On dynamic ensemble selection and data preprocessing for multi-class imbalance learning

Rafael M. O. Cruz *

Laboratoire d'imagerie, de vision et d'intelligence artificielle
École de Technologie Supérieure
Montreal, H3C 1K3, Canada
rafaelmenelau@gmail.com

Mariana A. Souza *

Centro de Informática
Universidade Federal de Pernambuco
Recife, 50.670-420, Brazil
mas2@cin.ufpe.br

Robert Sabourin

Laboratoire d'imagerie, de vision et d'intelligence artificielle
École de Technologie Supérieure
Montreal, H3C 1K3, Canada
Robert.Sabourin@etsmtl.ca

George D. C. Cavalcanti

Centro de Informática
Universidade Federal de Pernambuco
Recife, 50.670-420, Brazil
gdcc@cin.ufpe.br

Class-imbalance refers to classification problems in which many more instances are available for certain classes than for others. Such imbalanced datasets require special attention because traditional classifiers generally favor the majority class which has a large number of instances. Ensemble of classifiers have been reported to yield promising results. However, the majority of ensemble methods applied to imbalanced learning are static ones. Moreover, they only deal with binary imbalanced problems. Hence, this paper presents an empirical analysis of dynamic selection techniques and data preprocessing methods for dealing with multi-class imbalanced problems. We considered five variations of preprocessing methods and fourteen dynamic selection schemes. Our experiments conducted on 26 multi-class imbalanced problems show that the dynamic ensemble improves the AUC and the G-mean as compared to the static ensemble. Moreover, data preprocessing plays an important role in such cases.

Keywords: Imbalanced learning; multi-class imbalanced; ensemble of classifiers; dynamic classifier selection; data preprocessing.

*Indicates equal contribution

1. Introduction

Class-imbalance[34] refers to classification problems in which many more instances are available for certain classes than for others. Particularly, in a two-class scenario, one class contains the majority of instances (the *majority class*), while the other (the *minority class*) contains fewer instances. Imbalanced datasets may originate from real life problems including the detection of fraudulent bank account transactions,[50] telephone calls,[2] biomedical diagnosis,[38] image retrieval[41] and so on. Due to the under-representation of the minority class, traditional classification algorithms tend to favor the majority class in the learning process. This bias leads to a poor performance over the minority class,[42] which may be an issue since the latter is usually of higher importance than the majority class in many problems, such as in the diagnosis of rare diseases.

One of the biggest challenges in imbalance learning is dealing with multi-class imbalanced problems.[36] Multi-class imbalanced classification is not as well developed as the binary case, with only a few papers handling this issue.[1,22,23] It is also considered as a more complicated problem, since the relation among the classes is no longer obvious. For instance, one class may be the majority one when compared to some classes, and minority when compared to others. Moreover, we may easily lose performance on one class while trying to gain it on another.[22]

A plethora of techniques was designed for addressing imbalanced problems, and they can be classified into one of the following four groups:[25,21] algorithm-level approaches, data-level approaches, cost-sensitive learning frameworks, and ensemble-based approaches. Algorithm-level approaches modify the existing learning algorithms so that they take into account the imbalance between the problem's classes. The data-level approaches include sampling-based preprocessing techniques which rebalance the original imbalanced class distribution to reduce its impact in the learning process. Cost-sensitive learning frameworks combine both data-level and algorithm-level approaches by assigning misclassification costs to each class and modifying the learning algorithms to incorporate them. Lastly, the ensemble-based approaches integrate any of the previous approaches (usually preprocessing techniques) with an ensemble learning algorithm. This work focus on this group of solutions.

As shown in Ref. 19, an ensemble of diverse classifiers can better cope with imbalanced distributions. In particular, Dynamic Selection (DS) techniques is seen as an alternative to deal with multi-class imbalanced problems as it explores the local competence of each base classifier according to each new query sample.[13,36,43] A few recent works on ensemble-based approaches apply DS for dealing with multi-class imbalanced problems. In Ref. 26, the authors proposed a Dynamic Ensemble Selection (DES) technique that combines a preprocessing method based on random balance and a DS scheme which assigns a higher competence to classifiers that correctly label minority class samples in the local region where the query sample is located. The proposed preprocessing technique makes use of Random

Under-Sampling (RUS), Random Over-Sampling (ROS) and Synthetic Minority Oversampling Technique (SMOTE) for obtaining balanced sets for training the base-classifiers in the pool. A DES technique was also proposed in Ref. 37 to deal with multi-class imbalanced problems. The clustering-based technique divides the feature space into regions and assigns different weights to the base-classifiers in each area. The output of the technique is then given by the weighted response of the local ensemble assigned to the region where the query sample is located. The classifiers' weights and the clusters' locations are obtained using an evolutionary scheme with a skew-intensive optimization criterion, with the purpose of reducing the class bias in the responses of the defined local ensembles.

A key factor in dynamic selection is the estimation of the classifiers' competences according to each test sample. Usually the estimation of the classifiers competences are based on a set of labelled samples, called the dynamic selection dataset (DSEL). However, as reported in Ref. 14, dynamic selection performance is very sensitive to the distribution of samples in DSEL. If the distribution of DSEL itself becomes imbalanced, then there is a high probability that the region of competence for a test instance will become lopsided. Thus, the dynamic selection algorithms might end up biased towards selecting base classifiers that are experts for the majority class. With this in mind, we propose the use of data preprocessing methods for training a pool of classifiers as well as balancing the class distribution in DSEL for the DS techniques.

Hence, in this paper, we perform a study on the application of dynamic selection techniques and data preprocessing for handling with multi-class imbalance. Five data preprocessing techniques and nine DS techniques as well as static ensemble combination are considered in our experimental analysis. We also evaluate five of the DS techniques within the Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES) framework.[39] Experiments are conducted using 26 multi-class imbalanced datasets with varying degrees of class imbalance. The following research questions are studied in this paper:

(1) Does data preprocessing play an important role in the performance of dynamic selection techniques?
(2) Which data preprocessing technique is better suited for dynamic and static ensemble combination?
(3) Do dynamic ensembles present better performance than static ensembles?

This paper is organized as follows: Section 2 presents the related works on dynamic selection and describes the DCS and DES methods considered in this analysis. Data preprocessing techniques for imbalance learning are presented in Section 3. Experiments are conducted in Section 4. Conclusion and future works are presented in the last section.

2. Dynamic selection

A dynamic selection (DS) enables the selection of one or more base classifiers from a pool, given a test instance. This is based on the assumption that each base classifier is an expert in a different local region in the feature space. Therefore, the most competent classifiers should be selected in classifying a given unknown instance. The notion of competence is used in DS as a way of selecting, from a pool of classifiers, the best classifiers to classify a given test instance.

The general process for obtaining a specific Ensemble of Classifiers (EoC) for each query instance can be divided in three steps:[13] Region of Competence (RoC) definition, competence estimation and selection approach. In the first step, the local area in which the query sample is located is obtained. This area is called the Region of Competence (RoC) of the query instance. Then, the competence of each classifier in the query sample's RoC is estimated according to a given competence measure in the second step. Finally, either the classifier with highest competence level is singled out or an ensemble composed of the most competent classifiers is selected in the last step. If the former, the selection approach is a Dynamic Classifier Selection (DCS) scheme. If the latter, a Dynamic Ensemble Selection (DES) approach is used. Choosing more than one classifier to label the query instance can be advantageous since the risk of selecting an unsuitable one is distributed in DES schemes.[35] Both these strategies have been studied in recent years, and some papers are available examining them.[9,15]
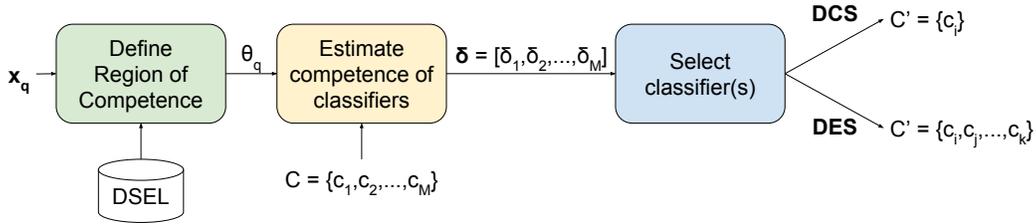

Fig. 1: Steps of a DS scheme. DSEL is the dynamic selection dataset, which contains labelled samples, $\mathbf{x_q}$ is the query sample, θ_q is the query sample's Region of Competence (RoC), C is the pool of classifiers, $\boldsymbol{\delta}$ is the competence vector composed of the estimated competences δ_i of each classifier c_i and C' is the resulting EoC of the selection scheme. If the selection approach is DCS, C' will contain only one classifier from C. Otherwise, the most competent classifiers in C will be chosen to form the EoC.

Figure 1 illustrates the usual procedure for dynamically selecting classifiers. The query sample $\mathbf{x_q}$ and a set of labelled instances called the dynamic selection dataset (DSEL) are used to define the query sample's RoC (θ_q). The DSEL dataset can be either the training or validation set. The RoC θ_q consists of a subset of the DSEL

dataset, and it usually contains the closest labelled instances to the query sample obtained using the k-nearest neighbors (KNN) rule. Then, the competence of each classifier from the original pool C is estimated over θ_q using a competence measure. The estimated competence of classifier c_i is denoted as δ_i. The *competence vector* $\boldsymbol{\delta}$, which contains the competence estimates from all classifiers in C, is then used in the selection approach, which can be a DCS or a DES method, to select the EoC C' that will be used to label $\mathbf{x_q}$.

To establish the classifiers' competence, given a test instance and the DSEL, the literature reports a number of measures classified into individual-based and group-based measures.[13] The former only take into account the individual performance of each base classifier, while the latter consider the interaction between the classifiers in the pool for obtaining the competence estimates. However, the competence level of the classifiers are calculated differently by different methods within each category. That is, the DS techniques may be based on different criteria to estimate the local competence of the base classifiers, such as accuracy,[54] ranking,[44] oracle information,[35] diversity[47] and meta-learning,[15] among others. Implementation of several DS techniques can be found on DESLib,[12] a dynamic ensemble learning library in Python available at `https://github.com/Menelau/DESlib`. In this paper, we evaluate the three DCS and six DES strategies described next, which are based on various criteria and were among the best performing DS methods according to the experimental analysis conducted in Ref. 13. We also evaluate the performance of five of these DS techniques using the FIRE-DES framework,[39] also described in this section, namely the FIRE-LCA, FIRE-MCB, FIRE-KNE, FIRE-KNU and FIRE-DES-KNN.

Modified Classifier Rank (RANK)

The Modified Classifier Rank[44] is a DCS method that exploits ranks of individual classifiers in the pool for each test instance $\mathbf{x_q}$. The competence δ_i of a given classifier c_i is estimated as the number of consecutive nearest neighbors $\mathbf{x_k} \in \theta_q$ it correctly classifies. The classifiers are then ranked with respect to their competence level, and the one with highest rank is selected to label the query sample. Algorithm 1 describes this procedure.

Local Class Accuracy (LCA)

The Local Class Accuracy method[54] estimates the classifiers' competence as their accuracy over the query sample's local region, taking into account the label they assigned to the test instance $\mathbf{x_q}$, as shown in Algorithm 2. Thus, for a given classifier c_i, which assigned the label ω_l to the query sample, its competence δ_i is defined as the percentage of correctly classified instances among the ones in the region of competence θ_q that belong to class ω_l. The classifier presenting the highest competence is used for the classification of the query sample.

6 *R. M. O. Cruz, M. A. Souza, R. Sabourin and G. D. C. Cavalcanti*

Algorithm 1 RANK method.

Input: $DSEL, C$ ▷ DSEL dataset and pool of classifiers
Input: $\mathbf{x_q}, K$ ▷ Query sample and RoC size
Output: c_k ▷ The highest ranked classifier
 1: $\theta_q \leftarrow KNN(DSEL, K)$ ▷ RoC definition
 2: **for** every c_i in C **do**
 3:     $\delta_i \leftarrow 0$
 4:     **for** every $\mathbf{x_j}$ in θ_q **do**
 5:         **if** c_i correctly labels $\mathbf{x_j}$ **then**
 6:             $\delta_i \leftarrow \delta_i + 1$
 7:         **else**
 8:             **break**
 9:         **end if**
10:     **end for**
11: **end for**
12: Rank the classifiers in C according to $\boldsymbol{\delta}$
13: **return** the highest ranked classifier c_k

Algorithm 2 LCA method.

Input: $DSEL, C$ ▷ DSEL dataset and pool of classifiers
Input: $\mathbf{x_q}, K$ ▷ Query sample and RoC size
Output: c_k ▷ The most competent classifier
 1: $\theta_q \leftarrow KNN(DSEL, K)$ ▷ RoC definition
 2: **for** every c_i in C **do**
 3:     $\delta_i \leftarrow 0$
 4:     $\omega_l \leftarrow c_i(\mathbf{x_q})$ ▷ Predict output of the query sample
 5:     $\theta'_q \leftarrow \{\mathbf{x_j} \in \theta_q | \mathbf{x_j}$ belongs to class $\omega_l\}$ ▷ Select neighbors from class ω_l
 6:     **for** every $\mathbf{x_j}$ in θ'_q **do**   ▷ Compute accuracy within θ'_q
 7:         **if** c_i correctly labels $\mathbf{x_j}$ **then**
 8:             $\delta_i \leftarrow \delta_i + 1$
 9:         **end if**
10:     **end for**
11:     $\delta_i \leftarrow \delta_i / |\theta'_q|$ ▷ Calculate proportion of correctly classified samples in θ'_q
12: **end for**
13: Select the classifier c_k in C with highest δ_k
14: **return** the most competent classifier c_k

Multiple Classifier Behavior (MCB)

The Multiple Classifier Behavior[29] is a DCS technique based on both accuracy and behavior of the classifiers. The latter is a characteristic that comes from the *decision space*, which relates to the responses given by the base-classifiers in the pool. A sample $\mathbf{x_i}$ in the feature space may be represented as an instance in the decision space by its output profile $\mathbf{u_i}$, which consists of all base-classifiers' predictions for that sample. The decision space may be used to various ends, from RoC definition, as in the case of MCB, to even learning the behavior of the classifiers in the pool.[49,15]

Algorithm 3 describes the selection procedure by the MCB method. Given an

unknown instance $\mathbf{x_q}$, the output profile ($\mathbf{u_k}$) of each of its neighbors $\mathbf{x_k} \in \theta_q$ is first obtained. Then, the output profile of the test instance ($\mathbf{u_q}$) is also obtained and compared to the output profile of each of its neighbors using a similarity measure (Eq. (1)) defined as the proportion of equal corresponding coordinate values between the two profiles.

$$
\begin{aligned}
Sim(\mathbf{u_i}, \mathbf{u_j}) &= \frac{1}{M} \sum_{k=1}^{M} T_k(\mathbf{u_i}, \mathbf{u_j}), \\
T_k(\mathbf{u_i}, \mathbf{u_j}) &= \begin{cases} 0, \text{if } u_{i,k} \neq u_{j,k} \\ 1, \text{if } u_{i,k} = u_{j,k} \end{cases}
\end{aligned}
\tag{1}
$$

Algorithm 3 MCB method.

Input: $DSEL, C$ ▷ DSEL dataset and pool of classifiers
Input: $\mathbf{x_q}, K$ ▷ Query sample and RoC size
Input: t_s, t_c ▷ Similarity and competence thresholds
Output: c_k ▷ The most competent classifier
 1: $\mathbf{u_q} \leftarrow C(\mathbf{x_q})$ ▷ Compute the responses of all classifiers (i.e., the output profile) of $\mathbf{x_q}$
 2: $\theta_q \leftarrow KNN(DSEL, K)$ ▷ RoC definition
 3: $\theta_q' \leftarrow \{\}$ ▷ New RoC
 4: **for** every $\mathbf{x_j}$ in θ_q **do**
 5:     $\mathbf{u_j} \leftarrow C(\mathbf{x_j})$ ▷ Compute the output profile of $\mathbf{x_j}$
 6:     **if** $Sim(\mathbf{u_q}, \mathbf{u_j}) > t_s$ **then**   ▷ Compare the output profiles (Eq. (1))
 7:         $\theta_q' \leftarrow \theta_q' \cup \mathbf{x_j}$
 8:     **end if**
 9: **end for**
10: $\delta_i \leftarrow 0$
11: **for** every $\mathbf{x_j}$ in θ_q' **do**   ▷ Compute accuracy within θ_q'
12:     **if** c_i correctly labels $\mathbf{x_j}$ **then**
13:         $\delta_i \leftarrow \delta_i + 1$
14:     **end if**
15:     $\delta_i \leftarrow \delta_i / |\theta_q|$ ▷ Calculate proportion of correctly classified samples in θ_q
16: **end for**
17: Select the classifier c_k in C with highest δ_k
18: **if** δ_k is greater by t_c than all other competences in $\boldsymbol{\delta}$ **then**
19:     **return** the most competent classifier c_k
20: **else**
21:     **return** C
22: **end if**

The instances with output profiles similar to the query sample's, according to a pre-defined threshold, remain in the region of competence, while the rest is removed from θ_q. Then, the competence δ_i of classifier c_i is estimated as its local accuracy over the new region of competence, and if the difference between the most accurate classifier and the second most accurate surpasses a second threshold, the former is

selected to label the query sample. Otherwise, the majority voting of all classifiers is used for classification.

KNORA-Eliminate (KNE)

The KNORA-Eliminate technique [35] explores the concept of Oracle, which is the upper limit of a DCS technique. Given the region of competence θ_q, only the classifiers that correctly recognize all samples belonging to the region of competence are selected. In other words, all classifiers that achieved a 100% accuracy in this region (i.e., that are local Oracles) are selected to compose the ensemble of classifiers. Then, the decisions of the selected base classifiers are aggregated using the majority voting rule. If no base classifier is selected, the size of the region of competence is reduced by one, and the search for the competent classifiers is restarted. This procedure is described in Algorithm 4.

Algorithm 4 KNE method.

Input: $DSEL, C$ ▷ DSEL dataset and pool of classifiers
Input: $\mathbf{x_q}, K$ ▷ Query sample and RoC size
Output: C' ▷ EoC
 1: $\theta_q \leftarrow KNN(DSEL, K)$ ▷ RoC definition
 2: $\theta'_q \leftarrow \theta_q$ ▷ New RoC is initially the original RoC
 3: **while** θ'_q is not empty **do**
 4:     **for** every c_i in C **do**
 5:         $\delta_i \leftarrow 0$
 6:         **for** every $\mathbf{x_j}$ in θ'_q **do**   ▷ Compute accuracy within θ'_q
 7:             **if** c_i correctly labels $\mathbf{x_j}$ **then**
 8:                 $\delta_i \leftarrow \delta_i + 1$
 9:             **end if**
10:         **end for**
11:     **end for**
12:     **if** there are c_k with $\delta_k = |\theta'_q|$ **then**
13:         $C' \leftarrow C' \cup c_k$ ▷ Add perfectly accurate classifiers to C'
14:     **else**
15:         remove the most distant sample $\mathbf{x_j}$ from θ'_q ▷ Reduce the new RoC θ'_q by one
16:     **end if**
17: **end while**
18: **if** θ'_q is not empty **then**
19:     **return** C'
20: **else**
21:     **return** C
22: **end if**

KNORA-Union (KNU)

The KNORA-Union technique[35] selects all classifiers that are able to correctly recognize at least one sample in the region of competence θ_q. The competence δ_i of

a given classifier c_i is estimated as the number of samples in θ_q for which it predicted the correct label. The method then selects all classifiers with competence level above zero. The KNU selection scheme is shown in Algorithm 5. The responses of the selected classifiers are combined using a majority voting scheme which considers that a base classifier can vote more than once when it correctly classifies more than one instance in the region of competence. For instance, if a given base classifier c_i predicts the correct label for three samples belonging to the region of competence, it gains three votes for the majority voting scheme. The votes collected by all base classifiers are aggregated to obtain the ensemble decision. So, in addition to the selected EoC, the KNU technique returns the competence vector $\boldsymbol{\delta}$ to be used by the aggregation scheme, as shown in Algorithm 5.

Algorithm 5 KNU method.

Input: $DSEL, C$ ▷ DSEL dataset and pool of classifiers
Input: $\mathbf{x_q}, K$ ▷ Query sample and RoC size
Output: $C', \boldsymbol{\delta}$ ▷ EoC and competence estimates
 1: $\theta_q \leftarrow KNN(DSEL, K)$ ▷ RoC definition
 2: **for** every c_i in C **do**
 3:     $\delta_i \leftarrow 0$
 4:     **for** every $\mathbf{x_j}$ in θ_q **do**   ▷ Compute accuracy within θ_q
 5:         **if** c_i correctly labels $\mathbf{x_j}$ **then**
 6:             $\delta_i \leftarrow \delta_i + 1$
 7:         **end if**
 8:     **end for**
 9:     **if** $\delta_i > 0$ **then**
10:         $C' \leftarrow C' \cup c_i$ ▷ Add classifiers with competence above zero
11:     **end if**
12: **end for**
13: **return** the EoC C' and the competence estimates $\boldsymbol{\delta}$

DES-KNN

The DES-KNN[47] technique relies on both an individual-based measure and a group-based measure, namely local accuracy and diversity, to estimate the classifiers' competence. Algorithm 6 describes its selection scheme. Firstly, the individual local accuracy of each classifier c_i is estimated over the query instance's region of competence θ_q, obtained using the KNN. Then, the N classifiers with highest local accuracies are pre-selected, with N being a pre-defined parameter. The pre-selected classifiers are then ranked again based on their diversity using a diversity measure so that the J most diverse ones among them, with J also being pre-set, are used for classifying the query sample $\mathbf{x_q}$. The diversity measure used was the Double-Fault measure,[28] since it provided the highest correlation with accuracy in Ref. 46.

Algorithm 6 DES-KNN method.

Input: $DSEL, C$ ▷ DSEL dataset and pool of classifiers
Input: $\mathbf{x_q}, K$ ▷ Query sample and RoC size
Input: N, J ▷ Parameters
Output: C' ▷ EoC
  1: $\theta_q \leftarrow KNN(DSEL, K)$ ▷ RoC definition
  2: **for** every c_i in C **do**
  3:      $\delta_i \leftarrow 0$
  4:      **for** every $\mathbf{x_j}$ in θ_q **do**   ▷ Compute accuracy within θ_q
  5:          **if** c_i correctly labels $\mathbf{x_j}$ **then**
  6:              $\delta_i \leftarrow \delta_i + 1$
  7:          **end if**
  8:      **end for**
  9: **end for**
10: **for** every c_i in C **do**
11:      **for** every c_j in C **do**
12:          **if then** $i \neq j$
13:              $D_{i,j} \leftarrow DoubleFault(c_i, c_j)$ ▷ Compute the diversity between c_i e c_j
14:          **end if**
15:      **end for**
16: **end for**
17: Rank the classifiers in C according to $\boldsymbol{\delta}$
18: $C'' \leftarrow$ the N most accurate classifiers in C
19: Rank the classifiers in C'' according to D
20: $C' \leftarrow$ the j most diverse classifiers in C''
21: **return** the EoC C'

Dynamic Ensemble Selection Performance (DES-P)

The Dynamic Ensemble Selection Performance technique[53] estimates the competence of each classifier as the difference between its local accuracy over the region of competence θ_q and the performance of the random classifier, that is, the classification model that randomly chooses a class with equal probabilities. The performance of the random classifier is defined by $RC = 1/L$, with L being the number of classes in the problem. Thus, for a given classifier c_i, its competence δ_i is estimated by Eq. (2), with $\hat{P}(c_i|\theta_q)$ being the classifier's local accuracy over the region of competence.

$$\delta_i = \hat{P}(c_i|\theta_q) - \frac{1}{L} \tag{2}$$

After estimating the classifiers' competences, the technique selects the ones with competence level above zero, that is, the classifiers with local accuracy greater than the random classifier's. If no classifier meets this requirement, the entire pool is used for classifying the query sample. The selection procedure of the DES-P method is shown in Algorithm 7.

Algorithm 7 DES-P method.

Input: $DSEL, L, C$ ▷ DSEL dataset, number of classes and pool of classifiers
Input: $\mathbf{x_q}, K$ ▷ Query sample and RoC size
Output: C' ▷ EoC
1: $\theta_q \leftarrow KNN(DSEL, K)$ ▷ RoC definition
2: $RC \leftarrow 1/L$ ▷ Compute the performance of the random classifier
3: **for** every c_i in C **do**
4:      $\delta_i \leftarrow 0$
5:      **for** every $\mathbf{x_j}$ in θ_q **do**   ▷ Compute accuracy within θ_q
6:         **if** c_i correctly labels $\mathbf{x_j}$ **then**
7:            $\delta_i \leftarrow \delta_i + 1/|\theta_q|$
8:         **end if**
9:      **end for**
10:     $\delta_i \leftarrow \delta_i - RC$ ▷ Compute competence level (as in Eq. (2))
11: **end for**
12: **if** there are c_k with $\delta_k > 0$ **then**
13:     $C' \leftarrow C' \cup c_k$ ▷ Add classifiers with competence above RC to C'
14:     **return** C'
15: **else**
16:     **return** C
17: **end if**

Randomized Reference Classifier (DES-RRC)

The Randomized Reference Classifier[52] is a probability-based DS technique which makes use of the random reference classifier (RRC) from Ref. 51 for obtaining the classifiers' competence. More specifically, the competence level δ_i of a given classifier c_i is defined as the weighted sum of its source of competence C_{src}, calculated using the probability of correct classification of a RRC, over all validation instances (Eq. (3)). The weights are given by a Gaussian potential function, shown in Eq. (4), whose value decreases with the increase of the Euclidean distance between the query instance and the validation sample.

$$\delta_i = \sum_{\mathbf{x_k} \in DSEL} C_{src} K(\mathbf{x_q}, \mathbf{x_k}) \tag{3}$$

$$K(\mathbf{x_q}, \mathbf{x_k}) = exp(-d(\mathbf{x_q}, \mathbf{x_k})^2) \tag{4}$$

The classifiers with competence estimate greater than the probability of random classification are selected and used for classifying the unknown sample.

META-DES

The META-DES[15] is a framework based on the premise that the dynamic ensemble selection problem can be considered as a meta-problem,[16] which uses different criteria regarding the behavior of c_i to decide if it is competent or not to label the query sample $\mathbf{x_q}$. The meta-problem is defined as follows:

- The **meta-classes** are either "competent" (1) or "incompetent" (0) to classify $\mathbf{x_q}$.
- Each set of **meta-features** f_i corresponds to a different criterion for measuring the level of competence of a base classifier.
- The meta-features are encoded into a meta-features vector $v_{i,q}$.
- A **meta-classifier** λ is trained based on the meta-features $v_{i,q}$ to predict whether or not c_i will achieve the correct prediction for $\mathbf{x_q}$, i.e., if it is competent enough to classify $\mathbf{x_q}$.

Thus, a meta-classifier λ is trained and used to decide whether or not each classifier in the pool is competent enough to classify a given query sample $\mathbf{x_q}$. During the meta-training phase of the framework, the meta-features are extracted from each sample in the training and DSEL sets. The set of meta-features may include different criteria for measuring a classifier's competence, such as local accuracy, output profile, posterior probability, and so on. The meta-classifier λ is then trained using the extracted meta-features, in order to learn the classifier selection rule. During the generalization phase, the meta-features of a given unknown sample are extracted and used as input to the meta-classifier, which estimates the competence level δ_i of each classifier c_i in the pool, deeming them as competent or not for classifying that specific sample. The competent classifiers are then selected to perform the classification task using majority voting.

FIRE-DES

Though not a DS technique, the FIRE-DES framework exploits the local competence of the classifiers by performing a dynamic ensemble pruning on the original pool of classifiers C prior to the use of a DS scheme. The idea behind the FIRE-DES is that a number of DS techniques may still select incompetent classifiers for labelling unknown instances located in *indecision regions*, that is, areas near the class borders. This issue is illustrated in Figure 2, in which $\mathbf{x_q}$ is a query sample located in an indecision region and c_1 and c_2 are two classifiers from the pool.

In this example, both classifiers correctly label 4 out of 5 neighbors of $\mathbf{x_q}$. However, while c_1 only recognizes the green class in the query sample's RoC, c_2 is able to predict different labels in this region because it "crosses" the RoC. Moreover, c_2 recognizes the local border in the RoC since it correctly labels at least one pair of *frienemies*, that is, samples from different classes in it. Thus, it would be preferable to select c_2 instead of c_1 to label $\mathbf{x_q}$. Still, as both classifiers have the same local accuracy, a DS technique could select c_1 instead of c_2, and thus misclassify $\mathbf{x_q}$. This issue may be intensified for highly imbalanced problems, since the DS techniques may select classifiers that simply recognize well the majority class in the RoC, which would yield them a high local accuracy estimate regardless of their recognition of a local border near the query sample.

Thus, in order to avoid the selection of classifiers that do not "cross", that is, that do not recognize more than one class in the indecision region where $\mathbf{x_q}$ is,
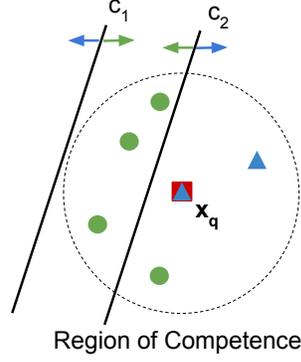
Fig. 2: Example in which a query sample $\mathbf{x_q}$, which belongs to the blue class, is located in an indecision region. While classifier c_1 only recognizes the instances from the majority class (green), c_2 crosses the RoC and correctly labels instances from both classes.

Algorithm 8 DFP method.

Input: $DSEL, C$ ▷ DSEL dataset and pool of classifiers
Input: $\mathbf{x_q}, K$ ▷ Query sample and RoC size
Output: C'' ▷ Subset of classifiers
1: $\theta_q \leftarrow KNN(DSEL, K)$ ▷ RoC definition
2: **if** there is more than one class in θ_q **then**
3:     $F \leftarrow \{\}$ ▷ Set of frienemies in θ_q
4:     **for** every $\mathbf{x_i}$ in θ_q **do**
5:         **for** every $\mathbf{x_j}$ in θ_q **do**
6:             **if** $\mathbf{x_i}$ and $\mathbf{x_j}$ belong to different classes **then**
7:                 $F \leftarrow F \cup (\mathbf{x_i}, \mathbf{x_j})$ ▷ Single out the pair of frienemies
8:             **end if**
9:         **end for**
10:     **end for**
11:     $C'' \leftarrow \{\}$
12:     **for** every c_i in C **do**
13:         **if** c_i correctly labels one pair of samples in F **then**
14:             $C'' \leftarrow C'' \cup c_i$ ▷ Add the classifiers that label a pair of frienemies correctly
15:         **end if**
16:     **end for**
17:     **return** C''
18: **else**
19:     **return** C
20: **end if**

---

the FIRE-DES dynamically removes such classifiers from the original pool before proceeding with the DS technique execution. To that end, the FIRE-DES framework uses the Dynamic Frienemy Pruning (DFP) method (Algorithm 8), which performs

an online pruning of the pool of classifiers based on the neighborhood of each test sample. Thus, if an unknown instance's RoC θ_q contains more than one class, the DFP pre-selects the classifiers from the pool that correctly label at least one pair of frienemies. The DS technique is then executed using the pre-selected pool yielded by the DFP instead of the original unpruned pool. The FIRE-DES scheme was designed for two-class problems and it yielded a significant improvement in the performance of most DS techniques, specially for highly imbalanced binary problems.[39]

Nevertheless, a crucial aspect in the performance of the dynamic selection techniques is the distribution of the dynamic selection dataset (DSEL), as the local competence of the base classifiers are estimated based on this set. Hence, preprocessing techniques can really benefit DS techniques as they can be employed to edit the distribution of DSEL, prior to performing dynamic selection.

3. Data preprocessing

Changing the distribution of the training data to compensate for poor representativeness of the minority class is an effective solution for imbalanced problems, and a plethora of methods are available in this regards. Branco et al.[7] divided such methods into three categories, namely, stratified sampling, synthesizing new data, and combinations of the two previous methods. While the complete taxonomy is available in Ref. 7, we will center our attention on the methods that have been used together with ensemble learning.[19]

One important category is under-sampling, which removes instances from the majority class to balance the distribution. Random under-sampling (RUS)[6] is one such method. RUS has been coupled with boosting (RUSBoost)[45] and with Bagging.[6] These combined techniques have been applied to several inherently imbalanced classification problems, such as sound event detection,[55] phishing detection,[30] software defect prediction,[48] detection of cerebral microbleds[5] and breast cancer cytological malignancy grading,[4] among others. A major drawback of RUS is that it can discard potentially useful data which can be a problem when using dynamic selection approaches.

The other strategy is the generation of new synthetic data. Synthesizing new instances has several known advantages,[10] and a wide number of proposals are available for building new synthetic examples. In this context, a famous method that uses interpolation to generate new instances is SMOTE.[10] SMOTE over-samples the minority class by generating new synthetic data. A number of methods have been developed based on the principle of SMOTE, such as, Borderline-SMOTE,[31] ADASYN,[33] RAMO[11] and Random balance.[18] Furthermore, Garcia et al.[27] observed that over-sampling consistently outperforms under-sampling for strongly imbalanced datasets.

Hence, in this work we considered three over-sampling techniques. Similar to Ref. 1, the class with the highest number of examples is considered the majority

class, while all others are considered minority classes. Then, the over-sampling techniques are applied to generate synthetic samples for each minority class.

Synthetic Minority Over-sampling Technique (SMOTE)

The Synthetic Minority Over-sampling Technique[10] creates artificial instances for the minority class by interpolating several samples that are within a defined neighborhood. The general idea of the process is as follows: Let $\mathbf{x_i}$ be a randomly selected instance from the minority class. To create an artificial instance from $\mathbf{x_i}$, SMOTE first isolates the k-nearest neighbors of $\mathbf{x_i}$, from the minority class. Afterwards, it randomly selects one neighbor and randomly generates a synthetic example along the imaginary line connecting $\mathbf{x_i}$ and the selected neighbor. The complete SMOTE procedure is shown in Algorithm 9, in which T, N and K are the three input parameters denoting the number of minority class samples, the amount of oversampling in terms of percentage and the number of nearest neighbors, respectively.

Algorithm 9 SMOTE procedure.

Input: S_P ▷ Set of minority class samples
Input: T, N, K ▷ No. of minority class samples, amount of oversampling and neighborhood size
Output: S_S ▷ Set of synthetic samples (of size $(N/100) * T$)
 1: **if** $N < 100$ **then**
 2:     Randomize the T minority class samples
 3:     $T \leftarrow (N/100) * T$
 4:     $N \leftarrow 100$
 5: **end if**
 6: $N \leftarrow \lfloor N/100 \rfloor$ ▷ N is assumed to be in integral multiples of 100
 7: $S_s \leftarrow \{\}$
 8: **for** i in $[1, T]$ **do**
 9:     $\theta \leftarrow KNN(S_P, K)$ ▷ Get nearest neighbors of $\mathbf{x_i}$
10:     **while** $N > 0$ **do**
11:         Randomly select a $\mathbf{x_k} \in \theta$
12:         **for** each feature j **do**
13:            $d \leftarrow x_{k,j} - x_{i,j}$ ▷ Difference in attribute j
14:            $g \leftarrow random([0, 1])$ ▷ Randomized gap
15:            $x'_{i,j} \leftarrow x_{i,j} + g * d$ ▷ New value in attribute j
16:         **end for**
17:         $S_S \leftarrow S_S \cup \mathbf{x'_i}$
18:         $N \leftarrow N - 1$
19:     **end while**
20: **end for**
21: **return** S_S

Ranked Minority Over-sampling (RAMO)

The Ranked Minority Over-sampling[11] method performs a sampling of the minority class according to a probability distribution, followed by the creation of synthetic instances. The RAMO process works as follows: For each instance $\mathbf{x_i}$ in the minority class, its k_1 nearest neighbors (k_1 is a user defined neighborhood size) from the whole dataset are isolated. The weight r_i of $\mathbf{x_i}$ is defined as:

$$r_i = \frac{1}{1 + exp(-\alpha.\delta_i)}, \tag{5}$$

where δ_i is the number of majority cases in the k-nearest neighborhood. Evidently, an instance with a large weight indicates that it is surrounded by majority class samples, and thus difficult to classify.

After determining all weights, the minority class is sampled using these weights to get a sampling minority dataset G. The synthetic samples are generated for each instance in G by using SMOTE on k_2 nearest neighbors where k_2 is a user-defined neighborhood size.

Random Balance (RB)

The Random Balance method[18] relies on the amount of under-sampling and over-sampling that is problem specific and that has a significant influence on the performance of the classifier concerned. RB maintains the size of the dataset, but varies the proportion of the majority and minority classes, using a random ratio. This includes the case where the minority class is over represented and the imbalance ratio is inverted. Thus, repeated applications of RB produce datasets having a large imbalance ratio variability, which promotes diversity.[18] SMOTE and random under-sampling are used to respectively increase or reduce the size of the classes to achieve the desired ratios.

The RB procedure is described in Algorithm 10, in which S is the dataset, S_P is the minority class set and S_N is the majority class set. Firstly, the new size of the majority class ($newMajSize$) is obtained by generating a random number between 2 and $|S| - 2$, which leaves the minority class with size $|S| - newMajSize$. Then, if the new majority class size is smaller than its original size $|S_N|$, the majority class S'_N is created by RUS the original S_N so that the final size $|S'_N| = newMajSize$. Consequently, the new minority class S'_P is obtained from S_P using SMOTE to create $newMinSize - |S_P|$ artificial instances. Otherwise, the new minority class S'_P is obtained by RUS the original minority class S_P until $|S'_P| = newMinSize$, while the remaining $newMajSize - |S_N|$ samples from the majority class are created using SMOTE, so that the final size of $|S'_N|$ is $newMajSize$.

Algorithm 10 RB procedure.

Input: S, S_P, S_N ▷ Original dataset, set of minority class samples and set of majority class samples

Input: K ▷ Neighborhood size of SMOTE

Output: S' ▷ Set of generated samples

1: $totalSize \leftarrow |S|$
2: $majSize \leftarrow |S_P|$
3: $minSize \leftarrow |S_N|$
4: $newMajSize \leftarrow random(2, totalSize - 2)$ ▷ Randomly obtain the new class sizes
5: $newMinSize \leftarrow totalSize - newMajSize$
6: $S' \leftarrow \{\}$
7: **if** $newMajSize < majSize$ **then**
8:     $S' \leftarrow S_P$ ▷ Include all minority samples
9:     $S' \leftarrow S' \cup RUS(S_N, newMajSize)$ ▷ Apply RUS on the majority class
10:    $S' \leftarrow S' \cup SMOTE(S_P, (newMinSize - minSize)/|S_P|, K)$ ▷ Generate minority class samples using SMOTE
11: **else**
12:    $S' \leftarrow S_N$ ▷ Include all majority samples
13:    $S' \leftarrow S' \cup RUS(S_P, newMinSize)$ ▷ Apply RUS on the minority class
14:    $S' \leftarrow S' \cup SMOTE(S_N, (newMajSize - majSize)/|S_N|, K)$ ▷ Generate majority class samples using SMOTE
15: **end if**
16: **return** S'

---

4. Experiments

4.1. *Datasets*

A total of 26 multi-class imbalanced datasets taken from the Keel repository[3] was used in this analysis. The key features of the datasets are presented in Table 1. The IR is computed as the proportion of the number of the majority class examples to the number of minority class examples. In this case, the class with maximum number of examples is the majority class, and the class with the minimum number of examples is the minority one. We grouped the datasets according to their IRs using the group definitions suggested in Ref. 20. Datasets with low IR ($IR < 3$) are highlighted with dark gray, whereas datasets with medium IR ($3 < IR < 9$) are in light gray.

4.2. *Experimental setup*

Experiments were performed using the DESLib,[12] and the results were obtained with a 5×2 stratified cross-validation. Performance evaluation is conducted using multi-class generalizations of the AUC, F-measure and G-mean, as the standard classification accuracy is not suitable for imbalanced learning.[19] In particular, the multi-class generalization of the AUC used in this work was the one from Ref. 32, which performs an estimation of the AUC for each pair of classes and then returns their averaged score. The generalization of the F-measure followed the weighted

Table 1: Characteristics of the 26 multi-class imbalanced datasets taken from the Keel repository. Column #E shows the number of instances in the dataset, column #A the number of attributes (numeric/nominal), #C shows the number of classes in the dataset, and column IR the imbalance ratio.

Dataset	#E	#A	#C	IR	Dataset	#E	#A	#C	IR
Vehicle	846	(18/0)	4	1.09	CTG	2126	(21/0)	3	9.40
Wine	178	(13/0)	3	1.48	Zoo	101	(16/0)	7	10.25
Led7digit	500	(7/0)	10	1.54	Cleveland	467	(13/0)	5	12.62
Contraceptive	1473	(9/0)	3	1.89	Faults	1941	(27/0)	7	14.05
Hayes-Roth	160	(4/0)	3	2.10	Autos	159	(16/10)	6	16.00
Column3C	310	(6/0)	3	2.50	Thyroid	7200	(21/0)	3	40.16
Satimage	6435	(36/0)	7	2.45	Lymphography	148	(3/15)	4	40.50
Laryngeal3	353	(16/0)	3	4.19	Abalone	4139	(7/1)	18	45.93
New-thyroid	215	(5/0)	3	5.00	Post-Operative	87	(1/7)	3	62.00
Dermatology	358	(33/0)	6	5.55	Wine-quality red	1599	(11/0)	11	68.10
Balance	625	(4/0)	3	5.88	Ecoli	336	(7/0)	8	71.50
Flare	1066	(0/11)	6	7.70	Page-blocks	5472	(10/0)	5	175.46
Glass	214	(9/0)	6	8.44	Shuttle	2175	(9/0)	5	853.00

mean approach, which calculates the metric for each class individually and then combines them using a weighted sum, with the proportion of samples from each corresponding class as the weights. Lastly, the G-mean score was obtained in this work as a higher root of the product of sensitivity for each class.

The pool size for all ensemble techniques was set to 100. The classifier used as a base classifier in all ensembles was the decision tree implementation from the Python library Scikit-learn,[40] which uses the CART[8] algorithm. Here, the decision trees were used without pruning and collapsing as recommended in Ref. 19. However, the minimum impurity decrease was set to 0.05 to function as an early stop in the training phase in order to improve the probability estimation of the unpruned trees.

Table 2: Preprocessing methods used for classifier pool generation. *Note.* Reprinted from "A study on combining dynamic selection and data preprocessing for imbalance learning," by A. Roy, R. Cruz, R. Sabourin and G. Cavalcanti, 2018, *Neurocomputing, 286*, 179–192. Copyright 2018 by Elsevier.

Bagging based methods		
Abbr.	Name	Description
Ba	Bagging	Bagging without preprocessing
Ba-RM100	Bagging+RAMO 100%	RAMO to double the minority class
Ba-RM	Bagging+RAMO	RAMO to make equal size for both classes
Ba-SM100	Bagging+SMOTE 100%	SMOTE to double the minority class
Ba-SM	Bagging+SMOTE	SMOTE to make equal size for both classes
Ba-RB	Bagging+RB	RB to randomly balance the two classes

All preprocessing techniques were combined with Bagging during the pool generation phase. Table 2 lists such combinations. The preprocessing techniques, RAMO and SMOTE, have user-specified parameters. In the case of RAMO, we used $k_1 = 10$, $k_2 = 5$ and $\alpha = 0.3$. For SMOTE and RB, the number of nearest neighbors was 5. These parameter settings were adopted from Ref. 19. Finally, for all the dynamic selection methods, we used 7 nearest neighbors to define the region of competence as in Ref. 15 and Ref. 13.
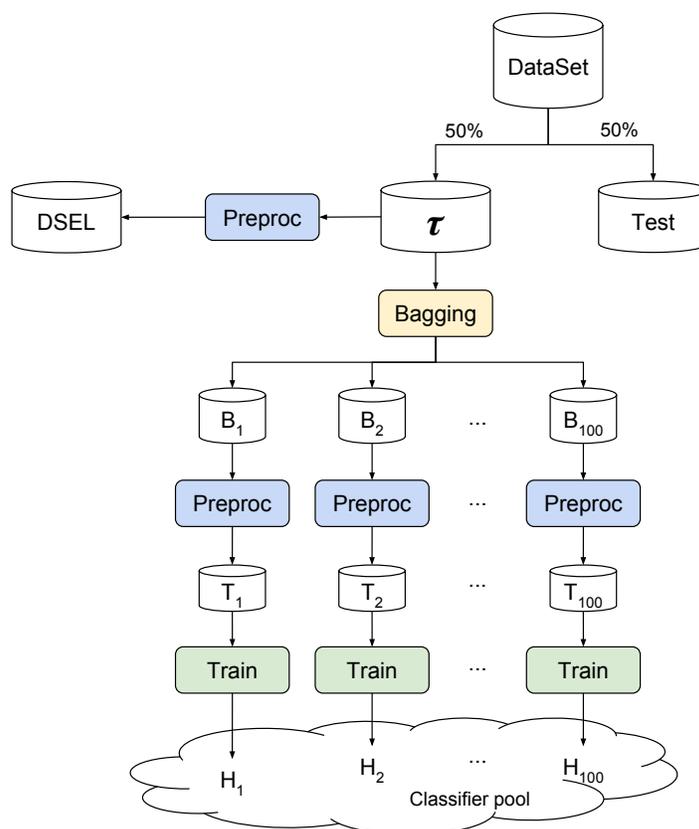
Fig. 3: The framework for training base classifiers and to prepare a DSEL for testing. Here, τ is the training data derived from the original dataset, B_i is the dataset generated from the ith Bagging iteration, T_i is the dataset produced by preprocessing (*Preproc*) B_i and H_i is the ith base classifier. Reprinted from "A study on combining dynamic selection and data preprocessing for imbalance learning," by A. Roy, R. Cruz, R. Sabourin and G. Cavalcanti, 2018, *Neurocomputing*, *286*, 179–192. Copyright 2018 by Elsevier.

The complete framework for a single replication is presented in Figure 3. The

original dataset was divided into two equal halves. One of them was set aside for testing, while the other half was used to train the base classifiers and to derive the dynamic selection set. Let us now highlight the process of setting up the DSEL. Here, instead of dividing the training set, we augment it using the data preprocessing, to create DSEL. Moreover, the Bagging method is applied to the training set, generating a bootstrap with 50% of the data. Then, the preprocessing method is applied to each bootstrap, and the resulting dataset is used to generate the pool of classifiers. Since we considered a single training dataset, the DSEL dataset has an overlap with the datasets used during Bagging iterations. However, the randomized nature of the preprocessing methods allows the DSEL not to be exactly the same as the training datasets, thus avoiding overfitting issues. Moreover, the use of overlapping training and validation sets was shown to improve the performance of classifier ensembles in comparison with disjoints datasets in Ref. 17.

4.3. *Results according to data preprocessing method*

In this section, we compare the performance of each preprocessing method with respect to each ensemble technique. Tables 3a, 3b and 3c show the average rank for the AUC, F-measure and G-mean, respectively. The best average rank is in bold. We can see that the RM and RM100 obtained the best results. Furthermore, the configuration using only Bagging always presented the greatest average rank for the AUC and G-mean.

The Finner's[24] step-down procedure was conducted at a 95% significance level to identify all methods that were equivalent to the best ranked one. The analysis demonstrates that considering the AUC and G-mean, the result obtained using preprocessing techniques is always statistically better when compared to using only Bagging. The same was not observed for the F-measure, probably because we used the weighted multi-class generalization of this measure, in which the classes with more samples have greater weights, and thus influence, in the calculation of the final F-measure score.

It can also be observed from Table 3 that, for all metrics, the best preprocessing method for any given DS technique was also the best one for the same technique within the FIRE-DES framework. This shows that there is not a single preprocessing technique that particularly favors the FIRE-DES scheme. Rather, the impact of the preprocessing procedure over the DS techniques is hardly changed by using them within the framework.

Moreover, we conducted a pairwise comparison between each ensemble methods using data preprocessing with the same methods using only Bagging (baseline). For the sake of simplicity, only the best data preprocessing for each technique was considered (i.e., the best result of each row of Table 3). The pairwise analysis is conducted using the Sign test, calculated on the number of wins, ties, and losses obtained by each method using preprocessing techniques, compared to the baseline. The results of the Sign test is presented in Figure 4.

Table 3: Average rankings according to (a) AUC, (b) F-measure and (c) G-mean. Methods in brackets are statistically equivalent to the best one.

(a)

Algorithm	Bagging	RM	RM100	SM	SM100	RB
STATIC	4.96	**2.39**	[3.39]	[3.25]	4.21	[2.79]
RANK	5.07	**2.54**	[2.89]	[3.04]	[3.57]	3.89
LCA	4.93	**2.14**	3.57	[2.79]	3.39	4.18
F-LCA	4.68	**2.43**	[3.57]	[2.96]	[3.29]	4.07
MCB	4.98	**2.66**	[2.91]	[2.88]	[3.70]	3.88
F-MCB	5.04	**2.36**	[3.30]	[3.07]	[3.38]	3.86
KNE	5.61	[2.68]	**2.66**	[3.45]	[3.79]	[2.82]
F-KNE	5.64	[2.71]	**2.64**	[3.50]	[3.79]	[2.71]
KNU	5.70	[2.48]	[3.14]	[3.23]	4.23	**2.21**
F-KNU	5.71	[2.41]	[3.20]	[3.27]	4.12	**2.29**
DES-KNN	5.96	**2.09**	[3.14]	3.39	4.05	[2.36]
F-DES-KNN	5.96	**2.20**	[3.07]	3.41	3.96	[2.39]
DESP	5.88	[2.45]	[2.89]	[3.38]	3.98	**2.43**
DES-RRC	5.89	**2.43**	[2.96]	[3.36]	3.86	[2.50]
META-DES	5.89	[2.59]	[2.98]	[3.20]	3.77	**2.57**

The Sign test demonstrated that the data preprocessing significantly improved the results of these techniques according to the AUC and G-mean. Considering these two metrics, all techniques obtained a significant number of wins for a significance level $\alpha = 0.01$. For the F-measure, nearly half of the techniques presented a significant number of wins for $\alpha = 0.10$. Hence, the results obtained demonstrate that data preprocessing techniques indeed play an important role when dealing with multi-class imbalanced problems.

Furthermore, DS techniques are more benefited from the application of data preprocessing (i.e., presented a higher number of wins). This result can be explained by the fact the data preprocessing techniques are applied in two stages. First, it is used in the ensemble generation stage in order to generate a diverse pool of classifiers. Then, they are also used in order to balance the distribution of the dynamic selection dataset for the estimation of the classifiers' competences.

Figure 4 also shows that, for the DS techniques within the FIRE-DES framework, the number of wins is almost always less or equal to the number of wins from the same DS technique by itself, for all metrics. This suggests that, as opposed to the two-class problems' case, the FIRE-DES scheme does not yield an improvement in performance for the DS techniques for multi-class imbalanced problems.

22   *R. M. O. Cruz, M. A. Souza, R. Sabourin and G. D. C. Cavalcanti*

(b)

Algorithm	Bagging	RM	RM100	SM	SM100	RB
STATIC	[3.54]	[4.27]	**2.86**	[3.21]	[3.25]	[3.88]
RANK	[3.43]	4.20	**2.55**	[3.16]	[2.96]	4.70
LCA	[3.32]	3.89	[3.20]	[2.93]	**2.41**	5.25
F-LCA	[3.04]	4.00	[3.18]	[3.14]	**2.46**	5.18
MCB	[3.36]	4.32	**2.68**	[3.04]	[3.11]	4.50
F-MCB	[3.32]	4.05	**2.46**	[3.30]	[3.20]	4.66
KNE	[3.70]	4.00	**2.54**	[3.54]	[3.02]	4.21
F-KNE	[3.77]	4.05	**2.50**	[3.50]	[3.00]	4.18
KNU	[3.75]	[4.14]	**2.93**	[3.21]	[3.32]	[3.64]
F-KNU	[3.61]	[4.20]	**2.98**	[3.21]	[3.43]	[3.57]
DES-KNN	4.11	3.89	**2.36**	[3.11]	[3.21]	4.32
F-DES-KNN	4.07	3.86	**2.50**	[3.14]	[3.36]	4.07
DESP	[3.57]	[4.18]	**2.79**	[3.23]	[3.27]	[3.96]
DES-RRC	[3.57]	[4.25]	**2.86**	[3.32]	[3.18]	[3.82]
META-DES	[4.00]	[3.96]	[2.91]	**2.86**	[3.34]	[3.93]

(c)

Algorithm	Bagging	RM	RM100	SM	SM100	RB
STATIC	5.54	**2.46**	[3.29]	[3.43]	3.75	[2.54]
RANK	5.59	**2.30**	[3.14]	[3.14]	3.64	[3.18]
LCA	5.32	**2.04**	3.57	3.29	3.29	3.50
F-LCA	5.14	**2.18**	3.64	[3.21]	[3.20]	3.62
MCB	5.48	**2.32**	[3.12]	[3.00]	4.07	[3.00]
F-MCB	5.71	**2.14**	[2.79]	[3.07]	4.21	[3.07]
KNE	5.75	**2.29**	[2.79]	3.64	3.82	[2.71]
F-KNE	5.75	**2.30**	[2.79]	3.59	3.86	[2.71]
KNU	5.61	[2.46]	3.39	3.57	3.86	**2.11**
F-KNU	5.75	[2.43]	3.45	3.34	3.93	**2.11**
DES-KNN	5.75	**2.18**	[3.00]	3.32	4.04	[2.71]
F-DES-KNN	5.82	**2.32**	[3.02]	[3.32]	3.84	[2.68]
DESP	5.61	[2.64]	[3.30]	[3.25]	3.89	**2.30**
DES-RRC	5.54	**2.43**	[3.39]	[3.32]	3.79	2.54]
META-DES	5.82	[2.57]	[3.29]	3.46	3.64	**2.21**
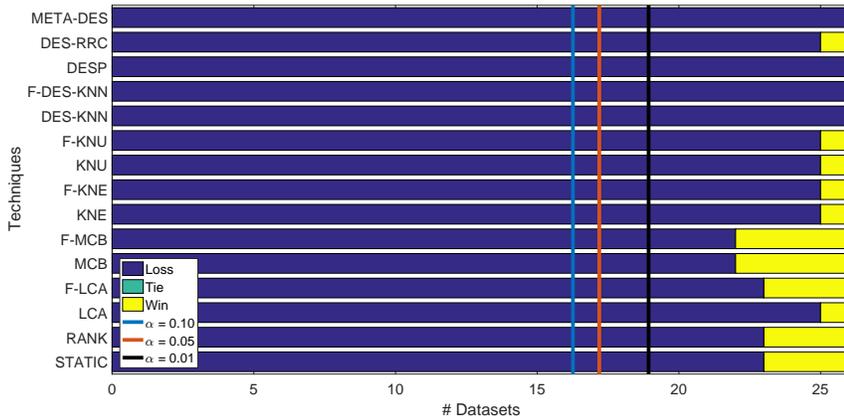
(a) AUC
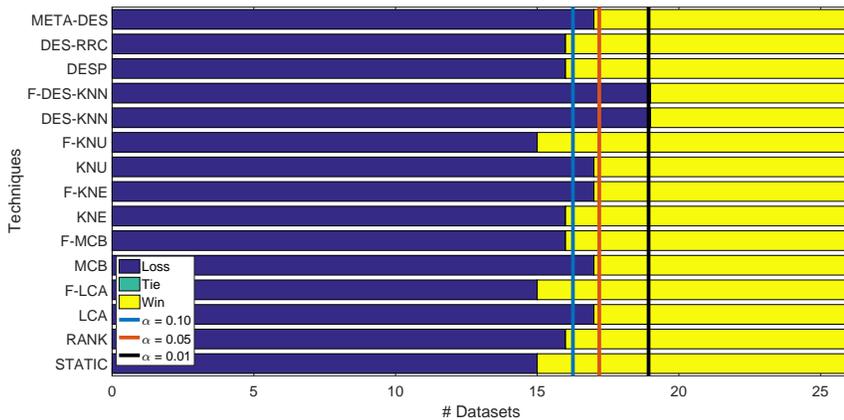
(b) F-measure
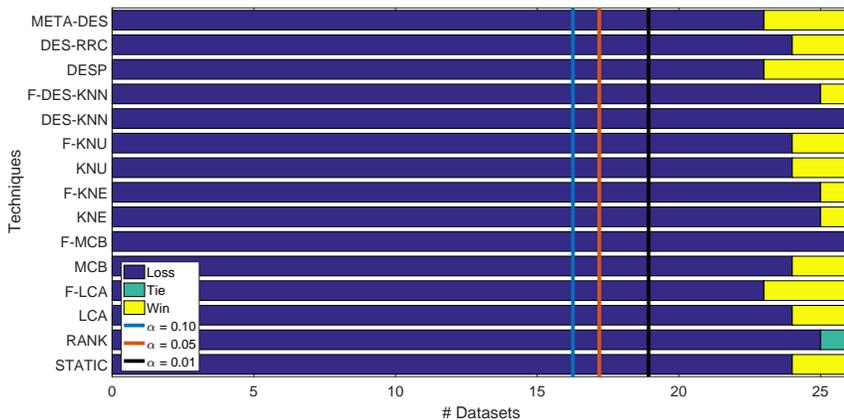
(c) G-mean

Fig. 4: Sign test computed over the wins, ties and losses. The vertical lines represents the critical value for at a significance level $\alpha = \{0.1, 0.05, 0.01\}$.

4.4. *Dynamic selection vs static combination*

In this experiment we compare the performance of the dynamic selection approaches versus static ones. For each technique, the best performing data preprocessing technique is selected (i.e., best result from each row of Table 3). Then, new average ranks are calculated for these methods. Table 4 presents the average rank of the top techniques according to each metric.

Based on the average ranks, we can see that all DES techniques presented a lower average rank when compared to that of the static combination for the three performance measures. In fact, the performance of most DES techniques was significantly better than the static combination's for both the AUC and the G-mean. Hence, DES techniques are suitable for dealing with multi-class imbalance. The DCS techniques, however, obtained a greater average rank in comparison with the static combination for the F-measure and G-mean, suggesting that they may not be fit for handling multi-class imbalanced problems. Furthermore, the DS techniques within the FIRE-DES framework yielded, in almost every case, a greater average rank than the same technique by itself, further suggesting that the FIRE-DES may be unsuitable for multi-class imbalanced classification.

Table 4: Average ranks for the best ensemble methods. (a) According to AUC, (b) according to F-measure and (c) according to G-mean. Results that are statistically equivalent to the best one are in brackets.
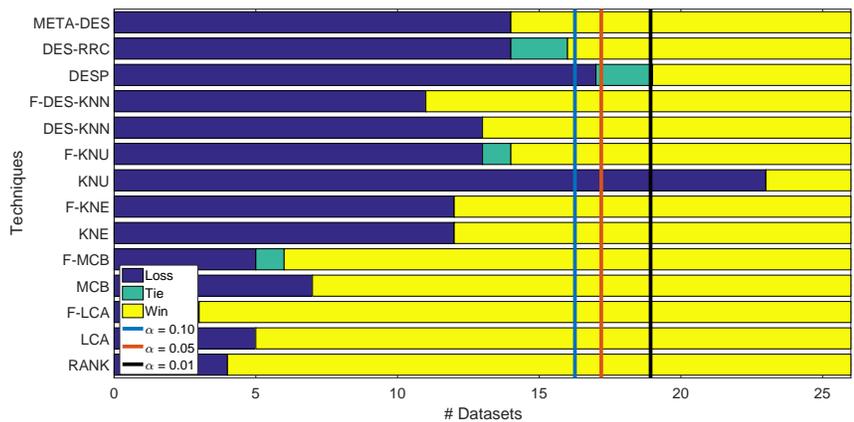
(a) AUC		(b) F-measure		(c) G-mean	
Methods	Rank	Method	Rank	Method	Rank
Ba-RB+DESP	**2.64**	Ba-RM100+KNU	**4.73**	Ba-RB+KNU	**4.95**
Ba-RB+DES-RRC	[3.16]	Ba-RM100+DES-RRC	[5.30]	Ba-RB+META-DES	[5.20]
Ba-RB+META-DES	[3.25]	Ba-RM100+DES-KNN	[5.54]	Ba-RB+F-KNU	[5.41]
Ba-RM+DES-KNN	[4.73]	Ba-SM+META-DES	[6.05]	Ba-RB+DESP	[6.34]
Ba-RB+KNU	[4.80]	Ba-RM100+DESP	[6.18]	Ba-RM+F-DES-KNN	[6.52]
Ba-RM+F-DES-KNN	[5.02]	Ba-RM100+F-DES-KNN	[6.29]	Ba-RM+DES-KNN	[6.54]
Ba-RB+F-KNU	6.25	Ba-RM100+KNE	[6.43]	Ba-RM+KNE	[6.79]
Ba-RM100+KNE	8.30	Ba-RM100+F-KNE	[7.05]	Ba-RM+DES-RRC	[7.12]
Ba-RM100+F-KNE	8.73	Ba-RM100+F-KNU	[7.41]	Ba-RM+F-KNE	[7.21]
Ba-RM+F-MCB	11.80	Ba-RM100	[7.41]	Ba-RM	7.79
Ba-RM+MCB	11.84	Ba-RM100+MCB	10.54	Ba-RM+MCB	9.57
Ba-RM+LCA	11.96	Ba-RM100+F-MCB	10.86	Ba-RM+F-MCB	10.27
Ba-RM	12.14	Ba-RM100+RANK	11.36	Ba-RM+LCA	11.52
Ba-RM+F-LCA	12.50	Ba-SM100+LCA	12.02	Ba-RM+RANK	12.14
Ba-RM+RANK	12.86	Ba-SM100+F-LCA	12.84	Ba-RM+F-LCA	12.64

We also performed a pairwise comparison between the best data preprocessing for each DS technique and the best data preprocessing for the static combination, that is, between each row from Tables 4a, 4b and 4c and the row in the same table
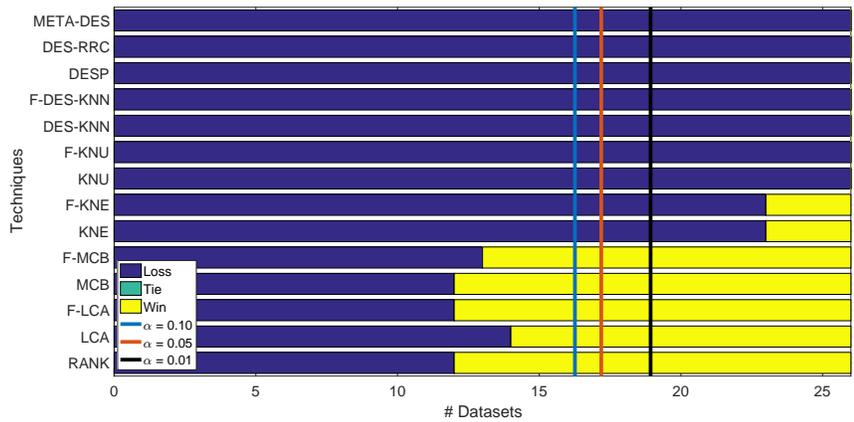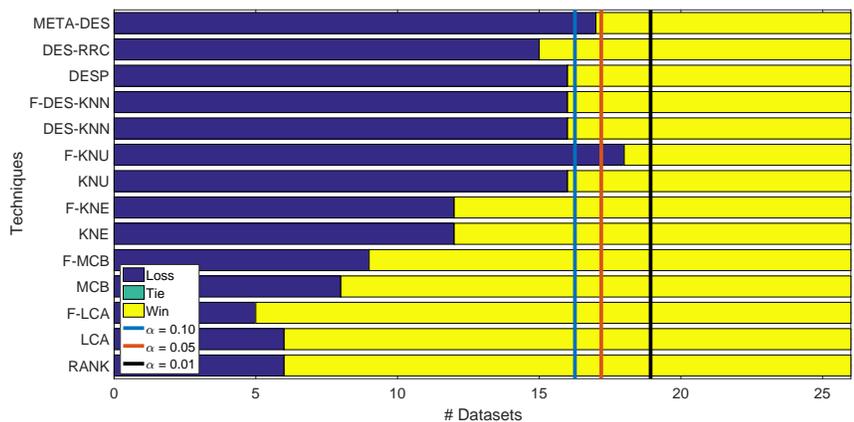
(a) AUC

(b) F-measure

(c) G-mean

Fig. 5: Sign test computed over the wins, ties and losses. The vertical lines represents the critical value for at a significance level $\alpha = \{0.1, 0.05, 0.01\}$.

that corresponds to Bagging. Figure 5 shows the results of the Sign test for each metric.

It can be observed that, for the AUC, all DES techniques performed significantly better than Bagging for $\alpha = 0.01$. The DCS techniques, on the other hand, obtained fewer wins in comparison with the static combination, save for LCA. For the F-measure, only the KNU and the DESP yielded a significantly superior performance compared to Bagging. However, nearly half of the DES techniques obtained more wins than losses in the Sign test for this metric. As in the AUC case, the DCS techniques yielded a poorer performance in comparison with the static combination. For the G-mean, the F-KNU and the DESP obtained a significantly superior performance in comparison with Bagging for $\alpha = 0.01$ and $\alpha = 0.10$, respectively. Moreover, most DES techniques also yielded a greater number of wins for this metric, while all DCS techniques obtained much fewer wins in comparison with the static combination.

Overall, the DES techniques performed better than the static combination, further showing its suitability for handling multi-class imbalanced problems. The DCS techniques, however, obtained a poorer performance in comparison with Bagging for the three metrics, which suggests it may be unsuitable for multi-class imbalanced classification. As for the FIRE-DES, save for a few exceptions, the DS techniques within the framework obtained the same or fewer number of wins in comparison with the same technique on its own, which again suggests that it may not be fit for dealing with multi-class imbalance.

5. Conclusion

In this work, we conducted a study on dynamic ensemble selection and data preprocessing for solving the multi-class imbalanced problems. A total of fourteen dynamic selection schemes and five preprocessing techniques were evaluated in this experimental study.

Results obtained over 26 multi-class imbalanced problems demonstrate that the dynamic ensemble selection techniques studied obtained a better result than static ensembles based on AUC, F-measure and G-mean. Moreover, the use of data preprocessing significantly improves the performance of DS and static ensembles. In particular, the RAMO technique presented the best overall results. Furthermore, DS techniques seems to benefit more of data preprocessing methods since they are applied not only to generate the pool of classifiers but also to edit the distribution of the dynamic selection dataset.

Future works would involve the definition of new pre-processing techniques specific to deal with multi-class imbalance as well as the definition of cost-sensitive dynamic selection techniques to handle multi-class imbalanced problems.

Acknowledgments

References

1. L. Abdi and S. Hashemi, To combat multi-class imbalanced problems by means of over-sampling techniques, *Transactions on Knowledge and Data Engineering* **28**(1) (2016) 238–251.
2. R. Akbani, S. Kwek and N. Japkowicz, Applying support vector machines to imbalanced datasets, in *European conference on machine learning* (2004) pp. 39–50.
3. J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García and F. Sánchez, L. Herrera, 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) (2011) 255–287.
4. M. Alsaedi, T. Fevens, A. Krzyżak and l. Jeleń, Hybrid rusboost versus data sampling to address data imbalance for breast cancer cytological malignancy grading, in *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI) 2018* pp. 545–551.
5. T. Ateeq, M. N. Majeed, S. M. Anwar, M. Maqsood, Z.-u. Rehman, J. W. Lee, K. Muhammad, S. Wang, S. W. Baik and I. Mehmood, Ensemble-classifiers-assisted detection of cerebral microbleeds in brain mri, *Computers & Electrical Engineering* (2018).
6. R. Barandela, R. Valdovinos and J. Sánchez, New applications of ensembles of classifiers, *Pattern Analysis & Applications* **6**(3) (2003) 245–256.
7. P. Branco, L. Torgo and R. P. Ribeiro, A survey of predictive modeling on imbalanced domains, *ACM Computing Surveys* **49**(2) (2016) 31:1–31:50.
8. L. Breiman, J. Friedman, R. Olshen and C. Stone, *Classification and Regression Trees* (Taylor & Francis, Monterey, CA, 1984).
9. A. S. Britto, R. Sabourin and L. E. S. Oliveira, Dynamic selection of classifiers - a comprehensive review, *Pattern Recognition* **47**(11) (2014) 3665–3680.
10. N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, Smote: Synthetic minority over-sampling technique, *Journal of Artificial Intelligence Research* **16**(1) (2002) 321–357.
11. S. Chen, H. He and E. A. Garcia, RAMOBoost: ranked minority oversampling in boosting, *Transactions on Neural Networks* **21**(10) (2010) 1624–1642.
12. R. M. O. Cruz, L. G. Hafemann, R. Sabourin and G. D. C. Cavalcanti, DESlib: A dynamic ensemble selection library in python, *arXiv:1802.04967* (2018).
13. R. M. O. Cruz, R. Sabourin and G. D. C. Cavalcanti, Dynamic classifier selec-

tion: Recent advances and perspectives, *Information Fusion* **41** (2018) 195 – 216.

14. R. M. O. Cruz, R. Sabourin and G. D. C. Cavalcanti, Prototype selection for dynamic classifier and ensemble selection, *Neural Computing and Applications* **29**(2) (2018) 447–457.

15. R. M. O. Cruz, R. Sabourin, G. D. C. Cavalcanti and T. I. Ren, META-DES: a dynamic ensemble selection framework using meta-learning, *Pattern Recognition* **48**(5) (2015) 1925 – 1935.

16. R. M. Cruz, R. Sabourin and G. D. Cavalcanti, On meta-learning for dynamic ensemble selection, in *International Conference on Pattern Recognition (ICPR)* (2014) pp. 1230–1235.

17. C. Dietrich, G. Palm and F. Schwenker, Decision templates for the classification of bioacoustic time series, *Information Fusion* **4**(2) (2003) 101–109.

18. J. F. Díez-Pastor, J. J. Rodríguez, C. I. García-Osorio and L. I. Kuncheva, Random balance: Ensembles of variable priors classifiers for imbalanced data, *Knowledge-Based Systems* **85** (2015) 96 – 111.

19. J. F. Díez-Pastor, J. J. Rodríguez, C. I. García-Osorio and L. I. Kuncheva, Diversity techniques improve the performance of the best imbalance learning ensembles, *Information Sciences* **325** (2015) 98 – 117.

20. A. Fernández, S. García, M. J. del Jesus and F. Herrera, A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets, *Fuzzy Sets and Systems* **159**(18) (2008) 2378 – 2398.

21. A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk and F. Herrera, *Learning from Imbalanced Data Sets* (Springer International Publishing, 2018).

22. A. Fernández, V. López, M. Galar, M. J. Del Jesus and F. Herrera, Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches, *Knowledge-Based Systems* **42** (2013) 97–110.

23. F. Fernández-Navarro, C. Hervás-Martínez and P. A. Gutiérrez, A dynamic over-sampling procedure based on sensitivity for multi-class problems, *Pattern Recognition* **44**(8) (2011) 1821–1833.

24. H. Finner, On a monotonicity problem in step-down multiple test procedures, *Journal of the American Statistical Association* **88**(423) (1993) 920–923.

25. M. Galar, A. Fernandez, E. Barrenechea, H. Bustince and F. Herrera, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **42**(4) (2012) 463–484.

26. S. García, Z.-L. Zhang, A. Altalhi, S. Alshomrani and F. Herrera, Dynamic ensemble selection for multi-class imbalanced datasets, *Information Sciences* **445** (2018) 22–37.

27. V. García, J. S. Sánchez and R. A. Mollineda, On the effectiveness of preprocessing methods when dealing with different levels of class imbalance, *Knowledge-*

Based Systems **25** (February 2012) 13–21.

28. G. Giacinto and F. Roli, Design of effective neural network ensembles for image classification purposes, *Image and Vision Computing* **19**(9-10) (2001) 699–707.

29. G. Giacinto and F. Roli, Dynamic classifier selection based on multiple classifier behaviour, *Pattern Recognition* **34** (2001) 1879–1881.

30. C. Gutierrez, T. Kim, R. Della Corte, J. Avery, M. Cinque, D. Goldwasser and S. Bagchi, Learning from the ones that got away: Detecting new forms of phishing attacks, *IEEE Transactions on Dependable and Secure Computing* (2018).

31. H. Han, W.-Y. Wang and B.-H. Mao, Borderline-smote: A new over-sampling method in imbalanced data sets learning, in *International Conference on Advances in Intelligent Computing (ICIC)* (2005) pp. 878–887.

32. D. J. Hand and R. J. Till, A simple generalisation of the area under the roc curve for multiple class classification problems, *Machine Learning* **45**(2) (2001) 171–186.

33. H. He, Y. Bai, E. Garcia and S. Li, Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in *International Joint Conference on Neural Networks* (2008) pp. 1322–1328.

34. H. He and E. Garcia, Learning from imbalanced data, *Trans. on Knowledge and Data Engineering* **21**(9) (2009) 1263–1284.

35. A. Ko, R. Sabourin and J. A. Britto, From dynamic classifier selection to dynamic ensemble selection, *Pattern Recognition* **41**(5) (2008) 1718–1731.

36. B. Krawczyk, Learning from imbalanced data: open challenges and future directions, *Progress in Artificial Intelligence* **5**(4) (2016) 221–232.

37. B. Krawczyk, A. Cano and M. Woźniak, Selecting local ensembles for multiclass imbalanced data classification, in *2018 International Joint Conference on Neural Networks (IJCNN)* (2018) pp. 1–8.

38. 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 networks* **21**(2-3) (2008) 427–436.

39. D. V. Oliveira, G. D. Cavalcanti and R. Sabourin, Online pruning of base classifiers for dynamic ensemble selection, *Pattern Recognition* **72** (2017) 44 – 58.

40. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12** (2011) 2825–2830.

41. L. Piras and G. Giacinto, Synthetic pattern generation for imbalanced learning in image retrieval, *Pattern Recognition Letters* **33**(16) (2012) 2198–2205.

42. R. C. Prati, G. E. Batista and D. F. Silva, Class imbalance revisited: a new

experimental setup to assess the performance of treatment methods, *Knowledge and Information Systems* **45**(1) (2015) 247–270.

43. A. Roy, R. M. Cruz, R. Sabourin and G. D. Cavalcanti, A study on combining dynamic selection and data preprocessing for imbalance learning, *Neurocomputing* **286** (2018) 179–192.

44. M. Sabourin, A. Mitiche, D. Thomas and G. Nagy, Classifier combination for hand-printed digit recognition, in *International Conference on Document Analysis and Recognition* (1993) pp. 163–166.

45. C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse and A. Napolitano, RUSBoost: a hybrid approach to alleviating class imbalance, *Transactions on Systems Man Cybernetics Part A* **40** (January 2010) 185–197.

46. C. A. Shipp and L. I. Kuncheva, Relationships between combination methods and measures of diversity in combining classifiers, *Information Fusion* **3**(2) (2002) 135–148.

47. R. G. Soares, A. Santana, A. M. Canuto and M. C. P. de Souto, Using accuracy and diversity to select classifiers to build ensembles, in *International Joint Conference on Neural Networks* (2006) pp. 1310–1316.

48. Q. Song, Y. Guo and M. Shepperd, A comprehensive investigation of the role of imbalanced learning for software defect prediction, *IEEE Transactions on Software Engineering* (2018) 1–1.

49. V. Tayanov, A. Krzyżak and C. Suen, Learning classifier predictions: is this advantageous?, in *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI) 2018* pp. 541–544.

50. W. Wei, J. Li, L. Cao, Y. Ou and J. Chen, Effective detection of sophisticated online banking fraud on extremely imbalanced data, *World Wide Web* **16**(4) (2013) 449–475.

51. T. Woloszynski and M. Kurzynski, A measure of competence based on randomized reference classifier for dynamic ensemble selection, *International Conference on Pattern Recognition* (2010) 4194–4197.

52. T. Woloszynski and M. Kurzynski, A probabilistic model of classifier competence for dynamic ensemble selection, *Pattern Recognition* **44**(10-11) (2011) 2656–2668.

53. T. Woloszynski, M. Kurzynski, P. Podsiadlo and G. W. Stachowiak, A measure of competence based on random classification for dynamic ensemble selection, *Information Fusion* **13**(3) (2012) 207–213.

54. K. Woods, W. P. Kegelmeyer and K. Bowyer, Combination of multiple classifiers using local accuracy estimates, *Transactions on Pattern Analysis and Machine Intelligence* **19**(4) (1997) 405–410.

55. W. Yang and S. Krishnan, Sound event detection in real-life audio using joint spectral and temporal features, *Signal, Image and Video Processing* (2018) 1–8.