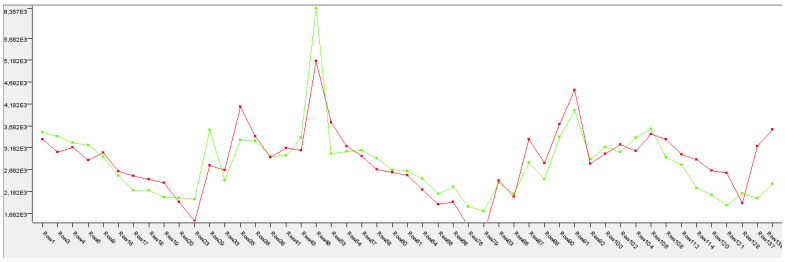


Figure 4.21: Use case 3: regression results of *Liner Regression Learner* in KNIME

R implementation

The implementation of algorithms in R includes scripts creation. Same as for other previous tools we did not perform any data transformations. The set of steps to perform Linear Regression in R is following: first it is necessary to add a library (lines $1-2$) and then import data as it was done in previous use cases (lines $3-4$), then we choose relative columns with necessary attributes for three groups (lines $5-7$). Next step is to build time series (lines $8-9$) and fit a linear regression model: $lm(Y\ model)$, where Y contains the dependent variables to be predicted and model represents the formula for chosen model (lines $10-11$). The last step is to do a forecast (lines $12-18$).

Listing 4.9: Use case 3: implementation of a Linear Regression model in R

```
1 # used libraries
2 library(stats)
3 # read datase from file
4 ds_2_values.s1d5 <- read.csv(file=/path/to/file,head=TRUE,sep=",")
5 # take 1 column - use case 1
6 ds_2_values.s1d5 <-ds_2_values.s1d5[,c(5)]
7 ds_2_values.s1d5
8 # built time series
```

```
ds2.ts =ts(as.numeric(ds_2_values.s1d5$ws_1_5), start =c(2010,02),freq=52)
# fit a linear regression model
fitmodel= lm(ds2.ts~time(ds2.ts))
# forecast
Seas=cycle(ds2.ts)
Time = time(ds2.ts)
ds2.lm=lm(ds2.ts~0+Time+factor(Seas))
new.t <- seq (2012, len= 2 * 24, by = 1/12)
new.dat <- data.frame (Time = new.t, Seas = rep(1:12, 2))
ds2.forecast = ts(predict(ds2.lm,new.dat)[1:24],start=c(2012,10), freq=52)
```

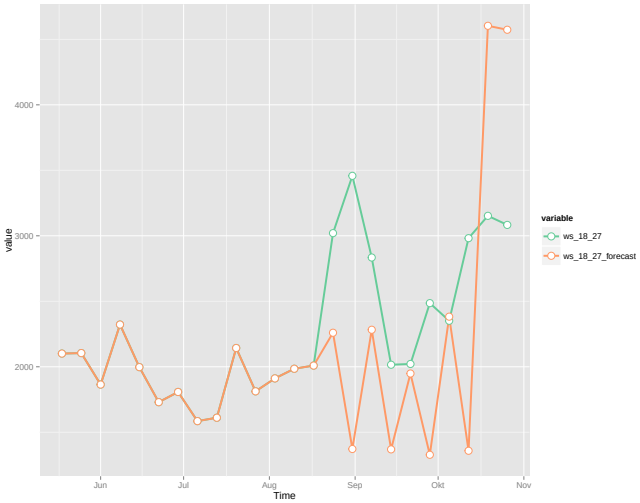
Before accepting the model results of a Linear Regression it is important to evaluate it suitability and one of the main ways apart from Mean Error computing is a visualisation. Table 4.17 represents the widely used measure for regression comparison - Mean Error. Figure 4.22 illustrates application of MLP for three observed groups, where *red* line represents the forecast and *blue* line the actual value.

Results

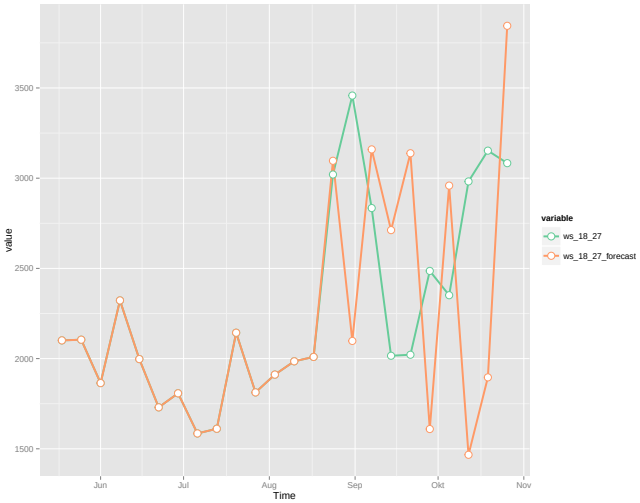
Results of the use case indicate the following findings: have compared two regression algorithms LR and MLP and observations show that the effectiveness on the current data set of a regression algorithms depends on number of variables used for analysis. Because of application limitations we used only Weka and R tools and forecast store sales for ten timestamps. We built three groups of cases that very in number of attributes. The best suitable result was obtained with group number 3 where apart from store sales of main department, other two departments and related attributes were used to form a multi parameters equation and build model on the top of it. Table 4.17 represents the overall effort of model creation and processing for used applications. As can be observed, MLP algorithm requests more time to perform evaluation.

Model	Group	Weka	KNIME	R	Mahout
LinearRegression	Group1	4s	62 s	60 s	✗
	Group2	180 s	300 s	300 s	✗
	Group3	840 s	600 s	420 s	✗
MuliLayerPerceptron	Group1	60 s	120 s	60 s	✗
	Group2	3600 s	2400 s	2700 s	✗
	Group3	14400 s	12600 s	10800 s	✗

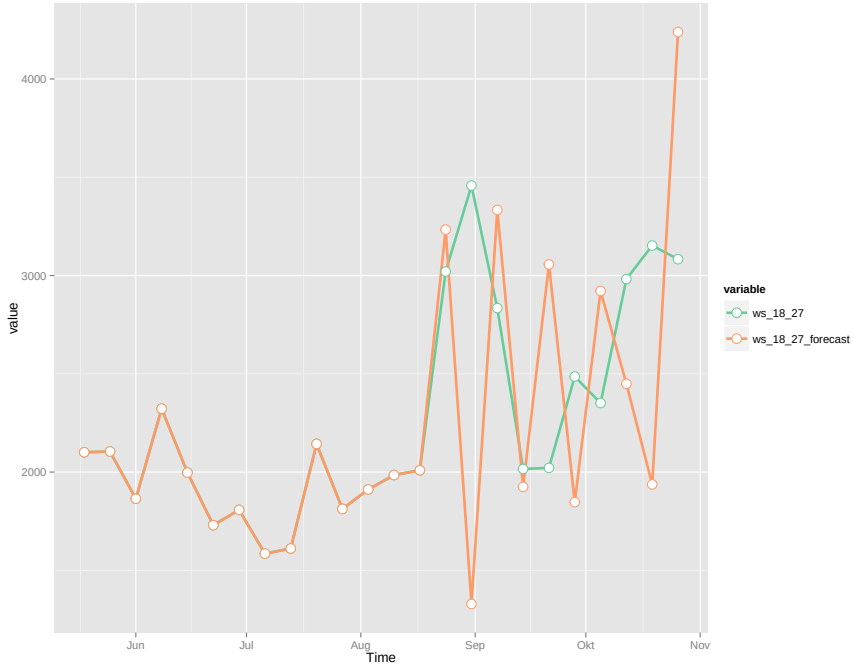
Table 4.17: Use case 3: effort analysis required for data preparation and model run



(a) Group 1



(b) Group 2



(c) Group 3

Figure 4.22: Use case 3: compare of the regression groups in R

CHAPTER 5

Application Comparison

Chapter 4 describes the implementation process of three scenarios with four tools for big data analysis and evaluation of these tools. This chapter will include the assessment and comparison of Knime, Weka, Apache Mahout, and R tools in the context of performance, use case modelling and other characteristics such as visualization capabilities, application characteristics. Nowadays there are various options that user can select to perform big data analytic. Given the broad review for chosen products in the domain of usage of various functions and features, we compare applications qualitatively that intends comparison of reliability, visualization possibilities, scalability, and ease of data representation. Much of the present research activities are concerned with reducing time for data preparation and transformations, proper parameter selection, fast calculation, and visualization of achieved results.

Big data analysis tools were used to uncover relationships and extract values from the time series datasets. Each software application includes capable techniques to work with different data set sizes, attributes and measurement types. One common feature among the various applications is their possibility to leverage the combination of collections of the following computing characteristics as storages and memory (large number of tools load data sets in main memory and therefore has a limit of processing). Memory is loosely coupled with possibility of the application to scale, enlarge computing memory and work with unlimited data volumes. Processing capability refereed to processors, number of nodes and CPUs.

Basic Specifications

Table 5.1 represents basic specifications such as availability of GUI. As it described in Table, R does not have a GUI, but there are various open source studios that implements CRAN functionality and allow to write scripts, install packages, observe variables and state of the calculations at any time point. From this point of view Apache Mahout has a disadvantage, because the only possibility to see the results is a log file after the invoked function was finished.

Feature	Weka	KNIME	R	Mahout
GUI	✓	✓	✗	✗
Command line	✓	✓	✓	✓
API	✓	✓	✓	✓
Technical support	online/forum	online/forum	online/forum	online/forum
Plug-ins installation	✓	✓	✓	✓

Table 5.1: Basic specifications

Database access and file format

Data sets can be presented in various format. Table 5.2 represents basic file formats that are available for applications and limitations of input data volume. Although, according to technical documentation there are no limitations on input data set volume, the effectiveness how fast data mining models will be processed varies. Because single-node machines are limited in their memory, R, Weka and KNIME applications cannot easily accommodate massive data. It was observed in the Chapter 4 during use case processing when time spent to calculate SVM classification model in KNIME was 10 times higher than for the same model in Weka.

All tools have access to databases: Weka and KNIME can be connected from GUI to MySQL, Oracle, SQLLite and other databases using JDBC driver; R has various packages such as RODBC [1] that implements ODBC database connectivity and allows to connect to various databases like Oracle and MySQL. Data mining library Mahout can be integrated with such external data source as MySQL and also with internal one such as HDFS.

Feature	Weka	KNIME	R	Mahout
Db connection	✓	✓	✓	✓
CSV data format	✓	✓	✓	✓
ARFF data format	✓	✓	✓	✓
PMML data format	✓	✓	✓	✓
File format converter	✓	-	-	-
Real data processing	✗	✗	✗	✗
Data volume limit	no limits	no limits	no limits	no limits
Possibility to edit data manually	✓	✓	✓	✓

Table 5.2: Data access and file format

Data Mining Techniques

Data that is collected from various data sources can include noise, missing values and inconsistency. Such low-quality data will produce a low-quality result, therefore data preprocessing techniques such as data cleaning, integration, normalization of values and basic operations help

[1] http://cran.r-project.org/web/packages/RODBC/RODBC.pdf

to correct and prepare data for a data mining algorithm. Basic operations with data include the following methods: create, read, copy attributes, replace and generate values, re-sample, merge attributes and values. Data cleaning techniques include methods to remove noise and correct inconsistencies [20]. Data integration techniques allow to merge data from various database sources. Table 5.2 represents data processing techniques available for each application. Weka contains functions for all basic preprocessing techniques that can be applied to attributes and instances separately. Since Weka Explorer can work with only one file, it is not possible to integrate several data storages in one time period. KNIME and R includes implementation of basic data mining techniques. Mahout also includes various preprocessing techniques, but the user do not have to decide what method or filter to use with given dataset. For example, in case of preprocessing classification model it is required to describe the data set using Mahout class `org.apache.mahout.classifier.df.tools.Describe` [2] and a file descriptor for a given dataset will be generated.

Feature	**Weka**	**KNIME**	**R**	**Mahout**
Basic operations with data	✓	✓	✓	✓
Conversion into nominal value	✓	✓	✓	✓
Normalization	✓	✓	✓	✓
Discretization	✓	✓	✓	✓
Data cleaning	✓	✓	✓	✓
Data integration	✗	✓	✓	✓
Missing values	✓	✓	✓	✓

Table 5.3: Data processing

Availability of classification algorithms in software applications is observed in Table 4.4. Algorithms are formed in several groups such as Naive Bayes, Decision trees, Rules, and Neural Networks. *Decision trees* group for the data classification includes the following well-known algorithms as J48, Random Forest. Group of *Naive bayes* classifiers includes various modifications of Naive Bayes algorithm and quality measures. *Rules* is an alternative to decision trees that include various types such as association rules, JRip.

Algorithm	**Weka**	**KNIME**	**R**	**Mahout**
Naive Bayes	✓	✓	✓	✓
Decision trees	✓	✓	✓	✓
Rules	✓	✓	✓	✗
KNN	✓	✓	✓	✗
SVM	✓	✓	✓	✗
Neural networks	✓	✓	✓	✗

Table 5.4: Data classification algorithms

[2] http://grepcode.com/file/repo1.maven.org/maven2/org.apache.mahout/mahout-core/0.6/org/apache/mahout/classifier/df/tools/Describe.java

Table 5.5 represents the availability of clustering algorithms in applications. Clustering methods are formed in the following classes as hierarchical clusters (include bottom-up and top-down algorithms, COBWEB method), partitioning methods that include k-means and density based algorithms, probabilistic clusters such as EM method and other.

Algorithm	Weka	KNIME	R	Mahout
K-Means	✓	✓	✓	✓
EM	✓	✗	✓	✗
Fuzzy clustering	✗	✓	✓	✗
Fathers first	✓	✗	✗	✗
COBWEB	✓	✗	✗	✗
Hierarchical cluster group	✓	✓	✓	✗
Density based cluster group	✓	✗	✓	✗

Table 5.5: Algorithms for data clustering

Availability of regression algorithms is observed in Table 5.6. Regression analysis includes regression methods and classification methods that can be adapted for prediction. For application comparison we have chosen well known regression algorithm groups such as Neural Networks, Naive Bayes algorithms, various statistical methods, regression analysis group that includes Liner Regression and Multi Linear Regression methods, and ARIMA model.

Algorithm	Weka	KNIME	R	Mahout
Regression analysis	✓	✓		✗
ARIMA	✗	✗	✓	✗
Neural networks	✓	✓	✓	✗
Fuzzy	✗	✗	✓	✗
Genetic algorithms	✗	✗	✓	✗

Table 5.6: Algorithms for data regression

The performance and accuracy of each algorithm can be measured using various metrics such as Kappa Statistics, Mean Absolute Error. Table 5.7 represents the metrics availability in applications and possibility to compare different algorithms based on them inside the application. Application investigation gave the result that in all applications such metrics are implemented. Additionally, Weka Experimenter enables to run several classification algorithms and compare performance, train, and test time inside Experimenter GUI.

Feature	Weka	KNIME	R	Mahout
Model quality	✓	✓	✓	✓
Quality comparison of several models	✓	✗	✓	✗

Table 5.7: Quality control

Visualization Capabilities

Visualization capabilities are useful for proper analysis of results, especially in cases such as clustering and regression. In table 5.8 the comparison of visualization possibilities and representation of achieved results is observed. Both KNIME and Weka KnowledgeWorkflow enable user to create data flow from input of the values until results scoring using spacial nodes that can be drag and drop.

Feature	Weka	KNIME	R	Mahout
Pipeline environment	✓	✓	✗	✗
Description of methods/nodes	online/application	online/application	online	online
Possibility of data visualization	✓	✓	✓	✗
2-D graphs	✓	✓	✓	
3-D graphs	✗	✓	✓	
Possibility of reporting	✗	✓	✓	✗

Table 5.8: Possibilities and approaches of data visualization

Big data capabilities

Table 5.9 represents the possibilities of applications to process large data sets. Among all chosen applications only Apache Mahout that utilizes Apache Hadoop has a computational capabilities to scale using MapReduce paradigm from several servers to thousands machines. Weka does not have a possibility to distribute jobs, but software application team developed new packages to connect to Hadoop and distribute tasks. KNIME application also has plug-ins to work with Hadoop, but it also includes possibility to run several workflows in parallel.

Feature	Weka	KNIME	R	Mahout
Parallel execution	✗	✓	✓	✓
Scalability	✗	✗	✗	✓

Table 5.9: Efficiency to work with big data

Summary

In the research work four applications were compared. As it was observed during first steps of tools installation and application for the use cases, each data mining tool should be selected depending on the specific project, size of the data set and time that can be spent to achieve results. The recommendations below are based on the research experience.

Weka. Weka is a fully functional data analysis application compatible with almost any platform that provides a high level of functionality. This is a widely known tool in scientific world. It can be used for small size experiment projects, where it is necessary to run several data mining algorithms on the existed data set, compare the result based on various metrics to

define best fitted method. Weka enables fast and easy preprocessing of the data set, has a solid community support and can be easily extensible. This is a proper tool for those who need a solid, easy-learning tool with large amount of data mining techniques (approx 1000 algorithms).

KNIME. KNIME is an open source tool with pipeline style of data flow building that can be used for projects were specific data mining techniques are required. Applications have a large amount of additional plug-ins in the field of chemistry informatics and bio-informatics. It is a powerful solution for data mining that is easy to use, but flexible enough to cope with various types of problems. KNIME enables access from GUI to various databases, powerful visualization techniques and reporting. Application has a unique database port system and connects to any db that is JDBC compliant, therefore users can easy manipulate with data storage using SQL. For users who are not familiar with SQL KNIME provides a possibility to filter and import data in databases through GUI.

R. R is the most comprehensive statistical analysis package available that incorporates almost all standard statistical models and analyses. It provides large variety of methods for data preprocessing, managing and manipulating. R is a package based system, that has a core module and user enabled to easily install additional required packages. R provides a comprehensive visualization capabilities. Although language is implementation in the way to consume initial memory, it is a proper solution for small and medium projects, where user is expected to have programming knowledges.

Apache Mahout. Apache Mahout is the proper solution for large data sets and the performance of this scalable library can be observed when the benchmark set has more than 1 million of training samples. Apache Mahout utilizes Apache Hadoop and together with additional modules HDFS, Hive, HBase correspond as a very powerful solution for processing large amounts of data very fast and perform various data mining tasks such as classification, recommendation, clustering etc. In spite of absence of GUI, lack of implemented algorithms and time that can be spend to install the modules because of Java version incompatibility, this is very powerful and suitable solution for large projects.

CHAPTER 6

Conclusions and Future Work

This chapter contains the thesis summary. It includes the description of the completed research, application of big data analysis tools and use cases that were performed, the possible future work.

6.1 Conclusions

Big Data analysis is a widely used process to examine massive data volumes and uncover hidden patterns and unknown correlations. The primary goal of analysis of Big Data is to help companies, scientists and stakeholders to receive valuable information using various techniques as data mining and machine learning.

The important issue related to Big Data analysis is the amount of applications, which support processing of large data sets. The amount of such tools grows faster than the capabilities for their processing and each of them provide various functionality in data storing and preprocessing, modelling, and visualization. Some of them required additional knowledges, other have limitations of data format.

The goal of the research was to investigate and compare applications for big data analysis by running three standard use cases based on time-series benchmarks. Based on this, two main goals were distinguished.

First task was to define big data tools and input data sets for them. One the basis of the open source application survey we chose four widely used tools Weka, Apache Mahout, KNIME and R and three benchmarks with real data that contains historical information about customers, products and enterprises. We defined three use cases for these benchmarks that utilize main data mining techniques such as classification, clustering and regression. Each use case included various steps as data source analysis, filtering, data partition, visualization, and reporting.

Second task was to perform evaluation of applications. We compare chosen applications on the base of achieved results using benchmarking methodology with appropriate set of features. We mainly concentrated on quantitative characteristics such as overall completeness, effort to

develop and run model and qualitative characteristics such as visualization capabilities, feasibility, and scalability.

Based on the performed evaluation we distinguished important features, weaknesses and strengths of applications that can be useful for the users in decision making process to choose the appropriate application.

6.2 Future Work

This study is an initial attempt to compare the big data analysis applications in order to reveal the differences between tools that can be applied for solving analogue tasks and include similar functionality.

Based on our evaluation there are various features that can be added to support more precise big data applications comparison:

- *Data set*: we used only real datasets. They vary in volume, number of attributes and types. However after preprocessing and adaptation them for the use case requirements, we obtained less number of instances. It is possible to use additional benchmark generators that produce synthetic data of different volume and number of instances. From our point of view, it will be useful to use benchmarks with volume over 10 GB and number of instances around 500 millions.

- *Comparison*: the application comparison was the main goal. Since classification, clustering and regression model results in these application varied, it will be possible to focus on comparison of algorithms quality and veracity of the results between tools.

- *Algorithms*: In the description of the applications we mentioned several data mining algorithms that have not been used in the current version of project: EM clustering algorithm, COBWEB, classification rules, ARIMA model. Additionally to this methods, it is possible to add recommendation engine implemented in Apache Mahout.

- *Applications*: We used only four data analytic applications Weka, KNIME, Apache Mahout, and R. It is possible to used other various framework and tools for big data analysis such as Pentaho and Orange.

With these improvements we could make our research more useful and all-sufficient. With the rapid growth of big data analysis applications and approaches of storing and processing large amounts of data we need to follow the trends.

APPENDIX A

List of Abbreviations

API	Application Programming Interface
AMPP	Asymmetric Massively Parallel Processing
ANN	Artificial Neural Network
ARFF	Attribute-Relation File Format
ARIMA	Autoregressive Integrated Moving Averages
BI	Business Intelligence
CART	Classification and Regression Trees
CLI	Command-Line Interface
CSV	Comma-separated values
DBMS	Database Management System
DQ	Data Quality
DWH	Data Warehouse
EM	Expectation-Maximization
ERP	Enterprise Resource Planning
ETL	Extract, Transform, and Load
GUI	Graphical User Interface
HDFS	Hadoop Distributed File System
IDE	Integrated development environment
KNIME	Konstanz Information Miner
KNN	K-Nearest Neighbour
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
MOA	Massive On-line Analysis
MLP	Multilayer Perceptron
NN	Neural Network
ODBC	Open Database Connectivity
OLAP	Online Analytical Processing

Continued on Next Page. . .

PMML	Predictive Model Markup Language
PNN	Probabilistic Neural Network
RAM	Random-Access Memory
RDBMS	Relational Database
RDF	Resource Description Framework File Format
SRM	Structural Risk Minimization
SSL	Secure Sockets Layer
SVM	Support Cector Machine
SQL	Structured Query Language
WEKA	Waikato Environment for Knowledge Analysis
XML	Extensible Markup Language
YCSB	Yahoo! Cloud Serving Benchmark

APPENDIX B

Listings

Listing B.1: Use case 2: Random Forest methods in R

```
1  # return the rules of a tree
2  getConds<-function(tree){
3    # store all conditions into a list
4    conds<-list()
5    # start by the terminal nodes and find previous conditions
6    id.leafs<-which(tree$status==-1)
7    j<-0
8    for(i in id.leafs){
9      j<-j+1
10     prevConds<-prevCond(tree,i)
11     conds[[j]]<-prevConds$cond
12     while(prevConds$id>1){
13       prevConds<-prevCond(tree,prevConds$id)
14       conds[[j]]<-paste(conds[[j]]," & ",prevConds$cond)
15       if(prevConds$id==1){
16         conds[[j]]<-paste(conds[[j]]," => ",tree$prediction[i])
17         break()
18       }
19     }
20   }
21   return(conds)
22 }
23 # find the previous conditions in the tree
24 prevCond<-function(tree,i){
25   if(i %in% tree$right_daughter){
26     id<-which(tree$right_daughter==i)
```

```
27      cond<-paste(tree$split_var[id],">",tree$split_point[id])
28    }
29    if(i %in% tree$left_daughter){
30      id<-which(tree$left_daughter==i)
31      cond<-paste(tree$split_var[id],"<",tree$split_point[id])
32    }
33    return(list(cond=cond,id=id))
34  }
35  # remove spaces in a word
36  collapse<-function(x){
37    x<-sub(" ","_",x)
38    return(x)
39  }
```

Bibliography

[1] Ralph Abbey Allison Ames and Wayne Thompson. Big Data Analytics Benchmarking SAS®, R, and Mahout. *SAS Institute*, 2012.

[2] Apache Mahout Web Site. http://mahout.apache.org/. Accessed: 2010-11-09.

[3] Apache Web Site. http://www.apache.org/. Accessed: 2010-11-09.

[4] Ataccama Web Site. http://www.ataccama.com/. Accessed: 2010-11-09.

[5] Jules Berman. *Principles of Big Data. Preparing, Aharing, and Analzying complex information*. Elsevier, 2013.

[6] Bigdata Startups. http://www.bigdata-startups.com/. Accessed: 2010-11-09.

[7] BigDataBench. http://prof.ict.ac.cn/bigdatabench/summary. Accessed: 2010-11-09.

[8] T. Meinl C. Sieb and M.R. Berthold. Parallel and Distributed Data Pipelining with KNIME. *Mediterranean Journal of Computers and Networks*, 2(3):43–51, 2007.

[9] Udaigiri Chandrasekhar, Amareswar Reddy, and Rohan Rath. A Comparative Study of Enterprise and Open Source Big Data Analytical Tools. *Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013)*, 2013.

[10] CRAN Web Site. http://cran.r-project.org/. Accessed: 2010-11-09.

[11] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplied Data Processing on Large Clusters. *OSDI*, 2004.

[12] Escience.washington. http://escience.washington.edu/get-help-now/what-hadoop. Accessed: 2010-11-09.

[13] Michael Berthold et al. KNIME: The Konstanz Information Miner. *Springer*, 2008.

[14] Remco Bouckaert et. al. *WEKA Manual for Version 3-7-11*, 2014.

[15] Wen Xiong et al. A Characterization of Big Data Benchmarks. *2013 IEEE International Conference on Big Data*, 2013.

[16] Alamgir Hossain Faisal Kabir, Chowdhury Mofizur and Keshav Dahal. EnhancedClassification Accuracy on Naive Bayes Data Mining Models. *International Journal of Computer Applications*, 28(3):9–16, 2011.

[17] Leo Osvald Floreal Morandat, Brandon Hill and Jan Vitek. Evaluating and Design of the R Language. *ECOOP 2012 – Object-Oriented Programming Lecture Notes in Computer Science*, 7313:104–131, 2012.

[18] Olaiya Folorunsho. Comparative Study of Different Data Mining Techniques Performance in knowledge Discovery from Medical Database. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3:11–15, 2013.

[19] Anshul Goyal and Rajni Mehta. Performance Comparison of Naïve Bayes and J48 Classification Algorithms. *International Journal of Applied Engineering Research*, 7(11), 2012.

[20] Jiawei Ham and Micheline Kamber. *Data Mining Concepts and Techiques*. Elsevier, 2. edition, 2006.

[21] Rui Han and Xiaoyi Lu. On Big Data Benchmarking. *BPOE-4*, 2014.

[22] Mosin Hasan Hardik Maniya and Komal Patel. Comparative study of Naïve Bayes Classifier and KNN for Tuberculosis. *International Conference on Web Services Computing (ICWSC)*.

[23] HiBench. https://github.com/intel-hadoop/hibench. Accessed: 2010-11-09.

[24] Eibe Frank Ian Witten and Mark Hall. *Data Mining. Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers is an imprint of Elsevier, 3. edition, 2011 (in USA).

[25] Nasir Ahmad Jehad Ali, Rehanullah Khan and Imran Maqsood. Random Forests and Decision Trees. *IJCSI International Journal of Computer Science Issues*, 9(5):272–278, 2012.

[26] Emmanuel Jeannot Jens Gustedt and Martin Quinson. Experimental Methodologies for Large-Scale Systems: a Survey of Methodologies. *Institut National de Rechrche en Imformatique et en Automatique, INRIA, N6859*, pages 399–418, 2009.

[27] Kaggle. http://www.kaggle.com/. Accessed: 2014-05-05.

[28] Kaggle Acquire Valued Shoppers Challenge. http://www.kaggle.com/c/acquire-valued-shoppers-challenge/. Accessed: 2014-05-05.

[29] Kaggle Allstate Purchase Prediction Challenge. http://www.kaggle.com/c/allstate-purchase-prediction-challenge/. Accessed: 2014-05-05.

[30] Kaggle Walmart Recruiting Store Sales Forecasting. http://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/. Accessed: 2014-05-05.

[31] KNIME. *KNIME Quickstart Guide*.

[32] KNIME Web Site. http://www.knime.org. Accessed: 2010-11-09.

[33] KNIME Web Site Server Products. http://www.knime.org/server-products. Accessed: 2010-11-09.

[34] Raj Kumar and Rajesh Verma. Classification Algorithms for Data Mining: A Survey. *International Journal of Innovations in Engineering and Technology (IJIET)*, 1(2):7–14, 2012.

[35] Krzysztof Lapinski and Bohdan Wyznikiewicz. Report. Cloud Computing: Flexibility, Efficiency, Security. *Gdansk Institute for Market economics (IBnGr), ThinkTank, Microsoft*, 2011.

[36] David Loshin. *Big Data Analytics. From Strategic Planning to Enterprise Integration with Tools, Technquies, NoSQL, and Graph*. Elsevier, 2013.

[37] Oded Maimon and Lior Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer, 2. edition, 2010.

[38] Michael Manoochehri. *Data Just Right. Introduction to Large-Scale Data and Analytics*. Addison-Wesley, 2014.

[39] O'Reilly Media. *Big Data Now: 2012 Edition*. O'Reilly Media, Inc., 2012.

[40] Bernd Wiswedel Michael Berthold and Thomas Gabriel. Fuzzy Logic in KNIME – Modules for Approximate Reasoning. *International Journal of Computational Intelligence Systems*, 6(1):34–45, 2013.

[41] Aman Bajpai Narendra Sharma and Ratnesh Litoriya. Comparison the various clustering algorithms of weka tools. *International Journal of Emerging Technology and Advanced Engineering*, 2(5):73–80, 2012.

[42] NASCCOM. Big Data The Next Big Thing. 2012.

[43] A. Nasridinov and Y.H. Park. Visual Analytics for Big Data using R. *IEEE Third International Conference on Cloud and Green Computing*, 2013.

[44] Paulraj Ponniah. *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals*. John Wiley & Sons Inc., 2001.

[45] Dsvgk Kaladhar Gr Sridhar Pv Nageswara Rao, T.Uma Devi and Allam A. Rao. A Probabilistic Neural Network Approach for Protein Superfamily Classification. *Journal of Theoretical and Applied Information Technology*, 2005-2009.

[46] R Web Site. http://adv-r.had.co.nz/. Accessed: 2010-11-09.

[47] Philip Russom. *Big Data Analytics*. TDWI Report Q4, 2011.

[48] Mohd Saboor and Rajesh Rengasamy. Designing and developing complex event processing applications. *Spaient Global Markets*, 2013.

[49] Samir K. Sarangi and Vivek Jaglan. Performance Comparison of Machine Learning Algorithms on Integration of Clustering and Classification Techniques. *International Association of Scientific Innovation and Research (IASIR)*, pages 251–257, 2013.

[50] Ted Dunning Sean Owen, Robin Anil and Ellen Friedman. *Mahout in Action*. Manning, 2012 (in USA).

[51] Aman K. Sharma and Suruchi Sahni. A Comparative Study of Classification Algorithms for Spam Email Data Analysis . *International Journal on Computer Science and Engineering (IJCSE)*, 3(5):1890–1895, 2011.

[52] Rosario Silipo and Phil Winters. *Big Data, Smart Energy, and Predictive Analytics. Time Series Prediction of Smart Energy Data*, 2013.

[53] Hitachi Data Systems. Create the Data Center of the Future. Accelerate the Value of Cloud Computing With the Right Infrastructure. *Hitachi Data Systems*, 2014.

[54] T.Balasubramanian and R.Umarani. Classification: An Analysis Technique in Data Mining for Health Hazards of High Levels of Fluoride in Potable water. *European Journal of Scientific Research*, 78(3):384–394, 2012.

[55] Weka. http://www.cs.waikato.ac.nz/ml/weka/. Accessed: 2010-11-09.

[56] Weka. Big Data. http://www.cs.waikato.ac.nz/ml/weka/bigdata.html. Accessed: 2010-11-09.

[57] Weka. Pentaho. Time Series Analysis. http://wiki.pentaho.com/display/datamining/time+series+analysis+and+forecasting+with+weka. Accessed: 2010-11-09.