

# Chapter 10

## A Data Mining Software Package Including Data Preparation and Reduction: KEEL

**Abstract** KEEL software is an open source Data Mining tool widely used in research and real life applications. Most of the algorithms described, if not all of them, throughout the book are actually implemented and publicly available in this Data Mining platform. Since KEEL enables the user to create and run single or concatenated preprocessing techniques in the data, such software is carefully introduced in this section, intuitively guiding the reader across the step needed to set up all the data preparations that might be needed. It is also interesting to note that the experimental analyses carried out in this book have been created using KEEL, allowing the consultant to quickly compare and adapt the results presented here. An extensive revision of Data Mining software tools are presented in Sect. 10.1. Among them, we will focus on the open source KEEL platform in Sect. 10.2 providing details of its main features and usage. For the practitioners interest, the most common used data sources are introduced in Sect. 10.3 and the steps needed to integrate any new algorithm in it in Sect. 10.4. Once the results have been obtained, the appropriate comparison guidelines are provided in Sect. 10.5. The most important aspects of the tool are summarized in Sect. 10.6.

### 10.1 Data Mining Softwares and Toolboxes

As we have indicated in Chap. 1, Data Mining (DM) is the process for automatic discovery of high level knowledge by obtaining information from real world, large and complex data sets [1], and is the core step of a broader process, called KDD. In addition to the DM step, the KDD process includes application of several preprocessing methods aimed at facilitating application of DM algorithms and postprocessing methods for refining and improving the discovered knowledge. The evolution of the available techniques and their wide adoption demands to gather all the steps involved in the KDD process in the least amount of pieces of software as possible for the sake of easier application and comparisons among the results obtained, yet allowing non expert practitioners to have access to KDD techniques.

Many DM software tools have been developed in the last few years due to the popularization of DM. Although a lot of them are commercially distributed (some of the leading commercial software are mining suites such as SPSS Clementine,<sup>1</sup> Oracle Data Mining<sup>2</sup> and KnowledgeSTUDIO<sup>3</sup>), only a few were available as open source. Fortunately this tendency has changed and free and open source DM tools have appeared to cover many specialized tasks in the process as well as general tools that include most of the steps of KDD. Among the latter we can highlight Weka [2], Orange [3] or Java-ML [4] as the most well-known of a growing family of open source toolboxes for DM.

Most programming languages have a DM software so any user has the possibility of performing experiments. While Weka, RapidMiner<sup>4</sup> [5], Java-ML and *αMiver* are written in Java, ADaM<sup>5</sup> and Orange are written in Python. Statistical languages also have their software tools as Rattle [6] for R.

It is also common to find libraries for some popular programming languages that can be added to a particular project. Their aim is not the novel user but an experienced practitioner who wants to add functionality to real-world cases without dealing with a multi-purpose GUI or having to rip off the methods they want. A well-known library written in C++ for fast programs is MLC++,<sup>6</sup> and R has their own statistical analysis package.<sup>7</sup> In Java the MLJ library<sup>8</sup> is available to be integrated in any project with ease.

Apart from the aforementioned toolboxes, the reader can find more alternatives to suit to their needs. Many specialized webpages are devoted to the presentation, promotion and publishing of DM news and software. We recommend visiting the KDnuggets software directory<sup>9</sup> and the-Data-Mine site.<sup>10</sup> In the research field open source tools are playing an increasingly important role as is pointed out in [7]. To this regard the link page of the Knowledge Extraction based on Evolutionary Learning (KEEL) webpage<sup>11</sup> contains an extensive list of open source DM tools and related fields such as metaheuristic optimization.

KEEL [8] is a open source Java software tool which empowers the user to assess the behavior of ML, evolutionary learning and soft computing based techniques for different kinds of DM problems: regression, classification, clustering, pattern mining and so on. This tool can offer several advantages:

---

<sup>1</sup> <http://www.spss.com/clementine>.

<sup>2</sup> <http://www.oracle.com/technology/products/bi/odm>.

<sup>3</sup> <http://www.angoss.com/products/studio/index.php>.

<sup>4</sup> <http://sourceforge.net/projects/rapidminer/>.

<sup>5</sup> <http://projects.itsc.uah.edu/datamining/adam/>.

<sup>6</sup> <http://www.sgi.com/tech/mlc/>.

<sup>7</sup> <http://www.r-project.org/>.

<sup>8</sup> <http://www.kddresearch.org/Groups/Machine-Learning/MLJ/>.

<sup>9</sup> <http://www.kdnuggets.com/software>.

<sup>10</sup> <http://the-data-mine.com/bin/view/Software>.

<sup>11</sup> <http://sci2s.ugr.es/keel/links.php>.

- *It reduces programming work.* It includes libraries of different paradigms as evolutionary learning algorithms based on different paradigms (Pittsburgh, Michigan and IRL), fuzzy learning, lazy learning, ANNs, SVMs models and many more; simplifying the integration of DM algorithms with different pre-processing techniques. It can alleviate the work of programming and enable researchers to focus on the analysis of their new learning models in comparison with the existing ones.
- *It extends the range of possible users applying ML algorithms.* An extensive library of ML techniques together with easy-to-use software considerably reduce the level of knowledge and experience required by researchers in DM. As a result researchers with less knowledge, when using this tool, would be able to successfully apply these algorithms to their problems.
- *It has an unparalleled range of preprocessing methods included for DM,* from discretization algorithms to noisy data filters. Few DM platforms offer the same amount of preprocessing techniques as KEEL does. This fact combined with a well-known data format facilitates the user to treat and include their data in the KEEL work flow and to easily prepare it to be used with their favourite techniques.
- *Cross platform compatibility.* Due to the use of a strict object-oriented approach for the library and software tool, these can be used on any machine with Java. As a result, any researcher can use KEEL on their machine, regardless of the operating system.

## 10.2 KEEL: Knowledge Extraction Based on Evolutionary Learning

KEEL<sup>12</sup> is a software tool that facilitates the analysis of the behaviour of ML in the different areas of learning and pre-processing tasks, making the management of these techniques easy for the user. The models correspond with the most well-known and employed models in each methodology, such as feature and instance selection [9, 10], decision trees [11], SVMs [12], noise filters [13], lazy learning [14], evolutionary fuzzy rule learning [15], genetic ANNs [16], Learning Classifier Systems [17], and many more.

The current available version of KEEL consists of the following function blocks<sup>13</sup>:

- *Data Management:* This part is made up of a set of tools that can be used to build new data, to export and import data in other formats to or from KEEL format, data edition and visualization, to apply transformations and partitioning to data, etc...
- *Design of Experiments (off-line module):* The aim of this part is the design of the desired experimentation over the selected data sets and providing for many options in different areas: type of validation, type of learning (classification, regression, unsupervised learning), etc...

---

<sup>12</sup> <http://keel.es>.

<sup>13</sup> <http://www.keel.es/software/prototypes/version1.0/ManualKeel.pdf>.

- *Educational Experiments (on-line module)*: With a similar structure to the aforementioned, this permits the design of experiment to be run step-by-step in order to display the learning process of a certain model by using the software tool for educational purposes.

With all of these function blocks, we can attest that KEEL can be useful by different types of users who may expect to find specific features in a DM software.

In the following subsections we describe in detail the user profiles for whom KEEL is intended, its main features and the different integrated function blocks.

### 10.2.1 Main Features

KEEL is a software tool developed to ensemble and use different DM models. Although it was initially focused on the use of evolutionary algorithms for KDD, its continuous development has broadened the available ML paradigms for DM. We would like to note that this is the first software toolkit of this type containing a library of evolutionary algorithms with open source code in Java. The main features of KEEL are:

- Almost one hundred of data preprocessing algorithms proposed in specialized literature are included: data transformation, discretization, MVs treatment, noise filtering, instance selection and FS.
- More than two hundred of state-of-the-art techniques for classification, regression, subgroup discovery, clustering and association rules, ready to be used within the platform or to be extracted and integrated in any other particular project.
- Specialized modules for recent and difficult challenges in DM such as imbalanced learning and multiple instance learning.
- Being the initial key role of KEEL, EAs are presented in predicting models, pre-processing (evolutionary feature and instance selection) and post-processing (evolutionary tuning of fuzzy rules).
- It contains a statistical library to analyze algorithm results and comprises of a set of statistical tests for analyzing the normality and heteroscedasticity of the results, as well as performing parametric and non-parametric comparisons of the algorithms.
- Some algorithms have been developed using the Java Class Library for Evolutionary Computation (JCLEC) software [18].<sup>14</sup>
- A user-friendly interface is provided, oriented towards the analysis of algorithms.
- The software is designed for experiments containing multiple data sets and algorithms connected to each other to obtain the desired result. Experiments are independently script-generated from the user interface for an off-line run in the same or other machines.
- KEEL also allows for experiments in on-line mode, intended as an educational support for learning the operation of the algorithms included.

---

<sup>14</sup> <http://jclec.sourceforge.net/>.

- It contains a Knowledge Extraction Algorithms Library<sup>15</sup> with the incorporation of multiple evolutionary learning algorithms, together with classical learning approaches. The principal families of techniques included are:
  - *Evolutionary rule learning models*. Including different paradigms of evolutionary learning.
  - *Fuzzy systems*. Fuzzy rule learning models with a good trade-off between accuracy and interpretability.
  - *Evolutionary neural networks*. Evolution and pruning in ANNs, product unit ANNs, and RBFN models.
  - *Genetic programming*. Evolutionary algorithms that use tree representations for knowledge extraction.
  - *Subgroup discovery*. Algorithms for extracting descriptive rules based on patterns subgroup discovery.
  - *Data reduction (instance and feature selection and discretization)*. EAs for data reduction.

KEEL integrates the library of algorithms in each of its function blocks. We have briefly presented its function blocks above but in the following subsections, we will describe the possibilities that KEEL offers in relation to data management, off-line experiment design and on-line educational design.

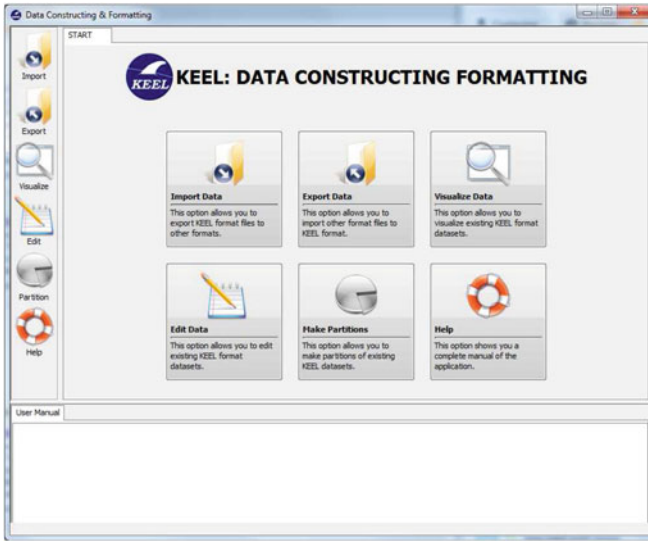
### 10.2.2 Data Management

The fundamental purpose of data preparation is to manipulate and transform raw data so that the information content enfolded in the data set can be exposed, or made more accessible [19]. Data preparation comprises of those techniques concerned with analyzing raw data so as to yield quality data, mainly including data collecting, data integration, data transformation, data cleaning, data reduction and data discretization [20]. Data preparation can be even more time consuming than DM, and can present similar challenges. Its importance lies in that the real-world data is impure (incomplete, noisy and inconsistent) and high-performance mining systems require quality data (the removal of anomalies or duplications). Quality data yields high-quality patterns (to recover missing data, purify data and resolve conflicts).

The *Data Management* module integrated in KEEL allows us to perform the data preparation stage independently of the remaining DM processes. This module is focused on the group of users denoted as domain experts. They are familiar with their data, they know the processes that produce the data and they are interested in reviewing to improve them or analyze them. On the other hand, domain users are those whose interests lies in applying processes to their own data and are usually not experts in DM.

---

<sup>15</sup> <http://www.keel.es/software/prototypes/version1.0/VAlgorithmsList.pdf>.



**Fig. 10.1** Data management

Figure 10.1 shows an example window of the *Data Management* module in the section of *Data Visualization*. The module has seven sections, each of which is accessible through the buttons on the left side of the window. In the following, we will briefly describe them:

- *Creation of a new data set*: This option allows us to generate a new data set compatible with the other KEEL modules.
- *Import data to KEEL format*: Since KEEL works with a specific data format (similar to the ARFF format) in all its modules, this section allows us to convert various data formats to KEEL format, such as CSV, XML, ARFF, extracting data from data bases, etc.
- *Export data from KEEL format*: This option is the reverse of the previous one. It converts the data handled by KEEL procedures in other external formats to establish compatibility with other software tools.
- *Visualization of data*: This option is used to represent and visualize the data. With it, we can see a graphical distribution of each attribute and comparisons between two attributes.
- *Edition of data*: This area is dedicated to managing the data manually. The data set, once loaded, can be edited by modifying values, adding or removing rows and columns, etc.
- *Data Partition*: This zone allows us to make the partitions of data needed by the experiment modules to validate results. It supports  $k$ -FCV,  $5 \times 2$ -CV and hold-out validation with stratified partition.

- *Data Preparation*: This section allows us to perform automatic data preparation for DM, including cleaning, transformation and reduction of data. All techniques integrated in this section are also available in the experiments-related modules.

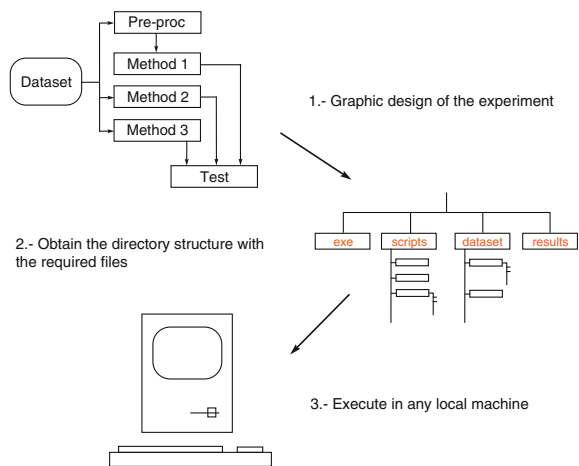
### 10.2.3 Design of Experiments: Off-Line Module

In the last few years, a large number of DM software tools have been developed for research purposes. Some of them are libraries that allow reductions in programming work when developing new algorithms: ECJ [21], JCLEC [18], learning classifier systems [22], etc. Others are DM suites that incorporate learning algorithms (some of them may use EAs for this task) and provide a mechanism to establish comparisons among them. Some examples are Weka [2], D2K [23], etc.

This module is a Graphical User Interface (GUI) that allows the design of experiments for solving various problems of regression, classification and unsupervised learning. Having designed the experiments, it generates the directory structure and files required for running them in any local machine with Java (see Fig. 10.2).

The experiments are graphically modeled, based on data flow and represented by graphs with node-edge connections. To design an experiment, we first have to indicate the type of validation ( $k$ -FCV [24] or  $5 \times 2$ -CV [25]) and the type of learning (regression, classification or unsupervised) to be used. Then, we have to select the data sources, drag the selected methods into the workspace and connect methods and data sets, combining the evolutionary learning algorithms with different pre-processing and post-processing techniques, if needed. Finally, we can add statistical tests to achieve a complete analysis of the methods being studied, and a report box to obtain a summary of the results. Notice that each component of the experiment is configured in separate dialogues that can be opened by double-clicking the respective node.

**Fig. 10.2** Design of experiments



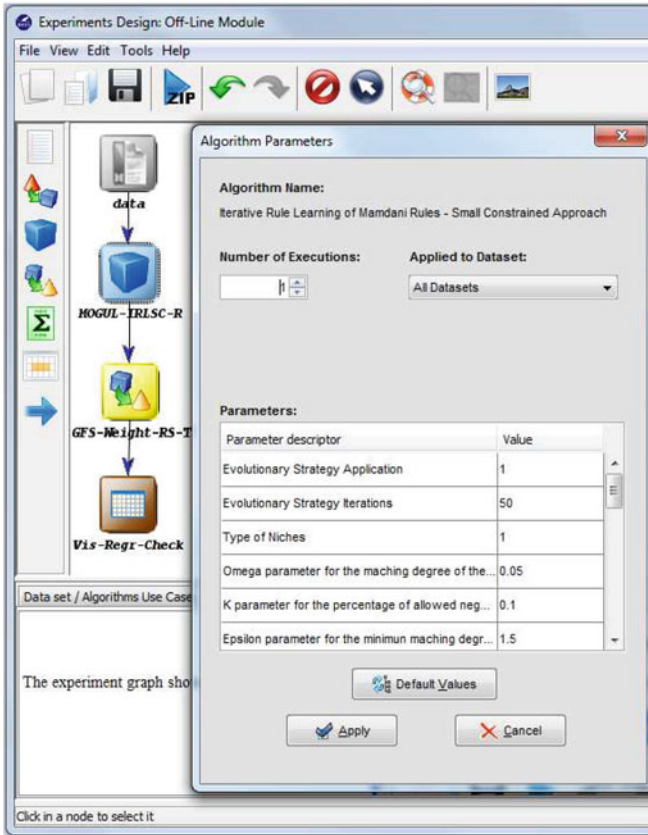


Fig. 10.3 Example of an experiment and the configuration window of a method

Figure 10.3 shows an example of an experiment following the MOGUL methodology and using a report box to obtain a summary of the results. The configuration window of one of the used post-processing methods is also shown in this figure.

When the experiment has been designed, the user can choose either to save the design in a XML file or to obtain a zip file. If the user chooses a zip file, then the system will generate the file with the directory structure and required files for running the designed experiment in any local machine with Java. This directory structure contains the data sources, the jar files of the algorithms, the configuration files in XML format, a script file with all the indicated algorithms in XML format, and a Java tool, named *RunKeel*, to run the experiment. *RunKeel* can be seen as a simple EA scripting environment that reads the script file in XML format, runs all the indicated algorithms and saves the results in one or several report files.

Obviously, this kind of interface is ideal for experts of specific areas who, familiar with the methodologies and methods used in their particular area of interest, intend to develop a new method and would like to compare it with the well-known methods available in KEEL.



### 10.2.4 Computer-Based Education: On-Line Module

There is a variety of terms used to describe the use of computers in education [26]. Computer-assisted instruction (CAI), computer-based education (CBE) and computer-based instruction (CBI) are the broadest terms and can refer to virtually any kind of computer use in educational environments. These terms may refer either to stand-alone computer learning activities or to computer activities which reinforce material introduced and taught by teachers.

Most of the software developed in DM and evolutionary computation domain is designed for research purposes (libraries, algorithms, specific applications, etc.). But there is some free software that is designed not only for research but also for educational purposes. These systems are easy to use due to the fact that they provide a GUI to assist user interaction with the system in all the tasks (selecting data, choosing parameters, running algorithms, visualize the results, etc.). Some examples of open source DM systems are Weka [2], Yale [27] and Tanagra [28].

This module is a GUI that allows the user to design an experiment (with one or more algorithms), run it and visualize the results on-line. The idea is to use this part of KEEL as a guideline to demonstrate the learning process of a certain model. This module has a similar structure to the previous one but only includes algorithms and options that are suitable for academic purposes.

When an experiment is designed the user can choose either to save the experiment in a XML file or to run it. If the user chooses to run it, then the system will show an auxiliary window to manage and visualize the execution of each algorithm. When the run finishes, this window will show the results obtained for each algorithm in separate tags, showing for example the confusion matrices for classification or the mean square errors for regression problems (see Fig. 10.4).

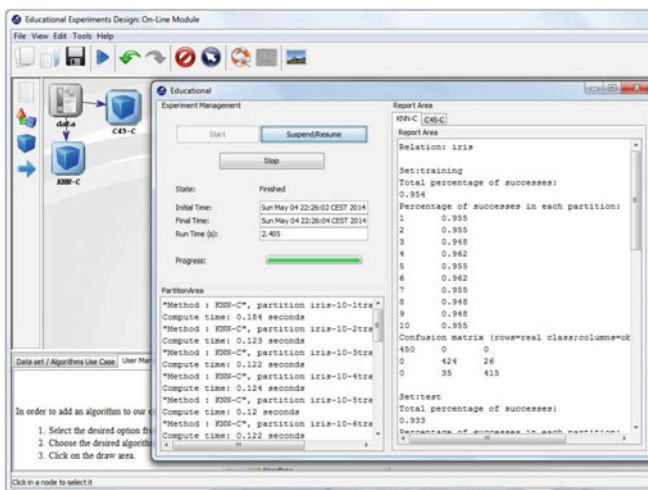


Fig. 10.4 Auxiliary window of an experiment with two algorithms

## 10.3 KEEL-Dataset

In this section we present the KEEL-dataset repository. It can be accessed through the main KEEL webpage.<sup>16</sup> The KEEL-dataset repository is devoted to the data sets in KEEL format which can be used with the software and provides:

- A detailed categorization of the considered data sets and a description of their characteristics. Tables for the data sets in each category have been also created.
- A descriptions of the papers which have used the partitions of data sets available in the KEEL-dataset repository. These descriptions include results tables, the algorithms used and additional material.

KEEL-dataset contains two main sections according to the previous two points. In the first part, the data sets of the repository are presented. They have been organized in several categories and sub-categories arranging them in tables. Each data set has a dedicated webpage in which its characteristics are presented. These webpages also provide the complete data set and the partitions ready to download.

On the other hand, the experimental studies section is a novel approach in these types of repositories. It provides a series of webpages for each experimental study with the data sets used and their results in different formats as well, ready to perform a direct comparison. Direct access to the paper's PDF for all the experimental studies included in this webpage is also provided.

In Fig. 10.5 the main webpage, in which these two main sections appear, is depicted.

In the rest of this section we will describe the two main sections of the KEEL-dataset repository webpage.

### 10.3.1 Data Sets Web Pages

The categories of the data sets have been derived from the topics addressed in the experimental studies. Some of them are usually found in the literature, like supervised (classification) data sets, unsupervised and regression problems. On the other hand, new categories which have not been tackled or separated yet are also present. The categories in which the data sets are divided are the following:

- Classification problems. This category includes all the supervised data sets. All these data sets contains one or more attributes which label the instances, mapping them into different classes. We distinguish three subcategories of classification data sets:
  - *Standard data sets.*
  - *Imbalanced data sets* [29–31]. Imbalanced data sets are standard classification data sets where the class distribution is highly skewed among the classes.

---

<sup>16</sup> <http://keel.es/datasets.php>.

## Data sets

### Classification

- Standard data sets
- Imbalanced data sets
- Multi instance data sets

### Regression

- Regression data sets

### Unsupervised (Clustering and Associations)

- Unsupervised data sets

### Low quality

- Low quality data sets
- 

## Experimental studies and results with these data sets

### Classification

- Experimental studies in supervised classification
- Experimental studies with imbalanced data sets
- Experimental studies with multi instance data sets

### Regression

- Experimental studies in regression

### Unsupervised (Clustering and Associations)

- Experimental studies in unsupervised learning

### Low quality

- Experimental studies with low quality data

Fig. 10.5 KEEL-dataset webpage (<http://keel.es/datasets.php>)

– *Multi instance data sets* [32]. Multi-Instance data sets represent problems where there is a many-to-one relationship between feature vectors and their output attribute.

- Regression problems. These are data sets with a real valued output attribute, and the objective is to better approximate this output value using the input attributes.
- Unsupervised (Clustering and Associations) problems. Unsupervised data sets represent a set of data whose examples have been not labeled.
- Low quality data [33]. In this category the data sets which contain imprecise values in their input attributes are included, caused by noise or restrictions in the measurements. Therefore these low quality data sets can contain a mixture of crisp and fuzzy values. This is a unique category.

In Fig. 10.6 the webpage for the classification standard data sets is shown as an illustrative example of a particular category webpage. These webpages are structured in two main sections:

## 1 Introduction

This section shows the classification data sets available in the repository. Every one defines a supervised classification problem, where each of its examples is composed by some nominal or numerical attributes and a nominal output attribute (its class).

Each data file has the following structure:

- **@relation:** Name of the data set
- **@attribute:** Description of an attribute (one for each attribute)
- **@inputs:** List with the names of the input attributes
- **@output:** Name of the output attribute
- **@data:** Starting tag of the data

The rest of the file contains all the examples belonging to the data set, expressed in comma separated values format.

## 1 Data sets

Below you can find all the Classification data sets available. For each data set, it is shown its name and its number of examples (instances), attributes (features,variables) and classes (number of possible values of the output variable). Also, the table shows if the corresponding data set has missing values or not. You can download each data set in KEEL format (inside a ZIP file). Additionally, it is possible to obtain the data set already partitioned, by means of a 10-folds / 5-folds cross validation procedure. Finally, we also provide a header file (in KEEL format) to define completely every attribute of the data set.

By clicking in the column headers, you can order the table by names (alphabetically) or by the number of examples, attributes or classes. Clicking again will sort the rows in reverse order.

Name ▼	#Attributes ▼	#Examples ▼	#Classes ▼	Miss Val. ▼	Data set	10-fcv	5-fcv	Header
haberman	3	306	2	No				
iris	4	150	3	No				
balance	4	625	3	No				

Fig. 10.6 Fraction of Keel-dataset standard data sets' webpage

- First, the structure of the header of this type of Keel data set file is pointed out. This description contains the tags used to identify the different attributes, the name of the data set and indicates the starting point of the data.
- The second part is a enumeration of the different data sets contained in the webpage. This enumeration is presented in a table. The table shows the characteristics of all the data sets: the name of the data set, number of attributes, number of examples and number of classes (if applicable). Moreover the possibility of downloading the entire data set or different kind of partitions in Keel format in a ZIP file is presented. A header file is also available with particular information of the data set.

The tables' columns can be also sorted attending to the different data set's characteristics, like the number of attributes or examples.

Clicking on the name of the data set in the table will open the specific webpage for this data set. This webpage is composed of tables which gather all information available on the data set.

- The first table will always contain the general information of the data set: name, number of attributes, number of instances, number of classes, presence of MVs, etc.
- The second table contains the relation of attributes of the data set. For each attribute, the domain of the values is given. If it is a numerical attribute, the minimum and maximum values of the domain are presented. In the case of nominal attributes, the complete set of values is shown. The class attribute (if applicable) is stressed with a different color.

Additional information of the data set is also included, indicating its origin, applications and nature. In a second part of the webpage, the complete data set and a number of partitions can be downloaded in Keel format.

### 10.3.2 Experimental Study Web Pages

This section contains the links to the different experimental studies for the respective data set categories. For each category, a new webpage has been built. See Fig. 10.7 for the webpage devoted to the experimental studies with standard classification data sets. These webpages contain published journal publications which use the correspondent kind of data sets in the repository. The papers are grouped by the publication year. Each paper can contain up to four links:

#### Introduction

This section shows some relevant research papers in which some of the classification data sets available in KEEL-dataset have been employed.



For each study, we provide its reference (plain text and BibTeX formats), abstract and summary. A pdf version of the article can also be downloaded. Additionally, we offer complementary material about the experimental studies carried up: Algorithms tested, data sets employed and results obtained (XLS and CVS formats).

---

#### Experimental studies and results with these data sets

**Jump to year:** 2010 (1)

**Year 2010 (1):**

	<p>A. Fernandez, S. García, J. Luengo, E. Bernadó-Mansilla, F. Herrera, Genetics-Based Machine Learning for Rule Induction: State of the Art, Taxonomy and Comparative Study. IEEE Transactions on Evolutionary Computation, in press (2010).</p>	
<p><b>Link to:</b> Data sets, algorithms and <b>Link to:</b> Website associated to this experimental results. paper.</p>		

**Fig. 10.7** Keel-dataset experimental studies with standard classification data sets webpage

- The first link is the PDF file of the paper.
- The second link is the Bibtex reference of the paper.
- The bottom-left link *Data sets, algorithms and experimental results* is always present. It references to the particular Keel-dataset webpage for the paper.
- The bottom-right link *Website associated to this paper* is only present for some papers which have a particular and external webpage related to them.

The particular Keel-dataset for the paper presents the relevant information of the publication. The abstract of the paper, an outline and the details of the experimental study are included. These details consist of the names of the algorithms analyzed, the list of data sets used and the results obtained. Both data sets used and the complete results of the paper are available for download in separate ZIP files. Moreover, the results are detailed and listed in CSV and XLS (Excel) formatted files. In Fig. 10.8 an example of the webpage for a specific publication with all these fields is shown.

## 10.4 Integration of New Algorithms into the KEEL Tool


In this section the main features that any researcher must take into account to integrate a new algorithm into the KEEL software tool are described. Next, a simple codification example is provided in order to clarify the integration process.

### 10.4.1 Introduction to the KEEL Codification Features


This section is devoted to describing in detail how to implement or to import an algorithm into the KEEL software tool. The KEEL philosophy tries to include the least possible constraints for the developer, in order to ease the inclusion of new algorithms. Thus, it is not necessary to follow the guidelines of any design pattern or framework in the development of a new method. In fact, each algorithm has its source code in a single folder and does not depend on a specific structure of classes, making the integration of new methods straightforward.

We enumerate the list of details to take into account before codifying a method for the KEEL software, which is also detailed at the KEEL Reference Manual (<http://www.keel.es/documents/KeelReferenceManualV1.0.pdf>).

- The programming language used is Java.
- In KEEL, every method uses a configuration file to extract the values of the parameters which will be employed during its execution. Although it is generated automatically by the KEEL GUI (by using the information contained in the corresponding method description file, and the values of the parameters specified by the user), it is important to fully describe its structure because any KEEL method must be able to read it completely, in order to get the values of its parameters specified in each execution.



A. Fernandez, S. Garcia, J. Luengo, E. Bernadó-Mansilla, F. Herrera. Genetics-Based Machine Learning for Rule Induction: State of the Art, Taxonomy and Comparative Study. IEEE Transactions on Evolutionary Computation, in press (2010).




---

**This webpage contains:**

- Abstract
- Summary
- Experimental study

**Additional links:**

- External website associated to this paper.

---

**Abstract:**

The classification problem can be addressed by numerous techniques and algorithms, which belong to different paradigms of Machine Learning. In this work, we are interested in evolutionary algorithms, the so-called Genetics-Based Machine Learning algorithms. In particular, we will focus on evolutionary approaches that evolve a set of rules, i.e., evolutionary rule-based systems, applied to classification tasks, in order to provide a state-of-the-art in this field.

This study has been done with a double aim: to present a taxonomy of the Genetics-Based Machine Learning approaches for rule induction, and to develop an empirical analysis both for standard classification and for classification with imbalanced data sets.

We also include a comparative study of the GBML methods with some classical non-evolutionary algorithms, in order to observe the suitability and high power of the search performed by evolutionary algorithms and the behaviour for the GBML algorithms in contrast to the classical approaches, in terms of classification accuracy.

---

**Summary:**

1. Introduction
2. Taxonomy of genetics-based machine learning algorithms for classification
3. Experimental framework
4. Analysis of the GBML algorithms for rule induction in standard classification
5. Analysis of the GBML algorithms for rule induction in imbalanced data sets
6. Discussion: Lessons learned and new challenges
7. Concluding remarks

---

**Experimental study:**













- **Algorithms analyzed:** XCS, UCS, SIA, HIDER, CORE, OCEC, COGIN, GIL, Pitts-GIRLA, DMEL, GASSIST, OIGA, ILGA, DT-GA, Oblique-DT, TARGET, CART, AQ, CN2, C4.5, C4.5-Rules, Ripper.
- **Data sets used:** ZIP file 
  - **Standard:** [5-fcv] abalone, australian, balance, breast, bupa, car, cleveland, contraceptive, crx, dermatology, ecoli, flare, german, glass, haberman, heart, hepatitis, iris, lymphography, magic, new-thyroid, nursery, penbased, pima, ring, tic-tac-toe, vehicle, wisconsin, zoo.
  - **Imbalanced:** [5-fcv] glass1, ecoli0vs1, wisconsin, pima, iris0, glass0, yeast1, vehicle1, vehicle2, vehicle3, haberman, glass0123vs456, vehicle0, ecoli1, new-thyroid2, new-thyroid1, ecoli2, segment0, glass6, yeast3, ecoli3, page-blocks0, vowel0, glass2, ecoli4, glass4, abalone9vs18, glass5, yeast2vs8, yeast4, yeast5, yeast6, abalone19.
- **Results obtained:** ZIP file 
  -  XLS file  CSV file - Standard results
  -  XLS file  CSV file - Standard results (non-evolutionary)
  -  XLS file  CSV file - Imbalanced results
  -  XLS file  CSV file - Imbalanced results (non-evolutionary)
  -  XLS file  CSV file - Imbalanced results (SMOTE)
  -  XLS file  CSV file - Imbalanced results (non-evolutionary) (SMOTE)
  -  XLS file  CSV file - Kappa results
  -  XLS file  CSV file - Kappa results (non-evolutionary)

Fig. 10.8 Keel-dataset example of an experimental study dedicated webpage

Each configuration file has the following structure:

- *algorithm*: Name of the method.
- *inputData*: A list of the input data files of the method.
- *outputData*: A list of the output data files of the method.
- *parameters*: A list of parameters of the method, containing the name of each parameter and its value (one line is used for each one).

Next we show a valid example of a Method Configuration file (data files lists are not fully shown):

```
algorithm = Genetic Algorithm
inputData = ``../datasets/iris/iris.dat`` ...
outputData = ``../results/iris/result0.tra`` ...
```

```
Seed = 12345678
Number of Generations = 1000
Crossover Probability = 0.9
Mutation Probability = 0.1
...
```

A complete description of the parameters file can be found in Sect. 3 of the KEEL Manual.

- The input data sets follow a specific format that extends the “arff” files by completing the header with more metadata information about the attributes of the problem. Next, the list of examples is included, which is given in rows with the attribute values separated by commas.

For more information about the input data sets files please refer to Sect. 4 of the KEEL Manual. Furthermore, in order to ease the data management, we have developed an API data set, the main features of which are described in Sect. 7 of the Manual.

- The output format consists of a header, which follows the same scheme as the input data, and two columns with the output values for each example separated by a whitespace. The first value corresponds to the expected output, and the second one to the predicted value. All methods must generate two output files: one for training and another one for testing.

For more information about the obligatory output files please refer to Sect. 5 of the KEEL Manual.

Although the list of constraints is short, the KEEL development team have created a simple template that manages all these features. Our KEEL template includes four classes:

1. **Main:** This class contains the main instructions for launching the algorithm. It reads the parameters from the file and builds the “algorithm object”.

```
public class Main {
    private parseParameters parameters;

    private void execute(String confFile) {
        parameters = new parseParameters();
        parameters.parseConfigurationFile(confFile);
        Algorithm method = new Algorithm(parameters);
        method.execute();
    }
}
```



```

public static void main(String args[]) {
    Main program = new Main();
    System.out.println("Executing Algorithm.");
    program.execute(args[0]);
}
}

```

2. **ParseParameters:** This class manages all the parameters, from the input and output files, to every single parameter stored in the parameters file.

```

public class parseParameters {

    private String algorithmName;
    private String trainingFile, validationFile, testFile;
    private ArrayList <String> inputFiles;
    private String outputTrFile, outputTstFile;
    private ArrayList <String> outputFiles;
    private ArrayList <String> parameters;

    public parseParameters() {
        inputFiles = new ArrayList<String>();
        outputFiles = new ArrayList<String>();
        parameters = new ArrayList<String>();
    }

    public void parseConfigurationFile(String fileName) {
        StringTokenizer line;
        String file = Files.readFile(fileName);

        line = new StringTokenizer(file, "\n\r");
        readName(line);
        readInputFiles(line);
        readOutputFiles(line);
        readAllParameters(line);
    };

    ...
}

```

3. **myDataset:** This class is an interface between the classes of the API data set and the algorithm. It contains the basic options related to data access.

```

public class myDataset {

    private double[][] X;
    private double[] outputReal;
    private String[] output;

    private int nData;
    private int nVars;
    private int nInputs;

    private InstanceSet IS;

    public myDataset() {
        IS = new InstanceSet();
    }

    public double[] getExample(int pos) {
        return X[pos];
    }
}

```

```

    }

    public void readClassificationSet(String datasetFile,
        boolean train) throws IOException {
        try {
            IS.readSet(datasetFile, train);
            nData = IS.getNumInstances();
            nInputs = Attributes.getInputNumAttributes();
            nVars = nInputs + Attributes.getOutputNumAttributes();

            ...
        }
    }
}

```

4. **Algorithm:** This class is devoted to storing the main variables of the algorithm and naming the different procedures for the learning stage. It also contains the functions for writing the obligatory output files.

```

public class Algorithm {

    myDataset train, val, test;
    String outputTr, outputTst;
    private boolean somethingWrong = false;

    public Algorithm(parseParameters parameters) {

        train = new myDataset();
        val = new myDataset();
        test = new myDataset();
        try {
            System.out.println("\nReading the training set:" +
                parameters.getTrainingInputFile());
            train.readClassificationSet(parameters.getTrainingInputFile(),
                true);
            System.out.println("\nReading the validation set:" +
                parameters.getValidationInputFile());
            val.readClassificationSet(parameters.getValidationInputFile(),
                false);
            System.out.println("\nReading the test set:" +
                parameters.getTestInputFile());
            test.readClassificationSet(parameters.getTestInputFile(),
                false);
        } catch (IOException e) {
            System.err.println("There was a problem while reading
                the input data sets:" + e);
            somethingWrong = true;
        }

        outputTr = parameters.getTrainingOutputFile();

        ...
    }
}

```

The template can be downloaded by clicking on the link [http://www.keel.es/software/KEEL\\_template.zip](http://www.keel.es/software/KEEL_template.zip), which additionally supplies the user with the whole API data set together with the classes for managing files and the random number generator.

Most of the functions of the classes presented above are self-explanatory and fully documented to help the developer understand their use. Nevertheless, in the

next section we will explain in detail how to encode a simple algorithm within the KEEL software tool.

## 10.5 KEEL Statistical Tests

Nowadays, the use of statistical tests to improve the evaluation process of the performance of a new method has become a widespread technique in the field of DM [34–36]. Usually, they are employed inside the framework of any experimental analysis to decide when an algorithm is better than other one. This task, which may not be trivial, has become necessary to confirm when a new proposed method offers a significant improvement over the existing methods for a given problem.

Two kinds of tests exist: parametric and non-parametric, depending on the concrete type of data employed. As a general rule, a non-parametric test is less restrictive than a parametric one, although it is less robust than a parametric when data is well conditioned.

Parametric tests have been commonly used in the analysis of experiments in DM. For example, a common way to test whether the difference between the results of two algorithms is non-random is to compute a paired t-test, which checks whether the average difference in their performance over the data sets is significantly different from zero. When comparing a set of multiple algorithms, the common statistical method for testing the differences between more than two related sample means is the repeated-measures ANOVA (or within-subjects ANOVA) [37]. Unfortunately, parametric tests are based on assumptions which are most probably violated when analyzing the performance of computational intelligence and DM algorithms [38–40]. These assumptions are known as independence, normality and homoscedasticity.

Nonparametric tests can be employed in the analysis of experiments, providing the researcher with a practical tool to use when the previous assumptions can not be satisfied. Although they are originally designed for dealing with nominal or ordinal data, it is possible to conduct ranking based transformations to adjust the input data to the test requirements. Several nonparametric methods for pairwise and multiple comparison are available to contrast adequately the results obtained in any Computational Intelligence experiment. A wide description about the topic with examples, cases of studies, bibliographic recommendations can be found in the SCI2S thematic public website on *Statistical Inference in Computational Intelligence and Data Mining*.<sup>17</sup>

KEEL is one of the fewest DM software tools that provides the researcher with a complete set of statistical procedures for pairwise and multiple comparisons. Inside the KEEL environment, several parametric and non-parametric procedures have been coded, which should help to contrast the results obtained in any experiment performed with the software tool. These tests follow the same methodology that the rest of elements of KEEL, facilitating both its employment and its integration inside a complete experimental study.

---

<sup>17</sup> <http://sci2s.ugr.es/sicidm/>.

**Table 10.1** Statistical procedures available in KEEL

Procedure	References	Description
5x2cv-f test	[25]	Approximate f statistical test for 5x2-CV
T test	[41]	Statistical test based on the Student's t distribution
F test	[42]	Statistical test based on the Snedecor's F distribution
Shapiro-Wilk test	[43]	Variance test for normality
Mann-Whitney U test	[44]	U statistical test of difference of means
Wilcoxon test	[45]	Nonparametric pairwise statistical test
Friedman test	[46]	Nonparametric multiple comparisons statistical test
Iman-Davenport test	[47]	Derivation from the Friedman's statistic (less conservative)
Bonferroni-Dunn test	[48]	Post-Hoc procedure similar to Dunnet's test for ANOVA
Holm test	[49]	Post-Hoc sequential procedure (most significant first)
Hochberg test	[50]	Post-Hoc sequential procedure (less significant first)
Nemenyi test	[51]	Comparison with all possible pairs
Hommel test	[52]	Comparison with all possible pairs (less conservative)

Table 10.1 shows the procedures existing in the KEEL statistical package. For each test, a reference and a brief description is given (an extended description can be found in the *Statistical Inference in Computational Intelligence and Data Mining* website and in the KEEL website<sup>18</sup>).

### 10.5.1 Case Study

In this section, we present a case study as an example of the functionality and process of creating an experiment with the KEEL software tool. This experimental study is focused on the comparison between the new algorithm imported (SGERD) and several evolutionary rule-based algorithms, and employs a set of supervised classification domains available in KEEL-dataset. Several statistical procedures available in the KEEL software tool will be employed to contrast the results obtained.

#### 10.5.1.1 Algorithms and Classification Problems

Five representative evolutionary rule learning methods have been selected to carry out the experimental study: Ant-Miner, CO-Evolutionary Rule Extractor (CORE), Hierarchical DEcision Rules (HIDER), Steady-State Genetic Algorithm for Extracting Fuzzy Classification Rules From Data (SGERD) and Tree Analysis with Randomly Generated and Evolved Trees (TARGET) methodology. Table 10.2 shows their references and gives a brief description of each one.

<sup>18</sup> <http://www.keel.es>.

**Table 10.2** Algorithms tested in the experimental study

Method	Reference	Description
Ant-Miner	[53]	An Ant Colony System based using a heuristic function based in the entropy measure for each attribute-value
CORE	[54]	A coevolutionary method which employs as fitness measure a combination of the true positive rate and the false positive rate
HIDER	[55]	A method which iteratively creates rules that cover
SGERD	[56]	Randomly selected examples of the training set A steady-state GA which generates a prespecified number of rules per class following a GCCL approach
TARGET	[57]	A GA where each chromosome represents a complete decision tree

On the other hand, we have used 24 well-known classification data sets (they are publicly available on the KEEL-dataset repository web page,<sup>19</sup> including general information about them, partitions and so on) in order to check the performance of these methods. Table 10.3 shows their main characteristics where *#Ats* is the number of attributes, *#Ins* is the number of instances and *#Cla* is the number of Classes. For each data set the number of examples, attributes and classes of the problem described are shown. We have employed a 10-FCV procedure as a validation scheme to perform the experiments.

**Table 10.3** Data sets employed in the experimental study

Name	#Ats	#Ins	#Cla	Name	#Ats	#Ins	#Cla
HAB	3	306	2	Wisconsin	9	699	2
IRI	4	150	3	Tic-tac-toe	9	958	2
BAL	4	625	3	Wine	13	178	3
NTH	5	215	3	Cleveland	13	303	5
MAM	5	961	2	Housevotes	16	435	2
BUP	6	345	2	Lymphography	18	148	4
MON	6	432	2	Vehicle	18	846	4
CAR	6	1,728	4	Bands	19	539	2
ECO	7	336	8	German	20	1,000	2
LED	7	500	10	Automobile	25	205	6
PIM	8	768	2	Dermatology	34	366	6
GLA	9	214	7	Sonar	60	208	2

<sup>19</sup> <http://www.keel.es/datasets.php>.

### 10.5.1.2 Setting up the Experiment Under KEEL Software

To do this experiment in KEEL, first of all we click on the Experiment option in the main menu of the KEEL software tool, define the experiment as a Classification problem and use a 10-FCV procedure to analyze the results. Next, the first step of the experiment graph setup is to choose the data sets to be used in Table 10.3. The partitions in KEEL are static, meaning that further experiments carried out will stop being dependent on particular data partitions.

The graph in Fig. 10.9 represents the flow of data and results from the algorithms and statistical techniques. A node can represent an initial data flow (group of data sets), a pre-process/post-process algorithm, a learning method, test or a visualization of results module. They can be distinguished easily by the color of the node. All their parameters can be adjusted by clicking twice on the node. Notice that KEEL incorporates the option of configuring the number of runs for each probabilistic algorithm, including this option in the configuration dialog of each node (3 in this case study). Table 10.4 shows the parameter's values selected for the algorithms employed in this experiment (they have been taken from their respective papers following the indications given by the authors).

The methods present in the graph are connected by directed edges, which represent a relationship between them (data or results interchange). When the data is interchanged, the flow includes pairs of train-test data sets. Thus, the graph in this specific example describes a flow of data from the 24 data sets to the nodes of the

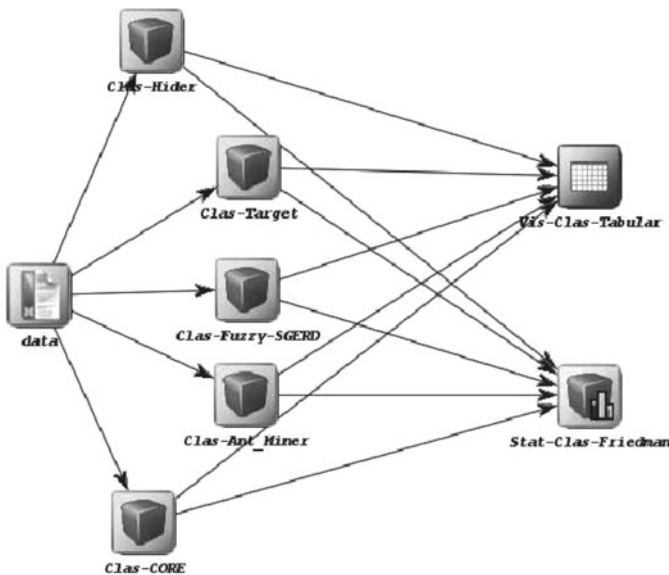


Fig. 10.9 Graphical representation of the experiment in KEEL

**Table 10.4** Parameter' values employed in the experimental study

Algorithm	Parameters
Ant-Miner	Number of ants: 3000, Maximum uncovered samples: 10, Maximum samples by rule: 10
	Maximum iterations without converge: 10
CORE	Population size: 100, Co-population size: 50, Generation limit: 100
	Number of co-populations: 15, Crossover rate: 1.0
	Mutation probability: 0.1, Regeneration probability: 0.5
HIDER	Population size: 100, Number of generations: 100, Mutation probability: 0.5
	Cross percent: 80, Extreme mutation probability: 0.05, Prune examples factor: 0.05
	Penalty factor: 1, Error coefficient: 1
SGERD	Number of Q rules per class: Computed heuristically, Rule evaluation criteria = 2
TARGET	Probability of splitting a node: 0.5, Number of total generations for the GA: 100
	Number of trees generated by crossover: 30, Number of trees generated by mutation: 10
	Number of trees generated by clonation: 5, Number of trees Generated by immigration: 5

five learning methods used (Clas-AntMiner, Clas-SGERD, Clas-Target, Clas-Hider and Clas-CORE).

After the models are trained, the instances of the data set are classified. These results are the inputs for the visualization and test modules. The module Vis-Clas-Tabular receives these results as input and generates output files with several performance metrics computed from them, such as confusion matrices for each method, accuracy and error percentages for each method, fold and class, and a final summary of results. Figure 10.9 also shows another type of results flow, the node Stat-Clas-Friedman which represents the statistical comparison, results are collected and a statistical analysis over multiple data sets is performed by following the indications given in [38].

Once the graph is defined, we can set up the associated experiment and save it as a zip file for an off-line run. Thus, the experiment is set up as a set of XML scripts and a JAR program for running it. Within the results directory, there will be directories used for housing the results of each method during the run. For example, the files allocated in the directory associated to an interval learning algorithm will contain the knowledge or rule base. In the case of a visualization procedure, its directory will house the results files. The results obtained by the analyzed methods are shown in the next section, together with the statistical analysis.

### 10.5.1.3 Results and Analysis

This subsection describes and discusses the results obtained from the previous experiment configuration. Tables 10.5 and 10.6 show the results obtained in training and test stages, respectively. For each data set, the average and standard deviations in accuracy obtained by the module Vis-Clas-Tabular are shown, with the best results stressed in **boldface**.

Focusing on the test results, the average accuracy obtained by Hider is the highest one. However, this estimator does not reflect whether or not the differences among the methods are significant. For this reason, we have carried out an statistical analysis based on multiple comparison procedures (see <http://sci2s.ugr.es/sicidm/> for a full

**Table 10.5** Average results and standard deviations of training accuracy obtained

Data set	Ant Miner		CORE		HIDER		SGERD		TARGET	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
HAB	<b>79.55</b>	1.80	76.32	1.01	76.58	1.21	74.29	0.81	74.57	1.01
IRI	97.26	0.74	95.48	1.42	<b>97.48</b>	0.36	97.33	0.36	93.50	2.42
BAL	73.65	3.38	68.64	2.57	75.86	0.40	76.96	2.27	<b>77.29</b>	1.57
NTH	<b>99.17</b>	0.58	92.66	1.19	95.97	0.83	90.23	0.87	88.05	2.19
MAM	81.03	1.13	79.04	0.65	<b>83.60</b>	0.75	74.40	1.43	79.91	0.65
BUP	<b>80.38</b>	3.25	61.93	0.89	73.37	2.70	59.13	0.68	68.86	0.89
MON	97.22	0.30	87.72	7.90	97.22	0.30	80.56	0.45	<b>97.98</b>	7.90
CAR	77.95	1.82	<b>79.22</b>	1.29	70.02	0.02	67.19	0.08	77.82	0.29
ECO	87.90	1.27	67.03	3.69	<b>88.59</b>	1.77	73.02	0.86	66.22	4.69
LED	59.42	1.37	28.76	2.55	<b>77.64</b>	0.42	40.22	5.88	34.24	3.55
PIM	71.86	2.84	72.66	2.62	<b>77.82</b>	1.16	73.71	0.40	73.42	2.62
GLA	81.48	6.59	54.26	1.90	<b>90.09</b>	1.64	53.84	2.96	45.07	0.90
WIS	92.58	1.65	94.71	0.64	<b>97.30</b>	0.31	93.00	0.85	96.13	0.64
TAE	69.62	2.21	69.46	1.20	69.94	0.53	69.94	0.53	<b>69.96</b>	2.20
WIN	<b>99.69</b>	0.58	99.06	0.42	97.19	0.98	91.76	1.31	85.19	1.58
CLE	60.25	1.35	56.30	1.97	<b>82.04</b>	1.75	46.62	2.23	55.79	2.97
HOU	94.28	1.84	<b>96.98</b>	0.43	<b>96.98</b>	0.43	<b>96.98</b>	0.43	<b>96.98</b>	0.43
LYM	77.11	5.07	65.99	5.43	<b>83.70</b>	2.52	77.48	3.55	75.84	4.43
VEH	59.52	3.37	36.49	3.52	<b>84.21</b>	1.71	51.47	1.19	51.64	2.52
BAN	67.61	3.21	66.71	2.01	<b>87.13</b>	2.15	63.84	0.74	71.14	2.01
GER	71.14	1.19	70.60	0.63	<b>73.54</b>	0.58	67.07	0.81	70.00	1.37
AUT	69.03	8.21	31.42	7.12	<b>96.58</b>	0.64	52.56	1.67	45.66	6.12
DER	86.18	5.69	31.01	0.19	<b>94.91</b>	1.40	72.69	1.04	66.24	1.81
SON	74.68	0.79	53.37	0.18	<b>98.29</b>	0.40	75.69	1.47	76.87	1.18
Average	79.52	2.51	68.16	2.14	<b>86.09</b>	1.04	71.76	1.37	72.43	2.33



**Table 10.6** Average results and standard deviations of test accuracy obtained

Data set	Ant Miner		CORE		HIDER		SGERD		TARGET	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
HAB	72.55	5.27	72.87	4.16	<b>75.15</b>	4.45	74.16	2.48	71.50	2.52
IRI	96.00	3.27	92.67	4.67	<b>96.67</b>	3.33	<b>96.67</b>	3.33	92.93	4.33
BAL	70.24	6.21	70.08	7.11	69.60	3.77	75.19	6.27	<b>75.62</b>	7.27
NTH	<b>90.76</b>	6.85	<b>90.76</b>	5.00	90.28	7.30	88.44	6.83	86.79	5.83
MAM	81.48	7.38	77.33	3.55	<b>82.30</b>	6.50	74.11	5.11	79.65	2.11
BUP	57.25	7.71	61.97	4.77	65.83	10.04	57.89	3.41	<b>65.97</b>	1.41
MON	<b>97.27</b>	2.65	88.32	8.60	<b>97.27</b>	2.65	80.65	4.15	96.79	5.15
CAR	77.26	2.59	<b>79.40</b>	3.04	70.02	0.16	67.19	0.70	77.71	2.70
ECO	58.58	9.13	64.58	4.28	<b>75.88</b>	6.33	72.08	7.29	65.49	4.29
LED	55.32	4.13	27.40	4.00	<b>68.20</b>	3.28	40.00	6.75	32.64	6.75
PIM	66.28	4.26	73.06	6.03	73.18	6.19	<b>73.71</b>	3.61	73.02	6.61
GLA	53.74	12.92	45.74	9.36	<b>64.35</b>	12.20	48.33	5.37	44.11	5.37
WIS	90.41	2.56	92.38	2.31	<b>96.05</b>	2.76	92.71	3.82	95.75	0.82
TAE	64.61	5.63	<b>70.35</b>	3.77	69.93	4.73	69.93	4.73	69.50	2.73
WIN	92.06	6.37	<b>94.87</b>	4.79	82.61	6.25	87.09	6.57	82.24	7.57
CLE	<b>57.45</b>	5.19	53.59	7.06	55.86	5.52	44.15	4.84	52.99	1.84
HOU	93.56	3.69	<b>97.02</b>	3.59	<b>97.02</b>	3.59	<b>97.02</b>	3.59	96.99	0.59
LYM	73.06	10.98	65.07	15.38	72.45	10.70	72.96	13.59	<b>75.17</b>	10.59
VEH	53.07	4.60	36.41	3.37	<b>63.12</b>	4.48	51.19	4.85	49.81	5.85
BAN	59.18	6.58	64.23	4.23	62.15	8.51	62.71	4.17	<b>67.32</b>	6.17
GER	66.90	3.96	69.30	1.55	<b>70.40</b>	4.29	66.70	1.49	70.00	0.49
AUT	53.74	7.79	32.91	6.10	<b>62.59</b>	13.84	50.67	10.27	42.82	13.27
DER	81.16	7.78	31.03	1.78	<b>87.45</b>	3.26	69.52	4.25	66.15	4.25
SON	71.28	5.67	53.38	1.62	52.90	2.37	73.45	7.34	<b>74.56</b>	8.34
Average	72.22	5.97	66.86	5.01	<b>75.05</b>	5.69	70.27	5.20	71.06	4.87

description), by including a node called Stat-Clas-Friedman in the KEEL experiment. Here, we include the information provided by this statistical module:

- Table 10.7 shows the obtained average rankings across all data sets following the Friedman procedure for each method. They will be useful to calculate the  $p$ -value and to detect significant differences between the two methods.
- Table 10.8 depicts the results obtained from the use of the Friedman and Iman-Davenport test. Both, the statistics and  $p$ -values are shown. As we can see, a level of significance  $\alpha = 0.10$  is needed in order to consider that differences among the methods exist. Note also that the  $p$ -value obtained by the Iman-Davenport test is lower than that obtained by Friedman, this is always true.

**Table 10.7** Average rankings of the algorithms by Friedman procedure

Algorithm	Ranking
AntMiner	3.125
CORE	3.396
Hider	<b>2.188</b>
SGERD	3.125
Target	3.167

**Table 10.8** Results of the Friedman and Iman-Davenport tests

Friedman value	$p$ -value	Iman-Davenport value	$p$ -value
8.408	0.0777	2.208	0.0742

**Table 10.9** Adjusted  $p$ -values. Hider is the control algorithm

I	Algorithm	Unadjusted $p$	$p_{Holm}$	$p_{Hoch}$
1	CORE	0.00811	0.032452	0.03245
2	Target	0.03193	0.09580	0.03998
3	AntMiner	0.03998	0.09580	0.03998
4	SGERD	0.03998	0.09580	0.03998

- Finally, in Table 10.9 the adjusted  $p$ -values are shown considering the best method (Hider) as the control algorithm and using the three post-hoc procedures explained above. The following analysis can be made:
  - The procedure of Holm verifies that Hider is the best method with  $\alpha = 0.10$ , but it only outperforms CORE considering  $\alpha = 0.05$ .
  - The procedure of Hochberg checks the supremacy of Hider with  $\alpha = 0.05$ . In this case study, we can see that the Hochberg method is the one with the highest power.

## 10.6 Summarizing Comments

In this chapter we have introduced a series of non-commercial Java software tools, and focused on a particular one named KEEL, that provides a platform for the analysis of ML methods applied to DM problems. This tool relieves researchers of much technical work and allows them to focus on the analysis of their new learning models in comparison with the existing ones. Moreover, the tool enables researchers with little knowledge of evolutionary computation methods to apply evolutionary learning algorithms to their work.

We have shown the main features of this software tool and we have distinguished three main parts: a module for data management, a module for designing experiments with evolutionary learning algorithms, and a module educational goals. We have also shown some case studies to illustrate functionalities and the experiment set up processes.

Apart from the presentation of the main software tool, three other complementary aspects of KEEL have been also described:

- KEEL-dataset, a data set repository that includes the data set partitions in the KEEL format and shows some results obtained in these data sets. This repository can free researchers from merely “technical work” and facilitate the comparison of their models with the existing ones.
- Some basic guidelines that the developer may take into account to facilitate the implementation and integration of new approaches within the KEEL software tool. We have shown the simplicity of adding a simple algorithm (SGERD in this case) into the KEEL software with the aid of a Java template specifically designed for this purpose. In this manner, the developer only has to focus on the inner functions of their algorithm itself and not on the specific requirements of the KEEL tool.
- A module of statistical procedures which let researchers contrast the results obtained in any experimental study using statistical tests. This task, which may not be trivial, has become necessary to confirm when a new proposed method offers a significant improvement over the existing methods for a given problem.

## References

1. Han, J., Kamber, M., Pei, J.: Data mining: Concepts and techniques, second edition (The Morgan Kaufmann series in data management systems). Morgan Kaufmann, San Francisco (2006)
2. Witten, I.H., Frank, E.: Data mining: practical machine learning tools and techniques, second edition (Morgan Kaufmann series in data management systems). Morgan Kaufmann Publishers Inc., San Francisco (2005)
3. Demšar, J., Curk, T., Erjavec, A., Gorup, Črt, Hočvar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., Zupan, B.: Orange: Data mining toolbox in python. *J. Mach. Learn. Res.* **14**, 2349–2353 (2013)
4. Abeel, T., de Peer, Y.V., Saeys, Y.: Java-ML: A machine learning library. *J. Mach. Learn. Res.* **10**, 931–934 (2009)
5. Hofmann, M., Klinkenberg, R.: RapidMiner: Data mining use cases and business analytics applications. Chapman and Hall/CRC, Florida (2013)
6. Williams, G.J.: Data mining with rattle and R: The art of excavating data for knowledge discovery. Use R!. Springer, New York (2011)
7. Sonnenburg, S., Braun, M., Ong, C., Bengio, S., Bottou, L., Holmes, G., LeCun, Y., Müller, K.R., Pereira, F., Rasmussen, C., Rätsch, G., Schölkopf, B., Smola, A., Vincent, P., Weston, J., Williamson, R.: The need for open source software in machine learning. *J. Mach. Learn. Res.* **8**, 2443–2466 (2007)
8. Alcalá-Fdez, J., Sánchez, L., García, S., del Jesus, M., Ventura, S., Garrell, J., Otero, J., Romero, C., Bacardit, J., Rivas, V., Fernández, J., Herrera, F.: KEEL: A software tool to assess evolutionary algorithms to data mining problems. *Soft Comput.* **13**(3), 307–318 (2009)
9. Derrac, J., García, S., Herrera, F.: A survey on evolutionary instance selection and generation. *Int. J. Appl. Metaheuristic Comput.* **1**(1), 60–92 (2010)
10. Kudo, M., Sklansky, J.: Comparison of algorithms that select features for pattern classifiers. *Pattern Recognit.* **33**(1), 25–41 (2000)
11. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers, San Francisco (1993)

12. Schölkopf, B., Smola, A.J.: Learning with kernels : support vector machines, regularization, optimization, and beyond. Adaptive computation and machine learning. MIT Press, Cambridge (2002)
13. Frenay, B., Verleysen, M.: Classification in the presence of label noise: A survey. *Neural Netw. Learn. Syst., IEEE Trans.* **25**(5), 845–869 (2014)
14. García, E.K., Feldman, S., Gupta, M.R., Srivastava, S.: Completely lazy learning. *IEEE Trans. Knowl. Data Eng.* **22**(9), 1274–1285 (2010)
15. Alcalá, R., Alcalá-Fdez, J., Casillas, J., Cordon, O., Herrera, F.: Hybrid learning models to get the interpretability-accuracy trade-off in fuzzy modeling. *Soft Comput.* **10**(9), 717–734 (2006)
16. Rivas, A.J.R., Rojas, I., Ortega, J., del Jesús, M.J.: A new hybrid methodology for cooperative-coevolutionary optimization of radial basis function networks. *Soft Comput.* **11**(7), 655–668 (2007)
17. Bernadó-Mansilla, E., Ho, T.K.: Domain of competence of xcs classifier system in complexity measurement space. *IEEE Trans. Evol. Comput.* **9**(1), 82–104 (2005)
18. Ventura, S., Romero, C., Zafra, A., Delgado, J.A., Hervas, C.: Jclec: A java framework for evolutionary computation. *Soft Comput.* **12**(4), 381–392 (2007)
19. Pyle, D.: Data preparation for data mining. Morgan Kaufmann Publishers Inc., San Francisco (1999)
20. Zhang, S., Zhang, C., Yang, Q.: Data preparation for data mining. *Appl. Artif. Intel.* **17**(5–6), 375–381 (2003)
21. Luke, S., Panait, L., Balan, G., Paus, S., Skolicki, Z., Bassett, J., Hubley, R., Chircop, A.: ECJ: A Java based evolutionary computation research system. <http://cs.gmu.edu/eclab/projects/ecj>
22. Meyer, M., Hufschlag, K.: A generic approach to an object-oriented learning classifier system library. *J. Artif. Soc. Simul.* **9**(3) (2006) <http://jasss.soc.surrey.ac.uk/9/3/9.html>
23. Llorá, X.: E2k: Evolution to knowledge. *SIGEVolution* **1**(3), 10–17 (2006)
24. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence. IJCAI'95, vol. 2, pp. 1137–1143. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995)
25. Dietterich, T.G.: Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.* **10**(7), 1895–1923 (1998)
26. Ortega, M., Bravo, J. (eds.): Computers and education in the 21st century. Kluwer, Dordrecht (2000)
27. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In: Ungar, L., Craven, M., Gunopulos, D., Eliassi-Rad, T. (eds.) KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 935–940. NY, USA, New York (2006)
28. Rakotomalala, R.: Tanagra : un logiciel gratuit pour l'enseignement et la recherche. In: S. Pinson, N. Vincent (eds.) EGC, Revue des Nouvelles Technologies de l'Information, pp. 697–702. Cpadus-ditions (2005)
29. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behaviour of several methods for balancing machine learning training data. *SIGKDD Explor.* **6**(1), 20–29 (2004)
30. He, H., Garcia, E.A.: Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.* **21**(9), 1263–1284 (2009)
31. Sun, Y., Wong, A.K.C., Kamel, M.S.: Classification of imbalanced data: A review. *Int. J. Pattern Recognit. Artif. Intel.* **23**(4), 687–719 (2009)
32. Dietterich, T., Lathrop, R., Lozano-Perez, T.: Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell.* **89**(1–2), 31–71 (1997)
33. Sánchez, L., Couso, I.: Advocating the use of imprecisely observed data in genetic fuzzy systems. *IEEE Trans. Fuzzy Syst.* **15**(4), 551–562 (2007)
34. Džemsar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**, 1–30 (2006)
35. García, S., Fernández, A., Luengo, J., Herrera, F.: Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Inf. Sci.* **180**(10), 2044–2064 (2010)

36. García, S., Herrera, F.: An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. *J. Mach. Learn. Res.* **9**, 2579–2596 (2008)
37. Fisher, R.A.: *Statistical methods and scientific inference* (2nd edition). Hafner Publishing, New York (1959)
38. García, S., Fernández, A., Luengo, J., Herrera, F.: A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability. *Soft Comput.* **13**(10), 959–977 (2009)
39. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC 2005 special session on real parameter optimization. *J. Heuristics* **15**, 617–644 (2009)
40. Luengo, J., García, S., Herrera, F.: A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and non-parametric tests. *Expert Syst. with Appl.* **36**, 7798–7808 (2009)
41. Cox, D., Hinkley, D.: *Theoretical statistics*. Chapman and Hall, London (1974)
42. Snedecor, G.W., Cochran, W.C.: *Statistical methods*. Iowa State University Press, Ames (1989)
43. Shapiro, S.S.: M.W.: An analysis of variance test for normality (complete samples). *Biometrika* **52**(3–4), 591–611 (1965)
44. Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat* **18**, 50–60 (1947)
45. Wilcoxon, F.: Individual comparisons by ranking methods. *Biometrics* **1**, 80–83 (1945)
46. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J. the Am. Stat. Assoc.* **32**(200), 675–701 (1937)
47. Iman, R., Davenport, J.: Approximations of the critical region of the friedman statistic. *Commun. Stat.* **9**, 571–595 (1980)
48. Sheskin, D.: *Handbook of parametric and nonparametric statistical procedures*. Chapman and Hall/CRC, Boca Raton (2006)
49. Holm, S.: A simple sequentially rejective multiple test procedure. *Scand. J. Stat.* **6**, 65–70 (1979)
50. Hochberg, Y.: A sharper bonferroni procedure for multiple tests of significance. *Biometrika* **75**, 800–803 (1988)
51. Nemenyi, P.B.: *Distribution-free multiple comparisons*, ph.d. thesis (1963)
52. Bergmann, G., Hommel, G.: Improvements of general multiple test procedures for redundant systems of hypotheses. In: Bauer, G.H.P., Sonnemann, E. (eds.) *Multiple hypotheses testing*, pp. 100–115. Springer, Berlin (1988)
53. Parpinelli, R., Lopes, H., Freitas, A.: Data mining with an ant colony optimization algorithm. *IEEE Trans. Evol. Comput.* **6**(4), 321–332 (2002)
54. Tan, K.C., Yu, Q., Ang, J.H.: A coevolutionary algorithm for rules discovery in data mining. *Int. J. Syst. Sci.* **37**(12), 835–864 (2006)
55. Aguilar-Ruiz, J.S., Giráldez, R., Riquelme, J.C.: Natural encoding for evolutionary supervised learning. *IEEE Trans. Evol. Comput.* **11**(4), 466–479 (2007)
56. Mansoori, E., Zolghadri, M., Katebi, S.: SGERD: A steady-state genetic algorithm for extracting fuzzy classification rules from data. *IEEE Trans. Fuzzy Syst.* **16**(4), 1061–1071 (2008)
57. Gray, J.B., Fan, G.: Classification tree analysis using TARGET. *Comput. Stat. Data Anal.* **52**(3), 1362–1372 (2008)