Comparison of KEEL versus open source Data Mining tools: Knime and Weka software

Complementary Material for the paper

KEEL 3.0: An Open Source Software for Multi-Stage Analysis in Data Mining

Isaac Triguero, Sergio González, Jose M. Moyano, Salvador García, Jesus Alcala-Fdez, Julián Luengo, Alberto Fernandez, María José del Jesus, Luciano Sanchez, Francisco Herrera

Report Date: 05-17-2016

Comparison of KEEL versus open source Data Mining tools: Knime and Weka software

Index of Contents

Introduction

It is well known that a wide amount of real application areas are benefited from Data Mining (DM) tasks with Machine Learning (ML) algorithms. For this reason, researchers, experts, and users interested in a myriad of different topics, have focused on the use of these types of tools.

In order to ease the access to the knowledge extraction models for people not related to computer science, many commercial and non-commercial software suites have been made available. The majority of the former are commercially distributed (some of the leading commercial software are mining suites such as SPSS Clementine, Oracle Data Mining and KnowledgeSTUDIO), but there is still several well-known and useful open source software.

Among these, a particular subset is represented by Workflow environments, in which loosely coupled, individual processing nodes can be 'bolted together' to permit complex computational operations.

Focusing on this type of software, we must of course stress the KEEL Software Suite [KEEL09, KEEL11]. It is devoted to cover the whole range of the DM field by providing a rather complete list of methods from different paradigms and fields. However, nowadays the most well-known open source DM package is maybe WEKA [WEKA11]. This software also includes a large collection of Java implementations of ML algorithms. We complete this list with The Konstanz Information Miner (KNIME) [KNIME09]. It is designed as a modular environment, which enables easy visual assembly and interactive execution of a data pipeline.

In this report, our objective is to review the capabilities of these three representative tools in the topic of DM. In order to do so, first we describe these software suites. Then we provide a thorough comparison of the former with respect to several significant features. Finally, we summarize the findings extracted throughout the previous analysis.

KNIME software tool

KNIME[1] [KNIME09] is a modular environment enables easy integration of new algorithms, data manipulation and visualization methods as models. The interface is configurable by selecting among several different methods. Specifically, one can select data sources, data preprocessing steps, model building algorithms, as well as visualization tools. To create the workflow, nodes are dragged from the node repository, dropped onto the workbench, and linked using the mouse to join their output and input ports (shown as small triangles).

A screenshot of an example analysis flow is shown in Figure 1. This process depicts a reading in data from one source (the iris dataset) which is then addressed by several parallel analysis flows, consisting of statistics, preprocessing, modeling, and

[1] http://www.knime.org

visualization nodes. In particular, Figure 2 shows the windows for the scatter plot and scorer nodes, including the confusion matrix and some metrics of performance.

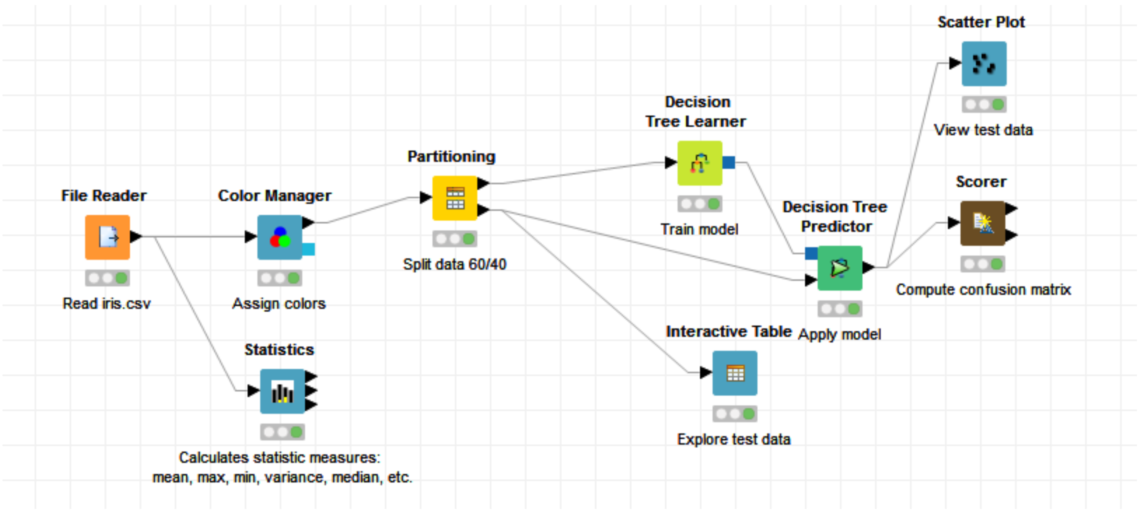
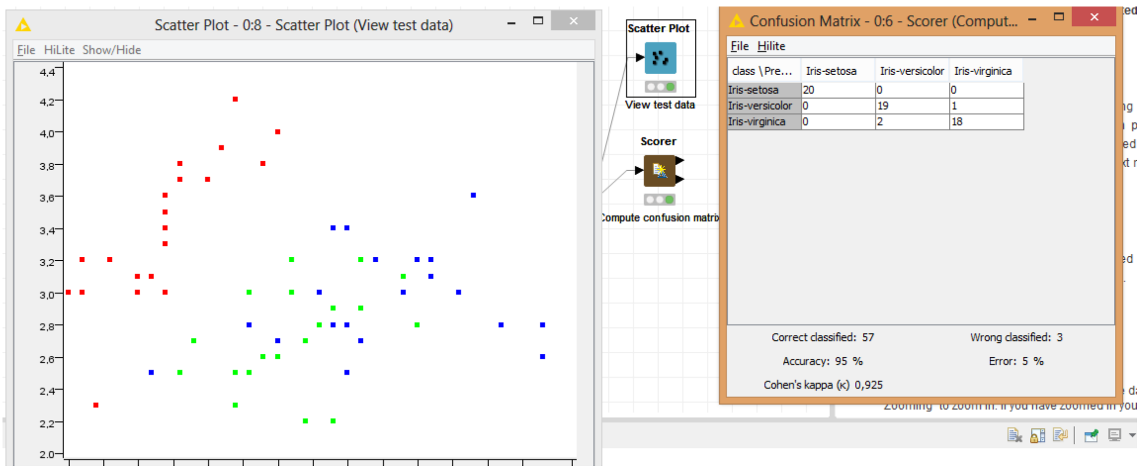


Figure 1. Workflow example for the KNIME software tool


Scatter Plot - 0:8 - Scatter Plot (View test data)
File HiLite Show/Hide
Confusion Matrix - 0:6 - Scorer (Comput...
File Hilite
Scatter Plot
View test data
Scorer
Compute confusion matrix

class \ Pre...	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	20	0	0
Iris-versicolor	0	19	1
Iris-virginica	0	2	18

Correct classified: 57 Wrong classified: 3

Accuracy: 95 % Error: 5 %

Cohen's kappa (κ) 0,925

Figure 2. Windows of the scatter plot and scorer visualization nodes.

The type of processing ranges from basic data operations such as filtering, to simple statistical functions, as well as computation intensive data modeling operators (clustering, decision trees, neural networks, to name just a few). In addition, most of the modeling nodes allow for an interactive exploration of their results through accompanying views. Next, it is defined a list of the capabilities provided by KNIME:

- Data I/O: Generic file reader and reader for the attribute-relation file format (ARFF), database connector, CSV and ARFF writer, Excel spreadsheet writer.
- Data manipulation: row and column filtering, data partitioning and sampling, sorting or random shuffling, data joiner and merger. Furthermore feature selection and feature extraction methods are also available.

- Data transformation: missing value replacer, matrix transposer, binners, nominal value generators.
- Mining algorithms: clustering (k-means, sota, fuzzy c-means, DBscan), bayesian, standard and ensemble decision trees, (fuzzy) rule induction, regression, subgroup and association rule mining, neural networks (probabilistic neural networks and multi-layer-perceptrons), and SVMs.
- Visualization: scatter plot, histogram, parallel coordinates, multidimensional scaling, rule plotters. Additionally, scorers and results plots can be also selected.
- Statistical tests: T-Test and ANOVA, as well as Wilcoxon test.
- Misc: scripting nodes

In addition to the former, and as stated at the beginning of this section, KNIME integrates functionality of different open source projects that essentially cover all major areas of data analysis. These plug-ins allow users to create wrappers for the former data analysis tools without having to modify these executables themselves:

• WEKA: for ML and DM. Essentially all algorithm implementations, for instance support vector machines, Bayes networks and Bayes classifier, decision tree learners

• R-project environment: for statistical computations and graphics console node to interactively execute R commands, basic R plotting node

• JFreeChart: for visualization. Various line, pie and histogram charts

Regarding the former, adding new nodes to KNIME is a straightforward process, also for native new operations. For this task, one needs to extend three abstract classes, namely “NodeModel”, “NodeDialog”, and “NodeView”. In addition to the three model, dialog, and view classes the programmer also needs to provide a NodeFactory, which serves to create new instances of the above classes. The factory also provides names and other details such as the number of available views or a flag indicating absence or presence of a dialog. A wizard integrated in the Eclipse-based development environment enables convenient generation of all required class bodies for a new node.

Finally, another advantage of KNIME is related to its modular architecture. This design allows at easily designating specific nodes to be run on separate machines, allowing parallelization of the processes.

WEKA software tool

WEKA[2] [WEKA11] is the most well-known software tool to perform ML and DM tasks. Its algorithms can either be applied directly to a dataset from its own interface or used in your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. Due to its enormous widespread usage, a complete set of extra packages are available for completing its functionalities.

WEKA supports four different frameworks for developing and executing the methods for DM tasks:

- Explorer: An environment for exploring data with WEKA.
- Experimenter: An environment for performing experiments and conducting statistical tests between learning schemes.
- KnowledgeFlow: This environment supports essentially the same functions as the Explorer but with a drag-and-drop interface. One advantage is that it supports incremental learning.
- SimpleCLI: Provides a simple command-line interface that allows direct execution of WEKA commands for operating systems that do not provide their own command line interface.

While for initial experiments the included graphical user interface is quite sufficient, for in-depth usage the command line interface is recommended, because it offers some functionality which is not available via the GUI - and uses far less memory.

WEKA Explorer is maybe the most used framework, and its aim is to provide comparison among algorithms. It presents seven custom tabs with specific options for the accomplishment of different DM tasks, such as data preprocessing, classification, and so on.

In this document, we will focus on the “KnowledgeFlow” environment of WEKA as it adheres to the user interface model of KNIME and KEEL. However, we must point out that it is still under development and it does not include the whole functionality for the Explorer and Experimenter sections. On the other hand, there are things that can be done in the KnowledgeFlow but not in the Explorer. KnowledgeFlow overcomes the limitations the Explorer has for dealing with incremental model building algorithms.

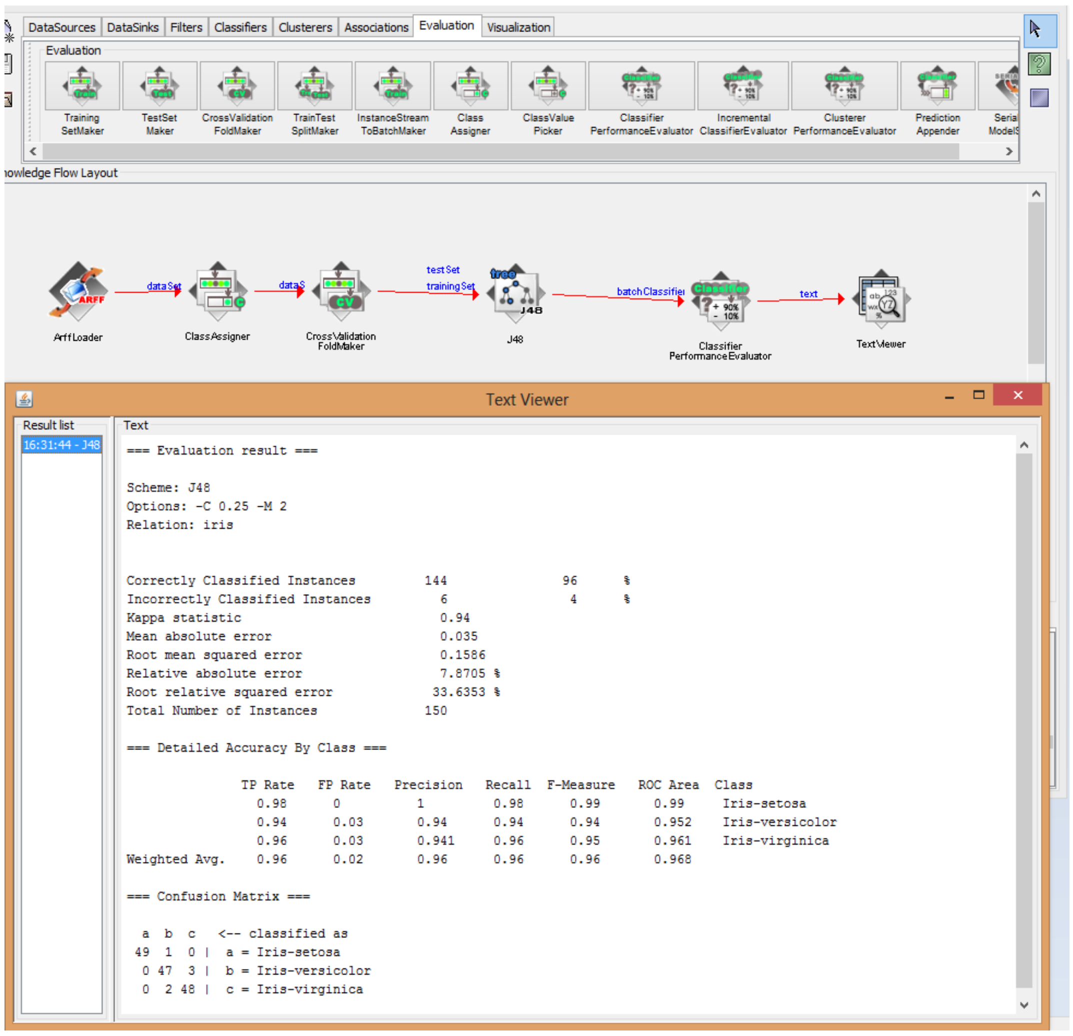
The KnowledgeFlow can handle data either incrementally or in batches (the Explorer handles batch data only). Of course learning from data incrementally requires a classifier that can be updated on an instance by instance basis. Currently in Weka there are five classifiers that can handle data incrementally: NaiveBayesUpdateable, IB1, IBk, LWR (locally weighted regression). There is also one meta classifier - RacedIncrementalLogitBoost - that can use of any regression base learner to learn from discrete class data incrementally.

[2] http://www.cs.waikato.ac.nz/ml/weka/

In summary, the features of the KnowledgeFlow, as stated in the WEKA guide, are the following ones:

- Intuitive data flow style layout
- Process data in batches or incrementally
- Process multiple batches or streams in parallel! (each separate flow executes in its own thread)
- Chain filters together
- View models produced by classifiers for each fold in a cross validation
- Visualize performance of incremental classifiers during processing (scrolling plots of classification accuracy, RMS error, predictions, and so on)
- Plugin facility for allowing easy addition of new components to the KnowledgeFlow

Specifically, we have depicted in Figure 3 an example of a workflow created within the KnowledgeFlow environment. Several nodes have been selected to carry out the experiment. First, the "ArffLoader" is devoted to read the dataset, which is then processed to the "classAsigner" to determine the column for the output class. Then, the "CrossValidationFoldMaker" divides the input data into training and test sets, which are then processed by the J48 (C4.5) decision tree. Finally the "ClassifierPerformanceEvaluator" computes the evaluation metrics that are then shown in the "TextViewer", whose window has been placed at the bottom of Figure 3.



```
=== Evaluation result ===

Scheme: J48
Options: -C 0.25 -M 2
Relation: iris


Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.035
Root mean squared error                  0.1586
Relative absolute error                  7.8705 %
Root relative squared error             33.6353 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.98      0          1         0.98      0.99       0.99     Iris-setosa
                 0.94      0.03       0.94      0.94      0.94       0.952    Iris-versicolor
                 0.96      0.03       0.941     0.96      0.95       0.961    Iris-virginica
Weighted Avg.    0.96      0.02       0.96      0.96      0.96       0.968

=== Confusion Matrix ===

  a  b  c   <-- classified as
 49  1  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

Figure 3. Example of the workflow for the WEKA KnowledgeFlow environment. Bottom of the figure shows the ouput for the "TextViewer" node.

WEKA includes a wide amount of ML algorithms from different paradigms and devoted to different tasks within the DM processes. These methods can be arranged within the following types:

- Filters: these are preprocessing approaches from supervised and unsupervised types. We may find attribute management such as feature selection and discretization, and instance preprocessing such as resampling.
- Classification models:
  - Bayes: contains bayesian classifiers, for example NaiveBayes.
  - Functions: such as Support Vector Machines, regression algorithms, or neural nets.
  - Lazy: "learning" is performed at prediction time, e.g., k-nearest neighbor (k-NN) or IBk.
  - Meta: meta-classifiers that use a base one or more classifiers as input. Some of these methods are boosting, bagging or stacking.
  - MI: classifiers that handle multi-instance data. CitationKNN and MI-Wrapper are two examples of this set of approaches.

    - Rules: rule-based classifiers, from ZeroR to JRip (Ripper). In developers' version 3.7 we may also find the FURIA fuzzy algorithm, an extension to Ripper.
    - Trees: tree classifiers, like decision trees with J48 (C4.5) being a very common one.
    - Misc: various classifiers that don't fit in any another category.
- Clustering: there are different approaches from the well-known kMeans, to DBScan.
- Association mining: in order to extract association rules, state-of-the-art methods are implements, such as Apriori, FP-Growth, and GSP.
- Evaluation: classes related to evaluation, e.g., confusion matrix, threshold curve (= ROC)
- Statistical tests: within the "Experimenter" environment, the Paired T-test is available.

In order to extend the number of implemented methods in WEKA with any new users' algorithm, a hierarchical and well-structured tree of Java classes is defined within the core source code. As an example, if any interested user wants to add a new classifier, it must override the "weka.classifiers.Classifier" an abstract class that defines the basic procedures that any method must contain in order to work within the WEKA environment.

Additionally, there are several more specific classes that ease the implementation for more sophisticated approaches. Hence, users can take advantage of the already codified methods in order to reduce the effort in developing their own code.

KEEL Software Suite

KEEL [KEEL09, KEEL11] is a free software (GPLv3) Java tool which empowers the user to assess the behavior of evolutionary learning and soft computing based techniques for different kind of data mining problems: regression, classification, clustering, pattern mining and so on. The main features of KEEL are:

- It contains a large collection of evolutionary algorithms for predicting models, preprocessing methods (evolutionary feature and instance selection among others) and postprocessing procedures (evolutionary tuning of fuzzy rules). It also presents many state-of-the-art methods for different areas of data mining such as decision trees, fuzzy rule based systems or crisp rule learning.
- It includes around 100 data preprocessing algorithms proposed in the specialized literature: data transformation, discretization, instance and feature selection, noise filtering and so forth.
- It incorporates a statistical library to analyze the results of the algorithms.
- It comprises a set of statistical tests for analyzing the suitability of the results and for performing parametric and nonparametric comparisons among the algorithms. It provides a user-friendly interface, oriented to the analysis of algorithms.
- The software is aimed to create experimentations containing multiple datasets and algorithms to obtain results. Experiments are independently script-generated from the user interface for an off-line run in any machine that supports a Java Virtual Machine.

The current version of KEEL (version 3.0) consists of the following function blocks:

- Data Management: The data management section brings together all the operations related to the datasets that are used during the data mining process. Some operations are related to the conversion of the dataset files from other dataset formats used in data management tools or data mining tools to the KEEL dataset format and vice versa.
- The Design of Experiments part has the objective of designing the desired experiments using a graphical interface. After the experiment is designed, the interface generates a .ZIP file containing a directory structure with all the necessary files needed to run those experiments in the local computer. Additionally, new developments can be compared versus standard algorithms already implemented and available in KEEL 3.0
- Educational: The educational section tries to be a helpful tool in a teaching environment. In order to achieve this objective, the educational section offers a real-time view of the evolution of the algorithms, allowing the students to use this information in order to learn how to adjust their parameters. In this sense, the educational module is a simplified version of the main KEEL research tool, where only the most relevant techniques are available. Using it, the user has a

visual feedback of the progress of the algorithms, and can access the final results from the same interface used to design the experiments.

- Modules: This part includes new modules extending the functionalities of the KEEL software tool for specific tasks associated to the data mining process that require special treatment:
    - Imbalanced Learning: This module features several algorithms specifically designed for Imbalanced Classification. The graphical interface gives the user access to a specific set of problems, algorithms and evaluation procedures covering the state-of-the-art in Imbalanced Classification maintaining the same structure and the same objectives as the Experiments section.
    - Non-Parametric Statistical Analysis: This module provides the user with several Non-parametric Statistical procedures for pairwise (Wilcoxon test) and multiple comparisons (Friedman, Friedman Aligned, Quade and Contrast Estimation), together with several post-hoc procedures for advanced verification of results, given in raw CSV format. Furthermore, this module outputs all the results of the analyses in latex format, easing the inclusion of the reports obtained in any experimental report.
    - Semi-Supervised Learning: This module, similar to the imbalanced learning module, is devoted to the creation and design of experiments related to semi-supervised learning. It features an interface similar to the experiments section featuring related datasets and methods which are useful in this scenario.
    - Multiple Instance Learning: This module (similarly to what the imbalanced learning module does) allows the user to create and prepare experiments for Multi Instance Learning. It features a graphical interface similar to the experiments section that gives access to specific Multi Instance datasets and algorithms designed to tackle this problem.

In this report, we focus on the “Design of Experiments” environment, following the same trend that in the case of KNIME and WEKA. As stated previously, this part of the software tool allows one to build easily experiments with ML algorithms by means of a simple graphical interface.

The objective is to use available datasets and algorithms to generate a directory structure with all the necessary files needed to run the designed experiments in the local computer selected by the user. The main advantage is forgetting about scripts and others parameter files that made arduous the design of an experiment, and begin using the new windows based interface.

As in other Experimental workflows, the user just needs to select the input data (data sets), the algorithms he/she wants to use and to make the opportune connections between them. Also it is possible to concatenate methods, insert statistical tests, and so on.

The task which is more simplified is probably the configuration of the parameters; everything can be done from a simple dialog without requirement of external configuration files.

Figure 5 shows a simplified example of the structure of a classification experiment. This flowchart includes the input datasets, a discretization node as preprocessing for the ID3 classifier, and the C4.5 classifier. Results for both methods will be contrasted using the Wilcoxon signed ranks tests. Finally, the evaluation will be summarized in a tabular way within the last “Vis-Clas-General” node.

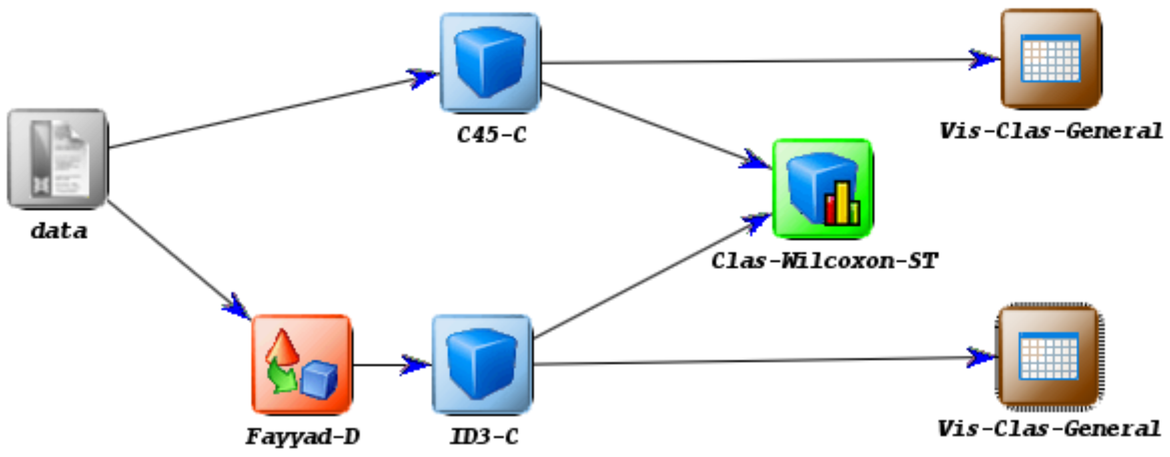


Figure 5. KEEL flowchart example for a full classification experiment

One of the greatest valuable features of KEEL is the substantial collection of ML algorithms that are implemented. Specifically, methods are divided into the following families:

- Pre-processing methods: Up to 100 methods to pre-process over the initial data sets. Among them, we may find approaches for data transformation, more than 30 different discretizers, feature selection methods, instance generation and instance selection approaches, missing values management methods, and data complexity computation.
- Standard methods: A wide amount of different DM methods are included in this family. Next we introduce all types of algorithms:
    - Classification Algorithms: There are almost 250 classification algorithms of different paradigms. These are listed below:
        - Decision Trees: Methods for building decision trees.
        - Fuzzy Rule Learning: Methods for performing fuzzy rule-based learning.

        - Hyperrectangles Learning: Methods using hyperrectangles to extract knowledge from data.
        - Lazy Learning: Learning methods which do not build a model in its training phase.
        - Neural networks: Artificial neural networks.
        - Rule Learning: Methods for performing rule-based learning.
        - Statistical Classifiers: Classifiers based on statistical models.
        - SVM: Support vector machines.
    - Imbalanced classification: these methods are adapted to address the different distribution between classes. These algorithms are divided into:
        - Methods for resampling the data space: 20 different preprocessing approaches.
        - Cost-sensitive learning: 3 methods from decision trees, SVM and neural networks.
        - Ensemble methods for class imbalance: 21 algorithms from bagging and boosting schemes.
        - Regression algorithms: 50 models including statistical models, symbolic regression, as well as (fuzzy) rule learning, neural networks and SVMs.
    - Association Rules: 16 different methods for extracting association rules from data.
    - Clustering Algorithms: Clustering methods
    - Subgroup Discovery: 7 methods implemented for subgroup discovery tasks.
- Post-processing methods: Up to 15 evolutionary post-processing approaches over the results of standard methods.
- Statistical tests: 25 Statistical procedures to contrast the results achieved in the experiment, divided into test analysis and post-hoc procedures.
- Visualization modules: Show the results of the experiments in an upgraded way.

Another interesting feature of the KEEL software suite is that it provides information of the output models, such as the tree structure of decision trees, or the knowledge base of a fuzzy rule based system.

Regarding the evaluation procedures, it includes several metrics of performance, such as the Area Under the ROC curve for imbalanced classification. Additionally, with the output files any interested researcher could extract the confusion matrix for computing any desired performance measure.

Finally, the extension of the current version of KEEL for including new approach is easy and straightforward. There are just some few guidelines for including algorithms, helping the researchers to make their methods easily accessible to other authors and to compare the results of many approaches already included within the KEEL software:

- The programming language used must be Java.

- A simple configuration file is requested to read the parameters. It must include the name of the method, a list of input and ouput files, and the list of parameters of the method, containing the name of each parameter and its value (one line is employed for each one).
- 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
- 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 test.

Although the list of constraints is short, the KEEL development team has 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".
2. ParseParameters: This class manages all the parameters, from the input and output files, to every single parameter stored in the parameters file.
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.
4. Algorithm: This class is devoted to storing the main variables of the algorithm and to naming the different procedures for the learning stage. It also contains the functions for writing the obligatory output files.

The template can be downloaded following the link 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.

As a summary, we must stress two main interesting features that KEEL supplies for a researcher: (1) a plethora of implemented algorithms for tackling problems such as data preprocessing, classification, numeric regression, subgroup discovery and association rule mining; and (2) the statistical testing for algorithms comparison, an important feature not considered in other tools that can be of great aid in the algorithm design task. On the contrary, KEEL's main drawback is possibly the lack of good visualization methods.

Analysis based on functionality

In the previous part of this report, we have described the main features of the widest known open source DM software tools. Our aim now is to provide a thorough analysis among them based on functionality criteria.

We must point out that the ultimate objective of this study is to stress which are those characteristics that support the main strengths for each particular tool, rather than to establish a comparison for the advantages of one over another.

Therefore, we have selected several features that will allow the reader to determine the major differences among the software tools, and then to categorize KEEL as a valuable alternative to these suites when other research requirements are needed.

Specifically, we distinguish four levels of support in these characteristics: none (N), basic support (B), intermediate support (I) and advanced support (A). If features do not have intermediate levels of support, the notation used is Yes (Y) for supporting and No (N) for no-supporting.

Selected criteria are briefly explained as follows:

- **Off/On-line run of the experiment set up**. An On-line run implies that the tool interface and algorithm modules need to be in the same machine and the experiments are completely dependent on the software tool. An off-line run entails the independence of the experiments created with respect to the suite interface, allowing the experiment to be executed in other machines.

- **Pre-processing Variety**. This comprises of discretization, feature selection, instance selection and missing values imputation. The trend of most of the suites is to offer a good feature selection and discretization set of methods, but they overlook specialized methods of missing values imputation and instance selection. Usually, the contributions included are basic modules of replacing or generating null values and methods for sampling the data sets by random (stratified or not) or by value-dependence.

- **Learning Variety.** It is supported over main areas of DM, such as predictive tasks (classification, regression, anomaly/deviation detection), and descriptive tasks (clustering, association rule discovery, sequential pattern discovery). Additionally, we take into account several novel DM scenarios such as Semi-Supervised Learning (some of the training instances are not labelled), Imbalanced Classification (distribution between examples of different classes is skewed) and Multi-Instance Learning ("bags" of instances are classified as a whole).

Intermediate level is awarded if the tool includes the classical models, and advance level is awarded it the tool contains advanced DM models from these areas.

- **Advanced Features**. This part includes some of the less common criteria incorporated for extending the functionality of the software tool.
    a. *Post-processing*, usually for tuning the model learned by an algorithm.
    b. *Meta-learning*, which includes more advanced learning schemes, such as bagging or boosting, or meta learning of the algorithm parameters.
    c. *Statistical tests* for establishing comparisons of results. An advanced support of this property requires a complete set of parametric and nonparametric statistical tests; a basic support implies the existence of well-known standard statistical tests (such as t-test).
    d. *EA support* indicates the integration of EAs into the DM areas that the software tool offers. A basic support of this feature implies the use of genetic algorithms in some techniques (usually, genetic feature selection). To upgrade the level it is necessary to incorporate EAs in learning or meta-learning models.
    e. *Fuzzy Learning Schemes* it refers to the case that Fuzzy Rule Based Systems are included for the learning stage. We consider a basic support if at least one algorithm of this paradigm is included, whereas an advanced level is given to the tool with the most recent approaches.
    f. *Multi-classifiers* stands for the binarization schemes, i.e. to carry out the learning of multiple class dataset in a divide and conquer approach. This includes both One-vs-One and One-vs-All methodologies.
- **Popularity**. Tool's popularity is measured using three variables: the number of results given by Google Scholar, the number of citations that the tool's most cited presenting paper has in Google Scholar, and finally the number of questions about the tool asked in Stackoverflow, where results are required to be included under the tags: machine-learning or data-mining.

All these characteristics are summarized in Tables 1, 2, 3, and 4. This analysis has been divided into three different parts for the sake of clarity in the presentation.

From these tables, it is shown that the main differences between KEEL and the remaining software tools are related to the Run-Mode, preprocessing variety, and the novel learning schemes, i.e. mainly due to the modules for semi-supervised learning, and imbalanced classification. Additionally, the support for non-parametrical statistical tests, evolutionary algorithms, fuzzy learning, and multi-classifiers also allow a clear distinction of KEEL versus WEKA and KNIME.

All the lessons learned from this study are given with detail in the next section.