J. of Mult.-Valued Logic & Soft Computing, Vol. 17, pp. 255–287
Reprints available directly from the publisher
Photocopying permitted by license only

KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework

J. ALCALÁ-FDEZ[1], A. FERNÁNDEZ[1], J. LUENGO[1], J. DERRAC[1],
S. GARCÍA[2], L. SÁNCHEZ[3] AND F. HERRERA[1]

[1]*Department of Computer Science and Artificial Intelligence, CITIC-UGR,*
University of Granada, 18071, Granada, Spain
E-mail: jalcala@decsai.ugr.es
[2]*Department of Computer Science, University of Jaén, 23071, Jaén, Spain*
[3]*Department of Computer Science, University of Oviedo, 33204, Gijón, Spain*

Received: December 15, 2009. Accepted: April 20, 2010

This work is related to the KEEL[1] (Knowledge Extraction based on Evolutionary Learning) tool, an open source software that supports data management and a designer of experiments. KEEL pays special attention to the implementation of evolutionary learning and soft computing based techniques for Data Mining problems including regression, classification, clustering, pattern mining and so on.

The aim of this paper is to present three new aspects of KEEL: KEEL-dataset, a data set repository which includes the data set partitions in the KEEL format and shows some results of algorithms in these data sets; some guidelines for including new algorithms in KEEL, 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; and a module of statistical procedures developed in order to provide to the researcher a suitable tool to contrast the results obtained in any experimental study. A case of study is given to illustrate a complete case of application within this experimental analysis framework.

Keywords: Data mining, Data set repository, evolutionary algorithms, java, knowledge extraction, machine learning

[1]http://www.keel.es

1 INTRODUCTION

Data Mining (DM) is the process for automatic discovery of high level knowledge by obtaining information from real world, large and complex data sets [26], and is the core step of a broader process, called Knowledge Discovery from Databases (KDD). In addition to the DM step, the KDD process includes application of several preprocessing methods aimed at faciliting application of DM algorithms and postprocessing methods for refining and improving the discovered knowledge. This idea of automatically discovering knowledge from databases present a very attractive and challenging task, both for academia and industry.

Evolutionary Algorithms (EAs) [14] are optimization algorithms based on natural evolution and genetic processes. They are currently considered to be one of the most successful search techniques for complex problems in Artificial Intelligence. The main motivation for applying EAs to knowledge extraction tasks is that they are robust and adaptive search methods that perform a global search in place of candidate solutions (for instance, rules or other forms of knowledge representation). They have proven to be an important technique both for learning and knowledge extraction, making them a promising technique in DM [8, 16, 22, 24, 35, 46].

In the last few years, many DM software tools have been developed. Although a lot of them are commercially distributed (some of the leading commercial software are mining suites such as SPSS Clementine[2], Oracle Data Mining[3] and KnowledgeSTUDIO[4]), only a few are available as open source software such as Weka [45] or Java-ML [1] (we recommend visiting the KDnuggets software directory[5] and The-Data-Mine site[6]). Open source tools can play an important role as is pointed out in [39].

KEEL (Knowledge Extraction based on Evolutionary Learning) [5] is a open source Java software tool which empowers the user to assess the behavior of 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:

- It reduces programming work. It includes a library with evolutionary learning algorithms based on different paradigms (Pittsburgh, Michigan and IRL) and simplifies the integration of evolutionary learning 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.

[2] http://www.spss.com/clementine
[3] http://www.oracle.com/technology/products/bi/odm
[4] http://www.angoss.com/products/studio/index.php
[5] http://www.kdnuggets.com/software
[6] http://the-data-mine.com/bin/view/Software

- It extends the range of possible users applying evolutionary learning algorithms. An extensive library of EAs together with easy-to-use software considerably reduce the level of knowledge and experience required by researchers in evolutionary computation. As a result researchers with less knowledge, when using this tool, would be able to successfully apply these algorithms to their problems.
- 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 his or hers machine, independently of the operating system.

In [5] we can find a description in detail of KEEL. In this paper, our aim is to present three new aspects of KEEL:

- KEEL-dataset, a data set repository that includes the data set partitions in the KEEL format and shows some results of algorithms in these data sets. This repository can free researchers from merely "technical work" and make the comparison of their models with the existing ones easier.
- KEEL has been developed with the idea of being easily extended with new algorithms. For this reason, we introduce some basic guidelines that the developer may take into account for managing the specific constraints of the KEEL tool. Moreover, a source code template have been made available to manage all the restrictions of the KEEL software, including the input and output functions, the parsing of the parameters, and the class structure. We will describe in detail this template showing a simple algorithm, the "Steady-State Genetic Algorithm for Extracting Fuzzy Classification Rules From Data" (SGERD) procedure. [33].
- A module of statistical procedures developed in order to provide to the researcher a suitable tool to contrast the results obtained in any experimental study performed inside the KEEL environment. We will describe this module and show a case of study using some non-parametric statistical tests for the multiple comparison of the performance of several genetic rule learning methods for classification.

This paper is arranged as follows. Section 2 presents an introduction to the KEEL software tool, including a short description of its structure and design of experiments. Section 3 describes KEEL-dataset and its main characteristics. In Section 4 we show how to implement or to import an algorithm into the KEEL software tool and an example of codification using the KEEL template. Section 5 presents the module of statistical procedures and shows a case of study to illustrate a complete case of application. In Section 6 some concluding remarks are made. Finally, we include two appendices with a description of the methods and non-parametric tests used in our case of study.

2 KEEL DESCRIPTION

KEEL is a software tool to assess EAs for DM problems including regression, classification, clustering, pattern mining and so on. The version of KEEL presently available consists of the following function blocks (see Fig. 1):

- *Data Management*: This part is made up of a set of tools that can be used to export and import data in other formats to or from the KEEL format, data edition and visualization, to apply partitioning to data and so on.
- *Design of Experiments*: The aim of this part is the design of the desired experimentation over the selected data sets and the provision of many options in different areas: type of validation, type of learning (classification, regression, unsupervised learning) and so on. Once the experiment has been generated, the user may execute it in batch mode.
- *Educational Experiments*: With a similar structure to the previous part, this allows for the design of experiments that can be run step-by-step in order to display the learning process of a certain model by using the software tool for educational purposes.

This structure makes KEEL software useful for different types of user, who expect to find different functionalities in a piece of DM software. Here is a brief description of the main features of KEEL:

- It presents a large collection of EAs for predicting models, pre-processing (evolutionary feature and instance selection) and post-processing (evolutionary tuning of fuzzy rules). It also contains some state-of-the-art methods for different areas of DM such as decision trees, fuzzy rule based systems or interval rule-based learning.

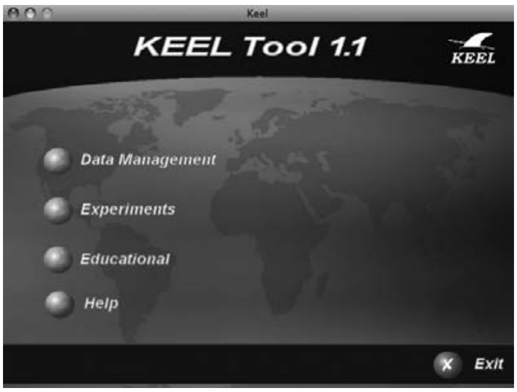


FIGURE 1
Screenshot of the main window of KEEL software tool

- It includes data pre-processing algorithms proposed in specialized literature: data transformation, discretization, instance selection and feature selection.
- It has a statistical library to analyze results of algorthms. It comprises a set of statistical tests for analyzing the suitability of the results and performing parametric and non-parametric comparisons between the algorithms.
- Some algorithms have been developed using Java Class Library for Evolutionary Computation (JCLEC) [43].
- It provides a user-friendly interface, oriented to the analysis of algorithms.
- The software is aimed at creating experiments containing multiple data sets and algorithms connected among themselves to obtain an expected results. Experiments are independently script-generated from the user interface for an off-line run in the same or other machines.
- KEEL also allows the creation of experiments in on-line mode, aiming to provide an educational support in order to learn the operation of the algorithm included.

For more information about the main features of the KEEL tool, such as the *Data Management*, the *Design of Experiments* function block, or the *On-Line Module for Computer-Based Education*, please refer to [5].

3 KEEL-DATASET

In this section we present the KEEL-dataset repository. It can be accessed through the main KEEL webpage[7]. 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

[7] http://keel.es/datasets.php

are presented. These webpages also provide the complete data set and the partitions ready to download.

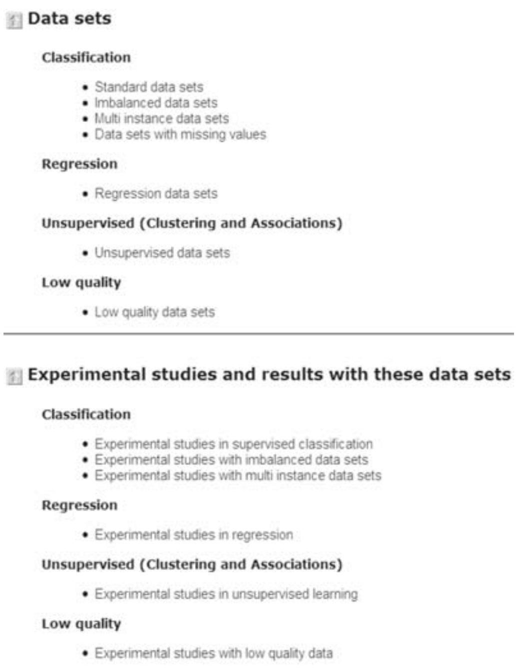
On the other hand, the experimental studies section is a novel approach in this type of repositories. It provides a series of webpages for each experimental study with the data sets used and their results in different formats as well, ready to perform a direct comparison. A direct access to the paper's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

In Figure 2 the main webpage is depicted in which the two mentioned main sections appear.

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

3.1 Data sets webpages

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:



Data sets

Classification

- Standard data sets
- Imbalanced data sets
- Multi instance data sets
- Data sets with missing values

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

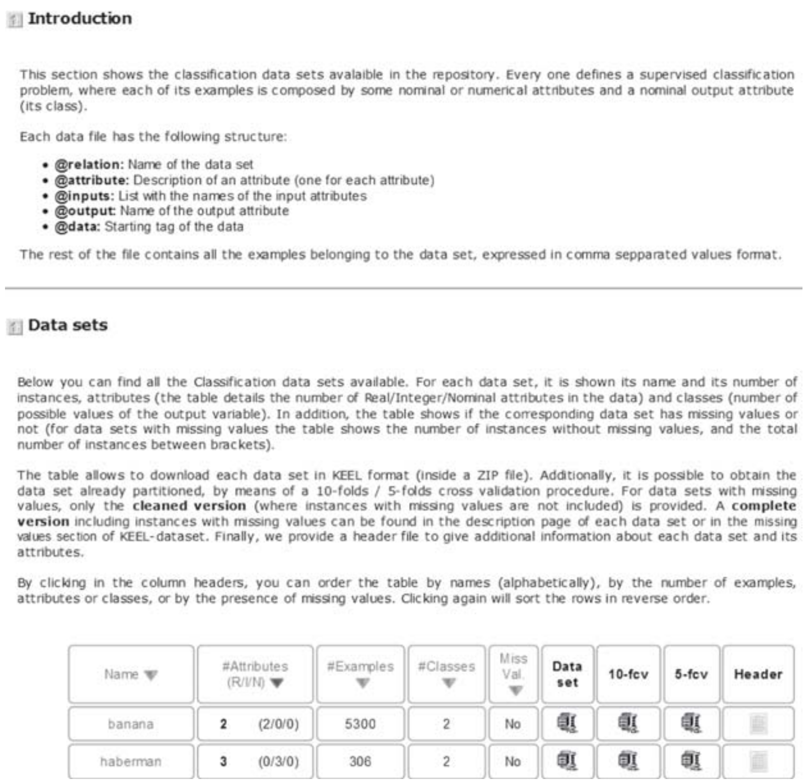
FIGURE 2
KEEL-dataset webpage (http://keel.es/datasets.php)

- 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 four subcategories of classification data sets:
  - *Standard data sets*.
  - *Imbalanced data sets* [6, 28, 41]. Imbalanced data sets are standard classification data sets where the class distribution is highly skewed among the classes.
  - *Multi instance data sets* [12]. Multi-Instance data sets represents problems where there is a many-to-one relationship between feature vectors and its output attribute.
  - *Data sets with missing values*. These include the classification data sets which contain missing values.
- Regression problems. These are data sets with a real valued output attribute, and the objective is to approximate this output value the better using the input attributes.
- Unsupervised (Clustering and Associations) problems. Unsupervised data sets represents a set of data whose examples have been not labeled.
- Low quality data [37]. In this category the data sets which contains imprecise values in their inputs attributes are included, caused by noise or restrictions in the measurements. Therefore this low quality data sets can contain a mixture of crisp and fuzzy values. This is a unique category.

In Figure 3 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:

- 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 to indicate the begin of the data.
- The second part is a relation of the different data sets contained in the webpage. This relation is presented in a table. The table shows the characteristics of all the data sets: the name of the data set, number of attributes (with the number of real, integer and nominal attributes in parenthesis respectively), number of examples and number of classes (if applicable). Moreover the possibility of download 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 can be also sorted by the different data set's characteristics columns, like the number of attributes or examples.



Introduction

This section shows the classification data sets avalaible 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 sepparated values format.

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 instances, attributes (the table details the number of Real/Integer/Nominal attributes in the data) and classes (number of possible values of the output variable). In addition, the table shows if the corresponding data set has missing values or not (for data sets with missing values the table shows the number of instances without missing values, and the total number of instances between brackets).

The table allows to 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. For data sets with missing values, only the **cleaned version** (where instances with missing values are not included) is provided. A **complete version** including instances with missing values can be found in the description page of each data set or in the missing values section of KEEL-dataset. Finally, we provide a header file to give additional information about each data set and its attributes.

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

Name ▼	#Attributes (R/I/N) ▼	#Examples ▼	#Classes ▼	Miss Val. ▼	**Data set**	**10-fcv**	**5-fcv**	**Header**
banana	**2** (2/0/0)	5300	2	No	🗜	🗜	🗜	📄
haberman	**3** (0/3/0)	306	2	No	🗜	🗜	🗜	📄

FIGURE 3
Fraction of Keel-dataset standard data sets' webpage

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

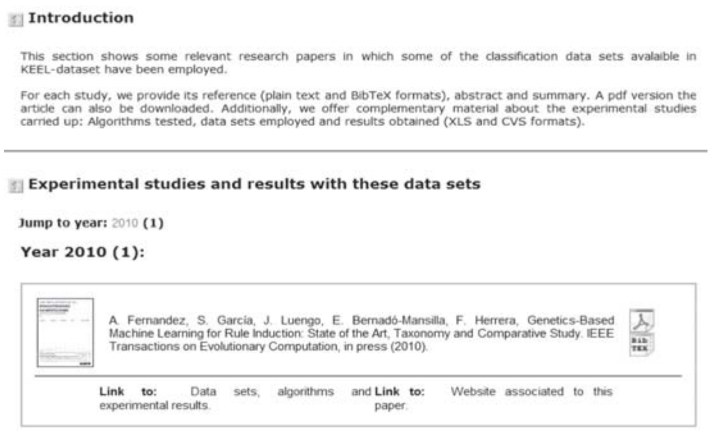
3.2 Experimental study webpages

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 Figure 4 for the webpage devoted to the experimental studies with standard classification data sets.

These webpages contains 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:

- The first link is the PDF file of the paper.
- The second link is the Bibtex reference of the paper.
- At the bottom on the left link *Data sets, algorithms and experimental results* is always present. It references to the particular Keel-dataset webpage for such paper.
- At the bottom on the right link *Website associated to this paper* is only present for some papers which have a particular and external webpage related with 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 to download in separated ZIP files. Moreover, the results are detailed and listed in CSV and XLS (Excel) formatted files. In the Figure 5 an example of the webpage for a specific publication with all these fields is shown.



Introduction

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

For each study, we provide its reference (plain text and BibTeX formats), abstract and summary. A pdf version 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

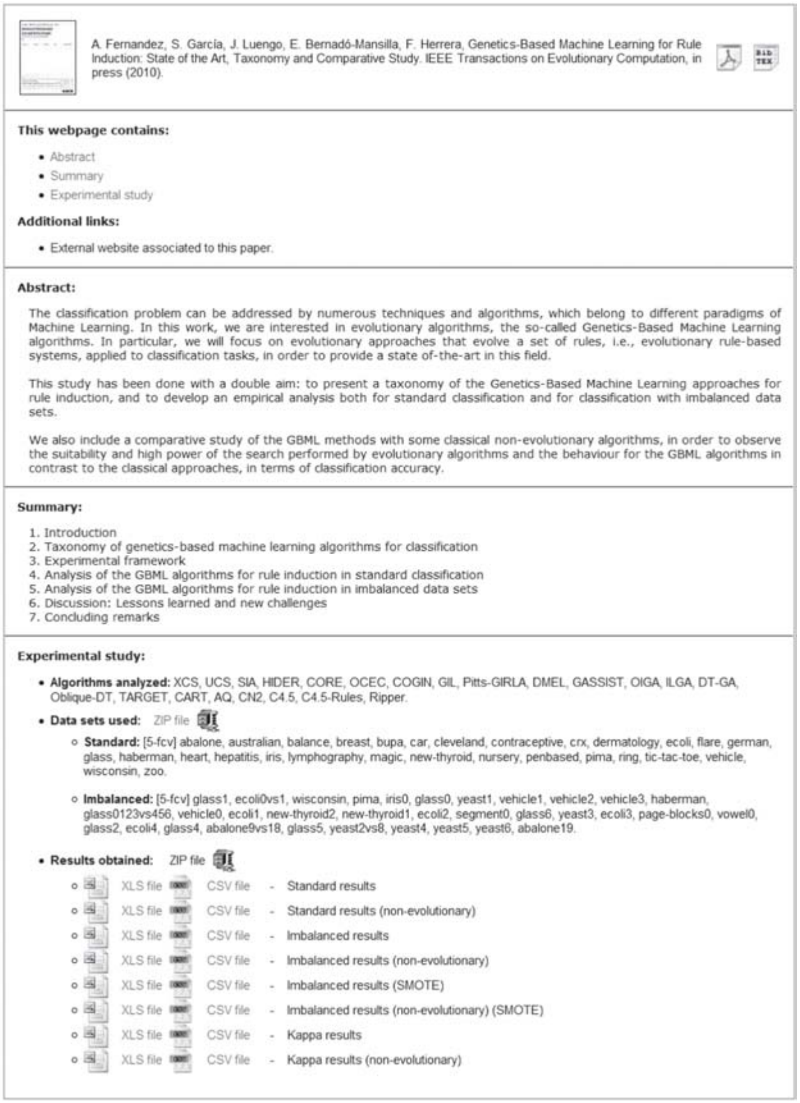
Jump to year: 2010 **(1)**

Year 2010 (1):

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).

Link to: Data sets, algorithms and experimental results. and **Link to:** Website associated to this paper.

FIGURE 4
Keel-dataset experimental studies with standard classification data sets webpage



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).

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)

FIGURE 5
Keel-dataset example of an experimental study dedicated webpage

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.

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 fewest possible constraints for the developer, in order to ease the inclusion of new algorithms within this tool. 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 depends 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.

  Each configuration file has the following structure:

  - *algorithm*: Name of the method.
  - *inputData*: A list with the input data files of the method.
  - *outputData*: A list with 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 employed 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 Section 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 Section 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 Section 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 test.

  For more information about the obligatory output files please refer to Section 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 to 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 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. Nevertheless, in the next section we will explain in detail how to encode a simple algorithm within the KEEL software tool.

4.2 Encoding example using the "Steady-State Genetic Algorithm for Extracting Fuzzy Classification Rules From Data" method

Including new algorithms in the KEEL software tool is very simple using the source code template presented in the previous section. We will show how this template enables the programming within KEEL to be straightforward, since the user does not need to pay attention to the specific KEEL constraints because they are completely covered by the functions implemented in the template. To illustrate this, we have selected one classical and simple method, the SGERD procedure [33].

Neither the Main nor the ParseParameters classes need to be modified, and we just need to focus our attention on the Algorithm class and the inclusion of two new functions in myDataset. We enumerate below the steps for adapting this class to this specific algorithm:

1. First of all, we must store all the parameters values within the constructor of the algorithm. Each parameter is selected with the `getParameter` function using its corresponding position in the parameter file, whereas the optional output files are obtained using the function `getOutputFile`. Furthermore, the constructor must check the capabilities of the algorithm, related to the data-set features, that is, whether it has missing values, real or nominal attributes, and so on. These operations are shown in the following source code, in which the operations that are added to the template are stressed in **boldface**:

```
public SGERD(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);
    train.computeOverlapping();
  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;
  }

 somethingWrong = somethingWrong || train.hasMissingAttributes();

  outputTr = parameters.getTrainingOutputFile();
  outputTst = parameters.getTestOutputFile();

  fileDB = parameters.getOutputFile(0);
  fileRB = parameters.getOutputFile(1);

  long seed = Long.parseLong(parameters.getParameter(0));

  Q = Integer.parseInt(parameters.getParameter(1));
  if ((Q < 1) || (Q > (14*train.getnInputs())))
```

```
        Q = Math.min((14*train.getnInputs()) /
                (2*train.getnClasses()), 20);

    typeEvaluation = Integer.parseInt(parameters.getParameter(2));
    K = 5;

    Randomize.setSeed(seed);
  }
```

2. Next, we execute the main process of the algorithm (procedure `execute`). The initial step is to abort the program if we have found a problem during the building phase of the algorithm (constructor). If everything is alright, we perform the algorithm's operations. In the case of the SGERD method we must first build the Data Base (DB) and then generate an initial Rule Base (RB). Next, the GA is executed in order to find the best rules in the system. When this process is complete, we perform the final output operations. This process is shown below in its entirety (again the new inserted code is stressed in **boldface**):

```
public void execute() {
  if (somethingWrong) {
  System.err.println("An error was found, the data-set has MV.");
  System.err.println("Please remove the examples with missing"+
          "data or apply a MV preprocessing.");
  System.err.println("Aborting the program");
  }
  else {

    dataBase = new DataBase(K, train.getnInputs(),
          train.getRanges(), train.varNames());
    ruleBase = new RuleBase(dataBase, train, typeEvaluation);
    ruleBase.initialization();

    Population pobl = new Population(ruleBase, Q, train,
          dataBase.numLabels());
    pobl.Generation();

    dataBase.saveFile(fileDB);
    ruleBase = pobl.bestRB();
    ruleBase.saveFile(fileRB);

    doOutput(val, outputTr);
    doOutput(test, outputTst);

    System.out.println("Algorithm Finished");
  }
}
```

3. We write in an output file the DB and the RB to save the generated fuzzy model, and then we continue with the classification step for both the validation and test files. The `doOutput` procedure simply iterates all examples and returns the predicted class as a string value (in regression problems it will return a double value). This prediction is carried out in the `classificationOutput` function, which only runs the Fuzzy Reasoning Method of the generated RB (noted in **boldface**):

```
private void doOutput(myDataset dataset, String filename) {
    String output = new String("");
    output = dataset.copyHeader();
    for(int i = 0; i < dataset.getnData(); i++) {
      output += dataset.getOutputAsString(i) + " " +
        classificationOutput(dataset.getExample(i)) + "\n";
    }
    Files.writeFile(filename, output);
  }

  private String classificationOutput(double[] example) {
    String output = new String("?");

    int clas = ruleBase.FRM(example);

    if (clas >= 0) {
      output = train.getOutputValue(clas);
    }
    return output;
  }
```

4. Finally, we show the new functions that are implemented in the myDataset class in order to obtain some necessary information from the training data during the rule learning stage. We must point out that the remaining functions of this class remain unaltered.

```
  public void computeOverlapping() {
    int i;

    classOverlapping = new double[nClasses];

    outliers = new int[nClasses];
    nExamplesClass = new int[nClasses];

    for (i = 0; i < nClasses; i++) {
      outliers[i] = nExamplesClass[i] = 0;
    }

    KNN knn = new KNN(IS, 5);
    knn.ejecutar(outliers, nExamplesClass);

    for (i = 0; i < nClasses; i++) {
      if (nExamplesClass[i] > 0) {
        classOverlapping[i] = (1.0 - (outliers[i] /
            nExamplesClass[i]));
      }
      else {
        classOverlapping[i] = 1.0;
      }
    }
  }

  public double getOverlapping(int nClass) {
    return (classOverlapping[nClass]);
  }
```

Once the algorithm has been implemented, it can be executed directly on a terminal with the parameters file as an argument. Nevertheless, when included within the KEEL software, the user can create a complete experiment with automatically generated scripts for a batch-mode execution. Furthermore, we must clarify that the "validation file" is used when an instance-selection preprocessing step is performed, and contains the original training set data; hence, the training and validation files match up in the remaining cases.

Finally, we should point out that the complete source code for the SGERD method (together with the needed classes for the fuzzy rule generation step) can be downloaded at http://www.keel.es/software/SGERD_source.zip.

5 STATISTICAL TOOLS AND EXPERIMENTAL STUDY

One of the important features of the KEEL software tool is the availability of a complete package of statistical procedures, developed with the aim of providing to the researcher a suitable tool to contrast the results obtained in any experimental study performed inside the KEEL environment. This section is devoted to present them (Section 5.1), and to show a complete case of application (Section 5.2) within the framework of an experimental comparison of several genetic rule learning methods for classification.

5.1 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 Data Mining [10, 19, 20]. 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.

There exist two kinds of test: parametric and non-parametric, depending of 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 are 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) [15]. Unfortunately, parametric tests are based on assumptions which are most probably violated when analyzing the performance of computational intelligence and data mining algorithms [21, 18, 32]. These assumpitions are known as independence, normality and homoscedasticity.

Nonparametric tests can be employed in the analysis of experiments, providing to the researcher 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 nonparemetric 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* [8].

KEEL is one of the fewest Data Mining software tools that provides to the researcher 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, making easy both its employment and its integration inside a complete experimental study.

Table 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 [9]).

5.2 Case of 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

Procedure	Ref.	Description
5x2cv-f test	[11]	Approximate f statistical test for 5x2 cross validation
T test	[9]	Statistical test based on the Student's t distribution
F test	[25]	Statistical test based on the Snedecor's F distribution
Shapiro-Wilk test	[40]	Variance test for normality
Mann-Whitney U test	[27]	U statistical test of difference of means
Wilcoxon test	[44]	Nonparametric pairwise statistical test
Friedman test	[17]	Nonparametric multiple comparisons statistical test
Iman-Davenport test	[31]	Derivation from the Friedman's statistic (less conservative)
Bonferroni-Dunn test	[38]	Post-Hoc procedure similar to Dunnet's test for ANOVA
Holm test	[30]	Post-Hoc sequential procedure (most significant first)
Hochberg test	[29]	Post-Hoc sequential procedure (less significant first)
Nemenyi test	[34]	Comparison with all possible pairs
Hommel test	[7]	Comparison with all possible pairs (less conservative)

TABLE 1
Statistical procedures available in KEEL

[8] http://sci2s.ugr.es/sicidm/
[9] http://www.keel.es

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.

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 2 shows their references and gives a brief description of each one. This description is extended in the Appendix A.

On the other hand, we have used 24 well-known classification data sets (they are publicly available on the KEEL-dataset repository web page [10], including general information about them, partitions and so on) in order to check the performance of these methods. Table 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 ten fold cross-validation (10-fcv) procedure as a validation scheme to perform the experiments.

Setting up the Experiment under KEEL software

To do this experiment in KEEL, first of all we click the Experiment option in the main menu of the KEEL software tool, define the experiment as a

Method	Ref.	Description
Ant-Miner	[36]	An Ant Colony System based using a heuristic function based in the entropy measure for each attribute-value
CORE	[42]	A coevolutionary method which employs as fitness measure a combination of the true positive rate and the false positive rate
HIDER	[2, 4]	A method which iteratively creates rules that cover randomly selected examples of the training set
SGERD	[33]	A steady-state GA which generates a prespecified number of rules per class following a GCCL approach
TARGET	[23]	A GA where each chromosome represents a complete decision tree.

TABLE 2
Algorithms tested in the experimental study

[10] http://www.keel.es/datasets.php

Name	#Ats	#Ins	#Cla	Name	#Ats	#Ins	#Cla
Haberman	3	306	2	Wisconsin	9	699	2
Iris	4	150	3	Tic-tac-toe	9	958	2
Balance	4	625	3	Wine	13	178	3
New Thyroid	5	215	3	Cleveland	13	303	5
Mammographic	5	961	2	Housevotes	16	435	2
Bupa	6	345	2	Lymphography	18	148	4
Monk-2	6	432	2	Vehicle	18	846	4
Car	6	1728	4	Bands	19	539	2
Ecoli	7	336	8	German	20	1000	2
Led-7	7	500	10	Automobile	25	205	6
Pima	8	768	2	Dermatology	34	366	6
Glass	9	214	7	Sonar	60	208	2

TABLE 3
Data sets employed in the experimental study

Classification problem and use a 10-fold cross validation 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 3. The partitions in KEEL are static, allowing that further experiments carried out will give up being dependent on particular data partitions.

The graph in Figure 6 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 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 five learning methods used (Clas-AntMiner, Clas-SGERD, Clas-Target, Clas-Hider and Clas-CORE).

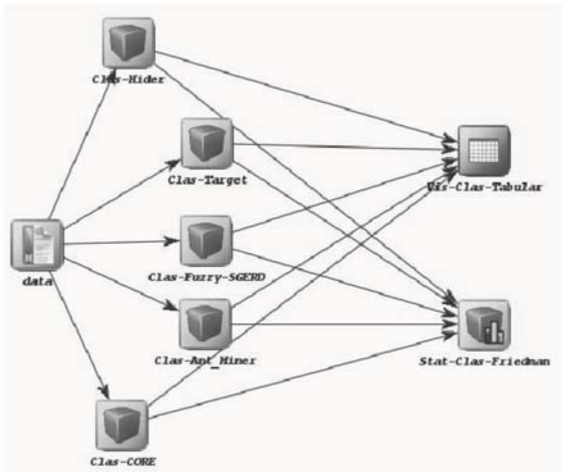


FIGURE 6
Graphical representation of the experiment in KEEL

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

TABLE 4
Parameter’ values employed in the experimental study

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 6 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 [18].

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.

Results and Analysis

This subsection describes and discusses the results obtained from the previous experiment configuration. Tables 5 and 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 Appendix B for a full description), by including a node called Stat-Clas-Friedman in the KEEL experiment. Here, we include the information provided by this statistical module:

- Table 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 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.

	Ant Miner		CORE		HIDER		SGERD		TARGET	
Data set	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Haberman	**79.55**	1.80	76.32	1.01	76.58	1.21	74.29	0.81	74.57	1.01
Iris	97.26	0.74	95.48	1.42	**97.48**	0.36	97.33	0.36	93.50	2.42
Balance	73.65	3.38	68.64	2.57	75.86	0.40	76.96	2.27	**77.29**	1.57
New Thyroid	**99.17**	0.58	92.66	1.19	95.97	0.83	90.23	0.87	88.05	2.19
Mammographic	81.03	1.13	79.04	0.65	**83.60**	0.75	74.40	1.43	79.91	0.65
Bupa	**80.38**	3.25	61.93	0.89	73.37	2.70	59.13	0.68	68.86	0.89
Monk-2	97.22	0.30	87.72	7.90	97.22	0.30	80.56	0.45	**97.98**	7.90
Car	77.95	1.82	**79.22**	1.29	70.02	0.02	67.19	0.08	77.82	0.29
Ecoli	87.90	1.27	67.03	3.69	**88.59**	1.77	73.02	0.86	66.22	4.69
Led7Digit	59.42	1.37	28.76	2.55	**77.64**	0.42	40.22	5.88	34.24	3.55
Pima	71.86	2.84	72.66	2.62	**77.82**	1.16	73.71	0.40	73.42	2.62
Glass	81.48	6.59	54.26	1.90	**90.09**	1.64	53.84	2.96	45.07	0.90
Wisconsin	92.58	1.65	94.71	0.64	**97.30**	0.31	93.00	0.85	96.13	0.64
Tic-tac-toe	69.62	2.21	69.46	1.20	69.94	0.53	69.94	0.53	**69.96**	2.20
Wine	**99.69**	0.58	99.06	0.42	97.19	0.98	91.76	1.31	85.19	1.58
Cleveland	60.25	1.35	56.30	1.97	**82.04**	1.75	46.62	2.23	55.79	2.97
Housevotes	94.28	1.84	**96.98**	0.43	**96.98**	0.43	**96.98**	0.43	**96.98**	0.43
Lymphography	77.11	5.07	65.99	5.43	**83.70**	2.52	77.48	3.55	75.84	4.43
Vehicle	59.52	3.37	36.49	3.52	**84.21**	1.71	51.47	1.19	51.64	2.52
Bands	67.61	3.21	66.71	2.01	**87.13**	2.15	63.84	0.74	71.14	2.01
German	71.14	1.19	70.60	0.63	**73.54**	0.58	67.07	0.81	70.00	1.37
Automobile	69.03	8.21	31.42	7.12	**96.58**	0.64	52.56	1.67	45.66	6.12
Dermatology	86.18	5.69	31.01	0.19	**94.91**	1.40	72.69	1.04	66.24	1.81
Sonar	74.68	0.79	53.37	0.18	**98.29**	0.40	75.69	1.47	76.87	1.18
Average	79.52	2.51	68.16	2.14	**86.09**	1.04	71.76	1.37	72.43	2.33

TABLE 5
Average results and standard deviations of training accuracy obtained

- Finally, in Table 9 the adjusted *p*-values are shown considering the best method (Hider) as control 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.

	Ant Miner		CORE		HIDER		SGERD		TARGET	
Data set	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Haberman	72.55	5.27	72.87	4.16	**75.15**	4.45	74.16	2.48	71.50	2.52
Iris	96.00	3.27	92.67	4.67	**96.67**	3.33	**96.67**	3.33	92.93	4.33
Balance	70.24	6.21	70.08	7.11	69.60	3.77	75.19	6.27	**75.62**	7.27
New Thyroid	**90.76**	6.85	**90.76**	5.00	90.28	7.30	88.44	6.83	86.79	5.83
Mammographic	81.48	7.38	77.33	3.55	**82.30**	6.50	74.11	5.11	79.65	2.11
Bupa	57.25	7.71	61.97	4.77	65.83	10.04	57.89	3.41	**65.97**	1.41
Monk-2	**97.27**	2.65	88.32	8.60	**97.27**	2.65	80.65	4.15	96.79	5.15
Car	77.26	2.59	**79.40**	3.04	70.02	0.16	67.19	0.70	77.71	2.70
Ecoli	58.58	9.13	64.58	4.28	**75.88**	6.33	72.08	7.29	65.49	4.29
Led7Digit	55.32	4.13	27.40	4.00	**68.20**	3.28	40.00	6.75	32.64	6.75
Pima	66.28	4.26	73.06	6.03	73.18	6.19	**73.71**	3.61	73.02	6.61
Glass	53.74	12.92	45.74	9.36	**64.35**	12.20	48.33	5.37	44.11	5.37
Wisconsin	90.41	2.56	92.38	2.31	**96.05**	2.76	92.71	3.82	95.75	0.82
Tic-tac-toe	64.61	5.63	**70.35**	3.77	69.93	4.73	69.93	4.73	69.50	2.73
Wine	92.06	6.37	**94.87**	4.79	82.61	6.25	87.09	6.57	82.24	7.57
Cleveland	**57.45**	5.19	53.59	7.06	55.86	5.52	44.15	4.84	52.99	1.84
Housevotes	93.56	3.69	**97.02**	3.59	**97.02**	3.59	**97.02**	3.59	96.99	0.59
Lym	73.06	10.98	65.07	15.38	72.45	10.70	72.96	13.59	**75.17**	10.59
Vehicle	53.07	4.60	36.41	3.37	**63.12**	4.48	51.19	4.85	49.81	5.85
Bands	59.18	6.58	64.23	4.23	62.15	8.51	62.71	4.17	**67.32**	6.17
German	66.90	3.96	69.30	1.55	**70.40**	4.29	66.70	1.49	70.00	0.49
Automobile	53.74	7.79	32.91	6.10	**62.59**	13.84	50.67	10.27	42.82	13.27
Dermatology	81.16	7.78	31.03	1.78	**87.45**	3.26	69.52	4.25	66.15	4.25
Sonar	71.28	5.67	53.38	1.62	52.90	2.37	73.45	7.34	**74.56**	8.34
Average	72.22	5.97	66.86	5.01	**75.05**	5.69	70.27	5.20	71.06	4.87

TABLE 6
Average results and standard deviations of test accuracy obtained

Algorithm	Ranking
AntMiner	3.125
CORE	3.396
Hider	**2.188**
SGERD	3.125
Target	3.167

TABLE 7
Average Rankings of the algorithms by Friedman procedure

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

TABLE 8
Results of the Friedman and Iman-Davenport Tests

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

TABLE 9
Adjusted p-values. Hider is the control algorithm

6 CONCLUDING REMARKS

The objective of this paper was to present three new aspects of KEEL:

- 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 make the comparison of their models with the existing ones easier.
- 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 has only to focus on the inner functions of his or hers 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.

We have shown a case of study to illustrate the simplicity of designing a experimental study with a statistical analysis into the KEEL software. In this case, the results obtained have been contrasted through a statistical analysis following the indications given in [18], concluding that the Hider method is the best performing method when compared with the remaining methods analyzed in this study.

ACKNOWLEDGMENTS

This work has been supported by the Spanish Ministry of Education and Science under Project TIN2008-06681-C06-(01 and 04). J. Luengo holds a FPU scholarship from Spanish Ministry of Education and Science and J. Derrac holds a research scholarship from the University of Granada.

REFERENCES

[1] T. Abeel, Y. Van de Peer, and Y. Saeys. (2009). Java-ML: A machine learning library. *Journal of Machine Learning Research*, 10:931–934.

[2] J. S. Aguilar-Ruiz, R. Giráldez, and J. C. Riquelme. (2007). Natural encoding for evolutionary supervised learning. *IEEE Transactions on Evolutionary Computation*, 11(4):466–479.

[3] J.S. Aguilar-Ruiz, J. Bacardit, and F. Divina. (2004). Experimental evaluation of discretization schemes for rule induction. In *Genetic and Evolutionary Computation GECCO-2004*, volume 3102 of *LNCS*, pages 828–839. Springer-Verlag.

[4] J.S. Aguilar-Ruiz, J.C. Riquelme, and M. Toro. (2003). Evolutionary learning of hierarchical decision rules. *IEEE Transactions on Systems, Man, and Cyberneticts - Part B: Cybernetics*, 33(2):324–331.

[5] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, and F. Herrera. (2009). KEEL: A software tool to assess evolutionary algorithms to data mining problems. *Soft Computing*, 13(3):307–318.

[6] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and María Carolina Monard. (2004). A study of the behaviour of several methods for balancing machine learning training data. *SIGKDD Explorations*, 6(1):20–29.

[7] G. Bergmann and G. Hommel. (1988). Improvements of general multiple test procedures for redundant systems of hypotheses. In G. Hommel P. Bauer and E. Sonnemann, editors, *Multiple Hypotheses Testing*, page 100–115. Springer, Berlin.

[8] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena. (2001). *Genetic fuzzy systems: Evolutionary tuning and learning of fuzzy knowledge bases*. World Scientific.

[9] D.R. Cox and D.V. Hinkley. (1974). *Theoretical Statistics*. Chapman and Hall.

[10] J. Demšar. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

[11] T.G. Dietterich. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923.

[12] T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artifical Intelligence*, 89(1–2):31–71.

[13] M. Dorigo and T. Stützle. (2004). *Ant Colony Optimization*. MIT Press.

[14] A.E. Eiben and J.E. Smith. (2003). *Introduction to Evolutionary Computing*. Springer-Verlag.

[15] R. A. Fisher. (1959). *Statistical methods and scientific inference (2nd edition)*. Hafner Publishing Co.

[16] A.A. Freitas. (2002). *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag.

[17] M. Friedman. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701.

[18] S. García, A. Fernández, J. Luengo, and F. Herrera. (2009). A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability. *Soft Computing*, 13(10):959–977.

[19] S. García, A. Fernández, J. Luengo, and F. Herrera. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*. DOI: 10.1016/j.ins.2009.12.010.

[20] S. García and F. Herrera. (2008). An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2579–2596.

[21] S. García, D. Molina, M. Lozano, and F. Herrera. (2009). A study on the use of nonparametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the cec'2005 special session on real parameter optimization. *Journal of Heuristics*, 15: 617–644.

[22] A. Ghosh and L.C. Jain. (2005). *Evolutionary Computation in Data Mining*. Springer-Verlag.

[23] J. Brian Gray and Guangzhe Fan. (2008). Classification tree analysis using TARGET. *Computational Statistics & Data Analysis*, 52(3):1362–1372.

[24] J.J. Grefenstette. (1993). *Genetic Algorithms for Machine Learning*. Kluwer Academic Publishers.

[25] W.G. Cochran G.W. Snedecor. (1989). *Statistical Methods*. Iowa State University Press.

[26] J. Han and M. Kamber. (2006). *Data mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 2nd edition.

[27] D.R. Whitney H.B. Mann. (1947). On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18:50–60.

[28] Haibo He and Edwardo A. Garcia. (2009). Learning from imbalanced data. *IEEE Transactions On Knowledge And Data Engineering*, 21(9):1263–1284.

[29] Y. Hochberg. (1988). A sharper bonferroni procedure for multiple tests of significance. *Biometrika*, 75:800–803.

[30] S. Holm. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70.

[31] R.L. Iman and J.M. Davenport. (1980). Approximations of the critical region of the friedman statistic. *Communications in Statistics*, 9:571–595.

[32] J. Luengo, S. García, and F. Herrera. (2009). A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and nonparametric tests. *Expert Systems with Applications*, 36:7798–7808.

[33] E.G. Mansoori, M.J. Zolghadri, and S.D. Katebi. (2008). SGERD: A steady-state genetic algorithm for extracting fuzzy classification rules from data. *IEEE Transactions on Fuzzy Systems*, 16(4):1061–1071.

[34] P. B. Nemenyi, (1963). Distribution-free multiple comparisons, phd thesis.

[35] S.K. Pal and P.P. Wang. (1996). *Genetic Algorithms for Pattern Recognition*. CRC Press.

[36] R.S. Parpinelli, H.S. Lopes, and A.A. Freitas. (2002). Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 6(4):321–332.

[37] L. Sánchez and I. Couso. (2007). Advocating the use of imprecisely observed data in genetic fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 15(4):551–562.

[38] D. Sheskin. (2006). *Handbook of parametric and nonparametric statistical procedures*. Chapman & Hall/CRC.

[39] S. Sonnenburg, M.L. Braun, Ch.S. Ong, S. Bengio, L. Bottou, G. Holmes, Y. LeCun, K.-R. Müller, F. Pereira, C.E. Rasmussen, G. Rätsch, B. Schölkopf, A. Smola, P. Vincent,

J. Weston, and R. Williamson. (2007). The need for open source software in machine learning. *Journal of Machine Learning Research*, 8:2443–2466.

[40] M.B. Wilk S.S. Shapiro. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611.

[41] Yanmin Sun, Andrew K. C. Wong, and Mohamed S. Kamel. (2009). Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(4):687–719.

[42] K. C. Tan, Q. Yu, and J. H. Ang. (2006). A coevolutionary algorithm for rules discovery in data mining. *International Journal of Systems Science*, 37(12):835–864.

[43] S. Ventura, C. Romero, A. Zafra, J.A. Delgado, and C. Hervás. (2008). Jclec: A java framework for evolutionary computation. *Soft Computing*, 12(4):381–392.

[44] F. Wilcoxon. (1945). Individual comparisons by ranking methods. *Biometrics*, 1:80–83.

[45] I.H. Witten and E. Frank. (June 2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition.

[46] M.L. Wong and K.S. Leung. (2000). *Data mining using grammar based genetic programming and applications*. Kluwer Academic Publishers.

[47] S. P. Wright. (1992). Adjusted p-values for simultaneous inference. *Biometrics*, 48:1005–1013.

[48] J. H. Zar. (1999). *Biostatistical Analysis*. Prentice Hall.

A ALGORITHMS ANALYZED

In the experimental study performed, five evolutionary rule learning methods have been compared in order to test their performance over a wide range of classification problems. A brief description of each one is shown as follows:

- *Ant-Miner*

  Ant-Miner [36] is based on an ant colony system [13]. In this case, the rule stands for the path that the ant must follow. Each ant starts with an empty rule and the decision to add a new term depends on a heuristic function and a pheromone value. The heuristic function is the entropy measure for each attribute-value. There is also a prune step that removes one by one a term of the rule while this process improves the quality of that rule. Once the antecedent of the rule is totally built, the system chooses as the consequent class the majority class of the covered examples. The algorithm then selects the best ant/rule of the current iteration and adds it to the rule-set. This process iterates until all examples are covered (depending on the parameters of the user).

- *CORE*

  CO-Evolutionary Rule Extractor (CORE) [42] evolves a set of rules, which are initialized randomly, using as fitness a combination of the true positive rate and the false positive rate, together with a token competition that reduces the size of the rule-set. It uses a specific regeneration operator that re-initializes those chromosomes that have a fitness below the average. For nominal attributes it uses the one-point crossover, whereas for the numerical attributes it applies a linear combination of the parents.

- *HIDER*

  HIerarchical DEcision Rules (HIDER) [2, 4] uses natural coding (defined by the authors in [2]) to represent each rule. That is, each rule is encoded as IF $x_1 = L_1 \wedge \ldots \wedge x_n = L_n$ THEN c^k,. For numerical attributes, each L_i is a label obtained by means of the natural coding representation, which is a tabular representation of the computed cut-points of a specific discretization method designed for this method [3]. Therefore it is directly translated into an interval-rule by taking the lower and upper cut-points.

  The EA initializes the population by randomly selecting some examples and creating rules that cover these examples. Then, the evolutionary search is run during a specific number of generations, with the guidance of the fitness function which considers both the accuracy and the generalization of the rules. The best rule obtained is added to the final rule set and the examples covered by the rule are removed from the training data set. The process is repeated until there are less than the number of maximum examples allowed by a threshold called the "examples pruning factor".

- *SGERD*

  Steady-State Genetic Algorithm for Extracting Fuzzy Classification Rules From Data (SGERD) [33] is a steady-state GA to generate a prespecified number of Q rules per class following a GCCL approach. In each iteration, parents and their corresponding offspring compete to select the best Q rules for each class. This method also simultaneously uses multiple fuzzy partitions with different granularities and a don't care condition for fuzzy rule extraction.

- *TARGET*

  The Tree Analysis with Randomly Generated and Evolved Trees (TARGET) methodology [23] is a novel approach that uses a GA to build decision trees in which each chromosome represents a complete decision tree. The population is initialized randomly with a pre-specified probability of adding a new condition (node) to the tree. To evaluate the fitness of each chromosome, the authors use a measure based on the correct classifications and the length of the chromosome (number of conditions/nodes).

  The genetic search uses specific operators for crossover and mutation. In the case of crossover, a node swap or a subtree swap is applied. In the case of mutation, there are four possibilities: split set mutation, split rule mutation, node swap mutation and subtree swap mutation. It also uses elitism (called cloning) and reinitialization (called transplantation) to reach a good trade-off between convergence and diversity.

B STATISTICAL PROCEDURES EMPLOYED

When a new method is developed, and in our case is integrated with the KEEL software, it could be interesting to compare it with previous proposals. Making pairwise comparisons allows us to conduct this analysis, but the experiment wise error cannot be previously fixed. Moreover, a pairwise comparison is not influenced by any external factor, whereas in a multiple comparison, the set of algorithms chosen can determine the results of the analysis.

Multiple Comparisons procedures are designed to allow us to fix the Family Wise Error Rate (FWER) before performing the analysis and to take into account all the influences that can exist within the set of results for each algorithm.

In order to perform a multiple comparison, it is necessary to check whether all the results obtained by the algorithms present any inequality. In the case of finding some, we can then find out, by using a post-hoc test, what algorithms partners average results are dissimilar. Next, we describe the non-parametric tests used:

- The first one is the Friedman test [38], which is a non-parametric test equivalent to the repeated-measures ANOVA. Under the null-hypothesis, it states that all the algorithms are equivalent, so a rejection of this hypothesis implies the existence of differences among the performance of all the algorithms studied. After this, a post-hoc test could be used in order to find whether the control or proposed algorithm presents statistical differences with regard to the remaining methods in the comparison. The simplest of them is Bonferroni-Dunn's test, but it is a very conservative procedure and we can use more powerful tests that control the FWER and reject more hypotheses than Bonferroni-Dunn's test; for example, Holm's method [30].

  Friedman's test's way of working is described as follows: It ranks the algorithms for each data set separately, the best performing algorithm is given the rank of 1, the second best rank 2, and so on. In the case of a tie average ranks are assigned.

  Let r_i^j be the rank of the j-th of k algorithms on the i-th of N_{ds} data sets. The Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N_{ds}} \sum_i r_i^j$. Under the null-hypothesis, which states that all the algorithms are equivalent and so their ranks R_j should be equal, the Friedman statistic:

$$\chi_F^2 = \frac{12N_{ds}}{k(k+1)} \left[\sum jR_j^2 - \frac{k(k+1)^2}{4} \right] \tag{1}$$

  is distributed according to χ_F^2 with $k-1$ degrees of freedom.

- The second one of them is the Iman and Davenport test [31], which is a non-parametric test, derived from the Friedman test but, less conservative than the Friedman statistic:

$$F_F = \frac{(N_{ds} - 1)\chi_F^2}{N_{ds}(K-1) - \chi_F^2} \tag{2}$$

  which is distributed according to the F-distribution with $k - 1$ and $(k - 1)(N_{ds} - 1)$ degrees of freedom. Statistical tables for critical values can be found at [38, 48].

- Holm's test [30]: it is a multiple comparison procedure that can work with a control algorithm (which is usually the best according to Friedman rankings computation) and compares it with the remaining methods. The test statistics for comparing the i-th and j-th method using this procedure is:

$$z = (R_i - R_j) / \sqrt{\frac{k(k+1)}{6N_{ds}}} \tag{3}$$

  The z value is used to find the corresponding probability from the table of normal distribution, which is then compared with an appropriate level of confidence α. In the Bonferroni-Dunn comparison, this α value is always $\alpha(k-1)$, but Holm's test adjusts the value for α in order to compensate for multiple comparison and control the FWER.

  Holm's test is a step-up procedure that sequentially tests the hypotheses ordered by their significance. We will denote the ordered p-values by $p_1, p_2, \ldots$, so that $p_1 \le p_2 \le \ldots \le p_{k-1}$. Holm's test compares each p_i with $\alpha(k-i)$, starting from the most significant p value. If p_1 is below $\alpha/(k-1)$, the corresponding hypothesis is rejected and we allow it to compare p_2 with $\alpha/(k-2)$. If the second hypothesis is rejected, the test proceeds with the third, and so on. As soon as a certain null hypothesis cannot be rejected, all the remaining hypotheses are retained as well.

- Hochberg's procedure [29]: It is a step-up procedure that works in the opposite direction to Holm's method, comparing the largest p-value with α, the next largest with $\alpha/2$, the next with $\alpha/3$ and so forth until it encounters a hypothesis that it can reject. All hypotheses with smaller p values are then rejected as well. Hochberg's method is more powerful than Holm's when the hypotheses to test are independent (in this case they are independent given that we are comparing a control algorithm with the remaining algorithms).

The post-hoc procedures described above allow us to know whether or not a hypothesis of comparison of means could be rejected at a specified level of significance α. However, it is very interesting to compute the p-value associated with each comparison, which represents the lowest level of significance of a hypothesis that results in a rejection. In this manner, we can find out whether two algorithms are significantly different and we can also have a metric of how different they are.

Next, we will describe the method used for computing these exact p-values for each test procedure, which are called "adjusted p-values" [47]:

- The adjusted p-value for the Holm procedure is computed by $p_{Holm} = (k - i)p_i$. Once all of them are computed for all hypotheses, it is not possible to find an adjusted p-value for the hypothesis i lower than for the hypothesis $j, j < i$. In this case, the adjusted p-value for hypothesis i is set to the same value as the one associated to hypothesis j.
- The adjusted p-value for the Hochberg method is computed with the same formula as in the Holm procedure, and the same restriction is applied to the process, but in the opposite sense, that is, it is not possible to find an adjusted p-value for the hypothesis i lower than for the hypothesis $j, j > i$.