COST-SENSITIVE LEARNING OF FUZZY RULES FOR IMBALANCED CLASSIFICATION PROBLEMS USING FURIA

ANA PALACIOS

Dpto. de Ciencias de la Computación e I. A., Univ. de Granada, Spain

KRZYSZTOF TRAWIŃSKI

European Centre for Soft Computing, Asturias, Spain

OSCAR CORDÓN

Dpto. de Ciencias de la Computación e I. A. and CITIC-UGR, Univ. de Granada, Spain and European Centre for Soft Computing, Asturias, Spain

LUCIANO SÁNCHEZ

Dpto. de Informática, Univ. de Oviedo, Spain

Received (received date)
Revised (revised date)

This paper is intended to verify that cost-sensitive learning is a competitive approach for learning fuzzy rules in certain imbalanced classification problems. It will be shown that there exist cost matrices whose use in combination with a suitable classifier allows for improving the results of some popular data-level techniques. The well known FURIA algorithm is extended to take advantage of this definition. A numerical study is carried out to compare the proposed cost-sensitive FURIA to other state-of-the-art classification algorithms, based on fuzzy rules and on other classical machine learning methods, on 64 different imbalanced datasets.

Keywords: Fuzzy Rule Learning; Imbalanced datasets; Cost-sensitive learning; FURIA; SMOTE

1. Introduction

The problem of imbalanced datasets in classification or "datasets with rare classes" occurs when the number of instances of a class is much lower than that of the other classes[49]. In these problems it often happens that the minority class is the most interesting. However, minimum-error oriented classifiers tend to ignore the minority class and produce wrong conclusions[17,29,38,40,49]. This happens in many applications such as medical diagnosis[35], fraud detection[39], risk management[24], among others.

Solving the imbalanced learning problem consists of reducing the false negatives as much as possible without increasing too much the number of false positives. The strategies for achieving this objective can be grouped into two principal categories[11]:

cost-sensitive learning or internal approach and *data-level* or external approach. For internal methods, classifiers optimizing criteria different than the expected error rate are sought. For example, the minimum risk Bayes rule[4] is implicit or explicitly adopted in certain methods[13,14,17,53] where a higher risk (proportional to the imbalance ratio, i.e., to the ratio between the a priori probabilities for the minority class and the remaining classes) is assigned to misclassifications in the minority class. In contrast, in external methods, data is preprocessed for equalizing the prior probabilities of the classes. Oversampling, undersampling or combinations of both are used for rebalancing false positives and negatives[3,5,44].

Other authors[37] suggest that for every performance criteria, for example area under the ROC curve[10,23], or arithmetic or geometric mean of the confusion matrix diagonal[31], a cost matrix can be found for which the optimal classifier coincides with the minimum risk Bayes rule. However, the method for computing this cost matrix is still undefined. Lastly, there are not many publications detailing numerical experimentations where the performance of both internal and external approaches are compared. It is worth mentioning that some authors claim that cost-sensitive learning does not improve preprocessing algorithms, albeit the differences found in these studies were not statistically significant[32].

Multiple studies regarding fuzzy rule-based classification systems (FRBCSs) have been published. Learning fuzzy rules or fuzzy decision trees from imbalanced datasets has been solved with scalar[9,32,36,42,47,48,51] and multi-objective techniques[16,19]. In particular, imbalanced classification has been regarded as a multi-objective problem, where accuracy and complexity are balanced and the ROC convex hull used to select a good trade-off[16]. An external approach has also been shown to produce good results[18,20,21,22]. In the current contribution it will be shown that cost-sensitive learning can be at least as effective or even better than preprocessing the data. For this purpose, the Fuzzy Unordered Rule Induction Algorithm (FURIA)[25,27] will be generalized to cost-sensitive learning. In addition, two heuristics are proposed for defining the cost matrix in terms of the classification problem imbalance ratio. The results of this new algorithm, that will be called FURIA cost-sensitive (FURIA_CS), will be compared to those of FURIA on datasets that have been rebalanced with state-of-the-art methods, including Synthetic Minority Oversampling Technique (SMOTE)[5] and its variant with the Wilson's Edited Nearest Neighbor rule (ENN)[52]. These techniques have been chosen because of their robust behaviours[3,18].

This paper is organized as follows. Section 2 introduces the problem of imbalanced datasets. Preprocessing methods, cost-sensitive learning, and the employed metrics are defined in this part. Section 3 recalls the parts of the FURIA algorithm relevant to this study. Section 4 introduces FURIA_CS and makes a detailed description of the effected changes. In Section 5, numerical results are provided. FURIA_CS is compared to a combination of FURIA with preprocessing and to other selected state-of-the-art classification algorithms. The paper concludes in Section 6.

Table 1. Confusion matrix for two classes problems

	Positive class	Negative class
Positive Prediction	True Positive (TP)	False Positive (FP)
Negative Prediction	False Negative (FN)	True Negative (TN)

2. Imbalanced classification problem. Notation and metrics for two-classes problem

In two-classes problems, the confusion matrix divides the results of classifying a set of instances into four different categories, as shown in Table 1: true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

The fraction of misclassified instances is

$$\mathrm{Err} = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{1}$$

while the accuracy is 1-Err.

For independently measuring the classification quality for positive and negative classes, the following values are defined:

$$\begin{aligned} \mathrm{TP}_{rate} &= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad & \mathrm{FN}_{rate} &= \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}} \\ \mathrm{TN}_{rate} &= \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \quad & \mathrm{FP}_{rate} &= \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}} \end{aligned} \tag{2}$$

and the terms "specificity" or $\mathrm{acc} = \mathrm{TN}_{rate}$, and "sensitivity" or $\mathrm{acc}^{+} = \mathrm{TP}_{rate}$ are commonly used.

Learning algorithms minimizing the fraction of misclassified instances tend to produce classifiers where TN_{rate} is too low[40,49]. For this reason, criteria more appropiate than the average classification error are considered[31]. The most common metrics for imbalanced, two-classes problems are:

- The geometric mean (GM)[31] of the sensitivity and the specificity. GM is an interesting indicator of the quality of a classifier for imbalanced data, because it is high when both acc^{+} and acc are high or when the different between acc^{+} and acc is small[30].
- The Area Under the ROC Curve (AUC)[10,23] which is a trade-off between benefits (TP_{rate}) and costs (FP_{rate}). AUC is approximated by the value that follows:

$$\mathrm{AUC} = \frac{1 + \mathrm{TP}_{rate} - \mathrm{FP}_{rate}}{2} \tag{3}$$

As already mentioned, there are two approaches for solving imbalanced classification problems: cost-sensitive learning and preprocessing for equalizing the prior probabilities of the classes. Both will be described in the following sections.

2.1. *Cost-sensitive learning*

Cost-sensitive learning[14,17] can be categorized into two classes[45] :

- Class-dependent costs[14,17,50,55]. The cost depends on the pair (true class, assigned class).
- Example-dependent costs[1,33,34,53,54]. Different examples can have different misclassification costs, irrespectively of their true classes or the classes they are assigned.

In this paper, classifiers of the first category are used. These classifiers depend on a cost matrix C, where $C(i,j)$ is the cost of assigning the class i to an example whose true class is j. In binary classification problems, the notation $C(+,-)$ is used for naming the cost of misclassifying a positive (minority class) example, and $C(-,+)$ is the cost of the opposite case. It is needed that the cost of misclassifying instances of the minority class is higher or equal than the cost of misclassifying the majority class, i.e. $C(+,-) \geq C(-,+)$. It is intuitive, but not mandatory that $C(-,-) = C(+,+) = 0$[17,45]. Heuristic cost assignments are common[43,44].

2.2. *Preprocessing imbalanced datasets. SMOTE and SMOTE+ENN algorithms*

In this paper, the SMOTE algorithm[5], and a hybrid approach, SMOTE+ENN[3] are used. In the SMOTE algorithm, the minority class is over-sampled. New synthetic instances are introduced along the line segments joining any or all of the nearest neighbors of each instance in the minority class. SMOTE+ENN is a variant of SMOTE where Wilson's ENN Rule[52] is used after oversampling for removing from the training set any example whose class is not in agreement with its three nearest neighbours.

3. FURIA outline

Fuzzy Unordered Rules Induction Algorithm (FURIA)[25,27] is a novel fuzzy rule-based classification method extending the classical RIPPER[7]. The most important differences between FURIA and RIPPER concern the type of of rule model and the use of default rules[27].

With respect to the rule model type, FURIA performs a fuzzification of the rule antecedents, using a greedy algorithm that extends the support of each rule so as to improve a purity criteria measuring the component-wise confidence of the fuzzy classification rule. With respect to the use of default decisions, rules in RIPPER are in ascending order by the prior probability of the classes in their consequents. The first rule matching the query pattern is used for classifying it. Uncovered examples are assigned to the most frequent class (default rule). In contrast, FURIA uses a one-vs-rest decomposition. No default rule is needed and the order of the classes is irrelevant, but uncovered instances may happen. When a query instance is uncovered by the fuzzy classification rules derived from FURIA, the nearest rule in the fuzzy knowledge base is applied to the query. This fuzzy rule is determined by a process called "rule stretching", where all rules are gradually generalized until one of the stretched antecedents is satisfied by the uncovered instance.

In order to make this paper more self-contained, an algorithmic description of FURIA is included below, where the parts that will be altered in the cost-based learning generalization (see Section 4) are marked in boldface. The interested reader is referred to the original references[25,27] and also to the source code of the software implementation provided by the authors[26] for a full description of FURIA.

The outer loop of the FURIA algorithm is as follows:

```
Method FURIA()
   Select a class and learn crisp classification rules discriminating
      this class from the others (call method RuleSetForOneClass())
   Remove redundant antecedents
   Fuzzify rules maximizing the purity of the fuzzification of each attributte
   Compute confidence degrees for all rules considering the certainty factor
   Evaluate rules and apply rule stretching if there are uncovered examples
End of Method
```

This schema needs not to be altered in order to introduce classification costs, however there are three parts that need a new, cost-based definition:

(1) the rule purity, that quantifies the quality of the fuzzification procedure, depends on the costs of the partially covered examples
(2) the certainty factor, that measures the confidence assigned to the piece of information described by the rule, depends also on the costs
(3) the rule stretching procedure, that is used to simplify the antecedents for improving generalization, should not depend on the number of examples covered by the rule but on their relative costs.

Second, the method `RuleSetForOneClass()` referenced before, contains a pruning stage that depends on the cost matrix too. The pseudocode of this method is as follows:

```
Method RuleSetForOneClass()
   While StoppingConditions() == false do
     Call method RuleGrowing()
     If StoppingConditions() == true then
       Delete the newly created rule
     End If
   End While
   Perform rule pruning.
End of method
```

Third, the method `RuleGrowing()` is based on a measure of information gain.

The information gain in error-based classification depends on the probabilities of the classes, nonetheless probabilities must be replaced by expected costs in this context. The pseudocode of this method follows:

```
Method RuleGrowing()
    Grow rule using an information gain measure to choose the best conjunct
      to be added into the rule antecedent.
    Stop adding conjuncts when the rule starts covering negative instances.
End of method
```

Lastly, the stopping conditions of FURIA are based on classification error and that must be updated to classification risk. These conditions are:

```
Method StoppingConditions()
    If there are not uncovered instances of the current class
      then StoppingConditions=true
    If rule error ≥ 0.5 then StoppingConditions=true
    If the description length of the ruleset is 64 bits greater than
      the smallest found then StoppingConditions=true
    StoppingConditions=false
End of method
```

4. A proposal for a cost-sensitive FURIA algorithm

Those parts marked in boldface in the preceding description will be explained in detail in this section, along with their proposed extensions to cost-based classification. In the following, the training set is $D \subset \mathbb{R}^k$ and instances are vectors $x = (x_1, \ldots, x_k) \in D$. Each antecedent of a FURIA fuzzy classification rule is a multivariate trapezoidal fuzzy set whose membership is

$$I^F(x) = \bigoplus_{i=1,\ldots k} I_i^F(x_i) \tag{4}$$

and its core is the interval $I = I_1 \times \cdots \times I_k$, where the indicator function of I_i, $i = 1, \ldots, k$ is

$$I_i(x_i) = \begin{cases} 1 & \text{if } I_i^F(x_i) = 1 \\ 0 & \text{else.} \end{cases} \tag{5}$$

and the operator $\oplus$ is the fuzzy addition,

$$\mu_{A \oplus B}(x) = \sup_{a+b=x} \{\alpha \mid \min(\mu_A(a), \mu_B(b)) \geq \alpha\} \tag{6}$$

4.1. *Information gain*

This criterion measures the improvement of a rule with respect to the default for the target class and is used as a stopping condition in the rule growing procedure.

Let I be the core of the antecedent of the rule at hand, and let l be the target class. Then, the number of positive examples for the fuzzy classification rule r is

$$p_r = \#\{x \in I \mid \text{class}(x) = l\} \tag{7}$$

and the number of negative examples for that rule is

$$n_r = \#\{x \in I \mid \text{class}(x) \neq l\}. \tag{8}$$

The total number of positive and negative examples in the dataset are named p and n, respectively. Then, the information gain is defined as follows[26]:

$$\text{IG}_r = p_r \times \left(\log_2\left(\frac{p_r + 1}{p_r + n_r + 1}\right) - \log_2\left(\frac{p + 1}{p + n + 1}\right)\right). \tag{9}$$

The information gain depends on the quotient between the expected fraction of instances well classified by the rule at hand and by the default rule, as well as the fraction of the number of positive examples for the rule r and by the number of negative examples for that rule r. These expressions must guard against the division by zero, thus the approximations

$$\frac{p}{p+n} \approx \frac{p+1}{p+n+1} \tag{10}$$

and

$$\frac{p_r}{p_r+n_r} \approx \frac{p_r+1}{p_r+n_r+1} \tag{11}$$

were made in the reference software implementation of FURIA[26].

4.1.1. *Cost-sensitive extension*

The proposed generalization of this expression to cost-based learning consists of replacing the expected fraction of misclassified instances by the expected risk.

Let l be the class in the consequent of the rule being grown and I the core or its antecedent, then the cost-sensitive version of the number of positive examples p_r is defined as follows:

$$p_r^{\text{CS}} = \sum_{x \in I} 1 - C(l, \text{class}(x)). \tag{12}$$

Notice that, if the cost of every misclassification was 1,

$$C(i,j) = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } i \neq j. \end{cases} \tag{13}$$

then $p_r^{\text{CS}} = p_r$, the number of positive examples for the rule at hand. Analogously, the number of positive instances in the dataset is generalized to

$$p^{\text{CS}} = \sum_{x \in D} 1 - C(l, \text{class}(x)). \tag{14}$$

Hence, the cost-based information gain is defined as follows:

$$\mathrm{IG}_r^{\mathrm{CS}} = p_r^{\mathrm{CS}} \times \left(\log_2\left(\frac{p_r^{\mathrm{CS}}+1}{p_r+n_r+1}\right) - \log_2\left(\frac{p^{\mathrm{CS}}+1}{p+n+1}\right) \right). \tag{15}$$

4.2. *Pruning*

Each rule comprises q antecedents, that will be named $a_1, \dots, a_q$. The list $\langle a_1, \dots, a_q \rangle$ makes reference to an AND combination of these antecedents. Antecedents comprise three parts:

- The index of an attribute
- The split point of this attribute
- The condition for comparing the value of the attribute and the split point (lower or equal, higher or equal).

For instance, the antecedent $(2, 3, \leq)$ is true if the value of the second variable is lower or equal than 3.

The order of the antecedents reflects their importance thus pruning a rule consists of selecting a sublist $\langle a_1, \dots, a_i \rangle$, with $i \leq q$. In order to find a suitable value for i, the following rule-value metric is computed first[26]:

$$V_r = \frac{p_r + 1}{p_r + n_r + 2} \tag{16}$$

Let the number of positive covered and negative uncovered examples of the rule, when pruned at the i-th antecedent, respectively be P_i and N_i:

$$P_i = \#\{x \mid x \text{ is covered by } \langle a_1, \dots, a_i \rangle \wedge \mathrm{class}(x) = l\} \tag{17}$$

$$N_i = \#\{x \mid x \text{ is not covered by } \langle a_1, \dots, a_i \rangle \wedge \mathrm{class}(x) \neq l\}. \tag{18}$$

and let be defined the value[26]

$$\mathrm{worth}_i = \frac{P_i + N_i}{p + n} \tag{19}$$

This value measures how likely is each antecedent to be pruned. If

$$\max_{i=1,\dots,q} \mathrm{worth}_i > V_r, \tag{20}$$

then the term where the value of "worth$_i$" is maximum is selected for pruning.

4.2.1. *Cost-sensitive extension*

The extension of the value defined in Eq. 16 is

$$V_r^{\mathrm{CS}} = \frac{p_r^{\mathrm{CS}} + 1}{p_r + n_r + 2}. \tag{21}$$

Table 2. Dataset for example 1

Instance	x_0	x_1	x_2	Class
1	1	1	3	1
2	1	2	4	**2**
3	2	2	4	1
4	1	1	2	1
5	3	3	3	1
6	3	4	3	1

In addition, the worth concept is also extended as follows:

$$\text{worth}_i^{\text{CS}} = \frac{P_i^{\text{CS}} + N_i^{\text{CS}}}{p+n} \tag{22}$$

where

$$P_i^{\text{CS}} = \sum_{\{x \text{ is covered by } \langle a_1 \ldots a_i \rangle\}} 1 - C(l, \text{class}(x)) \tag{23}$$

$$N_i^{\text{CS}} = \sum_{\{x \text{ is not covered by } \langle a_1 \ldots a_i \rangle\}} C(l, \text{class}(x)) \tag{24}$$

The following example clarifies the meaning of this generalized pruning in an imbalanced classification context.

Example 1. Let $D \subset \mathbb{R}^3$ be the dataset in Table 2, comprising 6 instances of classes 1 (majority) and 2 (minority). The list of antecedents of the rule to be pruned is $\langle a_1, a_2 \rangle$. For example, a_1 and a_2 are as follows:

$$a_1 = (1, 2, \leq) \tag{25}$$

$$a_2 = (2, 2, \leq). \tag{26}$$

The consequent of the rule is "class is 2". Instances #1,#2,#3 and #4 are compatible with a_1. Instance #4 is compatible with both a_1 and a_2 .

Applying Eq. 19, the following results are obtained:

$$\text{worth}_1 = \frac{P_1 + N_1}{p+n} = \frac{1+2}{6} = 0.5. \tag{27}$$

$$\text{worth}_2 = \frac{P_2 + N_2}{p+n} = \frac{0+4}{6} = 0.66. \tag{28}$$

Since worth_2 is greater than worth_1, the cut point is $i = 2$ and therefore the rule is not pruned.

Suppose that the following cost matrix is adopted:

	Positive class	Negative class
Positive Prediction	0	0.25
Negative Prediction	1	0

Applying eq. (22), the results are:

$$\text{worth}_1^{\text{CS}} = \frac{P_1^{\text{CS}} + N_1^{\text{CS}}}{p+n} = \frac{3.25+0.5}{6} = 0.625 \tag{29}$$

$$\text{worth}_2^{\text{CS}} = \frac{P_2^{\text{CS}} + N_2^{\text{CS}}}{p+n} = \frac{0.75+1}{6} = 0.29. \tag{30}$$

In this case, $\text{worth}_1^{\text{CS}}$ is greater than $\text{worth}_2^{\text{CS}}$ and the rule is pruned at $i = 1$. The higher cost assigned to the misclassification of the minority class (4 times higher than the opposite) produces a pruning where the simplified rule covers instance #2, the element of the minority class in the dataset. By contrast, instance #2 was not covered by the pruned rule if an error-based approach was followed, as seen in the first part of this example.

4.3. ***Purity***

This value measures the quality of the fuzzification procedure and it is used for determining the support of the fuzzy sets defining the rule antecedents. Let D^i be the subset of the training data that follows:

$$D^i = \{(x_1, \ldots, x_k) \mid x_j \in I_j^F(x_j) \text{ for all } j \neq i\}. \tag{31}$$

D is partitioned into positive and negative instances, D_+^i and D_-^i. Given the values

$$p_i = \sum_{x \in D_+^i} I_i^F(x_i) \tag{32}$$

$$n_i = \sum_{x \in D_-^i} I_i^F(x_i), \tag{33}$$

the purity of the fuzzification of the i-th attribute is[26]:

$$\text{pur}_r = \frac{p_i}{p_i + n_i} \tag{34}$$

4.3.1. *Cost-sensitive extension*

The extension of Eq. 34 to cost-sensitive learning is

$$\text{pur}_r^{\text{CS}} = \frac{p_i^{\text{CS}}}{p_i + n_i} \tag{35}$$

where

$$p_i^{\text{CS}} = \sum_{x \in D^i} I_i^F(x_i)(1 - C(l, \text{class}(x))) \tag{36}$$

and p_i, n_i, were defined in Eqs. 32 and 33.

4.4. *Certainty factor*

The certainty factor CF of a rule $\langle I^F, l\rangle$, for a training set D_T, is[26]:

$$\mathrm{CF} = \frac{2 \cdot \dfrac{\sum\limits_{x \in D_T,\, \mathrm{class}(x)=l} p(x)}{\sum\limits_{x \in D_T} p(x)} + \sum\limits_{x \in D_T,\, \mathrm{class}(x)=l} I^F(x)}{2 + \sum\limits_{x \in D_T} I^F(x)} \tag{37}$$

where $p(x)$ is the weight of instance x, often 1. It is remarked that the FURIA algorithm is able to learn from a weighed dataset where the contribution of each instance to the total classification error is a preset value, however these weights $p(x)$ are not related to the cost matrix neither they evolve during the learning process.

4.4.1. *Cost-sensitive extension*

The cost-sensitive certainty factor of a rule $\langle I^F, l\rangle$, for a training set D_T, is:

$$\mathrm{CF}^{\mathrm{CS}} = \frac{2 \cdot \overline{\mathrm{acc}}^{\mathrm{CS}} + \sum_{x \in D_T} I^F(x)(1 - C(l, \mathrm{class}(x))}{2 + \sum\limits_{x \in D_T} I^F(x)} \tag{38}$$

where

$$\overline{\mathrm{acc}}^{\mathrm{CS}} = \frac{\sum\limits_{x \in D_T} p(x)(1 - C(l, \mathrm{class}(x)))}{\sum\limits_{x \in D_T} p(x)} \tag{39}$$

and $p(x)$ is the weight of instance x, mentioned before.

4.5. *Rule stretching*

Rule stretching (or generalization) deals with uncovered examples (those classified by the default rule in RIPPER). The generalization procedure consists of making (preferably minimal) simplifications of the antecedents of the rules until the query instance is covered. The instance is then classified by the rule with the highest evaluation, according to the value[26]

$$\mathrm{STR} = \mathrm{CF} \cdot \frac{k+1}{m+2} \cdot I^F(x) \tag{40}$$

where k is the size of the generalized antecedent and m is the size of the entire antecedent before applying this procedure. Notice that, $\frac{k+1}{m+2}$ aims at discarding heavily pruned rules. If no streched rule is able to cover the given example x_i, it is assigned a class based on the *a priori* distribution.

4.5.1. *Cost-sensitive extension*

The cost-sensitive extension of Eq. 40 is straightforward:

$$\mathrm{STR}^{\mathrm{CS}} = \mathrm{CF}^{\mathrm{CS}} \cdot \frac{k+1}{m+2} \cdot I^F(x)(1 - C(l, \mathrm{class}(x))). \tag{41}$$

If the query example cannot be covered by any stretched rule, the class with minimum *a priori* risk is chosen. The risk of a class $\lambda = 1, \ldots, q$ is estimated as follows:

$$\mathrm{risk}(\lambda) = \sum_{x \in D_T} C(\lambda, \mathrm{class}(x)) \tag{42}$$

4.6. *Stopping conditions*

The three following are considered:

(1) There are no more uncovered positive examples in the dataset.
(2) The description length of the ruleset is 64 bits greater than the smallest value met so far.
(3) The number of false positives of a rule, divided by the number of covered instances, is greater or equal than 0.5. In other words, the error rate of the rule is greater or equal than 0.5.

In this section, a cost-sensitive adaptation of FURIA, named FURIA_CS is described. This adaptation is designed for tackling imbalanced classification problems. The extended algorithm depends on a cost matrix C, where $C(i, j)$ is the cost of assigning the i-th class to an example whose true class is j, as discussed in Section 2.1. Without loss of generality, it will be assumed that $C(i, j) \leq 1$ for all i, j.

The expressions used in the preceding section for computing information gain, pruning, purity, certainty factor and rule stretching, as well as the stopping conditions, must be adapted to reflect these costs, as described in the paragraphs that follow.

4.6.1. *Cost-sensitive extension*

The algorithm proposed here must be stopped when the risk of the rule is higher than certain threshold relative to the maximum risk. It is proposed that the learning will be ended when the error rate of the rule surpasses the STC values defined below, which are based on the imbalance ratio (IR) of the current one vs. others classification problem:

- If the consequent of the rule is the majority class,

$$\mathrm{STC}_{\mathrm{maj}} = \begin{cases} \dfrac{1}{\mathrm{IR}} & \text{if IR} > 2 \\ 0.5 & \text{else.} \end{cases} \tag{43}$$

- If the consequent of the rule is the minority class,

$$\mathrm{STC}_{\min} = \begin{cases} 1 - \dfrac{1}{\mathrm{IR}} & \text{if IR} > 2 \\ 0.5 & \text{else.} \end{cases} \tag{44}$$

5. Experimental study

The purpose of the experimental study is to show that cost-sensitive algorithms are competitive against state-of-the-art preprocessing algorithms when learning fuzzy rules for imbalanced classification problems. The experimental setup, comprising a description of datasets, data partitions, selected classifiers of different types, parameters of the classifiers, and misclassification costs is described in Section 5.1. In Section 5.2, the performance of FURIA_CS, FURIA+SMOTE and FURIA+SMOTE_ENN is compared for two different misclassification costs and cost matrices. Lastly, in Section 5.3, FURIA_CS is compared to the results of several classifiers of different types: C4.5[41], SVM[46], FH-GBML[28] and k-NN[8].

5.1. *Experimental setup: Datasets, data partitions and parameters*

Sixty-four binary classification problems from the KEEL dataset repository[2] were selected. The imbalance ratio of all of them is higher than 1.8 and the datasets are divided into three categories: low (IR $<$ 9), medium (9 $\leq$ IR $<$ 11) and high (IR $\geq$ 11). Their properties are summarized in Table 3, where "#Ex." represents the number of examples, "#Atts." the number of attributes, "Class(-,+)" the name of each class, "%Class(-,+)" the percentage of each class, and "IR" the class distribution (i.e., the imbalance ratio). The outcomes of the application of SMOTE and SMOTE+ENN to these datasets have also been obtained from the same data repository.

Table 3: Summary of the imbalanced datasets.

Data-sets	#Ex.	#Atts.	Class (-,+)	%Class(-;+)	IR
Glass1	214	9	(build-win-non float-proc; remainder)	(35.51, 64.49)	1.82
Ecoli0vs1	220	7	(im; cp)	(35.00, 65.00)	1.86
Wisconsin	683	9	(malignant; benign)	(35.00, 65.00)	1.86
Pima	768	8	(tested-positive; tested-negative)	(34.84, 66.16)	1.90
Iris0	150	4	(Iris-Setosa; remainder)	(33.33, 66.67)	2.00
Glass0	214	9	(build-win-float-proc; remainder)	(32.71, 67.29)	2.06
Yeast1	1484	8	(nuc; remainder)	(28.91, 71.09)	2.46
Vehicle1	846	18	(Saab; remainder)	(28.37, 71.63)	2.52
Vehicle2	846	18	(Bus; remainder)	(28.37, 71.63)	2.52
Vehicle3	846	18	(Opel; remainder)	(28.37, 71.63)	2.52
Haberman	306	3	(Die; Survive)	(27.42, 73.58)	2.68
Glass0123vs456	214	9	(non-window glass; remainder)	(23.83, 76.17)	3.19
Vehicle0	846	18	(Van; remainder)	(23.64, 76.36)	3.23
Ecoli1	336	7	(im; remainder)	(22.92, 77.08)	3.36
New-thyroid2	215	5	(hypo; remainder)	(16.89, 83.11)	4.92
New-thyroid1	215	5	(hyper; remainder)	(16.28, 83.72)	5.14
Ecoli2	336	7	(pp; remainder)	(15.48, 84.52)	5.46
Segment0	2308	19	(brickface; remainder)	(14.26, 85.74)	6.01
Glass6	214	9	(headlamps; remainder)	(13.55, 86.45)	6.38
Yeast3	1484	8	(me3; remainder)	(10.98, 89.02)	8.11

Continued on next page

Table 3 – *Continued from previous page*

Data-sets	#Ex.	#Atts.	Class (-,+)	%Class(-;+)	IR
Ecoli3	336	7	(imU; remainder)	(10.88, 89.12)	8.19
Page-blocks0	5472	10	(remainder; text)	(10.23, 89.77)	8.77
Ecoli034vs5	200	7	(p,imL,imU; om)	(10.00, 90.00)	9.00
Yeast2vs4v	514	8	(cyt; me2)	(9.92, 90.08)	9.08
Ecoli067vs35	222	7	(cp,omL,pp; imL,om)	(9.91, 90.09)	9.09
Ecoli0234vs5	202	7	(cp,imS,imL,imU; om)	(9.90, 90.10)	9.10
Glass015vs2	172	9	(build-win-non float-proc,tableware, build-win-float-proc; ve-win-float-proc)	(9.88, 90.12)	9.12
Yeast0359vs78	506	8	(mit,me1,me3,erl; vac,pox)	(9.88, 90.12)	9.12
Yeast02579vs368	1004	8	(mit,cyt,me3,vac,erl; me1,exc,pox)	(9.86, 90.14)	9.14
Yeast0256vs3789	1004	8	(mit,cyt,me3,exc; me1,vac,pox,erl)	(9.86, 90.14)	9.14
Ecoli046vs5	203	6	(cp,imU,omL; om)	(9.85, 90.15)	9.15
Ecoli01vs235	244	7	(cp,im; imS,imL,om)	(9.83, 90.17)	9.17
Ecoli0267vs35	224	7	(cp,imS,omL,pp; imL,om)	(9.82, 90.18)	9.18
Glass04vs5	92	9	(build-win-float-proc,containers; tableware)	(9.78, 90.22)	9.22
Ecoli0346vs5	205	7	(cp,imL,imU,omL; om)	(9.76, 90.24)	9.25
Ecoli0347vs56	257	7	(cp,imL,imU,pp; om,omL)	(9.73, 90.27)	9.28
Yeast05679vs4	528	8	(me2; mit,me3,exc,vac,erl)	(9.66, 90.34)	9.35
Ecoli067vs5	220	6	(cp,omL,pp; om)	(9.09, 90.91)	10.00
Vowel0	988	13	(hid; remainder)	(9.01, 90.99)	10.10
Glass016vs2	192	9	(ve-win-float-proc; build-win-float-proc, build-win-non float-proc,headlamps)	(8.89, 91.11)	10.29
Glass2	214	9	(Ve-win-float-proc; remainder)	(8.78, 91.22)	10.39
Ecoli0147vs2356	336	7	(cp,im,imU,pp; imS,imL,om,omL)	(8.63, 91.37)	10.59
Led7digit02456789vs1	443	7	(0,2,4,5,6,7,8,9; 1)	(8.35, 91.65)	10.97
Glass06vs5	108	9	(build-win-float-proc,headlamps; tableware)	(8.33, 91.67)	11.00
Ecoli01vs5	240	6	(cp,im; om)	(8.33, 91.67)	11.00
Glass0146vs2	205	9	(build-win-float-proc,containers,headlamps, build-win-non float-proc;ve-win-float-proc)	(8.29, 91.71)	11.06
Ecoli0147vs56	332	6	(cp,im,imU,pp; om,omL)	(7.53, 92.47)	12.28
Cleveland0vs4	177	13	(0; 4)	(7.34, 92.66)	12.62
Ecoli0146vs5	280	6	(cp,im,imU,omL; om)	(7.14, 92.86)	13.00
Ecoli4	336	7	(om; remainder)	(6.74, 93.26)	13.84
Yeast1vs7	459	8	(nuc; vac)	(6.72, 93.28)	13.87
Shuttle0vs4	1829	9	(Rad Flow; Bypass)	(6.72, 93.28)	13.87
Glass4	214	9	(containers; remainder)	(6.07, 93.93)	15.47
Page-blocks13vs2	472	10	(graphic; horiz.line,picture)	(5.93, 94.07)	15.85
Glass016vs5	184	9	(tableware; build-win-float-proc, build-win-non float-proc,headlamps)	(4.89, 95.11)	19.44
Shuttle2vs4	129	9	(Fpv Open; Bypass)	(4.65, 95.35)	20.5
Yeast1458vs7	693	8	(vac; nuc,me2,me3,pox)	(4.33, 95.67)	22.10
Glass5	214	9	(tableware; remainder)	(4.20, 95.80)	22.81
Yeast2vs8	482	8	(pox; cyt)	(4.15, 95.85)	23.10
Yeast4	1484	8	(me2; remainder)	(3.43, 96.57)	28.41
Yeast1289vs7	947	8	(vac; nuc,cyt,pox,erl)	(3.17, 96.83)	30.56
Yeast5	1484	8	(me1; remainder)	(2.96, 97.04)	32.78
Ecoli0137vs26	281	7	(pp,imL; cp,im,imU,imS)	(2.49, 97.51)	39.15
Yeast6	1484	8	(exc; remainder)	(2.49, 97.51)	39.15

The experimental design follows a 5-fold cross validation model (5-cv): 5 random partitions of data, 20% for testing. The error values in this section are the average test results at these 5 partitions.

FURIA_CS depends on a cost matrix (see Table 4). Cost tables are normalized[17] and the cost of misclassifying a positive example is C(+,-)=1/IR while the cost of misclassifying a negative example is C(-,+)=1. A penalization factor PF will be assigned to each correct classification of a negative example[45].

Four classifiers of different types will be considered to benchmark the performance of FURIA_CS: the classical C4.5 method to derive decision trees[41]; Support Vector Machine (SVM) implementation[46]; K-nearest Neighbor (K-NN)[8]; and

Table 4. Cost matrix for a two-class problem

	Positive Class	Negative Class
Positive Prediction	0	1/IR
Negative Prediction	1	PF

Table 5. Choice of parameters for the algorithms considered in the experimentation.

Algoritm Family	Parameters
C4.5	- pruned = True - confidence = 0.25 - minimum number of item-sets per leaf = 2 - C(+,-)= IR, C(-,+)=1, C(+,+)=0, C(-,-)=0
SVM	- kernel type = polynomial - C = 100 - tolerance of termination criterion = 0.001 - degree (for kernel function) = 1 - gamma (for kernel function) = 0.01 - coef()(for kernel funcion) = 0 - use shrinking heuristics = true - C(+,-)= IR, C(-,+)=1, C(+,+)=0, C(-,-)=0
k-NN	- k=3 - distance = Heterogeneous Value Difference Metric (HVDM) - C(+,-)= IR, C(-,+)=1, C(+,+)=0, C(-,-)=0
FH-GBML	- conjunction operator = product t-norm - rule weight = PCF (FH-GBML and FH-GBML+preprocessing) and PCF-SC[32](FH-GBML-CS) - fuzzy reasoning method = winning rule - number of fuzzy rules = 5.d (max. 50 rules) - number of rule sets = 200 - crossover probability = 0.9 - mutation probability = 1/d - number of replaced rules = all rules except the best-one (Pittsburgh-part, elitist approach) and number of rules/5 (GCCL-part) - total number of generations = 1000 - don't care probability = 0.5 - probability of the application of the GCCL iteration = 0.5 - C(+,-)= IR, C(-,+)=1, C(+,+)=0, C(-,-)=0

a state-of-the-art FRBCS learning method, Fuzzy Hybrid Genetic-based Machine Learning (FH-GBML)[28]. Parameters defining these four classifiers are shown in Table 5 and were selected to match those in previous references[32].

5.2. *FURIA for imbalanced data*

This section is devoted to develop a detailed performance study on FURIA_CS. Three different aspects of the proposed extension of FURIA to imbalanced problems are analyzed:

Table 6. AUC-based comparison of FURIA and FURIA_CS with penalization factors $PF_0 = 0$, $PF_1 = 1/(2\,IR)$, $PF_2 = 1 - (1/IR)$.

Datasets	AUC - FURIA	AUC - PF_0	AUC - PF_1	AUC - PF_2
$1.82 \le IR < 9$ (22)	0.837	0.866	0.875	**0.880**
$9 \le IR < 11$ (21)	0.776	0.802	0.823	**0.832**
$11 \le IR \le 39.15$ (21)	0.771	0.795	**0.845**	0.825
Average	0.794	0.821	**0.847**	0.846

Table 7. GM-based comparison of FURIA and FURIA_CS with penalization factors $PF_0 = 0$, $PF_1 = 1/(2\,IR)$, $PF_2 = 1 - (1/IR)$.

Datasets	GM - FURIA	GM - PF_0	GM - PF_1	GM - PF_2
$1.82 \le IR < 9$ (22)	0.850	0.857	0.865	**0.876**
$9 \le IR < 11$ (21)	0.711	0.744	0.773	**0.800**
$11 \le IR \le 39.15$ (21)	0.662	0.696	**0.790**	0.782
Average	0.741	0.765	0.809	**0.819**

(1) The performance of FURIA_CS with respect to different penalization factors in the misclassification costs matrix.
(2) The performance of FURIA_CS (an internal method) against the combination of FURIA and preprocessing (the counterpart external methods): FURIA+SMOTE and FURIA+SMOTE+ENN.
(3) The influence of some design decisions in the performance of FURIA_CS.

5.2.1. *Penalization factors in the cost matrix*

As mentioned, there are cases where it makes sense to add a penalty to correct classifications of instances to the negative class[45]. In this study two different heuristic values will be considered standing for a low and high penalization:

$$PF_1 = \frac{1}{2\,IR} \tag{45}$$

$$PF_2 = 1 - \frac{1}{IR} \tag{46}$$

The summarized of the average results of FURIA, FURIA-CS with penalizations 0, PF_1 and PF_2, relative to the AUC metric (Eq. 3), are shown in Table 6. The same study is shown in Table 7 for the GM metric. In view of the obtained results, non null penalty factors are preferred. The p-value of the Friedman Rank Sum Test is 0.042, showing the relevance of the choice of PF (95% confidence level). FP_2 is preferred in low or medium imbalanced datasets ($1.82 \le IR < 11$). FP_1 performs better for highly imbalanced problems ($11 \le IR \le 39.15$). See Tables 19 and 20 in the Appendix for detailed results.

5.2.2. *FURIA_CS vs. the combination of FURIA and preprocessing*

In Table 8, AUC-based performances of the original FURIA algorithm, FURIA combined with two preprocessing methods (SMOTE and SMOTE+ENN), and FURIA_CS are displayed. FURIA_CS outperforms the other approaches (the statistical relevance of the differences will be analyzed later). Notice this result seems to contradict the conclusions of recent references[32], however in these works a different cost-sensitive algorithm was used and the diagonal of the cost matrix was assumed to be zero. Differences between SMOTE and SMOTE+ENN were not significant.

Table 8: Comparison between FURIA_CS and FURIA with and without preprocessing methods in terms of $\mathrm{Tst}_{\mathrm{AUC}}$

Dataset	FURIA	FURIA+SMOTE	FURIA+SMOTE+ENN	FURIA_CS
	$\mathrm{Tst}_{\mathrm{AUC}}$	$\mathrm{Tst}_{\mathrm{AUC}}$	$\mathrm{Tst}_{\mathrm{AUC}}$	$\mathrm{Tst}_{\mathrm{AUC}}$
Glass1	0.704	0.773	0.762	0.780
Ecoli0vs1	0.986	0.979	0.979	0.986
Wisconsin	0.960	0.963	0.965	0.978
Pima	0.672	0.729	0.745	0.736
Iris0	1.000	1.000	1.000	1.000
Glass0	0.797	0.806	0.845	0.836
Yeast1	0.668	0.719	0.706	0.707
Vehicle1	0.650	0.720	0.766	0.770
Vehicle2	0.969	0.975	0.959	0.977
Vehicle3	0.653	0.749	0.794	0.785
Haberman	0.577	0.639	0.624	0.687
Glass0123vs456	0.868	0.909	0.913	0.903
Vehicle0	0.929	0.952	0.940	0.946
Ecoli1	0.840	0.901	0.872	0.886
New-thyroid2	0.937	0.965	0.960	0.951
New-thyroid1	0.948	0.977	0.986	0.963
Ecoli2	0.856	0.899	0.866	0.917
Segment0	0.987	0.500	0.500	0.992
Glass6	0.841	0.886	0.919	0.908
Yeast3	0.877	0.922	0.913	0.921
Ecoli3	0.769	0.831	0.859	0.855
Page-blocks0	0.929	0.952	0.947	0.942
Ecoli034vs5	0.819	0.901	0.867	0.891
Yeast2vs4v	0.825	0.873	0.847	0.895
Ecoli067vs35	0.877	0.880	0.818	0.872
Ecoli0234vs5	0.838	0.848	0.895	0.880
Glass015vs2	0.526	0.750	0.763	0.615
Yeast0359vs78	0.584	0.697	0.672	0.715
Yeast02579vs368	0.895	0.898	0.887	0.915
Yeast0256vs3789	0.688	0.753	0.789	0.792
Ecoli046vs5	0.816	0.885	0.849	0.889
Ecoli01vs235	0.735	0.821	0.816	0.825
Ecoli0267vs35	0.802	0.847	0.823	0.827
Glass04vs5	0.994	0.979	0.979	0.994
Ecoli0346vs5	0.841	0.932	0.902	0.897
Ecoli0347vs56	0.815	0.899	0.901	0.769
Yeast05679vs4	0.696	0.814	0.780	0.801
Ecoli067vs5	0.840	0.847	0.849	0.865
Vowel0	0.950	0.958	0.956	0.966
Glass016vs2	0.519	0.631	0.736	0.635
Glass2	0.558	0.662	0.702	0.738
Ecoli0147vs2356	0.821	0.866	0.896	0.845
Led7digit02456789vs1	0.881	0.884	0.858	0.908
Glass06vs5	0.945	0.971	0.977	0.945
Ecoli01vs5	0.838	0.798	0.869	0.861
Glass0146vs2	0.497	0.760	0.688	0.740
Ecoli0147vs56	0.796	0.879	0.852	0.865
Cleveland0vs4	0.748	0.500	0.500	0.751
Ecoli0146vs5	0.744	0.874	0.845	0.865

Continued on next page

Table 8 – *Continued from previous page*

Dataset	FURIA	FURIA+SMOTE	FURIA+SMOTE+ENN	FURIA_CS
	Tst_{AUC}	Tst_{AUC}	Tst_{AUC}	Tst_{AUC}
Ecoli4	0.815	0.860	0.829	0.864
Yeast1vs7	0.546	0.671	0.701	0.690
Shuttle0vs4	1.000	1.000	1.000	1.000
Glass4	0.870	0.889	0.881	0.935
Page-blocks13vs2	0.997	0.992	0.992	0.997
Glass016vs5	0.844	0.921	0.812	0.888
Shuttle2vs4	0.950	0.995	1.000	0.950
Yeast1458vs7	0.493	0.521	0.531	0.649
Glass5	0.847	0.942	0.828	0.897
Yeast2vs8	0.773	0.718	0.783	0.773
Yeast4	0.545	0.720	0.770	0.869
Yeast1289vs7	0.566	0.599	0.535	0.771
Yeast5	0.885	0.914	0.965	0.965
Ecoli0137vs26	0.746	0.833	0.831	0.848
Yeast6	0.739	0.805	0.813	0.892
Total Mean	0.796	0.836	0.834	**0.858**

Table 9: Performance ranking of FURIA_CS and FURIA with and without preprocessing methods in terms of Tst_{AUC}

Dataset	1^{st}	2^{nd}	3^{rd}	4^{th}
Glass1	CS	SMOTE	SMOTE+ENN	ORIGINAL
Ecoli0vs1	CS/ORIGINAL	SMOTE/SMOTE+ENN	-	-
Wisconsin	CS	SMOTE+ENN	SMOTE	ORIGINAL
Pima	SMOTE+ENN	CS	SMOTE	ORIGINAL
Iris0	ALL	-	-	-
Glass0	SMOTE+ENN	CS	SMOTE	ORIGINAL
Yeast1	SMOTE	CS	SMOTE+ENN	ORIGINAL
Vehicle1	CS	SMOTE+ENN	SMOTE	ORIGINAL
Vehicle2	CS	SMOTE	ORIGINAL	SMOTE+ENN
Vehicle3	SMOTE+ENN	CS	SMOTE	ORIGINAL
Haberman	CS	SMOTE	SMOTE+ENN	ORIGINAL
Glass0123vs456	SMOTE+ENN	SMOTE	CS	ORIGINAL
Vehicle0	SMOTE	CS	SMOTE+ENN	ORIGINAL
Ecoli1	SMOTE	CS	SMOTE+ENN	ORIGINAL
New-thyroid2	SMOTE	SMOTE+ENN	CS	ORIGINAL
New-thyroid1	SMOTE+ENN	SMOTE	CS	ORIGINAL
Ecoli2	CS	SMOTE	SMOTE+ENN	ORIGINAL
Segment0	CS	ORIGINAL	SMOTE/SMOTE+ENN	-
Glass6	SMOTE+ENN	CS	SMOTE	ORIGINAL
Yeast3	SMOTE	CS	SMOTE+ENN	ORIGINAL
Ecoli3	SMOTE+ENN	CS	SMOTE	ORIGINAL
Page-blocks0	SMOTE	SMOTE+ENN	CS	ORIGINAL
Ecoli034vs5	SMOTE	CS	SMOTE+ENN	ORIGINAL
Yeast2vs4v	CS	SMOTE	SMOTE+ENN	ORIGINAL
Ecoli067vs35	SMOTE	ORIGINAL	CS	SMOTE+ENN
Ecoli0234vs5	SMOTE+ENN	CS	SMOTE	ORIGINAL
Glass015vs2	SMOTE+ENN	SMOTE	CS	ORIGINAL
Yeast0359vs78	CS	SMOTE	SMOTE+ENN	ORIGINAL
Yeast02579vs368	CS	SMOTE	SMOTE+ENN	ORIGINAL
Yeast0256vs3789	CS	SMOTE+ENN	SMOTE	ORIGINAL
Ecoli046vs5	CS	SMOTE	SMOTE+ENN	ORIGINAL
Ecoli01vs235	CS	SMOTE	SMOTE+ENN	ORIGINAL
Ecoli0267vs35	SMOTE	CS	SMOTE+ENN	ORIGINAL
Glass04vs5	CS/ORIGINAL	SMOTE/SMOTE+ENN	-	-
Ecoli0346vs5	SMOTE	SMOTE+ENN	CS	ORIGINAL
Ecoli0347vs56	SMOTE+ENN	SMOTE	ORIGINAL	CS
Yeast05679vs4	SMOTE	CS	SMOTE+ENN	ORIGINAL
Ecoli067vs5	CS	SMOTE+ENN	SMOTE	ORIGINAL
Vowel0	CS	SMOTE	SMOTE+ENN	ORIGINAL
Glass016vs2	SMOTE+ENN	CS	SMOTE	ORIGINAL

Continued on next page

Table 10. Ranking obtained through Friedman's test

Algorithm	Ranking
CS	1.703
SMOTE	2.078
SMOTE_ENN	2.265
ORIGINAL	3.515

Table 9 – *Continued from previous page*

Dataset	1st	2nd	3rd	4th
Glass2	CS	SMOTE+ENN	SMOTE	ORIGINAL
Ecoli0147vs2356	SMOTE+ENN	SMOTE	CS	ORIGINAL
Led7digit02456789vs1	CS	SMOTE	ORIGINAL	SMOTE+ENN
Glass06vs5	SMOTE+ENN	SMOTE	CS/ORIGINAL	-
Ecoli01vs5	SMOTE+ENN	CS	ORIGINAL	SMOTE
Glass0146vs2	SMOTE	CS	SMOTE+ENN	ORIGINAL
Ecoli0147vs56	SMOTE	CS	SMOTE+ENN	ORIGINAL
Cleveland0vs4	CS	ORIGINAL	SMOTE+ENN/SMOTE	-
Ecoli0146vs5	SMOTE	CS	SMOTE+ENN	ORIGINAL
Ecoli4	CS	SMOTE	SMOTE+ENN	ORIGINAL
Yeast1vs7	SMOTE+ENN	CS	SMOTE	ORIGINAL
Shuttle0vs4	ALL	-	-	-
Glass4	CS	SMOTE	SMOTE+ENN	ORIGINAL
Page-blocks13vs2	CS	ORIGINAL	SMOTE+ENN/SMOTE	-
Glass016vs5	SMOTE	CS	ORIGINAL	SMOTE+ENN
Shuttle2vs4	SMOTE+ENN	SMOTE	CS/ORIGINAL	-
Yeast1458vs7	CS	SMOTE+ENN	SMOTE	ORIGINAL
Glass5	SMOTE	CS	ORIGINAL	SMOTE+ENN
Yeast2vs8	SMOTE+ENN	CS	ORIGINAL	SMOTE
Yeast4	CS	SMOTE+ENN	SMOTE	ORIGINAL
Yeast1289vs7	CS	SMOTE	ORIGINAL	SMOTE+ENN
Yeast5	CS/SMOTE+ENN	SMOTE	ORIGINAL	-
Ecoli0137vs26	CS	SMOTE	SMOTE+ENN	ORIGINAL
Yeast6	CS	SMOTE+ENN	SMOTE	ORIGINAL

In Table 9 the ranking of the four algorithms on each dataset is shown. FURIA_CS appears 31 times in the first position, 22 times in the second position, 10 times in the third, and only 1 time in the last position. Those 11 datasets where FURIA_CS was in the third or fourth positions are marked in boldface. Table 10 displays the mean rankings of each algorithm, as part of the Friedman tests used for assessing the statistical significance of the differences[12].

The mean value of the AUC in all problems is shown in Table 8. The dispersion of the results is illustrated with the help of box plots (see Figure 1). The differences are relevant according to the Friedman test. The p-values of the paired comparisons between the best ranked algorithm and the alternatives are in Table 11. A Wilcoxon test has been used to assess these differences.

The conclusions of this part of the study are:

- Equal means hypothesis is rejected in favour of FURIA_CS, FURIA + SMOTE, and FURIA+SMOTE+ENN with respect to FURIA, as expected.
- Equal means hypothesis is rejected in favour of FURIA_CS with respect to FURIA + SMOTE and FURIA+SMOTE+ENN.

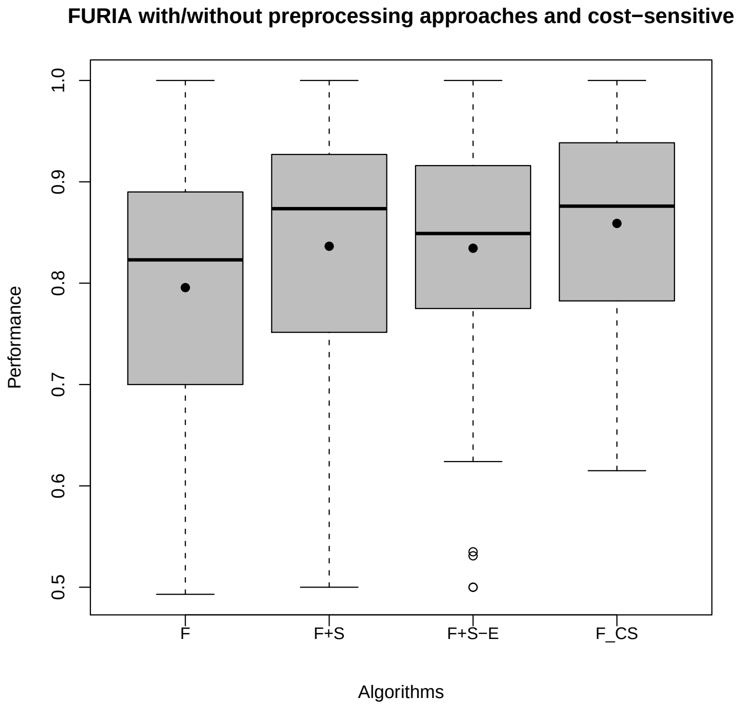

Fig. 1. Dispersion of the performances of FURIA, FURIA+SMOTE, FURIA+SMOTE+ENN and FURIA_CS.

Table 11. Wilcoxon's test for comparing FURIA with and without preprocessing approaches and cost-sensitive learning.

Comparison	p-value	Hypothesis (α= 0.05)
FURIA_CS vs. FURIA	3.1e-10	Reject
FURIA_SMOTE vs. FURIA	5.6e-08	Reject
FURIA_SMOTE+ENN vs. FURIA	7.3e-07	Reject
FURIA_CS vs. FURIA+SMOTE	0.017	Reject
FURIA_CS vs. FURIA+SMOTE+ENN	0.0027	Reject
FURIA+SMOTE+ENN vs. FURIA+SMOTE	0.72	Not reject

- Equal means hypothesis is not rejected in the case of FURIA + SMOTE vs. FURIA+SMOTE+ENN.

5.2.3. *Influence of some design decisions in the performance of FURIA_CS*

According to[15], the splitting criteria in decision trees is not sensitive to costs. If the conclusions of that reference could be applied to fuzzy classification rule learning, the redefinition of Information Gain in Eq. 15 should not be needed, in contradiction with the postulates of the current study.

The experiments in Table 12 were designed for assessing the need of a cost-adapted information gain criterion in FURIA_CS. The results in column "FURIA_CS-IG" were computed after reverting Eq. 15 to its original definition (Eq. 9). Since FURIA_CS-IG is significantly inferior to FURIA_CS and not different than FURIA combined with SMOTE or SMOTE+ENN, Table 12 actually shows that the cost-based definition proposed in Eq. 15 significantly contributes to the performance of the new cost-sensitive algorithm.

Table 12: Comparison between FURIA+preprocessing approaches and FURIA_CS with IG = 9.

Dataset	FURIA_CS-IG	FURIA+SMOTE	FURIA+SMOTE+ENN
	Tst_{AUC}	Tst_{AUC}	Tst_{AUC}
Glass1	0.804	0.773	0.762
Ecoli0vs1	0.986	0.979	0.979
Wisconsin	0.974	0.963	0.965
Pima	0.809	0.729	0.745
Iris0	1.000	1.000	1.000
Glass0	0.825	0.806	0.845
Yeast1	0.736	0.719	0.706
Vehicle1	0.788	0.720	0.766
Vehicle2	0.964	0.975	0.959
Vehicle3	0.800	0.749	0.794
Haberman	0.759	0.639	0.624
Glass0123vs456	0.907	0.909	0.913
Vehicle0	0.927	0.952	0.940
Ecoli1	0.880	0.901	0.872
New-thyroid2	0.951	0.965	0.960
New-thyroid1	0.982	0.977	0.986
Ecoli2	0.865	0.899	0.866
Segment0	0.985	0.5	0.5
Glass6	0.907	0.886	0.919
Yeast3	0.904	0.922	0.913
Ecoli3	0.769	0.831	0.859
Page-blocks0	0.932	0.952	0.947
Ecoli034vs5	0.888	0.901	0.867
Yeast2vs4v	0.918	0.873	0.847
Ecoli067vs35	0.852	0.880	0.818
Ecoli0234vs5	0.914	0.848	0.895
Glass015vs2	0.625	0.750	0.763
Yeast0359vs78	0.640	0.697	0.672
Yeast02579vs368	0.914	0.898	0.887
Yeast0256vs3789	0.778	0.753	0.789
Ecoli046vs5	0.866	0.885	0.849
Ecoli01vs235	0.858	0.821	0.816
Ecoli0267vs35	0.822	0.847	0.823
Glass04vs5	0.994	0.979	0.979
Ecoli0346vs5	0.897	0.932	0.902
Ecoli0347vs56	0.811	0.899	0.901
Yeast05679vs4	0.722	0.814	0.780
Ecoli067vs5	0.842	0.847	0.849
Vowel0	0.953	0.958	0.956
Glass016vs2	0.583	0.631	0.736
Glass2	0.650	0.662	0.702
Ecoli0147vs2356	0.863	0.866	0.896

Continued on next page

Table 12 – *Continued from previous page*

Dataset	FURIA_CS-IG	FURIA+SMOTE	FURIA+SMOTE+ENN
	Tst_{AUC}	Tst_{AUC}	Tst_{AUC}
Led7digit02456789vs1	0.908	0.884	0.858
Glass06vs5	0.945	0.971	0.977
Ecoli01vs5	0.888	0.798	0.869
Glass0146vs2	0.641	0.760	0.688
Ecoli0147vs56	0.856	0.879	0.852
Cleveland0vs4	0.500	0.500	0.50
Ecoli0146vs5	0.865	0.874	0.845
Ecoli4	0.846	0.860	0.829
Yeast1vs7	0.579	0.671	0.701
Shuttle0vs4	1.000	1.000	1.000
Glass4	0.875	0.889	0.881
Page-blocks13vs2	0.978	0.992	0.992
Glass016vs5	0.844	0.921	0.812
Shuttle2vs4	0.950	0.995	1.000
Yeast1458vs7	0.509	0.521	0.531
Glass5	0.897	0.942	0.828
Yeast2vs8	0.773	0.718	0.783
Yeast4	0.655	0.720	0.770
Yeast1289vs7	0.566	0.599	0.535
Yeast5	0.919	0.914	0.965
Ecoli0137vs26	0.748	0.833	0.831
Yeast6	0.710	0.805	0.813
Total Mean	0.837	0.836	0.834

In another work related to decision tree learning[6] it was concluded that the pruning criteria are not relevant when designing a cost-sensitive algorithm. As done before, the algorithm FURIA_CS_SP is built by removing the pruning stage from FURIA_CS (Eqs. 21 and 22). In Table 13 the results of FURIA_CS and FURIA_CS_SP are compared, showing a small advantage in favor of FURIA_CS. Nevertheless, the difference found is actually less relevant than that found for the Information Gain study.

Table 13: Comparation between FURIA_CS and FURIA_CS-SP, without pruning stage.

Dataset	FURIA-CS	FURIA-CS-SP
	Tst_{AUC}	Tst_{AUCM}
Glass1	0.780	0.757
Ecoli0vs1	0.986	0.986
Wisconsin	0.978	0.978
Pima	0.736	0.733
Iris0	1.000	1.000
Glass0	0.836	0.808
Yeast1	0.707	0.766
Vehicle1	0.770	0.763
Vehicle2	0.977	0.977
Vehicle3	0.785	0.781
Haberman	0.687	0.687
Glass0123vs456	0.903	0.903
Vehicle0	0.946	0.944
Ecoli1	0.886	0.883
New-thyroid2	0.951	0.951
New-thyroid1	0.963	0.963
Ecoli2	0.917	0.917
Segment0	0.992	0.992
Glass6	0.908	0.908
Yeast3	0.921	0.917
Ecoli3	0.855	0.851
Page-blocks0	0.942	0.934

Continued on next page

Table 13 – *Continued from previous page*

Dataset	FURIA-CS	FURIA-CS-SP
	Tst_{AUC}	Tst_{AUCM}
Ecoli034vs5	0.891	0.891
Yeast2vs4v	0.895	0.885
Ecoli067vs35	0.872	0.875
Ecoli0234vs5	0.880	0.880
Glass015vs2	0.615	0.605
Yeast0359vs78	0.715	0.715
Yeast02579vs368	0.915	0.915
Yeast0256vs3789	0.792	0.792
Ecoli046vs5	0.889	0.864
Ecoli01vs235	0.825	0.807
Ecoli0267vs35	0.827	0.825
Glass04vs5	0.994	0.994
Ecoli0346vs5	0.897	0.897
Ecoli0347vs56	0.769	0.749
Yeast05679vs4	0.801	0.813
Ecoli067vs5	0.865	0.865
Vowel0	0.966	0.966
Glass016vs2	0.635	0.593
Glass2	0.738	0.738
Ecoli0147vs2356	0.845	0.845
Led7digit02456789vs1	0.908	0.908
Glass06vs5	0.945	0.945
Ecoli01vs5	0.861	0.838
Glass0146vs2	0.740	0.740
Ecoli0147vs56	0.865	0.833
Cleveland0vs4	0.751	0.751
Ecoli0146vs5	0.865	0.865
Ecoli4	0.864	0.864
Yeast1vs7	0.690	0.590
Shuttle0vs4	1.000	1.000
Glass4	0.935	0.904
Page-blocks13vs2	0.997	0.997
Glass016vs5	0.888	0.888
Shuttle2vs4	0.950	0.950
Yeast1458vs7	0.649	0.625
Glass5	0.897	0.897
Yeast2vs8	0.773	0.773
Yeast4	0.869	0.864
Yeast1289vs7	0.771	0.752
Yeast5	0.965	0.965
Ecoli0137vs26	0.848	0.848
Yeast6	0.892	0.826
Total Mean	0.858	0.852

5.3. *Comparison between FURIA_CS and other classification algorithms*

The compared results of FURIA_CS, C4.5[41], SVM[46], k-NN[8], and FH-GBML rule generation algorithm[28] are shown in Table 14. The cost-sensitive version of each technique was used, along with SMOTE and SMOTE+ENN preprocessed datasets combined with error-based versions of the algorithms[32].

Table 14: FURIA-CS against C4.5, SVM, k-NN and FH-GBML with preprocessing approaches and cost-sensitive learning.

	FURIA	C45			SVM			k-NN			FH-GBML		
Dataset	CS	SMO	SMO+EN	CS	SMO	SMO+EN	CS	SMO	SMO+EN	CS	SMO	SMO+EN	CS
Glass1	0.780	0.736	0.692	0.716	0.617	0.639	0.626	0.780	0.776	0.746	0.731	0.733	0.741
Ecoli0vs1	0.986	0.972	0.983	0.983	0.979	0.977	0.967	0.500	0.500	0.500	0.962	0.953	0.976

Continued on next page

Table 14 – *Continued from previous page*

	FURIA	C45			SVM			FH-GBML			k-NN		
Dataset	CS	SMO	SMO+EN	CS	SMO	SMO+EN	CS	SMO	SMO+EN	CS	SMO	SMO+EN	CS
Wisconsin	0.978	0.953	0.957	0.963	0.972	0.969	0.971	0.969	0.972	0.965	0.963	0.972	0.978
Pima	0.736	0.724	0.740	0.712	0.735	0.730	0.728	0.686	0.709	0.670	0.738	0.706	0.727
Iris0	1.000	0.990	0.990	0.990	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Glass0	0.836	0.775	0.799	0.821	0.737	0.724	0.507	0.818	0.836	0.777	0.754	0.790	0.770
Yeast1	0.707	0.709	0.695	0.677	0.710	0.706	0.674	0.677	0.707	0.685	0.700	0.704	0.701
Vehicle1	0.770	0.730	0.754	0.701	0.774	0.805	0.754	0.698	0.775	0.746	0.710	0.704	0.698
Vehicle2	0.977	0.949	0.941	0.943	0.960	0.957	0.657	0.969	0.962	0.954	0.871	0.869	0.873
Vehicle3	0.785	0.728	0.740	0.728	0.761	0.788	0.790	0.708	0.763	0.735	0.712	0.727	0.694
Haberman	0.687	0.616	0.588	0.575	0.634	0.633	0.538	0.563	0.576	0.651	0.613	0.606	0.606
Glass0123vs456	0.903	0.923	0.924	0.877	0.905	0.898	0.844	0.916	0.933	0.933	0.930	0.943	0.915
Vehicle0	0.946	0.918	0.907	0.928	0.963	0.961	0.949	0.947	0.941	0.946	0.893	0.869	0.887
Ecoli1	0.886	0.910	0.892	0.911	0.906	0.902	0.906	0.808	0.808	0.803	0.876	0.870	0.865
New-thyroid2	0.951	0.965	0.977	0.980	0.991	0.988	0.982	0.988	0.986	0.991	0.980	0.977	0.951
New-thyroid1	0.963	0.963	0.988	0.974	0.994	0.986	0.968	0.988	0.986	0.991	0.951	0.991	0.965
Ecoli2	0.917	0.881	0.897	0.890	0.906	0.905	0.500	0.838	0.827	0.827	0.886	0.936	0.897
Segment0	0.992	0.992	0.991	0.991	0.995	0.996	0.996	0.500	0.500	0.500	0.977	0.974	0.980
Glass6	0.908	0.884	0.920	0.889	0.906	0.900	0.872	0.941	0.933	0.941	0.882	0.829	0.838
Yeast3	0.921	0.890	0.923	0.911	0.891	0.906	0.895	0.868	0.863	0.877	0.929	0.916	0.907
Ecoli3	0.855	0.812	0.870	0.832	0.898	0.881	0.792	0.728	0.777	0.750	0.884	0.878	0.886
Page-blocks0	0.942	0.950	0.942	0.945	0.925	0.927	0.925	0.932	0.931	0.937	0.893	0.893	0.894
Mean	0.883	0.862	0.869	0.861	0.871	0.872	0.811	0.810	0.821	0.815	0.856	0.856	0.852
Ecoli034vs5	0.891	0.900	0.880	0.925	0.888	0.886	0.863	0.822	0.822	0.836	0.894	0.844	0.912
Yeast2vs4v	0.895	0.858	0.904	0.886	0.889	0.888	0.500	0.807	0.807	0.793	0.907	0.897	0.893
Ecoli067vs35	0.872	0.850	0.812	0.882	0.832	0.835	0.802	0.820	0.815	0.855	0.812	0.875	0.818
Ecoli0234vs5	0.880	0.897	0.894	0.833	0.889	0.889	0.841	0.853	0.853	0.861	0.857	0.843	0.805
Glass015vs2	0.615	0.677	0.795	0.600	0.509	0.519	0.500	0.675	0.693	0.709	0.600	0.720	0.648
Yeast0359vs78	0.715	0.704	0.702	0.676	0.745	0.745	0.500	0.724	0.720	0.692	0.722	0.735	0.757
Yeast02579vs368	0.915	0.914	0.913	0.899	0.901	0.906	0.500	0.902	0.901	0.898	0.909	0.893	0.900
Yeast0256vs3789	0.792	0.795	0.781	0.784	0.794	0.801	0.500	0.772	0.765	0.791	0.785	0.794	0.794
Ecoli046vs5	0.889	0.870	0.886	0.831	0.886	0.886	0.869	0.928	0.928	0.936	0.832	0.806	0.966
Ecoli01vs235	0.825	0.837	0.833	0.764	0.850	0.855	0.780	0.793	0.793	0.785	0.807	0.848	0.795
Ecoli0267vs35	0.827	0.815	0.817	0.852	0.825	0.853	0.785	0.840	0.832	0.802	0.833	0.799	0.831
Glass04vs5	0.994	0.981	0.975	0.994	0.956	0.950	0.900	0.963	0.951	0.994	0.967	0.857	0.919
Ecoli0346vs5	0.897	0.898	0.898	0.850	0.892	0.892	0.894	0.916	0.816	0.841	0.833	0.914	0.891
Ecoli0347vs56	0.769	0.856	0.854	0.758	0.908	0.906	0.813	0.792	0.500	0.836	0.860	0.852	0.832
Yeast05679vs4	0.801	0.760	0.780	0.724	0.807	0.787	0.500	0.744	0.768	0.796	0.806	0.731	0.770
Ecoli067vs5	0.865	0.847	0.845	0.882	0.847	0.807	0.745	0.837	0.825	0.867	0.833	0.875	0.861
Vowel0	0.966	0.950	0.945	0.942	0.962	0.962	0.846	0.999	0.999	0.999	0.956	0.913	0.939
Glass016vs2	0.635	0.606	0.638	0.615	0.533	0.526	0.500	0.716	0.644	0.789	0.634	0.689	0.663
Glass2	0.738	0.639	0.745	0.641	0.615	0.690	0.595	0.716	0.771	0.695	0.677	0.599	0.709
Ecoli0147vs2356	0.845	0.827	0.822	0.877	0.882	0.872	0.726	0.759	0.795	0.827	0.850	0.845	0.862
Led7digit02456789vs1	0.908	0.890	0.837	0.843	0.885	0.889	0.500	0.821	0.845	0.829	0.883	0.890	0.874
Mean	0.835	0.827	0.836	0.812	0.824	0.826	0.689	0.819	0.802	0.830	0.822	0.820	0.830
Glass06vs5	0.945	0.914	0.964	0.995	0.943	0.943	0.650	0.984	0.984	1.000	0.932	0.892	0.910
Ecoli01vs5	0.861	0.797	0.825	0.818	0.836	0.836	0.790	0.902	0.902	0.913	0.898	0.886	0.843
Glass0146vs2	0.740	0.784	0.709	0.679	0.612	0.631	0.500	0.701	0.701	0.756	0.706	0.634	0.761
Ecoli0147vs56	0.865	0.859	0.842	0.853	0.861	0.854	0.796	0.913	0.902	0.918	0.804	0.860	0.895
Cleveland0vs4	0.751	0.790	0.760	0.689	0.878	0.914	0.748	0.834	0.834	0.858	0.752	0.705	0.686
Ecoli0146vs5	0.865	0.898	0.898	0.838	0.890	0.880	0.792	0.901	0.900	0.913	0.920	0.875	0.852
Ecoli4	0.864	0.779	0.904	0.863	0.920	0.920	0.952	0.842	0.810	0.818	0.930	0.929	0.942
Yeast1vs7	0.609	0.700	0.737	0.613	0.786	0.774	0.500	0.739	0.699	0.745	0.719	0.642	0.738
Shuttle0vs4	1.000	0.999	0.999	0.999	0.996	1.000	1.000	0.996	0.996	0.996	0.998	1.000	0.992
Glass4	0.935	0.886	0.865	0.843	0.957	0.910	0.912	0.891	0.915	0.886	0.886	0.961	0.874
Page-blocks13vs2	0.997	0.995	0.991	0.978	0.956	0.964	0.856	0.997	0.998	0.997	0.951	0.945	0.974
Glass016vs5	0.888	0.812	0.862	0.988	0.942	0.945	0.500	0.927	0.918	0.985	0.899	0.892	0.819
Shuttle2vs4	0.950	0.991	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.950	0.994	0.987	1.000
Yeast1458vs7	0.649	0.536	0.556	0.554	0.637	0.626	0.500	0.694	0.692	0.660	0.628	0.659	0.631
Glass5	0.897	0.880	0.775	0.942	0.951	0.941	0.973	0.937	0.973	0.932	0.767	0.797	0.884
Yeast2vs8	0.773	0.833	0.819	0.865	0.766	0.764	0.766	0.720	0.737	0.801	0.744	0.722	0.741
Yeast4	0.869	0.712	0.725	0.722	0.824	0.828	0.815	0.744	0.757	0.748	0.813	0.794	0.822
Yeast1289vs7	0.771	0.683	0.633	0.676	0.719	0.707	0.500	0.658	0.676	0.646	0.723	0.717	0.639
Yeast5	0.965	0.933	0.940	0.933	0.965	0.962	0.965	0.950	0.956	0.942	0.946	0.977	0.974
Ecoli0137vs26	0.848	0.813	0.813	0.828	0.799	0.804	0.850	0.769	0.500	0.780	0.823	0.820	0.789
Yeast6	0.892	0.829	0.827	0.808	0.873	0.869	0.875	0.844	0.854	0.836	0.864	0.859	0.842

Continued on next page

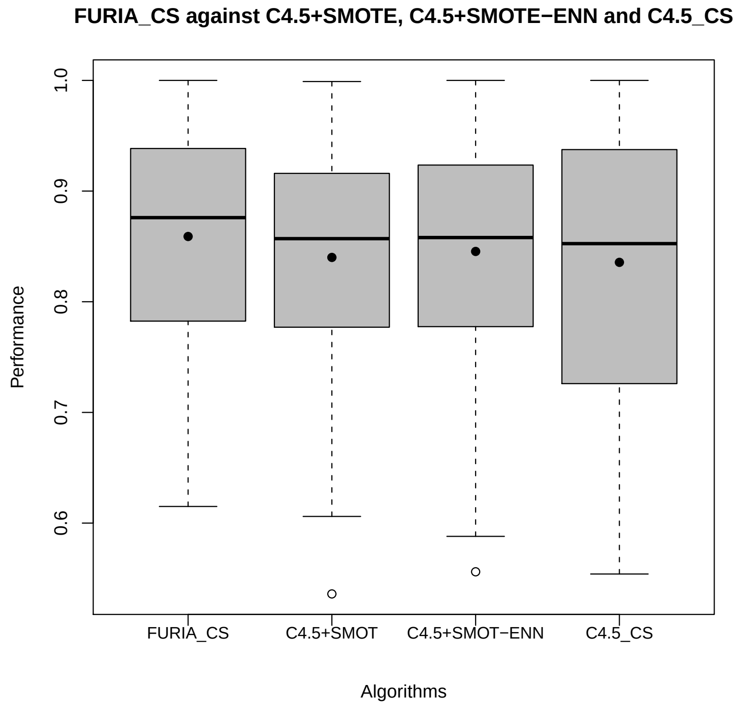

Fig. 2. Performance of FURIA_CS and C4.5 with preprocessing approaches and cost-sensitive learning.

Table 14 – *Continued from previous page*

	FURIA	C45			SVM			FH-GBML			k-NN		
Dataset	CS	SMO	SMO+EN	CS	SMO	SMO+EN	CS	SMO	SMO+EN	CS	SMO	SMO+EN	CS
Mean	0.854	0.830	0.831	0.833	0.862	0.861	0.773	0.854	0.843	0.861	0.843	0.836	0.838
Total Mean	**0.858**	0.840	0.845	0.835	0.853	0.853	0.758	0.840	0.838	0.841	0.828	0.822	0.835

The main conclusions that can be drawn from the preceding table are:

- **FURIA_CS with respect to C45 + SMOTE, C45 + SMOTE+ENN, and C45_CS:** The performance of FURIA_CS is better than that of C4.5 with both preprocessing methods, and also better than cost-based C4.5. The dispersion of the results is shown in Figure 2. The p-values of the paired comparisons (Wilcoxon test, see Table 15) indicate that the mean performance of FURIA_CS is significantly better than the alternatives.
- **FURIA_CS with respect to SVM + SMOTE, SVM + SMOTE+ENN, and SVM_CS:** The combination of preprocessing techniques and the error-based versions of SVM improves the results

Table 15. Wilcoxon's test to compare FURIA_CS against C4.5 with preprocessing approaches and cost-sensitive learning.

Comparison	p-value	Hypothesis (α= 0.05)
FURIA_CS vs. C4.5+SMOTE	0.00044	Reject
FURIA_CS vs. C4.5+SMOTE+ENN	0.013	Reject
FURIA_CS vs. C4.5_CS	0.00012	Reject

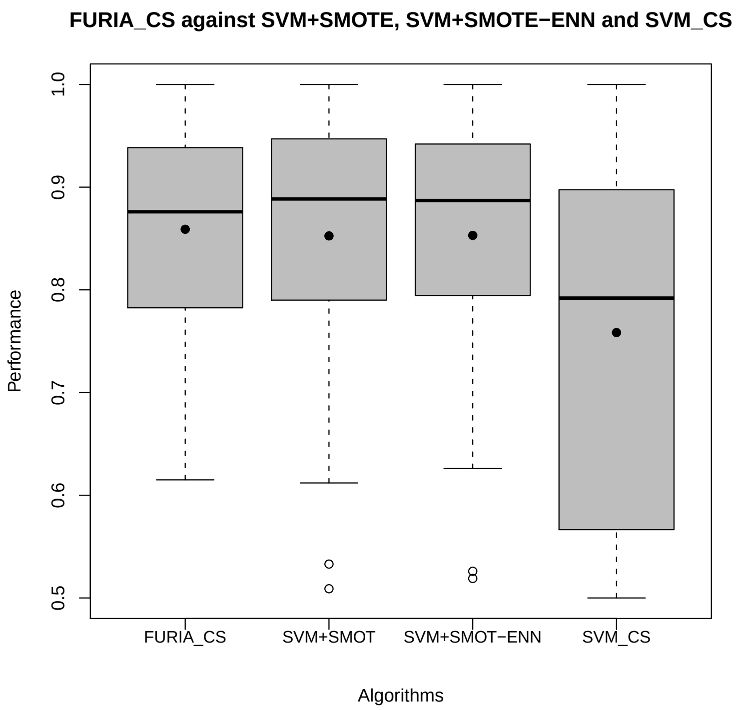

Fig. 3. Performance of FURIA_CS and SVM with preprocessing approaches and cost-sensitive learning.

of cost-based SVM (see Figure 3). Differences between performances of SVM+SMOTE, SVM+SMOTE+ENN and FURIA_CS are not statistically significant, as also shown in Figure 3. In Table 16 the p-values of the paired tests of the best ranked algorithm are shown: hypotheses of equal performance are not rejected for SVM+SMOTE and SVM+SMOTE+ENN, while they are rejected for SVM_CS.

- **FURIA_CS with respect to k-NN + SMOTE, k-NN + SMOTE+ENN, and k-NN_CS:** FURIA_CS is significantly better than

Table 16. Wilcoxon's test to compare FURIA_CS against SVM with preprocessing approaches and cost-sensitive learning.

Comparison	p-value	Hypothesis (α= 0.05)
FURIA_CS vs. SVM+SMOTE	0.26	Not reject
FURIA_CS vs. SVM+SMOTE+ENN	0.20	Not reject
FURIA_CS vs. SVM_CS	≈ 0	Reject

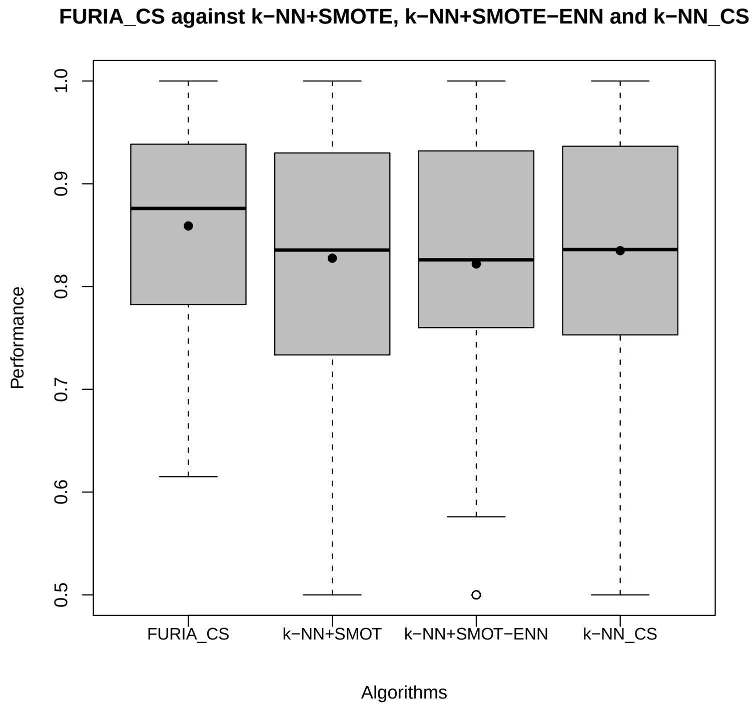

Fig. 4. Performance of FURIA_CS and k-NN with preprocessing approaches and cost-sensitive learning.

k-NN with preprocessing, but the advantage over k-NN_CS is not significant at 95% confidence level (it would be significant at 92.5% level). See Figure 4 and Table 17 for these results.

- **FURIA_CS with respect to FH-GBML + SMOTE, FH-GBML + SMOTE+ENN, and FH-GBML_CS:** The performance of FURIA_CS is better than that of FH-GBML both with preprocessing and cost-sensitive learning, as shown in Figure 5 and Table 18.

Table 17. Wilcoxon's test to compare FURIA_CS against k-NN with preprocessing approaches and cost-sensitive learning.

Comparison	p-value	Hypothesis (α= 0.05)
FURIA_CS vs. k-NN+SMOTE	0.015	Reject
FURIA_CS vs. k-NN+SMOTE+ENN	0.011	Reject
FURIA_CS vs. k-NN_CS	0.073	Not reject

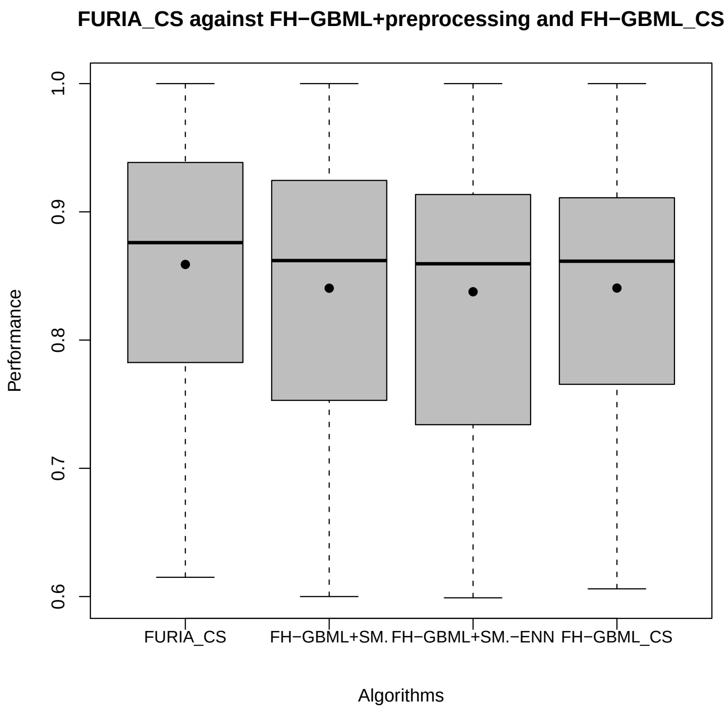

Fig. 5. Performance of FURIA_CS and FH-GBML with preprocessing approaches and cost-sensitive learning.

6. Concluding remarks

According to recent literature[32], external approaches are expected to perform better than cost-sensitive learning algorithms when tackling imbalanced classification problems. This paper was mainly intended to show that this conclusion must be nuanced. On the one hand, the choice of the learning algorithm is important. In this study, FURIA was chosen because it is very competitive with other fuzzy rule learning algorithms in terms of accuracy. Among other reasons, the accuracy of FURIA is not restricted by the choice of a linguistic partition, and the antecedents of

Table 18. Wilcoxon's test to compare FURIA_CS against FH-GBML with preprocessing approaches and cost-sensitive learning.

Comparison	p-value	Hypothesis (α= 0.05)
FURIA_CS vs. FH-GBML+SMOTE	0.00028	Reject
FURIA_CS vs. FH-GBML+SMOTE+ENN	0.0016	Reject
FURIA_CS vs. FH-GBML_CS	0.0014	Reject

rules dynamically change when an uncovered query appears. These properties allow for a better accuracy than that of static fuzzy linguistic knowledge bases. However, it is not discarded that an improved balance between understandability and accuracy can be achieved with future cost-sensitive generalizations of other fuzzy rule learning algorithms.

On the other hand, the outcome of a cost-sensitive learning algorithm is strongly influenced by the choice of cost matrix. It is intuitive to use a risk proportional to the imbalance ratio for quantifying errors in the minority class. It is also intuitive that correct classifications have a null risk. Unfortunately, the combination of both is not different than making a uniform reweigh of the minority instances. In other words, this cost matrix is equivalent to a crude resampling that is easily improved by state-of-the-art algorithms like SMOTE. As a consequence of this, comparisons between external and internal approaches for solving imbalanced problems should not only be supported by this last cost structure. Best results may be obtained with counter-intuitive assignments. In this study, it has been shown that adding a small penalty to correct classifications of the majority class noticeably improves the results for both AUC and GM metrics.

7. Acknowledgements

This work has been funded by Spanish Ministry of Economy and Competitiveness (Grant TIN2011-24302) and by Spanish Ministry of Education and Science (Grant TIN2009-07727), including EDRF fundings.

1. Abe N., Zadrozny B., Langford J. An iterative method for multi-class cost-sensitive learning. Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 3-11 (2004).
2. Alcalá-Fdez J., Fernández A., Luengo J., Derrac J., García S., Sánchez L., Herrera F. KEEL Data-Mining Software Tool: Data set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17(2-3), 255-287 (2011).
3. Batista G., Prati R., Monard M. A study of the behaviour of several methods for balancing machine learning training data. SIGKDD Explorations 6(1), 20-29 (2004).
4. Berger J. Statistical decision theory and Bayesian Analysis. Springer-Verlag (1985).
5. Chawla N.V., Bowywe K.W., Hall L.O., Kegelmeyer W.P. SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligent Research 16, 321-357 (2002).
6. Chawla N.V. C4.5 and imbalanced data sets: investigating the effect of sampling

method, probabilistic estimate, and decision tree structure. Proceedings of the ICML Workshop on Class Imbalances (2003).
7. Cohen W. Fast effective rule induction. In Prieditis A., Russel S. (eds), Proceeding of the 12th International Conference on Machine Learning (ICML), 115-123 (1995).
8. Cover T., Hart P. Nearest neighbor pattern classification. IEEE Transaction on Information Theory 13, 21-27 (1967).
9. Crockett K., Bandar Z., O'shea J. On producing balanced fuzzy decision tree classifiers. Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZIEEE), 1756-1762 (2006).
10. Bradley A.P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7), 1145-1159 (1997).
11. Chawla N.V., Japkowicz N., Kolcz A. Editorial: Special issue on learning from imbalanced data sets. SIGKDD Explorations 6(1), 1-6 (2004).
12. Demsar J. Statistical comparisons of classifiers over multiple datasets. Journal of Machine Learning Research 7, 1-30 (2006).
13. Dmochowski J.P., Sajda P., Parra L.C. Maximum likelihood in cost-sensitive learning: Model specification, approximators, and upper bounds. Journal of Machine Learning Research 11, 3313-3332 (2010).
14. Domingos P. Metacost: A general method for making classifiers cost-sensitive. In Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining (KDD), 155-164 (1999).
15. Drummond C., Holte R.C. Exploiting the cost (in)sensitivity of decision tree splitting criteria. Proceedings of the 17th International Conference on Machine Learning (ICML) 239-146 (2000).
16. Ducange P., Lazzerini B., Marcelloni F. Multi-objetive genetic fuzzy classifiers for imbalanced and cost-sensitive datasets. Soft Computing 14(7), 713-728 (2010).
17. Elkan C. The foundations of cost-sensitive learning. Proceedings of the 17th IEEE International Joint Conference on Artificial Intelligence (IJCAI), 973-978 (2001).
18. Fernández A., García S., del Jesus M.J., Herrera F. A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced datasets. Fuzzy Sets and Systems 159, 2378-2398 (2008).
19. Fazzolari, M., Alcala, R., Nojima, Y., Ishibuchi, H. and Herrera, F. A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions, IEEE Trans. on Fuzzy Systems, vol. 21, no. 1, pp. 45-65, February 2013.
20. Fernández A., del Jesus M.J., Herrera F. Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets. International Journal of Approximate Reasoning 50, 561-577 (2009).
21. Fernández A., del Jesus M.J., Herrera F. On the influence of an adaptative inference system in fuzzy rule based classification systems for imbalanced data-sets. Expert Systems with Applications 36, 9805-9812 (2009).
22. Fernández A, del Jesus M.J., Herrera F. On the 2-tuples based genetic tuning performance for fuzzy rule based classification systems in imbalanced data-sets. Information Sciences 180, 1268-1291 (2010).
23. Huang J., Ling C. Using auc and accuracy in evaluating learning algorithms - appendices. IEEE Transactions on Knowledge and Data Engineering 17, 299-310 (2005).
24. Huang Y.M., Hung C.M., Jiau H. Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem. Nonlinear Analysis: Real World Applications 7(4), 720-747 (2006).
25. Hühn J.C., Hüllermeier E. FURIA: an algorithm for unordered fuzzy rule induction. Data Mining and Knowledge Discovery 19, 293-319 (2009).

26. Hühn J.C., Hüllermeier E. FURIA: Fuzzy Unordered Rule Induction Algorithm. URL: `http://www.uni-marburg.de/fb12/kebi/research /software/furia` (2009).
27. Hühn J.C., Hüllermeier E. An analysis of the FURIA algorithm for fuzzy rule induction. In Advances in Machine Learning I, 32-344 (2010).
28. Ishibuchi H., Yamamoto T., Nakashima T. Hybridization of fuzzy GBML approaches for pattern classification problems. IEEE Transactions on System, Man and Cybernetics, PartB 35, 359-365 (2005).
29. Japkowicz N., Stephen S. The class imbalance problem: a systematic study. Intelligent Data Analysis 6(5), 429-450 (2002).
30. Kubal M., Holte R., Matwin S. Learning when negative examples abound. Proceeding of the European Conference on Machine Learning (ECML), 146-153 (1997).
31. Kubat M., Matwin S. Addressing the curse of imbalanced training sets: one-sided selection. Proceeding of the International Conference on Machine Learning (ICML), 170-186 (1997).
32. Lopez V., Fernandez A., Moreno-Torres J.G., Herrera F. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics. Expert Systems with Applications 39, 6585-6608 (2012).
33. Margineantu D. Methods for cost-sensitive learning. Technical report, Department of Computer Science, Oregon State University, Corvallis, OR, USA (2001).
34. Margineantu D. Class probability estimation and cost-sensitive classification decisions. Proceedings of the 13th European Conference on Machine Learning (ECML), 270-281 (2002).
35. Mazurowski M., Habas P., Zurada J., Lo J., Baker J., Tourassi G. Training neural network classifiers for medical decision making: The effect of imbalanced datasets on classification performance. Neural Networks 21(2-3), 417-436 (2008).
36. Palacios A., Sánchez L., Couso I. Linguistic cost-sensitive learning of genetic fuzzy classifiers for imprecise data. International Journal of Approximate Reasoning 52, 841-862 (2011).
37. Palacios A., Sánchez L., Couso I. Equalizing imbalanced imprecise datasets for genetic fuzzy classifiers. International Journal of Computational Intelligence Systems: Special Issue on Evolutionary Fuzzy Systems 5(2), 276-296 (2012).
38. Peng X., King I. Robust BMPM training based on second-order cone programming and its application in medical diagnosis. Neural Networks 21(2-3), 450-457 (2008).
39. Phua C., Alahakoon D., Lee V., Minority report in fraud detection: classification of skewed data. SIGDKK Explorations Newsletter 6(1), 50-59 (2004).
40. Provost F., Fawcett T. Robust classification systems for imprecise enviroments. Proceeding of the AAAI, 706-713 (1998).
41. Quinlan J. C4.5: Programs for Machine Learning. Morgan Kaufmann (1993).
42. Soler V., Cerquides J., Sabria J., Roig J., Prim M. Imbalanced datasets classification by fuzzy rule extraction and genetic algorithms. Proceeding of the IEEE International Conference on Data MiningWorkshop, 330-336 (2006).
43. Sun Y., Kamel M., Wong A.K.C., Wang Y. Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition 40, 3358-3378 (2007).
44. Sun Y., Wong A.K.C., Kemel M. Classification of imbalanced data: A review. International Journal of Pattern Recognition and Artificial Intelligence 23(4), 687-719 (2009).
45. Turney P. Types of cost in inductive concept learning. Proceeding of the Workshop on Cost-Sensitive Learning at the 17th International Conference on Machine Learning (WCSL at ICML), 15-21 (2000).
46. Vapnik V. Statistical Learning Theory. New York, USA: Wiley (1998).

47. Visa S., Ralescu A. Learning imbalanced and overlapping classes using fuzzy sets. Proceeding of the International Conference Machine Learning -Workshop on Learning from Imbalanced Datasets II (2003).
48. Visa S., Ralescu A. The effect of imbalanced data class distribution on fuzzy classifiers-experimental study. Proceeding of the IEEE International Conference on Fuzzy Systems, 749-754 (2005).
49. Weiss G. Mining with rarity: a unifying framework. SIGKDD Explorations 6(1), 7-19 (2004).
50. Xia F., Yan Y., Zhou L., Li F., Cai M., Zeng D. A closed-form reduction of multi-class cost-sensitive learning to weighted multi-class learning. Pattern Recognition 42, 1572-1581 (2009).
51. Xu L., Chow M., Taylor L. Power distribution fault cause identification with imbalanced data using the data mining-based fuzzy classification e-algorithm. IEEE Transactions on Power Systems 22(1), 164-171 (2007).
52. Wilson D.L. Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man and Cybernetics 2, 408-421 (1972).
53. Zadrozny B., Elkan C., Learning and making decisions when costs and probabilities are both unknown. In Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining (KDD), 204-213 (2001).
54. Zadrozny B. One-benefit learning: cost-sensitive learning with restricted cost information. Proceedings of the 1st International Workshop on Utility-based Data Mining, 53-58 (2005).
55. Zhou Z.H., Liu X.Y. On multi-class cost-sensitive learning. Proceedings of the 21st National Conference on Artificial Intelligence, 567-572 (2006).
56. Zhou Z.H., Liu X.Y. Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Transactions on Knowledge and Data Engineering 18(1), 63-77 (2006).

Appendix A. FURIA_CS and penalizations

The detailed results of the use of different penalizations for correctly classifying negative examples (from the experiment developed in Section 5.2.1) are collected below.

Table 19: AUC-based comparison between penalization factors $\mathrm{PF}_1 = 1/(2\,\mathrm{IR})$, $\mathrm{PF}_2 = 1 - (1/\mathrm{IR})$ and without penalization factor.

Dataset	FURIA	FURIA-CS (FP2)	FURIA-CS (FP1)	FURIA-CS
	$\mathrm{Tst}_{\mathrm{AUC}}$	$\mathrm{Tst}_{\mathrm{AUC}}$	$\mathrm{Tst}_{\mathrm{AUC}}$	$\mathrm{Tst}_{\mathrm{AUC}}$
Glass1	0.704	0.780	0.771	0.745
Ecoli0vs1	0.986	0.986	0.986	0.986
Wisconsin	0.960	0.972	0.978	0.974
Pima	0.672	0.733	0.736	0.753
Iris0	1.000	1.000	1.000	1.000
Glass0	0.797	0.836	0.836	0.819
Yeast1	0.668	0.706	0.707	0.700
Vehicle1	0.650	0.770	0.770	0.749
Vehicle2	0.969	0.977	0.977	0.975
Vehicle3	0.653	0.785	0.785	0.722
Haberman	0.577	0.687	0.674	0.668
Glass0123vs456	0.868	0.903	0.893	0.873
Vehicle0	0.929	0.946	0.938	0.934
Ecoli1	0.840	0.886	0.872	0.868
New-thyroid2	0.937	0.951	0.951	0.951
New-thyroid1	0.948	0.963	0.963	0.968

Continued on next page

Table 19 – *Continued from previous page*

Dataset	FURIA	FURIA-CS (FP2)	FURIA-CS (FP1)	FURIA-CS
	Tst_{AUC}	Tst_{AUC}	Tst_{AUC}	Tst_{AUC}
Ecoli2	0.856	0.917	0.866	0.874
Segment0	0.987	0.992	0.988	0.988
Glass6	0.841	0.908	0.908	0.891
Yeast3	0.877	0.921	0.919	0.889
Ecoli3	0.769	0.855	0.794	0.781
Page-blocks0	0.929	0.942	0.942	0.935
Mean	0.837	**0.880**	0.875	0.866
Ecoli034vs5	0.819	0.891	0.883	0.866
Yeast2vs4v	0.825	0.895	0.884	0.859
Ecoli067vs35	0.877	0.872	0.872	0.850
Ecoli0234vs5	0.838	0.880	0.866	0.866
Glass015vs2	0.526	0.615	0.615	0.618
Yeast0359vs78	0.584	0.715	0.629	0.662
Yeast02579vs368	0.895	0.915	0.911	0.884
Yeast0256vs3789	0.688	0.792	0.774	0.713
Ecoli046vs5	0.816	0.881	0.889	0.841
Ecoli01vs235	0.735	0.825	0.771	0.753
Ecoli0267vs35	0.802	0.802	0.827	0.822
Glass04vs5	0.994	0.994	0.994	0.994
Ecoli0346vs5	0.841	0.897	0.897	0.894
Ecoli0347vs56	0.815	0.796	0.769	0.751
Yeast05679vs4	0.696	0.782	0.801	0.709
Ecoli067vs5	0.840	0.855	0.865	0.862
Vowel0	0.950	0.966	0.963	0.954
Glass016vs2	0.519	0.628	0.635	0.565
Glass2	0.558	0.738	0.717	0.690
Ecoli0147vs2356	0.821	0.845	0.825	0.805
Led7digit02456789vs1	0.881	0.908	0.908	0.894
Mean	0.776	**0.832**	0.823	0.802
Glass06vs5	0.945	0.945	0.945	0.945
Ecoli01vs5	0.838	0.838	0.861	0.836
Glass0146vs2	0.497	0.740	0.658	0.634
Ecoli0147vs56	0.796	0.858	0.865	0.830
Cleveland0vs4	0.748	0.751	0.751	0.754
Ecoli0146vs5	0.744	0.865	0.834	0.813
Ecoli4	0.815	0.864	0.840	0.840
Yeast1vs7	0.546	0.690	0.609	0.580
Shuttle0vs4	1.000	1.000	1.000	1.000
Glass4	0.870	0.904	0.935	0.870
Page-blocks13vs2	0.997	0.997	0.997	0.997
Glass016vs5	0.844	0.844	0.888	0.844
Shuttle2vs4	0.950	0.950	0.950	0.950
Yeast1458vs7	0.493	0.542	0.649	0.543
Glass5	0.847	0.897	0.897	0.895
Yeast2vs8	0.773	0.773	0.773	0.773
Yeast4	0.545	0.770	0.869	0.633
Yeast1289vs7	0.566	0.599	0.771	0.530
Yeast5	0.885	0.965	0.914	0.905
Ecoli0137vs26	0.746	0.746	0.848	0.746
Yeast6	0.739	0.802	0.892	0.781
Mean	0.771	0.825	**0.845**	0.795
Total Mean	0.794	0.846	**0.847**	0.821

Table 20: GM-base comparison between penalization factors $PF_1 = 1/(2\,IR)$, $PF_2 = 1 - (1/IR)$ and without penalization factor.

Dataset	FURIA	FURIA-CS (FP2)	FURIA-CS (FP1)	FURIA-CS
	Tst_{GM}	Tst_{GM}	Tst_{GM}	Tst_{GM}
Glass1	0.794	0.774	0.760	0.731
Ecoli0vs1	0.995	0.986	0.986	0.986
Wisconsin	0.975	0.971	0.977	0.974
Pima	0.705	0.720	0.715	0.742

Continued on next page

Table 20 – *Continued from previous page*

Dataset	FURIA	FURIA-CS (FP2)	FURIA-CS (FP1)	FURIA-CS
	Tst_{GM}	Tst_{GM}	Tst_{GM}	Tst_{GM}
Iris0	1.000	1.000	1.000	1.000
Glass0	0.855	0.823	0.823	0.813
Yeast1	0.678	0.682	0.682	0.676
Vehicle1	0.694	0.751	0.751	0.734
Vehicle2	0.982	0.977	0.977	0.974
Vehicle3	0.758	0.766	0.766	0.692
Haberman	0.532	0.667	0.637	0.645
Glass0123vs456	0.838	0.902	0.891	0.868
Vehicle0	0.925	0.946	0.936	0.933
Ecoli1	0.893	0.883	0.866	0.863
New-thyroid2	0.924	0.950	0.950	0.950
New-thyroid1	0.957	0.962	0.962	0.967
Ecoli2	0.883	0.916	0.859	0.867
Segment0	0.992	0.992	0.988	0.988
Glass6	0.854	0.903	0.903	0.884
Yeast3	0.860	0.920	0.918	0.884
Ecoli3	0.699	0.850	0.765	0.749
Page-blocks0	0.935	0.941	0.941	0.933
Mean	0.850	**0.876**	0.865	0.857
Ecoli034vs5	0.816	0.882	0.872	0.839
Yeast2vs4v	0.865	0.891	0.875	0.850
Ecoli067vs35	0.794	0.771	0.771	0.747
Ecoli0234vs5	0.761	0.870	0.854	0.854
Glass015vs2	0.050	0.373	0.373	0.376
Yeast0359vs78	0.557	0.679	0.493	0.583
Yeast02579vs368	0.861	0.913	0.906	0.877
Yeast0256vs3789	0.624	0.773	0.742	0.649
Ecoli046vs5	0.852	0.870	0.878	0.822
Ecoli01vs235	0.608	0.795	0.715	0.694
Ecoli0267vs35	0.738	0.773	0.805	0.797
Glass04vs5	0.982	0.994	0.994	0.994
Ecoli0346vs5	0.931	0.890	0.890	0.887
Ecoli0347vs56	0.737	0.771	0.729	0.699
Yeast05679vs4	0.555	0.770	0.784	0.648
Ecoli067vs5	0.818	0.828	0.837	0.834
Vowel0	0.978	0.966	0.962	0.949
Glass016vs2	0.162	0.535	0.486	0.321
Glass2	0.475	0.731	0.587	0.554
Ecoli0147vs2356	0.912	0.825	0.802	0.773
Led7digit02456789vs1	0.948	0.902	0.902	0.886
Mean	0.711	**0.800**	0.773	0.744
Glass06vs5	0.955	0.936	0.936	0.936
Ecoli01vs5	0.763	0.820	0.845	0.818
Glass0146vs2	0.340	0.707	0.511	0.471
Ecoli0147vs56	0.811	0.848	0.854	0.814
Cleveland0vs4	0.635	0.702	0.702	0.705
Ecoli0146vs5	0.779	0.856	0.801	0.704
Ecoli4	0.835	0.854	0.811	0.811
Yeast1vs7	0.323	0.679	0.358	0.311
Shuttle0vs4	1.000	1.000	1.000	1.000
Glass4	0.925	0.892	0.926	0.855
Page-blocks13vs2	0.993	0.997	0.997	0.997
Glass016vs5	0.868	0.736	0.873	0.736
Shuttle2vs4	0.971	0.941	0.941	0.941
Yeast1458vs7	0.081	0.493	0.495	0.195
Glass5	0.899	0.797	0.797	0.795
Yeast2vs8	0.614	0.728	0.728	0.728
Yeast4	0.506	0.763	0.866	0.455
Yeast1289vs7	0.152	0.386	0.636	0.162
Yeast5	0.916	0.964	0.910	0.900
Ecoli0137vs26	0.269	0.538	0.740	0.540
Yeast6	0.606	0.789	0.885	0.739
Mean	0.662	0.782	**0.796**	0.696
Total Mean	0.741	**0.819**	0.809	0.765