Genetics-Based Machine Learning for Rule Induction: State of the Art, Taxonomy, and Comparative Study

Alberto Fernández, Salvador García, Julián Luengo, Ester Bernadó-Mansilla, and Francisco Herrera

***Abstract*—The classification problem can be addressed by numerous techniques and algorithms which belong to different paradigms of machine learning. In this paper, 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 paper has 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 genetics-based machine learning (GBML) methods with some classical non-evolutionary algorithms, in order to observe the suitability and high potential of the search performed by evolutionary algorithms and the behavior of the GBML algorithms in contrast to the classical approaches, in terms of classification accuracy.**

***Index Terms*—Classification, evolutionary algorithms, genetics-based machine learning, imbalanced data sets, learning classifier systems, rule induction, taxonomy.**

I. Introduction

CLASSIFICATION in machine learning [1] is a technique that, from a set of n input patterns $w_1, \ldots, w_n$ characterized by i attributes $a_1, \ldots, a_i$, which can include numerical or nominal values, and m classes $c_1, \ldots, c_m$, has the objective of obtaining a system that automatically assigns to each pattern a class label c_n. Specifically, a classifier is a mapping function defined over the patterns, $\mathbb{A}^i \rightarrow \{c_1, \ldots, c_m\}$ ($\mathbb{A}$ stands for the set of attributes) generated by a learning algorithm. This methodology is also known as pattern recognition [2].

Manuscript received March 2, 2009; revised October 21, 2009. Date of publication June 21, 2010; date of current version November 30, 2010. This work was supported by the Spanish Ministry of Science and Technology under Project TIN2008-06681-C06-01/05. The work of J. Luengo was supported by a Financial Peace University Scholarship from the Spanish Ministry of Education and Science.

A. Fernández, J. Luengo, and F. Herrera are with the Department of Computer Science and Artificial Intelligence, University of Granada, Granada 18071, Spain (e-mail: alberto@decsai.ugr.es; julianlm@decsai.ugr.es; herrera@decsai.ugr.es).

S. García is with the Department of Computer Science, University of Jaén, Jaén 23071, Spain (e-mail: sglopez@ujaen.es).

E. Bernadó-Mansilla is with the Grup de Recerca en Sistemes Intelligents, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Barcelona 08022, Spain (e-mail: esterb@salle.url.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2009.2039140

Learning is the building process of the classifier. In this process, the input patterns will be used to train the model in order to describe the attribute space accurately. Supervised learning is carried out when the input training patterns have labeled classes and, in this manner, the learning algorithm is able to build a model that matches regions of the attribute space with an associated class.

Over the years, several approaches have been designed for classification tasks such as decision trees [3], support vector machines [4], instance-based algorithms [5], and probabilistic classifiers (such as Naïve–Bayes [6]), among others (see [7] as an interesting book covering different machine learning techniques).

Rule induction algorithms aim at discovering a description for the target concept in the form of explicit rules formulated in terms of tests for certain values of the attributes [8]. The resulting rule set should be able to correctly recognize instances of the target concept and discriminate them from objects that do not belong to it. The use of rule sets as knowledge representation also makes them very competitive in terms of interpretability since the rules can be read easily by human experts.

Evolutionary rule-based systems [9] are a type of genetics-based machine learning (GBML) that use sets of rules as knowledge representation [10]. One of the strengths of these approaches is the use of the evolutionary algorithms (EAs) as search mechanisms which allows for efficient searches over complex search spaces [11].

At present, there is a huge set of algorithms proposed under the evolutionary rule-based paradigm. Some of them have been in use for the last decade, such as XCS [12], the supervised inductive algorithm (SIA) [13], and genetic-based inductive learning (GIL) [14], and many others have been proposed recently, such as UCS [15] and hierarchical decision rules (HIDER) [16]. Nevertheless, there is no explicit framework where all these methods can be categorized. Furthermore, there are no exhaustive comparisons of the performance of the methods. In many cases, the new algorithms are compared to a limited set of problems and with respect to a small number of learners, usually, of a given type [17], [18]. There is no exhaustive analysis of the algorithms' performance across different families. Our motivation for this paper is to provide a state-of-the-art summary of the GBML algorithms for rule induction in classification tasks, proposing a taxonomy that

defines a general framework within which each algorithm can be placed.

Classically, rule-based systems have been classified into two groups: Michigan-style GBML [19], [20], and Pittsburgh-style GBML [21], [22]. But many approaches do not fall easily into one of these categories and are simply called hybrid approaches. We propose a taxonomy based on the representation of the chromosome of the associated EA. The taxonomy preserves the Michigan and Pittsburgh categories and defines other families in this field, hence including all the state-of-the-art evolutionary GBML methods for rule induction.

The main algorithms proposed in the literature have been selected in order to place them within the framework of the proposed taxonomy. This allows the development of an exhaustive and well-supported analysis to compare the performance of all the algorithms among themselves and with respect to some well-known non-evolutionary algorithms, such as classification and regression tree (CART) [23], automatic qualifier (AQ) [24], CN2 [25], C4.5 [3], C4.5-Rules [26], and Ripper [27]. To address this, a methodology based on hierarchical comparisons is proposed, first identifying the best performers inside each category and then, selecting them as representatives of the category for the cross-family comparison. This hierarchical comparison has been used in order to simplify the empirical study and to focus on the algorithms with better behavior.

In addition to standard classification problems, this paper is additionally concerned with imbalanced data sets, also known as the class imbalance problem, which has been defined as a current challenge of the data mining community [28]. This refers to the cases where one class, usually the one that contains the concept to be learned (the positive class), is underrepresented in the data set [29]. Since this issue is present in many real-world situations, we are particularly interested in analyzing the performance of the algorithms in such cases. We will analyze the results using the original data sets and with preprocessing in order to study two facts: how the different algorithms directly deal with the imbalance problem and the need of using preprocessing algorithms to help improve classification performance in GBML methods.

The experimental framework has been designed so as to provide well-founded conclusion. We use a set of 30 real-world problems, which is usually considered to be a representative sample. We use two different sets of real-world problems. The first one is selected randomly from the University of California Irvine (UCI) repository [30], without any prerequisite, and the second one is selected to be representative of imbalanced class problems. The measures of performance are based on the accuracy rate and Cohen's Kappa metric [31], which is less biased. In the case of imbalanced problems, we use the geometric mean of the accuracy rates per class. The significance of the results is supported by the proper statistical tests as suggested in the literature [32], [33].

Finally, we will discuss some lessons learned and new challenges within this topic, in correspondence to the results acquired in this paper.

The rest of this paper is organized as follows. In Section II, we present our taxonomy proposal for GBML algorithms for rule induction and describe the approaches used in this paper. Section III introduces the experimental framework, that is, the performance measures used in the paper, the benchmark data sets and the parameters, the statistical tests, the classical algorithms employed for comparison, the software tool in which all the selected methods are implemented and that we have used for the design of the experiments, and the description of a Web page associated with the paper[1] that contains complementary material to the experimental study. We develop the empirical analysis for standard classification in Section IV, whereas in Section V this analysis is oriented toward imbalanced data sets. The lessons learned throughout this paper are presented in Section VI. Finally, in Section VII we make our concluding remarks.

II. Taxonomy of Genetics-Based Machine Learning Algorithms for Classification

In this section, we first propose a taxonomy for the different GBML approaches for rule induction, classifying them into different families. Then, we introduce some basic features of the GBML algorithms that enable us to highlight the similarities and differences among the existing methods in this context. Finally, we describe each one of the GBML families, presenting the representative algorithms that we have selected within each category.

A. Taxonomy

Many proposals have been developed under the label of GBML for rule induction. All these approaches share an underlying commonality: the codification of the rule or rule set in a chromosome and the use of an evolutionary algorithm as the search mechanism. However, there are many differences among the algorithms, such as the type of learning scheme (supervised versus reinforcement learning or online versus offline learning), the rule set representation (i.e., a decision list or a set of overlapping rules), the representation of each rule (i.e., the management of numerical and nominal values), or the design of the EA that guides the search.

With the aim of categorizing the GBML methods for rule induction presented in the literature, we distinguish among three different families based on the chromosome codification.

1) *Chromosome = Rule:* In this type of algorithms, each rule is encoded in a single chromosome and the whole population evolves to the final rule-set. We consider three subcategories of methods.

1) The Michigan approach [34], [35], in which the rule-set is incrementally updated through the sequential observation of training examples and their corresponding classification. Eventually, the rule-set is improved by the action of the EA.
2) The iterative rule learning (IRL) approach [13], where the EA learns rule by rule iteratively and removes from the training set the examples covered (matched) by each new rule, in a divide-and-conquer style.
3) The genetic cooperative competitive learning (GCCL) approach [36], where all rules/chromosomes are evolved

[1] http://sci2s.ugr.es/gbml/index.php

TABLE I
SUMMARY OF THE ALGORITHMS EMPLOYED IN THIS PAPER

Algorithm name	Acronym	Family	Reference
XCS	XCS	Michigan	Wilson [12]
UCS	UCS	Michigan	Bernadó-Mansilla and Garrell [15]
Supervised inductive algorithm	SIA	IRL	Venturini [13]
Hierarchical decision rules	HIDER	IRL	Aguilar-Ruiz *et al.* [16]
Co-evolutionary rule extractor	CORE	GCCL	Tan *et al.* [39]
Organizational co-evolutionary algorithm for classification	OCEC	GCCL	Jiao *et al.* [40]
Coverage-based genetic induction	COGIN	GCCL	Greene and Smith [36]
Genetic-based inductive learning	GIL	Pittsburgh	Janikow [14]
Pittsburgh genetic interval rule learning algorithm	Pitts-GIRLA	Pittsburgh	Corcoran and Sen [41]
Data mining for evolutionary learning	DMEL	Pittsburgh	Au *et al.* [42]
Genetic algorithms based classifier system	GASSIST	Pittsburgh	Bacardit *et al.* [43]
Ordered incremental genetic algorithm	OIGA	Pittsburgh	Zhu and Guan [44]
Incremental learning with genetic algorithms	ILGA	Pittsburgh	Guan and Zhu [45]
Hybrid decision tree—genetic algorithm	DT-GA	HEDT	Carvalho and Freitas [46]
Oblique decision tree	Oblique-DT	HEDT	Cantú-Paz and Kamath [47]
Tree analysis with randomly generated and evolved trees	TARGET	HEDT	Gray and Fan [48]

together in the EA. The chromosomes cooperate among themselves to perform the classification task, but the final rule set does not need to include all the rules. Hence, there is a competition to be among the best fitted rules and therefore to remain in the rule base.

2) *Chromosome = Rule Set:* This category is known as the Pittsburgh approach [22], [37]. These types of methods are a result of directly extending genetic algorithm (GAs) [38] to supervised learning problems. The system maintains a population of candidate rule sets whose quality is evaluated with a fitness function that considers different aspects such as the prediction accuracy and the generalization of the rule sets. Once the classifier is trained, the best individual found during the evolutionary process is used to predict the class of unknown examples.

3) *Chromosome = Decision Tree or Tree Rule:* This type of GBML contains mechanisms that combine decision trees with GAs [46], [47]. The underlying idea is to use the search capability of the GA to find a highly accurate final tree. The tree can then be interpreted as a rule set, formed by the disjunction of the rules that are obtained in each branch of the tree. Thus, the chromosome can encode the full tree or a just tree node/rule. We have defined this family as hybrid evolutionary decision trees (HEDTs).

In this paper, we have considered the algorithms that have been published in the most influential journals on the topic and only classical methods proposed in conference contributions. Genetic fuzzy systems [49], methods based on genetic programming [50], and distributed approaches [51], [52] are beyond the scope of this paper.

Table I summarizes the list of selected algorithms of each type. All algorithms were implemented by the authors of this paper, except for genetic algorithms based classifier system (GASSIST) and HIDER whose implementations were directly supplied by their corresponding authors. Furthermore, these methods are included in the knowledge extraction based on evolutionary learning (KEEL) software [53].

B. Preliminaries: Genetics-Based Machine Learning Features

With the aim of giving a general background of the selected algorithms, in this section we describe the main components which are relevant to GBML algorithms and at the same time distinguish them from non-evolutionary learning algorithms. This also leads to providing an educational character to the paper content. We considered the following characteristics.

1) *Chromosome Representation:* As mentioned before, the underlying commonality of GMBL is the use of an EA as the search mechanism. In this respect, the first consideration is the representation of the solution in a population of chromosomes. The choice of the chromosome as codifying a single rule or a rule set determines largely the operation of the algorithm, and that is the motivation for basing our taxonomy on the chromosome representation.

In any case, the rule set in its whole may operate as a set of non-ordered rules, either overlapped or not overlapped, or as a decision list. Also, the inference type (the classification process itself) is very dependent on the type of rule used in the algorithm. The details are as follows.

1) *Non-ordered overlapping rule sets, also called "IF–THEN" rules:* Since the rules can be overlapping, the whole rule set does not necessarily cover the full space of inputs. At classification time, if the example matches more than a rule, a voting method is usually applied, where the output is the class advocated by the majority. Other approaches include a weighted voting, or a measure of the distance to the nearest rule [54]. In the case of a tie, some approaches leave the example as non-classified, while others assign the class randomly.

2) *Non-ordered, non-overlapped rule sets:* This is the case of rule sets obtained from decision trees or axis-parallel hyperplanes, which split the search space into a set of non-overlapping regions. In this sense, the rule set covers all the input space. The classification process with these rules is trivial, since only one rule can match the example.

3) *Ordered rule sets, also called "IF–THEN–ELSE" rules or decision lists [55]:* The classification is simply made using the first rule in the list that matches the example. It is very common to use a "default rule" that explains the input space not covered by any of the rules in the set.

Each rule has the form *condition* $\rightarrow$ *class*, where the condition specifies the set of input states that the rule matches, i.e., the region of the search space covered by the rule, and the class is the classification that the rule advocates for that region. The condition part is usually represented as a conjunction of tests over the attributes. There are two main schemes.

1) *"Attribute <operator> value:"* A test is a condition over an attribute value, which is specified differently depending on whether the attribute is nominal or numerical.

a) *Nominal attributes:* When the attribute can take a value among a list of possible categories, the test over the attribute consists of checking for equality to a given category or belonging to a subset of categories. That is, "<operator>" may be in either $\{=, \neq, \in, \neg \in\}$.

b) *Numerical attributes:* If the attribute can take numerical values, the test checks whether the value belongs to a given interval. That is, the "<operator>" may be "$<, \leq, =, \geq, >$." Interval-rules take the form "Attribute $\in$ [min, max]" which is equal to the case of "Attribute $\geq$ min AND Attribute $\leq$ max."

Disjunction of attributes within a condition can also be codified, although this is rarely implemented in GBML algorithms. The disjunction of attribute values can be alternatively codified with the disjunction of several rules. This may enlarge the rule set, but simplifies the complexity of the rule.

We must also remark that some algorithms treat the nominal attributes as ordered numerical values and consequently use the representations designed for numerical attributes.

2) *Oblique rules:* The tests over the attributes are codified by a hyperplane, with the following form: $\sum_{i=1}^{d} a_i x_i + a_{d+1} > 0$, where the a_i are real-valued coefficients [56], [57] and x_i is the value of the ith attribute. This is an extension of the axis-parallel hyperplanes in the attribute space used by decision trees.

It is common that not all the attributes are relevant to the classification. Some algorithms explicitly allow the codification of irrelevant attributes with the so-called "don't care" symbol. This favors the generalization of the rules. Some approaches enable the chromosome to represent variable length conditions. For algorithms using a fixed chromosome length, the representation of the "don't care" condition depends on the selected rule scheme defined above.

1) For the "Attribute <operator> value" rules, we can represent the "don't care" using an specific special value with the = operator, including all possible values for $\in$ (none for $\neg \in$), using the maximum range or minimum range for the $<, \leq$ and $>, \geq$ operators, respectively, or applying an inversion of operators for the interval-rules, that is, having the "min" value higher than the "max" one.

2) For the oblique rules, it is only needed to assign a weight 0 for those variables that are not included in the antecedent.

2) *Learning Scheme:* This feature refers to the procedure in which the algorithm learns the rule set from a set of pre-classified examples.

Depending on how examples are provided, learning can be performed in an incremental mode (online method), or in a batch mode (offline method). The former scheme implies that the knowledge can be updated as new examples arrived to the system. Michigan approaches usually follow this. The latter model does not easily allow updates of the rule set once the algorithm is trained. If new information is available, the whole rule base should be retrained, often from scratch. Training starts with the whole training data set available. The internal operation of the algorithm may decide whether to use the whole data set in each iteration to build the model (as in the Pittsburgh approach), or incrementally reduce it by deleting the examples already covered, such as in those approaches following a "divide-and-conquer" strategy.

Learning can also happen in a supervised, unsupervised, or reinforcement learning environment. In a supervised environment, the learning algorithm knows the class of the example. In reinforcement learning, the algorithm gets a reward after predicting the class of the example. The algorithm should build its model based on the positive and negative (or absence of) rewards. Unsupervised classification belongs to the case where no external feedback is given to the learner. This case does not fall into the scope of the paper.

3) *Fitness Function:* The assessment of the quality of the rule set is essential to evolve an appropriate model from examples. Since the search of the EA is guided by the fitness function, it must include the quality measures that reflect the type of rule set required. The optimal rule set is usually that with high accuracy and low complexity. Accuracy is usually considered as the percentage of correct classifications, although more sophisticated metrics may arise such as those considering the percentage of classification per class. Complexity is measured with the number of rules or the generalization of the individual rules. We must remark that these measures of quality are applied both at the rule set level and for the individual rules, depending on what the EA evolve.

Fitness may also include niching or sharing techniques, or token competition as in [58]. These types of techniques enable us to adjust the global fitness and optimize the search toward different areas in the problem space by penalizing those solutions/rules which are similar or which cover the same training examples.

4) *Design of the EA:* As mentioned before, all GBML algorithms have as a common characteristic the use of a search process based on an EA to learn the final rule set. They usually apply a GA which can be generational or steady-state. In the former case, the population is replaced completely iteration by

iteration, possibly with an elitism scheme in which the best individual(s) remain(s) in the new population. The steady-state model only modifies a subset of individuals (normally only two) during each iteration.

5) *Specific Operators:* Each new proposal in the context of GBML algorithms not only implies the choice of representation, learning methodology, and design of the EA scheme, but also the use of specific operators for the knowledge discovery. These usually represent the most significant novelty of the approach and accentuate the differences among the methods.

These specific operators include the heuristics to initialize the population, the use of covering algorithms, and the crossover and mutation operators. Since they are particular to each algorithm, we will focus on this issue on each single description when needed.

C. Taxonomy Families and Algorithms' Description

The description of the methods used in this paper is summarized in a table for each one of the proposed families, in which we detail the main characteristics with regard to the GBML features stated previously. Therefore, the reader can observe the similarities and differences among the algorithms of the same category or between any two methods. Afterward, we present a brief description of the intrinsic operators associated with each one of the algorithms to provide the specific details for each of them.

1) *Michigan Approach:* Michigan-style algorithms [34], [35] evolve a set of rules called "classifiers" which are incrementally updated through the sequential observation of training examples. The best "classifiers," which are those that correctly cover a higher proportion of training examples, are selected and propagated over other less useful "classifiers," leading to an improvement in the overall system performance.

Some of the first developments of Michigan-style GBML are SCS [59] and NewBoole [60]. In this paper, we select XCS, as it is the most influential algorithm in this family of classifiers, and UCS, a system that specializes XCS for supervised learning. The features of both algorithms are summarized in Table II.

TABLE II
FEATURES OF THE MICHIGAN ALGORITHMS

Features	XCS	UCS
Learning method	Online Reinforcement learning	Online Supervised learning
Type of rule-set	IF–THEN	IF–THEN
Inference type	Weighted voting	Weighted voting
Type of rule	Conjunction of atts.	Conjunction of atts.
Nominal representation	None: transformation to ordered numerical values	None: transformation to ordered numerical values
Numerical representation	Interval-rule	Interval-rule
"Don't Care" management	Complete interval	Complete interval
Fitness	Inverse of prediction error. Sharing	Accuracy Sharing
EA type	Steady-state GA	Steady-state GA
EA features	Roulette-wheel selection 2-Point crossover Random mutation	Roulette-wheel selection 2-Point crossover Random mutation
Chromosome representation	Real coding normalized to [0, 1] Fixed length	Real coding normalized to [0, 1] Fixed length

TABLE III
FEATURES OF THE IRL APPROACHES

Features	SIA	HIDER
Learning method	Batch Divide-and-conquer	Batch Divide-and-conquer
Type of rule-set	IF–THEN	IF–THEN–ELSE
Inference type	Distance to the nearest rule	Decision list
Type of rule	Conjunction of atts.	Conjunction of atts.
Nominal representation	Operator "="	Operator "="
Numerical representation	Interval-rule	Natural coding Discretization
"Don't Care" management	Special value	Complete interval
Fitness	Accuracy and complexity	Accuracy and coverage
EA type	Steady-state GA	Generational GA
EA features	Random selection Uniform crossover Generalization operator (see description)	Roulette-wheel selection. Specific crossover and mutation elitism
Chromosome representation	Real and binary coding Fixed length	Natural coding (see description) Fixed length

1) *XCS:* XCS [12], [61] was designed for reinforcement learning, although it can be used for pattern recognition by considering that a classification problem is a reinforcement problem in which maximum rewards are given to correct classifications and low rewards correspond to incorrect classifications.

Learning takes place incrementally. In each iteration, an example is provided to the system. Then, XCS finds the matching rules and randomly chooses one of the possible classes. If no rules match, then covering is applied. It creates a new rule with a condition, which is generalized from the attribute values of the example, and a random class. Once the class is decided, a reward is received. This reward is used to update the quality of the rules that advocated that particular class. Eventually, the GA is applied to the rules that proposed the chosen class as described in [62]. The GA selects two parents (two rules) based on their fitness, which is a measure of the accuracy of the reward prediction relative to the accuracies of the overlapping rules. The rules are crossed and mutated and the final offspring are introduced into the population, deleting other rules if the population is full. The rules follow the interval rule representation, which was first introduced in XCS in [63].

2) *UCS:* UCS [15] is a method derived from XCS. It inherits the main features of XCS, but mainly differs in two respects. First, the learning interaction is adjusted to a supervised learning scheme. This means that each example shown to the system comes along with the class. Then, UCS uses the class information to build the set of correct rules. The rules that belong to this set get their fitness improved, while rules that matched the

TABLE IV
FEATURES OF THE GCCL APPROACHES

Features	CORE	OCEC	COGIN
Learning method	Batch	Batch	Batch
Type of rule-set	IF–THEN–ELSE	IF–THEN	IF–THEN–ELSE
Inference type	Decision list	Weighted voting	Decision list
Type of rule	Conjunction of attributes	Conjunction of attributes	Conjunction of attributes
Nominal representation	Operators "= and $\neq$"	Operator "="	Operator "$\in$"
Numerical representation	All comparisons: "$<, \leq, =, \geq, >$" and interval-rule	None: discretization required	None: discretization required
"Don't Care" management	Absence of conditions	Absence of conditions	Absence of conditions Special symbol
Fitness	Accuracy with token competition	Accuracy and complexity	Accuracy and complexity with token competition
EA type	Generational GA	Generational GA	Generational GA
EA features	Tournament selection 1-Point and linear crossover Simple random mutation No elitism	Random selection Special crossover (see description) No mutation Elitism	Random selection 1-Point crossover No mutation Elitism based in token competition
Chromosome representation	Real and binary coding Variable length	Binary coding Variable length	Binary coding Variable length

example but not the class get their fitness reduced. The second main difference with XCS is on fitness. In UCS, fitness is directly the proportion of correct predictions of the rule with respect to the total number of matches, whereas fitness in XCS is a windowed average of the accuracy of the reward prediction.

2) *Iterative Rule Learning Approaches:* IRL GBML systems use a divide-and-conquer methodology to create an ordered list of rules [13]. The system iteratively invokes an EA where the best individual returned is added to the end of a list of rules and all the matching examples are removed from the training data set. This process is repeated until the training data set is empty.

In this case, we selected two algorithms. First, we considered a classical approach, the SIA algorithm and then, a recent representative method of this type: HIDER. Their characteristics are shown in Table III.

1) *SIA:* SIA [13] is a classical IRL mechanism. The main procedure of SIA is detailed as follows: first, a training sample which has not been classified yet (defined as "uncovered") is selected, and the most specific rule that matches that example is built. Then the condition part of the rule is generalized using a GA with a fitness function based on a linear combination of the number of classifications and misclassifications and the generalization of the rule.

The generalization operator works by enlarging the bounds of the conditions in the case of a numerical attribute and by converting the condition into a "don't care" for nominal attributes. The GA runs iteratively under it reaches the stop condition (number of iterations without generating a better rule). The best rule in the final population is then added to the rule set and the examples covered by it are removed from the training set. The process is repeated until no more examples remain uncovered.

2) *HIDER:* HIDERs [16], [64] use natural coding (defined by the authors in [16]) 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 [65]. 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, similarly to SIA. The process is repeated until there are less than the number of maximum examples allowed by a threshold called "examples pruning factor."

3) *Genetic Cooperative Competitive Learning Approaches:* In this approach, each individual codifies a rule, and the whole rule set is evolved simultaneously. Thus, rules should cooperate among them to get an optimal rule set jointly, and at the same time, rules compete against each other to survive in the population. Population diversity must be enforced to avoid individuals converging to the same area of the search space.

The algorithms selected for this family are co-evolutionary rule extractor (CORE), organizational co-evolutionary algorithm for classification (OCEC), and coverage-based genetic induction (COGIN). Next, we describe each algorithm, whose main features are shown in Table IV.

1) *CORE:* CORE [39] 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

TABLE V
FEATURES OF THE PITTSBURGH APPROACHES

Features	GIL	Pitts-GIRLA	DMEL	GASSIST	OIGA	ILGA
Learning method	Batch	Batch	Batch	Batch	Batch	Batch
Type of rule-set	IF–THEN–ELSE	IF–THEN	IF–THEN	IF–THEN–ELSE	IF–THEN	IF–THEN
Inference type	Decision list	Voting	Weighted voting	Decision list	Voting	Voting
Type of rule	Conjunction of atts.	Conjunction of atts.	Conjunction of atts.	Conjunction and disjunction of atts.	Conjunction of atts.	Conjunction of atts.
Nominal representation	Operator "$\in$"	None: transformation to ordered numerical values	Operator "$=$"	Operator "$\in$"	None: transformation to ordered numerical values	None: transformation to ordered numerical values
Numerical representation	None: discretization required	Interval-rule	None: discretization required	Interval-rule	Interval-rule	Interval-rule
"Don't Care" management	Absence of conditions All values for a condition	Reverse interval	None	Absence of conditions	Reverse interval	Reverse interval
Fitness	Accuracy and complexity	Accuracy	Accuracy	Accuracy and complexity	Accuracy	Accuracy
EA type	Generational GA	Generational GA	Steady-state GA	Generational GA	Steady-state GA	Steady-state GA
EA features	Special operators (see description) No elitism	Random selection 1-Point crossover Random and "creep" mutation Elitism	Roulette-wheel selection 1 and 2-Point crossover Mutation based on local search	Tournament selection Special crossover and mutation (see description) Elitism	Roulette-wheel selection 1-Point crossover Random mutation	Roulette-wheel selection 1-Point crossover Random mutation
Chromosome representation	Binary coding Variable length	Real coding Fixed length	Binary coding Variable length	Binary coding and ADI representation Variable length	Real coding Fixed length	Real coding Fixed length

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.

2) *OCEC:* OCEC [40] creates groups of similar examples with the same class label and then builds a rule for each group discovered. Thus, the chromosome has a variable length and it stores the position of the example in the training set. At classification time, the most general rule that covers the example defines its class.

Three specific evolutionary operators are used in this algorithm, the "migrating operator" that moves a predefined number of examples from one chromosome to another, the "exchanging operator" that exchanges a predefined number of examples between two chromosomes, and the "merging operator" that joins two chromosomes. It also uses an elitist scheme in which the best pair between the parents and the offspring remains in the population. The fitness function depends on the number of conditions and the confidence of the rule.

3) *COGIN:* COGIN [36] manages a variable number of rules, which are initialized randomly until they cover all the examples of the training set, making half of the conditions "don't care." It generates a new offspring population by randomly selecting two parents and applying a one-point crossover. Then it applies a token competition to remove those rules that do not match any example already covered by better rules. The fitness is based on a linear combination of the entropy and the generality of the rule (number of active bits).

4) *Pittsburgh Approaches:* In the Pittsburgh-style GBML systems, the GA encodes the rule set directly into the chromosome. In this manner, each individual represents the whole solution and the different rule sets are evolved until convergence.

There are a large number of Pittsburgh style approaches in the literature. In this paper, we consider the following methods: GIL, Pittsburgh genetic interval rule learning algorithm (Pitts-GIRLA), data mining for evolutionary learning (DMEL), GASSIST, ordered incremental genetic algorithm (OIGA), and incremental learning with GAs (ILGA). As with the previous families introduced in this section, we summarize the main characteristics of these algorithms in Table V.

1) *GIL:* The GIL [14] learns rules for each class sequentially, starting for the minority class, leaving the majority class as a default rule. Thus, it runs the GIL algorithm $n-1$ times, where n is the number of classes. For a given run i, class i is the positive class and all the other classes represent the negative class. Thus, a multiple class problem is converted into $n-1$ different concept learning problems.

The initialization of the chromosomes is carried out randomly for half of the population and using some of the training examples for the other half. The fitness of each individual is based on four conditions: accuracy rate, completeness (covering all positive examples), correctness (minimization of the number of negative examples covered), and complexity of the rule (computed as the number of rules and conditions). GIL implements 14 genetic operators belonging to three kinds of levels. Six of them work at the rule set level and consist of adding and exchanging rules to the rule set, and generalizing or specialising the rule set. Five operators work at the rule

level by adding, removing, or exchanging conditions. The final three operators work at the condition level by adding or removing values. All these operators have an associated probability which is auto-adapted according to the needs of the learning process, that is, to favor the generalization or specialization of the rule set.

2) *Pitts-GIRLA:* The Pitts-GIRLA [41] initializes all chromosomes at random, where a "don't care" condition occurs when the lower bound of the interval associated with a given variable is higher than the upper bound. Apart from the conventional 1-point crossover and random mutation, it uses a so-called "creep" mutation which is applied to the bounds of each attribute condition and increments or decrements them with a fraction of the valid range for the attribute value.

3) *DMEL:* DMEL [42] starts with the generation of an initial set of first-order rules (rules with one condition) using a probabilistic induction technique and based on these rules, rules of a higher order (two or more conditions) are obtained iteratively. Then DMEL begins an evolutionary learning process by generating an initial population of individuals by randomly combining the rules in the rule base of order $l-1$ to form a set of rules of order l. The consequent of the rule is not encoded in the chromosome but determined by the computation of the fitness value. Finally, the classification is made by the rule with the highest fitness that matches the example.

4) *GASSIST:* GASSIST [43], [66], [67] inherits from the GABIL algorithm [68]. The GASSIST algorithm initializes the population at random and uses as a fitness function the minimum description length principle [69], which takes into account both the accuracy and the complexity of the rule set.

The representation of the chromosome differs for nominal and numerical attributes. For the first type, it uses a string of bits, one per each category allowed for that attribute, whereas for continuous variables it applies an adaptive discretization intervals rule representation [66]. This representation consists, in general terms, of a hierarchical uniform-width discretization of each attribute having different cut-points for the same attribute in different rules. We should point out that this representation can be directly translated into an interval-rule by taking the lower and upper cut-points of each attribute.

It uses the multiple-point crossover operator inherited from GABIL that forces the selected cut points in both parents to be in the same position of the variable so that semantically correct offspring are obtained. The mutation operator randomly adds or removes one value of a given variable. GASSIST introduces a new deletion operator that removes rules from individuals to limit the rule set size. This operator is activated after a predefined number of iterations and removes the rules of an individual that do not match any input example.

5) *OIGA:* OIGA [44] carries out the learning in two steps: first, it learns one-condition rules (generated randomly) for each attribute, and then optimising their values using a GA. Then, once all the attributes have been explored separately, OIGA joins the obtained one-condition rule sets ordered by their fitness. This process adds the one-condition rule sets to the current increasing rule sets one by one, randomly merging the rules with the same consequent from the best one-condition rule-set.

This incremental process involves a new optimization procedure in each merging step, in which the current increasing rule sets are refined by means of a GA (with the same features as the previous one). This process stops once all attributes have been added, obtaining a rule set with all the attributes. Both genetic processes use the accuracy rate as fitness.

6) *ILGA:* ILGA [45] is an evolution of the OIGA learning process. ILGA expands the possibilities of the incremental process, by means of using, as a seed for all the initial population, either the best one-condition rule set (as OIGA does) or the whole population of chromosomes in the current solution. It also introduces a bias in the mutation and crossover operators. This bias affects the "old" part of the chromosome, that is, the genes added in the previous iterations but not in the current one. If the randomly chosen point for mutation or crossover is located in the "old" genes, the corresponding rates may be reduced with a reduction rate. The motivation behind this is that ILGA tends to preserve the structure of old elements and explore more of the combination between old and new elements.

5) *Hybrid Evolutionary Decision Trees:* Apart from the classical GBML approaches described previously, in this paper we study some hybrid approaches that unify the use of decision trees with GAs, selecting the most important algorithms from the literature, decision tree-genetic algorithm (DT-GA) and oblique decision tree (Oblique-DT), and including the tree analysis with randomly generated and evolved trees (TARGET) algorithm as a recent representative. The features of these algorithms are listed in Table VI.

1) *DT-GA:* The hybrid DT-GA method [46] discovers rules in two training phases. In the first phase, it runs C4.5 with the whole training set and transforms the resulting tree into an "IF–THEN" set of rules. The examples covered by each rule are considered either as a small disjunct or as a large disjunct, depending on whether their number is smaller than or equal to a given threshold. The second phase consists of using a GA with the joint of the instances in the small disjuncts as a "second training set." To predict the output class of an example, it is pushed down the decision tree until it reaches a leaf node, as usual. If the leaf node is a large disjunct, the example is assigned the class predicted by that leaf node. Otherwise, the example is assigned the class of one of the small-disjunct rules discovered by the GA.

The rules from the GA are randomly initialized selecting a value for each attribute from the examples of the "second training set." The chromosome representation

TABLE VI
FEATURES OF THE HEDT APPROACHES

Features	DT-GA	Oblique-DT	TARGET
Learning method	Batch Divide-and-conquer	Batch	Batch
Type of rule-set	Axis-parallel hyperplanes and IF–THEN	Oblique-rules	Oblique-rules and IF–THEN
Inference type	Matching rule	Matching rule or weighted voting (see description)	Matching rule
Type of rule	Conjunction of atts.	Conjunction of linear combination of atts.	Conjunction and linear combination of atts.
Nominal representation	Operator "$\in$"	None: transformation to ordered numerical values	Operator "$\in$"
Numerical representation	Operators "$<$ and $\geq$"	Linear combination	Linear combination (up to three variables)
"Don't Care" management	Special symbol	Not implicit	Not implicit
Fitness	Accuracy	Accuracy	Accuracy and complexity
EA type	Generational GA	Generational GA	Generational GA
EA features	Tournament selection 1-Point crossover Random mutation Elitism	Tournament selection Uniform crossover No mutation Elitism	Roulette-wheel selection Special crossover and mutation (see description) Elitism and reinitialization
Chromosome representation	Binary and real coding Fixed length	Real coding Fixed length	Binary and real coding Variable length

encodes the conditions for all attributes (nominal and numerical), together with a bit that specifies the "don't care" condition. The rule consequent is not encoded into the genome, but dynamically chosen as the most frequent class in the set of examples covered by that rule's antecedent. The fitness function is given by a quadratic version of the geometric mean of the true rates. It also uses a specific rule pruning operator that transforms a condition into a "don't care" based on the accuracy rate associated with each attribute. The stopping criterion is given by a threshold value of the remaining examples in the "second training set."

2) *Oblique-DT:* The Oblique-DT algorithm [47] extends the classical OC1 method [70] (a greedy optimizer for oblique-type decision trees) to find oblique partitions by means of a standard generational GA.

Each chromosome encodes the coefficient of the linear combination that defines the oblique-hyperplane. To initialize the population, we first compute the best axis-parallel hyperplane using an impurity measure defined below, and we copy it to 10% of the initial population. The remainder of the population is initialized randomly with coefficients $a_i \in [-200, 200]$. The fitness value is computed as the impurity of a split at each tree node using the twoing rule [23], which is based on the balance of the number of examples on the left and right of the split.

3) *TARGET:* The TARGET methodology [48] 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.

III. EXPERIMENTAL FRAMEWORK

In this section, we first describe the measures employed to evaluate the performance of the algorithms analyzed in this paper, both for standard classification problems and for imbalanced data sets (Sections III-A and III-B, respectively). Then, we introduce some of the best known state-of-the-art non-evolutionary rule learning algorithms which are included in the study (Section III-C). Next, we provide details of the real-world problems chosen for the experimentation and the configuration parameters of the GBML and non-evolutionary methods (Sections III-D and III-E). Finally, we present the statistical tests applied to compare the results obtained with the different classifiers (Section III-F), we describe the KEEL software tool, which was used to run all the algorithms and to design the set of experiments (Section III-G) and we introduce the information shown at the Web page associated with the paper (Section III-H).

A. Performance Measures for Standard Classification

In this paper, we deal with two-class and multiclass data sets. However, in the literature, most of the performance measures are only designed for two-class problems [71], [72].

TABLE VII
CONFUSION MATRIX FOR AN N-CLASS PROBLEM

Correct Class	Predicted Class				
	C_1	C_2	...	C_m	Total
C_1	h_{11}	h_{12}	...	h_{1m}	T_{r1}
C_2	h_{12}	h_{22}	...	h_{2m}	T_{r2}
⋮			⋱		⋮
C_m	h_{1m}	h_{2m}	...	h_{mm}	T_{rm}
Total	T_{c1}	T_{c2}	...	T_{cm}	T

Well-known accuracy measures for two-class problems are: classification rate, precision, sensitivity, specificity, G-mean [73], F-score [74], area under the receiver operating characteristic (ROC) curve [75], Youden's index γ [76], and Cohen's kappa [77].

Some of the two-class accuracy measures have been adapted for multiclass problems. For example, a recent paper [78] proposes an approximating multiclass ROC analysis, which is theoretically possible, but the exponential computational complexity as a function of the number of classes is restrictive. Two measures are widely used because of their simplicity and successful application for both binary and multiclass problems. We refer to classification rate and Cohen's kappa measures, which we will now explain as follows.

1) *Classification Rate:* It is the number of successful hits (correct classifications) relative to the total number of classifications. It has been by far the most commonly used metric for assessing the performance of classifiers for years [79]–[81].

2) *Cohen's Kappa:* It is an alternative measure to *classification rate*, since it compensates for random hits [31], [77]. In contrast to classification rate, kappa evaluates the portion of hits that can be attributed to the classifier itself (i.e., not to mere chance), relative to all the classifications that cannot be attributed to chance alone. An easy way of computing Cohen's kappa is by making use of the resulting confusion matrix (Table VII) in a classification task. With (1), we can obtain Cohen's kappa

$$kappa = \frac{n\sum_{i=1}^{m} h_{ii} - \sum_{i=1}^{m} T_{ri}T_{ci}}{n^2 - \sum_{i=1}^{m} T_{ri}T_{ci}} \quad (1)$$

where h_{ii} is the cell count in the main diagonal (the number of true positives for each class), n is the number of examples, m is the number of class labels, and T_{ri}, T_{ci} are the rows' and columns' total counts, respectively ($T_{ri} = \sum_{j=1}^{m} h_{ij}$, $T_{ci} = \sum_{j=1}^{m} h_{ji}$).

Cohen's kappa ranges from -1 (total disagreement) through 0 (random classification) to 1 (perfect agreement). Being a scalar, it is less expressive than the ROC curves applied to binary-class cases. However, for multiclass problems, kappa is a very useful, yet simple, meter for measuring a classifier's classification rate while compensating for random successes.

The main difference between the classification rate and Cohen's kappa is the scoring of the correct classifications. Classification rate scores all the successes over all classes, whereas Cohen's kappa scores the successes independently for each class and aggregates them. The second way of scoring is less sensitive to randomness caused by a different number of examples in each class.

TABLE VIII
SUMMARY DESCRIPTION OF STANDARD DATA SETS

id	Data Set	#Ex.	#Atts.	#Num.	#Nom.	#Cl.
aba	abalone	418	8	7	1	28
aus	australian credit approval	690	14	8	6	2
bal	balance scale	625	4	4	0	3
bre	breast cancer	286	9	0	9	2
bup	liver disorders [Bupa]	345	6	6	0	2
car	car evaluation	1728	6	0	6	4
cle	cleveland	297	13	13	0	5
con	contraceptive method choice	1473	9	6	3	3
crx	japanese credit screening	125	15	6	9	2
der	dermatology	366	33	1	32	6
eco	ecoli	336	7	7	0	8
fla	solar flare	1389	10	0	10	6
ger	german credit data	1000	20	6	14	2
gla	glass identification	214	9	9	0	7
hab	haberman	306	3	3	0	2
hea	heart	270	13	6	7	2
hep	hepatitis	155	19	6	13	2
iri	iris	150	4	4	0	3
lym	lymphography	148	18	3	15	4
mag	magic	1902	10	10	0	2
new	new-thyroid	215	5	5	0	3
nur	nursery	1296	8	0	8	5
pen	pen-based recognition	1099	16	16	0	10
pim	pima	768	8	8	0	2
rin	ring	740	20	20	0	2
tic	tic-tac-toe endgame	958	9	0	9	2
veh	vehicle	846	18	18	0	4
win	wine	178	13	13	0	3
wis	wisconsin	683	9	9	0	2
zoo	zoo	101	17	0	17	7

B. Imbalanced Data Sets: Formulation and Performance Measures

In this paper, we perform a comparative study of the GBML algorithms for rule induction on standard classification, but we also focus on one of the emergent challenges in data mining [28]: the problem of imbalanced data sets [29]. In this context, one class is represented by only a few examples (known as positive class), whereas the other is described by many instances (negative class). The minority class is usually the target concept from the point of view of learning and, for this reason, the cost derived from a misclassification of one of the examples of this class is higher than that of the majority class. This problem is very representative since it appears in a variety of real-world applications including, but not limited to, fraud detection [82], risk management [83], and medical applications [84].

Standard classifier algorithms tend to be biased toward the negative class, since the rules that predict the highest number of examples are rewarded by the accuracy metric. A related issue of this type of problem is the presence of small disjuncts (small regions in the attribute space) which are rarely covered by the learners due to their preference for evolving rules covering larger regions [85]. Furthermore, overlapping regions between positive and negative examples pose an added difficulty for the identification and discovery of rules covering the positive, and under-represented, samples [86].

TABLE IX
SUMMARY DESCRIPTION OF IMBALANCED DATA SETS

Data Set	#Ex.	#Atts.	Class (Min., Maj.)	%Class Min.
Glass1	214	9	(build-win-non_float-proc; remainder)	35.51
Ecoli0vs1	220	7	(im; cp)	35.00
Wisconsin	683	9	(malignant; benign)	35.00
Pima	768	8	(tested-positive; tested-negative)	34.84
Iris0	150	4	(Iris-Setosa; remainder)	33.33
Glass0	214	9	(build-win-float-proc; remainder)	32.71
Yeast1	1484	8	(nuc; remainder)	28.91
Vehicle1	846	18	(Saab; remainder)	28.37
Vehicle2	846	18	(Bus; remainder)	28.37
Vehicle3	846	18	(Opel; remainder)	28.37
Haberman	306	3	(Die; Survive)	27.42
Glass0123vs456	214	9	(non-window glass; remainder)	23.83
Vehicle0	846	18	(Van; remainder)	23.64
Ecoli1	336	7	(im; remainder)	22.92
New-thyroid2	215	5	(hypo; remainder)	16.89
New-thyroid1	215	5	(hyper; remainder)	16.28
Ecoli2	336	7	(pp; remainder)	15.48
Segment0	2308	19	(brickface; remainder)	14.26
Glass6	214	9	(headlamps; remainder)	13.55
Yeast3	1484	8	(me3; remainder)	10.98
Ecoli3	336	7	(imU; remainder)	10.88
Page-blocks0	5472	10	(remainder; text)	10.23
Vowel0	988	13	(hid; remainder)	9.01
Glass2	214	9	(Ve-win-float-proc; remainder)	8.78
Ecoli4	336	7	(om; remainder)	6.74
Glass4	214	9	(containers; remainder)	6.07
Abalone9vs18	731	8	(18; 9)	5.65
Glass5	214	9	(tableware; remainder)	4.20
Yeast2vs8	482	8	(pox; cyt)	4.15
Yeast4	1484	8	(me2; remainder)	3.43
Yeast5	1484	8	(me1; remainder)	2.96
Yeast6	1484	8	(exc; remainder)	2.49
Abalone19	4174	8	(19; remainder)	0.77

We will develop an empirical analysis in the context of imbalance classification for binary data sets, since all studies in this area are focused on two-class problems [87]. The modification of the algorithms to better adapt to imbalanced problems is beyond the scope of this paper. Instead, we aim to establish a fair comparison of all the algorithms under the same conditions in a two-objective way.

1) To analyze how the different algorithms directly deal with the imbalance problem.
2) To apply a preprocessing step that balances the proportion of positive and negative examples in the training data set. Specifically, we use the synthetic minority over-sampling technique (SMOTE) [88], which has been generally demonstrated to improve the performance of rule-based classifiers and decision trees. Thus, we aim at showing the positive synergy between this preprocessing method and the algorithms of this paper.

Regarding the performance measures, the accuracy rate cannot be considered for imbalanced data sets, since it does not distinguish between the number of correct classifications of the different classes, which may lead to erroneous conclusions in this case. As a classical example, if the ratio of imbalance presented in the data is 1:100, i.e., there is one positive instance versus 99 negatives, a classifier that obtains an accuracy rate of 99% is not truly accurate if it does not cover correctly any minority class instance. Because of this, instead of using accuracy, other metrics are considered. Specifically, we use in this paper the geometric mean of the true rates [73], which can be defined as

$$GM = \sqrt{\frac{TP}{TP + FN} \cdot \frac{TN}{TN + FP}} \tag{2}$$

where TP/TN stands for the correct classifications of the positive/negative class, and FN/FP for the misclassifications of the positive/negative class, respectively. This metric considers the accuracy on each one of the two classes with the same weight.

C. Selected Classical Machine Learning Algorithms

In our empirical study our aim is to compare the algorithms of the different GBML families for rule induction, and also to analyze their behavior in contrast to some classical and well-known classification algorithms.

Specifically, the selected algorithms are the following ones.

1) CART analysis [23] is a form of binary recursive partitioning. The term binary implies that each group of data/individuals, represented by a node in a decision tree, can only be split into two groups.
2) AQ [24] is a classification model based on covering rules (divide-and-conquer). It implements the *STAR* method of inductive learning. When a decision rule is being built, AQ performs a heuristic search through a space of logical expressions to determine those that account for all positive and not negative examples.
3) CN2 [25] induces an ordered list of classification rules from examples using the entropy as its search heuristic.

TABLE X

PARAMETER SPECIFICATION FOR THE ALGORITHMS EMPLOYED IN THE EXPERIMENTATION

Michigan Algorithms	
Algorithm	Parameters
XCS	Number of explores = 100 000, population size = 6400, α = 0.1, β = 0.2, δ = 0.1, ν = 10.0, θ_{mna} = 2, θ_{del} = 50.0, θ_{sub} = 50.0, ϵ_0 = 1, do Action Set Subsumption = false, fitness reduction = 0.1, p_I = 10.0, F_I = 0.01, ϵ_I = 0.0, γ = 0.25, χ = 0.8, μ = 0.04, θ_{GA} = 50.0, doGASubsumption = true, type of selection = Roulette wheel selection (RWS), type of mutation = free, type of crossover = 2 point, $P_{\#}$ = 0.33, r_0 = 1.0, m_0 = 0.1, l_0 = 0.1, doSpecify = false, nSpecify = 20.0, pSpecify = 0.5
UCS	Number of explores = 100 000, population size = 6400, δ = 0.1, acc_0 = 0.99, P_{cross} = 0.8 P_{mut} = 0.04, θ_{GA} = 50.0, θ_{del} = 50.0, θ_{sub} = 50.0, doGASubsumption = true, r_0 = 0.6, type of selection = RWS, type of mutation = free, type of crossover = 2 point, m_0 = 0.1
IRL Algorithms	
Algorithm	Parameters
SIA	Number of iterations = 200, α = 150, β = 0, threshold strength = 0
HIDER	Pop. size = 100, number of gen. = 100, mutation prob.= 0.5, percentage of crossing = 80, extreme mutation prob. = 0.05, Prune examples factor = 0.05, penalty factor = 1, error coefficient = 0
GCCL Algorithms	
Algorithm	Parameters
CORE	Pop. size = 100, co-population size = 50, gen. limit = 100, number of co-populations = 15, crossover rate = 1.0, mutation prob. = 0.1, regeneration prob. = 0.5
OCEC	Number of total generations = 500, number of migrating/exchanging members = 1
COGIN	Misclassification error level = 2, gen. limit = 1000, crossover rate = 0.9, negation bit = yes
Pittsburgh Algorithms	
Algorithm	Parameters
GIL	Pop. size = 40, number of gen. = 1000, w_1 = 0.5, w_2 = 0.5, w_3 = 0.01, rules exchange = 0.2, rule exchange selection = 0.2, rules copy = 0.1, new event = 0.4, rules generalization = 0.5 rules drop = 0.5, rules specialization = 0.5, rule split = 0.005, nominal rule split = 0.1, linear rule split = 0.7, condition drop = 0.1 conjunction to disjunction = 0.02, introduce condition = 0.1, rule directed split = 0.03, reference change = 0.02, reference extension = 0.03, reference restriction = 0.03, condition level prob. = 0.5, lower threshold = 0.2, upper threshold = 0.8
Pitts-GIRLA	Number of rules: 30, number of generations: 10 000, population size: 61 chromosomes, crossover probability: 0.7, mutation probability: 0.5.
DMEL	Pop. size = 30, crossover prob. = 0.5, mutation prob. = 0.0001, number of gen. = 1000
GASSIST-ADI	Threshold in hierarchical selection = 0, iteration of activation for rule deletion operator = 5, iteration of activation for hierarchical selection = 24, minimum number of rules before disabling the deletion operator = 12, minimum number of rules before disabling the size penalty operator = 4, number of iterations = 750, initial number of rules = 20, population size = 400, crossover probability = 0.6, probability of individual mutation = 0.6, probability of value 1 in initialization = 0.90, tournament size = 3, possible size in *micro-intervals* of an attribute = {4, 5, 6, 7, 8, 10, 15, 20, 25}, maximum number of intervals per attribute = 5, p_{split} = 0.05, p_{merge} = 0.05, probability of reinitialize begin = 0.03, probability of reinitialize end = 0, Use MDL = true, iteration MDL = 25, initial theory length ratio = 0.075, weight relaxation factor = 0.90, class initialization method = cwinit, default class = auto
OIGA	Crossover prob. = 1.0, mutation prob. = 0.01, pop. size = 200, number of rules = 30, stagnation = 30, generations = 200, survivors percent = 0.5, attribute order = descendent
ILGA	Crossover prob. = 1.0, mutation prob. = 0.01, pop. size = 200, number of rules = 30, stagnation = 30, generations = 200, survivors percent = 0.5, attribute order = descendent, crossover reduction rate = 0.5, mutation reduction rate = 0.5, incremental strategy = 1
Hybrid Evolutionary Decision Trees	
Algorithm	Parameters
DT-GA	Confidence = 0.25, instances per leaf = 2, threshold S to consider a small disjunct = 10, number of gen. = 50, crossover prob. = 0.8, mutation prob. = 0.01, examples threshold = 5
Oblique-DT	Number of total generations for the GA = 25, population size = $20\sqrt{d}$ (d = dimensionality)
TARGET	Split prob. = 0.5, number of gen. = 100, number of trees = 50, number of crossovers = 30, number of mutations = 10, number of clones = 5, number of transplants = 5

It works in an iterative fashion, each iteration searching for a complex that covers a large number of examples of a single class C and few examples of other classes.

4) C4.5 [3] is a decision tree generating algorithm. It induces classification rules in the form of decision trees from a set of given examples. The decision tree is constructed top-down using the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is the one used to make the decision.

5) C4.5-Rules [26] is based on the C4.5 algorithm. A C4.5 tree is generated and its rules extracted. A hill climbing algorithm is then performed in order to find the best subset of rules [according to the minimum description length (MDL) heuristic].

6) Ripper [27] builds a decision list of rules that correctly predicts the value of the target attribute. The list of rules is grown one by one and immediately pruned. Once a decision list for a given class is completely learned, an optimization stage is performed.

The choice of these non-GBML algorithms was made on the basis of the knowledge representation. Since we study GBML algorithms that evolve a set of rules, we aimed at comparing their results with other learners obtaining sets of rules, as well. It is well known that the quality of the model obtained by an algorithm depends both on the knowledge representation (language bias) and on the search mechanism (inductive bias). Our aim was to minimize the differences due to the language bias, so that the comparison focused mainly on the algorithm itself (inductive bias). This is the main reason for restricting non-GBML approaches to those that evolve sets of rules.

D. Data Sets

In the first part of the paper, which deals with standard classification, we selected 30 data sets from the UCI repository [30]. Table VIII summarizes the properties of the selected data sets. It shows, for each data set, the number of examples (#Ex.), the number of attributes (#Atts.), the number of numerical (#Num.) and nominal (#Nom.) attributes, and the number of classes (#Cl.). Some of the largest data sets (*abalone*, *magic*, *nursery*, *penbased*, and *ring*) were stratified sampled at 10% in order to reduce the computational time required for training. In the case of missing values (*cleveland*, *breast cancer*, *dermatology*, *hepatitis*, and *wisconsin*) we removed those instances from the data set.

In the second part of the paper, which deals with imbalanced data, we considered 33 data sets from UCI with different ratios between the minority and majority classes: from low imbalance to highly imbalanced data sets. Multiclass data sets were modified to obtain two-class non-balanced problems, so that the union of one or more classes became the positive class and the union of one or more of the remaining classes was labeled as the negative class.

Table IX summarizes the imbalanced data employed in this paper and shows, for each data set, the number of examples (#Ex.), number of attributes (#Atts.), class name of each class (minority and majority), and class attribute distribution. This table is ordered according to this last column, from low to high imbalance.

TABLE XI
PARAMETER SPECIFICATION FOR THE STATE-OF-THE-ART NON-EVOLUTIONARY ALGORITHMS EMPLOYED IN THE EXPERIMENTATION

Algorithm	Parameters
CART	Max depth of the tree = 90
AQ	Star size = 5
CN2	Star size = 5
C4.5 and C4.5-Rules	Prune = true, confidence level = 0.25, minimum number of item-sets per leaf = 2
Ripper	Size of growing subset = 66% Repetitions of the optimization stage = 2

E. Parameters

The configuration parameters for the GBML algorithms are shown in Table X, whereas in Table XI we show the parameters for the classical non-evolutionary algorithms. In both cases, the selected values are common for all problems, and they were selected according to the recommendation of the corresponding authors of each algorithm, which is the default parameters' setting included in the KEEL software [53].

Although we acknowledge that the tuning of the parameters for each method on each particular problem could lead to better results, we preferred to maintain a baseline performance of each method as the basis for comparison. Our hypothesis is that methods that win on average on all problems would also win if a better setting was performed. Furthermore, in a framework where no method is tuned, winner methods tend to correspond to the most robust learners, which is also a desirable characteristic.

Finally, a preprocessing discretization step was applied to algorithms that do not cope with numerical vales. Specifically, we used the class-attribute dependent discretizer [89].

F. Statistical Tests for Performance Comparison

In this paper, we use the hypothesis testing techniques to provide statistical support for the analysis of the results [90], [91]. Specifically, we use non-parametric tests, due to the fact that the initial conditions that guarantee the reliability of the parametric tests may not be satisfied, causing the statistical analysis to lose credibility with these parametric tests [32].

Specifically, we use the Wilcoxon signed-rank test [92] as a non-parametric statistical procedure for performing pairwise comparisons between two algorithms. For multiple comparisons, we use the Iman–Davenport test [91] to detect statistical differences among a group of results and the Shaffer post-hoc test [93] in order to find out which algorithms are distinctive among an $n \times n$ comparison.

The post-hoc procedure allows us to know whether 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 know whether

two algorithms are significantly different and how different they are.

Furthermore, we consider the average ranking of the algorithms in order to show graphically how good a method is with respect to its partners. This ranking is obtained by assigning a position to each algorithm depending on its performance for each data set. The algorithm which achieves the best accuracy in a specific data set will have the first ranking (value 1); then, the algorithm with the second best accuracy is assigned rank 2, and so forth. This task is carried out for all data sets and finally an average ranking is computed as the mean value of all rankings.

These tests are suggested in the studies presented in [32], [33], [90], where its use in the field of machine learning is highly recommended.

G. KEEL Software

All the methods selected for this paper are included in the KEEL software [53], which is a non-commercial Java software tool that empowers the user to assess the behavior of evolutionary learning for different kinds of data mining problems: regression, classification, clustering, pattern mining and so on. This tool can offer several advantages.

1) It reduces programming work. It includes a library with evolutionary learning algorithms based on different paradigms (those presented in this paper) and simplifies the integration of evolutionary learning algorithms with different pre-processing techniques. It can alleviate researchers from the mere "technical work" of programming and enable them to focus more on the analysis of their new learning models in comparison with the existing ones.
2) It extends the range of possible users applying evolutionary learning algorithms. An extensive library of EAs, together with easy-to-use software, considerably reduces the level of knowledge and experience required by researchers in evolutionary computation. As a result, researchers from other fields would be able to apply successfully these algorithms to their problems.
3) Due to the development of the library of algorithms and the application interface on Java, KEEL can be run on any machine with a Java interpreter.

Researchers can develop comparative experiments with a large collection of state-of-the-art methods for different areas of data-mining, such as decision trees, fuzzy rule-based systems, or rule-based learning, but they can also implement their own algorithms within the KEEL software [94] in order to contrast the results of these methods with the ones already included, or to make them easily accessible to other authors.

H. Web Page Associated With the Paper

In order to provide additional material to the paper content, we have developed a Web page[2] in which we have included the following information.

1) The proposed taxonomy for GBML algorithms for rule induction.
2) The data set partitions employed in the paper, both for standard classification and for classification with imbalanced data sets. In the latter case, we provide both the original data sets and the training sets preprocessed with SMOTE.
3) The complete tables of results for standard classification and for classification with imbalanced data sets, both with the original data sets and with SMOTE preprocessing. We include some Excel files with the train and test results for all the algorithms so that any interested researcher can use them to include their own results and extend the present comparison.

[2]http://sci2s.ugr.es/gbml/index.php

IV. Analysis of the GBML Algorithms for Rule Induction in Standard Classification

In this section, we present the experimental study focused on standard classification problems. To tackle all the results obtained by the different methods we designed an hierarchical analysis that highlighted the best methods in each family. The methodology is as follows.

1) First, we perform a comparison of the methods within each family. This leads to the selection of the best methods in each family, which will be identified as the family representatives and will be used to compare the performance among the families.
2) The family representatives are also used in the comparison with the state-of-the-art non-evolutionary rule learners.

Estimates of accuracy rate and kappa metric were obtained by means of a five-iterated five-fold cross-validation. That is, the data set was split into five folds, each one containing 20% of the patterns of the data set. For each fold, the algorithm was trained with the examples contained in the remaining folds and then tested with the current fold. The process was repeated five times using a different splitting. Therefore each result is the average over 25 runs.

This experimental study is carried out in the following two sections.

A. Study of the Behavior of the Algorithms in the Different Families

Our first objective is to study the behavior of the GBML approaches for rule induction, separately for each of the families presented in our taxonomy.

Table XII shows the average results in training and test using accuracy and Cohen's kappa as performance measures ($\pm$standard deviation). The average rank (and the rank position) are also included, measured for both metrics in the test scenario. This table is divided into five parts, each one belonging to a different family, i.e., Michigan, IRL, GCCL, Pittsburgh and HEDT. The best global result within each family is stressed in **bold-face**.

These results are also represented as a box plot (only for test) in Figs. 1 and 2. Box plots proved a most valuable tool in data reporting, since they allowed the graphical representation of the performance of the algorithms, indicating important

TABLE XII
AVERAGE ACCURACY AND KAPPA RESULTS FOR STANDARD CLASSIFICATION

		Accuracy			Kappa		
Family	Algorithm	Training	Test	Avg. Rank	Training	Test	Avg. Rank
Michigan	XCS	88.10 ± 2.64	**77.81 ± 4.12**	3.82 (1)	0.7900 ± 0.0569	**0.5866 ± 0.0874**	4.35 (2)
	UCS	93.44 ± 2.94	76.68 ± 5.58	6.08 (3)	0.8748 ± 0.0589	0.5688 ± 0.1125	5.98 (3)
IRL	SIA	97.57 ± 0.23	**74.65 ± 3.58**	7.05 (5)	0.9619 ± 0.0044	**0.5237 ± 0.0696**	6.75 (6)
	HIDER	85.05 ± 1.14	72.28 ± 3.64	7.30 (6)	0.6833 ± 0.0268	0.4653 ± 0.0749	7.73 (7)
GCCL	CORE	69.52 ± 3.29	67.26 ± 4.64	10.77 (13)	0.4503 ± 0.0719	0.3989 ± 0.0959	11.93 (14)
	OCEC	81.15 ± 3.28	**70.42 ± 4.67**	9.85 (11)	0.6656 ± 0.0524	**0.4840 ± 0.0810**	8.57 (9)
	COGIN	86.79 ± 1.59	67.95 ± 4.35	10.37 (12)	0.7261 ± 0.0315	0.4393 ± 0.0689	10.23 (13)
Pittsburgh	GIL	77.41 ± 3.73	67.63 ± 5.19	11.33 (14)	0.6262 ± 0.0522	0.4559 ± 0.0859	9.55 (10)
	Pitts-GIRLA	74.80 ± 2.81	62.86 ± 10.10	11.42 (15)	0.5704 ± 0.0499	0.3255 ± 0.1558	12.07 (15)
	DMEL	48.01 ± 5.84	46.47 ± 6.06	14.50 (16)	0.2305 ± 0.0657	0.1697 ± 0.0693	14.27 16)
	GASSIST	85.68 ± 1.28	**77.78 ± 3.71**	3.95 (2)	0.7471 ± 0.0258	**0.5953 ± 0.0731**	3.67 (1)
	OIGA	78.27 ± 2.41	72.66 ± 4.14	7.30 (7)	0.5878 ± 0.0535	0.4959 ± 0.0822	7.93 (8)
	ILGA	75.60 ± 2.74	70.58 ± 4.28	9.05 (9)	0.5305 ± 0.0562	0.4505 ± 0.0813	10.13 (12)
HEDT	DT-GA	82.57 ± 1.86	76.28 ± 3.39	6.13 (4)	0.6790 ± 0.0371	0.5495 ± 0.0626	6.53 (4)
	Oblique-DT	99.27 ± 0.04	**76.58 ± 3.34**	7.50 (8)	0.9874 ± 0.0011	**0.5779 ± 0.0647**	6.60 (5)
	TARGET	70.14 ± 2.42	68.78 ± 4.02	9.58 (10)	0.4750 ± 0.0516	0.4336 ± 0.0808	9.70 (11)

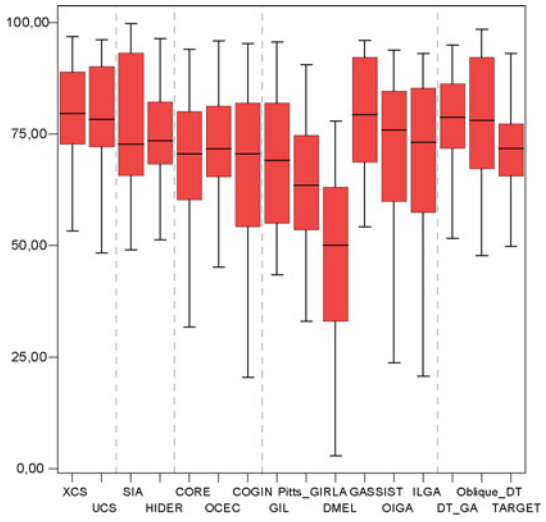


Fig. 1. Box plot of the accuracy results for all GBML algorithms.

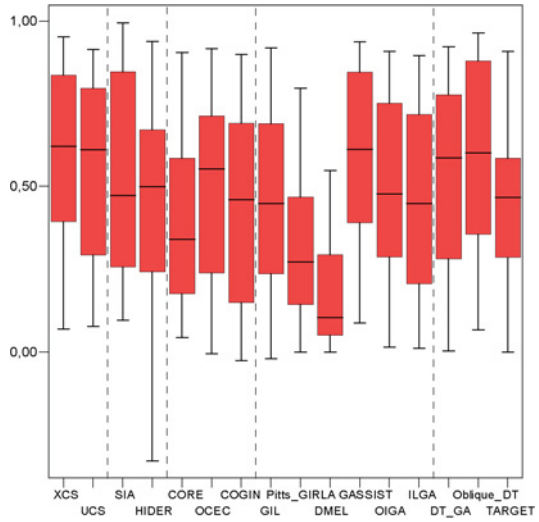


Fig. 2. Box plot of the kappa results for all GBML algorithms.

TABLE XIII
WILCOXON TEST TO COMPARE THE MICHIGAN APPROACHES

Measure	Comparison	R^+	R^-	Hypothesis ($\alpha = 0.05$)	p-value
Accuracy	XCS versus UCS	345.0	120.0	Rejected for XCS	0.021
Kappa	XCS versus UCS	325.0	140.0	Not rejected	0.057

R^+ corresponds to the sum of the ranks for XCS and R^- for UCS.

features such as the median, extreme values, and spread of values about the median in the form of quartiles.

The statistical comparison for the Michigan algorithms is shown in Table XIII, where we can observe that the null hypothesis of equivalence is: 1) rejected in favor of the XCS algorithm with level of significance $\alpha = 0.05$ using accuracy rate, and 2) not rejected using kappa, although a low p-value is obtained (p-value = 0.057).

In the case of the IRL methods, the statistical study is shown in Table XVI. No statistical differences were found in this case, although SIA obtains a better rank than HIDER.

The results of the statistical analysis of the GCCL family are as follows. Under accuracy rate, the null hypothesis that all the methods are equivalent cannot be rejected, since the p-value returned by the Iman–Davenport test (0.41341) is higher than our α-value (0.05). Nevertheless, under the kappa measure, Iman–Davenport's p-value is 0.00304 so the null hypothesis can be rejected. In this latter case, we proceed with a Shaffer test (see Table XVII) where we conclude that OCEC is statistically better than CORE and COGIN. In this table, a "+" symbol implies that the algorithm in the row is statistically better than the one in the column, whereas "−" implies the contrary; "=" means that the two algorithms compared have no significant differences. In brackets, the adjusted p-value associated to each comparison is shown.

Regarding the Pittsburgh methods, the Iman–Davenport test detects significant differences among the algorithms using both accuracy and kappa as performance measures, being the associated p-value near to zero in both cases. The statistical study is shown in Tables XIV and XV, where a Shaffer post-hoc test is shown for accuracy and kappa, respectively. Observing

TABLE XIV
SHAFFER TEST FOR PITTSBURGH ALGORITHMS USING ACCURACY AS PERFORMANCE MEASURE

	GIL	Pitts-GIRLA	DMEL	GASSIST	OIGA	ILGA
GIL	x	= (0.89023)	+ (0.04043)	− (6.76927E-8)	− (0.04271)	= (0.44991)
Pitts-GIRLA	= (0.89023)	x	+ (0.04271)	− (2.946285E-8)	− (0.03266)	= (0.44991)
DMEL	− (0.04043)	− (0.04271)	x	− (1.74046E-16)	− (4.99417E-7)	− (2.56050E-4)
GASSIST	+ (6.76927E-8)	+ (2.946285E-8)	+ (1.74046E-16)	x	+ (0.01331)	+ (1.37753E-4)
OIGA	+ (0.04271)	+ (0.03266)	+ (4.99417E-7)	− (0.01331)	x	= (0.44991)
ILGA	= (0.44991)	= (0.44991)	+ (2.56050E-4)	− (1.37753E-4)	= (0.44991)	x

TABLE XV
SHAFFER TEST FOR PITTSBURGH ALGORITHMS USING KAPPA AS PERFORMANCE MEASURE

	GIL	Pitts-GIRLA	DMEL	GASSIST	OIGA	ILGA
GIL	x	= (0.33799)	+ (0.00194)	− (3.77429E-5)	= (0.72225)	= (0.72225)
Pitts-GIRLA	= (0.33799)	x	= (0.27223)	− (2.17279E-9)	− (0.02626)	= (0.72225)
DMEL	− (0.00194)	= (0.27223)	x	− (1.02574E-15)	− (9.60977E-6)	− (0.02103)
GASSIST	+ (3.77429E-5)	+ (2.17279E-9)	+ (1.02574E-15)	x	+ (0.00392)	+ (7.34537E-7)
OIGA	= (0.72225)	+ (0.02626)	+ (9.60977E-6)	− (0.00392)	x	= (0.27223)
ILGA	= (0.72225)	= (0.72225)	+ (0.02103)	− (7.34537E-7)	= (0.27223)	x

TABLE XVI
WILCOXON TEST TO COMPARE THE IRL APPROACHES

Measure	Comparison	R^+	R^-	Hypothesis ($\alpha = 0.05$)	p-value
Accuracy	SIA versus HIDER	246.0	219.0	Not rejected	0.781
Kappa	SIA versus HIDER	258.0	207.0	Not rejected	0.600

R^+ corresponds to the sum of the ranks for SIA and R^- for HIDER.

TABLE XVII
SHAFFER TEST FOR GCCL ALGORITHMS USING KAPPA AS PERFORMANCE MEASURE

	CORE	OCEC	COGIN
CORE	x	− (0.00375)	= (0.30170)
OCEC	+ (0.00375)	x	+ (0.02819)
COGIN	= (0.30170)	− (0.02819)	x

the results from these tables, we conclude that GASSIST has a better behavior than the rest of the Pittsburgh approaches both in accuracy and kappa. DMEL is always outperformed by the remaining algorithms and OIGA is statistically better than the GIL and Pitts-GIRLA methods in accuracy and only better than Pitts-GIRLA in kappa.

In the HEDT family, the Iman–Davenport test (0.05834 in accuracy and 0.06965 in kappa) gets a higher p-value than our α-value 0.05, so there are no statistical differences.

Finally, we present a list of algorithms that have been selected as representative of each family. The criterion for choosing each method is based on the best performing method according to the differences supported by the statistical analysis. If no statistical differences are found, then the best mean result is taken, as in the case of SIA and Oblique-DT.

1) Chromosome = Rule:
   a) Michigan approach: XCS;
   b) IRL approach: SIA;
   c) GCCL approach: OCEC.
2) Chromosome = Rule Set (Pittsburgh): GASSIST.
3) HEDTs: Oblique-DT.

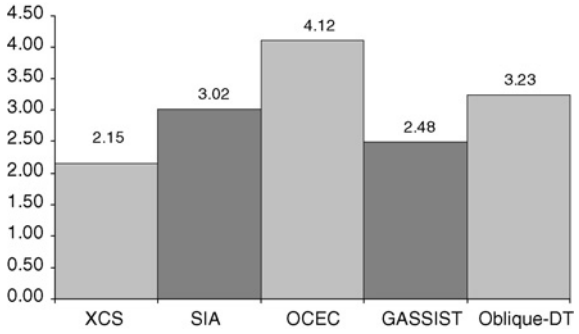


Fig. 3. Ranking in accuracy for the selected GBML algorithms.

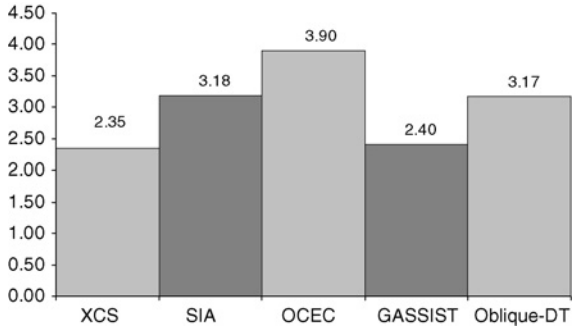


Fig. 4. Ranking in kappa for the selected GBML algorithms.

Figs. 3 and 4 show the average ranking computed for these approaches in accuracy and kappa, respectively. This task is carried out for all data sets and finally an average ranking is computed as the mean value of all rankings. We can observe that XCS and GASSIST are the best algorithms in both cases, whereas OCEC obtains the worst ranking. SIA and Oblique-DT have similar behavior.

To sum up, we included the star plot representations in accuracy and kappa for the best algorithms in each family, that

TABLE XVIII
AVERAGE ACCURACY AND KAPPA RESULTS FOR STANDARD CLASSIFICATION IN GBML ALGORITHMS FOR RULE INDUCTION AND STATE-OF-THE-ART NON-EVOLUTIONARY ALGORITHMS

GBML Algorithms for Rule Induction						
	Accuracy			Kappa		
Algorithm	Training	Test	Avg. Rank	Training	Test	Avg. Rank
XCS	88.10 ± 2.64	**77.81 ± 4.12**	3.42 (1)	0.7900 ± 0.0569	0.5866 ± 0.0874	3.75 (1)
SIA	97.57 ± 0.23	74.65 ± 3.58	5.75 (5)	0.9619 ± 0.0044	0.5237 ± 0.0696	5.92 (7)
OCEC	81.15 ± 3.28	70.42 ± 4.67	7.92 (10)	0.6656 ± 0.0524	0.4840 ± 0.0810	7.37 (9)
GASSIST	85.68 ± 1.28	77.78 ± 3.71	3.68 (2)	0.7471 ± 0.0258	**0.5953 ± 0.0731**	3.60 (2)
Oblique-DT	99.27 ± 0.04	76.58 ± 3.34	6.07 (6)	0.9874 ± 0.0011	0.5779 ± 0.0647	5.77 (6)
Non-Evolutionary Algorithms for Rule Induction						
	Accuracy			Kappa		
Algorithm	Training	Test	Avg. Rank	Training	Test	Avg. Rank
CART	83.69 ± 3.71	73.91 ± 3.91	6.20 (7)	0.7013 ± 0.0664	0.5167 ± 0.0744	6.57 (8)
AQ	81.89 ± 3.36	67.77 ± 5.18	8.83 (11)	0.6803 ± 0.0587	0.4240 ± 0.0971	9.23 (11)
CN2	80.98 ± 1.46	72.80 ± 3.51	7.40 (9)	0.6053 ± 0.0314	0.4484 ± 0.0746	8.53 (10)
C4.5	89.93 ± 1.47	**77.77 ± 3.49**	4.35 (3)	0.8020 ± 0.0362	**0.5821 ± 0.0647**	4.40 (3)
C4.5-Rules	83.84 ± 2.05	76.59 ± 4.10	5.55 (4)	0.7075 ± 0.0440	0.5717 ± 0.0779	5.23 (4)
Ripper	88.70 ± 2.42	73.96 ± 4.24	6.83 (8)	0.8147 ± 0.0387	0.5643 ± 0.0712	5.63 (5)

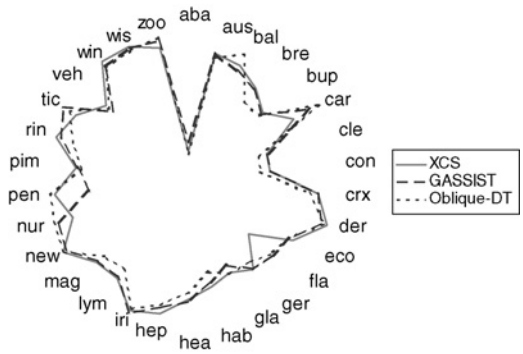


Fig. 5. Star plot representation in accuracy for the selected GBML algorithms.

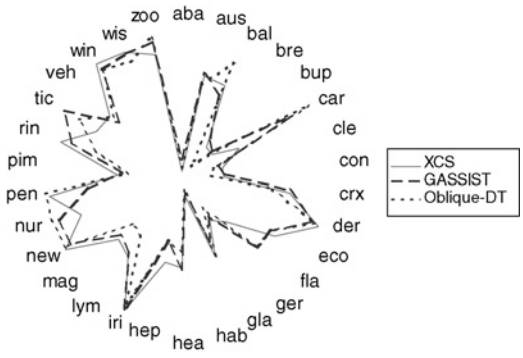


Fig. 6. Star plot representation in kappa for the selected GBML algorithms.

is, XCS for "Chromosome = Rule," GASSIST for "Pittsburgh" and Oblique-DT for "HEDTs." These star plots represent the performance as the distance from the center; hence a higher area determines the best average performance. We did not include all five algorithms in the representation for the sake of simplicity. The plots allow us to visualize the performance of the algorithms comparatively for each problem and in general. In Fig. 5, we can observe that under the accuracy rate metric, XCS achieves very good results except in those problems that only have nominal attributes, such as car, flare, nursery, and tic-tac-toe. GASSIST and Oblique-DT obtain a similar plot, but GASSIST has a higher area, which implies a better general behavior. Regarding kappa (Fig. 6), the conclusions extracted from the star plot are equivalent to those obtained with the accuracy rate.

B. Study of the Representative Algorithms of Each Family Versus Classical Ones

In the first part of this paper we selected the representative algorithms for each family defined in our taxonomy. The goal of this paper is also to perform a comparative study among the main GBML approaches with some state-of-the-art non-evolutionary rule classification methods listed in Section III-C. These algorithms are CART [23], AQ [24], CN2 [25], C4.5 [3], C4.5-Rules [26], and Ripper [27]. As we state previously, these methods were run using the KEEL software [53], following the recommended parameter settings given in KEEL, which correspond with the settings reported in the papers in which these methods have been proposed.

In Table XVIII, we show the results for accuracy and Cohen's kappa for all the selected GBML approaches for rule induction, that is, XCS, SIA, OCEC, GASSIST and Oblique-DT and for the classical non-evolutionary methods. The results for the test partitions are also depicted in Figs. 7 and 8 using box plots as representation scheme. Again, for brevity, we do not include the results for every single data set but the average performance of each algorithm.

According to the average results presented in Table XVIII, we may conclude that most of the selected GBML approaches for rule induction perform better than the non-evolutionary algorithms. Only in the case of C4.5 and C4.5-Rules do we obtain a similar behavior among these classical methods and the evolutionary algorithms.

We first check for significant differences using an Iman–Davenport test, which obtains a p-value below our level of significance, and near to zero in both cases, accuracy and kappa. The associated statistical study is developed in Tables XIX and XX, where we show the p-values computed by a

TABLE XIX
SHAFFER TEST FOR GBML ALGORITHMS FOR RULE INDUCTION AGAINST NON-EVOLUTIONARY APPROACHES USING ACCURACY AS PERFORMANCE MEASURE

	CART	AQ	CN2	C4.5	C4.5-Rules	Ripper
XCS	+ (0.04266)	+ (1.39004E-8)	+ (1.48266E-4)	= (1.0)	= (0.39467)	+ (0.00298)
SIA	= (1.0)	+ (0.01175)	= (1.0)	= (1.0)	= (1.0)	= (1.0)
OCEC	= (1.0)	= (1.0)	= (1.0)	− (0.00140)	= (0.19381)	= (1.0)
GASSIST	= (0.12189)	+ (8.14983E-8)	+ (6.40788E-4)	= (1.0)	= (0.84890)	+ (0.01056)
Oblique-DT	= (1.0)	+ (0.04568)	= (1.0)	= (1.0)	= (1.0)	= (1.0)

TABLE XX
SHAFFER TEST FOR GBML ALGORITHMS FOR RULE INDUCTION AGAINST NON-EVOLUTIONARY APPROACHES USING KAPPA AS PERFORMANCE MEASURE

	CART	AQ	CN2	C4.5	C4.5-Rules	Ripper
XCS	+ (0.03718)	+ (6.84899E-9)	+ (1.04724E-6)	= (1.0)	= (1.0)	= (0.75221)
SIA	= (1.0)	+ (0.00398)	= (0.08086)	= (1.0)	= (1.0)	= (1.0)
OCEC	= (1.0)	= (0.75221)	= (1.0)	− (0.01967)	= (0.39467)	= (1.0)
GASSIST	+ (0.01967)	+ (2.61693E-9)	+ (3.76517E-7)	= (1.0)	= (1.0)	= (0.50972)
Oblique-DT	= (1.0)	+ (0.00232)	+ (0.04568)	= (1.0)	= (1.0)	= (1.0)

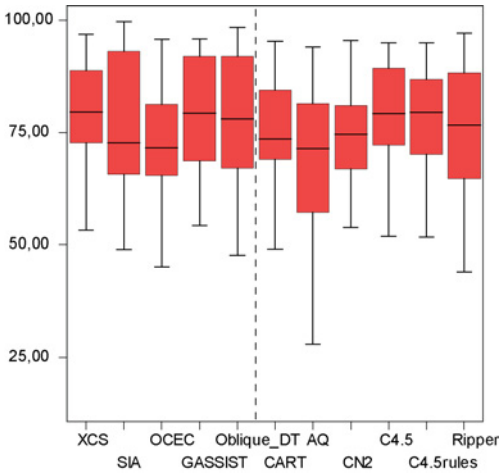


Fig. 7. Box plot representation of the accuracy results for the selected GBML and non-evolutionary algorithms.

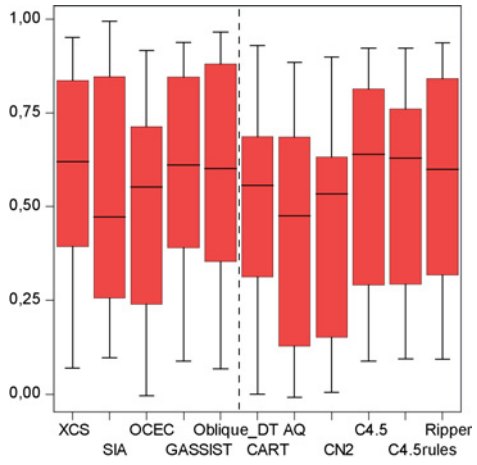


Fig. 8. Box plot representation of the kappa results for the selected GBML and non-evolutionary algorithms.

Shaffer test with which we compare every GBML algorithm for rule induction against all the classical approaches using accuracy and kappa measures, respectively. This test was explained in Section IV-A.

In these tables, we may observe that XCS and GASSIST are the algorithms with the best behavior in contrast to the non-evolutionary algorithms. The remaining GBML methods are statistically similar to most of the state-of-the-art non-evolutionary approaches. AQ is in general worse than the other approaches, and C4.5 only outperforms OCEC, but none of the other GBML approaches. There are few differences between the results in both tables, although we observe that the p-values are, in general, lower in the case of the kappa measure. Even more clearly, we learn that GASSIST is better than CART regarding kappa but not with accuracy. The same happens for Oblique-DT against CN2. Nevertheless, we observe the contrary case for XCS and GASSIST against Ripper, where the former algorithms achieve a better behavior only in the case of the accuracy metric.

Considering the performance results of Table XVIII and the statistical study in Tables XIX and XX, we select C4.5 and C4.5-Rules as representative algorithms for the non-evolutionary approaches for rule induction. These methods will be employed in the next section of the experimental study.

As we did in the previous part of this paper, we depict in Figs. 9 and 10 the rank computed for the selected approaches in accuracy and kappa. In this case, C4.5 is one of the three best algorithms behind XCS and GASSIST. The remaining methods have similar behavior, except for OCEC which is the last ranked algorithm.

Finally, we take the three best ranked algorithms, XCS, GASSIST, and C4.5, and we show the star plot representations in Figs. 11 and 12 for accuracy and kappa, respectively. In accuracy, XCS has a better performance in all problems except for those that have only nominal values (car, flare, nursery, tic-tac-toe, and zoo). GASSIST and C4.5 are very similar in most data sets, also alternating the best results for some data sets,

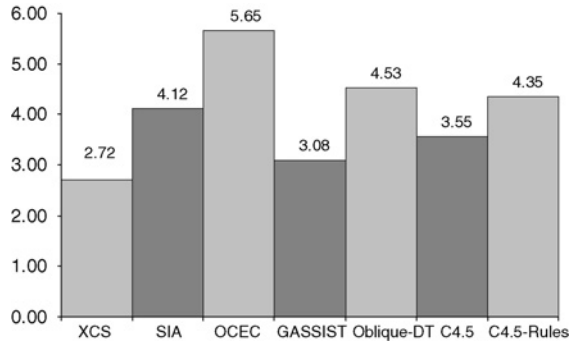


Fig. 9. Ranking in accuracy for the selected GBML and non-evolutionary algorithms.

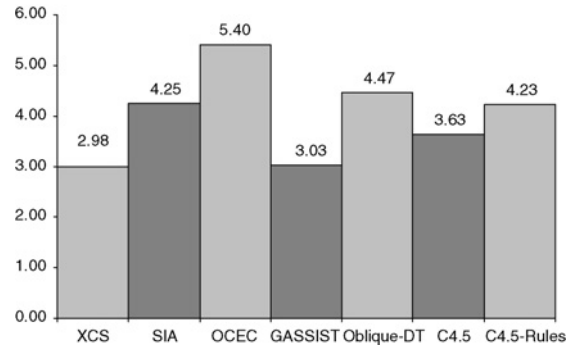


Fig. 10. Ranking in kappa for the selected GBML and non-evolutionary algorithms.

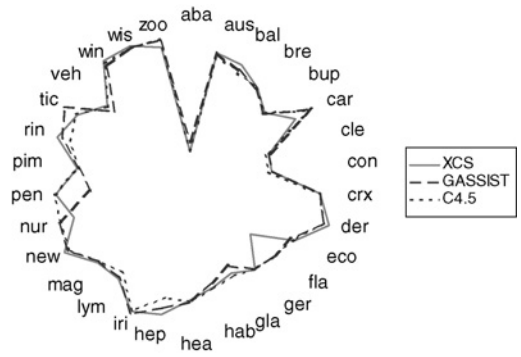


Fig. 11. Star plot representation in accuracy for the selected GBML and non-evolutionary algorithms.

i.e., GASSIST outperforms C4.5 in tic-tac-toe and we observe the contrary in penbased. For the kappa metric we have more differences among the algorithms although XCS is still the one that performs best in most of the cases under study.

V. Analysis of the GBML Algorithms for Rule Induction in Imbalanced Data Sets

In this section, we analyze the effect of imbalance on the classification quality of all the algorithms selected in our study, following the same methodology than in the previous section. We decide not to use just the representatives of the different families of GBML algorithms for rule induction because we cannot assume that the algorithms that perform best on standard classification will also be the ones that obtain the best behavior on imbalanced data sets.

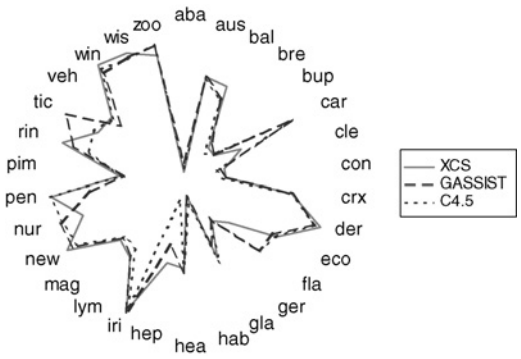


Fig. 12. Star plot representation in kappa for the selected GBML and non-evolutionary algorithms.

TABLE XXI
Average Results for Imbalanced Data Sets Classification With GBML Algorithms for Rule Induction

Family	Algorithm	GM_{Tr}	GM_{Tst}	Avg. Rank
Michigan	XCS	69.37 ± 10.30	58.93 ± 9.72	6.93 (8)
	UCS	76.46 ± 13.12	**64.92 ± 16.55**	6.77 (5)
IRL	SIA	99.55 ± 0.38	**69.62 ± 9.49**	6.03 (4)
	HIDER	64.67 ± 7.94	56.13 ± 11.05	9.29 (9)
GCCL	CORE	57.75 ± 10.26	52.73 ± 14.74	9.35 (10)
	OCEC	80.00 ± 3.74	**70.88 ± 10.38**	6.86 (6)
	COGIN	60.18 ± 10.65	48.08 ± 16.13	11.62 (15)
Pittsburgh	GIL	79.87 ± 3.60	**70.96 ± 9.87**	6.86 (7)
	Pitts-GIRLA	55.11 ± 10.61	39.59 ± 17.52	11.17 (14)
	DMEL	22.99 ± 10.26	20.76 ± 12.07	14.15 (16)
	GASSIST	78.65 ± 5.02	67.58 ± 9.69	5.79 (3)
	OIGA	69.41 ± 5.65	57.48 ± 10.65	9.71 (11)
	ILGA	64.48 ± 8.95	51.76 ± 11.12	10.86 (13)
HEDT	DT-GA	78.60 ± 7.81	69.87 ± 11.60	5.68 (2)
	Oblique-DT	99.96 ± 0.01	**75.81 ± 8.55**	4.58 (1)
	TARGET	48.14 ± 14.14	46.93 ± 14.68	10.33 (12)

This paper is divided into two parts. First, we will present the results directly using the original data sets in order to determine which algorithms are best adapted to the framework of imbalanced data sets. Then, we will apply preprocessing to the training data sets to rebalance the distribution of examples of the different classes. Our aim is to analyze whether class rebalancing leads to performance improvement of the algorithms used in this paper, and of GBML methods in particular.

A. Empirical Analysis With the Original Data Sets

In the remainder of this section, we will present the results of all the GBML methods in the original imbalanced data sets. We will analyze how the different methods respond to the class imbalances and will select the best representative within each family. Finally, we will contrast the results of these selected evolutionary algorithms for rule induction with the classical methods, that is, CART, AQ, CN2, C4.5, C4.5-Rules, and Ripper.

1) *Study of the Behavior of the Algorithms in the Different Families With Imbalanced Data Sets:* Table XXI shows the average results for all data sets using the geometric mean

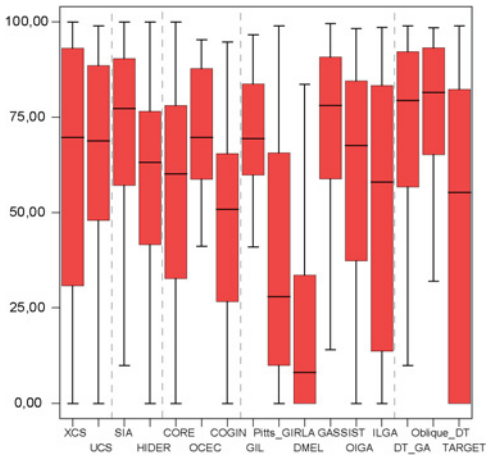


Fig. 13. Box plot representation of the geometric mean of all GBML algorithms in imbalanced data sets.

TABLE XXII
WILCOXON TEST TO COMPARE THE MICHIGAN APPROACHES IN IMBALANCED DATA SETS

Comparison	R^+	R^-	Hypothesis ($\alpha = 0.05$)	p-value
XCS versus UCS	231.5	329.5	Not rejected	0.360

R^+ corresponds to the sum of the ranks for XCS and R^- for UCS.

of the true rates for all GBML algorithms, also showing the average ranking. We stress in **boldface** the best value within each family.

For a visual comparison, we show a box plot representation for the results in test in Fig. 13. The results obtained in the framework of imbalanced data sets are interesting since in this case we observe larger differences in the average performance among the algorithms of the same families. Furthermore, both UCS and GIL achieve the highest value for the geometric mean both for Michigan and Pittsburgh approaches, in contrast with the case of standard classification, suggesting their good adaptation to imbalanced data sets.

The next step is to carry out a statistical analysis for each proposed family of GBML algorithms for rule induction, in the same way we did for standard classification, in order to select the representative algorithms for the comparison with the classical non-evolutionary approaches.

The first statistical comparison will be carried out for the Michigan algorithms, and it is shown in Table XXII. We can observe that the null hypothesis of equivalence is not rejected, but UCS obtains a higher sum of ranks in this case. This behavior is supported by the higher value of the average performance of the geometric mean (Table XXI) and in the lower spread observed in the box plot representation (Fig. 13) when contrasting UCS with XCS. This conclusion is in concordance with the results obtained by Orriols and Bernadó-Mansilla in [95].

In the case of the IRL methods, the statistical study is shown in Table XXIII. The Wilcoxon test concludes that SIA outperforms HIDER in the framework of imbalanced data sets, so we can state that the rules found by the former are less biased for the majority class examples.

TABLE XXIII
WILCOXON TEST TO COMPARE THE IRL APPROACHES IN IMBALANCED DATA SETS

Comparison	R^+	R^-	Hypothesis ($\alpha = 0.05$)	p-value
SIA versus HIDER	400.0	161.0	Rejected for SIA	0.033

R^+ corresponds to the sum of the ranks for SIA and R^- for HIDER.

TABLE XXIV
SHAFFER TEST FOR GCCL ALGORITHMS IN IMBALANCED DATA SETS

	CORE	OCEC	COGIN
CORE	x	− (0.00561)	+ (0.02672)
OCEC	+ (0.00561)	x	+ (1.85688E-6)
COGIN	− (0.02672)	− (1.85688E-6)	x

The results of the statistical analysis of the GCCL family are as follows. The p-value returned by the Iman–Davenport test is near to zero, so the null hypothesis can be rejected. In this case, we proceed with a Shaffer test (see Table XXIV) where we conclude that OCEC is statistically better than CORE and COGIN and that CORE is also better than COGIN.

Regarding the Pittsburgh methods, the Iman–Davenport test detects significant differences among the algorithms, being the associated p-value near to zero also in this case. The statistical study is shown in Table XXV, where a Shaffer post-hoc test is shown. Observing the results from this table, we conclude that GASSIST has a better behavior than the rest of the Pittsburgh except for GIL, which is also better than the remaining algorithms but OIGA. DMEL is outperformed by all methods but Pitts-GIRLA and ILGA, but in this latter case the p-value is lower than 0.1 (0.06546).

In the HEDT family, the Iman–Davenport test has also a p-value near to zero, and we proceed with a Shaffer test (Table XXVI) in which it is shown that TARGET is statistically worst than DT-GA and Oblique-DT, which have no differences between them. Nevertheless, we must highlight the goodness of the Oblique-DT algorithm since it obtains the best average result for the geometric mean and also a very low spread in the box plot representation, hence having a low sensitivity to the imbalance degree of the data sets and being a good method in this framework.

Finally, we present a list of algorithms that have been selected as representative of each family for imbalanced classification. The criterion for choosing each method is exactly the same as for standard classification; that is, we extract the best performing method according to the differences supported by the statistical analysis and if no statistical differences were found, then we select the best ranking method, as in the case of UCS, GASSIST, and Oblique-DT.

1) Chromosome = Rule:
   a) Michigan approach: UCS;
   b) IRL approach: SIA;
   c) GCCL approach: OCEC.
2) Chromosome = Rule Set (Pittsburgh): GASSIST.
3) HEDTs: Oblique-DT.

2) *Study of the Representative Algorithms of Each Family Versus Classical Ones With Imbalanced Data Sets:* Once we

TABLE XXV
SHAFFER TEST FOR PITTSBURGH ALGORITHMS IN IMBALANCED DATA SETS

	GIL	Pitts-GIRLA	DMEL	GASSIST	OIGA	ILGA
GIL	x	+ (0.00136)	+ (7.65851E-9)	= (0.44705)	= (0.14101)	+ (0.00267)
Pitts-GIRLA	− (0.00136)	x	= (0.11703)	− (4.82031E-6)	= (0.34857)	= (0.79241)
DMEL	− (7.65851E-9)	= (0.11703)	x	− (2.57755E-12)	− (5.20107E-4)	= (0.06546)
GASSIST	= (0.44705)	+ (4.82031E-6)	+ (2.57755E-12)	x	+ (0.00624)	+ (1.84089E-5)
OIGA	= (0.14101)	= (0.34857)	+ (5.20107E-4)	− (0.00624)	x	= (0.44328)
ILGA	− (0.00267)	= (0.79241)	= (0.06546)	− (1.84089E-5)	= (0.44328)	x

TABLE XXVI
SHAFFER TEST FOR HEDT ALGORITHMS IN IMBALANCED DATA SETS

	DT-GA	Oblique-DT	TARGET
DT-GA	x	= (0.62246)	+ (0.00209)
Oblique-DT	= (0.62246)	x	+ (0.00107)
TARGET	− (0.00209)	− (0.00107)	x

TABLE XXVII
AVERAGE RESULTS FOR IMBALANCED DATA SETS CLASSIFICATION WITH THE BEST GBML ALGORITHMS FOR RULE INDUCTION AND STATE-OF-THE-ART NON-EVOLUTIONARY ALGORITHMS

	GM_{Tr}	GM_{Tst}	Avg. Rank
UCS	76.46 ± 13.12	64.92 ± 16.55	6.15 (7)
SIA	99.55 ± 0.38	69.62 ± 9.49	5.91 (6)
OCEC	80.00 ± 3.74	70.88 ± 10.38	6.36 (9)
GASSIST	78.65 ± 5.02	67.58 ± 9.69	5.64 (5)
Oblique-DT	99.96 ± 0.01	*75.81 ± 8.55*	4.74 (3)
CART	87.39 ± 5.31	69.72 ± 11.51	6.26 (8)
AQ	67.00 ± 3.87	59.81 ± 8.72	8.73 (10)
CN2	56.53 ± 5.70	45.97 ± 12.09	9.82 (11)
C4.5	84.03 ± 6.01	73.28 ± 10.82	4.77 (4)
C4.5-Rules	84.81 ± 4.59	75.21 ± 10.23	4.61 (2)
Ripper	96.38 ± 0.86	**79.34 ± 8.98**	3.02 (1)

have selected the representative algorithms of each family we aim at comparing now their classification ability in imbalanced data sets versus the classical non-evolutionary algorithms. The average results using the geometric mean, together with the associated ranking for each method, are shown in Table XXVII, whereas a box plot representation that summarizes the performance over all data sets is depicted in Fig. 14.

From the table of results we can observe that the best performing algorithm is Ripper, followed by those based on decision trees, both for evolutionary and non-evolutionary methods. We also observe that the associated ranking of each algorithm does not necessarily correspond to the average performance value, since the metric we used gives a 0-value for those problems that do not have any minority class example well-classified resulting in a decrease of the mean value for some algorithms like GASSIST or UCS.

To carry out the statistical study, we only select Ripper, C4.5, and C4.5-Rules as representatives of the state-of-the-art non-evolutionary approaches, since they obtained the highest average performance and also the best ranks. This has been done for the sake of simplicity in the comparisons.

We first check for significant differences among the algorithms using an Iman–Davenport test. The p-value is lower than our level of confidence α and near to zero. Thus, we

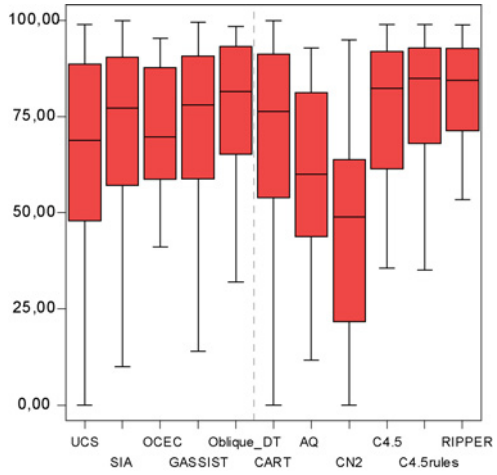


Fig. 14. Box plot representation of the geometric mean of the selected GBML algorithms and non-evolutionary algorithms in imbalanced data sets.

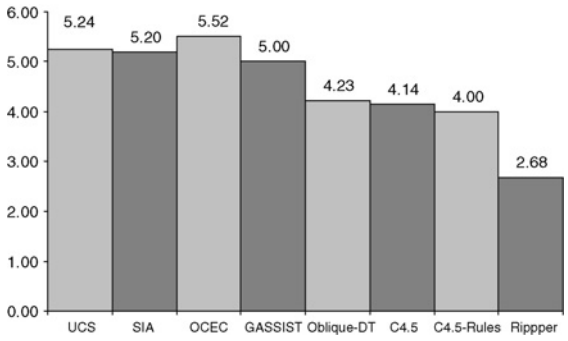


Fig. 15. Ranking computed for the selected GBML and non-evolutionary algorithms in imbalanced data sets.

can conclude that significant differences do exist, proceeding with a Shaffer test. The ranks of the algorithms are presented in Fig. 15, and the results of the multiple comparison test performed on all algorithms are shown in Table XXVIII.

Observing the statistical results, we may conclude that Oblique-DT is the GBML algorithm with the best behavior and that Ripper is globally the best-suited algorithm when classifying imbalanced data. We did not find any other difference among the remaining methods, which agrees with the similarities in the average performance and also with the rank-value of the algorithms.

Finally, Fig. 16 shows the star plot representation for the Ripper algorithm and the two best ranked GBML algorithms in imbalanced data sets, that is, Oblique-DT and GASSIST.

TABLE XXVIII
SHAFFER TEST FOR GBML ALGORITHMS FOR RULE INDUCTION AGAINST NON-EVOLUTIONARY APPROACHES IN IMBALANCED DATA SETS

	UCS	SIA	OCEC	GASSIST	Oblique-DT	C4.5	C4.5-Rules	Ripper
UCS	x	= (1.0)	= (1.0)	= (1.0)	= (1.0)	= (1.0)	= (0.62987)	− (4.56423E-4)
SIA	= (1.0)	x	= (1.0)	= (1.0)	= (1.0)	= (1.0)	= (0.75440)	− (6.37118E-4)
OCEC	= (1.0)	= (1.0)	x	= (1.0)	= (0.52325)	= (0.46677)	= (0.25168)	− (7.33636E-5)
GASSIST	= (1.0)	= (1.0)	= (1.0)	x	= (1.0)	= (1.0)	= (1.0)	− (0.00254)
Oblique-DT	= (1.0)	= (1.0)	= (0.52325)	= (1.0)	x	= (1.0)	= (1.0)	= (0.21802)
C4.5	= (1.0)	= (1.0)	= (0.46677)	= (1.0)	= (1.0)	x	= (1.0)	= (0.33309)
C4.5-Rules	= (0.62987)	= (0.75440)	= (0.25168)	= (1.0)	= (1.0)	= (1.0)	x	= (0.46677)
Ripper	+ (4.56423E-4)	+ (6.37118E-4)	+ (7.33636E-5)	+ (0.00254)	= (0.21802)	= (0.33309)	= (0.46677)	x

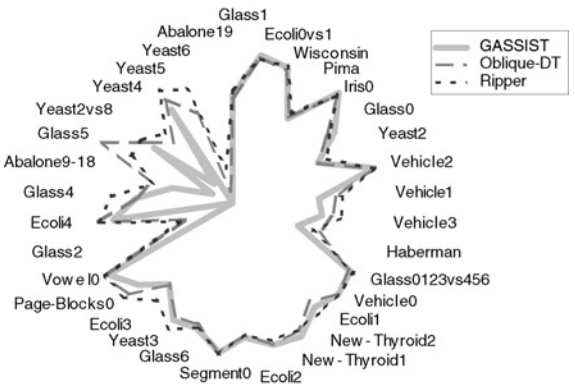


Fig. 16. Star plot representation for Ripper, Oblique-DT, and GASSIST in imbalanced data sets.

The data sets are ordered clockwise according to their degree of imbalance, starting at data set glass1 which has the lowest class imbalance. We can observe the irregular shape for the last ten data sets (from glass2 to abalone19), which suggests the difficulty for learning an accurate model when the minority class examples are highly under-represented. Clearly, Oblique-DT and Ripper maintain the most regular behavior.

B. Empirical Analysis Applying Preprocessing

Along the previous section we have studied the behavior of the GBML and non-evoluationary approaches by means of an empirical study using the original imbalanced data sets. Our aim is also to study whether a preprocessing method [87], [96] that rebalances the imbalance ratio of the data set helps to improve the performance and in which cases. We use the SMOTE method [88] as one of the best known preprocessing methods that compensate for class imbalances.

With this aim, we will proceed as usual, first detecting which algorithms have a better behavior within each family of GBML algorithms for rule induction and then we will compare this selection of methods with the classical non-evolutionary methods.

1) *Study of the Behavior of the Algorithms in the Different Families With Imbalanced Data Sets Applying Preprocessing:* Table XXIX shows the average results of all GBML methods for all data sets. The results shown are the geometric mean and the average ranking of each method. We stress in **boldface** the best value within each family.

When SMOTE preprocessing is used, the GBML algorithms highly improve their results with respect to the performance achieved when they are trained with the original imbalanced data sets. Specifically, we must highlight the good performance of XCS, GASSIST, and SIA. The box plot representation of the results shown in Fig. 17 supports this conclusion.

Furthermore, observing the box plot for the results with the original data sets (Fig. 13) and with preprocessing (Fig. 17) the "boxes" are smaller for the case of the application of SMOTE. According to this, when using this technique for

TABLE XXIX
AVERAGE RESULTS FOR IMBALANCED DATA SETS CLASSIFICATION WITH GBML ALGORITHMS FOR RULE INDUCTION WITH SMOTE PREPROCESSING

Family	Algorithm	GM_{Tr}	GM_{Tst}	Avg. Rank
Michigan	XCS	94.91 ± 1.51	**84.92 ± 5.69**	3.48 (1)
	UCS	78.89 ± 10.38	67.30 ± 12.53	9.29 (11)
IRL	SIA	89.47 ± 1.09	**81.79 ± 6.56**	5.23 (3)
	HIDER	89.11 ± 2.00	78.26 ± 9.05	7.55 (7)
GCCL	CORE	79.88 ± 4.05	**76.19 ± 8.77**	7.73 (9)
	OCEC	81.85 ± 2.02	71.70 ± 7.58	9.29 (12)
	COGIN	65.89 ± 7.41	60.79 ± 11.65	12.94 (14)
Pittsburgh	GIL	72.45 ± 3.42	66.80 ± 7.86	11.27 (13)
	Pitts-GIRLA	20.42 ± 9.82	21.15 ± 15.93	14.48 (15)
	DMEL	25.86 ± 14.84	23.89 ± 15.16	15.08 (16)
	GASSIST	93.47 ± 1.03	**83.69 ± 7.26**	4.03 (2)
	OIGA	89.40 ± 1.97	77.91 ± 9.99	7.55 (8)
	ILGA	88.29 ± 2.36	75.95 ± 8.70	8.64 (10)
HEDT	DT-GA	92.36 ± 1.97	**80.61 ± 7.66**	5.61 (4)
	Oblique-DT	99.96 ± 0.01	80.51 ± 7.12	6.42 (5)
	TARGET	82.80 ± 3.16	80.02 ± 6.86	7.42 (6)

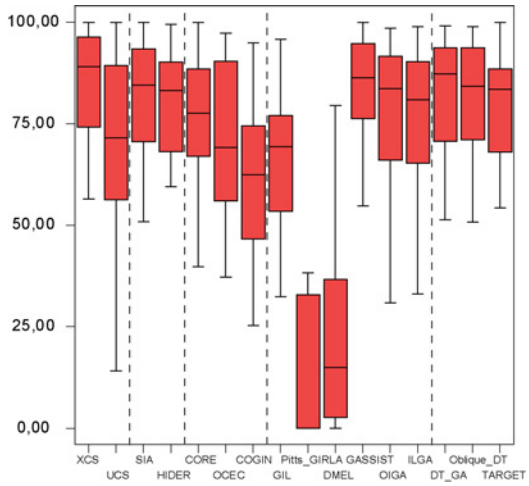


Fig. 17. Box plot for the results in imbalanced data sets with SMOTE preprocessing for all GBML algorithms.

TABLE XXX
SHAFFER TEST FOR PITTSBURGH ALGORITHMS IN IMBALANCED DATA SETS WITH SMOTE PREPROCESSING

	GIL	Pitts-GIRLA	DMEL	GASSIST	OIGA	ILGA
GIL	x	+ (0.00160)	+ (7.25460E-4)	− (2.87809E-5)	− (0.04115)	= (0.22697)
Pitts-GIRLA	− (0.00160)	x	= (0.85959)	− (1.13066E-15)	− (4.09033E-9)	- (4.73462E-7)
DMEL	− (7.25460E-4)	= (0.85959)	x	− (3.16485E-16)	− (1.13391E-9)	= (1.52804E-7)
GASSIST	+ (2.87809E-5)	+ (1.13066E-15)	+ (3.16485E-16)	x	= (0.16554)	+ (0.02800)
OIGA	+ (0.04115)	+ (4.09033E-9)	+ (1.13391E-9)	= (0.16554)	x	= (0.85959)
ILGA	= (0.22697)	+ (4.73462E-7)	+ (1.52804E-7)	− (0.02800)	= (0.85959)	x

TABLE XXXI
WILCOXON TEST TO COMPARE THE MICHIGAN APPROACHES IN IMBALANCED DATA SETS WITH SMOTE PREPROCESSING

Comparison	R^+	R^-	Hypothesis ($\alpha = 0.05$)	p-value
XCS versus UCS	544.5	16.5	Rejected for XCS	0.000

R^+ corresponds to the sum of the ranks for XCS and R^- for UCS.

TABLE XXXII
WILCOXON TEST TO COMPARE THE IRL APPROACHES IN IMBALANCED DATA SETS WITH SMOTE PREPROCESSING

Comparison	R^+	R^-	Hypothesis ($\alpha = 0.05$)	p-value
SIA versus HIDER	386.5	174.5	Not rejected	0.056

R^+ corresponds to the sum of the ranks for SIA and R^- for HIDER.

TABLE XXXIII
SHAFFER TEST FOR GCCL ALGORITHMS IN IMBALANCED DATA SETS WITH SMOTE PREPROCESSING

	CORE	OCEC	COGIN
CORE	x	= (0.53825)	+ (2.54822E-6)
OCEC	= (0.53825)	x	+ (1.64587E-5)
COGIN	− (2.54822E-6)	− (1.64587E-5)	x

dealing with imbalanced data, the quartiles computed from the results have a lower range and the standard deviation also decreases, resulting on a most robust behavior.

In order to select the most representative algorithm within each family, we will proceed as usual, performing a statistical analysis for each group of the proposed taxonomy. In this manner, we show in Table XXXI the comparison regarding the Michigan approaches, in which the Wilcoxon test shows the superiority of XCS versus UCS.

The Wilcoxon test shown in Table XXXII for the IRL methods cannot reject the null-hypothesis of equality, but the p-value obtained is very low (0.056) and thus we can conclude that there are significant differences between SIA and HIDER in favor of the former with a high degree of confidence.

Next, we carry out the statistical analysis for the GCCL family. First, we obtain the p-value computed by the Iman–Davenport test, and since it is near to zero, the test rejects the null hypothesis suggesting the presence of differences among the methods. We show a Shaffer test in Table XXXIII where we conclude that COGIN is outperformed by CORE and OCEC, which have no statistical differences between them.

Regarding the Pittsburgh methods, the Iman–Davenport test detects significant differences among the algorithms, being the associated p-value near to zero also in this case. The statistical study is shown in Table XXX, where a Shaffer post-hoc test is shown. The main conclusion extracted from this table is that GASSIST is the algorithm with the best behavior as it outperforms all methods but OIGA. Both Pitts-GIRLA and DMEL are the algorithms with worst behavior in this case.

In the HEDT family, the Iman–Davenport test does not detect statistical differences among the results, since the p-value (0.10857) is higher than our level of confidence (95%).

Finally, we present the list of algorithms that have been selected as representative of each family for imbalanced classification when applying SMOTE preprocessing. The criterion for choosing each method is the same as we defined in the previous section of the paper.

1) Chromosome = Rule:
   a) Michigan approach: XCS;
   b) IRL approach: SIA;
   c) GCCL approach: CORE.
2) Chromosome = Rule Set (Pittsburgh): GASSIST.
3) HEDTs: DT-GA.

TABLE XXXIV
AVERAGE RESULTS FOR IMBALANCED DATA SETS CLASSIFICATION IN GBML ALGORITHMS FOR RULE INDUCTION AND STATE-OF-THE-ART NON-EVOLUTIONARY ALGORITHMS. SMOTE PREPROCESSING

	GM_{Tr}	GM_{Tst}	Avg. Rank
XCS	94.91 ± 1.51	**84.92 ± 5.69**	3.41 (1)
SIA	89.47 ± 1.09	81.79 ± 6.56	4.55 (4)
CORE	79.88 ± 4.05	76.19 ± 8.77	6.74 (8)
GASSIST	93.47 ± 1.03	83.69 ± 7.26	3.80 (2)
DT-GA	92.36 ± 1.97	80.61 ± 7.66	5.33 (6)
CART	88.99 ± 5.64	70.55 ± 11.69	7.74 (9)
AQ	57.06 ± 5.96	52.52 ± 7.61	10.02 (11)
CN2	70.85 ± 4.41	63.61 ± 10.39	9.20 (10)
C4.5	95.85 ± 1.38	*82.43 ± 6.95*	4.23 (3)
C4.5-Rules	93.04 ± 1.72	81.79 ± 7.36	4.77 (5)
Ripper	95.45 ± 1.91	79.48 ± 10.56	6.21 (7)

2) *Study of the Representative Algorithms of Each Family Versus Classical Ones With Imbalanced Data Sets Applying Preprocessing:* Table XXXIV shows the average results with SMOTE preprocessing for the selected GBML algorithms and the classical non-evolutionary algorithms. The best methods in this case are XCS and GASSIST for the GBML algorithms, followed by C4.5 and C4.5-Rules from the non-evolutionary approaches. We can also observe the behavior of each algorithm in the box plot depicted in Fig. 18.

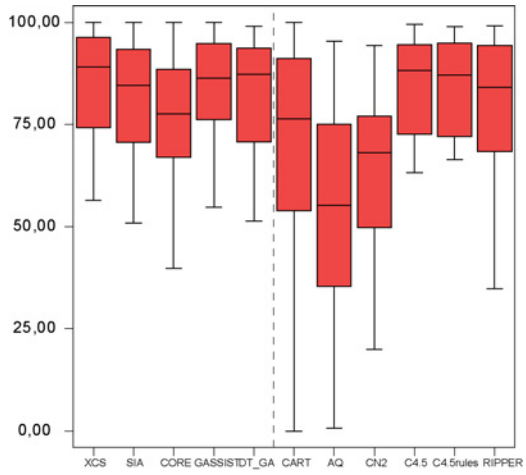


Fig. 18. Box plot representation of the results in imbalanced data sets with SMOTE preprocessing for the selected GBML and non-evolutionary algorithms.

TABLE XXXV
WILCOXON TEST TO COMPARE THE BEHAVIOR OF PREPROCESSING

Method	R^+	R^-	Hypothesis ($\alpha = 0.05$)	p-value
XCS	87.5	507.5	Rejected for SMOTE	0.000
SIA	66.0	529.0	Rejected for SMOTE	0.000
CORE	2.5	558.5	Rejected for SMOTE	0.000
GASSIST	24.0	571.0	Rejected for SMOTE	0.000
DT-GA	56.5	480.5	Rejected for SMOTE	0.000
CART	248.0	347.0	Not rejected	0.109
AQ	514.0	81.0	Rejected for original data set	0.000
CN2	77.0	518.0	Rejected for SMOTE	0.000
C4.5	37.5	557.5	Rejected for SMOTE	0.000
C4.5-Rules	52.5	542.5	Rejected for SMOTE	0.000
Ripper	320.5	274.5	Not rejected	0.675

R^+ corresponds to the sum of the ranks for the algorithms with the original data sets and R^- to the application of SMOTE.

The first step in this empirical study is to show the necessity of the application of preprocessing and the positive synergy of most of the methods with the well-known SMOTE technique. With this aim, we carry out a Wilcoxon test (Table XXXV) in which we observe that for all methods, except for CART, AQ and Ripper, the results when using SMOTE outperform the ones with no preprocessing. The increase in performance is stressed in the case of the GBML approaches, where all the selected approaches gain more than 15 points in the geometric mean metric, enabling them to obtain the first positions in the ranking among the algorithms. The representation of the differences between the use and absence of preprocessing is depicted in Fig. 19.

To carry out the statistical study we first check for significant differences among the algorithms using an Iman–Davenport test. The p-value is lower than our level of confidence α and near to zero. Thus, we can conclude that significant differences do exist, proceeding with a Shaffer test. The ranks of the algorithms are presented in Fig. 20, and the results of the multiple comparison test performed on all algorithms are shown in Table XXXVI.

Observing the statistical results from Table XXVIII we may conclude that CORE has the worst classification ability when

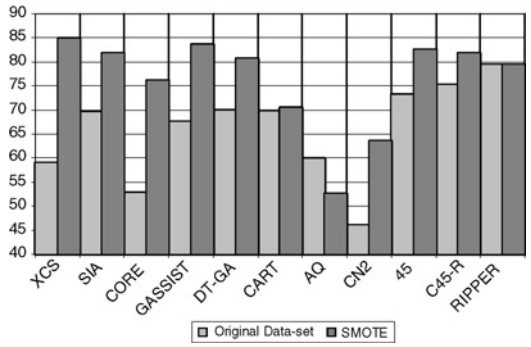


Fig. 19. Graphical representation of the differences in performance between the use of the original data sets and preprocessed with SMOTE.

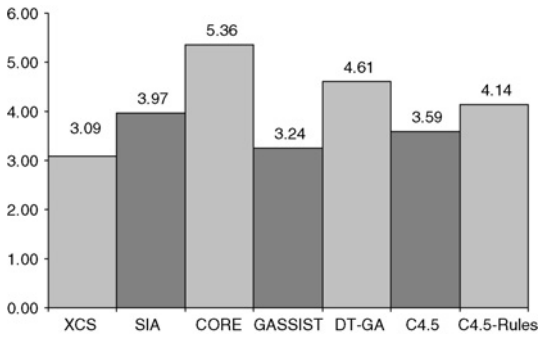


Fig. 20. Ranking computed for the selected GBML and non-evolutionary algorithms in imbalanced data sets.

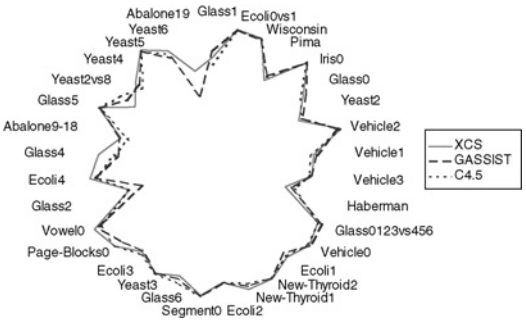


Fig. 21. Star plot representation for XCS, GASSIST, and C4.5 in imbalanced data sets.

compared with the other algorithms, since it is outperformed statistically by most of the remaining algorithms. XCS and GASSIST have good behavior in imbalanced data sets when applying preprocessing, obtaining a higher performance than the non-evolutionary approaches C4.5 and C4.5-Rules, although no significant differences among these methods can be observed. SIA also performs well in the scenario of imbalanced data sets with preprocessing, but it does not statiscally outperform any algorithm. These methods have the best ranking together with C4.5, as depicted in Fig. 20.

Finally, Fig. 21 shows the star plot representation for the three best ranked algorithms in imbalanced data sets with preprocessing, that is, XCS, GASSIST, and C4.5, where the data sets are ordered clockwise according to their degree of imbalance. XCS maintains the most regular behavior in this case.

TABLE XXXVI
SHAFFER TEST FOR GBML ALGORITHMS FOR RULE INDUCTION AGAINST NON-EVOLUTIONARY APPROACHES IN IMBALANCED DATA SETS WITH SMOTE PREPROCESSING

	XCS	SIA	CORE	GASSIST	DT-GA	C4.5	C4.5-Rules
XCS	x	= (1.0)	+ (4.04060E-4)	= (1.0)	= (0.06578)	= (1.0)	= (0.54251)
SIA	= (1.0)	x	= (0.13147)	= (1.0)	= (1.0)	= (1.0)	= (1.0)
CORE	− (4.04060E-4)	= (0.13147)	x	− (9.96881E-4)	= (1.0)	− (0.01287)	= (0.31524)
GASSIST	= (1.0)	= (1.0)	+ (9.96881E-4)	x	= (0.15516)	= (1.0)	= (1.0)
DT-GA	= (0.06578)	= (1.0)	= (1.0)	= (0.15516)	x	= (0.61911)	= (1.0)
C4.5	= (1.0)	= (1.0)	+ (0.01287)	= (1.0)	= (0.61911)	x	= (1.0)
C4.5-Rules	= (0.54251)	= (1.0)	= (0.31524)	= (1.0)	= (1.0)	= (1.0)	x

VI. DISCUSSION: LESSONS LEARNED AND NEW CHALLENGES

This paper has provided an exhaustive review of the main evolutionary algorithms that extract classification models using rule sets as the knowledge representation. We proposed a taxonomy that is structured in three main categories according to the rule representation, namely: 1) chromosome = one rule; 2) chromosome = rule set; and 3) hybrid approaches, and has furthermore contributed to the description of several families inside each category. Thus, the taxonomy presents a general framework in which all the current GBML algorithms for rule induction can be placed. The practitioner could find the taxonomy useful because it places the diversity of GBML methods in a general framework, helping him with the selection of a particular method for the problem at hand. This selection can be made regarding performance, by using the results provided in the paper as a guideline, or with respect to other characteristics such as the type of rules evolved or the computational cost associated with training. We acknowledge that these issues should be further investigated and for this purpose, we make the algorithms, data sets and results available through the KEEL software [53] and Web page (http://sci2s.ugr.es/gbml/index.php) so that the paper can be further extended.

We emphasize five important lessons learned.

1) We identified a representative method with good performance in each one of the families proposed in the taxonomy. Furthermore, we specifically observed that XCS and GASSIST exhibit the best behavior among all GBML approaches. These methods obtained very good average results and the statistical analysis concluded that they outperformed most of the algorithms included in this paper. To our knowledge, this is the first exhaustive experimental study that compares current state-of-the-art GBML with former GBML approaches. Our study places XCS and GASSIST as the two most outstanding learners in the GBML history, which justifies its extended use in the recent literature and in data mining applications.

2) The reasons why an algorithm such as XCS and GASSIST outperform the remaining methods are due to the compound interaction of several learning components. It is difficult to identify single elements that are mostly responsible for the observed good performance. Although the identification of such reasons was not the purpose of this paper, we acknowledge that an introspective examination and internal comparison with other approaches may help understand the results of the algorithms. However, such an analysis can not be carried out from the simple observation of the results of each algorithm on each problem, due to the difficulty in identifying the intrinsic complexities of the data sets. A possible extension of this paper might include the analysis of the geometrical complexity of data sets [97], or the use of synthetic data sets that test particular characteristics of the algorithms, such as in [15].

3) Both XCS and GASSIST obtain very competitive results in terms of classification accuracy. However, XCS tends to evolve less interpretable models than GASSIST. XCS ran populations of 6400 rules, which means that the result is not manageable by the end user, unless a post-processing stage is applied at the risk of losing accuracy. On the other hand, GASSIST presents a good trade-off between precision and interpretability by using a knowledge representation based on the disjunctive normal form and a generalization pressure method based on the minimum description length principle [69].

4) We also observed that XCS's performance is degraded with respect to that of GASSIST in data sets with nominal attributes. This might be due to the different knowledge representations dealing with nominal attributes. In our current implementation, as well as in other previous reports [17], XCS transforms nominal values into numerical ones and then, uses the interval representation designed for numerical attributes. On the contrary, GASSIST uses a string of bits per each category allowed for that attribute, which is better suited to deal with nominal attributes. Thus, the expressive power of the latter representation is higher than that used by XCS. The introduction of this type of representations in XCS remains as a future work.

5) Imbalanced data sets have a clear influence in the performance of the GBML algorithms. We determined that the algorithms that obtained a better behavior in the framework of standard classification are not necessarily those that achieved the best performance for imbalanced data sets. This is due to several reasons. First, if the search process is guided by the standard accuracy rate, search is benefited from applying covering

of majority class examples. If the same algorithm is applied to highly imbalanced datasets, minority class examples would not be favored by this type of search. Second, classification rules that predict the positive class are often highly specialized and thus their coverage is very low, hence they are discarded in favor of more general rules, i.e., those that predict the negative class. Furthermore, it is not easy to distinguish between noise examples and minority class examples, and they can be completely ignored by the classifier. We have found higher differences among the algorithms within a family since only few of them obtained a good performance. Oblique-DT achieves the highest performance among the GBML methods, whereas Ripper gets the best behavior among the classical non-evolutionary methods.

6) Another issue regarding imbalanced data sets is that the use of SMOTE helps the algorithms to deal with extreme imbalance ratios. Our empirical study shows a large performance improvement of most of the algorithms when they are trained using a balanced data set, also obtaining a most robust behavior. The algorithms which show the best behavior with the preprocessed data sets are XCS, SIA, and GASSIST. Furthermore, these methods have superior performance in contrast to the non-evolutionary approaches.

7) The good performance of the GBML methods with respect to the non-evolutionary algorithms shows the suitability and high potential of the search performed by EAs. Since all the non-GBML methods were selected on the basis that they used rule sets as the knowledge representation, the comparison is the closest we can get to compare the power of the search mechanism performed by the EA (in the GBML approaches) with the rest of non-evolutionary based rule inductive algorithms, minimizing the differences due to the knowledge representation (i.e., the language bias).

Throughout this paper, we have identified that the application of GBML algorithms for classification is a promising line of work, but still many challenges need to be addressed.

1) *Scalability:* one of the challenges of data mining is the design of learners that extract information from large data sets [98], namely, data sets with a high number of instances and attributes. In these cases, the learners have efficiency problems and it is necessary to apply scaling up approaches in order to reduce the computation time, but with the premise of obtaining accurate solutions, not just approximate. Evolutionary algorithms also find the same difficulties, which can be addressed in different ways. XCS may offer some advantages due to its online learning architecture, while GASSIST has designed an incremental sampling procedure to minimize the cost of evaluating the rule sets against the complete data set. Also distributed approaches [51], [52] may offer promising solutions which should be further explored. In summary, scalability is an emergent issue that should be addressed explicitly by the GBML community.

2) *Adaptation to Highly Imbalanced Problems:* Most classifiers generally perform poorly on imbalanced data sets because they are designed to minimize the global error rate and, in this manner, they tend to classify almost all instances as negative (i.e., the majority class). In this paper, we have shown the differences between the performance of the algorithms in the framework of standard classification and for imbalanced data sets, identifying the algorithms which are better suited to this specific problem. In this manner, an algorithm specifically designed to take into account the ratio between the examples of the classes may lead to better results in imbalanced domains.

3) *Data Complexity:* The prediction capabilities of classifiers are strongly dependent on the problem's characteristics. An emergent field, the analysis of data complexity, has recently arisen. The data complexity analysis consists in measuring different aspects of the problem which are considered as complex to the classification task [99]. Studies of data complexity metrics applied to particular classification algorithms can be found in [97], [99]–[101]. The complexity of data has been used for characterising the performance of XCS [97], and as a guide to the identification of the domains of competence of classifiers [102]. Thus, it can be considered a new trend in the use of GBML in pattern recognition.

4) *Interpretability:* This is an important challenge for Machine Learning methods because most of the application domains ask for explanations of the model evolved. In this sense, rule-based algorithms have an added value over other techniques. However, it is necessary to analyze the definition of interpretability in depth. Currently, interpretability is measured as the number of rules and the average number of variables per rule, but this cannot be easily used to compare several approaches. In particular, GBML algorithms can produce different types of rule sets and different rule representations, e.g., XCS evolves a set of overlapping rules while GASSIST evolves a decision list. A related line of research is the use of multiobjective genetic algorithms to evolve rule sets with optimal compromise between precision and interpretability.

VII. Concluding Remarks

In this paper, we have presented a taxonomy of GBML algorithms for rule induction based on the codification of the chromosome associated with the learning process of each method. We performed an exhaustive study of the performance of a large set of GBML approaches and the comparison with non-evolutionary learners, both on standard classification problems and on imbalanced data sets.

The main conclusion extracted from our experimental study are the following ones.

1) For standard classification we must emphasize the good behavior observed for XCS and GASSIST in contrast to the remaining GBML methods for rule induction. Furthermore, compared to non-evolutionary algorithms,

we have shown that the GBML approaches are quite competitive and that in many cases the studied evolutionary methods outperform the non-evolutionary state-of-the-art algorithms. This fact supports the conclusion that the use of GBML methods in classification provides interesting results in practice.

2) When raw imbalanced data sets are used, we found that the algorithms that are better suited to manage both the minority and majority class examples are Oblique-DT for the GBML approaches and Ripper for the classical non-evolutionary methods. We must point out that the specific features of the framework of imbalanced data sets imply that the best algorithms are not necessarily the same as those for standard classification, as it was shown along the empirical study.

3) When imbalanced data sets are preprocessed (balanced), we can observe that the average performance is notably increased in most of the algorithms of our study. Also, when the rule discovering process is carried out using the same class distribution, we observe that both XCS and GASSIST are again the algorithms with the best behavior, also getting higher performance and average ranking than the classical non-evolutionary approaches.

Finally, we have discussed some future trends in the field of GBML algorithms such as: scalability when learning from large data sets; the adaptation to data sets with a high imbalance ratio; the application of data complexity for characterising the performance of different GBML methods for specific data sets; and the importance of interpretability in the obtained models.

REFERENCES

[1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[2] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 3rd ed. Amsterdam, The Netherlands: Elsevier, 2006.

[3] J. R. Quinlan, *C4.5: Programs for Machine Learning* San Mateo, CA: Morgan Kaufmann, 1993.

[4] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York: Springer, 1999.

[5] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, 1991.

[6] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proc. 11th Conf. Uncertainty Artif. Intell.*, San Mateo, CA, 1995, pp. 338–345.

[7] V. Cherkassky and F. Mulier, *Learning from Data: Concepts, Theory and Methods*, 2nd ed. New York: Wiley-Interscience. 2007.

[8] J. Fürnkranz, "Separate-and-conquer rule learning," *Artif. Intell. Rev.*, vol. 13, no. 1, pp. 3–54, 1999.

[9] A. A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Berlin, Germany: Springer-Verlag, 2002.

[10] J. J. Grefenstette, *Genetic Algorithms for Machine Learning*. Norwell, MA: Kluwer Academic, 1993.

[11] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Berlin, Germany: Springer-Verlag, 2003.

[12] S. W. Wilson, "Classifier fitness based on accuracy," *Evol. Comput.*, vol. 3, no. 2, pp. 149–175, 1995.

[13] G. Venturini, "SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts," in *Proc. Eur. Conf. Mach. Learn.*, LNAI 667. Berlin, Germany, 1993, pp. 280–296.

[14] C. Z. Janikow, "A knowledge-intensive genetic algorithm for supervised learning," *Mach. Learn.*, vol. 13, nos. 2–3, pp. 189–228, 1993.

[15] E. Bernadó-Mansilla and J. M. Garrell, "Accuracy-based learning classifier systems: Models, analysis and applications to classification tasks," *Evol. Comput.*, vol. 11, no. 3, pp. 209–238, 2003.

[16] J. S. Aguilar-Ruiz, R. Giráldez, and J. C. Riquelme, "Natural encoding for evolutionary supervised learning," *IEEE Trans. Evol. Comput.*, vol. 11, no. 4, pp. 466–479, Aug. 2007.

[17] E. Bernadó-Mansilla, X. Llorá, and J. M. Garrell, "XCS and GALE: A comparative study of two learning classifier systems on data mining," in *Proc. 4th Int. Workshop Adv. Learn. Classifier Syst. (IWLCS)*, LNAI 2321. 2001, pp. 115–132.

[18] J. Bacardit and M. V. Butz, "Data mining in learning classifier systems: Comparing XCS with GAssist," in *Proc. Revised Sel. Papers Int. Workshop Learn. Classifier Syst. (2003–2005)*, LNCS 4399. 2007, pp. 282–290.

[19] J. H. Holland, "Adaptation," in *Progress in Theoretical Biology*, vol. 4, R. Rosen and F. Snell, Eds. New York: Academic, 1976, pp. 263–293.

[20] J. H. Holland and J. S. Reitman, "Cognitive systems based on adaptive algorithms," in *Pattern-Directed Inference Systems*, D. A. Waterman and H. F. Roth, Eds. New York: Academic, 1978, pp. 313–329.

[21] K. A. DeJong and W. M. Spears, "Learning concept classification rules using genetic algorithms," in *Proc. Int. Joint Conf. Artif. Intell.*, 1991, pp. 651–656.

[22] S. F. Smith, "A learning system based on genetic adaptive algorithms," Ph.D. dissertation, Univ. Pittsburgh, Pittsburgh, PA, 1980.

[23] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Pacific Grove, CA, 1984.

[24] R. S. Michalksi, I. Mozetic, and N. Lavrac, "The multipurpose incremental learning system AQ15 and its testing application to three medical domains," in *Proc. 5th Int. Conf. Artif. Intell. (AAAI)*, 1986, pp. 1041–1045.

[25] P. Clark and T. Niblett, "The CN2 induction algorithm," *Mach. Learn.*, vol. 3, no. 4, pp. 261–283, 1989.

[26] J. R. Quinlan, "MDL and categorical theories (continued)," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 464–470.

[27] W. W. Cohen, "Fast effective rule induction," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 115–123.

[28] Q. Yang and X. Wu, "10 challenging problems in data mining research," *Int. J. Inform. Technol. Decision Making*, vol. 5, no. 4, pp. 597–604, 2006.

[29] N. V. Chawla, N. Japkowicz, and A. Kolcz, "Editorial: Special issue on learning from imbalanced data sets," in *Proc. Special Interest Group Knowl. Discovery Data Mining Explorations*, vol. 6, no. 1. 2004, pp. 1–6.

[30] A. Asuncion and D. J. Newman. (2007). *UCI machine learning repository* [Online]. Available: http://www.ics.uci.edu/mlearn/MLRepository.html

[31] J. A. Cohen, "Coefficient of agreement for nominal scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, Apr. 1960.

[32] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[33] S. García and F. Herrera, "An extension on 'Statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons," *J. Mach. Learn. Res.*, vol. 9, pp. 2677–2694, Dec. 2008.

[34] J. H. Holland, "Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems," in *Machine Learning: An Artificial Intelligence Approach*, vol. 2. Los Altos, CA: Morgan Kaufmann, 1986, pp. 593–623.

[35] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artif. Intell.*, vol. 40, nos. 1–3, pp. 235–282, 1989.

[36] D. P. Greene and S. F. Smith, "Competition-based induction of decision models from examples," *Mach. Learn.*, vol. 13, nos. 2–3, pp. 229–257, 1993.

[37] S. F. Smith, "Flexible learning of problem solving heuristics through adaptive search," in *Proc. 8th Int. Joint Conf. Artif. Intell.*, 1983, pp. 422–425.

[38] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[39] K. C. Tan, Q. Yu, and J. H. Ang, "A coevolutionary algorithm for rules discovery in data mining," *Int. J. Syst. Sci.*, vol. 37, no. 12, pp. 835–864, 2006.

[40] L. Jiao, J. Liu, and W. Zhong, "An organizational coevolutionary algorithm for classification," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 67–80, Feb. 2006.

[41] A. L. Corcoran and S. Sen, "Using real-valued genetic algorithms to evolve rule sets for classification," in *Proc. IEEE Conf. Evol. Comput.*, 1994, pp. 120–124.

[42] W.-H. Au, K. C. C. Chan, and X. Yao, "A novel evolutionary data mining algorithm with applications to churn prediction," *IEEE Trans. Evol. Comput.*, vol. 7, no. 6, pp. 532–545, Dec. 2003.

[43] J. Bacardit, D. E. Goldberg, and M. V. Butz, "Improving the performance of a Pittsburgh learning classifier system using a default rule," in *Proc. Revised Sel. Papers Int. Workshop Learn. Classifier Syst. (2003–2005)*, LNCS 4399. 2007, pp. 291–307.

[44] F. Zhu and S. U. Guan, "Ordered incremental training with genetic algorithms," *Int. J. Intell. Syst.*, vol. 19, no. 12, pp. 1239–1256, 2004.

[45] S. U. Guan and F. Zhu, "An incremental approach to genetic-algorithms-based classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 2, pp. 227–239, Apr. 2005.

[46] D. R. Carvalho and A. A. Freitas, "A hybrid decision tree/genetic algorithm method for data mining," *Inform. Sci.*, vol. 163, nos. 1–3, pp. 13–35, 2004.

[47] E. Cantú-Paz and C. Kamath, "Inducing oblique decision trees with evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 1, pp. 54–68, Feb. 2003.

[48] J. B. Gray and G. Fan, "Classification tree analysis using TARGET," *Comput. Statist. Data Anal.*, vol. 52, no. 3, pp. 1362–1372, 2008.

[49] F. Herrera, "Genetic fuzzy systems: Taxonomy, current research trends and prospects," *Evol. Intell.*, vol. 1, no. 1, pp. 27–46, 2008.

[50] C. Zhou, W. Xiao, T. M. Tirpak, and P. C. Nelson, "Evolving accurate and compact classification rules with gene expression programming," *IEEE Trans. Evol. Comput.*, vol. 7, no. 6, pp. 519–531, Dec. 2003.

[51] A. Giordana and F. Neri, "Search-intensive concept induction," *Evol. Comput.*, vol. 3, no. 4, pp. 375–419, 1995.

[52] C. Anglano and M. Botta, "NOW G-net: Learning classification programs on networks of workstations," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 463–480, Oct. 2002.

[53] 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, "KEEL: A software tool to assess evolutionary algorithms to data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.

[54] S. Salzberg, "A nearest hyperrectangle method," *Mach. Learn.*, vol. 6, no. 3, pp. 251–276, May 1991.

[55] R. L. Rivest, "Learning decision lists," *Mach. Learn.*, vol. 2, no. 3, pp. 229–246, 1987.

[56] D. G. Heath, S. Kasif, and S. Salzberg, "Induction of oblique decision trees," in *Proc. 13th Int. Joint Conf. Artif. Intell. (IJCAI)*, 1993, pp. 1002–1007.

[57] E. G. Henrichon and K. Fu, "A nonparametric partitioning procedure for pattern classification," *IEEE Trans. Comput.*, vol. C-18, no. 7, pp. 614–624, Jul. 1969.

[58] M. L. Wong and K. S. Leung, *Data Mining Using Grammar Based Genetic Programming and Applications*. London, U.K.: Kluwer Academic, 2000.

[59] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[60] P. Bonelli and A. Parodi, "An efficient classifier system and its experimental comparison with two representative learning methods on three medical domains," in *Proc. 4th Int. Conf. Genet. Algorithms (ICGA)*, 1991, pp. 288–295.

[61] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson, "Toward a theory of generalization and learning in XCS," *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 28–46, Feb. 2004.

[62] S. W. Wilson, "Generalization in the XCS classifier system," in *Proc. 3rd Annu. Conf. Genet. Programming*, 1998, pp. 665–674.

[63] S. W. Wilson, "Get real! XCS with continuous-valued inputs," in *Festschrift in Honor of John H. Holland*, L. Booker, S. Forrest, M. Mitchell, and R. L. Riolo, Eds. Ann Arbor, MI: Center for the Study of Complex Systems, 1999, pp. 111–121.

[64] J. S. Aguilar-Ruiz, J. C. Riquelme, and M. Toro, "Evolutionary learning of hierarchical decision rules," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 33, no. 2, pp. 324–331, Apr. 2003.

[65] J. S. Aguilar-Ruiz, J. Bacardit, and F. Divina, "Experimental evaluation of discretization schemes for rule induction," in *Proc. Genet. Evol. Comput. (GECCO)*, LNCS 3102. 2004, pp. 828–839.

[66] J. Bacardit and J. M. Garrell, "Evolving multiple discretizations with adaptive intervals for a Pittsburgh rule-based learning classifier system," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, LNCS 2724. 2003, pp. 1818–1831.

[67] J. Bacardit and J. M. Garrell, "Bloat control and generalization pressure using the minimum description length principle for a Pittsburgh approach learning classifier system," in *Proc. Revised Sel. Papers Int. Workshop Learn. Classifier Syst. (2003–2005)*, LNCS 4399. 2007, pp. 59–79.

[68] K. A. DeJong, W. M. Spears, and D. F. Gordon, "Using genetic algorithms for concept learning," *Mach. Learn.*, vol. 13, nos. 2–3, pp. 161–188, Nov.–Dec. 1993.

[69] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, Sep. 1978.

[70] S. K. Murthy, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *J. Artif. Intell. Res.*, vol. 2, no. 1, pp. 1–32, 1994.

[71] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation," in *Proc. Australian Conf. Artif. Intell.*, LNCS 4304. 2006, pp. 1015–1021.

[72] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification," *Pattern Recognit. Lett.*, vol. 30, no. 1, pp. 27–38, 2009.

[73] R. Barandela, J. S. Sánchez, V. García, and E. Rangel, "Strategies for learning in class imbalance problems," *Pattern Recognit.*, vol. 36, no. 3, pp. 849–851, 2003.

[74] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Reading, MA: Addison-Wesley, 1999.

[75] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 3, pp. 299–310, Mar. 2005.

[76] W. Youden, "Index for rating diagnostic tests," *Cancer*, vol. 3, no. 1, pp. 32–35, 1950.

[77] A. Ben-David, "A lot of randomness is hiding in accuracy," *Eng. Appl. Artif. Intell.*, vol. 20, no. 7, pp. 875–885, Oct. 2007.

[78] T. C. W. Landgrebe and R. P. W. Duin, "Efficient multiclass ROC approximation by decomposition via confusion matrix perturbation analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 810–822, May 2008.

[79] E. Alpaydin, *Introduction to Machine Learning*. Cambridge, MA: MIT Press, 2004.

[80] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Mach. Learn.*, vol. 40, no. 3, pp. 203–228, 2000.

[81] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco, CA: Morgan Kaufmann, 2005.

[82] T. Fawcett and F. J. Provost, "Adaptive fraud detection," *Data Mining Knowl. Discovery*, vol. 1, no. 3, pp. 291–316, 1997.

[83] Y. M. Huang, C. M. Hung, and H. C. Jiau, "Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem," *Nonlinear Anal. Real World Applicat.*, vol. 7, no. 4, pp. 720–747, 2006.

[84] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance," *Neural Netw.*, vol. 21, nos. 2–3, pp. 427–436, 2008.

[85] G. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *J. Artif. Intell. Res.*, vol. 19, pp. 315–354, Oct. 2003.

[86] V. García, R. A. Mollineda, and J. S. Sánchez, "On the k-NN performance in a challenging scenario of imbalance and overlapping," *Pattern Anal. Applicat.*, vol. 11, nos. 3–4, pp. 269–280, 2008.

[87] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," in *Proc. Special Interest Group Knowl. Discovery Data Mining Explorations*, vol. 6, no. 1. 2004, pp. 20–29.

[88] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[89] J. Y. Ching, A. K. C. Wong, and K. C. C. Chan, "Class-dependent discretization for inductive learning from continuous and mixed-mode data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 7, pp. 641–651, Jul. 1995.

[90] S. García, A. Fernández, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based

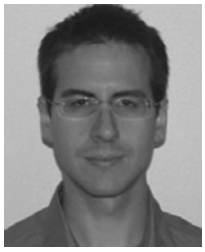
machine learning: Accuracy and interpretability," *Soft Comput.*, vol. 13, no. 10, pp. 959–977, 2009.
[91] D. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, 2nd ed. Boca Raton, FL: Chapman and Hall/CRC, 2006.
[92] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.
[93] J. P. Shaffer, "Modified sequentially rejective multiple test procedures," *J. Am. Statist. Assoc.*, vol. 81, no. 395, pp. 826–831, 1986.
[94] A. Fernández, J. Luengo, J. Derrac, J. Alcalá-Fdez, and F. Herrera, "Implementation and integration of algorithms into the KEEL data-mining software tool," in *Proc. 10th Int. Conf. Intell. Data Eng. Automated Learn. (IDEAL)*, LNCS 5788. 2009, pp. 562–569.
[95] A. Orriols-Puig and E. Bernadó-Mansilla, "Evolutionary rule-based systems for imbalanced datasets," *Soft Comput.*, vol. 13, no. 3, pp. 213–225, 2008.
[96] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," *Comput. Intell.*, vol. 20, no. 1, pp. 18–36, 2004.
[97] E. Bernadó-Mansilla and T. K. Ho, "Domain of competence of XCS classifier system in complexity measurement space," *IEEE Trans. Evol. Comput.*, vol. 9, no. 1, pp. 82–104, Feb. 2005.
[98] F. Provost and V. Kolluri, "A survey of methods for scaling up inductive algorithms," *Data Mining Knowl. Discovery*, vol. 3, no. 2, pp. 131–169, 1999.
[99] M. Basu and T. K. Ho, *Data Complexity in Pattern Recognition*. New York: Springer, 2006.
[100] R. Baumgartner and R. L. Somorjai, "Data complexity assessment in undersampled classification," *Pattern Recognit. Lett.*, vol. 27, no. 12, pp. 1383–1389, 2006.
[101] J. S. Sánchez, R. A. Mollineda, and J. M. Sotoca, "An analysis of how training data complexity affects the nearest neighbor classifiers," *Pattern Anal. Applicat.*, vol. 10, no. 3, pp. 189–201, 2007.
[102] E. Bernadó-Mansilla and T. K. Ho, "On classifier domains of competence," in *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, vol. 1. 2004, pp. 136–139.

Alberto Fernández received the M.S. degree in computer science from the University of Granada, Granada, Spain, in 2005, where he is currently working toward the Ph.D. degree in linguistic fuzzy rule based classification systems applied to problems with imbalanced classes from the Department of Computer Science and Artificial Intelligence.

He holds a scholarship from the Spanish Ministry of Science and Technology. His research interests include data mining, classification in imbalanced domains, fuzzy rule learning, evolutionary algorithms, and multiclassification problems.

Salvador García received the M.S. and Ph.D. degrees in computer science from the University of Granada, Granada, Spain, in 2004 and 2008, respectively.

He is currently an Assistant Professor with the Department of Computer Science, University of Jaén, Jaén, Spain. His research interests include data mining, data reduction, data complexity, imbalanced learning, statistical inference, and evolutionary algorithms.



Julián Luengo received the M.S. degree in computer science from the University of Granada, Granada, Spain, in 2006, where he is currently working toward the Ph.D. degree in data preparation and data complexity in knowledge discovery and data mining from the Department of Computer Science and Artificial Intelligence.

He is currently a Research Fellow with the Spanish Ministry of Science and Innovation, the Spanish Ministry for Scientific Research. His research interests include data mining, data preparation in knowledge discovery and data mining, missing values, data complexity, evolutionary algorithms, and fuzzy systems.

Ester Bernadó-Mansilla received the B.S. degree in telecommunications engineering, the M.S. degree in electronic engineering, and the Ph.D. degree in computer science from Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Barcelona, Spain, in 1992, 1995, and 2002, respectively.

She is currently an Associate Professor with the Grup de Recerca en Sistemes Intelligents, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull. She has co-edited two books on genetic-based machine learning. Her research interests include machine learning, pattern recognition, data mining, genetic algorithms, and genetic-based machine learning.

Dr. Bernadó-Mansilla serves as an Associate Editor of the *Pattern Recognition Letters*.

Francisco Herrera received the M.S. and Ph.D. degrees in mathematics from the University of Granada, Granada, Spain, in 1988 and 1991, respectively.

He is currently a Professor with the Department of Computer Science and Artificial Intelligence, University of Granada. He has published more than 150 papers in international journals. He is the coauthor of the book *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases* (World Scientific, 2001). He has co-edited five international books and 20 special issues in international journals on different soft computing topics. His current research interests include computing with words and decision making, data mining, data preparation, instance selection, fuzzy rule-based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, memetic algorithms, and genetic algorithms.

Dr. Herrera is an Associated Editor of the following journals: IEEE TRANSACTIONS ON FUZZY SYSTEMS, the *Journal of Information Sciences*, *Mathware and Soft Computing*, *Advances in Fuzzy Systems*, *Advances in Computational Sciences and Technology*, and the *International Journal of Applied Metaheuristics Computing*. He currently serves as an Area Editor of the *Journal of Soft Computing* (in the area of genetic algorithms and genetic fuzzy systems), and he serves as a member of several journal editorial boards, including those of: *Fuzzy Sets and Systems*, *Applied Intelligence*, *Knowledge and Information Systems*, *Information Fusion*, *Evolutionary Intelligence*, the *International Journal of Hybrid Intelligent Systems*, *Memetic Computation*.