Pós-Graduação em Ciência da Computação

DAYVID VICTOR RODRIGUES DE OLIVEIRA

FIRE-DES AND FIRE-DES++: DYNAMIC CLASSIFIER ENSEMBLE SELECTION FRAMEWORKS

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2018

Dayvid Victor Rodrigues de Oliveira

FIRE-DES and FIRE-DES++:
Dynamic Classifier Ensemble Selection Frameworks

Trabalho apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR: Prof. Dr. George Darmiton da Cunha Cavalcanti
COORIENTADOR: Prof. Dr. Robert Sabourin

RECIFE
2018

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

O48f Oliveira, Dayvid Victor Rodrigues de
FIRE-DES and FIRE-DES++: dynamic classifier ensemble selection frameworks / Dayvid Victor Rodrigues de Oliveira. – 2018.
162 f.: il., fig., tab.

Orientador: George Darmiton da Cunha Cavalcante.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2018.
Inclui referências e apêndices.

1. Inteligência artificial. 2. Seleção dinâmica de classificadores. I. Cavalcante, George Darmiton da Cunha (orientador). II. Título.

006.3 CDD (23. ed.) UFPE- MEI 2018-078

Dayvid Victor Rodrigues de Oliveira

FIRE-DES and FIRE-DES++: Dynamic Classifier Ensemble Selection Frameworks

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutora em Ciência da Computação

Aprovado em: 09/03/2018.

_____

Orientador: Prof. Dr. George Darmiton da Cunha Cavalcante

BANCA EXAMINADORA

_____

Prof. Dr. Ricardo Bastos Cavalcante Prudêncio
Centro de Informática /UFPE

_____

Prof. Dr. Adriano Lorena Inacio de Oliveira
Centro de Informática /UFPE

_____

Prof. Dr. Renato Vimieiro
Centro de Informática / UFPE

_____

Prof. Dr. Luiz Eduardo Soares de Oliveira
Departamento de Informática / UFPR

_____

Prof. Dr. Alceu de Souza Britto Junior
Programa de Pós-Graduação em Informática Aplicada/PUC-PR

I dedicate this work to my daughter, Isabela.

Acknowledgements

First of all, I express my gratitude to the Most High. *"What can I give back to God for all the blessings He's poured out on me?"* (Psalms 116). I also thank my family - my wife Nubia, my children Pedro and Isabela, and my parents Israel and Mirian - for the encouragement and support throughout these years.

I also express my gratitude to my supervisors, Prof. George Cavalcanti and Prof. Robert Sabourin, that thought me so much.

I thank the members of my thesis committee: Prof. Ricardo Prudêncio, Prof. Renato Vimieiro, Prof. Adriano Oliveira, Prof. Luiz Oliveira, and Prof. Alceu Junior for evaluating this thesis and providing constructive comments.

Finally, I want to thank some people that encouraged or inspired me in different moments over the last 4 years: Raisa Mirella, Diogo Pimentel, Paulo Henrique, Guillaume Lemaître, Ivo Lima, Thiago José, Roberto Pinheiro, Prof. Tsang Ing Ren, Prof. Ricardo Silva, Josh Nankivel and Allen Goetsch. A special thanks to Rafael Oliveira and Thyago Porpino that collaborated so much with my work: it was a pleasure to work with such talented people.

To God belong wisdom and power; counsel and understanding are his.
(Job, Hebrew Bible)

Abstract

Dynamic Ensemble Selection (DES) techniques aim to select only the most competent classifiers for the classification of each test sample. The key issue in DES is how to estimate the competence of classifiers for the classification of each new test sample. Most DES techniques estimate the competence of classifiers using a given criterion over the set of nearest neighbors of the test sample in the validation set, these nearest neighbors compose the region of competence. Despite achieving interesting results in several applications, DES techniques can select locally incompetent classifiers for the classification of a test sample because they do not consider different types of regions of competence: safe regions (neighborhood with homogeneous class labels), indecision regions (neighborhood surrounding classes boundaries), and noisy regions (with at least one sample from one class in an area of another class). In this work, we propose two dynamic selection frameworks for two-class problems: Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES) and Enhanced Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES++). Given a test sample, FIRE-DES decides if it is located in an indecision region and, if so, prunes the pool of classifiers, pre-selecting classifiers with decision boundaries crossing the region of competence of the test sample (if such classifier exists), then, it uses a DES technique to select the most competent classifiers from the pre-selected classifiers. FIRE-DES++ is an improvement of FIRE-DES that uses prototype selection to remove noise and reduce the overlap of classes in the validation set; defines the region of competence using an equal number of samples from each class, avoiding selecting a region of competence with samples of a single class; applies the pre-selection of classifiers crossing the region of competence of the test sample (if such classifiers exist); and finally, uses a DES technique to select the most competent classifiers from the pre-selected classifiers. Experiments were conducted using FIRE-DES and FIRE-DES++ with 8 different dynamic selection techniques on 40 classification datasets. Experimental results show that FIRE-DES and FIRE-DES++ increase the classification performance of all DES techniques considered in this work, with FIRE-DES++ outperforming FIRE-DES in 6 out of the 8 DES techniques, and FIRE-DES++ outperforming state-of-the-art DES techniques.

Keywords: Ensemble learning. Dynamic Classifier Selection. Dynamic Ensemble Selection.

Resumo

Técnicas de Seleção Dinâmica de Ensembles (DES) tem objetivo de selecionar os classificadores mais competentes para classificação de cada padrão de teste. A questão central em DES é como estimar a competência de classificadores para classificação de um padrão de teste. A maioria das técnicas de DES estimam a competência dos classificadores usando algum critério sobre o conjunto de vizinhos mais próximos do padrão de teste no conjunto de validação, estes vizinhos mais proximos compõem a região de competência. Apesar de alcançar resultados interessantes em várias aplicações, técnicas de DES podem selecionar classificadores localmente incompetentes para a classificação de um padrão de teste pelo fato de não considerar diferentes tipos de região de competência: regiões seguras (vizinhança com amostras de classes homogêneas), regiões de indecisão (vizinhança nas fronteiras de classes), regiões ruidosas (vizinhança com pelo menos um padrão de uma classe na área de uma outra classe). Neste trabalho, nós propomos dois frameworks de seleção dinâmica para problemas de classificação binários: *Frienemy Indecision Region Dynamic Ensemble Selection*(FIRE-DES), and *Enhanced Frienemy Indecision Region Dynamic Ensemble Selection* (FIRE-DES++). Dado um padrão de teste, FIRE-DES verifica se o padrão está localizado em uma região de indecisão, se sim, FIRE-DES poda o conjunto de classificadores, pre-selecionando classificadores com fronteiras de decisão cruzando a região de competência do padrão de teste (se houver classificadores que atendam este critério), e então, FIRE-DES usa uma técnica de DES para selecionar os classificadores mais competentes dentre os classificadores pre-selecionados. FIRE-DES++ é uma versão melhorada do FIRE-DES que usa seleção de protótipos para remover ruidos e reduzir a sobreposição de classes no conjunto de validação; define a região de competência usando um número equivalente de amostras de cada classe, evitando a seleção de uma região de competência composta de padrões pertencentes a uma única classe; aplica a pre-seleção de classificadores cruzando a região de competência do padrão de teste (se houver classificadores que atendam este critério); e finalmente, usa uma técnica de DES para selecionar os classificadores mais competentes dentre os classificadores pre-selecionados. Foram realizados experimentos usando FIRE-DES e FIRE-DES++ com 8 técnicas de DES diferentes em 40 bases de dados de classificação. Os resultados experimentais mostram que FIRE-DES e FIRE-DES++ aumentam a performance de classificação de todas as técnicas de DES consideradas neste trabalho, com FIRE-DES++ sendo superior a FIRE-DES com 6 dentre as 8 técnicas de DES utilizadas, e FIRE-DES++ sendo superior ao estado da arte em DES.

Palavras-chave: Aprendizagem baseada em Conjunto de Classificadores. Seleção Dinâmica de Classificadores. Seleção Dinâmica de Conjuntos de Classificadores.

List of Figures

List of Tables

Contents

1 INTRODUCTION

Multiple Classifiers Systems (MCS) combine classifiers expecting that several classifiers combined outperform any individual base classifier in classification performance. Recently, MCS have been widely presented as an alternative for increasing classification performance of several machine learning tasks and competitions (SINGH; SINGH, 2005) (CRUZ et al., 2013) (BATISTA; GRANGER; SABOURIN, 2010) (JAHRER; TÖSCHER; LEGENSTEIN, 2010) (BHATTACHARYYA et al., 2011) (TORRE et al., 2015) (PUURULA; READ; BIFET, 2014) (KOREN, 2009) (BELL; KOREN, 2007). MCS has also been widely studied as a promising approach for learning from challenging datasets such as imbalanced datasets (GALAR et al., 2012) and noisy datasets (SÁEZ et al., 2013).

MCS has three general phases (BRITTO; SABOURIN; OLIVEIRA, 2014): (1) Generation, in which the training set is used to generate a pool of classifiers; (2) Selection, in which a subset of the pool of classifiers is selected to perform the classification, we refer to this subset of classifiers as ensemble of classifiers; (3) Combination, in which the final decision is made based on the predictions of the classifiers.

Figure 1 shows the general phases of MCS (BRITTO; SABOURIN; OLIVEIRA, 2014): generation, selection and combination.

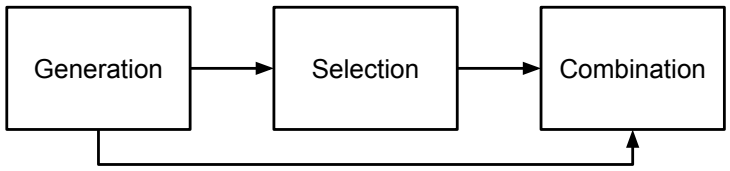


Figure 1 – MCS general phases: generation, selection and combination. (Adapted from (BRITTO; SABOURIN; OLIVEIRA, 2014)).

In the generation phase, a pool of classifiers is generated using the training set. The goal of ensemble generation techniques is to generate an ensemble of classifiers that is both accurate and diverse. Diverse classifiers are classifiers that complement each other in the sense that they are specialized in different local regions of the feature space (they misclassify different samples), and therefore, the combination of their prediction is likely to outperform any of them individually in classification accuracy (KUNCHEVA; WHITAKER, 2003) (TANG; SUGANTHAN; YAO, 2006). There are several ensemble generation methods proposed in the literature such as Bagging (BREIMAN, 1996), Random Subspace (HO, 1998), AdaBoost (FREUND; SCHAPIRE, 1995), Random Forest (BREIMAN, 2001), and Rotation Forest (RODRIGUEZ; KUNCHEVA; ALONSO, 2006).

In the selection phase, a subset of competent classifiers is selected from the pool of classifiers. A classifier is considered "competent" in a given local region if that local region is its area of expertise. The selection phase can be either static or dynamic. Static selection (GIACINTO; ROLI, 2001a) (PARTALAS; TSOUMAKAS; VLAHAVAS, 2008) (SANTOS; SABOURIN; MAUPIN, 2006) (LU et al., 2010) techniques select globaly competent classifiers in the training stage, and this selection is final for all test samples. Dynamic selection (CRUZ; SABOURIN; CAVALCANTI, 2018) (BRITTO; SABOURIN; OLIVEIRA, 2014) (KO; SABOURIN; JR, 2008) (GIACINTO; ROLI, 2000) (CAVALIN; SABOURIN; SUEN, 2013) (XIAO; HE, 2009a) (XIAO et al., 2010a) (XIAO et al., 2012) (CRUZ et al., 2015) (CRUZ; SABOURIN; CAVALCANTI, 2017b) (SANTOS; SABOURIN; MAUPIN, 2008) (GIACINTO; ROLI, 2001b) (SANTANA et al., 2006) (GIACINTO; ROLI, 1999) (WOODS; KEGELMEYER; BOWYER, 1997) (WOLOSZYNSKI; KURZYNSKI, 2011) (CRUZ; CAVALCANTI; REN, 2011) (CRUZ; SABOURIN; CAVALCANTI, 2017a) (CAVALIN; SABOURIN; SUEN, 2012) techniques select locally competent classifiers in the testing stage, and a different classifier (or ensemble of classifiers) is selected for each test sample. A dynamic selection solution can be: Dynamic Classifier Selection (DCS) (GIACINTO; ROLI, 2000), when only the best classifier is selected, or Dynamic Ensemble Selection (DES) (KO; SABOURIN; JR, 2008), when one or more competent classifiers are selected. Because DCS is a particular case of DES, in this work, we refer to all dynamic selection techniques as DES techniques.

In the combination phase (also known as integration phase or fusion phase), the predictions of classifiers for a given test sample are combined into a single prediction. There are several approaches for combining the predictions of classifiers such as stacking classifiers (DŽEROSKI; ŽENKO, 2004), mixtures of local experts (JACOBS et al., 1991), and, most commonly, applying rules such as majority vote, mean rule, median rule, min rule and max rule to probabilistic models (KITTLER et al., 1998).

1.1 Problem Statement

The rationale behind DES techniques is that different classifiers are competent (or "experts") in different local regions of the feature space (ZHU; WU; YANG, 2004), meaning no classifier is competent for the classification of all test samples, hence, the idea of selecting classifiers for the classification of each new test sample.

The crucial issue in DES is how to estimate the level of competence of a base classifier for the classification of a new test sample. Most DES methods evaluate the competence of base classifiers using some criterion on the region of competence of the test sample x_{query}, usually defined by as K nearest neighbors of x_{query} in the validation set $\mathcal{D}_{SEL}$. There are different criteria of competence estimation such as local accuracy

(WOODS; KEGELMEYER; BOWYER, 1997), probabilities (GIACINTO; ROLI, 1999), ranking (SABOURIN et al., 1993), diversity (SANTANA et al., 2006), ambiguity (SANTOS; SABOURIN; MAUPIN, 2008), oracle (KO; SABOURIN; JR, 2008), and meta-learning (CRUZ et al., 2015).

Figure 2 shows three types of regions of competence: safe regions, indecision regions, and noisy regions. Safe regions of competence are regions composed of safe samples (neighborhood with homogeneous class labels). Indecision regions of competence are regions composed of borderline samples (samples surrounding classes boundaries) from different classes. Noisy regions of competence are regions that contain noisy samples (samples from one class label in an area of another class label).

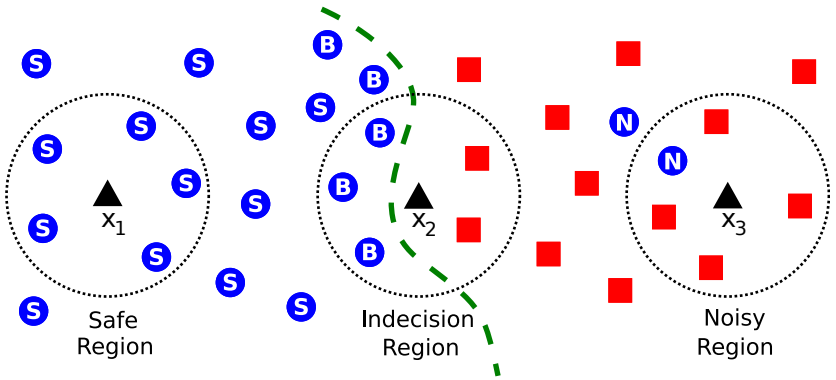


Figure 2 – Three test samples x_1, x_2, and x_3 with a safe region, indecision region, and noisy region, respectively. ▲'s are test samples, safe samples are labeled as S, borderline samples are labeled as B, and noisy samples labeled as N. The dashed line shows the decision boundary of classes, and the classes are represented by the markers "○" and "■" (Adapted from (OLIVEIRA; CAVALCANTI; SABOURIN, 2017) and (GARCÍA; LUENGO; HERRERA, 2015)).

A problem arises with the fact that DES techniques do not take into account the type of region of competence (that is, the type of samples that compose the region of competence) when estimating the competence of classifiers for the classification of a test sample.

In the example from Figure 2, the test sample x_1 is located in a safe region, so classifiers that classify all samples as being from the same class "○" are expected to be competent. The test sample x_2 is located in an indecision region, so, classifiers that classify all sample as being from the same class (either "○" or "■") are not as competent as classifiers that can distinguish samples from different classes in this region. The test sample x_3 is located in a noisy region, so, classifiers that "correctly classify noisy samples" can be the result of overfitting and are not as competent as classifiers that classify all samples (except the noisy sample) correctly.

The "No Free Lunch" theorem (CORNE; KNOWLES, 2003) stipulates that there is no algorithm that is better than all others over all possible classes of problems. Analogously,

we can say that there is no single criterion to select classifiers is better than all others over all possible types of regions of competence. However, DES techniques from the literature seem to ignore this.

1.2 Objectives

The main objective of this thesis is to develop a dynamic selection framework that takes into account different types of regions of competence (especially indecision regions, where most misclassifications occur) and the decision boundaries of the base classifiers (whether they classify all samples to the same class or not), improving the classification performance of DES techniques from the literature. Since ensemble learning has been considered a solution for dealing with imbalanced datasets (GALAR et al., 2012) an additional objective is that the proposed framework should be able to improve the performance of DES techniques on such datasets.

To acomplish those objectives, in this thesis we propose two DES frameworks: Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES) and Enhanced Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES++). FIRE-DES is a DES framework that, given a test sample, evaluates if the test sample is located in an indecision region, if so, it uses the Dynamic Frienemy Pruning (DFP) to pre-select classifiers with decision boundaries crossing the region of competence (if such classifiers exist), and then, uses a DES technique to perform the final selection of classifiers for the classification of that test sample.

FIRE-DES++ is an enhanced version of FIRE-DES that, given a test sample, it removes noise from the validation set (tackling the noisy regions), defines the region of competence of the test sample selecting an equal number of samples from each class from the validation set (tackling safe regions), and applies the DFP to pre-select classifiers with decision boundaries crossing the region of competence of the test sample (if such classifiers exist). Finally, FIRE-DES++ uses a DES technique to perform the final selection of classifiers for the classification of that test sample.

1.3 Contributions

The FIRE-DES and FIRE-DES++ frameworks are the main contributions of this thesis. These frameworks have led to the following publications (Chapter 3 and 4, respectively):

- OLIVEIRA, D. V.; CAVALCANTI, G. D.; SABOURIN, R. Online pruning of base classifiers for Dynamic Ensemble Selection. **Pattern Recognition**, [S.l.], v.72, p.44–58, 2017.

- OLIVEIRA, D. V.; CAVALCANTI, G. D.; SABOURIN, R.; CRUZ R. M. O.; FIRE-DES++: Enhanced online pruning of base classifiers for dynamic ensemble selection. **Submitted to Pattern Recognition**, 2018.

Three additional contributions were made in the course of this doctoral work: (1) Two dynamic selection techniques named K-Nearest Oracles Borderline (KNORA-B) and K-Nearest Oracles Borderline Imbalanced (KNORA-BI). (2) Two ensemble generation techniques named Iterative Classifier Selection Bagging (ICS-Bagging) and SMOTE Iterative Classifier Selection Bagging (SICS-Bagging). (3) One prototype generation technique named Evolutionary Adaptive Self-Generation Prototypes (EASGP). These additional contributions have led to the following publications (Appendix A, B, C, respectively):

- OLIVEIRA, D. V. et al. K-nearest oracles borderline dynamic classifier ensemble selection. **Accepted to IJCNN**, 2018.
- OLIVEIRA, D. V. et al. A bootstrap-based iterative selection for ensemble generation. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN). **Anais...** [S.l.: s.n.], 2015. p.1–7. (OLIVEIRA et al., 2015)
- OLIVEIRA, D. V. et al. Evolutionary Adaptive Self-Generating Prototypes for imbalanced datasets. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN). **Anais...** [S.l.: s.n.], 2015. p.1–8. (OLIVEIRA et al., 2015)

1.4 Organization

This thesis is organized into five chapters. Figure 3 presents this thesis overview. Boxes are chapters and appendixes, and arrows indicate the flow of the thsis (solid arrows indicate required dependencies, dashed arrows indicate suggested dependencies). Chapters inside the dotted region represent articles that were published or submitted during the development of this thesis, chapters inside the blue box are contribution on the dynamic selection topic, and chapters inside the red box are contributions on other machine learning topics.

The thesis starts with Chapter 1 (current chapter) presenting the introduction of the thesis. Chapter 2 presents a background of dynamic selection techniques for a better understanding of the following chapters.

Chapter 3 introduces the FIRE-DES framework. First, a brief introduction is presented, followed by a formalized problem statement. Then, the FIRE-DES framework is presented, and its effectiveness is demonstrated through experiments with benchmark datasets using 9 DES approaches. Finally, the remarks of FIRE-DES are listed as the

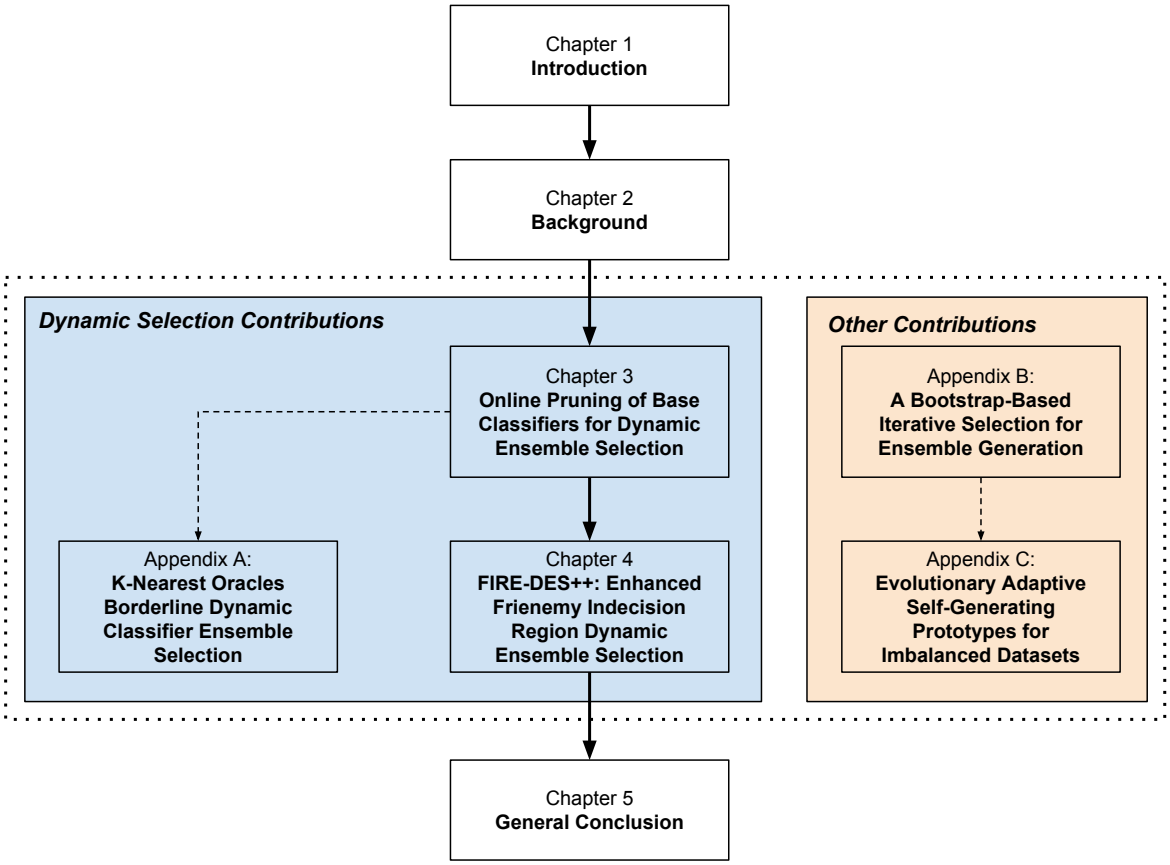


Figure 3 – Thesis overview. The boxes are chapters and appendixes, and the arrows are the flow of the thesis. Solid arrows indicate dependencies between the chapters and apendixes. Dashed arrows indicate suggested readings either for a better comprehension of the thesis. Chapters inside the dotted box represent articles that were published or submitted during the development of this thesis, chapters inside the blue box are contribution on the dynamic selection topic, and chapters inside the red box are contributions on other machine learning topics.

chapter is concluded. The contents of this chapter have been published in the Pattern Recognition journal (OLIVEIRA; CAVALCANTI; SABOURIN, 2017).

In Chapter 4, FIRE-DES++ is introduced. The chapter begins with a brief introduction, followed by a detailed problem statement presenting two drawbacks of FIRE-DES: (1) FIRE-DES mistakes noisy regions for indecision regions, and (2) FIRE-DES only applies the DFP to test samples classified as being located in indecision regions, and not to test samples classified as being located in safe regions (even if they are located close to the borders of classes). Then, FIRE-DES++ is presented, followed by a series of experiments that demonstrate that FIRE-DES++ outperforms FIRE-DES with statistical confidence. Finally, the remarks of FIRE-DES++ over FIRE-DES are listed as the chapter is concluded. The contents of this chapter have been submitted to the Pattern Recognition journal.

In Chapter 5, a general conclusion and future works are presented.

In addition, Appendix A, B, and C presents other contributions of this doctoral work. Appendix A presents two new dynamic selection techniques: K-Nearest Oracles Borderline (KNBORA-B) and K-Nearest Oracles Borderline Imbalanced (KNORA-BI). The problem statement from FIRE-DES (Chapter 3) derived from the study in this appendix. The contents of this appendix have been accepted for publication in the Internacional Joint Conference of Neural Networks (IJCNN) 2018.

Appendix B presents two new ensemble generation techniques: Iterative Classifier Selection Bagging (ICS-Bagging) and SMOTE Iterative Classifier Selection Bagging (SICS-Bagging). ICS-Bagging and SICS-Bagging generate a pool of classifiers by generating several classifiers for each new classifier to be added to the pool, and including only the one that maximizes the global competence of the pool. Despite being an ensemble generation work, the techniques proposed in this appendix served as inspiration for selecting competent classifiers able to cope with the class imbalance problem. The contents of this appendix have been published in the proceedings of the Internacional Joint Conference of Neural Networks (IJCNN) 2015.

Appendix C presents a new Prototype Selection (PS) technique: Evolutionary Adaptive Self-Generating Prototype for Imbalanced Datasets (EASGP). EASGP is an "evolution" of the Adaptive Self-Generating Prototypes (ASGP) (OLIVEIRA et al., 2012) and Self-Generating Prototypes (SGP) (FAYED; HASHEM; ATIYA, 2007). Given a training set, EASGP generates a smaller training set by generating new samples from the existing ones. It uses the SGP as its initial process, followed by an *iterative merge* and *evolutionary pruning*. The contribution in this appendix is orthogonal to the main contributions of this thesis, however, we include Appendix C in this thesis as this work was the first of this doctoral work, serving as an inspiration for KNORA-BI (when keeping minority class samples) and later FIRE-DES. The contents of this appendix have been published in the proceedings of the Internacional Joint Conference of Neural Networks (IJCNN) 2015.

2 BACKGROUND

Dynamic selection (DS) consist in selecting base classifiers online for each new test sample to be classified. Given a test sample x_{query} and a pool of classifiers C, DS techniques find a subset of C composed of competent classifiers (C') for the classification of x_{query}.

The rationale behind DS techniques resides in the fact that different classifiers are competent (or "experts") in different local regions of the feature space (ZHU; WU; YANG, 2004). That is to say, no classifier is competent for the classification of all test samples, hence, selecting the most competent classifier (or an ensemble of the most competent classifiers) for the classification of each test sample is an interesting approach.

The crucial issue in DS is how to estimate the level of competence of a base classifier for the classification of a new test sample (CRUZ et al., 2015). Most DS techniques evaluate the competence of base classifiers using some criterion on the region of competence of the test sample.

In general, DS techniques have three main steps when classifying a test sample (CRUZ; SABOURIN; CAVALCANTI, 2018) (Figure 4):

1. **region of competence definition**, in which the local region off the test sample is defined for competence level estimation.
2. **selection criteria**, in which the competence level of the base classifiers are estimated.
3. **selection approach**, in which the classifier (DCS) or ensemble of classifiers (DES) are selected based on their competence level.

Each of the phases are detailed in the following sections.

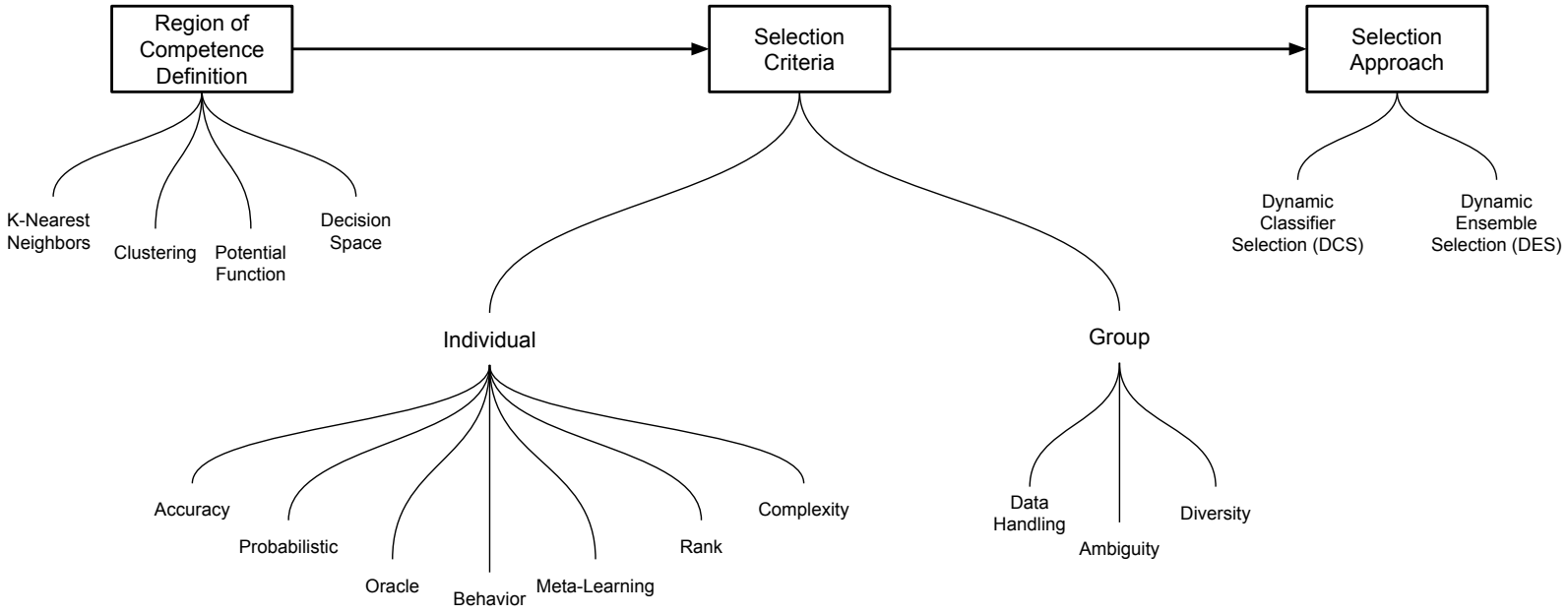


Figure 4 – Taxonomy of dynamic selection (from (CRUZ; SABOURIN; CAVALCANTI, 2018)).

2.1 Region of Competence Definition

The region of competence of a test sample is the local region of the feature space where the test sample is located that is used to evaluate the competence of classifiers for the classification of the test sample. Several studies (CRUZ; SABOURIN; CAVALCANTI, 2017a) (CRUZ; SABOURIN; CAVALCANTI, 2016) (ROY et al., 2018) demonstrate that DS techniques are very sensitive to the distribution of the region of competence, in fact, many works in DS focus better defining the region of competence as a way of improving the performance of DS techniques (CRUZ; CAVALCANTI; REN, 2011) (DIDACI; GIACINTO, 2004) (CRUZ; SABOURIN; CAVALCANTI, 2017a) (LIMA; LUDERMIR, 2013) (LIMA; SERGIO; LUDERMIR, 2014) (ROY et al., 2018).

Usually, DS techniques define the region of competence using either a K-Nearest Neighbors approach, a clustering approach, a potential function approach, or a decision space approach. All these approaches require a set of labeled samples named validation set (D_{SEL}), where the region of competence is defined.

2.1.1 K-Nearest Neighbors

DS techniques that use the KNN approach to define the region of competence rely on the assumption that a classifier that is competent for the classification of a test sample is also competent for the classification of samples that are similar to the test sample. For this reason, given a test sample x_{query}, these DS techniques define the region of competence (Ψ) as the K-nearest neighbors of x_{query} (samples that are most similar to x_{query}) in the validation set.

Different versions of the KNN algorithm have been used to define the region of competence. In (CRUZ; SABOURIN; CAVALCANTI, 2016), the authors used the Adaptive K-Nearest Neighbors (AKNN) (WANG; NESKOVIC; COOPER, 2007), which calculates the distance of a test sample to a sample x_a as the regular distance (i.e. Euclidean) divided by the distance between x_a and its nearest sample from a different class. Using the AKNN, the region of competence tends to be composed of samples in the classes centers, minimizing the probability of selecting a noisy sample.

In (MENDIALDUA et al., 2015), the authors used the K-Nearest Neighbors Equality (KNNE) (SIERRA et al., 2011) to define the region of competence. The KNNE is a KNN extension that treats the classes independently by selecting the K nearest neighbors from each class and uses a weighted majority vote (the closest to the test sample the more relevant the vote). The results in (MENDIALDUA et al., 2015) show that using the KNNE is an interesting approach when DS techniques consider the distances of the test sample to each sample in the region of competence when estimating the competence of classifiers, otherwise, using KNN achieved better results.

In (DIDACI; GIACINTO, 2004), Didaci and Giacinto investigated the choice of neighbourhood adaptive size, neighbourhood shape, and suitable distance metric and concluded that tunning parameters can lead to classification performance improvement of DS techniques.

2.1.2 Clustering

DS techniques that use the clustering approach (KUNCHEVA, 2000) (LIN et al., 2014) (SOARES et al., 2006) (SOUTO et al., 2008) (SANTANA et al., 2006) to define the region of competence rely on the assumption that a classifier that is competent for the classification of a cluster (over all samples in that cluster), is competent for the classification of test samples located in the local region defined by that cluster.

In training stage, these techniques apply a clustering technique to the validation set $\mathcal{D}_{SEL}$ and estimate the competence of base classifiers for each clusters (using the set of samples in each cluster). In testing stage, given a test sample x_{query}, the classifier (or ensemble of classifiers) that is competent in the nearest cluster of x_{query} is selected for the classification of x_{query}.

DS techniques that use the clustering approach to define the region of competence are usually faster than DS techniques that use the KNN approach. The reasons for such are: (1) instead of calculating the distance between the test sample and all samples in D_{SEL}, DS techniques that use the clustering approach calculate the distance between the test sample and the centroids of each cluster to define the region of competence (nearest cluster); and (2) instead of estimating the competence of the classifiers in the pool after defining the region of competence, the competence level of each classifier for all clusters are estimated during training stage.

2.1.3 Potential Function

DS techniques that use a potential function model define the region of competence as a weighted validation set ($\mathcal{D}^{W}_{SEL}$). The weights, influence of each sample in the validation set, are calculated using a potential function that gives the higher weights to samples nearest to the test sample, and lower weights to samples distant from the test sample. Most techniques use a gaussian potential function (Equation 2.1):

$$K(x_i, x_j) = exp(-dist(x_i, x_j)^2) \tag{2.1}$$

where K is the potential function, x_i and x_j are a pair of samples (a test sample and a sample from $\mathcal{D}_{SEL}$), and $dist$ is the Euclidean distance between x_i and x_j.

There are several DS techniques that use a potential function model to define the region of competence: Dynamic Ensemble Selection based on Kullback-Leibler divergence (DES-KL) (WOLOSZYNSKI et al., 2012), Random Reference Classifier (RRC) (WOLOSZYNSKI; KURZYNSKI, 2011), and DS techniques that use exponential and logarithmic functions (WOLOSZYNSKI; KURZYNSKI, 2009).

This approach for region of competence definition has been proven very effective, considering the RRC technique achieved the third best classification performance in (CRUZ; SABOURIN; CAVALCANTI, 2018). However, a major drawback for such techniques is that the computational complexity for selecting classifiers for the classification of each new test sample is high, since the competence of each base classifier in the pool of classifiers has to be esimated using the entire D_{SEL} instead of only a subset of D_{SEL} (either a cluster or a set of K nearest neighbors).

2.1.4 Decision Space

DS techniques that estimate the region of competence of classifiers using the decision space approach transform the test sample (x_{query}) and the validation set (D_{SEL}) into output profiles. An output profile is a vector composed of predictions of the classifiers in the pool of classifiers either (hard decision (HUANG; SUEN, 1995), or probabilities (CAVALIN; SABOURIN; SUEN, 2013) (CAVALIN; SABOURIN; SUEN, 2012) (BATISTA; GRANGER; SABOURIN, 2011)). Then, the samples in D_{SEL} with the most similar output profiles to the output profile of x_{query} are selected to compose the region of competence of x_{query}.

Examples of techniques that use the decision space approach to define the region of competence are: Multiple Classifier Behavior (MCB) (GIACINTO; ROLI, 2001b) K-Nearest Output Profile (KNOP) (CAVALIN; SABOURIN; SUEN, 2013), and META-DES (CRUZ et al., 2015).

2.2 Selection Criteria

There are several selection criteria to estimate the competence of classifiers for the classification of a test sample given its region of competence. In (BRITTO; SABOURIN; OLIVEIRA, 2014), the authors organized DS selection criteria in two groups: individual-based and group-based. Individial-based techniques use the individual performance of base classifiers to estimate their level of competence (the competence of one classifier depends only on its own performance).

2.2.1 Individual-Based

Individial-based techniques can be categorized into different subgroups according to the approach that is used to measure the competence of base classifiers such as local accuracy (WOODS; KEGELMEYER; BOWYER, 1997), probabilities (GIACINTO; ROLI, 1999), ranking (SABOURIN et al., 1993), oracle (KO; SABOURIN; JR, 2008), behavior (GIACINTO; ROLI, 2001b), complexity (BRUN et al., 2016), and meta-learning (CRUZ et al., 2015).

- **Accuracy**: Accuracy-based measure DS techniques estimate the accuracy of base classifiers in the region of competence of the test sample and selects the best (most locally accurate) classifier. (most locally accurate) classifier. This accuracy is the percentage of samples correctly classified (KO; SABOURIN; JR, 2008). Examples of accuracy based DS techniques are Overall Local Accuracy (OLA) and Local Class Accuracy (LCA) (WOODS; KEGELMEYER; BOWYER, 1997).

- **Probabilistic**: Probabilistic-based measure DS techniques are similar to accuracy-based measures techniques, but they use probabilistic representations to better select the most competent classifier (not only the most accurate, but the most confident among the most accurate).

- **Rank**: Rank-based measure DS techniques exploit a rank of the base classifiers in the pool of classifiers. One example of rank-based technique is the Modified Classifier Rank (MCR) (SABOURIN et al., 1993) (WOODS; KEGELMEYER; BOWYER, 1997) that creates a ranking of the base classifiers using the number of correctly classified samples in the region of competence of the test sample (the more samples, the best the ranking). The classifier with best ranking in the region of competence of a test sample is selected to classify the test sample.

- **Oracle**: Oracle-based measure DS techniques use the concept or *oracle* (the one who may provide wise concil) to select competent classifiers (BRITTO; SABOURIN; OLIVEIRA, 2014). These oracles can be either a classifier (KUNCHEVA; RODRIGUEZ, 2007) or the samples in the region of competence (KO; SABOURIN; JR, 2008).

- **Behavior**: Behavior-based measure DS techniques perform a behavior analysis using the classifier outputs as information sources (KO; SABOURIN; JR, 2008). The most well known example that uses the behavior-based measure is Multiple Classifier Behavior (MCB) (GIACINTO; ROLI, 2001b).

- **Complexity** Complexity-based measure DS techniques takes into account the data complexity to dynamically select competent classifiers for the classification of each

new test sample. In (BRUN et al., 2016), the authors proposed the Dynamic Selection on Complexity (DSOC), a complexity-based DS technique that selects a base classifier that (1) has high classification performance in the local region of the test sample; and (2) was trained using samples that presents a distribution with similar complexity measures (i.e. overlap between classes).

- **Meta-learning**: Meta-learning-based measure DS techniques treats the dynamic selection problem as a meta-problem. For each new test sample, these techniques use a meta-classifier to decide if, given a test sample (in the form of its meta-features) and its region of competence (their meta-features), it decides if a given classifier is competent for the classification of that test sample. Examples of such techniques are META-DES (CRUZ et al., 2015), META-DES.H (CRUZ; SABOURIN; CAVALCANTI, 2015b), and META-DES.Oracle (META-DES.O) (CRUZ; SABOURIN; CAVALCANTI, 2017b). Despite its complexity, meta-learning techniques achieved the best classification performance in (CRUZ; SABOURIN; CAVALCANTI, 2018), proving to be a promising approach for for DS.

2.2.2 Group-Based

Group-based techniques take into account the predictions of other classifiers in the pool when estimating the competence of a base classifier. The idea behind these techniques is not to select the most individually competent classifiers, but rather the most relevant classifiers (if a base classifier is relevant given the other pre-selected classifiers).

Group-based techniques can be categorized into three subgroups: diversity (SANTANA et al., 2006) (SOARES et al., 2006) (SANTOS; SABOURIN; MAUPIN, 2009), ambiguity (SANTOS; SABOURIN; MAUPIN, 2007), and data handling (XIAO et al., 2012),

- **Diversity**: Dynamic-based measure DS techniques aim to select a set of classifiers that are both accurate and diverse. In (SANTANA et al., 2006), the authors proposed the DS-KNN, in which the N_1 most accurate classifiers are selected, and then the N_2 most diverse classifiers are selected from the most accurate. This approach relies on the assumption that selecting classifiers that are diverse will lead to a good classification performance.

- **Ambiguity**: Ambiguity-based measure DS techniques are similar to diversity-based, but they use ambiguity metrics instead of diversity metrics. The idea is to select from a set of accurate classifiers an ensemble that minimizes the ambiguity in the ensemble (SANTOS; SABOURIN; MAUPIN, 2007) (SANTOS; SABOURIN; MAUPIN, 2008).

- **Data handling**: Data handling-based measure DS techniques use different group information. In (XIAO; HE, 2009b), the authors used an adaptive ensemble selection based on the group method of data handling theory (GMDH) (IVAKHNENKO, 1970). The idea is to select an ensemble with optimal complexity (weights) for the classification of each new test sample.

2.3 Selection Approach

There are two selection approaches for DS techniques: Dynamic Classifier Selection (DCS) or Dynamic Ensemble Selection (DES).

DCS techniques select the most competent classifier for the classification of each new test sample. Examples of DCS techniques are: Overall Local Accuracy (OLA) (WOODS; KEGELMEYER; BOWYER, 1997), Local Class Accuracy (LCA) (WOODS; KEGELMEYER; BOWYER, 1997), a Priori and a Posteriori (GIACINTO; ROLI, 1999), and Multiple Classifier Behavior (MCB) (GIACINTO; ROLI, 2001b).

DES techniques select an ensemble of one or more competent classifiers for the classification of each new test sample. Because they select several classifiers, DES techniques can select classifiers that complement each other, outperforming any individual single classifier of the pool. Examples of DES techniques are: K-Nearest Oracles Eliminate (KNORA-E) (KO; SABOURIN; JR, 2008), K-Nearest Oracles Union (KNORA-U) (KO; SABOURIN; JR, 2008), Randomized Reference Classifier (RRC) (WOLOSZYNSKI; KURZYNSKI, 2011) META-DES (CRUZ et al., 2015), and META-DES.Oracle (CRUZ; SABOURIN; CAVALCANTI, 2017b).

Because DCS is an specific case of DES (ensemble of one classifier), there are works that use the term DES to both DCS and DES (OLIVEIRA; CAVALCANTI; SABOURIN, 2017).

2.4 Dynamic Selection Techniques

There are several Dynamic Classifier Selection (DCS) and Dynamic Ensemble Selection (DES) techniques in the literature. This section presents what we believe to be the most relevant ones.

2.4.1 Overall Local Accuracy

The Overall Local Accuracy (OLA) (WOODS; KEGELMEYER; BOWYER, 1997) is a accuracy based DCS technique. OLA estimates the competence a base classifier from

the pool of classifiers as the accuracy of the classifier on the local region of the test sample. The best base classifier (most accurate) is selected.

Algorithm 1 presents the OLA method. Given a test sample x_{query}, OLA finds its K-neighbors (Line 1), evaluates the classification performance of all classifiers using these neighbors (Lines 2 - 4), and selects the best classifier (Lines 5 - 6).

Algorithm 1 OLA method

Require: $\mathcal{C}$: pool of classifiers
Require: $\mathcal{D}_{SEL}$: validation set
Require: x_{query}: test sample
Require: K: size of the neighborhood
1: $\Psi \leftarrow K$-nearest neighbors of x_{query} in $\mathcal{D}_{SEL}$
2: **for all** C_i in $\mathcal{C}$ **do**
3: $\quad OLA_i \leftarrow$ accuracy of C_i over Ψ
4: **end for**
5: $C_{best} \leftarrow C_{argmax_i(OLA_i)}$
6: **return** C_{best}

2.4.2 Local Class Accuracy

The Local Class Accuracy (LCA) (WOODS; KEGELMEYER; BOWYER, 1997) technique is an accuracy based DS technique very similar to OLA. Given a test sample, LCA estimates the competence of a classifier using the its accuracy in the region of competence, but considering only the samples that the classifier classifies as being from the same class that the classifier classified the test sample.

In other words, for all base classifiers, if a test sample is classified as ω, LCA estimates the local accuracy as the percentage of correctly classified samples in the local region that were also classified as being from the class ω by the classifier. The classifier with higher accuracy is selected.

Algorithm 2 presents the LCA method. For a given test sample x_{query}, LCA finds its K-neighbors Ψ (Line 1). For each classifier C_i, C_i is used to classify x_{query} (Line 3), all neighbors classified as the same class are selected Φ (Lines 5 - 9), and the classifier C_i is evaluated using Φ (Line 10). Finally, the best classifier is selected (Lines 12 - 13).

2.4.3 A Priori

The A Priori technique is a probabilistic based DCS technique that estimates the competence of a classifier as its accuracy within the local region of the test sample. This accuracy is calculated as the class prior probability of the classifier on the K-neighbors of the test sample in the validation set. This class prior probability is weighted by the inverse of the distance between each neighbor and the test sample.

Algorithm 2 LCA method

Require: $\mathcal{C}$: pool of classifiers
Require: $\mathcal{D}_{SEL}$: validation set
Require: x_{query}: test sample
Require: K: size of the neighborhood
1: $\Psi \leftarrow K$-nearest neighbors of x_{query} in $\mathcal{D}_{SEL}$
2: **for all** C_i in $\mathcal{C}$ **do**
3: $\omega \leftarrow C_i(x_{query})$
4: $\Phi \leftarrow 0$
5: **for all** $\Psi_i \in \Psi$ **do**
6: **if** $C_i(\Psi_i) = \omega$ **then**
7: $\Phi \leftarrow \Phi \cup \Psi_i$
8: **end if**
9: **end for**
10: $LCA_i \leftarrow$ accuracy of C_i over Φ
11: **end for**
12: $C_{best} \leftarrow C_{argmax_i(LCA_i)}$
13: **return** C_{best}

Consider x_{query} the test sample, Ψ the k-neighbors of x_{query}, and Ψ_j the j-th sample in Ψ, and ω a given class. The $\hat{p}(\omega|\Psi_j, c_i)$ is the measure of the classifier c_i accuracy for the test sample x_{query} based on its neighbor Ψ_j. Considering the N samples in Ψ, the best classifier is the one that maximizes the competence in Equation 2.2,

$$competence(c_i) = \frac{\sum_{j=1}^{N} \hat{p}(\omega|\Psi_j \in \omega, c_i) \times \delta_j}{\sum_{j=1}^{N} \delta_j} \tag{2.2}$$

where δ_j is the inverse of the Euclidean distance between the x_{query} and Ψ_j.

2.4.4 A Posteriori

The A Posteriori technique is a probabilistic based DCS technique that estimates the local accuracy of a given base classifier using the class posterior probabilities. Consider N samples in Ψ, and Ψ_j the j-th sample in Ψ. The best classifier with the output ω_l assigned to the test sample x_{query} is the one that maximizes the competence in Equation 2.3,

$$competence(c_i, \omega_l) = \frac{\sum_{\Psi_j \in \omega_l} \hat{p}(\omega_l|\Psi_j, c_i) \times \delta_j}{\sum_{j=1}^{N} \hat{p}(\omega_l|\Psi_j, c_i) \times \delta_j} \tag{2.3}$$

where δ_j is the inverse of the Euclidean distance between the x_{query} and Ψ_j.

2.4.5 K-Nearest Oracles Union (KNORA-U)

K-Nearest Oracles Union (KNORA-U) is an oracle-based DES technique that uses the K-nearest neighbors of the test sample as oracles, and selects all classifiers that correctly classify at least one of the oracles (the more oracles the classifier correctly classifies, the more votes the classifier has in the prediction of the test sample).

For the classification of a test sample x_{query}, KNORA-U selects all classifiers that correctly classify at least one of the samples in Ψ. After the selection, the classifiers votes are weighted, the more samples in Ψ a classifier correctly classify, the more votes it has.

Algorithm 3 presents the KNORA-U method.

Algorithm 3 KNORA-Union

Require: $\mathcal{C}$: pool of classifiers
Require: $\mathcal{D}_{SEL}$: validation set
Require: x_{query}: test sample
Require: K: size of the neighborhood
1: $EoC \leftarrow$ empty ensemble of classifiers
2: $\Psi \leftarrow K$-nearest neighbors of x_{query} in $\mathcal{D}_{SEL}$
3: **for all** ψ_i in Ψ **do**
4: **for all** c_i in $\mathcal{C}$ **do**
5: **if** c_i correctly classify ψ_i **then**
6: $EoC \leftarrow EoC \cup c_i$
7: **end if**
8: **end for**
9: **end for**
10: **if** EoC is empty **then**
11: $EoC \leftarrow \mathcal{C}$
12: **end if**
13: **return** EoC

KNORA-Union-W (KNORA-U-W) weights the samples in Ψ by the inverse of their distance to x_{query}, and each classifier vote is weighted, not by the number of samples correctly classified, but by the weighted sum of samples correctly classified by the classifier.

The experiments in (KO; SABOURIN; JR, 2008) showed that KNORA-U-W was not significantly better than KNORA-U.

2.4.6 K-Nearest Oracles Eliminate (KNORA-E)

K-Nearest Oracles Eliminate (KNORA-E) is an oracle-based DES technique that uses the K-nearest neighbors of the test sample as oracles, and selects all classifiers that

correctly classify all oracles (if no classifiers meet this criterion, one oracle is removed from the region of competence).

For the classification of a test sample x_{query}, KNORA-E finds its region of competence Ψ, and selects all classifiers that correctly classify **all** samples in Ψ. If no classifiers correctly classify all samples in Ψ, the value of K is decreased, until at least one classifier is selected. If K reaches 0, the most local accurate classifiers (using the original K) are selected.

Algorithm 4 presents the KNORA-E technique.

Algorithm 4 KNORA-Eliminate

Require: $\mathcal{C}$: pool of classifiers
Require: $\mathcal{D}_{SEL}$: validation set
Require: x_{query}: test sample
Require: K: size of the neighborhood

```
EoC ← ensemble of classifiers
while K > 0 and EoC is empty do
    Ψ ← K nearest neighbors of x_query in D_SEL
    for all c_i in C do
        if c_i correctly classify all samples in Ψ then
            EoC ← EoC ∪c_i
        end if
    end for
    if EoC is empty then
        K ← K − 1
    end if
end while
if EoC is empty then
    score_max ← accuracy of best classifier in C
    EoC ← all classifiers with accuracy score_max
end if
return EoC
```

KNORA-Eliminate-W (KNORA-E-W) weights samples in Ψ by the inverse of their distance to x_{query}. The use of weights in KNORA-E-W does not make any difference, unless the worst case scenario is reached ($K = 0$). In fact, in (KO; SABOURIN; JR, 2008), KNORA-E had exactly the same classification accuracy of KNORA-E-W for all datasets.

In (VRIESMANN et al., 2015), the authors proposed the Dynamic Ensemble Selection using Class and Overall Local Accuracies (DESCOLA), an oracle-based DES that combines the Overall Local Accuracy (OLA), Local Class Accuracy (LCA) and ambiguity

among classifiers to filter the pre-selection of classifiers done by the oracle. The OLA and LCA are computed on the neighborhood of the test sample in the validation set to filter out the classifiers selected by the KNORA. DESCOLA achieved interesting results, outperforming other DS techniques in 5 out of 8 datasets used in the experiments.

2.4.7 Multiple Classifier Behavior (MCB)

Multiple Classifier Behavior (MCB) (GIACINTO; ROLI, 2001b) is a behavior-based dynamic selection technique that estimates the competence of classifiers using a similarity function to measure the degree of similarity of the output profiles of all base classifiers.

Algorithm 5 presents the MCB technique. For a given test sample x_{query}, MCB finds the MCB_t vector composed by the predictions of the classifiers in the pool of classifiers for x_{query} (Line 1). Then, MCB finds Ψ the K-neighbors of x_{query} in the validation set $\mathcal{D}_{SEL}$ (Line 2). Now, MCB finds the MCB vector of all samples in Ψ, computes their similarities with MCB_t using Equation 2.4, and includes in Ψ^n all samples with similarity higher than a parameter *similarity_threshold* (Lines 4 - 10). Now, MCB estimates the overall local accuracy of the classifiers using Ψ^n (Lines 11 - 13) and selects the best classifier c_{best} (Line 14). If the classifier c_{best} is significantly better than all other classifiers in C, c_{best} is selected to perform the classification of x_{query}; otherwise, all classifiers in C are selected to perform the classification of x_{query}.

$$similarity(A, B) = \frac{1}{M} \times \sum_{j=1} MT(A_j, B_j) \tag{2.4}$$

where A and B are vectors, M is the size of the vectors A and B, and T is the $XNOR$ function (1 when $A_j = B_j$, otherwise 0).

2.4.8 DS-KNN

DS-KNN (SANTANA et al., 2006) is a diversity-based DES technique that combines the accuracy and diversity of the classifiers in the local region of the test sample to select an ensemble of classifiers for the classification of the test sample.

Algorithm 6 presents the DS-KNN algorithm. For the classification of a given test sample x_{query}, DS-KNN finds its K-nearest neighbors Ψ (Line 1) and estimates the accuracy of the classifiers in C on Ψ (Lines 2 - 4). Now, DS-KNN calculates the diversity of the classifiers in C in a pairwise fashion using the double fault diversity measure (TANG; SUGANTHAN; YAO, 2006) (Lines 5 - 10). Now, DS-KNN selects the N_1 most accurate classifiers (Line 11) and, among them, selects the N_2 most diverse classifiers (Line 12), and, finally, returns the final ensemble (Line 13).

Algorithm 5 MCB

Require: $\mathcal{C}$: pool of classifiers
Require: $\mathcal{D}_{SEL}$: validation set
Require: x_{query}: test sample
Require: K: size of the neighborhood

$MCB_t \leftarrow$ class labels assigned to x_{query} by all classifiers in C
$\Psi \leftarrow K$ nearest neighbors of x_{query} in $\mathcal{D}_{SEL}$
$\Psi^n \leftarrow \Phi$
for all $\Psi_i \in \Psi$ **do**
    $MCB_{\Psi_i} \leftarrow$ class labels assigned to Ψ_i by all classifiers in C
    $sim \leftarrow similarity(MCB_t, MCB_{\Psi_i}$
    if $sim \geq similarity\_threshold$ **then**
        $\Psi^n \leftarrow \Psi^n \cup \Psi_i$
    end if
end for
for all c_i in $\mathcal{C}$ **do**
    $OLA_i \leftarrow OLA(c_i, \Psi^n)$
end for
$c_{best} \leftarrow argmax(OLA)$, best classifier in C
if c_{best} is significantly better than the other classifiers **then**
    return c_best
end if
return C

Algorithm 6 DS-KNN

Require: $\mathcal{C}$: pool of classifiers
Require: $\mathcal{D}_{SEL}$: validation set
Require: x_{query}: test sample
Require: K: size of the neighborhood

$\Psi \leftarrow K$-nearest neighbors of x_{query} in $\mathcal{D}_{SEL}$
for all C_i in $\mathcal{C}$ **do**
    $OLA_i \leftarrow$ accuracy of C_i over Ψ
end for
for all c_i in $\mathcal{C}$ **do**
    for all c_j in $\mathcal{C}$ **do**
        $DIV_i \leftarrow DIV_i + double\_fault(c_i, c_j)$
    end for
    $DIV_i \leftarrow DIV_i/len(C)$
end for
$EoC \leftarrow N_1$ most accurate classifiers in C based on OLA
$EoC \leftarrow N_1$ most accurate classifiers in EoC based on DIV
return EoC

2.4.9 META-DES

In (CRUZ et al., 2015), the authors proposed the META-DES, a novel dynamic ensemble selection framework using meta-learning. META-DES considers the task of selecting competent classifiers as another classification problem (classifiers are classified as being either competent or incompetent for the classification of each new test sample).

META-DES is divided into three phases: (1) Overproduction, (2) Meta-Training, and (3) Generalization.

1. **Overproduction**: In this phase, META-DES generates a pool of classifiers using the training set. In (CRUZ et al., 2015), the authors used Bagging (BREIMAN, 1996), but any ensemble generates method can be used in this phase.

2. **Meta-Training**: This phase of META-DES is divided into three steps: selection, extraction, and training. In the selection step, META-DES selects from the training set samples in which the pool of classifiers has a low consensus degree (less than a threshold h_c), and divides the selected samples into meta-training and meta-validation. In the extraction step, META-DES extracts meta-features from the selected samples. In the training step, META-DES trains a selector λ using the meta-training partition of the selected samples (in (CRUZ et al., 2015), the authors used a Multi-Layer Perceptron neural network as the selector λ).

3. **Generalization**: In this phase, given a test sample x_{query}, its region of competence is extracted using the samples from the validation set and the meta-features are computed. The meta-features are passed to the selector λ and, for each classifier c_i in the pool of classifiers, the selector λ decides if c_i is competent for the classification of x_{query}. All competent classifiers are added to the final ensemble C' that classifies x_{query} using majority vote rule.

In (CRUZ; SABOURIN; CAVALCANTI, 2015a), the authors performed a deep analysis of the META-DES framework using linear classifiers, and demonstrated that META-DES can approximate complex non-linear distributions using few linear classifiers, outperforming all other DES techniques.

In (CRUZ; SABOURIN; CAVALCANTI, 2015b), the authors evaluated different models as meta-classifiers: Multi-Layer Perceptron (MLP), Random Forest, Support Vector Machines with Gaussian Kernel (SVM), and Naive Bayes (FERNÁNDEZ-DELGADO et al., 2014); and also proposed three versions of the META-DES framework: (1) META-DES.S, where all base classifiers with estimated competence higher than a threshold Υ are selected to submit a vote for the classification of the test sample. (2) META-DES.W, where all base classifiers submit weighted vote for the classification of the test sample,

where the weights are the estimated competence by the meta-classifier. (3) META-DES.H, combines the META-DES.S and META-DES.H, all classifiers with estimated competence higher than a threshold Υ submit a weighted vote for the classification of the test sample. The authors concluded that Naive Bayes was statistically better than MLP (meta-classifier used in (CRUZ et al., 2015)) and that META-DES.H was the best selection/voting scheme.

In (CRUZ; SABOURIN; CAVALCANTI, 2017b), the authors proposed an improvement to META-DES framework, named META-DES.Oracle framework. META-DES.Oracle uses 15 sets of meta-features and applies a meta-feature selection scheme using Binary Particle Swarm Optimization (BPSO) in order to optimize the performance of the meta-classifier. The BPSO algorithm uses a fitness function to be minimized defined as the difference between the competence estimated by the meta-classifier and the competence estimated by the Oracle. Experimental results demonstrated that META-DES.Oracle outperformed 10 state-of-the-art techniques, include META-DES.

To our knowledge, META-DES and META-DES.Oracle are the state-of-art in DS.

2.4.10 Randomized Reference Classifier (RRC)

The RRC technique is a randomized reference classifier based approach that decides if a given base classifier is significantly better than a random classifier. The competence level of a given classifier is estimated in two steps: (1) a gaussian potential function (Equation 2.1); and (2) a source of competence C_{src} estimated based on the RRC concept proposed in (WOLOSZYNSKI; KURZYNSKI, 2010).

The gaussian potential function is used to give higher relevance to samples in $\mathcal{D}_{SEL}$ nearest to the test sample, and lower relevance to samples furthest to the test sample. The competence level of each base classifier can be estimated as follows:

$$\delta = \sum_{x_i \in D_{SEL}} C_{src} K(x_{query}, x_i) \tag{2.5}$$

where x_{query} is the test sample, $\mathcal{D}_{SEL}$ is the validation set, C_{src} is the source of competence, and K is the potential function defined in Equation 2.1.

A given classifier is selected if it has estimated competence level δ higher than the competence level of a random classifier.

The RRC technique achieved interesting results. In fact, in (CRUZ; SABOURIN; CAVALCANTI, 2018), the authors evaluated the classification performance of 18 state-of-the-art DS techniques, and RRC achieved the third best performance, being outperformed only by META-DES and META-DES.Oracle.

2.4.11 Others

In (XIAO; HE, 2008), the authors introduced group method of data handling (GMDH) theory into MCS, proposing the Algorithm Adaptive Classifier Ensemble Selection based on GMDH (GAES). The GAES method uses GMDH to select an ensemble from the pool of classifiers and determine the combination weights. The multilayer algorithm in GMDH builds a structure of feedforward neural networks where each layer builds candidate models through the combination of two of the pre-selected models. Next, the external criterion is used to evaluate and select the best base classifiers for the next layer.

In (XIAO; HE, 2009b), the authors proposed the GMDH-based Dynamic Classifier Ensemble Selection (GDES), an extended GAES method that performs dynamic selection of classifiers. GDES selects a subset of classifiers from the pool of classifiers and determines the optimal weights for the classification of each new test sample. For a given test sample x_{query}, GDES finds the K-nearest neighbors of x_{query} in the validation set (the paper uses the training set), Ψ; uses all classifiers to classify the samples in Ψ; applies GAES to find the combination model; finally, uses the combination model to classify x_{query}.

In (XIAO et al., 2010b), the authors proposed the GMDH-based dynamic classifier ensemble selection according to accuracy and diversity (GDES-AD), which is a GDES that takes into consideration both accuracy and diversity of ensembles in the process of selecting classifiers. Experiments show that the GDES-AD is robust to noise when compared to other DES techniques due the fact that GDES-AD can better reduce the bias in classification error.

In (XIAO et al., 2012), the authors propose the Dynamic Classifier Ensemble Method for Imbalanced Data (DCEID), which is a DS technique that decides between selecting using the OLA or selecting using GDES (XIAO; HE, 2009b) and uses a cost-sensitive evaluation criteria to increase classification performance on imbalanced datasets. The authors in (XIAO et al., 2012) only evaluated DCEID using two customer classification datasets, and, to our knowledge, there is no benchmark study that considers this technique in its work.

There are other methods and categories outside of the scope of this work. An excellent review and categorization of DES methods can be found in (CRUZ; SABOURIN; CAVALCANTI, 2018) and (BRITTO; SABOURIN; OLIVEIRA, 2014).

3 ONLINE PRUNING OF BASE CLASSIFIERS FOR DYNAMIC ENSEMBLE SELECTION

Online Pruning of Base Classifiers for Dynamic Ensemble Selection

Dayvid V. R. Oliveira [1], George D. C. Cavalcanti [1], and Robert Sabourin [2]

[1] Centro de Informática - Universidade Federal de Pernambuco, Brazil

[2] École de Technologie Supérieure - Université du Québec

Article Published in « Pattern Recognition » 2017

Abstract

Dynamic Ensemble Selection (DES) techniques aim to select only the most competent classifiers for the classification of each test sample. The key issue in DES is how to estimate the competence of classifiers for the classification of each new test sample. Most DES techniques estimate the competence of classifiers using a given criterion over the set of nearest neighbors of the test sample in the validation set, these nearest neighbors compose the region of competence. However, using local accuracy criteria alone on the region of competence is not sufficient to accurately estimate the competence of classifiers for the classification of all test samples. When the test sample is located in a region with borderline samples of different classes (indecision region), DES techniques can select classifiers with decision boundaries that do not cross the region of competence, assigning all samples in the region of competence to the same class. In this paper, we propose a dynamic selection framework for two-class problems that detects if a test sample is located in an indecision region and, if so, prunes the pool of classifiers, pre-selecting classifiers with decision boundaries crossing the region of competence of the test sample (if such classifiers exist). After that, the proposed framework uses a DES technique to select the most competent classifiers from the set of pre-selected classifiers. Experiments are conducted using the proposed framework with 9 different dynamic selection approaches on 40 classification datasets. Experimental results show that for all DES techniques used in the framework, the proposed framework outperforms DES in classification accuracy, demonstrating that our proposal significantly improves the classification performance of DES techniques, achieving statistically equivalent classification performance to the current state-of-the-art DES frameworks.

3.1 Introduction

Multiple Classifier Systems (MCS) (WOŹNIAK; GRAÑA; CORCHADO, 2014) combine classifiers expecting that several classifiers outperform any single base classifier in classification accuracy (KUNCHEVA, 2004) (DIETTERICH, 2000). Several studies present MCS as an alternative to increase classification accuracy in many pattern recognition tasks, such as image labeling (SINGH; SINGH, 2005), handwritten recognition (CRUZ et al., 2013), signature verification (BATISTA; GRANGER; SABOURIN, 2010), recommendation systems (JAHRER; TÖSCHER; LEGENSTEIN, 2010), banking (BHATTACHARYYA et al., 2011), and face recognition (TORRE et al., 2015).

MCS has three general phases (BRITTO; SABOURIN; OLIVEIRA, 2014): (1) generation, in which the training set is used to generate a pool of classifiers; (2) selection, in which a subset of the pool of classifiers is selected to perform the classification, we refer to this subset of classifiers as ensemble of classifiers; (3) combination (or integration), in which the final decision is made based on the predictions of the classifiers.

The selection phase can be either static or dynamic. In static selection, the selection of classifiers is performed in the training phase. In dynamic selection, the selection of classifiers is performed for each new test sample in the classification phase. Recent works have shown that dynamic selection techniques achieve higher classification accuracy than static selection techniques, especially on ill-defined problems (KO; SABOURIN; JR, 2008) (BRITTO; SABOURIN; OLIVEIRA, 2014) (CRUZ; SABOURIN; CAVALCANTI, 2015b). Dynamic selection can be either Dynamic Classifier Selection (DCS) (GIACINTO; ROLI, 2000) or Dynamic Ensemble Selection (DES) (KO; SABOURIN; JR, 2008). DCS selects one single classifier for the classification of each new test sample, and DES selects one or more classifiers for the classification of each new test sample. Since DCS is a specific case of DES, in this paper, we refer both as DES.

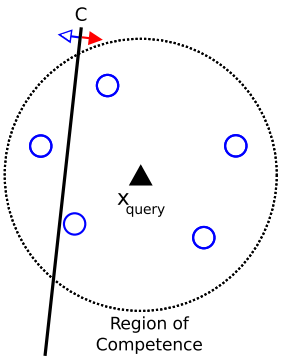
DES techniques rely on the assumption that different classifiers are competent ("experts") in different local regions of the feature space. For this reason, given a test sample x_{query} and a pool of classifiers C, dynamic selection techniques try to select the most competent classifier c, or an ensemble of competent classifiers C', $C' \in C$, for the classification of x_{query}. The key issue in DES is how to estimate the competence of a base classifier for the classification of a new test sample. Many DES techniques (WOODS; KEGELMEYER; BOWYER, 1997) (GIACINTO; ROLI, 1999) (GIACINTO; ROLI, 2001b) (SANTANA et al., 2006) (KO; SABOURIN; JR, 2008) estimate the competence of a classifier c for the classification of a test sample x_{query} using the accuracy of the classifier c on a set of labeled samples similar to x_{query}, obtained using the K-Nearest Neighbors (KNN) on the validation set $\mathcal{D}_{SEL}$. The set of K nearest neighbors of x_{query} in $\mathcal{D}_{SEL}$ is called the region of competence (Ψ) of x_{query}.

According to Britto et al. (BRITTO; SABOURIN; OLIVEIRA, 2014), most DES techniques use some criteria on the region of competence of the test samples to estimate the competence of base classifiers. META-DES (CRUZ et al., 2015), a recent DES framework published after the survey (BRITTO; SABOURIN; OLIVEIRA, 2014), achieved the highest DES classification performance to this date, and it also uses the region of competence to extract meta-features that are used to predict the competence of base classifiers. So, a crucial issue in the design of DES techniques is the definition of the region of competence. We expect that the better the region of competence, the higher the precision of DES systems.

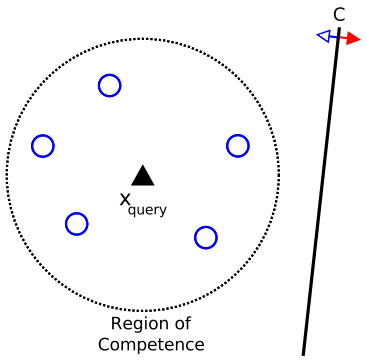
State-of-the-art DES techniques do not take into account the existence of different scenarios when estimating the competence of a classifier for the classification of a test sample using its region of competence. Given a test sample and a classifier, the test sample can be located in a region where almost all samples belong to the same class (safe region), or in a region where samples belong to more than one class (indecision region), and the classifier can have its decision boundary crossing or not crossing the region of competence of the test sample. Based on that, Figure 5 presents 4 scenarios, given a test sample x_{query}, a region of competence Ψ, and a classifier c: (I) x_{query} located in a safe region, and c's decision boundary crossing Ψ. (II) x_{query} located in a safe region, and c's decision boundary not crossing Ψ. (III) x_{query} located in an indecision region, and c's decision boundary crossing Ψ. (IV) x_{query} located in an indecision region, and c's decision boundary not crossing Ψ. Where ▲ is the test sample (x_{query}), the dotted circle delimits the region of competence, the markers "○" and "■" are samples from different classes, the continuous straight line is the decision boundary of the classifier c.

Scenarios I and II (Figure 5(a) and 5(b)) show a test sample ▲ located in a safe region (all samples in the region of competence of the test sample are from the class "○"). In Scenario I, the decision boundary of the classifier c crosses the region of competence, and c correctly classifies 20% of the samples in the region of competence. In Scenario II, the decision boundary of the classifier c does not cross the region of competence, and c correctly classifies 100% of the samples in the region of competence. A scenario with the classifier c not crossing the region of competence of the test sample ▲ and c misclassifying all samples in the region of competence of the test sample was not detailed because any accuracy based DES technique estimates the competence of such classifier as 0.0, and therefore, such classifier is never selected. This shows that, in safe regions, classifiers with decision boundaries not crossing the region of competence have higher competence estimation, meaning accuracy based DES techniques are sufficient to estimate the competence of base classifiers when the test sample is located in a safe region.

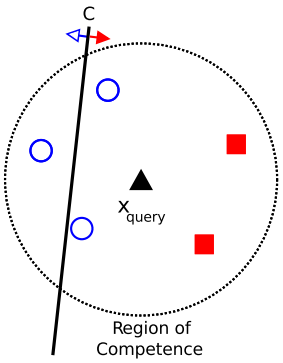
Scenarios III and IV (Figures 5(c) and 5(d)) show a test sample ▲ located in an indecision region (samples from different classes "○" and "■" in the region of competence



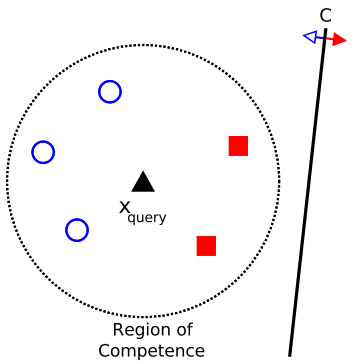
(a) Scenario I: test sample (▲) located in a safe region, and classifier c with decision boundary crossing the region of competence of ▲. The classifier c correctly classifies 20% of the samples in the region of competence.



(b) Scenario II: test sample (▲) located in a safe region, and classifier c with decision boundary not crossing the region of competence of ▲. The classifier c correctly classifies 100% of the samples in the region of competence.



(c) Scenario III: test sample (▲) located in an indecision region, and classifier c with decision boundary crossing the region of competence of ▲. The classifier c correctly classifies 60% of the samples in the region of competence.



(d) Scenario IV: test sample (▲) located in an indecision region, and classifier c with decision boundary not crossing the region of competence of ▲. The classifier c correctly classifies 60% of the samples in the region of competence.

Figure 5 – Four possible scenarios for a given test sample and a classifier: (a) Scenario I, where the test sample ▲ is located in a safe region, and the classifier c has its decision boundary crossing the region of competence of ▲. (b) Scenario II, where the test sample ▲ is located in a safe region, and the classifier c has its decision boundary not crossing the region of competence of ▲. (c) Scenario III, where the test sample ▲ is located in an indecision region, and the classifier c has its decision boundary crossing the region of competence of ▲. (d) Scenario IV, where the test sample ▲ is located in an indecision region, and the classifier c has its decision boundary not crossing the region of competence of ▲. In these scenarios, ▲ is the test sample, the continuous straight line is the decision boundary of the classifier c, the dotted circle delimits the region of competence of the test sample, and the markers ○ and ■ are samples of different classes.

of the test sample). In Scenario III, the decision boundary of the classifier c crosses the region of competence of the test sample, and c correctly classifies 60% of the samples in the region of competence. In Scenario IV, the decision boundary of the classifier c does not cross the region of competence of the test sample, and c correctly classifies 60% of the samples in the region of competence. Since the classifiers from Scenarios III and IV have the same classification accuracy (60%), accuracy based competence estimation schemes gives the same competence estimative to them, even though the classifier c from Scenario III is more competent because it correctly classifies samples of the different classes in the region of competence, and the classifier c from Scenario IV classifies all samples in the region of competence as being from the same class ("○").

The scenarios from Figure 5 show that depending on the type of region in which the test sample x_{query} is located and whether the decision boundary of the classifier c crosses the region of competence or not, the criteria used by DES techniques are not enough to decide if c is locally competent for the classification of x_{query}. Our hypothesis is that, when the test sample is located in an indecision region, performing the selection of classifiers from a subset of the pool containing only classifiers with decision boundaries that cross the region of competence of the test sample (if such classifiers exist) is a promising dynamic selection approach.

In this paper, we propose a dynamic ensemble selection framework for two-class classification problems. The framework is divided into three phases: (1) Overproduction, where the framework generates the pool of classifiers; (2) Region of Competence Definition, where the framework defines the region of competence of each new test sample; (3) Selection, where the framework selects locally competent classifiers for the classification of each new test sample. The Selection phase is divided in three main steps: Indecision Region Detection, Dynamic Pruning, and Dynamic Selection. The Indecision Region Detection step decides if the test sample is located in an indecision region. The Dynamic Pruning step pre-selects locally competent classifiers by removing (or "pruning") classifiers with decision boundaries that do not cross the region of competence of the test sample when the test sample is located in an indecision region, if no classifier has decision boundaries crossing the region of competence, the Dynamic Pruning step pre-selects all classifiers. The Dynamic Selection step selects the most competent classifiers from the set of pre-selected classifiers for the classification of the test sample using any DES technique.

In the experiments, we evaluated the proposed framework with 9 dynamic selection schemes from the literature using 40 datasets from KEEL (ALCALÁ et al., 2010), and compared the framework with 3 state-of-the-art DES approaches, named: Randomized Reference Classifier (RRC) (WOLOSZYNSKI; KURZYNSKI, 2011), META-DES (CRUZ et al., 2015), and META-DES.Oracle (CRUZ; SABOURIN; CAVALCANTI, 2017b). The results showed that the proposed framework outperforms DES, for all DES techniques

used in our experiments, demonstrating that the framework significantly improves the classification accuracy of Multiple Classifier Systems. Also, using simple DES techniques in the dynamic selection step, the proposed framework was able to achieve statistically equivalent performance to the current state-of-the-art DES frameworks from the literature. These results were confirmed by Wilcoxon Signed Rank Test (WILCOXON, 1945), Sign Test (DEMŠAR, 2006), Friedman test (FRIEDMAN, 1937), and Nemenyi post hoc test (NEMENYI, 1962).

This paper is organized as follows: Section II presents the problem statement. Section III presents the proposed framework. Section IV presents the experimental study. Finally, Section V presents the conclusion.

3.2 Problem Statement

In this section, we define indecision regions (Section 3.2.1), and show that DCS (Section 3.2.2) and DES (Section 3.2.3) techniques have problems evaluating the competence of classifiers and selecting competent classifiers for the classification of test samples located in indecision regions.

3.2.1 Indecision Regions

According to García et al. (GARCÍA; LUENGO; HERRERA, 2015), there are three types of test samples (Figure 6): safe samples, borderline samples, and noisy samples. Safe samples (labeled as S) are located in a neighborhood of samples with relatively homogeneous class labels. Borderline samples (labeled as B) are located in areas surrounding classes boundaries, where different classes overlap, or the samples of different classes are very close to each other. Noisy samples (labeled as N) are samples from one class occurring in areas of another class. Indecision regions (continuous line) are regions with borderline samples of different classes.

In the context of DES, we state that a test sample is located in an indecision region when its region of competence is crossed by one or more classes boundaries, that is, when its region of competence has borderline samples of different classes.

Correctly classifying test samples located in indecision regions is a difficult task because most misclassifications occur in areas near classes boundaries (NAPIERAŁA; STEFANOWSKI; WILK, 2010). In fact, the classification performance of classifiers is strongly affected by the number of borderline samples (GARCÍA et al., 2006).

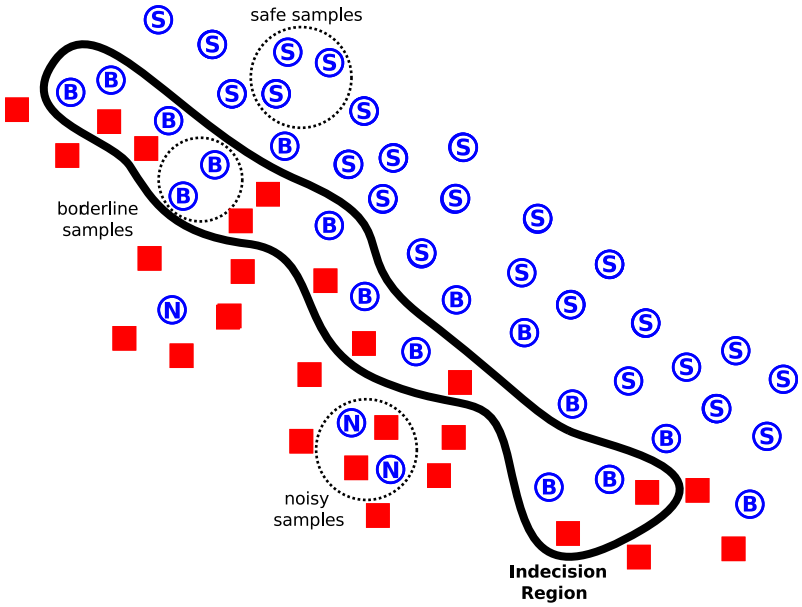


Figure 6 – Three types of samples: safe samples (labeled as S), borderline samples (labeled as B), and noisy samples (labeled as N). The continuous line shows the indecision region, where the classes are represented by the markers ○ and ■ (Adapted from (GARCÍA; LUENGO; HERRERA, 2015)).

3.2.2 DCS in Indecision Regions

The Overall Local Accuracy (OLA) (WOODS; KEGELMEYER; BOWYER, 1997) technique estimates the competence of a classifier using its accuracy in the region of competence of the test sample, meaning OLA selects the classifier that correctly classifies most samples in the region of competence of the test sample. When the test sample is located in an indecision region, OLA can select a classifier that assigns all samples in the region of competence to the same class.

The Local Class Accuracy (LCA) (WOODS; KEGELMEYER; BOWYER, 1997) technique estimates the competence of a classifier that assigns a test sample to the ω class as the classification accuracy of the classifier considering only the samples in the region of competence that were also assigned to the ω class. When the test sample is located in an indecision region, LCA can also select a classifier that assigns all samples in the region of competence of the test sample to the same class.

Figure 7 shows a test sample (▲) located in an indecision region, the region of competence of the test sample (5 nearest neighbors named A, B, C, D, and E), and the decision boundaries of two classifiers $c1$ and $c2$ (continuous straight lines). In this figure, the classifier $c1$ has its decision boundary crossing the region of competence of the test sample, and the classifier $c2$ has its decision boundary not crossing the region of competence (classifying all samples in the region of competence as "■").

Table 1 presents the true class and assigned classes by $c1$ and $c2$ of the samples A,

B, C, D and E from Figure 7. This table also shows the competence estimation of each classifier using OLA and LCA.

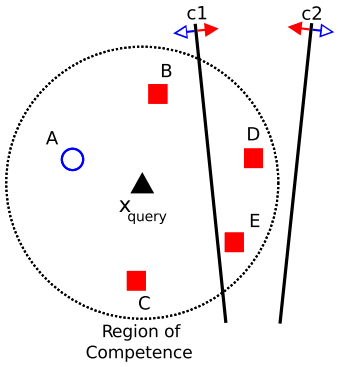


Figure 7 – Test sample located in an indecision region, where ▲ is the test sample, the dotted region is the region of competence (composed by the samples A, B, C, D and E), the markers ○ and ■ are samples from different classes, $c1$ and $c2$ are the classifiers and the continuous lines their decision boundaries, and the true class of the test sample is ○.

Table 1 – Outputs and competence estimatives of classifiers from Figure 7, where the line *true* shows the true class labels, and the lines $c1$ and $c2$ are the classifications of A, B, C, D, E, and the test sample (▲). Columns *OLA* and *LCA* are the competence estimatives of the classifiers using OLA and LCA.

Classifiers	A	B	C	D	E	▲	*OLA*	*LCA*
true	○	■	■	■	■	○	-	-
c1	○	○	○	■	■	○	0.60	0.33
c2	■	■	■	■	■	■	0.80	0.80

In the example from Figure 7 and Table 1, OLA and LCA estimate the competence of $c2$ higher than the competence of $c1$, selecting $c2$, and misclassifying the test sample. OLA and LCA select $c2$, despite the fact that the classifier $c2$ classifies all samples in the region of competence as being from the same class ("■") and $c1$ correctly classifies samples of different classes in the region of competence ("○" and "■").

This shows that classifiers with decision boundaries not crossing the region of competence of a test sample located in an indecision region are not locally competent for the classification of the test sample. These classifiers might be biased classifiers (always assign samples to the same class) or competent for other local regions.

These same drawbacks are present in other DCS techniques, since none of them specifically handles the different scenarios presented in Figure 5. The A Priori (GIACINTO;

ROLI, 1999) and A Posteriori (GIACINTO; ROLI, 1999) techniques, for example, have the same problem of selecting classifiers that are not locally competent when the test sample is located in an indecision region as OLA and LCA. This is due to the fact that A Priori and A Posteriori methods are very similar to OLA and LCA, respectively, except for using the probabilistic representations defined in Equations 3.1 and 3.2,

$$\text{competence}_{\text{a priori}}(c_i) = \frac{\sum_{j=1}^{N} \hat{p}(\omega|\Psi_j \in \omega, c_i) \times \delta_j}{\sum_{j=1}^{N} \delta_j} \tag{3.1}$$

$$\text{competence}_{\text{a posteriori}}(c_i, \omega_l) = \frac{\sum_{\Psi_j \in \omega_l} \hat{p}(\omega_l|\Psi_j, c_i) \times \delta_j}{\sum_{j=1}^{N} \hat{p}(\omega_l|\Psi_j, c_i) \times \delta_j} \tag{3.2}$$

where x_{query} is the test sample, Ψ is the set of samples in the region of competence of x_{query}, Ψ_j the j-th sample in Ψ, N is the number of samples in Ψ, ω a given class, ω_l is the class c_i assigned to x_{query}, δ_j is the inverse of the distance between x_{query} and Ψ_j, and $\hat{p}(\omega|\Psi_j, c_i)$ and $\hat{p}(\omega_l|\Psi_j, c_i)$ are, respectively, the A Priori and A Posteriori measures of the classifier c_i accuracy for the test sample x_{query} based on Ψ_j,

3.2.3 DES in Indecision Regions

In order to show that DES techniques can select classifiers that are not locally competent for the classification of a test sample when the test sample is located in an indecision region, we use K-Nearest Oracles Union (KNORA-U) (KO; SABOURIN; JR, 2008) and K-Nearest Oracles Eliminate (KNORA-E) (KO; SABOURIN; JR, 2008) DES techniques. KNORA-U and KNORA-E were selected because they are simple (facilitating the graphical representation of the problem) and provide equal or better classification performances than other more sophisticated DES techniques (BRITTO; SABOURIN; OLIVEIRA, 2014).

3.2.3.1 *KNORA-Union in Indecision Regions*

KNORA-U selects all classifiers that correctly classify at least one sample in the region of competence of the test sample. The more samples a classifier correctly classifies, the more votes it has.

Figure 8 shows a test sample (▲) located in an indecision region, the region of competence of the test sample (markers within the dotted circle),and the decision boundaries of four classifiers $c1$, $c2$, $c3$, and $c4$ (continuous straight lines). In this figure, the classifier $c2$ and $c3$ have decision boundaries crossing the region of competence, while $c1$ and $c4$ have decision boundaries not crossing the region of competence (classify all samples as "○" and "■", respectively).

In the example from Figure 8, KNORA-U selects all classifiers ($c1$, $c2$, $c3$, and $c4$) for the classification of the test sample, despite the fact that $c1$ and $c4$ have decision boundaries not crossing the region of competence.

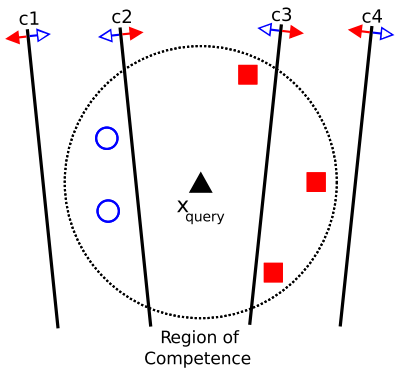


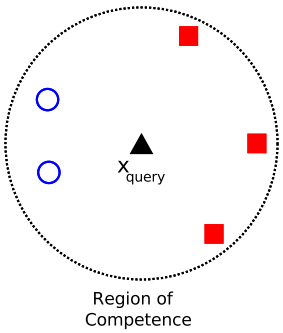
Figure 8 – Test sample located in an indecision region, where ▲ is the test sample, the dotted region is the region of competence, the markers ○ and ■ are samples from different classes, the continuous straight lines are the classifiers ($c1$, $c2$, $c3$, and $c4$).

When the test sample is located in an indecision region, an ideal classifier would be able to correctly distinguish samples of different classes in the region of competence of the test sample. Hence, classifiers that classify all samples in the region of competence of a test sample located in an indecision region as being from the same class are not locally competent for the classification of that test sample. The example from Figure 8 shows that DES techniques can select classifiers that are not locally competent ($c1$ and $c4$) when the test sample is located in an indecision region.

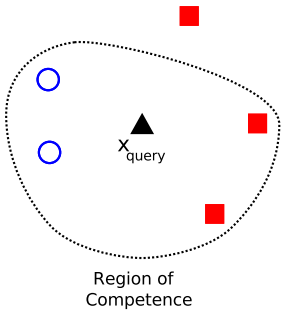
3.2.3.2 *KNORA-Eliminate in Indecision Regions*

KNORA-E selects all classifiers that correctly classify all samples in the region of competence of the test sample. If no classifiers are selected, the region of competence is reduced until at least one classifier is selected.

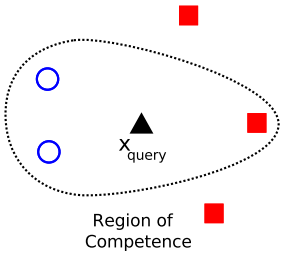
Figure 9 presents the iterations of KNORA-E (K=5) in an example in which no classifiers correctly classify all samples in the region of competence of the test sample until the last iteration. In the first three iterations, no classifiers correctly classify all K-nearest neighbors of the test sample, then, the region of competence is reduced (value of K is decreased). In the last iteration (Figure 9(d)), KNORA-E removed the third nearest sample (the last remaining sample from the class "■"), leaving only two samples of the class "○" in the region of competence. This behavior is not optimal because classifiers that classify all samples in the original region of competence as "○" class are selected.



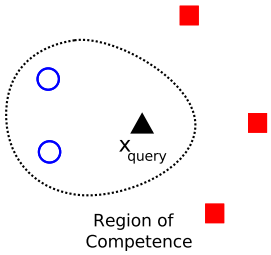
(a) 1st iteration of KNORA-E.



(b) 2nd iteration of KNORA-E.



(c) 3rd iteration of KNORA-E.



(d) 4th iteration of KNORA-E.

Figure 9 – Decreasing neighborhood of KNORA-E when no classifiers correctly classify all samples in the region of competence of the test sample. The ▲ is the test sample, the markers ○ and ■ are samples from different classes in the region of competence of the test sample.

The example from Figure 9 shows that, when the test sample is located in an indecision region, KNORA-E can reduce the region of competence until the region that was composed of borderline samples of different classes is composed of borderline samples of a single class. This is an issue because instead of selecting classifiers with decision boundaries crossing the region of competence of a given test sample located in an indecision region, KNORA-E changes the region of competence until it is no longer an indecision region, and selects classifiers with decision boundaries not crossing the region of competence.

3.3 The Proposed Framework: FIRE-DES

The Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES) framework is divided into three phases (Figure 10):

1. Overproduction phase, where the pool of classifiers C is generated using the training set ($\mathcal{T}$).

2. Region of Competence (RoC) Definition phase, where the region of competence Ψ of a given test sample x_{query} is extracted.

3. Selection phase, where the ensemble of classifiers for the classification of each new test sample is selected. Given a test sample x_{query}, the framework decides if x_{query} is located in an indecision region; if so, the framework pre-selects classifiers with decision boundaries crossing the region of competence of x_{query} if such classifiers exist, otherwise, all classifiers are pre-selected. Finally, the framework selects locally competent classifiers for the classification of x_{query} from the pre-selected classifiers; and finally, uses a combination rule to combine the predictions of the selected classifiers into a single prediction.

In Figure 10, $\mathcal{T}$ is the training set, *Generation* is an ensemble generation process (i.e. Bagging (BREIMAN, 1996)), and C is the generated pool of classifiers; $\mathcal{G}$ is the test set, x_{query} is the test sample; $\mathcal{D}_{SEL}$ is the validation set, *Region of Competence* is the process of selecting the region of competence of x_{query}, Ψ is the region of competence of x_{query}; *Indecision Region* is the Indecision Region Detection step, *Dynamic Pruning* is the Dynamic Pruning step, C_{pruned} is the pre-selected ensemble of classifiers, *Dynamic Selection* is the Dynamic Selection step; C' is the final selected classifiers, *Combination* is the process of combining the prediction of the classifiers in C', and *class(x_{query})* is the final prediction of x_{query}.

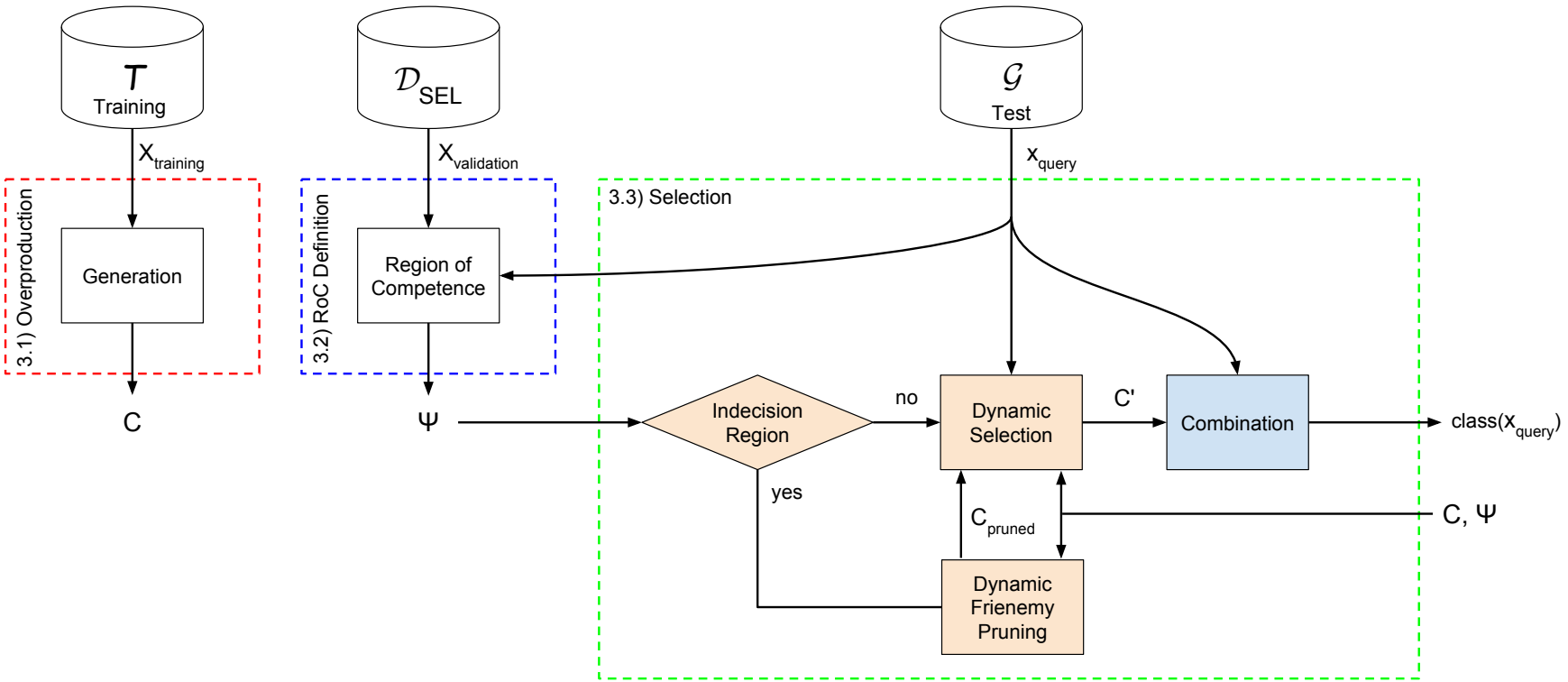


Figure 10 – Overview of the proposed FIRE-DES framework, where $\mathcal{G}$ is the test set, x_{query} is the test sample, $\mathcal{T}$ is the training set, *Generation* is a ensemble generation process (i.e. Bagging) used to generate the pool of classifiers C, $\mathcal{D}_{SEL}$ is the validation set, *Region of Competence* is the process that selects the region of competence Ψ of x_{query} with size K, *Indecision Region* is the Indecision Region Detection step, *Dynamic Pruning* is the Dynamic Pruning step, *Dynamic Selection* is the Dynamic Selection step, C_{pruned} is the set of pre-selected classifiers, C' is the ensemble of selected classifiers for the classification of x_{query}, *Combination* is a combination rule, and *class(x_{query})* is the final classification of x_{query}.

3.3.1 Overproduction

The overproduction phase generates the pool of classifiers C using any ensemble generation technique. In this work, we use the Bagging technique (BREIMAN, 1996) (SKURICHINA; DUIN, 1998). The idea behind Bagging is to build an ensemble of diverse classifiers by "bootstrapping" or randomly sampling (with replacement) samples from the training set, and use each bootstrap to train a new base classifier. Since the focus of this paper is on ensemble selection, and not on ensemble generation, only the bagging technique is considered.

3.3.2 Region of Competence (RoC) Definition

The region of competence definition phase defines the region of competence Ψ of the test sample x_{query}. In this work, Ψ is defined as the set of K nearest neighbors of x_{query} in the validation set $\mathcal{D}_{SEL}$.

3.3.3 Selection

The selection phase performs the selection of locally competent classifiers for the classification of each new test sample. This phase has three main steps:

1. The Indecision Region Detection step (*Indecision Region*), where the framework evaluates the region of competence of x_{query} (Ψ) in order to decide if it is located in an indecision region. If x_{query} is located in an indecision region, the framework goes to step 2, otherwise, step 3.

2. The Dynamic Pruning step, where the framework pre-selects locally competent classifiers with decision boundaries crossing the region of competence (C_{pruned}) from the pool of classifiers C.

3. The Dynamic Selection step, where a DS technique is used to select the final ensemble of classifiers for the classification of x_{query} from C_{pruned} (if x_{query} is located in an indecision region) or from C (if x_{query} is not located in an indecision region). Any DS technique can be used in this step.

3.3.3.1 *Indecision Region Detection*

The task of deciding if a test sample is located in an indecision region is difficult because indecision regions are regions that have borderline samples of different classes, and borderline samples and noisy samples are commonly mistaken for each other (GARCÍA; LUENGO; HERRERA, 2015). In fact, many prototype selection techniques aimed at removing noisy samples also remove borderline samples (GARCIA et al., 2012).

In this paper, we propose that a given test sample is located in an indecision region if its region of competence has samples of more than one class.

3.3.3.2 *Dynamic Pruning*

In order to explain this step, we first introduce the *frienemy samples* concept, and then present the Dynamic Frienemy Pruning (DFP) method.

For the classification of a test sample x_{query}, two samples x_a and x_b are frienemies if: (1) x_a and x_b are located in the region of competence of x_{query}; and (2) x_a and x_b have different classes.

Figure 11 shows the test sample ▲ and its region of competence defined by the samples A, B, C, D and E. In this example, the combination of pairs of opposite classes (○, ■), named (A, C), (A, D), (A, E), (B, C), (B, D), (B, E), are the frienemy samples, or *frienemies*, in relation to the test sample.

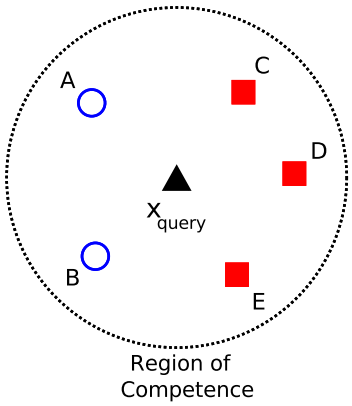


Figure 11 – Figure presenting the pairs of frienemies (A, C), (A, D), (A, E), (B, C), (B, D), (B, E) in the region of competence of the test sample ▲

Independently of which samples in the region of competence of a test sample are from the same class of the test sample (*friends*) or from a different class (*enemies*) the ideal classifier would be able to distinguish all *frienemies* in the region of competence of the test sample.

The DFP method pre-selects locally competent classifiers from the pool of classifiers for each new test sample. The general idea is, for each test sample x_{query}, to pre-select classifiers that correctly classify at least one pair of frienemies in the region of competence of x_{query}. When pre-selecting classifiers, DFP dynamically *prunes* the pool of classifiers, temporally removing locally incompetent classifiers for the classification of x_{query}.

Algorithm 7 presents the DFP method for the pre-selection of classifiers given the region of competence (Ψ) of a test sample.

Algorithm 7 DFP Method

Require: $\mathcal{C}$: pool of classifiers
Require: Ψ: region of competence of the test sample
1: $C_{pruned} \leftarrow$ empty ensemble of classifiers
2: $\mathcal{F} \leftarrow$ all pair of frienemies in Ψ
3: **for all** c_i in $\mathcal{C}$ **do**
4: $\quad \Phi \leftarrow$ samples in Ψ correctly classified by c_i
5: $\quad \mathcal{F}_i \leftarrow$ frienemies in Φ
6: $\quad$ **if** $|\mathcal{F}_i| \geq 1$ **then**
7: $\quad\quad C_{pruned} \leftarrow C_{pruned} \cup c_i$
8: $\quad$ **end if**
9: **end for**
10: **if** $|C_{pruned}| = 0$ **then**
11: $\quad C_{pruned} \leftarrow \mathcal{C}$
12: **end if**
13: **return** C_{pruned}

In Algorithm 7, first, DFP creates an empty set of classifiers C_{pruned} where all selected classifiers are added (Line 1), and finds $\mathcal{F}$, all pairs of frienemies in the region of competence of the test sample (Line 2). After that, DFP selects all classifiers that correctly classify at least 1 pair of frienemy samples (Lines 3 - 9). If no classifier correctly classifies at least one pair of frienemy samples, DFP includes all classifiers from C in C_{pruned} (Lines 10 - 12). Finally, the set of selected classifiers C_{pruned} is returned (Line 13).

The DFP method can be used as a pre-selector in order to keep only base classifiers with decision boundaries crossing the region of competence of a test sample, helping to define more precisely the concept of "local competence" evaluation of base classifiers for the classification of the test sample. However, if no DS method is used after the DFP, the DFP method works as a DS method that selects classifiers with decision boundaries crossing the region of competence of the test sample (if such classifiers exist).

3.3.3.3 *Dynamic Selection*

Section 3.2 presented the problems of DCS and DES methods when the test sample is located in an indecision region (they select locally incompetent classifiers, i.e., classifiers with decision boundaries not crossing the region of competence).

The FIRE-DES framework handles these problems by detecting when the test sample is located in an indecision region, using the DFP method to pre-select classifiers with decision boundaries crossing the region of competence, and finally, selecting the best classifiers from the classifiers pre-selected by the DFP method.

Using the FIRE-DES with OLA, LCA, A Priori, A Posteriori, KNORA-U and KNORA-E in the dynamic selection step, we now have: FIRE-OLA, FIRE-LCA, FIRE-A-Priori, FIRE-A-Posteriori, FIRE-KNORA-Union and FIRE-KNORA-E. FIRE-Bagging is the FIRE-DES without dynamic selection step (DFP is the final selector).

In the example of Figure 7 and Table 1, OLA and LCA selected the classifier $c2$, a classifier strongly biased towards the class "■", misclassifying the test sample. In this same example, FIRE-OLA and FIRE-LCA would not select $c2$, because the DFP would remove $c2$ before de final dynamic selection step, leaving only $c1$, and correctly classifying the test sample. The same thing would happen in FIRE-A-Priori and FIRE-A-Posteriori since the removal of $c2$ is performed by the DFP, and the A Priori and A Posteriori methods are basically OLA and LCA, respectively, except for using the probabilistic representations defined in Equations 3.1 and 3.2

In the example of Figure 8, KNORA-U selects all classifiers to submit a vote for the classification of the test sample. On the other hand, FIRE-KNORA-U selects only classifiers *c2* and *c3*, discarding classifiers with decision boundaries outside the region of competence (*c1* and *c2*) in the Dynamic Pruning step.

In the example of Figure 9, KNORA-E selects all classifiers that classify the last two samples in the region of competence as "○" class, including classifiers with decision boundaries outside of the original region of competence (region of competence in the first iteration). FIRE-KNORA-E handles this issue by removing all classifiers with decision boundaries outside of the original region of competence, ensuring all selected classifiers have decision boundaries crossing the radius defined by the K nearest neighbors (inside the original region of competence), if such classifiers exist.

3.4 Experiments

This section presents the methodology used in the experiments, the results, and its analysis.

3.4.1 Dynamic Selection Techniques

In our experiments, we evaluated the FIRE-DES framework using 9 dynamic selection approaches, where 5 were DCS techniques, 3 were DES techniques, and 1 majority vote (no selection). Table 2 presents the dynamic selection techniques considered in this work: Overall Local Accuracy (OLA), Local Class Accuracy (LCA), A Priori (APri), A Posteriori (APos), Multiple Classifier Behavior (MCB), Dynamic Selection KNN (DSKNN), K-Nearest Oracles Union (KNORA-U), K-Nearest Oracles Eliminate (KNORA-E), and Bagging (no selection). Respectively, the FIRE-DES using these techniques are: FIRE-

Table 2 – Dynamic selection techniques considered in the experiment.

Technique	Category	Reference
DCS		
Overall Local Accuracy (OLA)	Accuracy	Woods et al. (WOODS; KEGELMEYER; BOWYER, 1997)
Local Class Accuracy (LCA)	Accuracy	Woods et al. (WOODS; KEGELMEYER; BOWYER, 1997)
A Priori (APri)	Probabilistic	Giacinto et al. (GIACINTO; ROLI, 1999)
A Posteriori (APos)	Probabilistic	Giacinto et al. (GIACINTO; ROLI, 1999)
Multiple Classifier Behavior (MCB)	Behavior	Giacinto et al. (GIACINTO; ROLI, 2001b)
DES		
Dynamic Selection KNN (DSKNN)	Diversity	Santana et al. (SANTANA et al., 2006)
K-Nearests Oracles Union (KNORA-U)	Oracle	Ko et al. (KO; SABOURIN; JR, 2008)
K-Nearests Oracles Eliminate (KNORA-E)	Oracle	Ko et al. (KO; SABOURIN; JR, 2008)

OLA, FIRE-LCA, FIRE-A Priori, FIRE-A Posteriori, FIRE-MCB, FIRE-DSKNN, FIRE-KNORA-U, FIRE-KNORA-E, and FIRE-Bagging.

We also present the classification performance of the Single Best Classifier and the Oracle. The Single Best Classifier is a model that selects the classifier that correctly classify most samples in the test set. The Oracle is an abstract model which always selects the classifier that correctly classify a given test sample, if such classifier exists.

We also compare the FIRE-DES approach that achieved the highest classification performance with state-of-the-art DES approaches: Randomized Reference Classifier (RRC) (WOLOSZYNSKI; KURZYNSKI, 2011), META-DES (CRUZ et al., 2015), and META-DES.Oracle (CRUZ; SABOURIN; CAVALCANTI, 2017b).

3.4.2 Datasets

We evaluated FIRE-DES using 40 datasets from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository (ALCALÁ et al., 2010). Since dynamic selection techniques have been shown to be an effective approach for small datasets (CAVALIN; SABOURIN; SUEN, 2013), and ensemble learning has recently become popular in dealing with the class imbalance problem (GALAR et al., 2012) (NANNI; FANTOZZI; LAZZARINI, 2015) (GALAR et al., 2016), our experiments focused on small sized binary imbalanced datasets. Table 3 presents the summary of the datasets used in this experiment: label, name, number of features, number of samples, and imbalance ratio (IR).

3.4.3 Evaluation

The datasets were partitioned using *stratified 5-fold cross-validation* (1 fold for test, 4 folds to training/validation) followed by a *stratified 4-fold cross-validation* (the 4 folds in training/validation divided in 3 folds for training, 1 fold for validation), that is, 20 executions using stratified partitions with 3 folds for training, 1 fold for validation, and 1 fold for test. With the results of the 20 executions, we got the mean and standard deviation for each dataset. Since the META-DES and META-DES.Oracle techniques require two

Table 3 – Summary of the 40 datasets used in the experiments: label, name, number of features, number of samples, and imbalance ratio.

Label	Name	#Feats.	#Samples	IR
1	glass1	9	214	1.82
2	ecoli0vs1	7	220	1.86
3	wisconsin	9	683	1.86
4	pima	8	768	1.87
5	iris0	4	150	2.00
6	glass0	9	214	2.06
7	yeast1	8	1484	2.46
8	vehicle2	18	846	2.88
9	vehicle1	18	846	2.90
10	vehicle3	18	846	2.99
11	glass0123vs456	9	214	3.20
12	vehicle0	18	846	3.25
13	ecoli1	7	336	3.36
14	new-thyroid1	5	215	5.14
15	new-thyroid2	5	215	5.14
16	ecoli2	7	336	5.46
17	segment0	19	2308	6.00
18	glass6	9	214	6.38
19	yeast3	8	1484	8.10
20	ecoli3	7	336	8.60
21	yeast-2vs4	8	514	9.08
22	yeast-05679vs4	8	528	9.35
23	vowel0	13	988	9.98
24	glass-016vs2	9	192	10.29
25	glass2	9	214	11.59
26	shuttle-c0vsc4	9	1829	13.87
27	yeast-1vs7	7	459	14.30
28	glass4	9	214	15.47
29	ecoli4	7	336	15.80
30	page-blocks-13vs4	10	472	15.86
31	glass-0-1-6_vs_5	9	184	19.44
32	shuttle-c2-vs-c4	9	129	20.50
33	yeast-1458vs7	8	693	22.10
34	glass5	9	214	22.78
35	yeast-2vs8	8	482	23.10
36	yeast4	8	1484	28.10
37	yeast-1289vs7	8	947	30.57
38	yeast5	8	1484	32.73
39	ecoli-0137vs26	7	281	39.14
40	yeast6	8	1484	41.40

validation sets, we divided the training set in two parts to obtain the second validation set for the meta-training set.

For evaluation metric, we used the Area Under the ROC Curve (AUC) (BRADLEY, 1997). We used the AUC because it is a suitable metric for imbalanced datasets, especially for binary problems (LÓPEZ et al., 2013).

For individual dataset pairwise performance comparison, we used the Wilcoxon Signed Rank Test (WILCOXON, 1945). For overall datasets pairwise performance comparison, we used the Wilcoxon Signed Rank Test, and the Sign Test (DEMŠAR, 2006). For general evaluation, we used the Friedman test (FRIEDMAN, 1940) and the Nemenyi post-hoc test (NEMENYI, 1962).

3.4.4 Parameters setting

The performance of the proposed system depends on the following parameters: the dynamic selection procedure, the region of competence size (K), and the size of the pool (N).

The value of the parameter K was selected based on the results in (CRUZ; CAVALCANTI; REN, 2011). In this case, $K = 7$ presented the best average results for most dynamic selection techniques.

The value of the parameter N (size of the pool) was selected considering the results in (KO; SABOURIN; JR, 2008) (the higher the N value, the higher the classification accuracy until the pool converges). Following the strategy in (CRUZ et al., 2015), we generated a pool of $N = 100$ Perceptrons using the Bagging (BREIMAN, 1996) technique.

3.4.5 Indecision Regions

Figure 12 presents the proportion of test samples classified as being located in indecision regions (blue bars), and the proportion of test samples that had classifiers pre-selected by the Dynamic Frienemy Pruning (DFP) procedure (red bars). This figure shows that 25% of the samples are located in indecision regions and 23% of the samples had classifiers pre-selected by the DFP procedure, meaning that only 2% of the samples were located in indecision regions and had no classifiers with decision boundaries crossing the region of competence.

In (SMITH; MARTINEZ; GIRAUD-CARRIER, 2014), the authors estimated data complexity using average instance hardness. Instance hardness is metric that defines the difficulty of correctly classifying a given test sample (the higher the instance hardness, the higher the probability of being a noisy sample), the average instance hardness evaluates indicates the level of noise and overlap of classes in a given dataset. Figure 13 presents the scatter plot of the datasets (markers), where the horizontal axis is the average instance

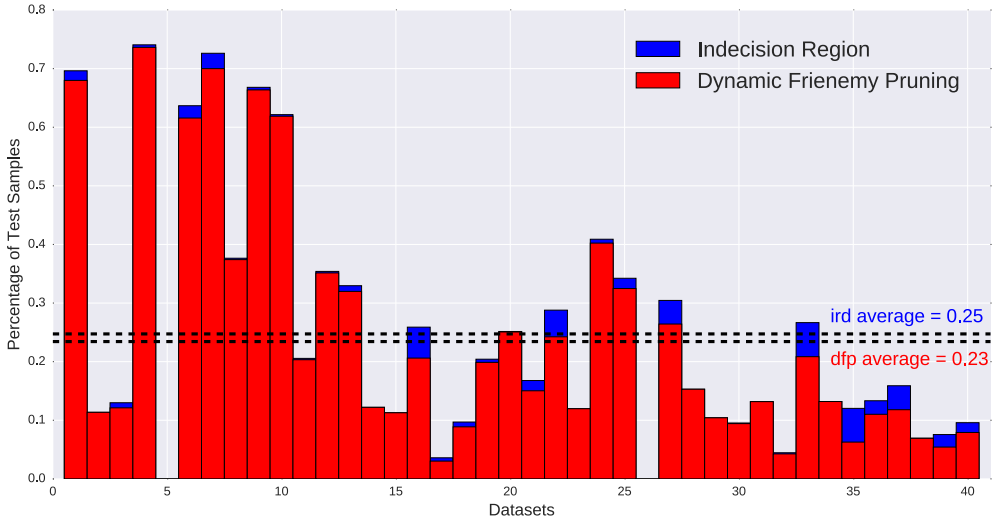


Figure 12 – Test samples affected by the FIRE-DES framework with Perceptrons as base classifiers. Where the blue bars represent the proportion of test samples classified as being located in indecision regions, and *ird average* its average, the red bars represent the proportion of test samples that had classifiers pre-selected by the DFP procedure for its classification, and *dfp average* its average.

hardness of the samples in the validation set, and the vertical axis is the proportion of test samples classified as being located in indecision regions. The instance hardnesses were estimated using K-Nearest Neighbors Classifiers (K=5) trained with the training sets.

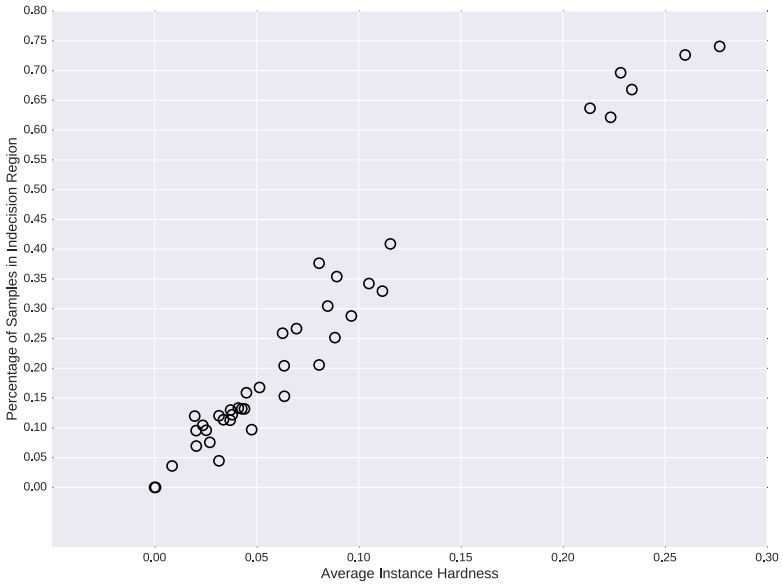


Figure 13 – Scatter plots of datasets, where the datasets are the markers, the horizontal axis is the average instance hardness of the datasets, and the vertical axis is the proportion of test samples classified as being located in indecision regions. Pearson correlation coefficient $r = 0.9829$.

The Pearson correlation coefficient of the percentage of samples classified as being located in an indecision region and the average instance hardness is $r = 0.9829$, meaning a positive linear correlation between percentage of samples located in indecision regions and average instance hardness. Figure 13 and the calculated Pearson correlation coefficient show that the difference between the proportion of test samples classified as being located in indecision regions in different datasets is simply due data distribution, meaning, the more "difficult" the dataset, the more samples classified as being located in indecision regions.

3.4.6 FIRE-DES vs. DES

Table 4 presents the average AUC of OLA, FIRE-OLA, LCA, FIRE-LCA, A Priori, FIRE-A Priori, KNORA-U, FIRE-KNORA-U, KNORA-E, FIRE-KNORA-E, Bagging (not using the validation set), FIRE-Bagging, Single Best Classifier, and Oracle. For each dataset, the best results are highlighted in bold, and significantly better results are marked with • (p-*value* ≤ 0.10) and •• (p-*value* ≤ 0.05), where p-*value* is obtained with the Wilcoxon Signed Rank Test. In the last three lines, *ranking* is the average ranking, p-*value* is the result of the Wilcoxon Signed Rank Test comparing the FIRE-DES approaches with the respective DES approaches on all datasets, and $W/T/L$ is the counts of wins, ties and

Table 4 – Mean results of the accuracy obtained for OLA, FIRE-OLA, LCA, FIRE-LCA, A Priori, FIRE-A Priori, MCB, FIRE-MCB, DSKNN, FIRE-DSKNN, KNORA-U, FIRE-KNORA-U, KNORA-E, FIRE-KNORA-E, Bagging, FIRE-Bagging, Single Best Classifier, and Oracle. A pool of 100 perceptrons as base classifiers is used for all techniques. Comparing FIRE-DES with DES techniques, the best results are in bold, and significantly better results are marked with • (*p-value* $\leq .10$) and •• (*p-value* $\leq .05$), where *p-value* is obtained with the Wilcoxon Signed Rank Test. In the last three lines, *ranking* is the average ranking, *p-value* is the result of the Wilcoxon Signed Rank Test compairing the the average classification performance of FIRE-DES and DES on all datasets, and $W/T/L$ is counts of wins, ties and losses of FIRE-DES compared to DES.

Label	OLA	F-OLA	LCA	F-LCA	APri	F-APri	APos	F-APos	MCB	F-MCB	DSKNN	F-DSKNN	KNU	F-KNU	KNE	F-KNE	Bag	F-Bag	SB	Oracle
1	**0.6269**	0.6179	**••0.6439**	0.6271	**••0.6434**	0.6114	**0.6473**	0.6309	0.5303	**••0.5821**	**0.6278**	0.6070	0.5958	**0.6100**	**••0.6701**	0.6439	0.5198	**••0.5738**	0.6985	1.0000
2	**•0.9566**	0.9497	**0.9549**	0.9488	**•0.9618**	0.9494	**0.9506**	0.9443	**••0.9750**	0.9628	**••0.9749**	0.9603	**••0.9733**	0.9602	**•0.9656**	0.9587	**••0.9764**	0.9602	0.9883	1.0000
3	**••0.9573**	0.9489	**••0.9601**	0.9514	**••0.9685**	0.9538	**••0.9614**	0.9519	**••0.9697**	0.9544	**••0.9684**	0.9576	**••0.9702**	0.9568	**••0.9601**	0.9527	**••0.9725**	0.9510	0.9854	0.9981
4	**•0.6899**	0.6840	**0.7106**	0.7094	**0.6954**	0.6864	0.6726	**0.6831**	**••0.7213**	0.6865	**0.6951**	0.6897	**••0.7206**	0.6947	0.6802	**0.6810**	**••0.7241**	0.6864	0.7751	0.9976
5	**0.9912**	**0.9912**	**0.9975**	**0.9975**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.9950**	**0.9950**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9975	**1.0000**	1.0000	1.0000
6	**••0.7171**	0.6879	**••0.7268**	0.7081	**••0.7439**	0.6780	**•0.7520**	0.7269	0.6562	**••0.6880**	**••0.7387**	0.6753	**•0.7245**	0.6880	**••0.7154**	0.6904	**0.7013**	0.6871	0.8126	1.0000
7	**••0.6636**	0.6479	**0.6607**	0.6554	0.6528	**0.6557**	0.6449	**0.6471**	0.6265	**0.6347**	**0.6582**	0.6528	0.6430	**0.6484**	0.6481	**0.6513**	0.6204	**•0.6343**	0.7274	0.9992
8	**••0.9292**	0.9225	**••0.9334**	0.9275	**0.9300**	0.9232	0.9265	**0.9272**	**0.9018**	0.9014	**••0.9448**	0.9333	**••0.9276**	0.9192	**••0.9442**	0.9348	**0.9008**	0.9005	0.9364	0.9992
9	0.6959	**0.6973**	0.7079	**0.7163**	0.6558	**••0.6898**	0.6488	**••0.6670**	0.6489	**••0.7186**	0.6709	**•0.6890**	0.6486	**••0.7152**	0.6936	**••0.7059**	0.6693	**••0.7162**	0.7601	1.0000
10	0.6736	**0.6788**	0.6818	**••0.6957**	0.6367	**••0.6683**	0.6327	**••0.6662**	0.5974	**••0.6598**	0.6550	**••0.6834**	0.6102	**••0.6646**	0.6797	**••0.6902**	0.5998	**••0.6576**	0.7410	0.9970
11	**0.9033**	0.9025	**0.8985**	0.8977	0.8615	**0.8714**	0.8635	**•0.8733**	0.8892	**••0.9074**	**0.8894**	0.8887	0.8867	**••0.9049**	0.8918	**0.8920**	0.8979	**0.9067**	0.9646	0.9992
12	**0.9381**	0.9374	**0.9402**	0.9389	0.9234	**••0.9384**	0.8983	**••0.9090**	0.9482	**••0.9533**	0.9433	**0.9473**	0.9521	**0.9535**	**0.9447**	0.9440	0.9500	**0.9546**	0.9760	1.0000
13	**••0.8453**	0.8301	**••0.8446**	0.8321	**0.8345**	0.8280	**0.8364**	**0.8364**	**••0.8240**	0.7947	**•0.8506**	0.8311	**••0.8320**	0.8051	**••0.8367**	0.8220	**••0.8279**	0.7952	0.9199	1.0000
14	**0.9623**	**0.9623**	**0.9523**	**0.9523**	0.8950	**••0.9308**	0.8464	**••0.8714**	0.9586	**••0.9836**	0.9773	**0.9809**	0.9586	**••0.9836**	**0.9858**	**0.9858**	0.9665	**••0.9836**	1.0000	1.0000
15	**0.9622**	**0.9622**	**0.9515**	**0.9515**	0.8993	**••0.9336**	0.8393	**••0.8607**	0.9658	**•0.9758**	**0.9687**	0.9615	0.9729	**0.9758**	**0.9687**	**0.9687**	0.9687	**0.9758**	1.0000	1.0000
16	**••0.8857**	0.8322	**••0.8876**	0.8561	**••0.9027**	0.8620	**••0.9142**	0.8747	**•0.7893**	0.7498	**••0.9115**	0.8323	**••0.8607**	0.7995	**••0.8726**	0.8327	**••0.8007**	0.7493	0.9274	0.9841
17	0.9889	**0.9891**	0.9905	**0.9907**	**••0.9903**	0.9891	**0.9912**	0.9910	0.9902	**0.9908**	0.9899	**0.9904**	0.9906	**0.9912**	**0.9908**	0.9907	**0.9909**	0.9908	0.9931	0.9994
18	**0.8530**	0.8516	**0.8480**	0.8466	0.8510	**0.8668**	**0.8697**	0.8655	0.8543	**•0.8697**	**0.8534**	0.8520	0.8585	**•0.8697**	**0.8667**	0.8653	0.8436	**•0.8697**	0.9630	0.9875
19	**0.8445**	0.8396	**0.8445**	0.8407	0.8190	**0.8272**	0.8197	**0.8247**	0.8091	**••0.8292**	**0.8427**	0.8381	0.8275	**0.8360**	0.8212	**•0.8287**	0.8165	**•0.8297**	0.9231	0.9996
20	0.7818	**0.7855**	0.7740	**0.7801**	0.7825	**0.7966**	0.7798	**0.7938**	0.7172	**••0.7857**	0.7769	**0.7873**	0.7560	**••0.7969**	0.7536	**0.7665**	0.7413	**••0.7925**	0.9295	0.9964
21	0.8133	**0.8143**	0.8192	**0.8202**	0.7863	**•0.8092**	0.7609	**0.7752**	0.7999	**••0.8235**	0.7994	**0.8239**	0.8075	**0.8263**	0.8126	**0.8168**	0.8178	**0.8224**	0.9245	0.9727
22	0.6981	**0.7082**	0.6890	**0.7012**	0.6363	**••0.6925**	0.5989	**••0.6437**	0.6362	**••0.7320**	0.6626	**••0.7088**	0.6354	**••0.7324**	0.6860	**0.6975**	0.6726	**••0.7381**	0.8490	0.9625
23	0.9180	**0.9233**	0.9058	**0.9097**	0.9183	**0.9310**	0.8893	**0.9037**	0.8411	**••0.9147**	0.8801	**••0.9197**	0.8471	**••0.9193**	**0.9272**	0.9269	0.8515	**••0.9143**	0.9558	1.0000
24	0.5091	**0.5489**	0.5246	**0.5565**	0.4986	**••0.5796**	0.4986	**0.5360**	0.5000	**••0.5682**	0.5000	**•0.5568**	0.5000	**•0.5508**	0.5379	**0.5477**	0.5000	**•0.5513**	0.7490	1.0000
25	0.4934	**••0.6045**	0.4916	**••0.5899**	0.4994	**••0.6222**	0.4994	**••0.5932**	0.5000	**••0.6208**	0.4974	**••0.6097**	0.5000	**••0.6223**	0.5592	**••0.6140**	0.4994	**••0.6216**	0.7910	1.0000
26	**0.9959**	**0.9959**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**0.9960**	0.9989	0.9990
27	0.6131	**••0.6425**	0.5868	**••0.6178**	0.5414	**••0.6266**	0.5164	**••0.5819**	0.5629	**••0.6598**	0.5643	**••0.6077**	0.5622	**••0.6541**	0.6058	**0.6214**	0.5626	**••0.6578**	0.8179	1.0000
28	0.7236	**0.7365**	**0.6707**	0.6700	0.6106	**••0.7123**	**0.5856**	0.5806	0.5586	**••0.7414**	0.6478	**••0.7249**	0.5586	**••0.7561**	0.7523	**0.7671**	0.5617	**••0.7639**	0.9703	1.0000
29	**0.8999**	0.8987	0.9082	**0.9132**	0.8488	**0.8652**	0.8434	**0.8543**	0.8847	**0.9155**	**0.9023**	0.9015	0.8976	**0.9116**	**0.9007**	0.9003	0.9093	**0.9151**	0.9968	1.0000
30	0.8571	**••0.9265**	0.8485	**••0.9310**	0.7925	**••0.9087**	0.7941	**••0.8743**	0.7208	**••0.9181**	0.7397	**••0.9137**	0.7213	**••0.8998**	**••0.9468**	0.9420	0.7183	**••0.9031**	0.9314	1.0000
31	0.8079	**0.8368**	0.7504	**•0.8100**	0.6118	**••0.8064**	0.5500	**••0.6936**	0.5375	**••0.8421**	0.5700	**••0.8382**	0.5375	**••0.8421**	0.8239	**0.8418**	0.5500	**••0.8504**	0.9807	1.0000
32	**0.9460**	**0.9460**	**0.7500**	**0.7500**	0.7845	**0.7855**	0.6500	**0.7000**	0.8855	**•0.9480**	**0.9470**	**0.9470**	0.8970	**0.9470**	**0.9480**	**0.9480**	0.8960	**••0.9480**	1.0000	1.0000
33	0.4996	**••0.5561**	0.4968	**••0.5488**	0.4996	**••0.5585**	0.4992	**••0.5504**	0.5000	**••0.5638**	0.5000	**••0.5576**	0.5000	**••0.5652**	0.5560	**0.5671**	0.5000	**••0.5714**	0.7111	0.9958
34	0.7314	**•0.8073**	0.6927	**0.7704**	0.5125	**••0.7479**	0.5238	**••0.6896**	0.5375	**••0.8226**	0.5500	**••0.7591**	0.5375	**••0.8125**	0.7195	**0.7716**	0.5500	**••0.8213**	0.9622	1.0000
35	0.7592	**0.7604**	0.7546	**0.7558**	0.7370	**0.7563**	0.7122	**0.7321**	0.7614	**0.7810**	0.7552	**0.7685**	0.7677	**0.7810**	0.7764	**0.7789**	0.7739	**0.7873**	0.8418	0.9375
36	0.6409	**••0.6913**	0.6349	**••0.6889**	0.5470	**••0.6831**	0.5454	**••0.6230**	0.5523	**••0.7146**	0.5981	**••0.6779**	0.5608	**••0.7035**	0.6478	**••0.6961**	0.5580	**••0.7188**	0.8474	0.9500
37	0.5470	**•0.5749**	0.5432	**0.5696**	0.4997	**••0.5695**	0.4997	**••0.5408**	0.5208	**••0.5866**	0.5206	**••0.5598**	0.5083	**••0.5842**	0.5477	**0.5738**	0.5333	**••0.5837**	0.7527	0.9708
38	0.8200	**0.8385**	0.8053	**0.8210**	0.7708	**••0.8183**	0.7484	**••0.7857**	0.6635	**••0.8490**	0.8165	**0.8213**	0.7107	**••0.8523**	0.7871	**•0.8107**	0.6728	**••0.8541**	0.9825	1.0000
39	**0.7811**	0.7784	0.7347	**0.7816**	**0.8477**	0.8316	0.7611	**0.7959**	**•0.8463**	0.8427	**0.8472**	0.8436	**•0.8463**	0.8422	**0.8454**	0.8431	**••0.8477**	0.8427	0.9078	0.9375
40	0.7218	**•0.7593**	0.7252	**••0.7596**	0.6542	**••0.7616**	0.6583	**••0.7307**	0.5948	**••0.7615**	0.6848	**••0.7346**	0.6376	**••0.7531**	0.7449	**0.7557**	0.6090	**••0.7562**	0.9075	0.9679
Mean	0.7911	**0.8017**	0.7809	**0.7946**	0.7560	**0.7930**	0.7407	**0.7681**	0.7443	**0.8058**	0.7728	**0.8006**	0.7560	**0.8081**	0.8003	**0.8055**	0.7516	**0.8058**	0.8975	0.9913
ranking	9.74	9.40	9.76	8.61	12.00	9.39	13.26	11.57	13.11	6.14	9.75	8.01	11.54	5.89	7.54	7.08	11.66	6.55	NA	NA
p-value		0.0406		0.0098		0.0001		$2.37 \times e^{-5}$		$4.40 \times e^{-6}$		0.0080		$4.53 \times e^{-5}$		0.0234		$3.36 \times e^{-5}$	NA	NA
W/T/L		20/5/15		21/5/14		28/2/10		29/3/8		31/2/7		22/3/15		30/2/8		21/5/14		30/1/9	NA	NA

losses of FIRE-DES for each DES technique.

Using the Wilcoxon Signed Rank Test, we confirm that FIRE-DES outperforms DES. Using $\alpha = 0.05$, we confirm with statistical confidence that: FIRE-OLA outperformed OLA ($p\text{-}value = 0.0406$), FIRE-LCA outperformed LCA ($p\text{-}value = 0.0098$), FIRE-APri outperformed A Priori ($p\text{-}value = 0.0001$), FIRE-APos outperformed A Posteriori ($p\text{-}value = 2.37 \times e^{-5}$), FIRE-MCB outperformed MCB ($p\text{-}value = 4.40 \times e^{-6}$), FIRE-DSKNN outperformed DSKNN ($p\text{-}value = 0.0080$), FIRE-KNORA-U outperformed KNORA-U ($p\text{-}value = 4.53 \times e^{-5}$), FIRE-KNORA-E outperformed KNORA-E ($p\text{-}value = 0.0234$), and FIRE-Bag outpeformed Bagging ($p\text{-}value = 3.36 \times e^{-5}$).

Figure 14 presents a pairwise comparison of FIRE-DES and the respective DES techniques. This comparison used the sign test calculated on the computed wins, ties and losses from the last line in Table 4. The null hypothesis H_0 was that the classification performances of FIRE-DES and the respective DES were equivalent, and a rejection in H_0 meant that FIRE-DES significantly outperformed the respective DES. In this evaluation, we used significance level $\alpha = 0.05$. To reject H_0, the number of wins plus half of the number of ties needs to be greater or equal to n_c (Equation 3.3):

$$n_c = \frac{n_{exp}}{2} + z_\alpha \times \frac{\sqrt[2]{n_{exp}}}{2} \tag{3.3}$$

where $n_{exp} = 40$ (the number of experiments), and $z_\alpha = 1.645$, hence, $n_c = 25.20$.

Figure 14 shows that FIRE-DES achieved significant classification performance gain over DES for A Priori, A Posteriori, MCB, KNORA-U and Bagging.

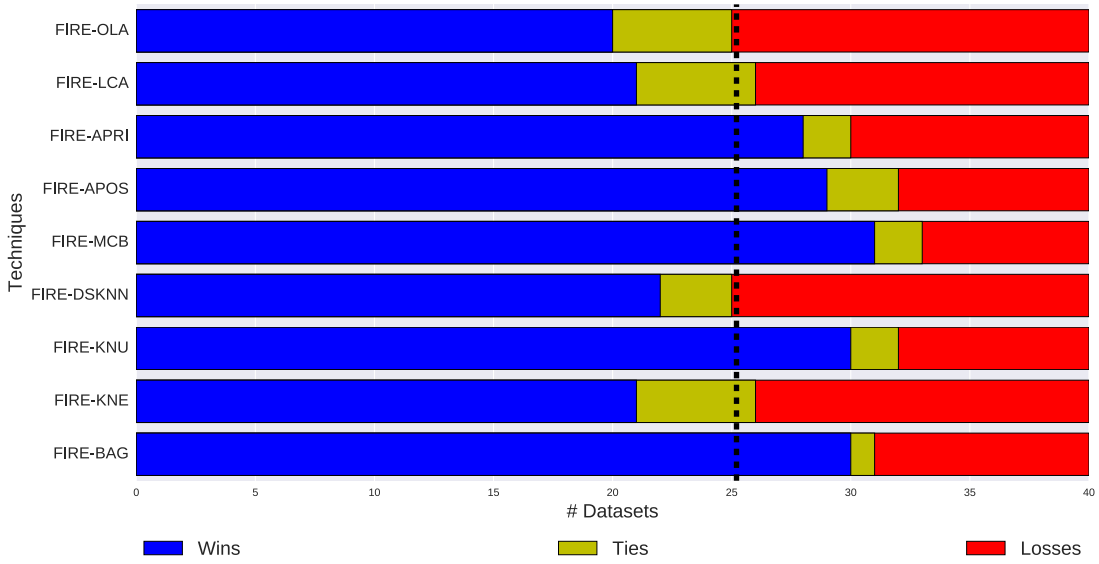


Figure 14 – Performance of the each DES using FIRE-DES in terms of wins, ties and losses. The dashed line illustrates the critical value $n_c = 25.20$.

Figure 15 shows the scatter plot of FIRE-DES (vertical axis) and DES (horizontal axis). In this figure, all markers are above the diagonal line, meaning that FIRE-DES caused an improvement in the average classification performance of all DES techniques used in our experiments.

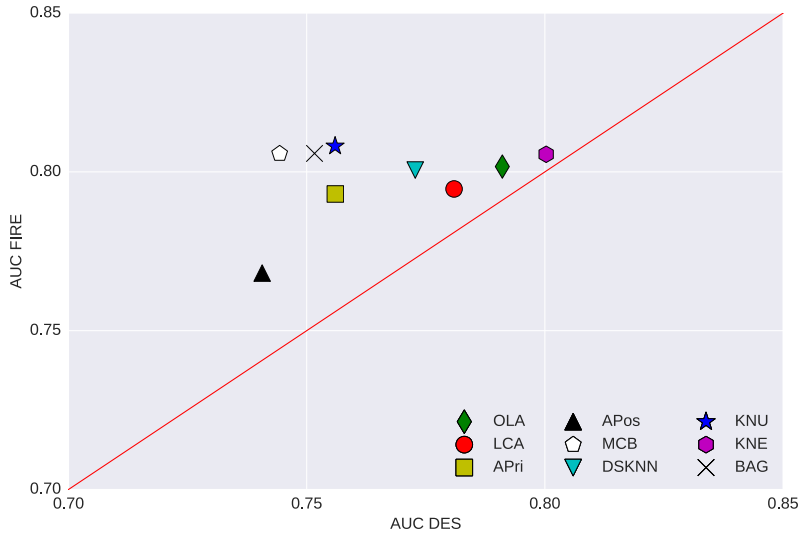


Figure 15 – Scatter plots of FIRE-DES (vertical axis) and DES (horizontal axis). Markers above the diagonal line indicates that FIRE-DES caused a classification performance gain.

Furthermore, the Friedman test was used to compare the results all DES techniques (with and without FIRE-DES) over the 40 classification datasets. The result of the Friedman test ($p\text{-}value = 4.08 \times e^{-20}$) indicates that there is a difference in the classification performances, so we can proceed with the Nemenyi test.

Figure 16 presents the Nemenyi test, in which the lower the ranking the better the technique, and connected ($CD = 3.8933$) techniques have no significant statistical difference. In this figure, FIRE-DES achieved a better ranking than DES for all dynamic selection approaches, and, with statistically significant difference, FIRE-KNORA-U (5.89) outperformed KNORA-U (11.54), FIRE-MCB (6.14) outperformed MCB (13.11), and FIRE-BAG (6.55) outperformed Bagging (11.66).

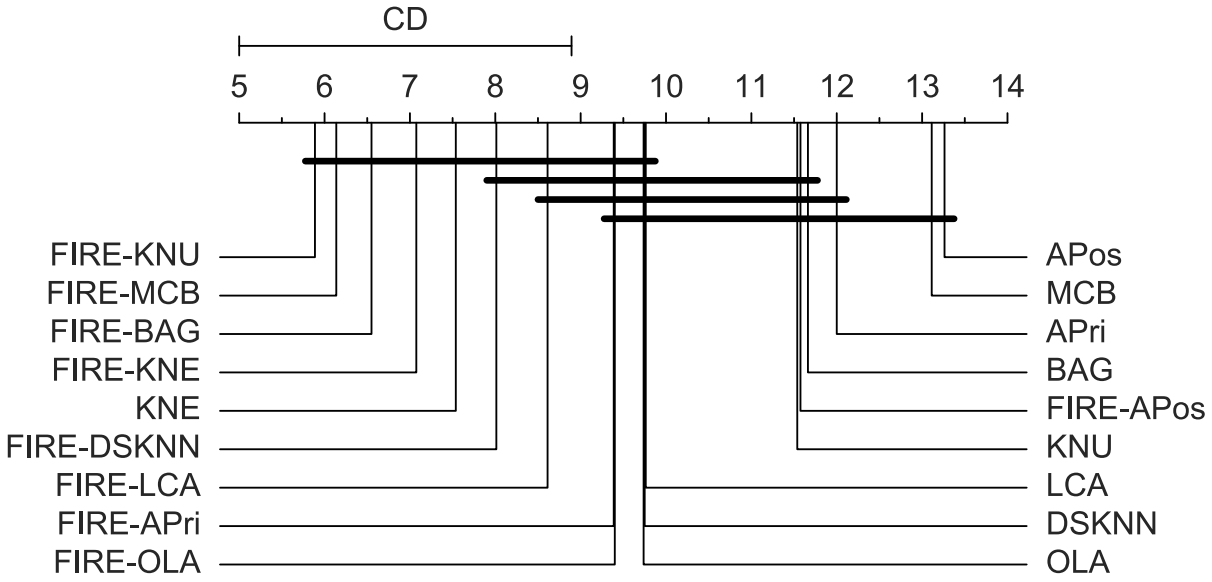


Figure 16 – CD diagram of Nemenyi post-hoc test considering all dynamic selection approaches, where $CD = 3.8933$, and Friedman $p\text{-}value = 4.08 \times e^{-20}$.

3.4.7 FIRE-DES vs. State-of-the-art

Table 5 presents the average AUC of the FIRE-DES technique that achieved the highest classification performance, named FIRE-KNORA-U, KNORA-U, and the state-of-the-art DES techniques: Randomized Reference Classifier (RRC) (WOLOSZYNSKI; KURZYNSKI, 2011), META-DES (META-DES) (CRUZ et al., 2015), and META-DES.Oracle (META-DES.O) (CRUZ; SABOURIN; CAVALCANTI, 2017b). For each dataset, the best results are highlighted in bold, and significantly better results are marked with • ($p\text{-}value \leq 0.10$) and •• ($p\text{-}value \leq 0.05$), where $p\text{-}value$ is obtained with the Wilcoxon Signed Rank Test.

Table 5 shows that the DES technique that achieved the highest classification performance was the META-DES (.8100), followed by FIRE-KNORA-U (.8081), META-DES.Oracle (.8067), RRC (.7934), and finally KNORA-U (0.7560).

Figure 17 presents the Nemenyi test, in which the lower the ranking the better the technique, and connected techniques ($CD = 0.9644$) have no significant statistical difference. In this test, the highest ranking was achieved META-DES (2.23), followed by META-DES.Oracle (2.64), FIRE-KNORA-U (2.70), RRC (3.39), and KNORA-U (4.05).

Table 5 – Mean results of the accuracy obtained for FIRE-KNORA-U, KNORA-U, RRC, META-DES, and META-ORACLE-DES. A pool of 100 perceptrons as base classifiers is used for all techniques. Best results are in bold, and significantly better results are marked with • (*p-value* $\leq .10$) and •• (*p-value* $\leq .05$), where *p-value* is obtained with the Wilcoxon Signed Rank Test.

Label	FKNU	KNU	RRC	META-DES	META-DES.Oracle
1	0.6100(0.0789)	0.5958(0.0643)	0.5693(0.0722)	0.7254(0.0589)	**0.7263 (0.0605)**
2	0.9602(0.0351)	**0.9733 (0.0237)**	0.9535(0.0310)	0.9671(0.0288)	0.9680(0.0270)
3	0.9568(0.0243)	**0.9702 (0.0069)**	0.9614(0.0116)	0.9636(0.0187)	0.9671(0.0146)
4	0.6948(0.0365)	**0.7206 (0.0247)**	0.7179(0.0287)	0.7127(0.0282)	0.7095(0.0340)
5	**1.0000 (0.0000)**	**1.0000 (0.0000)**	0.9975(0.0075)	**1.0000 (0.0000)**	**1.0000 (0.0000)**
6	0.6880(0.0789)	0.7245(0.0759)	0.7196(0.0613)	**0.7853 (0.0766)**	0.7753(0.0716)
7	0.6484(0.0374)	0.6430(0.0299)	0.6161(0.0458)	**0.6487 (0.0216)**	0.6333(0.0213)
8	0.9192(0.0260)	0.9276(0.0212)	0.9223(0.0217)	••**0.9559 (0.0142)**	0.9504(0.0159)
9	••**0.7152 (0.0418)**	0.6485(0.0437)	0.6949(0.0285)	0.6930(0.0300)	0.6685(0.0325)
10	0.6646(0.0510)	0.6102(0.0370)	••**0.7027 (0.0276)**	0.6773(0.0464)	0.6733(0.0484)
11	**0.9049 (0.0625)**	0.8867(0.0753)	0.8936(0.0615)	0.8871(0.0672)	0.9041(0.0631)
12	0.9535(0.0122)	0.9521(0.0125)	0.9476(0.0299)	•**0.9590 (0.0175)**	0.9512(0.0226)
13	0.8051(0.0735)	0.8320(0.0497)	0.8351(0.0492)	0.8408(0.0600)	**0.8460 (0.0579)**
14	**0.9836 (0.0279)**	0.9586(0.0416)	0.9802(0.0287)	0.9809(0.0272)	0.9787(0.0295)
15	0.9758(0.0313)	0.9729(0.0329)	**0.9858 (0.0285)**	0.9758(0.0448)	0.9700(0.0519)
16	0.7995(0.0810)	0.8607(0.0540)	0.8304(0.0730)	**0.8898 (0.0455)**	0.8828(0.0525)
17	0.9911(0.0067)	0.9906(0.0071)	0.9875(0.0096)	**0.9912 (0.0080)**	0.9911(0.0080)
18	0.8697(0.0645)	0.8585(0.0597)	0.9156(0.0584)	0.9180(0.0577)	**0.9214 (0.0560)**
19	0.8360(0.0610)	0.8275(0.0517)	0.8308(0.0601)	**0.8406 (0.0350)**	0.8376(0.0388)
20	**0.7969 (0.1255)**	0.7560(0.0929)	0.7623(0.1062)	0.7475(0.0998)	0.7630(0.0863)
21	0.8263(0.0564)	0.8075(0.0609)	0.8329(0.0515)	**0.8427 (0.0608)**	0.8339(0.0685)
22	**0.7323 (0.0698)**	0.6354(0.0591)	0.6726(0.0624)	0.7218(0.0782)	0.6923(0.0745)
23	0.9193(0.0439)	0.8471(0.0492)	0.9144(0.0415)	**0.9749 (0.0220)**	0.9732(0.0266)
24	**0.5508 (0.1244)**	0.5000(0.0000)	0.4953(0.0413)	0.5346(0.0738)	0.5465(0.0779)
25	••**0.6223 (0.1537)**	0.5000(0.0000)	0.5560(0.1135)	0.5579(0.1359)	0.5469(0.1234)
26	0.9960(0.0080)	0.9960(0.0080)	0.9936(0.0095)	0.9947(0.0088)	**0.9969 (0.0071)**
27	••**0.6541 (0.0837)**	0.5622(0.0581)	0.5770(0.0812)	0.5587(0.0663)	0.5423(0.0482)
28	0.7562(0.2035)	0.5586(0.0887)	**0.7650 (0.1917)**	0.7246(0.1701)	0.7608(0.1946)
29	0.9116(0.0553)	0.8976(0.0827)	0.9198(0.0569)	**0.9203 (0.0565)**	0.9144(0.0669)
30	0.8998(0.0776)	0.7213(0.1221)	0.8301(0.0866)	•**0.9744 (0.0393)**	0.9524(0.0517)
31	0.8421(0.1845)	0.5375(0.0893)	0.8421(0.1677)	**0.8450 (0.1679)**	0.8186(0.1816)
32	0.9470(0.0987)	0.8970(0.1642)	0.9710(0.0741)	0.9710(0.0741)	**0.9835 (0.0542)**
33	••**0.5652 (0.0805)**	0.5000(0.0000)	0.5028(0.0187)	0.5078(0.0365)	0.5040(0.0182)
34	0.8125(0.2174)	0.5375(0.0893)	0.8503(0.1871)	**0.8534 (0.2028)**	0.8296(0.2010)
35	**0.7810 (0.1097)**	0.7677(0.0895)	0.7731(0.0923)	0.7674(0.0892)	0.7796(0.0946)
36	••**0.7035 (0.0915)**	0.5608(0.0372)	0.5968(0.0677)	0.6126(0.0434)	0.6080(0.0498)
37	**0.5842 (0.1018)**	0.5083(0.0250)	0.5577(0.0655)	0.5348(0.0462)	0.5446(0.0756)
38	••**0.8523 (0.0577)**	0.7107(0.0859)	0.7406(0.0852)	0.7920(0.0709)	0.7869(0.0731)
39	0.8422(0.1968)	0.8463(0.1996)	0.8445(0.1989)	**0.8472 (0.2003)**	0.8463(0.1996)
40	•**0.7531 (0.0901)**	0.6376(0.0799)	0.6767(0.0970)	0.7040(0.1122)	0.6906(0.1153)
Mean	0.8081(0.0765)	0.7560(0.0548)	0.7934(0.0658)	0.8100(0.0635)	0.8067(0.0649)

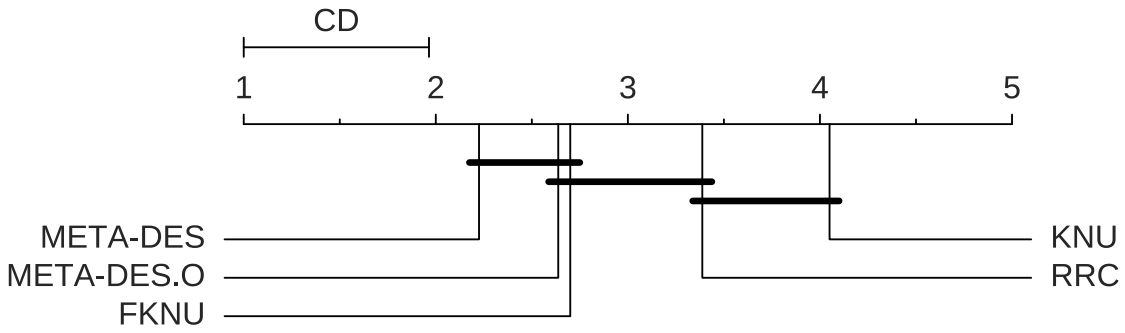


Figure 17 – CD diagram of Nemenyi post-hoc test considering FIRE-KNORA-U, RRC, META-DES, and META-DES.Oracle, where $CD = 0.9644$, and Friedman $p\text{-}value = 8.58 \times e^{-7}$.

Figure 17 shows that while KNORA-U had the worst ranking, being outperformed by all other DES techniques, and statistically outperformed by META-DES, and META-DES.Oracle. However, FIRE-KNORA-U achieved the third best ranking, outperforming RRC, and being statistically equivalent to META-DES and META-DES.Oracle.

Table 5 and Figure 17 show that the FIRE-DES framework was able to achieve equivalent performance to the state-of-the-art DES framework using a simple DES technique (KNORA-U) as dynamic selection strategy.

3.5 Conclusion

In this work, we presented 4 scenarios that dynamic selection techniques from the literature do not consider when estimating the competence of a classifier c for the classification of a test sample x_{query}: (1) x_{query} located in a safe region and c crossing the region of competence, (2) x_{query} located in a safe region and c not crossing the region of competence, (3) x_{query} located in an indecision region and c crossing the region of competence, (4) x_{query} located in an indecision region and c not crossing the region of competence. We also showed that, because they do not consider these scenarios, dynamic selection techniques from the literature can select locally incompetent classifiers for the classification of test samples located in indecision regions (i.e. classifiers that classify all samples in the region of competence as being from the same class).

In order to avoid the selection of such incompetent classifiers, we proposed the Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES) framework. The FIRE-DES framework pre-selects classifiers with decision boundaries crossing the region of competence of test samples located in indecision regions (if such classifiers exist).

The selection scheme of FIRE-DES is divided into three steps: (1) Indecision Region Detection step: identifies when the test sample is located in an indecision region;

(2) Dynamic Pruning step, pre-selects classifiers with decision boundaries within the region of competence of the test sample using the Dynamic Frienemy Pruning (DFP) method; (3) Dynamic Selection step: selects the best classifier (or classifiers) for the classification of the test sample.

Experiments were conducted using 40 datasets from KEEL, evaluating FIRE-DES using 9 dynamic selects schemes: Overall Local Accuracy (OLA), Local Class Accuracy (LCA), A Priori, A Posteriori, Multiple Classifier Behavior (MCB), Dynamic Selection KNN (DSKNN), K-Nearests Oracles Union (KNORA-U), K-Nearests Oracles Eliminate (KNORA-E), and Bagging (DFP as final selector). Experimental results show that FIRE-DES significantly improved the classification accuracy of all DS techniques in the majority of datasets. These results were confirmed by statistical tests. We also compared FIRE-KNORA-U with META-DES, META-DES.Oracle and RRC, and the results show that FIRE-DES outperformed RRC, and achieved statistically equivalent performance to META-DES and META-DES.Oracle.

Future works on this topic will involve evolving the framework for multi-class problems, adapting the framework so it can be combined with other DES frameworks, the improvement of the indecision region detection mechanism to cope with noisy data, and the development of an ensemble generation mechanism that maximizes the number of classifiers with decision boundaries within the region of competence of test samples located in indecision regions.

4 FIRE-DES++: ENHANCED ONLINE PRUNING OF BASE CLASSIFIERS FOR DYNAMIC ENSEMBLE SELECTION

FIRE-DES++: Enhanced Online Pruning of Base Classifiers for Dynamic Ensemble Selection

Dayvid V. R. Oliveira [1], George D. C. Cavalcanti [1], Rafael M. O. Cruz [2], and Robert Sabourin [2]

[1] Centro de Informática - Universidade Federal de Pernambuco, Brazil

[2] École de Technologie Supérieure - Université du Québec

Article Submitted to « Pattern Recognition »

Abstract

Dynamic Ensemble Selection (DES) techniques aim to select one or more competent classifiers for the classification of each new test sample. Most DES techniques estimate the competence of classifiers using a given criterion over the region of competence of the test sample, usually defined as the set of nearest neighbors of the test sample in the validation set. Despite being very effective in several classification tasks, DES techniques can select classifiers that classify all samples in the region of competence as being from the same class. The Frienemy Indecision REgion DES (FIRE-DES) tackles this problem by pre-selecting classifiers that correctly classify at least one pair of samples from different classes in the region of competence of the test sample. However, FIRE-DES applies the pre-selection for the classification of a test sample if and only if its region of competence is composed of samples from different classes (indecision region), even though this criterion is not reliable for determining if a test sample is located close to the borders of classes (true indecision region) when the region of competence is obtained using classical nearest neighbors approach. Because of that, FIRE-DES mistakes noisy regions for true indecision regions, leading to the pre-selection of incompetent classifiers, and mistakes true indecision regions for safe regions, leaving samples in such regions without any pre-selection. To tackle these issues, we propose the FIRE-DES++, an enhanced FIRE-DES that removes noise and reduces the overlap of classes in the validation set; and defines the region of competence using an equal number of samples of each class, avoiding selecting a region of competence with samples of a single class. Experiments are conducted using FIRE-DES++ with 8 different dynamic selection techniques on 40 classification datasets. Experimental results show that FIRE-DES++ increases the classification performance of all DES techniques considered in this work, outperforming FIRE-DES with 6 out of the 8 DES techniques,

and outperforming state-of-the-art DES frameworks.

4.1 Introduction

Dynamic Ensemble Selection (DES) has become an important research topic in the last few years (CRUZ; SABOURIN; CAVALCANTI, 2018). Given a test sample and a pool of classifiers, DES techniques select one or more competent classifiers for the classification of that test sample. The most crucial part in DES techniques is how to evaluate the competence level of each base classifier for the classification of a given test sample (CRUZ; SABOURIN; CAVALCANTI, 2016). In general, DES techniques evaluate the competence level of base classifiers for the classification of a test sample x_{query} based on the classification accuracy on a set of samples similar to x_{query}, named the region of competence of x_{query}. Most DES techniques select the region of competence of test samples using the K-Nearest Neighbors of the test sample in the validation set, we refer to this validation set as D_{SEL} (BRITTO; SABOURIN; OLIVEIRA, 2014).

Despite being very effective in several classification tasks, DES techniques can select classifiers that classify all samples in the region of competence of a test sample to the same class, even when the test sample is located close to a decision border (indecision region) (OLIVEIRA; CAVALCANTI; SABOURIN, 2017). Oliveira et al. (OLIVEIRA; CAVALCANTI; SABOURIN, 2017) proposed the Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES), a DES framework that pre-selects classifiers with decision boundaries crossing the region of competence (correctly classifying samples from different classes) when the test sample is located in an indecision region. Given a test sample x_{query}, FIRE-DES decides if x_{query} is located in an indecision region, if so, FIRE-DES uses the Dynamic Frienemy Pruning (DFP) to pre-select locally competent classifiers with decision boundaries crossing the region of competence of x_{query} (if such classifiers exist, otherwise, all classifiers are pre-selected). After the pre-selection, any DES technique can be used to perform the final selection of classifiers for the classification of the test sample.

FIRE-DES achieved interesting results, increasing the classification performance of several DES techniques, in fact, FIRE-DES using the K-Nearest Oracles Union (KNU) (KO; SABOURIN; JR, 2008) technique achieved statistically equivalent classification performance to state-of-the-art DES frameworks.

Despite being very effective in increasing the classification performance of DES techniques, the FIRE-DES does not consider whether or not the region of competence is a good representation of the type of region in which the test sample is located.

Figure 18 shows 4 test samples (x_1, x_2, x_3 and x_4) and their regions of competence of size $K = 3$ (dotted circles), where x_1 and x_4 are located in indecision regions, x_2 is located in a safe region, and x_3 is located in a noisy region; the markers "○" and "■" are

samples from different classes; Samples labeled as S are safe samples, labeled as B are borderline samples, and labeled as N are noise; and the true indecision region (region in between two classes) is delimited by the continuous line.

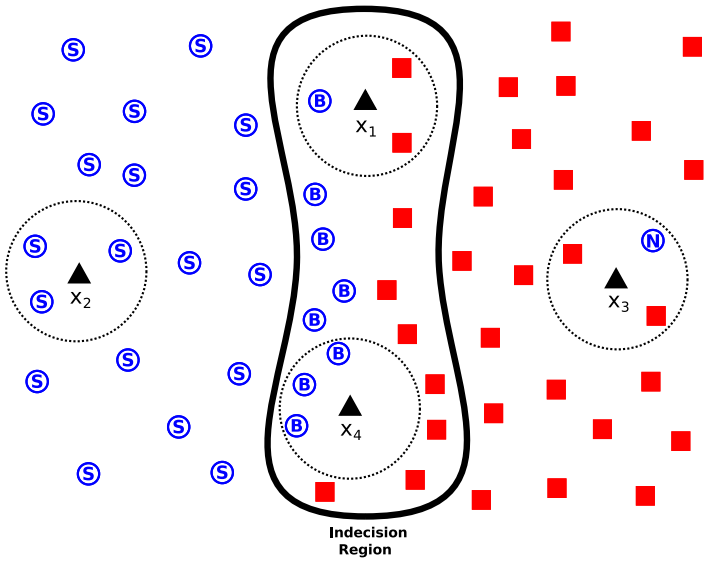


Figure 18 – Four test samples (x_1, x_2, x_3 and x_4), where x_1 and x_4 are located in indecision regions, x_2 is located in a safe region, and x_3 is located in a noisy region. The markers represent safe samples (labeled as S), borderline samples (labeled as B), and noisy samples (labeled as N). The continuous line shows the indecision region, where the classes are represented by the markers ○ and ■ (Adapted from (GARCÍA; LUENGO; HERRERA, 2015) and (OLIVEIRA; CAVALCANTI; SABOURIN, 2017)).

The test samples x_1 and x_2 from Figure 18 are located respectively in an indecision region and in a safe region. FIRE-DES correctly decides that x_1 is located in an indecision region, because the region of competence of x_1 is composed of samples from different classes. For this reason, FIRE-DES pre-selects classifiers crossing the region of competence of x_1 (if such classifiers exist). FIRE-DES also correctly decides x_2 is located in a safe region because the region of competence of x_2 is composed of samples of a single class. For this reason, FIRE-DES does not pre-select classifiers crossing the region of competence of x_2 (if such classifiers exist).

The test samples x_3 and x_4 from Figure 18 are located respectively in a noisy region and in an indecision region. FIRE-DES mistakes x_3 (located in a noisy region) as being located in an indecision region because the region of competence of x_3 is composed of samples from different classes. For this reason, FIRE-DES applies the DFP, pre-selecting classifiers that cross the region of competence of x_3 (if such classifiers exist), that is, classifiers that correctly classify the sample N, even though N is a noisy sample. Also, FIRE-DES mistakes x_4 (located in an indecision region) as being located in a safe region because the region of competence of x_4 is composed of samples of a single class. For this

reason, FIRE-DES does not apply the DFP, not pre-selecting classifiers that crosses the region of competence (and are therefore, competent for the indecision region).

The test samples x_3 and x_4 from Figure 18 exemplify 2 drawbacks of FIRE-DES: (1) **Noise Sensitivity**: FIRE-DES can mistake a noisy region for an indecision region, because it does not consider noise and outliers. (2) **Indecision Region Restriction**: FIRE-DES does not apply the DFP when the test sample has only samples of a single class in its region of competence, even if is located in a true indecision region. A test sample is located in a true indecision region when it is located close to the decision borders of classes, regardless of the classes represented in its region of competence.

The noise sensitivity drawback is specially important because DES techniques are highly sensitive to noise, outliers, and high level of overlap between classes in D_{SEL} (CRUZ; CAVALCANTI; REN, 2011). Cruz et al. (CRUZ; SABOURIN; CAVALCANTI, 2016) proposed the use of Prototype Selection (PS) (GARCIA et al., 2012) to remove noise from the validation set and achieved interesting results, where the proposed approach improved the classification performance of all DES techniques considered in their work. Using PS in the validation set is a good approach to attenuate the noise sensitivity problem of DES techniques, but for FIRE-DES, PS techniques can remove important samples, making indecision regions look like safe regions. For this reason, using PS with FIRE-DES is a promissing approach, but a second step is required so that the number of samples in which the DFP is applied is not reduced.

The indecision region restriction drawback limits the DFP to test samples classified as being located in indecision regions based on the samples that compose their regions of competence. The problem is that D_{SEL} is not always a good representation of the distribution of classes, meaning that a test sample can have a region of competence composed of samples from a single class (being classified as located in a safe region) when it is in fact located close to the decision borders, that is, a true indecision region (i.e. x_4 from Figure 18).

The K-Nearest Neighbors Equality (KNNE) (SIERRA et al., 2011) is a KNN approach that selects the K nearest neighbors from each class, and each sample submits a vote (weighted by the inverse of the distance to the test sample) for the classification of the test sample. Using the KNNE to select the region of competence of the test sample avoids the selection of a region of competence composed of samples from a single class, ensuring that all test samples are classified as being located in an indecision region, and the DFP is applied to all test samples. Mendialdua et al. (MENDIALDUA et al., 2015) proposed a framework that uses KNNE to select the region of competence of test samples. However, their framework does not consider that the KNNE can include noise in the region of competence, and the framework is also limited to the Overall Local Accuracy (OLA) (WOODS; KEGELMEYER; BOWYER, 1997) DES technique, and the use of

hybrid ensembles composed of classifiers from different natures.

In this paper, we propose the FIRE-DES++, an enhanced FIRE-DES framework that tackles the noise sensitivity drawback and indecision region restriction drawback. Like FIRE-DES, FIRE-DES++ can be used with any dynamic selection technique that uses the nearest neighbors to estimate the competence level of base classifiers.

The FIRE-DES++ is composed of four phases: (1) Overproduction, where the pool of classifiers C is generated using the training set $\mathcal{T}$. (2) Filtering phase, where a PS technique is applied to the validation set $\mathcal{D}_{SEL}$ in order to remove noise and outliers, and reduce the level of overlap between classes in $\mathcal{D}_{SEL}$. (3) Region of competence definition (RoCD) phase, where FIRE-DES++ defines the region of competence using the KNNE, selecting an equal number of samples from each class from $\mathcal{D}_{SEL}$, avoiding the definition of a region of competence with samples of a single class. (4) Selection phase, where FIRE-DES++ pre-selects base classifiers with decision boundaries crossing the region of competence (if such classifiers exist) using the DFP (OLIVEIRA; CAVALCANTI; SABOURIN, 2017), avoiding the selection of classifiers that classify all samples in the region of competence as being from the same class. After the pre-selection, any DES technique is applied to perform the final selection.

In the experiments, we evaluated FIRE-DES++ with 8 dynamic selection techniques from the literature over 40 datasets from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository (ALCALÁ et al., 2010). We also compared FIRE-DES++ using the better performing dynamic selection technique with 3 state-of-the-art DES approaches, named: Randomized Reference Classifier (RRC) (WOLOSZYNSKI; KURZYNSKI, 2011), META-DES (CRUZ et al., 2015), and META-DES.Oracle (CRUZ; SABOURIN; CAVALCANTI, 2017b). The results showed that, in classification performance, FIRE-DES++ outperformed DES in all 8 DES techniques, FIRE-DES++ outperformed FIRE-DES in 6 out of 8 DES techniques, and FIRE-DES++ statistically outperformed all 3 state-of-the-art techniques. These results were confirmed by the Wilcoxon Signed Rank Test (WILCOXON, 1945), Sign Test (DEMŠAR, 2006), Friedman test (FRIEDMAN, 1937), and Nemenyi post hoc test (NEMENYI, 1962).

This paper is organized as follows: Section 2 presents the problem statement, Section 3 presents the proposed framework, Section 4 presents the experimental study, and Section 5 concludes the paper.

4.2 Problem Statement

Dynamic Ensemble Selection (DES) techniques aim to select only the most competent classifiers for the classification of each test sample. However, DES techniques can select classifiers that classify all samples as being from the same class, even when the test

sample is located close to the decision border of classes. The Frienemy Indecision Region DES (FIRE-DES) handles this problem, avoiding the selection of locally incompetent classifiers when the test sample is located close to the decision borders. Given a test sample x_{query}, FIRE-DES decides if x_{query} is located in an indecision region (when its region of competence is composed of samples from different classes), if so, it uses the Dynamic Frienemy Pruning (DFP) technique to pre-select classifiers with decision boundaries crossing the region of competence (correctly classifying samples from different classes), and then FIRE-DES uses a DES technique to perform the final selection. If the test sample is located in an indecision region and no classifier crosses the region of competence, all classifiers are pre-selected by the DFP.

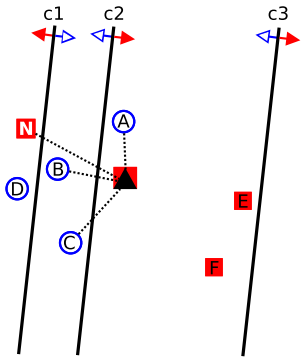
FIRE-DES achieved interesting results, outperforming all DES techniques considered in (OLIVEIRA; CAVALCANTI; SABOURIN, 2017) in classification performance, achieving statistically equivalent classification performance to the current state-of-art in DES frameworks. However, in this work we identified two drawbacks of the FIRE-DES framework: the noise sensitivity drawback and the indecision region restriction drawback. These drawbacks are detailed in the following subsections.

4.2.1 Drawback 1: Noise Sensitivity

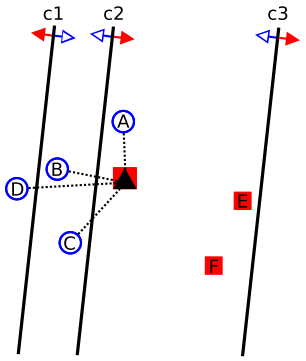
Figure 19(a) shows a test sample (▲) with true class ■ located in a noisy region, and three classifiers $c1$, $c2$, and $c3$. In this figure, the region of competence (Ψ) of the test sample is composed of the samples A, B, C, and N (sample N is noise). In the example from Figure 19(a), the classifier $c1$ correctly classifies 4 samples in Ψ (A, B, C, and N), the classifier $c2$ correctly classifies 2 samples in Ψ (B, and C), and the classifier $c3$ correctly classifies 3 samples in Ψ (A, B, and C).

The Overall Local Accuracy (OLA) (WOODS; KEGELMEYER; BOWYER, 1997) DES technique estimates the competence of classifiers using their accuracy in the region of competence, that is, the more samples a classifier correctly classifies, the more competent it is. OLA selects only the most competent classifier for the classification of the test sample.

In Figure 19(a), OLA selects $c1$ (the classifier that correctly classify most samples in Ψ) even though $c1$ was only considered the best because of a noisy sample (N). This selection leads to the misclassification of the test sample as ○. Also in this example, the FIRE-DES will mistake the noisy region (region with noisy samples) for an indecision region (region composed of samples from different classes), and pre-select classifiers that correctly classify at least one pair of samples from different classes (frienemies), in this case $c1$, also misclassifying the test sample as ○.



(a) Toy problem of noisy region of competence (A, B, C, and N), the markers ○ (A, B, C, and D) and ■ (N, E, and F) are samples of different classes, the sample labeled N is a noisy sample.



(b) Toy problem of a test sample ▲ and a filtered - noisy sample N was removed - region of competence (A, B, C, and D), the markers ○ (A, B, C, and D) and ■ (E, and F) are samples of different classes.

Figure 19 – DES applied to the classification of a test sample ▲ of class ■. The continuous straight lines are the decision boundaries of classifiers $c1$, $c2$, and $c3$, the markers ○ (A, B, C, and D) and ■ (N, E, and F) are samples of different classes, N is a noisy sample, and samples connected to the test sample by a dotted line define the region of competence of the test sample.

4.2.2 Drawback 2: Indecision Region Restriction

Figure 19(b) shows the scenario from Figure 19(a) without the noisy sample N. Figure 19(b) shows a test sample (▲) with true class ■ located in a true indecision region (close to the borders), and three classifiers $c1$, $c2$, and $c3$. In this figure, the region of competence (Ψ) of the test sample is composed of the samples A, B, C, and D all from class ○. In the example from Figure 19(b), the classifier $c1$ correctly classify 3 samples in Ψ (A, B, and C), the classifier $c2$ correctly classify 2 samples in Ψ (B, and C), and the classifier $c3$ correctly classify 4 samples in Ψ (A, B, C, and D).

In Figure 19(b), OLA selects the classifier that correctly classify the most samples in Ψ, that is, $c3$, even though $c3$ classify all samples in the region of competence of the test sample as being from the same class ○, misclassifying the test sample.

In the example from Figure 19(b), the FIRE-DES does not apply the DFP because it considers x_{query} as being located in a safe region, even though it is located in a true indecision region. Therefore, FIRE-DES with OLA also misclassifies the test sample as being from the class ○.

4.3 The proposed framework

In this section we propose an enhanced Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES++). FIRE-DES++ is divided into four phases (Figure 20):

1. **Overproduction phase**, where the pool of classifiers C is generated using the training set ($\mathcal{T}$). The overproduction phase is performed only once in the training stage.

2. **Filtering phase**, where a Prototype Selection (PS) (GARCIA et al., 2012) technique is applied to the validation set $\mathcal{D}_{SEL}$, removing noise and outliers, and reducing the level of overlap between classes in $\mathcal{D}_{SEL}$. The improved validation set is named $\mathcal{D}'_{SEL}$. The filtering phase is performed only once in the training stage.

3. **Region of competence definition (RoCD) phase**, there the framework defines the region of competence (Ψ) using the K-Nearest Neighbors Equality (KNNE) (SIERRA et al., 2011) to select samples from the improved validation set $\mathcal{D}'_{SEL}$. The KNNE is a nearest neighbor approach that selects an equal number of samples from each class, avoiding the definition of a region of competence with samples of a single class. The RoCD phase is performed in the testing stage for each new test sample.

4. **Selection phase**, where the ensemble of classifiers for the classification of each new test sample is selected. Given a test sample x_{query}, this phase pre-selects base classifiers with decision boundaries crossing the region of competence of x_{query} (C_{pruned}), if such classifier exists, using the Dynamic Frienemy Pruning (DFP) (OLIVEIRA; CAVALCANTI; SABOURIN, 2017). The DFP pre-selects classifiers that correctly classify at least one pair of samples from different classes ("frienemies") in the region of competence. The DFP avoids the selection of classifiers that classify all samples in the region of competence as being from the same class. After the pre-selection, any DES technique is applied to perform the final selection (C'). Finally, the framework uses a combination rule to combine the predictions of the selected classifiers into a single prediction.

In Figure 20, $\mathcal{T}$ is the training set, *Generation* is an ensemble generation process (i.e. Bagging (BREIMAN, 1996)), and C is the generated pool of classifiers; $\mathcal{G}$ is the test set, x_{query} is the test sample; $\mathcal{D}_{SEL}$ is the validation set, *Filtering* is the process of filtering $\mathcal{D}_{SEL}$ using a prototype selection algorithm which results in the improved validation set $\mathcal{D}'_{SEL}$, *Region of Competence Definition* is the process of selecting the region of competence of x_{query} using $\mathcal{D}'_{SEL}$, Ψ is the region of competence of x_{query}; *Dynamic Frienemy Pruning* is the Dynamic Pruning step, C_{pruned} is the pre-selected ensemble of

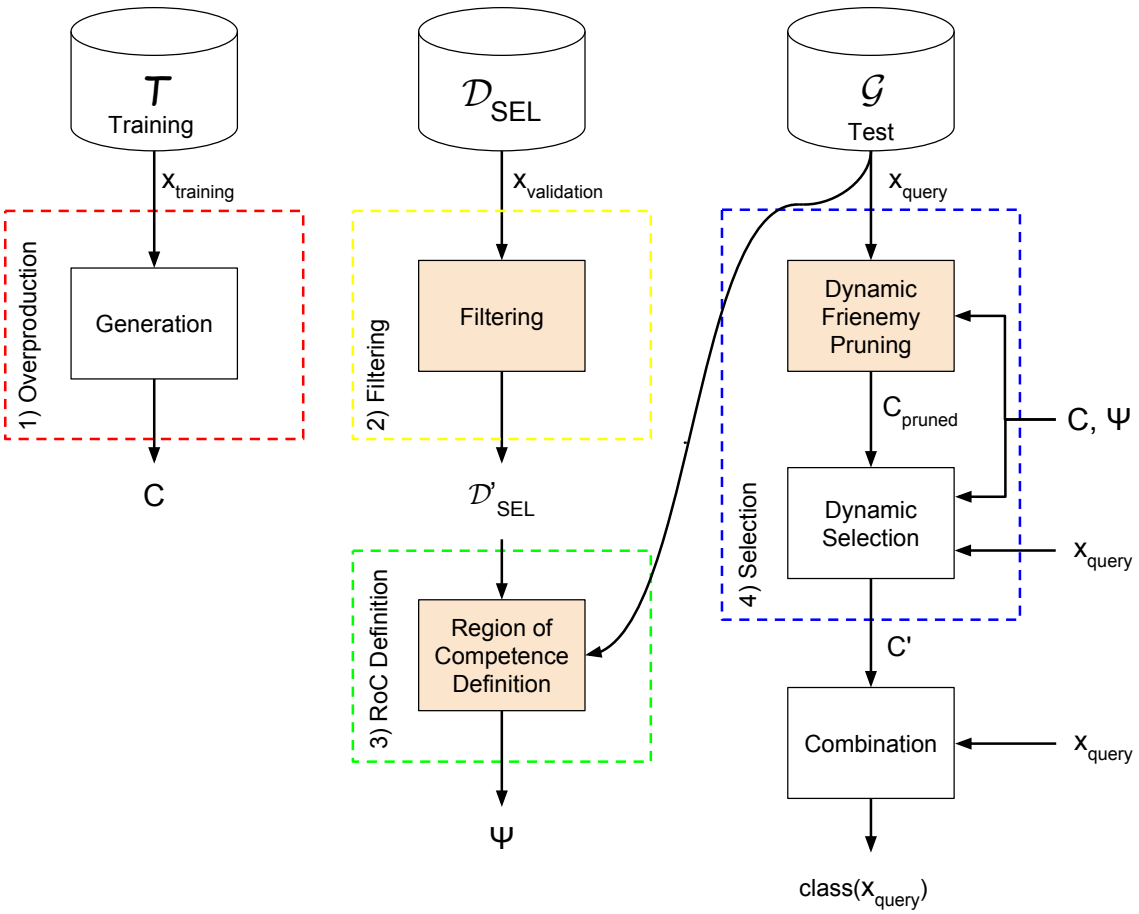


Figure 20 – Overview of FIRE-DES++, where $\mathcal{G}$ is the test set, x_{query} is the test sample, $\mathcal{T}$ is the training set, *Generation* is an ensemble generation process (i.e. Bagging) used to generate the pool of classifiers C, $\mathcal{D}_{SEL}$ is the validation set, *Filtering* is the process of filtering $\mathcal{D}_{SEL}$ using a prototype selection algorithm which results in the improved validation set $\mathcal{D}'_{SEL}$, *Region of competence definition (RoCD)* is the process of selecting the region of competence Ψ of x_{query} with size K, *Dynamic Frienemy Pruning* is the Dynamic Frienemy Pruning (DFP) step, *Dynamic Selection* is the Dynamic Selection step, C_{pruned} is the set of pre-selected classifiers, C' is the ensemble of selected classifiers for the classification of x_{query}, *Combination* is a combination rule, and *class(x_{query})* is the final classification of x_{query}.

classifiers, *Dynamic Selection* is the Dynamic Selection step; C' is the ensemble of selected classifiers, *Combination* is the process of combining the prediction of the classifiers in C', and *class(x_{query})* is the final prediction of x_{query}.

The phases of FIRE-DES++ complement each other as the filtering phase tackles the noise sensitivity drawback, removing noise and reducing the level of overlap between classes; the region of competence definition phase tackles the indecision region restriction drawback, as it ensures that all classes are represented in the region of competence of the test sample; and, finally, the selection phase pre-selects classifiers with decision boundaries crossing the region of competence, without having to consider the effect of noise (since noise is removed in the filtering phase), neither with deciding if a test sample is located in an indecision region or not (as the region of competence definition phase always selects regions of competence composed of samples of different classes).

The phases of FIRE-DES++ are detailed in the following subsections.

4.3.1 Overproduction

The overproduction phase uses any ensemble generation technique to generate the pool of classifiers C trained with the training set $\mathcal{T}$. Since the focus of this work is on dynamic selection, following the approach used in (OLIVEIRA; CAVALCANTI; SABOURIN, 2017), for simplicity, we use the Bagging technique (BREIMAN, 1996) (SKURICHINA; DUIN, 1998) to generate the pool of classifiers.

4.3.2 Filtering phase

The filtering phase tackles the noisy sensitivity drawback (Section 4.2.1), as removing noise from $\mathcal{D}_{SEL}$, preventing FIRE-DES from estimating the competence level of base classifiers using noisy data. We expect the filtering phase to cause a high performance gain to the FIRE-DES++, as in (CRUZ; SABOURIN; CAVALCANTI, 2016), the authors show that state-of-the-art techniques fail to obtain a good approximation of the decision boundaries of classes when noise is added to $\mathcal{D}_{SEL}$, and also demonstrate that using PS increases the classification performance of DES techniques.

In the filtering phase, FIRE-DES++ applies a PS technique to the validation set ($\mathcal{D}_{SEL}$), resulting in an improved validation set ($\mathcal{D}'_{SEL}$) with less noise, and less overlap between classes.

In (GARCIA et al., 2012), the authors presented a taxonomy of prototype selection, classifying prototype selection techniques in three categories: (1) Condensation techniques, that remove samples in the center of classes, maintaining the borderline samples. (2) Edition techniques, that remove sample in the borders of classes, maintaining safe samples

(located in the center of classes). (3) Hybrid techniques, combines condensation and edition approaches.

In (CRUZ; SABOURIN; CAVALCANTI, 2017a), the authors compared the use of several PS techniques applied on the validation set for dynamic selection purposes, and showed that using Relative Neighborhood Graph (RNG) (SÁNCHEZ; PLA; FERRI, 1997) was the best approach, followed by the Edited Nearest Neighborhood (ENN) (WILSON, 1972).

Since our experimental study is focused on small datasets with different levels of class imbalance, only samples of the majority class are removed from the validation set.

4.3.2.1 *Relative Neighborhood Graph (RNG)*

The RNG technique uses the concept of Proximity Graph (PG) to select prototypes. RNG builds a PG, G = (V, E), in which the vertices are samples (V = $\mathcal{D}_{SEL}$) and the set of edges E contains an edge connecting two samples (x_i, x_j) if and only if (x_i, x_j) satisfy the neighborhood criterion in Equation 4.1:

$$(x_i, x_j) \in E \Leftrightarrow dist(x_i, x_j) \leq max(dist(x_i, x_k), dist(x_j, x_k)) \quad \forall x_k \in X, k = i, j \tag{4.1}$$

where $dist$ is the Euclidean distance between two samples, and X is the set of samples in the validation set. The corresponding geometric is defined as the disjoint intersection between two hyperspheres centered in x_i and x_j, and radius equal to $dist(x_i, x_j)$. Two samples are relative neighbors if and only if this intersection does not contain any other sample from $\mathcal{D}_{SEL}$. The relative neighborhood of a sample is the set of all its relative neighbors. After building the PG and defining all graph neighbors, all samples with class label different from the majority of their respective relative neighbors are removed from $\mathcal{D}_{SEL}$.

Algorithm 8 presents the pseudo-code of the RNG technique used in this work. Given the validation set $\mathcal{D}_{SEL}$ (Line 1), all samples are added in the filtered validation set $\mathcal{D}'_{SEL}$ (Line 2), and the proximity graph of the samples in $\mathcal{D}_{SEL}$ are stored in PG (Line 3). Now, for each sample $x_i \in \mathcal{D}_{SEL}$, the relative neighbors (RN) of x_i are selected, and, if the most common class label in RN is different from the class label of x_i, and x_i is not from the minority class, x_i is removed from the filtered validation set $\mathcal{D}'_{SEL}$ (Line 3 - 10). Finally, the filtered validation set $\mathcal{D}'_{SEL}$ is returned (Line 11).

4.3.2.2 *Edited Nearest Neighbors (ENN)*

The ENN is an edition prototype selection technique well-known for its efficiency in removing noise and producing smoother classes boundaries. Due its simplicity and efficiency,

Algorithm 8 Relative Neighborhood Graph (RNG)

Require: $\mathcal{D}_{SEL}$: validation set
1: $\mathcal{D}'_{SEL} \leftarrow \mathcal{D}_{SEL}$
2: $PG \leftarrow$ *proximity-graph*($\mathcal{D}_{SEL}$)
3: **for all** $x_i \in \mathcal{D}_{SEL}$ **do**
4: $RN \leftarrow$ *relative-neighbors*(x_i, PG)
5: $label_{pred} \leftarrow$ most frequent class in RN
6: $label_{true} \leftarrow class(x_i)$
7: **if** $label_{true} \neq label_{pred} \wedge label_{true} \neq minority_{class}$ **then**
8: $\mathcal{D}'_{SEL} \leftarrow \mathcal{D}'_{SEL} \setminus x_i$
9: **end if**
10: **end for**
11: **return** $\mathcal{D}'_{SEL}$

in this work we use the ENN in the filtering phase, however, with the changes proposed in (LAURIKKALA, 2001), (implemented in (LEMAITRE; NOGUEIRA; ARIDAS, 2017)), where only majority class samples are removed in order to reduce the class imbalance.

Algorithm 9 presents the pseudo-code of the ENN technique used in this work. Given the validation set $\mathcal{D}_{SEL}$ (Line 1), all samples are added in the filtered validation set $\mathcal{D}'_{SEL}$ (Line 2), and for each sample $x_i \in \mathcal{D}_{SEL}$, if x_i is misclassified by its K nearest neighbors in $\mathcal{D}'_{SEL} \backslash x_i$ and x_i is not from the minority class, x_i is removed from the filtered validation set $\mathcal{D}'_{SEL}$ (Line 3 - 9). Finally, the filtered validation set $\mathcal{D}'_{SEL}$ is returned (Line 10).

Algorithm 9 Edited Nearest Neighbors (ENN)

Require: $\mathcal{D}_{SEL}$: validation set
1: $\mathcal{D}'_{SEL} \leftarrow \mathcal{D}_{SEL}$
2: **for all** $x_i \in \mathcal{D}_{SEL}$ **do**
3: $label_{pred} \leftarrow$ most frequent class in $KNN(x_i, \mathcal{D}_{SEL} \setminus x_i)$
4: $label_{true} \leftarrow class(x_i)$
5: **if** $label_{true} \neq label_{pred} \wedge label_{true} \neq minority_{class}$ **then**
6: $\mathcal{D}'_{SEL} \leftarrow \mathcal{D}'_{SEL} \setminus x_i$
7: **end if**
8: **end for**
9: **return** $\mathcal{D}'_{SEL}$

4.3.3 Region of competence definition phase

The region of competence definition phase of FIRE-DES tackles the indecision region restriction drawback (Section 4.2.2) of FIRE-DES, as it ensures that all classes are represented in the region of competence (since all test samples have samples of different

classes in the region of competence).

The nearest neighbor (NN) rule (COVER; HART, 1967) is one of the most well known supervised learning techniques, the general idea is to classify a test sample as being of the same class as its nearest sample from the training set. The K-nearest neighbor rule (KNN) (PATRICK; FISCHER, 1969) is an extension of the NN rule in which the test sample is classified as being from the most frequent class in the K nearest samples of the test sample in the training set. The KNN is very simple, and yet, is one of the most interesting and useful algorithm in pattern recognition (SHAKHNAROVICH; DARRELL; INDYK, 2005a).

In the context of dynamic selection of classifiers, the KNN is used to define the region of competence of the test sample which is used to evaluate the competence of base classifiers (BRITTO; SABOURIN; OLIVEIRA, 2014). The rationale behind this approach is that classifiers that are competent to classify a test sample x_{query} are the ones that are competent to classify samples that are similar to x_{query} (that is, the nearest neighbors of x_{query}).

In (MENDIALDUA et al., 2015), the authors proposed using the K-Nearest Neighbors Equality (KNNE) (SIERRA et al., 2011) to select the region of competence of test samples for the Overall Local Accuracy (OLA) (WOODS; KEGELMEYER; BOWYER, 1997) technique in a One versus One binarization approach. The KNNE selects the K nearest neighbors of the test sample for each class, and classifies the test sample as being from the class with lower mean distance.

In this work, we propose using the KNNE to select the region of competence of test samples. This region of competence is later used in the selection phase to pre-select classifiers with decision boundaries crossing the region of competence, and to perform the final selection of classifiers for the classification of each new test sample.

4.3.4 Selection phase

In the selection phase, first, the framework pre-selects classifiers using the DFP, and then, it uses a dynamic selection technique from the literature to perform the final selection of classifiers for each new test sample.

4.3.4.1 *Dynamic frienemy pruning*

The Dynamic Frienemy Pruning (DFP) (OLIVEIRA; CAVALCANTI; SABOURIN, 2017) aims to pre-select competent classifiers (classifiers with decision boundaries crossing the region of competence) for the classification of each new test sample, before the final selection of classifiers. The DFP algorithm uses the *frienemy samples* concept: Given a test sample x_{query} and its region of competence Ψ, two samples Ψ_a and Ψ_b are frienemy

samples in regards to x_{query} if, Ψ_a is in Ψ, Ψ_b is in Ψ, and Ψ_a and Ψ_b are from different classes. Figure 21 shows a test sample ▲ and its region of competence (samples A, B, C, D and E). In this example, the frienemy samples are the pairs of samples of opposite classes (○, ■), named (A, C), (A, D), (A, E), (B, C), (B, D), (B, E).

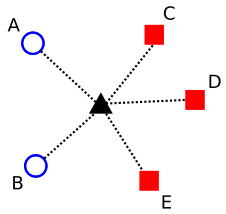


Figure 21 – Pairs of frienemies (A, C), (A, D), (A, E), (B, C), (B, D), (B, E) in the region of competence of the test sample ▲. Image adapted from (OLIVEIRA; CAVALCANTI; SABOURIN, 2017).

For each new test sample, if the test sample is located in an indecision region, the DFP algorithm pre-selects classifiers with decision boundaries crossing the region of competence. That is, if the test sample have samples of different classes in the region of competence, DFP pre-selects classifiers that correctly classify at least one pair of frienemy samples (if such classifier exists).

Algorithm 10 presents the DFP pseudo-code. Given the region of competence (Ψ) of the test sample, and the pool of classifiers (C), DFP creates an empty list C_{pruned} in which the pre-selected classifiers will be stored (Line 1), finds the pairs of frienemy samples ($\mathcal{F}$) in Ψ (Line 2), and, for each classifier c in C, c is included in C_{pruned} if c correctly classify at least one pair of frienemies (Lines 3 - 8). If no classifier is pre-selected, DFP includes all classifiers in C into C_{pruned} (lines 9 - 11). Finally, C_{pruned} is returned (Line 12).

4.3.5 Dynamic Selection

Section 4.2 presented the problems of FIRE-DES when the test sample is located in a noisy region (Subsection 4.2.1) and when the test sample has samples of a single class in its region of competence (Subsection 4.2.2).

Figure 22 shows the same scenario from Figure 19 (but without the noisy sample N and using the KNNE to define the region of competence of the test sample). Figure 22 shows a test sample (▲) with true class ■, and three classifiers $c1$, $c2$, and $c3$, and region of competence (Ψ) composed of the samples A, B, E, and F.

Algorithm 10 Dynamic Frienemy Pruning

Require: Ψ: region of competence of the test sample
Require: $\mathcal{C}$: pool of classifiers
1: $C_{pruned} \leftarrow$ empty ensemble of classifiers
2: $\mathcal{F} \leftarrow$ all pair of frienemies in Ψ
3: **for all** c_i in $\mathcal{C}$ **do**
4: $\quad \mathcal{F}_i \leftarrow$ pairs of samples in $\mathcal{F}$ correctly classified by c_i.
5: $\quad$ **if** $\mathcal{F}_i$ is not empty **then**
6: $\quad\quad C_{pruned} \leftarrow C_{pruned} \cup c_i$
7: $\quad$ **end if**
8: **end for**
9: **if** C_{pruned} is empty **then**
10: $\quad C_{pruned} \leftarrow \mathcal{C}$
11: **end if**
12: **return** C_{pruned}

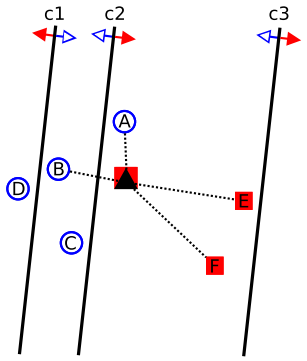


Figure 22 – DES applied to the classification of a test sample ▲ of class ■. The continuous straight lines are the decision boundaries of classifiers $c1$, $c2$, and $c3$, the markers ○ (A, B, C, and D) and ■ (E, and F) are samples of different classes, and samples connected to the test sample by a dotted line (A, B, E, and F) define the region of competence of the test sample.

Figure 22 shows the region of competence considered by FIRE-DES++ when applied to the problem from Figure 19. First, the FIRE-DES++ removes noise from the validation set (the example from Figure 19(a) is turned into the example from Figure 19(b)), tackling the noisy sensitivity drawback of FIRE-DES. Then, the framework uses the KNNE to define the region of competence of the test sample, selecting samples from different classes (the example from Figure 19(b) is turned into the example from Figure 22), tackling the indecision region restriction drawback of FIRE-DES. In the example from Figure 22, the classifier $c1$ now correctly classifies 2 samples in Ψ, the classifier $c2$ now correctly classifies 3 samples in Ψ, and the classifier $c3$ now correctly classifies 2 samples in Ψ. OLA now selects $c2$ correctly classifying the test sample.

By applying the DFP in this example (after the PS technique and the KNNE),

FIRE-DES++ pre-selects the classifier $c2$ as it is the only classifier that correctly classify at least one pair of frienemies, correctly classifying the test sample as being from the class ■. In this example, FIRE-DES++ performed optimal classification for OLA and the same concept can be extended to other DES techniques.

4.4 Experiments

In this section, we evaluate FIRE-DES++ using different dynamic selection techniques, and the impact of the filtering phase using the Edited Nearest Neighbors (ENN), the region of competence definition phase using the K-Nearest Neighbors Equality (KNNE), and the selection phase, using the Dynamic Frienemy Pruning (DFP). We also compare the filtering phase using the ENN and the Relative Neighborhood Graph (RNG).

In our evaluation, we used 8 dynamic classifier selection techniques from the literature. We also compare FIRE-DES++ (using the DES technique that achieved the highest classification performance) with state-of-the-art DES approaches: Randomized Reference Classifier (RRC) (WOLOSZYNSKI; KURZYNSKI, 2011), META-DES (CRUZ et al., 2015), and META-DES.Oracle (CRUZ; SABOURIN; CAVALCANTI, 2017b). These 3 state-of-the-art techniques have achieved the best classification performance in (CRUZ; SABOURIN; CAVALCANTI, 2018).

The RRC technique does not use the KNN to define the region of competence, rather, it uses the entire validation set to select the most competent classifier using a potential function model, therefore, neither the KNNE nor the DFP steps in the proposed framework can be applied to this technique. The META-DES and META-DES.Oracle use meta-features based on the distance between the test sample and its neighbors, for this reason, the KNNE can not be properly used, also, the META-DES and META-DES.Oracle require an extra data partition (meta-training dataset) for training the meta-classifiers, deviating from the experimental protocol of the other techniques used in this experiment.

4.4.1 Dynamic Selection Techniques

We considered a total of 8 dynamic selection techniques to be considered in the proposed framework: Overall Local Accuracy (OLA), Local Class Accuracy (LCA), A Priori (APRI), A Posteriori (APOS), Multiple Classifier Behavior (MCB), Dynamic Selection KNN (DSKNN), K-Nearest Oracles Union (KNU), and K-Nearest Oracles Eliminate (KNE).

Table 6 presents the 8 dynamic selection techniques considered in this work, 3 state-of-the-art frameworks, their category according to the taxonomy in (CRUZ; SABOURIN; CAVALCANTI, 2018), and their original references.

Table 6 – Dynamic selection techniques considered in the experiments.

Technique	Category	Reference
DCS		
Overall Local Accuracy (OLA)	Accuracy	Woods et al. (WOODS; KEGELMEYER; BOWYER, 1997)
Local Class Accuracy (LCA)	Accuracy	Woods et al. (WOODS; KEGELMEYER; BOWYER, 1997)
A Priori (APri)	Probabilistic	Giacinto et al. (GIACINTO; ROLI, 1999)
A Posteriori (APos)	Probabilistic	Giacinto et al. (GIACINTO; ROLI, 1999)
Multiple Classifier Behavior (MCB)	Behavior	Giacinto et al. (GIACINTO; ROLI, 2001b)
DES		
Dynamic Selection KNN (DSKNN)	Diversity	Santana et al. (SANTANA et al., 2006)
K-Nearests Oracles Union (KNU)	Oracle	Ko et al. (KO; SABOURIN; JR, 2008)
K-Nearests Oracles Eliminate (KNE)	Oracle	Ko et al. (KO; SABOURIN; JR, 2008)
State-of-the-art		
Randomized Reference Classifier (RRC)	Probabilistic	Woloszynski et al. (WOLOSZYNSKI; KURZYNSKI, 2011)
META-DES	Meta-learning	Cruz et al. (CRUZ et al., 2015)
META-DES.Oracle	Meta-learning	Cruz et al. (CRUZ; SABOURIN; CAVALCANTI, 2017b)

The parameters were set following the strategy used in (CRUZ et al., 2015) and (CRUZ; SABOURIN; CAVALCANTI, 2016). For each data partition, a pool of classifiers C of 100 Perceptrons was generated using the Bagging technique (BREIMAN, 1996). For all dynamic selection techniques, the region of competence size (K) was set to $K = 7$. The parameters specific to individual dynamic selection techniques were set using the default values within the range specified in the literature.

4.4.2 Datasets

We conducted the experiments on 40 datasets from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository (ALCALÁ et al., 2010). This experimental study is focused on small datasets with different levels of class imbalance, since dynamic selection techniques have been shown to be effective on small datasets (CAVALIN; SABOURIN; SUEN, 2013), and ensemble learning has recently become popular in dealing with imbalanced datasets (GALAR et al., 2012) (NANNI; FANTOZZI; LAZZARINI, 2015) (GALAR et al., 2016). Table 7 shows the characteristics of the datasets used in this experiment: label, name, number of features, number of samples, and imbalance ratio (IR).

Table 7 – Characteristics of the 40 datasets used in the experiments: label, name, number of features, number of samples, and imbalance ratio. The imbalance ratio (IR) is the ratio between the number of majority class samples and minority class samples.

Label	Name	#Feats.	#Samples	IR
1	glass1	9	214	1.82
2	ecoli0vs1	7	220	1.86
3	wisconsin	9	683	1.86
4	pima	8	768	1.87
5	iris0	4	150	2.00
6	glass0	9	214	2.06
7	yeast1	8	1484	2.46
8	vehicle2	18	846	2.88
9	vehicle1	18	846	2.90
10	vehicle3	18	846	2.99
11	glass0123vs456	9	214	3.20
12	vehicle0	18	846	3.25
13	ecoli1	7	336	3.36
14	new-thyroid1	5	215	5.14
15	new-thyroid2	5	215	5.14
16	ecoli2	7	336	5.46
17	segment0	19	2308	6.00
18	glass6	9	214	6.38
19	yeast3	8	1484	8.10
20	ecoli3	7	336	8.60
21	yeast-2vs4	8	514	9.08
22	yeast-05679vs4	8	528	9.35
23	vowel0	13	988	9.98
24	glass-016vs2	9	192	10.29
25	glass2	9	214	11.59
26	shuttle-c0vsc4	9	1829	13.87
27	yeast-1vs7	7	459	14.30
28	glass4	9	214	15.47
29	ecoli4	7	336	15.80
30	page-blocks-13vs4	10	472	15.86
31	glass-0-1-6_vs_5	9	184	19.44
32	shuttle-c2-vs-c4	9	129	20.50
33	yeast-1458vs7	8	693	22.10
34	glass5	9	214	22.78
35	yeast-2vs8	8	482	23.10
36	yeast4	8	1484	28.10
37	yeast-1289vs7	8	947	30.57
38	yeast5	8	1484	32.73
39	ecoli-0137vs26	7	281	39.14
40	yeast6	8	1484	41.40

4.4.3 Evaluation

For each dataset, the experiments were carried out using 20 replications. Each replication divided the samples using 20% as test set ($\mathcal{G}$), 20% as validation set ($\mathcal{D}_{SEL}$), and 60% as training set ($\mathcal{T}$). The datasets were partitioned using *stratified 5-fold cross-validation* (1 fold for testing, 4 folds for validation/training) followed by a *stratified 4-fold cross-validation* (the 4 folds in validation/training divided in 3 folds for training and 1 for validation).

For evaluation metric, we used the Area Under the ROC Curve (AUC) (BRADLEY, 1997). We used the AUC because this metric has been widely used to evaluate the performance of classifiers on imbalanced data (LÓPEZ et al., 2013).

For performance comparison, we used the Wilcoxon Signed Rank Test (WILCOXON, 1945), the Sign Test (DEMŠAR, 2006), The Friedman test (FRIEDMAN, 1940), and the Nemenyi post-hoc test (NEMENYI, 1962).

4.4.4 Filtering Phase: RNG vs. ENN

In (CRUZ; SABOURIN; CAVALCANTI, 2017a), the authors compared the performance of different prototype selection techniques applied on dynamic selection of classifiers and the results showed that edition techniques, named Relative Neighborhood Graph (RNG) (SÁNCHEZ; PLA; FERRI, 1997) and Edited Neighborhood Neighborhood (ENN) (WILSON, 1972), achieved respectively the best and second best classification performance. We evaluate FIRE-DES++ using RNG and ENN in the filtering phase, following the approach of maintaining all samples of the minority class. This comparison is important for verifying if FIRE-DES++ is sensitive to changes of PS techniques in the filtering phase, and also for finding the PS technique that causes the highest classification performance gain in FIRE-DES++.

Figure 23 shows the scatter plot of average AUC of FIRE-DES++ using the ENN (vertical axis) and the RNG (horizontal axis). In this figure, all markers are above the diagonal line, meaning that using the ENN was, on average, better than using the RNG for all DES techniques in the proposed framework.

Using the Wilcoxon Signed Rank Test ($\alpha = 0.10$), we can confirm that using the proposed framework with the ENN is statistically better than using the RNG for all techniques: OLA ($p\text{-}value = 0.0121$), LCA ($p\text{-}value = 0.0011$), APRI ($p\text{-}value = 0.0040$), APOS ($p\text{-}value = 0.0946$), MCB ($p\text{-}value = 0.0007$), DSKNN ($p\text{-}value = 0.0002$), KNU ($p\text{-}value = 0.0010$), and KNE ($p\text{-}value = 0.0002$).

Since FIRE-DES++ using ENN is statistically better than FIRE-DES++ using RNG, we consider FIRE-DES++ using ENN for the rest of this paper.

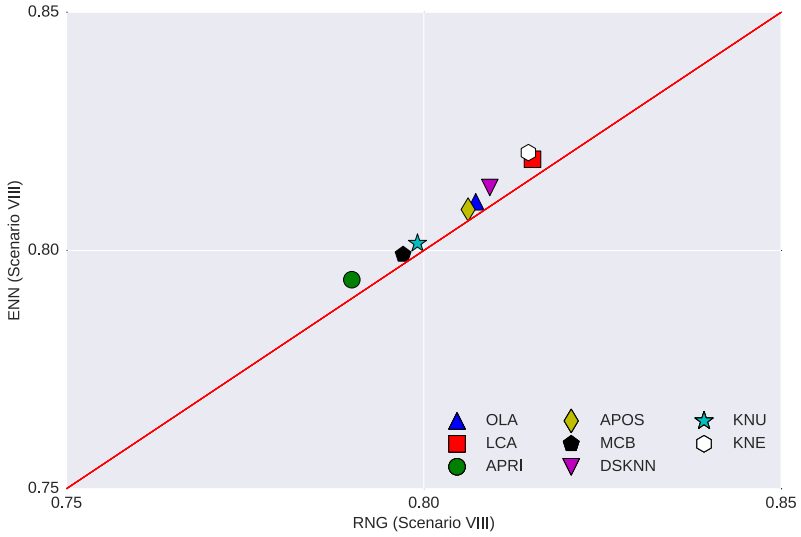


Figure 23 – Scatter plots of average AUC of FIRE-DES++ using the ENN (vertical axis) and the RNG (horizontal axis). Markers above the diagonal line indicates that the using the ENN had a better performance than using the RNG.

4.4.5 Comparison among different scenarios

Table 8 presents eight different scenarios for dynamic selection techniques considered in this experiments.

Table 8 – Eight test scenarios considered this work.

Scenario	KNNE	ENN	DFP
I	No	No	No
II	Yes	No	No
III	No	Yes	No
IV	No	No	Yes
V	Yes	Yes	No
VI	Yes	No	Yes
VII	No	Yes	Yes
VIII	Yes	Yes	Yes

For each scenario, we evaluated the classification performance of each DES technique over the 40 datasets, a total of 320 experiments (40 datasets $\times$ 8 techniques) per scenario. For each dataset and dynamic selection technique, we ranked each scenario from rank 1 to rank 8 (rank 1 is best), and calculated the average ranking. The result of the friedman test was $p\text{-}value = 2.39 \times e^{-70}$, indicating that there is statistical difference between the scenarios. Figure 24 presents the Nemenyi test (NEMENYI, 1962), where the lower the

average ranking the better, and techniques significantly different have a difference in ranking higher than the critical difference ($CD = 0.5383$).

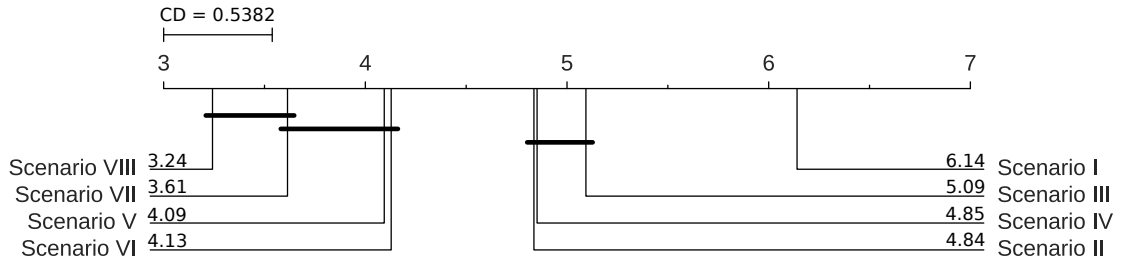


Figure 24 – CD diagram of Nemenyi post-hoc test considering all dynamic selection approaches, where $CD = 0.5383$, and Friedman *p-value* $= 2.39 \times e^{-70}$.

Figure 24 shows that:

- FIRE-DES++ (Scenario VIII) achieved the highest classification performance ranking (3.24), statistically outperforming Scenarios I, II, III, IV, V, and VI.
- Using the three phases of FIRE-DES++ (Scenario VIII), have a positive impact in the performance of DES techniques.
- Using any of the phases of FIRE-DES++ individually (Scenarios II, III, and IV) is enough to statistically outperform DES (Scenario I) in classification performance ranking.
- Using ENN and DFP (Scenario VII) had statistically equivalent ranking to using ENN, KNNE, and DFP (Scenario VIII), however, Scenario VIII had a better ranking.

This shows that all steps of FIRE-DES++ are important, and using the three of them combined leads to the highest overall improvement in classification performance of DES techniques.

Figure 25 shows the classification performance gain caused appending phases to regular DES, that is, the difference in classification performance (AUC) between Scenarios IV, VI, VII, and VIII and Scenario I. This figure shows that the three phases combined (DFP, KNNE, and ENN) causes the highest classification performance gain (0.0412), followed by DFP and ENN combined (0.0385), DFP and KNNE combined (0.0354), and finally DFP alone (0.0300). These results indicate that the filtering and the region of competence definition phases in the FIRE-DES++ framework cause performance gain over FIRE-DES.

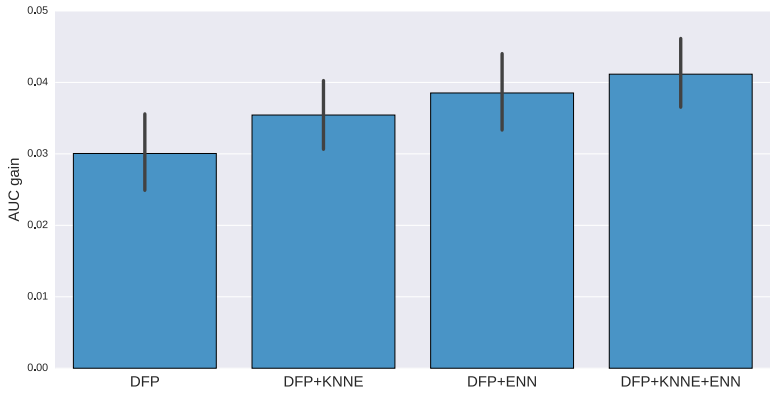


Figure 25 – Influence of each phase when compared to Scenario I, that is, the difference between Scenarios IV, VI, VII and VIII and Scenario I. The bars represent average classification performance gain (AUC) over the 40 classification datasets when adding DFP (0.0300), DFP+KNNE (0.0354), DFP+ENN (0.0385), and finally, DFP+KNNE+ENN (0.0412).

4.4.6 Comparison with FIRE-DES

In this section, we compare FIRE-DES++ and FIRE-DES for each DES technique considered in this work. The goal of this analysis is to investigate whether FIRE-DES++ significantly improves the performance of FIRE-DES.

Figure 26 presents the Nemenyi test comparing FIRE-DES++ (FOLA++, FLCA++, FAPRI++, FAPOS++, FMCB++, FDSKNN++, FKNU++, and FKNE++) with FIRE-DES (FOLA, FLCA, FAPRI, FAPOS, FMCB, FDSKNN, and FKNE). The result of the Friedman test was $p\text{-}value = 1.14 \times e^{-31}$, and the critical difference was 3.4021.

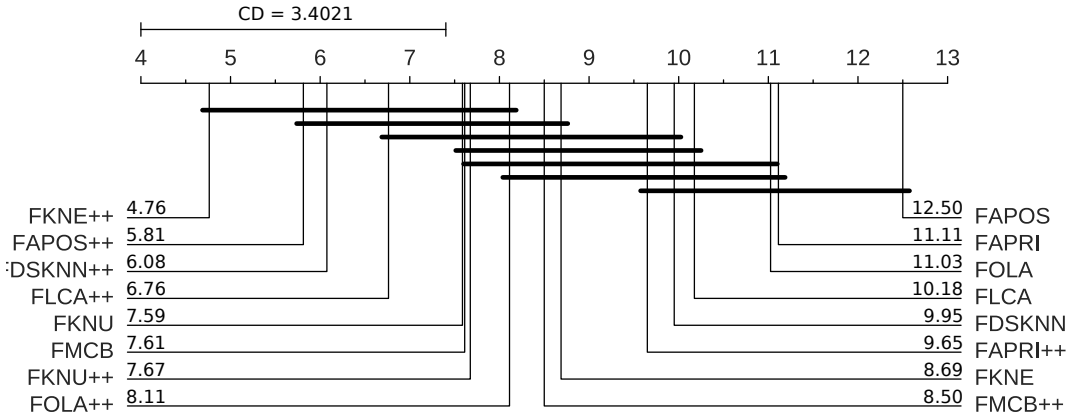


Figure 26 – CD diagram of Nemenyi post-hoc test considering all dynamic selection approaches, where $CD = 3.4021$, and Friedman $p\text{-}value = 1.14 \times e^{-31}$.

In Figure 26, FIRE-DES++ outperformed FIRE-DES in 6 DES techniques (KNE, APOS, DSKNN, LCA, OLA, APRI, APOS) and, with statistically significant differ-

ence, FKNE++ outperformed KNE, FAPOS++ outperformed FAPOS, FDSKNN++ outperformed FDSKNN, FLCA++ outperformed FLCA. FIRE-DES++ was outperformed by FIRE-DES only for 2 DES techniques (KNU and MCB), and even still, they were statistically equivalent.

Figure 27 presents a pairwise comparison of FIRE-DES++ and FIRE-DES for each DES technique. This comparison used the sign test calculated on the computed wins, ties and losses of FIRE-DES++. The null hypothesis H_0 was that using the FIRE-DES++ did not make any difference compared to FIRE-DES, and a rejection in H_0 meant that FIRE-DES++ significantly outperformed FIRE-DES. In this evaluation, we considered three levels of significance $\alpha = \{0.10, 0.05, 0.01\}$. To reject H_0, the number of wins plus half of the number of ties needs to be greater or equal to n_c (Equation 4.2):

$$n_c = \frac{n_{exp}}{2} + z_\alpha \times \frac{\sqrt[2]{n_{exp}}}{2} \tag{4.2}$$

where $n_{exp} = 40$ (the number of experiments), $n_c = \{24.05, 25.20, 27.37\}$, respectively for $\alpha = \{0.10, 0.05, 0.01\}$.

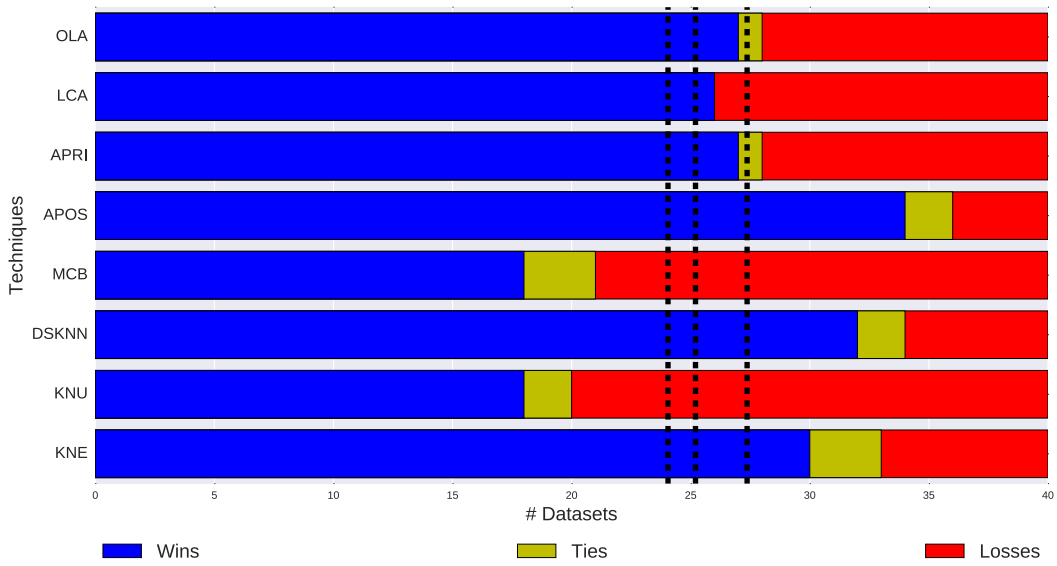


Figure 27 – Performance of FIRE-DES++ compared with FIRE-DES using different DES techniques in terms of wins, ties and losses considering the average AUC in the 40 datasets. The dashed lines (left to right) illustrates the critical values $n_c = \{24.05, 25.20, 27.37\}$ considering significance levels of $\alpha = \{0.10, 0.05, 0.01\}$, respectively.

Figure 27 shows that, for $\alpha = \{0.10, 0.05\}$ (first 2 lines left to right), with OLA, LCA, APRI, APOS, DSKNN, and KNE, FIRE-DES++ caused a significant performance gain over FIRE-DES. While for KNU and MCB, FIRE-DES++ and FIRE-DES had a

statistically equivalent classification performance, FIRE-DES++ being slightly worse than FIRE-DES. For $\alpha = 0.01$, only with OLA, APRI, APOS, DSKNN, and KNE, FIRE-DES++ caused a significant performance gain over FIRE-DES.

The only DES techniques in which the FIRE-DES++ did not outperform FIRE-DES were MCB and KNU. The reason for such behavior is because both MCB and KNU can select multiple classifiers (MCB selects all classifiers if the difference between the competence of the best classifier and all other classifiers is less than a predefined threshold, and KNU selects all classifiers that correctly classify any of the samples in the region of competence to submit a vote for the classification of the test sample - the more samples a classifier correctly classifies, the more votes the classifier has - (BRITTO; SABOURIN; OLIVEIRA, 2014)). Defining the region of competence of size K using the KNNE has this property of expanding the radius of the local region due to the fact that the selected samples are not necessarily the K closest patterns. With the region of competence expanded, it is more difficult to select a single classifier that is much better than all others (either to be selected or to outvote all others), therefore, this might lead to the selection of incompetent classifiers and the misclassification of test samples.

4.4.7 Comparison with state-of-the-art

Table 9 presents the average AUC of the FIRE-DES++ using KNE (FKNE++), FIRE-DES with KNE (FKNE), KNE, and the state-of-the-art DES techniques: RRC, META-DES, and META-DES.O. For each dataset, the best results are highlighted in bold, and significantly better results are marked with • (*p-value* ≤ 0.10) and •• (*p-value* ≤ 0.05), where *p-value* is obtained with the T-Student statistical test.

Table 9 shows that the FKNE++ achieved the highest AUC (0.8205), followed by META-DES (.8100), META-DES.Oracle (.8067), FKNE (0.8055), KNE (.8003), and RRC (.7934). Table 9 also shows that, considering $\alpha = 0.1$, the proposed framework outperformed the state-of-the-art dynamic selection techniques with statistical confidence.

Figure 28 presents the Nemenyi test comparing FKNE++, FKNE, KNE, META-DES, META-DES.Oracle and RRC, where the Friedman result was *p-value* $= 1.03 \times e^{-5}$, and the critical difference was $CD = 1.0828$. In this experiment, the best ranking was achieved by FKNE++ (2.45), being statistically better than FKNE, KNE, and RRC, and being statistically equivalent to META-DES and META-DES.Oracle.

Using the Wilcoxon Signed Rank Test (WILCOXON, 1945) ($\alpha = 0.10$) to compare the three best techniques (FKNE++, META-DES, and META-DES.Oracle), we can confirm with confidence that FKNE++ statistically outperformed META-DES (*p-value* $=$ 0.0595), and FKNE++ statistically outperformed META-DES.Oracle (*p-value* $= 0.0240$).

Table 9 and Figure 28 show that FIRE-DES++ outperformed state-of-the-art DES

Table 9 – Mean results of the AUC obtained for FKNE++, KNE, RSS, META-DES, and META-DES.Oracle. A pool of 100 perceptrons as base classifiers is used for all techniques. Best results are in bold, and significantly better results are marked with • (*p-value* $\leq .10$) and •• (*p-value* $\leq .05$), where *p-value* is obtained with the T-Student. The last line presents the *p-value* resulted from the Wilcoxon Signed Rank Test comparing the proposed framework (FKNE++) with FKNE, KNE, RRC, META-DES, and META-DES.Oracle.

Label	FKNE++	FKNE	KNE	RRC	META	META-O
1	0.6473(0.1067)	0.6439(0.0939)	0.6701(0.0775)	0.5693(0.0722)	0.7254(0.0589)	**••0.7263 (0.0605)**
2	0.9603(0.0288)	0.9587(0.0327)	0.9656(0.0214)	0.9535(0.0310)	0.9671(0.0288)	**0.9680 (0.0270)**
3	**•0.9710 (0.0114)**	0.9527(0.0228)	0.9601(0.0162)	0.9614(0.0116)	0.9636(0.0187)	0.9671(0.0146)
4	0.6999(0.0290)	0.6810(0.0361)	0.6802(0.0362)	**0.7179 (0.0287)**	0.7127(0.0282)	0.7095(0.0340)
5	**1.0000 (0.0000)**	**1.0000 (0.0000)**	**1.0000 (0.0000)**	0.9975(0.0075)	**1.0000 (0.0000)**	**1.0000 (0.0000)**
6	0.7425(0.0900)	0.6904(0.0805)	0.7154(0.0546)	0.7196(0.0613)	**••0.7853 (0.0766)**	0.7753(0.0716)
7	**0.6743 (0.0322)**	0.6513(0.0454)	0.6481(0.0316)	0.6161(0.0458)	0.6487(0.0216)	0.6333(0.0213)
8	0.9356(0.0184)	0.9348(0.0230)	0.9442(0.0192)	0.9223(0.0217)	**0.9559 (0.0142)**	0.9504(0.0159)
9	**••0.7173 (0.0321)**	0.7060(0.0379)	0.6937(0.0311)	0.6949(0.0285)	0.6930(0.0300)	0.6685(0.0325)
10	0.6994(0.0353)	0.6902(0.0451)	0.6797(0.0412)	**0.7027 (0.0276)**	0.6773(0.0464)	0.6733(0.0484)
11	**0.9067 (0.0538)**	0.8920(0.0631)	0.8918(0.0634)	0.8936(0.0615)	0.8871(0.0672)	0.9041(0.0631)
12	0.9575(0.0140)	0.9440(0.0163)	0.9447(0.0166)	0.9476(0.0299)	**••0.9590 (0.0175)**	0.9512(0.0226)
13	**0.8671 (0.0514)**	0.8220(0.0495)	0.8367(0.0579)	0.8351(0.0492)	0.8408(0.0600)	0.8460(0.0579)
14	**0.9894 (0.0211)**	0.9858(0.0247)	0.9858(0.0247)	0.9802(0.0287)	0.9809(0.0272)	0.9787(0.0295)
15	0.9751(0.0310)	0.9687(0.0401)	0.9687(0.0401)	**0.9858 (0.0285)**	0.9758(0.0448)	0.9700(0.0519)
16	0.8875(0.0569)	0.8327(0.0859)	0.8726(0.0559)	0.8304(0.0730)	**0.8898 (0.0455)**	0.8828(0.0525)
17	0.9901(0.0065)	0.9906(0.0071)	0.9908(0.0073)	0.9875(0.0096)	**0.9912 (0.0080)**	0.9911(0.0080)
18	0.8768(0.0673)	0.8653(0.0629)	0.8667(0.0612)	0.9156(0.0584)	0.9180(0.0577)	**••0.9214 (0.0560)**
19	**0.8663 (0.0395)**	0.8287(0.0476)	0.8212(0.0347)	0.8308(0.0601)	0.8406(0.0350)	0.8376(0.0388)
20	**0.8232 (0.0947)**	0.7665(0.1180)	0.7536(0.1018)	0.7623(0.1062)	0.7475(0.0998)	0.7630(0.0863)
21	0.8364(0.0571)	0.8168(0.0698)	0.8126(0.0711)	0.8329(0.0515)	**0.8427 (0.0608)**	0.8339(0.0685)
22	**•0.7264 (0.0693)**	0.6975(0.0759)	0.6860(0.0739)	0.6726(0.0624)	0.7218(0.0782)	0.6923(0.0745)
23	0.9437(0.0516)	0.9269(0.0477)	0.9272(0.0478)	0.9144(0.0415)	**••0.9749 (0.0220)**	0.9732(0.0266)
24	0.5473(0.0873)	**0.5477 (0.1317)**	0.5379(0.1074)	0.4953(0.0413)	0.5346(0.0738)	0.5465(0.0779)
25	0.5843(0.1209)	**•0.6140 (0.1574)**	0.5592(0.1087)	0.5560(0.1135)	0.5579(0.1359)	0.5469(0.1234)
26	0.9960(0.0080)	0.9960(0.0080)	0.9960(0.0080)	0.9936(0.0095)	0.9947(0.0088)	**0.9969 (0.0071)**
27	**•0.6493 (0.0955)**	0.6214(0.0812)	0.6058(0.0675)	0.5770(0.0812)	0.5587(0.0663)	0.5423(0.0482)
28	0.7643(0.1636)	**0.7671 (0.1697)**	0.7523(0.1636)	0.7650(0.1917)	0.7246(0.1701)	0.7608(0.1946)
29	0.9089(0.0534)	0.9003(0.0636)	0.9007(0.0633)	0.9198(0.0569)	**0.9203 (0.0565)**	0.9144(0.0669)
30	0.9436(0.0502)	0.9421(0.0521)	0.9469(0.0540)	0.8301(0.0866)	**0.9744 (0.0393)**	0.9524(0.0517)
31	**•0.9175 (0.1331)**	0.8418(0.1986)	0.8239(0.2136)	0.8421(0.1677)	0.8450(0.1679)	0.8186(0.1816)
32	0.9480(0.0992)	0.9480(0.0992)	0.9480(0.0992)	0.9710(0.0741)	0.9710(0.0741)	**0.9835 (0.0542)**
33	**0.5759 (0.0757)**	0.5670(0.0743)	0.5560(0.0662)	0.5028(0.0187)	0.5078(0.0365)	0.5040(0.0182)
34	0.7530(0.2241)	0.7716(0.2338)	0.7195(0.2349)	0.8503(0.1871)	**0.8534 (0.2028)**	0.8296(0.2010)
35	0.7772(0.0947)	0.7789(0.1105)	0.7764(0.0943)	0.7731(0.0923)	0.7674(0.0892)	**0.7796 (0.0946)**
36	**••0.7030 (0.0699)**	0.6961(0.0919)	0.6478(0.0800)	0.5968(0.0677)	0.6126(0.0434)	0.6080(0.0498)
37	**•0.5859 (0.0676)**	0.5738(0.0940)	0.5477(0.0667)	0.5577(0.0655)	0.5348(0.0462)	0.5446(0.0756)
38	**0.8633 (0.0765)**	0.8107(0.0922)	0.7871(0.0976)	0.7406(0.0852)	0.7920(0.0709)	0.7869(0.0731)
39	0.8422(0.1970)	0.8431(0.1975)	0.8454(0.1990)	0.8445(0.1989)	**0.8472 (0.2003)**	0.8463(0.1996)
40	**••0.7684 (0.0905)**	0.7557(0.0894)	0.7449(0.1009)	0.6767(0.0970)	0.7040(0.1122)	0.6906(0.1153)
Mean	0.8205(0.0676)	0.8055(0.0768)	0.8003(0.0703)	0.7934(0.0658)	0.8100(0.0635)	0.8067(0.0649)
p-value	NA	$3.80 \times e^{-5}$	$1.82 \times e^{-6}$	0.0002	0.0595	0.0240

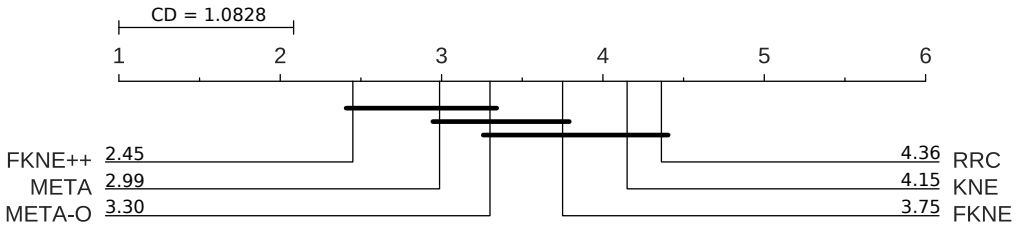


Figure 28 – CD diagram of Nemenyi post-hoc test considering all dynamic selection approaches, where $CD = 1.0828$, and Friedman $p\text{-}value = 1.03 \times e^{-5}$.

techniques META-DES, META-DES.Oracle, and RRC in classification performance and ranking.

4.5 Conclusion

In this paper, we presented 2 drawbacks of the Frienemy Indecision REgion Dynamic Ensemble Selection (FIRE-DES) framework: (1) noise sensitivity drawback: the classification performance of FIRE-DES is strongly affected by noise and outliers, as it mistakes noisy regions for indecision regions and applies the pre-selection of classifiers. (2) indecision region restriction drawback: FIRE-DES uses the region of competence to decide if a test sample is located in an indecision region, and only pre-selects classifiers when the region of competence of the test sample is composed of samples from different classes, restricting number of test samples in which the pre-selection is applied for its classification.

To tackle these drawbacks of FIRE-DES, we use the Edited Nearest Neighbors (ENN) (WILSON, 1972) to remove noise from the validation set (tackling the noise sensitivity drawback), and we use the K-Nearest Neighbors Equality (KNNE) (SIERRA et al., 2011) to define the region of competence selecting the nearest neighbors from each class (tackling the indecision region restriction drawback). We named this new framework FIRE-DES++.

FIRE-DES++ is composed of 4 phases: (1) Overproduction, where the pool of classifiers C is generated using the training set $\mathcal{T}$. (2) Filtering phase, where the framework removes noise and outliers from the validation set $\mathcal{D}_{SEL}$ using the ENN. (3) Region of competence definition phase, where FIRE-DES++ defines the region of competence by selecting an equal number of samples from each class from the validation set using the KNNE, avoiding the definition of a region of competence with samples of a single class. (4) Selection phase, where the framework pre-selects base classifiers with decision boundaries crossing the region of competence (if such classifiers exist) using the Dynamic Frienemy Pruning (DFP) (OLIVEIRA; CAVALCANTI; SABOURIN, 2017), avoiding the selection of classifiers that classify all samples in the region of competence as being from the same class. After the pre-selection, any DES technique is applied to perform the final selection.

In order to evaluate FIRE-DES++, we compared the results FIRE-DES++ with DES and FIRE-DES with 8 dynamic selection techniques over 40 datasets. The experimental results shows that the use of each individual phase (2, 3, and 4) causes significant improvement in all 8 DES techniques, and significant improvement in 6 out of 8 FIRE-DES techniques.

We also compared the proposed framework using the K-Nearest Oracles-Eliminate (KNE) (KO; SABOURIN; JR, 2008) with state-of-the-art DES framework, named, Randomized Reference Classifier (RRC) (WOLOSZYNSKI; KURZYNSKI, 2011), META-DES (CRUZ et al., 2015), and META-DES.Oracle (CRUZ; SABOURIN; CAVALCANTI, 2017b). The results showed that the proposed framework significantly outperformed these three state-of-the-art techniques with statistical confidence.

Future works on this topic will involve evolving the framework for handling multi-class problems, weighting the relevance of the samples in the region of competence selected by the KNNE (following the original weighting approach of the technique), and finally, developing an ensemble generation mechanism that generates diverse classifiers that maximizes the number of classifiers crossing different local regions in the feature-space.

5 GENERAL CONCLUSION

This thesis brought important contributions to Dynamic Ensemble Selection (DES). First we identified different types of regions of competence that state-of-art DES techniques from the literature do not take into account when selecting classifiers for the classification of test samples. Then, we proposed two DES frameworks: Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES) and Enhanced Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES++). FIRE-DES and FIRE-DES++ consider the different types of regions of competence when selecting classifiers. The proposed frameworks can be used with several DES techniques from the literature, improving their classification performance.

The three types of regions of competence that we identified are: (1) safe regions, composed of samples from the same class; (2) indecision regions, composed of samples from different classes; and (3) noisy regions, composed of samples from one class and one or more noisy samples from another class. Through a set of illustrative examples, we demonstrated that, by not considering these different types of regions of competence, DES techniques from the literature often select locally incompetent classifiers.

In Chapter 3, we proposed the FIRE-DES framework, a DES framework that focuses on optimizing DES for the selection of classifiers for the classification of test samples located in indecision regions. When the test sample is located in an indecision region, DES techniques can select incompetent classifiers that classify all samples as being from the same class. FIRE-DES avoids the selection of such incompetent classifiers by pre-selecting classifiers with decision boundaries crossing the region of competence of test samples located in indecision regions (if such classifiers exist). FIRE-DES obtained interesting results, improving the performance of all DES techniques from the literature considered in this study, achieving statistically equivalent classification performance to complex state-of-art DES techniques.

In Chapter 4, we proposed the FIRE-DES++ framework. FIRE-DES++ is an enhanced version of FIRE-DES that tackles two drawbacks of FIRE-DES (noise sensitivity and indecision region restriction), as it also focus in noisy regions and safe regions. The noise sensitivity drawback happens when noisy regions are often mistaken as indecision regions by FIRE-DES. FIRE-DES++ tackles the noise sensitivity drawback by applying a prototype selection technique to the validation set in order to remove noisy samples and outliers. The indecision region restriction drawback happens when the region of competence of a test sample is composed of samples from a single class, but the test sample is actually located near the border of classes. FIRE-DES++ tackles the indecision region restriction

drawback by defining the region of competence using an equal number of neighbors from each class, so DFP is also applied to samples otherwise ignored by FIRE-DES. The results of the experiments demonstrated that FIRE-DES++ outperforms DES and FIRE-DES techniques with statistical confidence, and also outperforms state-of-art DES techniques from the literature. Also, the experiments were performed in datasets of different levels of class imbalance, and the results demonstrate that FIRE-DES and FIRE-DES++ increased the robustness of DES to class imbalance.

Overall, this thesis presented a framework that introduces three modifications to DES systems that can be combined (FIRE-DES++) as proposed in this thesis, or used individually (i.e. FIRE-DES) as also demonstrated in this work. FIRE-DES++ can be used with several competence evaluation criteria (several DES techniques), making the framework very relevant, since it can be used with DES techniques that will be proposed in the future.

5.1 Future Works

The findings of this thesis suggests the following points for future works:

- An ensemble generation technique to be used with FIRE-DES and FIRE-DES++. The analysis conducted in Subsection 3.4.5 showed that, on average, 8% of the test samples classified as being located in an indecision region by FIRE-DES had no classifiers crossing their regions of competence. This indicates that an ensemble generation technique that maximizes the number of diverse classifiers crossing a local region of a test sample (for any test sample located in an indecision region) is a promissing approach.

- An instance hardness based DES technique. FIRE-DES++ tackles the noise issue by applying a prototype selection to the validation set from which the region of competence is selected. However, this binary approach (samples in the validation set are either removed or maintained) can be problematic because prototype selection techniques can remove important samples (samples that are not actually noise) and maintain noisy samples (samples with attribute noise that are harder to identify). This indicates that a DES technique that uses instance hardness to weight samples in the region of competence can achieve interesting results in datasets with different levels of noise.

- Using meta-learning to define when the Dynamic Frienemy Pruning (DFP) should be applied. FIRE-DES applies the DFP whenever the test sample has samples from different classes in the region of competence, FIRE-DES++ always applies the DFP as it defines all regions of competence using samples from different classes. In (CRUZ

et al., 2015), the authors defined dynamic selection of classifiers as a meta-problem and achieved interesting results. We believe that turning the decision of whether applying the DFP or not into a meta-problem is a promissing approach.

- Evolving FIRE-DES and FIRE-DES++ to: handle multi-class problems, be combined with META-DES; take the distance from the test sample to the samples in the region of competence into account when deciding if the test sample is located in an indecision region; and dismiss the need of a separate validation set for finding classifiers crossing the region of competence.

- Evaluating the impact of using different: techniques to generate the pool of classifiers; sizes of the pool of classifiers; base classifiers; sizes of the region of competence; and distance metrics when defining the region of competence.

REFERENCES

AKBANI, R.; KWEK, S.; JAPKOWICZ, N. Applying support vector machines to imbalanced datasets. In: *Machine Learning*. [S.l.]: Springer, 2004, (Lecture Notes in Computer Science, v. 3201). p. 39–50. Citado na página 131.

ALCALÁ, J. et al. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, Citeseer, v. 17, n. 2-3, p. 255–287, 2010. Citado 8 vezes nas páginas 48, 61, 77, 89, 123, 129, 138, and 157.

BATISTA, G. E.; CARVALHO, A. C.; MONARD, M. C. Applying one-sided selection to unbalanced datasets. In: *Proceedings of the Mexican International Conference on Artificial Intelligence: Advances in Artificial Intelligence*. [S.l.]: Springer, 2000. p. 315–325. Citado na página 133.

BATISTA, L.; GRANGER, E.; SABOURIN, R. Improving performance of hmm-based off-line signature verification systems through a multi-hypothesis approach. *International Journal on Document Analysis and Recognition*, Springer, v. 13, n. 1, p. 33–47, 2010. Citado 3 vezes nas páginas 20, 45, and 115.

BATISTA, L.; GRANGER, E.; SABOURIN, R. Dynamic ensemble selection for off-line signature verification. In: SPRINGER. *International Workshop on Multiple Classifier Systems*. [S.l.], 2011. p. 157–166. Citado na página 31.

BELL, R. M.; KOREN, Y. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, ACM, v. 9, n. 2, p. 75–79, 2007. Citado 2 vezes nas páginas 20 and 115.

BENAVOLI, A.; CORANI, G.; MANGILI, F. Should we really use post-hoc tests based on mean-ranks. *Journal of Machine Learning Research*, v. 17, n. 5, p. 1–10, 2016. Citado na página 125.

BHATTACHARYYA, S. et al. Data mining for credit card fraud: A comparative study. *Decision Support Systems*, Elsevier, v. 50, n. 3, p. 602–613, 2011. Citado 3 vezes nas páginas 20, 45, and 115.

BŁASZCZYŃSKI, J.; STEFANOWSKI, J.; IDKOWIAK, Ł. Extending bagging for imbalanced data. In: SPRINGER. *Proceedings of the International Conference on Computer Recognition Systems*. [S.l.], 2013. p. 269–278. Citado 2 vezes nas páginas 134 and 135.

BRADLEY, A. P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, Elsevier, v. 30, n. 7, p. 1145–1159, 1997. Citado 3 vezes nas páginas 63, 91, and 125.

BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996. Citado 11 vezes nas páginas 20, 41, 55, 57, 63, 80, 82, 89, 125, 132, and 134.

BREIMAN, L. Random forests. *Machine Learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado na página 20.

BRITTO, A. S.; SABOURIN, R.; OLIVEIRA, L. E. Dynamic selection of classifiers—a comprehensive review. *Pattern Recognition*, Elsevier, v. 47, n. 11, p. 3665–3680, 2014. Citado 13 vezes nas páginas 10, 20, 21, 31, 32, 43, 45, 46, 52, 74, 85, 96, and 115.

BRUN, A. L. et al. Contribution of data complexity features on dynamic classifier selection. In: IEEE. *Neural Networks (IJCNN), 2016 International Joint Conference on*. [S.l.], 2016. p. 4396–4403. Citado 2 vezes nas páginas 32 and 33.

CANO, J. R.; HERRERA, F.; LOZANO, M. Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study. *Evolutionary Computation, IEEE Transactions on*, IEEE, v. 7, n. 6, p. 561–575, 2003. Citado na página 148.

CARUANA, R. et al. Ensemble selection from libraries of models. In: *Proceedings of the International Conference on Machine Learning*. [S.l.: s.n.], 2004. p. 18. Citado na página 132.

CAVALIN, P. R.; SABOURIN, R.; SUEN, C. Y. Logid: An adaptive framework combining local and global incremental learning for dynamic selection of ensembles of hmms. *Pattern Recognition*, Elsevier, v. 45, n. 9, p. 3544–3556, 2012. Citado 2 vezes nas páginas 21 and 31.

CAVALIN, P. R.; SABOURIN, R.; SUEN, C. Y. Dynamic selection approaches for multiple classifier systems. *Neural Computing and Applications*, Springer, v. 22, n. 3-4, p. 673–688, 2013. Citado 4 vezes nas páginas 21, 31, 61, and 89.

CHAWLA, N. V. et al. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, v. 16, n. 1, p. 321–357, 2002. Citado na página 133.

CHAWLA, N. V. et al. Smoteboost: Improving prediction of the minority class in boosting. In: *Knowledge Discovery in Databases*. [S.l.]: Springer, 2003. p. 107–119. Citado na página 132.

CIESLAK, D. A.; CHAWLA, N. V.; STRIEGEL, A. Combating imbalance in network intrusion datasets. In: IEEE. *Granular Computing, International Conference on*. [S.l.], 2006. p. 732–737. Citado na página 131.

CORNE, D. W.; KNOWLES, J. D. No free lunch and free leftovers theorems for multiobjective optimisation problems. In: SPRINGER. *International Conference on Evolutionary Multi-Criterion Optimization*. [S.l.], 2003. p. 327–341. Citado na página 22.

COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, IEEE, v. 13, n. 1, p. 21–27, 1967. Citado 2 vezes nas páginas 85 and 146.

CRUZ, R. M.; CAVALCANTI, G. D.; REN, T. I. A method for dynamic ensemble selection based on a filter and an adaptive distance to improve the quality of the regions of competence. In: IEEE. *International Joint Conference on Neural Networks*. [S.l.], 2011. p. 1126–1133. Citado 4 vezes nas páginas 21, 29, 63, and 76.

CRUZ, R. M. et al. Feature representation selection based on classifier projection space and oracle analysis. *Expert Systems with Applications*, Elsevier, v. 40, n. 9, p. 3813–3827, 2013. Citado 3 vezes nas páginas 20, 45, and 115.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. A DEEP analysis of the META-DES framework for dynamic selection of ensemble of classifiers. *arXiv preprint arXiv:1509.00825*, 2015. Citado na página 41.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. META-DES.H: a dynamic ensemble selection technique using meta-learning and a dynamic weighting approach. In: IEEE. *International Joint Conference on Neural Networks*. [S.l.], 2015. p. 1–8. Citado 3 vezes nas páginas 33, 41, and 45.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. Prototype selection for dynamic classifier and ensemble selection. *Neural Computing and Applications*, Springer, p. 1–11, 2016. Citado 5 vezes nas páginas 29, 74, 76, 82, and 89.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. Analyzing different prototype selection techniques for dynamic classifier and ensemble selection. In: IEEE. *International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2017. p. 3959–3966. Citado 4 vezes nas páginas 21, 29, 83, and 91.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. META-DES.Oracle: Meta-learning and feature selection for dynamic ensemble selection. *Information Fusion*, Elsevier, v. 38, p. 84–103, 2017. Citado 12 vezes nas páginas 21, 33, 34, 42, 48, 61, 69, 77, 88, 89, 99, and 125.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, Elsevier, v. 41, p. 195–216, 2018. Citado 11 vezes nas páginas 10, 21, 27, 28, 31, 33, 42, 43, 74, 88, and 115.

CRUZ, R. M. et al. META-DES: A dynamic ensemble selection framework using meta-learning. *Pattern Recognition*, Elsevier, v. 48, n. 5, p. 1925–1935, 2015. Citado 20 vezes nas páginas 21, 22, 27, 31, 32, 33, 34, 41, 42, 46, 48, 61, 63, 69, 77, 88, 89, 99, 102, and 125.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, v. 7, n. 1, p. 1–30, 2006. Citado 6 vezes nas páginas 49, 63, 77, 91, 125, and 127.

DERRAC, J.; GARCÍA, S.; HERRERA, F. Stratified prototype selection based on a steady-state memetic algorithm: a study of scalability. *Memetic Computing*, Springer, v. 2, n. 3, p. 183–199, 2010. Citado na página 148.

DERRAC, J. et al. Evolutionary-based selection of generalized instances for imbalanced classification. *Knowledge-Based Systems*, Elsevier, v. 25, n. 1, p. 3–12, 2012. Citado na página 149.

DIDACI, L.; GIACINTO, G. Dynamic classifier selection by adaptive k-nearest-neighbourhood rule. In: SPRINGER. *International Workshop on Multiple Classifier Systems*. [S.l.], 2004. p. 174–183. Citado 2 vezes nas páginas 29 and 30.

DIETTERICH, T. G. Ensemble methods in machine learning. In: SPRINGER. *International workshop on multiple classifier systems*. [S.l.], 2000. p. 1–15. Citado 2 vezes nas páginas 45 and 131.

DŽEROSKI, S.; ŽENKO, B. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, Springer, v. 54, n. 3, p. 255–273, 2004. Citado na página 21.

ELKAN, C. The foundations of cost-sensitive learning. In: *International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2001. v. 17, n. 1, p. 973–978. Citado 3 vezes nas páginas 131, 133, and 151.

FAYED, H. A.; HASHEM, S. R.; ATIYA, A. F. Self-generating prototypes for pattern classification. *Pattern Recognition*, Elsevier, v. 40, n. 5, p. 1498–1509, 2007. Citado 3 vezes nas páginas 26, 147, and 149.

FERNÁNDEZ-DELGADO, M. et al. Do we need hundreds of classifiers to solve real world classification problems. *J. Mach. Learn. Res*, v. 15, n. 1, p. 3133–3181, 2014. Citado na página 41.

FERNÁNDEZ, F.; ISASI, P. Evolutionary design of nearest prototype classifiers. *Journal of Heuristics*, Springer, v. 10, n. 4, p. 431–454, 2004. Citado na página 148.

FERNÁNDEZ, F.; ISASI, P. Local feature weighting in nearest prototype classification. *Neural Networks, IEEE Transactions on*, IEEE, v. 19, n. 1, p. 40–53, 2008. Citado na página 146.

FREUND, Y. Boosting a weak learning algorithm by majority. *Information and Computation*, Elsevier, v. 121, n. 2, p. 256–285, 1995. Citado na página 134.

FREUND, Y.; SCHAPIRE, R. E. A desicion-theoretic generalization of on-line learning and an application to boosting. In: SPRINGER. *European Conference on Computational Learning Theory*. [S.l.], 1995. p. 23–37. Citado na página 20.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937. Citado 2 vezes nas páginas 49 and 77.

FRIEDMAN, M. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, JSTOR, v. 11, n. 1, p. 86–92, 1940. Citado 2 vezes nas páginas 63 and 91.

GALAR, M. et al. 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)*, IEEE, v. 42, n. 4, p. 463–484, 2012. Citado 10 vezes nas páginas 20, 23, 61, 89, 131, 132, 133, 134, 141, and 144.

GALAR, M. et al. Ordering-based pruning for improving the performance of ensembles of classifiers in the framework of imbalanced datasets. *Information Sciences*, Elsevier, v. 354, p. 178–196, 2016. Citado 2 vezes nas páginas 61 and 89.

GARAIN, U. Prototype reduction using an artificial immune model. *Pattern analysis and applications*, Springer, v. 11, n. 3-4, p. 353–363, 2008. Citado na página 149.

GARCÍA, S.; CANO, J. R.; HERRERA, F. A memetic algorithm for evolutionary prototype selection: A scaling up approach. *Pattern Recognition*, Elsevier, v. 41, n. 8, p. 2693–2709, 2008. Citado 4 vezes nas páginas 147, 148, 154, and 156.

GARCIA, S. et al. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 34, n. 3, p. 417–435, 2012. Citado 6 vezes nas páginas 57, 76, 80, 82, 146, and 148.

GARCÍA, S.; LUENGO, J.; HERRERA, F. Dealing with noisy data. In: *Data Preprocessing in Data Mining*. [S.l.]: Springer, 2015. p. 107–145. Citado 8 vezes nas páginas 10, 11, 12, 22, 49, 50, 57, and 75.

GARCÍA, V. et al. Combined effects of class imbalance and class overlap on instance-based classification. In: SPRINGER. *International Conference on Intelligent Data Engineering and Automated Learning*. [S.l.], 2006. p. 371–378. Citado na página 49.

GIACINTO, G.; ROLI, F. Methods for dynamic classifier selection. In: IEEE. *International Conference on Image Analysis and Processing*. [S.l.], 1999. p. 659–664. Citado 9 vezes nas páginas 21, 22, 32, 34, 45, 52, 61, 89, and 125.

GIACINTO, G.; ROLI, F. Dynamic classifier selection. In: SPRINGER. *International Workshop on Multiple Classifier Systems*. [S.l.], 2000. p. 177–189. Citado 2 vezes nas páginas 21 and 45.

GIACINTO, G.; ROLI, F. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, Elsevier, v. 19, n. 9, p. 699–707, 2001. Citado na página 21.

GIACINTO, G.; ROLI, F. Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*, Pergamon, v. 34, n. 9, p. 1879–1881, 2001. Citado 9 vezes nas páginas 21, 31, 32, 34, 39, 45, 61, 89, and 125.

HANSEN, L. K.; SALAMON, P. Neural network ensembles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 12, n. 10, p. 993–1001, 1990. Citado na página 132.

HE, H.; GARCIA, E. A. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, IEEE, v. 21, n. 9, p. 1263–1284, 2009. Citado na página 131.

HO, T. K. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 20, n. 8, p. 832–844, 1998. Citado 2 vezes nas páginas 20 and 132.

HUANG, Y. S.; SUEN, C. Y. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 17, n. 1, p. 90–94, 1995. Citado na página 31.

HULSE, J. V.; KHOSHGOFTAAR, T. M.; NAPOLITANO, A. Experimental perspectives on learning from imbalanced data. In: *Proceedings of the International Conference on Machine Learning*. [S.l.: s.n.], 2007. p. 935–942. Citado na página 131.

IVAKHNENKO, A. Heuristic self-organization in problems of engineering cybernetics. *Automatica*, Elsevier, v. 6, n. 2, p. 207–219, 1970. Citado na página 34.

JACOBS, R. A. et al. Adaptive mixtures of local experts. *Neural computation*, MIT Press, v. 3, n. 1, p. 79–87, 1991. Citado na página 21.

JAHRER, M.; TÖSCHER, A.; LEGENSTEIN, R. Combining predictions for accurate recommender systems. In: ACM. *International Conference on Knowledge Discovery and Data Mining*. [S.l.], 2010. p. 693–702. Citado 3 vezes nas páginas 20, 45, and 115.

KITTLER, J. et al. On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 20, n. 3, p. 226–239, 1998. Citado na página 21.

KO, A. H.; SABOURIN, R.; JR, A. S. B. From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition*, Elsevier, v. 41, n. 5, p. 1718–1731, 2008. Citado 18 vezes nas páginas 13, 21, 22, 32, 34, 37, 38, 45, 52, 61, 63, 74, 89, 99, 115, 116, 117, and 125.

KONONENKO, I.; KUKAR, M. *Machine Learning and Data Mining: Introduction to Principles and Algorithms*. [S.l.]: Horwood Publishing Limited, 2007. ISBN 1904275214, 9781904275213. Citado na página 146.

KOREN, Y. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, v. 81, 2009. Citado 2 vezes nas páginas 20 and 115.

KUBAT, M.; MATWIN, S. Addressing the curse of imbalanced training sets: One-sided selection. In: *Proceedings of the International Conference on Machine Learning*. [S.l.]: Morgan Kaufmann, 1997. v. 97, p. 179–186. Citado na página 133.

KUNCHEVA, L. I. Clustering-and-selection model for classifier combination. In: IEEE. *Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 2000. Proceedings. Fourth International Conference on*. [S.l.], 2000. v. 1, p. 185–188. Citado na página 30.

KUNCHEVA, L. I. A theoretical study on six classifier fusion strategies. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE, v. 24, n. 2, p. 281–286, 2002. Citado na página 116.

KUNCHEVA, L. I. *Combining pattern classifiers: methods and algorithms*. [S.l.]: John Wiley & Sons, 2004. Citado 4 vezes nas páginas 45, 115, 131, and 137.

KUNCHEVA, L. I.; RODRIGUEZ, J. J. Classifier ensembles with a random linear oracle. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 19, n. 4, p. 500–508, 2007. Citado na página 32.

KUNCHEVA, L. I.; WHITAKER, C. J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, Springer, v. 51, n. 2, p. 181–207, 2003. Citado 3 vezes nas páginas 20, 131, and 137.

LAURIKKALA, J. Improving identification of difficult small classes by balancing class distribution. *Artificial Intelligence in Medicine*, Springer, p. 63–66, 2001. Citado na página 84.

LEMAITRE, G.; NOGUEIRA, F.; ARIDAS, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, v. 18, n. 17, p. 1–5, 2017. Citado na página 84.

LIMA, T. P. F. D.; LUDERMIR, T. B. Optimizing dynamic ensemble selection procedure by evolutionary extreme learning machines and a noise reduction filter. In: IEEE. *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*. [S.l.], 2013. p. 546–552. Citado na página 29.

LIMA, T. P. F. de; SERGIO, A. T.; LUDERMIR, T. B. Improving classifiers and regions of competence in dynamic ensemble selection. In: IEEE. *Brazilian Conference on Intelligent Systems (BRACIS)*. [S.l.], 2014. p. 13–18. Citado na página 29.

LIN, C. et al. Libd3c: ensemble classifiers with a clustering and dynamic selection strategy. *Neurocomputing*, Elsevier, v. 123, p. 424–435, 2014. Citado na página 30.

LIU, H.; MOTODA, H. On issues of instance selection. *Data Mining and Knowledge Discovery*, Springer, v. 6, n. 2, p. 115–130, 2002. Citado na página 148.

LIU, Y.-H.; CHEN, Y.-T. Total margin based adaptive fuzzy support vector machines for multiview face recognition. In: IEEE. *International Conference o Systems, Man and Cybernetics*. [S.l.], 2005. v. 2, p. 1704–1711. Citado na página 131.

LÓPEZ, V. et al. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, Elsevier, v. 250, p. 113–141, 2013. Citado 10 vezes nas páginas 63, 91, 116, 122, 125, 131, 132, 133, 134, and 147.

LU, Z. et al. Ensemble pruning via individual contribution ordering. In: ACM. *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.], 2010. p. 871–880. Citado na página 21.

MAZUROWSKI, M. A. et al. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, Elsevier, v. 21, n. 2, p. 427–436, 2008. Citado na página 131.

MENDIALDUA, I. et al. Dynamic selection of the best base classifier in one versus one. *Knowledge-Based Systems*, Elsevier, v. 85, p. 298–306, 2015. Citado 3 vezes nas páginas 29, 76, and 85.

NANNI, L.; FANTOZZI, C.; LAZZARINI, N. Coupling different methods for overcoming the class imbalance problem. *Neurocomputing*, Elsevier, v. 158, p. 48–61, 2015. Citado 2 vezes nas páginas 61 and 89.

NANNI, L.; LUMINI, A. Particle swarm optimization for prototype reduction. *Neurocomputing*, Elsevier, v. 72, n. 4, p. 1092–1097, 2009. Citado na página 149.

NAPIERAŁA, K.; STEFANOWSKI, J.; WILK, S. Learning from imbalanced data in presence of noisy and borderline examples. In: SPRINGER. *International Conference on Rough Sets and Current Trends in Computing*. [S.l.], 2010. p. 158–167. Citado na página 49.

NEMENYI, P. Distribution-free multiple comparisons. In: *Biometrics*. [S.l.: s.n.], 1962. v. 18, n. 2, p. 263. Citado 5 vezes nas páginas 49, 63, 77, 91, and 92.

OLIVEIRA, D. V. et al. Evolutionary adaptive self-generating prototypes for imbalanced datasets. In: IEEE. *International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2015. p. 1–8. Citado na página 24.

OLIVEIRA, D. V.; CAVALCANTI, G. D.; SABOURIN, R. Online pruning of base classifiers for dynamic ensemble selection. *Pattern Recognition*, Elsevier, v. 72, p. 44–58, 2017. Citado 23 vezes nas páginas 10, 12, 13, 14, 16, 22, 25, 34, 74, 75, 77, 78, 80, 82, 85, 86, 98, 115, 118, 121, 123, 124, and 125.

OLIVEIRA, D. V. et al. A bootstrap-based iterative selection for ensemble generation. In: IEEE. *International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2015. p. 1–7. Citado na página 24.

OLIVEIRA, D. V. R. et al. Improved self-generating prototypes algorithm for imbalanced datasets. In: IEEE. *International Conference on Tools with Artificial Intelligence (ICTAI)*. [S.l.], 2012. v. 1, p. 904–909. Citado 5 vezes nas páginas 26, 132, 147, 150, and 159.

PARTALAS, I.; TSOUMAKAS, G.; VLAHAVAS, I. P. Focused ensemble selection: A diversity-based method for greedy ensemble selection. In: *Proceeding of the 18th European Conference on Artificial Intelligence*. [S.l.: s.n.], 2008. p. 117–121. Citado na página 21.

PATRICK, E. A.; FISCHER, F. A generalization of the k-nearest neighbor rule. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the International Joint Conference on Artificial Intelligence*. [S.l.], 1969. p. 63–63. Citado 2 vezes nas páginas 85 and 146.

PEREIRA, C. de S.; CAVALCANTI, G. D. C. Prototype selection: Combining self-generating prototypes and gaussian mixtures for pattern classification. In: IEEE. *Proceedings on International Joint Conference on Neural Networks*. [S.l.], 2008. p. 3505–3510. Citado 2 vezes nas páginas 147 and 149.

PUURULA, A.; READ, J.; BIFET, A. Kaggle lshtc4 winning solution. *arXiv preprint arXiv:1405.0546*, 2014. Citado 2 vezes nas páginas 20 and 115.

RODRIGUEZ, J. J.; KUNCHEVA, L. I.; ALONSO, C. J. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 28, n. 10, p. 1619–1630, 2006. Citado na página 20.

ROY, A. et al. A study on combining dynamic selection and data preprocessing for imbalance learning. *Neurocomputing*, Elsevier, 2018. Citado na página 29.

SABOURIN, M. et al. Classifier combination for hand-printed digit recognition. In: IEEE. *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*. [S.l.], 1993. p. 163–166. Citado 2 vezes nas páginas 22 and 32.

SÁEZ, J. A. et al. Tackling the problem of classification with noisy data using multiple classifier systems: Analysis of the performance and robustness. *Information Sciences*, Elsevier, v. 247, p. 1–20, 2013. Citado na página 20.

SÁNCHEZ, J. S.; PLA, F.; FERRI, F. J. Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognition Letters*, Elsevier, v. 18, n. 6, p. 507–513, 1997. Citado 2 vezes nas páginas 83 and 91.

SANTANA, A. et al. A dynamic classifier selection method to build ensembles using accuracy and diversity. In: IEEE. *Brazilian Symposium on Neural Networks*. [S.l.], 2006. p. 36–41. Citado 9 vezes nas páginas 21, 22, 30, 33, 39, 45, 61, 89, and 125.

SANTOS, E. M. D.; SABOURIN, R.; MAUPIN, P. Single and multi-objective genetic algorithms for the selection of ensemble of classifiers. In: IEEE. *International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2006. p. 3070–3077. Citado na página 21.

SANTOS, E. M. D.; SABOURIN, R.; MAUPIN, P. Ambiguity-guided dynamic selection of ensemble of classifiers. In: IEEE. *Information Fusion, 2007 10th International Conference on*. [S.l.], 2007. p. 1–8. Citado na página 33.

SANTOS, E. M. D.; SABOURIN, R.; MAUPIN, P. A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. *Pattern Recognition*, Elsevier, v. 41, n. 10, p. 2993–3009, 2008. Citado 3 vezes nas páginas 21, 22, and 33.

SANTOS, E. M. D.; SABOURIN, R.; MAUPIN, P. Overfitting cautious selection of classifier ensembles with genetic algorithms. *Information Fusion*, Elsevier, v. 10, n. 2, p. 150–162, 2009. Citado na página 33.

SCHAPIRE, R. E. The strength of weak learnability. *Machine learning*, Springer, v. 5, n. 2, p. 197–227, 1990. Citado 2 vezes nas páginas 132 and 134.

SCHAPIRE, R. E. The boosting approach to machine learning: An overview. In: *Nonlinear Estimation and Classification*. [S.l.]: Springer, 2003. p. 149–171. Citado na página 134.

SHAKHNAROVICH, G.; DARRELL, T.; INDYK, P. *Nearest-neighbor methods in learning and vision: theory and practice*. [S.l.]: MIT press Cambridge, MA, USA:, 2005. Citado na página 85.

SHAKHNAROVICH, G.; DARRELL, T.; INDYK, P. *Nearest-neighbor methods in learning and vision: theory and practice*. [S.l.]: MIT press Cambridge, MA, USA:, 2005. Citado na página 146.

SIERRA, B. et al. K-nearest neighbor equality: Giving equal chance to all existing classes. *Information Sciences*, Elsevier, v. 181, n. 23, p. 5158–5168, 2011. Citado 5 vezes nas páginas 29, 76, 80, 85, and 98.

SINGH, S.; SINGH, M. A dynamic classifier selection and combination approach to image region labelling. *Signal Processing: Image Communication*, Elsevier, v. 20, n. 3, p. 219–231, 2005. Citado 3 vezes nas páginas 20, 45, and 115.

SKURICHINA, M.; DUIN, R. P. Bagging for linear classifiers. *Pattern Recognition*, Elsevier, v. 31, n. 7, p. 909–930, 1998. Citado 2 vezes nas páginas 57 and 82.

SMITH, M. R.; MARTINEZ, T.; GIRAUD-CARRIER, C. An instance level analysis of data complexity. *Machine Learning*, Springer, v. 95, n. 2, p. 225–256, 2014. Citado na página 63.

SOARES, R. G. et al. Using accuracy and diversity to select classifiers to build ensembles. In: IEEE. *International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2006. p. 1310–1316. Citado 2 vezes nas páginas 30 and 33.

SOUTO, M. C. de et al. Empirical comparison of dynamic classifier selection methods based on diversity and accuracy for building ensembles. In: IEEE. *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. [S.l.], 2008. p. 1480–1487. Citado na página 30.

STEFANOWSKI, J.; WILK, S. Selective pre-processing of imbalanced data for improving classification performance. In: *Data Warehousing and Knowledge Discovery*. [S.l.]: Springer, 2008, (Lecture Notes in Computer Science, v. 5182). p. 283–292. Citado na página 133.

TANG, E. K.; SUGANTHAN, P. N.; YAO, X. An analysis of diversity measures. *Machine Learning*, Springer, v. 65, n. 1, p. 247–271, 2006. Citado 3 vezes nas páginas 20, 39, and 131.

TORRE, M. De-la et al. An adaptive ensemble-based system for face recognition in person re-identification. *Machine Vision and Applications*, Springer, v. 26, n. 6, p. 741–773, 2015. Citado 3 vezes nas páginas 20, 45, and 115.

TRIGUERO, I. et al. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, IEEE, v. 42, n. 1, p. 86–100, 2012. Citado 3 vezes nas páginas 146, 148, and 149.

TRIGUERO, I.; GARCÍA, S.; HERRERA, F. Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. *Pattern Recognition*, Elsevier, v. 44, n. 4, p. 901–916, 2011. Citado na página 149.

VRIESMANN, L. M. et al. Combining overall and local class accuracies in an oracle-based method for dynamic ensemble selection. In: IEEE. *Neural Networks (IJCNN), 2015 International Joint Conference on*. [S.l.], 2015. p. 1–7. Citado na página 38.

WANG, J.; NESKOVIC, P.; COOPER, L. N. Improving nearest neighbor rule with a simple adaptive distance measure. *Pattern Recognition Letters*, Elsevier, v. 28, n. 2, p. 207–213, 2007. Citado na página 29.

WANG, S.; YAO, X. Diversity analysis on imbalanced data sets by using ensemble models. In: IEEE. *Computational Intelligence and Data Mining, Symposium on*. [S.l.], 2009. p. 324–331. Citado 2 vezes nas páginas 132 and 134.

WILCOXON, F. Individual comparisons by ranking methods. *Biometrics bulletin*, JSTOR, v. 1, n. 6, p. 80–83, 1945. Citado 9 vezes nas páginas 49, 63, 77, 91, 96, 125, 139, 147, and 158.

WILSON, D. L. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, IEEE, n. 3, p. 408–421, 1972. Citado 3 vezes nas páginas 83, 91, and 98.

WILSON, D. R.; MARTINEZ, T. R. Reduction techniques for instance-based learning algorithms. *Machine learning*, Springer, v. 38, n. 3, p. 257–286, 2000. Citado na página 146.

WOLOSZYNSKI, T.; KURZYNSKI, M. On a new measure of classifier competence applied to the design of multiclassifier systems. In: SPRINGER. *International Conference on Image Analysis and Processing*. [S.l.], 2009. p. 995–1004. Citado na página 31.

WOLOSZYNSKI, T.; KURZYNSKI, M. A measure of competence based on randomized reference classifier for dynamic ensemble selection. In: IEEE. *International Conference on Pattern Recognition (ICPR)*. [S.l.], 2010. p. 4194–4197. Citado na página 42.

WOLOSZYNSKI, T.; KURZYNSKI, M. A probabilistic model of classifier competence for dynamic ensemble selection. *Pattern Recognition*, Elsevier, v. 44, n. 10, p. 2656–2668, 2011. Citado 11 vezes nas páginas 21, 31, 34, 48, 61, 69, 77, 88, 89, 99, and 125.

WOLOSZYNSKI, T. et al. A measure of competence based on random classification for dynamic ensemble selection. *Information Fusion*, Elsevier, v. 13, n. 3, p. 207–213, 2012. Citado na página 31.

WOODS, K.; KEGELMEYER, W. P.; BOWYER, K. W. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 19, n. 4, p. 405–410, 1997. Citado 13 vezes nas páginas 21, 22, 32, 34, 35, 45, 50, 61, 76, 78, 85, 89, and 125.

WOŹNIAK, M.; GRAÑA, M.; CORCHADO, E. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, Elsevier, v. 16, p. 3–17, 2014. Citado 3 vezes nas páginas 45, 115, and 131.

XIAO, J.; HE, C. Adaptive selection of classifier ensemble based on gmdh. In: IEEE. *Future Information Technology and Management Engineering, 2008. FITME'08. International Seminar on*. [S.l.], 2008. p. 61–64. Citado na página 43.

XIAO, J.; HE, C. Dynamic classifier ensemble selection based on gmdh. In: IEEE. *International Joint Conference on Computational Sciences and Optimization*. [S.l.], 2009. v. 1, p. 731–734. Citado na página 21.

XIAO, J.; HE, C. Dynamic classifier ensemble selection based on gmdh. In: IEEE. *Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on*. [S.l.], 2009. v. 1, p. 731–734. Citado 2 vezes nas páginas 34 and 43.

XIAO, J. et al. A dynamic classifier ensemble selection approach for noise data. *Information Sciences*, Elsevier, v. 180, n. 18, p. 3402–3421, 2010. Citado na página 21.

XIAO, J. et al. A dynamic classifier ensemble selection approach for noise data. *Information Sciences*, Elsevier, v. 180, n. 18, p. 3402–3421, 2010. Citado na página 43.

XIAO, J. et al. Dynamic classifier ensemble model for customer classification with imbalanced class distribution. *Expert Systems with Applications*, Elsevier, v. 39, n. 3, p. 3668–3675, 2012. Citado 3 vezes nas páginas 21, 33, and 43.

YANG, Q.; WU, X. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, World Scientific, v. 5, n. 04, p. 597–604, 2006. Citado 3 vezes nas páginas 131, 132, and 147.

ZHU, X.; WU, X.; YANG, Y. Dynamic classifier selection for effective mining from noisy data streams. In: IEEE. *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*. [S.l.], 2004. p. 305–312. Citado 2 vezes nas páginas 21 and 27.

APPENDIX A – K-NEAREST ORACLES BORDERLINE DYNAMIC CLASSIFIER ENSEMBLE SELECTION

abstract

Dynamic Ensemble Selection (DES) techniques aim to select locally competent classifiers for the classification of each new test sample. Most DES techniques estimate the competence of classifiers using a given criterion over the region of competence of the test sample (its the nearest neighbors in the validation set). The K-Nearest Oracles Eliminate (KNORA-E) DES selects all classifiers that correctly classify all samples in the region of competence of the test sample, if such classifier exists, otherwise, it removes from the region of competence the sample that is furthest from the test sample, and the process repeats. When the region of competence has samples of different classes, KNORA-E can reduce the region of competence in such a way that only samples of a single class remain in the region of competence, leading to the selection of locally incompetent classifiers that classify all samples in the region of competence as being from the same class. In this paper, we propose two DES techniques: K-Nearest Oracles Borderline (KNORA-B) and K-Nearest Oracles Borderline Imbalanced (KNORA-BI). KNORA-B is a DES technique based on KNORA-E that reduces the region of competence but maintains at least one sample from each class that is in the original region of competence. KNORA-BI is a variation of KNORA-B for imbalance datasets that reduces the region of competence but maintains at least one minority class sample if there is any in the original region of competence.

Experiments are conducted comparing the proposed techniques with 19 DES techniques from the literature using 40 datasets. The results show that the proposed techniques achieved interesting results, with KNORA-BI outperforming state-of-art techniques.

A.1 Introduction

Multiple Classifier Systems (MCS) (WOŹNIAK; GRAÑA; CORCHADO, 2014) combine classifiers in the hope that several classifiers outperform any individual classifier in classification accuracy (KUNCHEVA, 2004). MCS have been considered an interesting alternative for increasing the classification accuracy in several studies (SINGH; SINGH, 2005) (CRUZ et al., 2013) (BATISTA; GRANGER; SABOURIN, 2010) (JAHRER; TÖSCHER; LEGENSTEIN, 2010) (BHATTACHARYYA et al., 2011) (TORRE et al., 2015) and machine learning competitions (PUURULA; READ; BIFET, 2014) (KOREN, 2009) (BELL; KOREN, 2007).

Dynamic Ensemble Selection (DES) (KO; SABOURIN; JR, 2008) (CRUZ; SABOURIN; CAVALCANTI, 2018) (BRITTO; SABOURIN; OLIVEIRA, 2014) techniques select one or more classifiers for the classification of each new test sample. Relying on the assumption that different classifiers are competent ("experts") in different local regions of the feature space, most DES techniques estimate the level of competence of a classifier for the classification of a test sample x_{query}, using some criteria over the region of competence of x_{query}. The region of competence of x_{query} is the set of K nearest neighbors of x_{query} in the validation set $\mathcal{D}_{SEL}$.

In (KO; SABOURIN; JR, 2008), Ko et al. proposed two DES techniques: K-Nearest Oracles Eliminate (KNORA-E) and K-Nearest Oracles Union (KNORA-U). KNORA-E selects all classifiers that correctly classify all samples in the region of competence of a test sample. If no classifier is selected, KNORA-E removes from the region of competence the sample that is furthest from the test sample until at least one classifier is selected. KNORA-U selects all classifiers that correctly classify at least one sample in the region of competence, the more samples a classifier correctly classifies, the more votes it has for the classification of the test sample.

In (OLIVEIRA; CAVALCANTI; SABOURIN, 2017), Oliveira et al. showed that, when the region of competence of a test sample is composed of samples from different classes (indecision region), DES techniques can select classifiers that classify all samples in the region of competence as being from the same class. The authors then proposed the Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES) framework. This framework pre-selects classifiers with decision boundaries crossing the region of competence of the test sample if the test sample is located in an indecision region, preventing DES techniques from selecting classifiers that classify all samples in the region of competence

to the same class.

Considering KNORA-E, if no classifiers correctly classify all samples in the region of competence of a test sample, KNORA-E can change a region that was composed of samples from different classes into a smaller region composed of samples of a single class. FIRE-DES tackles this issue ensuring that, if the test sample is located in an indecision region, KNORA-E will select only classifiers with decision boundaries crossing the original region of competence.

Even though FIRE-DES tackles the indecision region problem of KNORA-E, the way in which KNORA-E reduces the region of competence can lead to the selection of incompetent classifiers, even when using FIRE-DES, as the pre-selection of classifiers is performed only once over the original region of competence. In this paper, we propose two DES techniques: K-Nearest Oracles Borderline (KNORA-B) and K-Nearest Oracles Borderline Imbalanced (KNORA-BI). KNORA-B is a DES technique based on KNORA-E that prevents the underrepresentation of classes in the region of competence when it is composed of samples from different classes. KNORA-BI is a variation of KNORA-B for imbalanced datasets that tackles the class imbalance problem (LÓPEZ et al., 2013) by preventing only the underrepresentation of the minority class (class with few samples) in the region of competence - but allowing the underrepresentation of the majority class (class with many samples).

The remainder of this paper is organized as follows: Section II presents the KNORA-E and KNORA-U. Section III presents the problem statement. Section IV presents the proposed techniques KNORA-B and KNORA-BI. Section IV presents the experiments. Finally, Section V presents the conclusion.

A.2 Background

The Oracle concept is a hypothetical dynamic selection approach that always selects the classifier that correctly classifies the test sample, if such classifier exists (KUNCHEVA, 2002).

In (KO; SABOURIN; JR, 2008), Ko et al. introduced the concept of K-Nearest Oracles and proposed two DES techniques: K-Nearest Oracles Union (KNORA-U) and K-Nearest Oracles Eliminate (KNORA-E). For the classification of a test sample, KNORA-E and KNORA-U find the region of competence of the test sample and use the samples in this region as *oracles* to perform the selection of classifiers (knowing the classifiers that correctly classify each sample in the region of competence). The following subsection details the KNORA-Eliminate.

A.2.1 KNORA-Eliminate

Given a test sample x_{query} to be classified, KNORA-E finds the region of competence (Ψ) of x_{query} by selecting the K nearest neighbors of x_{query} in the validation set $\mathcal{D}_{SEL}$. After that, KNORA-E selects all classifiers that correctly classify all samples in Ψ. If no classifiers correctly classify all samples in Ψ, KNORA-E reduces Ψ by removing the sample that is furthest from x_{query}, until at least one classifier is selected. If Ψ gets empty, and no classifier was selected, KNORA-E selects all classifiers with the same classification accuracy as the single best classifier in the original Ψ.

Fig. 29 shows a test sample ▲, its region of competence (darkened samples), and regions of expertise of the classifiers (circles on the right side). KNORA-E selects all classifiers that are experts for the classification of all samples in the region of competence (intersection of correct classifiers).

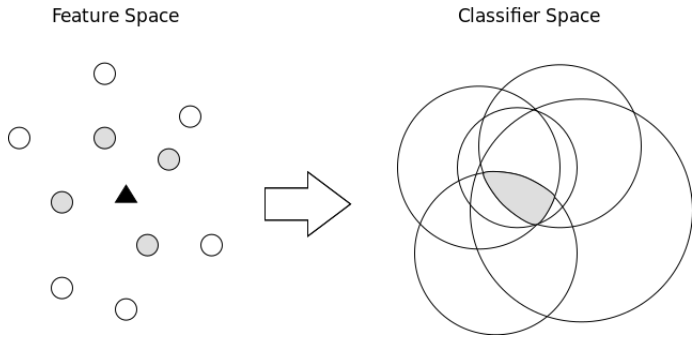
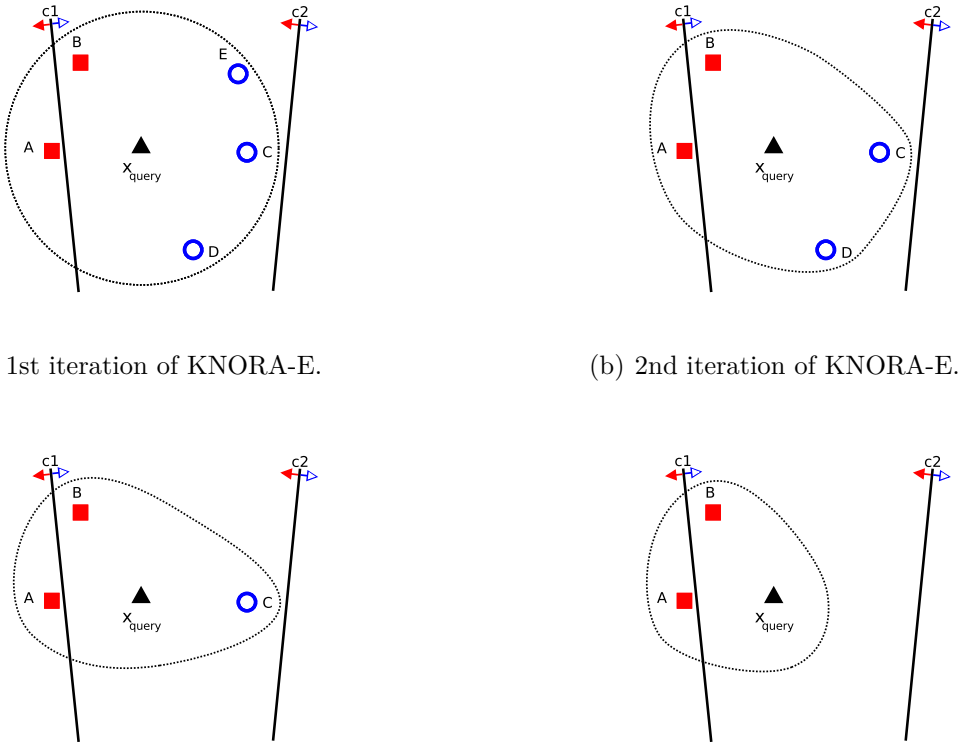


Figure 29 – Selection of KNORA-E. On the left side, ▲ is the test sample, and the darkened samples are the region of competence of the test sample. On the right side, the selected classifiers are darkened. (From (KO; SABOURIN; JR, 2008)).

A.3 Problem Statement

When no classifier correctly classifies all samples in the region of competence of a test sample, KNORA-E reduces the region of competence by removing the sample that is the furthest from x_{query}, regardless of its class. Because of that, KNORA-E can change a region of competence that is composed of samples from different classes into a region of competence composed of samples from a single class.

Figure 30 presents the iterations of KNORA-E ($K = 5$) for the classification of a test sample (x_{query}) when no classifier correctly classifies all samples in the region of competence (Ψ) of x_{query}. In this figure, ▲ is the test sample, the dotted line delimits the region of competence, A, B, C, D, and E are the K nearest neighbors of x_{query} in the validation set, $c1$ and $c2$ are classifiers and the continuous straight lines are their decision boundaries. The markers ○ and ■ represent samples from different classes, and the test sample is from the class "○".



(a) 1st iteration of KNORA-E.

(b) 2nd iteration of KNORA-E.

(c) 3rd iteration of KNORA-E.

(d) 4th iteration of KNORA-E.

Figure 30 – KNORA-E iterations when no classifier correctly classifies all samples in the region of competence of the test sample. The ▲ is the test sample, the markers ○ and ■ are samples from different classes in the region of competence of the test sample, and the class of the test sample is "○". (Adapted from (OLIVEIRA; CAVALCANTI; SABOURIN, 2017)).

In the scenario from Figure 30, in the first and second iterations (Figure 30(a) and 30(b)) no classifier correctly classifies all samples in Ψ, so KNORA-E removes the samples E and D from Ψ, in the first and second iterations, respectively.

In the third iteration (Figure 30(c)), again, no classifier correctly classifies all samples in Ψ, so KNORA-E removes the sample that is the furthest from x_{query}, C, (the last remaining sample of class "○" in Ψ), leaving only two samples of the class "■" in Ψ. In the fourth iteration (Figure 30(d)), the classifier $c2$ correctly classifies all samples in Ψ (A and B), so KNORA-E selects $c2$, misclassifying the test sample.

This is an issue because when the region of competence of the test sample has samples from different classes, KNORA-E may change the region of competence in such a way that it is no longer a good representation of the local region of the test sample. This behavior is not ideal because classifiers that classify all samples in the region of competence as "■" (such as $c2$) are selected. This is a problem especially when dealing

with imbalanced dataset, where a region of competence with samples from the minority class (class with few samples) can be changed into a region with only samples from the majority class (class with many samples), even though the minority class is usually the class of interest.

A.4 Proposed Techniques

A.4.1 KNORA-Borderline

The K-Nearest Oracles-Borderline (KNORA-B) is a DES technique based on KNORA-E that maintains the classes represented in the original region of competence of the test sample when reducing the region of competence.

Given a test sample x_{query}, KNORA-B finds its region of competence (Ψ) by selecting the K nearest neighbors of x_{query} in the validation set $\mathcal{D}_{SEL}$. Then, KNORA-B selects all classifiers that correctly classify all K samples in Ψ. If no classifier correctly classifies all samples in Ψ, KNORA-B reduces Ψ by removing the sample that is the furthest (Ψ_b) from x_{query}, only if all classes represented in Ψ are still represented in Ψ/Ψ_b, otherwise, KNORA-B evaluates the removal of the next furthest sample (Ψ_{b-1}). The process repeats until at least one classifier is selected. If KNORA-B reaches a state in which it is not possible to remove any sample from Ψ while maintaining the set of classes, KNORA-B uses the KNORA-E rule over the original region of competence.

Algorithm 11 presents the KNORA-B pseudocode. x_{query} is the test sample, C is the pool of classifiers, $\mathcal{D}_{SEL}$ is the validation set, and K is the size of the region of competence. First, KNORA-B initializes EoC as an empty list to insert the selected classifiers (Line 1), it then selects the region of competence Ψ of x_{query} by applying the K-nearest neighbors on $\mathcal{D}_{SEL}$ (Line 2), and stores this initial region of competence in $\Psi_{original}$ (Line 3). Next, until at least one classifier is selected or until the region of competence is empty (Line 4), KNORA-B tries to select all classifiers that correctly classify all samples in Ψ (Line 5 - 9), if no classifier is selected KNORA-B reduces the region of competence and the process repeats (Lines 10 - 12). If the region of competence can no longer be reduced, and no classifier was selected, KNORA-B performs the *fallback selection* (Lines 14 - 16) - the fallback selection of KNORA-B is the KNORA-E procedure. Finally, KNORA-B returns the selected ensemble of classifiers.

KNORA-B differs from KNORA-E in the *reduced region of competence* procedure applied when no classifier correctly classifies all samples in the region of competence (Ψ). Algorithm 12 presents the region of competence reduction process. Given a test sample x_{query} and the region of competence Ψ, KNORA-B gets the size S of the region of competence (Line 1), and assigns that to a variable b (Line 2). Now, until the region of

Algorithm 11 KNORA-B

Require: $\mathcal{C}$: pool of classifiers
Require: $\mathcal{D}_{SEL}$: validation set
Require: x_{query}: test sample
Require: K: size of the region of competence
1: $EoC \leftarrow$ ensemble of classifiers
2: $\Psi \leftarrow K$ nearest neighbors of x_{query} in $\mathcal{D}_{SEL}$
3: $\Psi_{original} \leftarrow \Psi$
4: **while** Empty(EoC) $\wedge$ $\neg$Empty(Ψ) **do**
5: **for all** c_i in $\mathcal{C}$ **do**
6: **if** c_i correctly classifies all samples in Ψ **then**
7: $EoC \leftarrow EoC \cup c_i$
8: **end if**
9: **end for**
10: **if** EoC is empty **then**
11: $\Psi \leftarrow$*reduced_region_of_competence*(x_{query}, Ψ)
12: **end if**
13: **end while**
14: **if** EoC is empty **then**
15: $EoC \leftarrow$ fallback_selection($C, \Psi_{original}, x_{query}, K$)
16: **end if**
17: **return** EoC

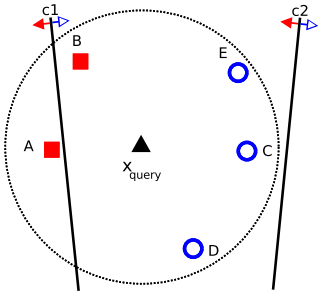
competence is not reduced and b is greater than zero (Line 3), KNORA-B gets the b-th nearest sample (starts with $b = SizeOf(\Psi)$, that is, from the furthest to the nearest) of x_{query} (Ψ_b) in Ψ (Line 4), and evaluates if all classes represented in Ψ are still represented if Ψ_b is removed from Ψ. If so, Ψ_b is removed from Ψ, otherwise, b is decreased (Lines 5 - 9). If no sample was removed from Ψ, this means that no reduction was possible (there is only one sample from each class that was represented in the original region of competence), then Ψ is assigned to an empty set (Lines 11 - 13) so that KNORA-B can use the fallback. Finally, the reduced region of competence is returned (Line 14).

Figure 31 presents the iterations of KNORA-B ($K = 5$) for the classification of a test sample (x_{query}) when no classifiers correctly classify all samples in the region of competence (Ψ) of x_{query}. In this figure, ▲ is the test sample, the dotted line delimits the region of competence, A, B, C, D, and E are the K nearest neighbors of x_{query} in the validation set, $c1$ and $c2$ are classifiers and the continuous straight lines are their decision boundaries. The markers ○ and ■ are samples from different classes, and the test sample is from the class "○".

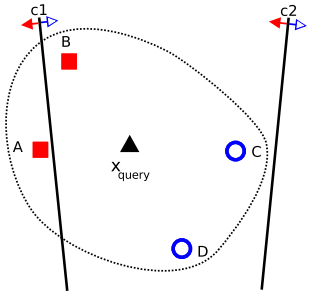
In the scenario from Figure 31, in the first and second iterations (Figure 31(a) and 31(b)) no classifier correctly classifies all samples in Ψ, so, KNORA-B removes the sample that is the furthest from x_{query}, respectively, E and D, from Ψ. In the third iteration (Figure 31(c)), again, no classifier correctly classifies all samples in Ψ, but instead of

Algorithm 12 KNORA-B - Reduced Region of Competence

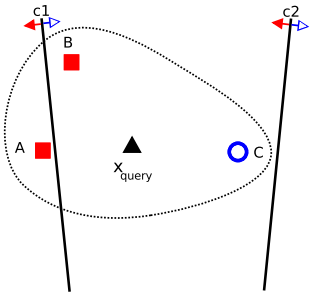
Require: x_{query}: test sample
Require: Ψ: region of competence of x_{query}
1: $S \leftarrow SizeOf(\Psi)$
2: $b \leftarrow$ S
3: **while** $SizeOf(\Psi) = S$ and $b > 0$ **do**
4: $\quad \Psi_b \leftarrow$ b-th nearest from x_{query} in Ψ
5: $\quad$ **if** $Set(classes(\Psi/\Psi_b)) = Set(classes(\Psi))$ **then**
6: $\quad\quad \Psi \leftarrow \Psi/\Psi_b$
7: $\quad$ **else**
8: $\quad\quad b \leftarrow b - 1$
9: $\quad$ **end if**
10: **end while**
11: **if** $SizeOf(\Psi) = S$ **then**
12: $\quad \Psi \leftarrow \emptyset$
13: **end if**
14: **return** Ψ



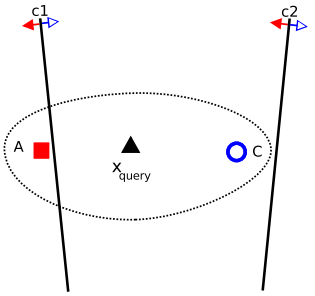
(a) 1st iteration of KNORA-B.



(b) 2nd iteration of KNORA-B.



(c) 3rd iteration of KNORA-B.



(d) 4th iteration of KNORA-B.

Figure 31 – KNORA-B iterations when no classifier correctly classifies all samples in the region of competence of the test sample. The ▲ is the test sample, the markers ○ and ■ are samples from different classes in the region of competence of the test sample, and the class of the test sample is "○". (Adapted from (OLIVEIRA; CAVALCANTI; SABOURIN, 2017)).

removing the furthest sample C, leaving only samples from the class "■" in Ψ, KNORA-B removes the second furthest sample B, maintaining samples from both classes "○" and "■" in Ψ. In the fourth iteration (Figure 31(d)), the classifier $c1$ correctly classifies all samples in Ψ (A and C), so, KNORA-B selects $c1$ and correctly classifies the test sample as being from the class "○".

Comparing Figure 30 with Figure 31, we can see that KNORA-B is better in selecting classifiers for the classification of the test sample that has a region of competence with samples of different classes. While KNORA-E selects $c2$, a base classifier that classifies all samples in Ψ as being from the same class "■", and misclassifies the test sample, KNORA-B selects $c1$, a base classifier that correctly classifies samples from different classes in Ψ, correctly classifying the test sample.

A.4.2 KNORA-Borderline-Imbalanced

When dealing with specific problems such as imbalanced datasets, KNORA-B can remove a minority class sample that is closer to the test sample instead of a majority class sample that is the furthest. This behavior is not desired in the context of imbalanced datasets, in which the minority class (class with few samples) is the class of interest (LÓPEZ et al., 2013).

KNORA-Borderline-Imbalanced (KNORA-BI) is a variation of KNORA-B that has a different reduced region of competence reduction procedure that allows the reduction of a region of competence that is composed of samples from the majority and minority classes into a region of competence composed only of samples from the minority class. By doing so, KNORA-BI favors only the minority class when reducing the region of competence.

KNORA-BI region of competence reduction pseudocode is presented in Algorithm 13. Given a test sample x_{query} the region of competence Ψ, and the minority class class_{min}, KNORA-BI gets the size S of the region of competence (Line 1) and assigns that to a variable b (Line 2). Now, until the region of competence is not reduced and b is greater than zero (Line 3), KNORA-BI gets the b-th nearest sample (starts with $b = \textit{SizeOf}(\Psi)$, that is, from the furthest to the nearest) of x_{query} (Ψ_b) in Ψ (Line 4), if Ψ_b is not from the minority class or if all classes represented in Ψ are still represented when Ψ_b is removed from Ψ, then, Ψ_b is removed from Ψ, otherwise, b is decreased (Lines 5 - 9). If no sample is removed from Ψ and b reaches zero, Ψ is assigned to an empty set (Lines 11 - 13). Thus, KNORA-BI can use the fallback. Finally, the reduced region of competence is returned (Line 14).

Considering the examples from Figure 30 and 31, KNORA-BI region of competence reduction procedure reduces the region of competence in such a way that:

- if the class "○" is the majority class, KNORA-BI acts as exemplified in Figure 30

Algorithm 13 KNORA-BI - Reduced Region of Competence

Require: x_{query}: test sample
Require: Ψ: region of competence of x_{query}
Require: class_{min}: minority class
1: $S \leftarrow$ *SizeOf*(Ψ)
2: $b \leftarrow$ S
3: **while** *SizeOf*(Ψ) $= S$ and $b > 0$ **do**
4: $\Psi_b \leftarrow$ b-th nearest from x_{query} in Ψ
5: **if** Class(Ψ_b) $\neq$ $class_{min} \vee$ *Set(classes*(Ψ/Ψ_b)) $=$ *Set(classes*(Ψ)) **then**
6: $\Psi \leftarrow \Psi/\Psi_b$
7: **else**
8: $b \leftarrow b - 1$
9: **end if**
10: **end while**
11: **if** *SizeOf*(Ψ) $= S$ **then**
12: $\Psi \leftarrow \emptyset$
13: **end if**
14: **return** Ψ

- as it allows the removal of all majority class samples, leaving all minority class samples.

- if the class "○" is the minority class, KNORA-BI acts as exemplified in Figure 31 - as it does not allow the removal of all minority class samples.

A.5 Experiments

We evaluated KNORA-B and KNORA-BI using 40 datasets as proposed in (OLIVEIRA; CAVALCANTI; SABOURIN, 2017). The datasets were taken from the imbalanced datasets module in the Knowledge Experiments based on Evolutionary Learning (KEEL) repository (ALCALÁ et al., 2010). Table 10 presents the details about the datasets used in our experiments: label, name, number of features, number of samples, and imbalance ratio (IR).

For each dataset, the data was partitioned using the *stratified 5-fold cross-validation* (1 fold used for testing, and 4 folds for validation/training) followed by a *stratified 4-fold cross-validation* (the 4 folds into validation/training divided in 1 for validation and 3 for training), resulting in 20 replications for each dataset using 20% for testing, 20% for validation, and 60% for training.

The analysis is conducted using 8 DES techniques from the literature, their respective FIRE-DES versions (using the F prefix), and 3 state-of-the-art DES techniques. Table 11 shows the dynamic selection techniques used in our experiments, their categories and references. Following the approach using in (OLIVEIRA; CAVALCANTI; SABOURIN,

Table 10 – Summary of the 40 datasets used in the experiments: label, name, number of features, number of samples, and imbalance ratio (from (OLIVEIRA; CAVALCANTI; SABOURIN, 2017)).

Label	Name	#Feats.	#Samples	IR
1	glass1	9	214	1.82
2	ecoli0vs1	7	220	1.86
3	wisconsin	9	683	1.86
4	pima	8	768	1.87
5	iris0	4	150	2.00
6	glass0	9	214	2.06
7	yeast1	8	1484	2.46
8	vehicle2	18	846	2.88
9	vehicle1	18	846	2.9
10	vehicle3	18	846	2.99
11	glass0123vs456	9	214	3.2
12	vehicle0	18	846	3.25
13	ecoli1	7	336	3.36
14	new-thyroid1	5	215	5.14
15	new-thyroid2	5	215	5.14
16	ecoli2	7	336	5.46
17	segment0	19	2308	6.00
18	glass6	9	214	6.38
19	yeast3	8	1484	8.10
20	ecoli3	7	336	8.60
21	yeast-2vs4	8	514	9.08
22	yeast-05679vs4	8	528	9.35
23	vowel0	13	988	9.98
24	glass-016vs2	9	192	10.29
25	glass2	9	214	11.59
26	shuttle-c0vsc4	9	1829	13.87
27	yeast-1vs7	7	459	14.30
28	glass4	9	214	15.47
29	ecoli4	7	336	15.80
30	page-blocks-13vs4	10	472	15.86
31	glass-0-1-6_vs_5	9	184	19.44
32	shuttle-c2-vs-c4	9	129	20.50
33	yeast-1458vs7	8	693	22.10
34	glass5	9	214	22.78
35	yeast-2vs8	8	482	23.10
36	yeast4	8	1484	28.10
37	yeast-1289vs7	8	947	30.57
38	yeast5	8	1484	32.73
39	ecoli-0137vs26	7	281	39.14
40	yeast6	8	1484	41.40

2017), we use a pool of classifiers composed of 100 Perceptrons generated using the Bootstrap AGGregatING (Bagging) technique (BREIMAN, 1996), and a region of competence size $K = 7$.

Table 11 – Dynamic selection techniques considered in the experiments (From (OLIVEIRA; CAVALCANTI; SABOURIN, 2017)).

Technique	Category	Reference
DES		
Overall Local Accuracy (OLA)	Accuracy	Woods et al. (WOODS; KEGELMEYER; BOWYER, 1997)
Local Class Accuracy (LCA)	Accuracy	Woods et al. (WOODS; KEGELMEYER; BOWYER, 1997)
A Priori (APri)	Probabilistic	Giacinto et al. (GIACINTO; ROLI, 1999)
A Posteriori (APos)	Probabilistic	Giacinto et al. (GIACINTO; ROLI, 1999)
Multiple Classifier Behavior (MCB)	Behavior	Giacinto et al. (GIACINTO; ROLI, 2001b)
Dynamic Selection KNN (DSKNN)	Diversity	Santana et al. (SANTANA et al., 2006)
K-Nearests Oracles Union (KNORA-U)	Oracle	Ko et al. (KO; SABOURIN; JR, 2008)
K-Nearests Oracles Eliminate (KNORA-E)	Oracle	Ko et al. (KO; SABOURIN; JR, 2008)
State-of-the-art		
Randomized Reference Classifier (RRC)	Probabilistic	Woloszynski et al. (WOLOSZYNSKI; KURZYNSKI, 2011)
META-DES	Meta-learning	Cruz et al. (CRUZ et al., 2015)
META-DES.Oracle	Meta-learning	Cruz et al. (CRUZ; SABOURIN; CAVALCANTI, 2017b)

Following the approach in (OLIVEIRA; CAVALCANTI; SABOURIN, 2017), we used the Area Under the ROC Curve (AUC) (BRADLEY, 1997) for performance evaluation since it is a suitable metric for binary imbalanced datasets (LÓPEZ et al., 2013) (OLIVEIRA; CAVALCANTI; SABOURIN, 2017). We also used the Wilcoxon Signed Rank Test (BENAVOLI; CORANI; MANGILI, 2016) (WILCOXON, 1945) and the Sign Test (DEMŠAR, 2006) to perform a pairwise comparison of the proposed techniques with the techniques from the literature.

A.5.1 Results

Table 12 presents the overall results. For each technique, the table shows: the mean AUC and standard deviation, the average ranking, and the *p-value* and result of the Wilcoxon Signed Rank Test comparing KNORA-B and KNORA-BI with the DES technique (+/=/− signs mean the proposed technique had statistically better, equal, and worse classification performance considering a confidence level $\alpha = 0.05$).

Table 12 show that KNORA-BI achieved the highest mean AUC (0.8136) and the best average ranking (6.80), outperforming all techniques considered in this work. Moreover, it statistically outperformed 18 out of 22 techniques according to the Wilcoxon Signed Rank test. Table 12 also shows that KNORA-B was not as good as KNORA-BI, achieving the 12th best AUC (0.7989), and being statistically equivalent to KNORA-E (0.8003). However, it was only statistically outperformed by 7 out of 22 techniques, where 3 of those are variations of the techniques proposed in this paper (KNORA-BI, FKNORA-B, and FKNORA-BI).

Table 12 – Overall results.

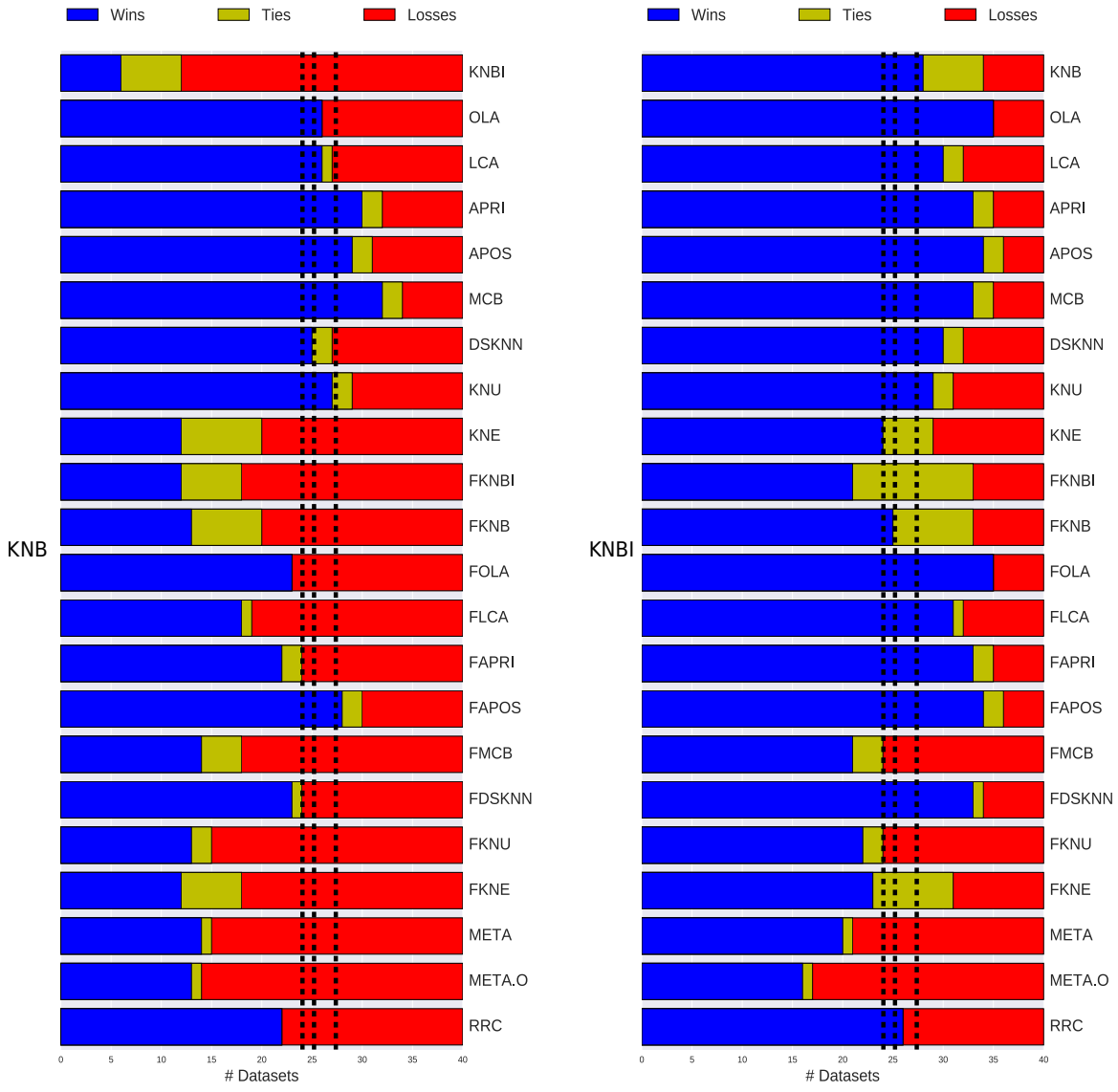
DES	AUC	RANK	KNORA-B (*p-value*)		KNORA-BI (*p-value*)	
KNORA-BI	**0.8136 (0.0743)**	**6.80**	0.9999	−	*N/A*	
META.O	0.8067 (0.0649)	8.16	0.9509	−	0.3454	=
META	0.8100 (0.0635)	8.20	0.9742	−	0.4362	=
FKNORA-U	0.8081 (0.0765)	8.38	0.9712	−	0.1009	=
FMCB	0.8058 (0.0760)	8.80	0.9235	=	0.0921	=
FKNORA-BI	0.8083 (0.0758)	9.01	0.9976	−	0.0003	+
FKNORA-E	0.8055 (0.0768)	9.59	0.9962	−	0.0002	+
FKNORA-B	0.8042 (0.0772)	10.34	0.9860	−	0.0001	+
KNORA-E	0.8003 (0.0703)	10.66	0.8298	=	0.0002	+
KNORA-B	0.7989 (0.0721)	11.28	*N/A*		$5.60 \times e^{-6}$	+
FDSKNN	0.8006 (0.0767)	11.43	0.5250	=	$2.00 \times e^{-6}$	+
FLCA	0.7946 (0.0777)	11.84	0.5554	=	$8.30 \times e^{-6}$	+
RRC	0.7934 (0.0658)	12.26	0.1805	=	0.0057	+
FOLA	0.8017 (0.0767)	12.97	0.5026	=	$3.80 \times e^{-6}$	+
FAPRI	0.7930 (0.0802)	13.10	0.1940	=	$1.60 \times e^{-6}$	+
LCA	0.7809 (0.0737)	13.30	0.0332	+	$7.20 \times e^{-6}$	+
DSKNN	0.7728 (0.0602)	13.32	0.0107	+	$6.10 \times e^{-5}$	+
OLA	0.7911 (0.0709)	13.60	0.0623	=	$1.40 \times e^{-7}$	+
KNORA-U	0.7560 (0.0548)	15.39	0.0002	+	$6.80 \times e^{-6}$	+
FAPOS	0.7681 (0.0809)	15.86	0.0001	+	$1.40 \times e^{-7}$	+
APRI	0.7537 (0.0628)	16.38	$3.20 \times e^{-5}$	+	$9.80 \times e^{-7}$	+
APOS	0.7380 (0.0658)	18.01	$1.10 \times e^{-5}$	+	$4.70 \times e^{-7}$	+
MCB	0.7443 (0.0566)	17.32	$2.10 \times e^{-6}$	+	$6.40 \times e^{-7}$	+

In addition, FKNORA-B (0.8042) outperformed KNORA-B (0.7989) with statistical confidence, meaning FIRE-DES caused a significant increase in classification performance in KNORA-B. This increase in AUC is explained in the scenario that no classifier was selected and the region of competence has only one sample from each class remaining. In this scenario, KNORA-B applies KNORA-E on the entire pool of classifiers while FKNORA-B applies KNORA-E on the set of pre-selected classifiers (avoiding the selection of incompetent classifiers that classify all samples in the region of competence as being from the same class).

On the other hand, KNORA-BI (0.8136) outperformed FKNORA-BI (0.8083) with statistical confidence, meaning FIRE-DES caused a significant decrease in classification performance in KNORA-BI. KNORA-BI allows the region of competence to be reduced until there is only minority class samples, allowing to the selection of classifiers that classify all samples in the region of competence as minority class samples, while FKNORA-BI does not allow the selection of such classifiers. Because the minority class is so rare, preventing a DES technique from selecting a classifier that classifies all samples in the region of competence as minority class sample is not advantageous, specially because correctly classifying 1 minority class sample has a higher impact than misclassifying 1 majority

class sample, which explains why KNORA-BI was better without FIRE-DES.

Figure 32 presents a pairwise comparison using the Sign Test (DEMŠAR, 2006), calculated using the wins, ties, and losses achieved by the KNORA-B compared to other techniques (Figure 32(a)) and by the KNORA-BI compared to other techniques (Figure 32(b)).



(a) Comparison of KNORA-B and other techniques. (b) Comparison of KNORA-BI and other techniques.

Figure 32 – Classification performance comparison of KNORA-B with different DES techniques (Figure 32(a)) and KNORA-BI with different DES techniques (Figure 32(b)) in terms of wins, ties and losses considering the average AUC in the 40 datasets. The dashed lines (left to right) illustrates the critical values $n_c = \{24.05, 25.20, 27.37\}$ considering significance levels of $\alpha = \{0.10, 0.05, 0.01\}$, respectively.

For a comparison between one of the proposed techniques and a technique T, the

null hypothesis H_0 was that the proposed technique is not statistically different than T, and a rejection of H_0 meant that the proposed technique is statistically better than T. H_0 is rejected if the number of wins plus half of the number of ties is greater or equal to n_c (Equation A.1):

$$n_c = \frac{n_{exp}}{2} + z_\alpha \times \frac{\sqrt[2]{n_{exp}}}{2} \tag{A.1}$$

where $n_{exp} = 40$ (the number of experiments), $n_c = \{24.05, 25.20, 27.37\}$, respectively for the levels of significance $\alpha = \{0.10, 0.05, 0.01\}$.

Figure 32(a) shows that, considering $\alpha = 0.05$, KNORA-B statistically outperformed 7 out of the 8 regular DES techniques, being only statistically equivalent to KNORA-E. Comparing with FIRE-DES framework, KNORA-B outperformed only FAPOS (the worst of FIRE-DES in our experiments). KNORA-B was not better than any of the state-of-art DES techniques.

On the other hand, Figure 32(b) shows that overall KNORA-BI statistically outperformed a significant number of techniques studied (18 out of 22), considering a significance level $\alpha = 0.05$. The only exceptions being the FMCB, FKNORA-U, META-DES, and META-DES.Oracle.

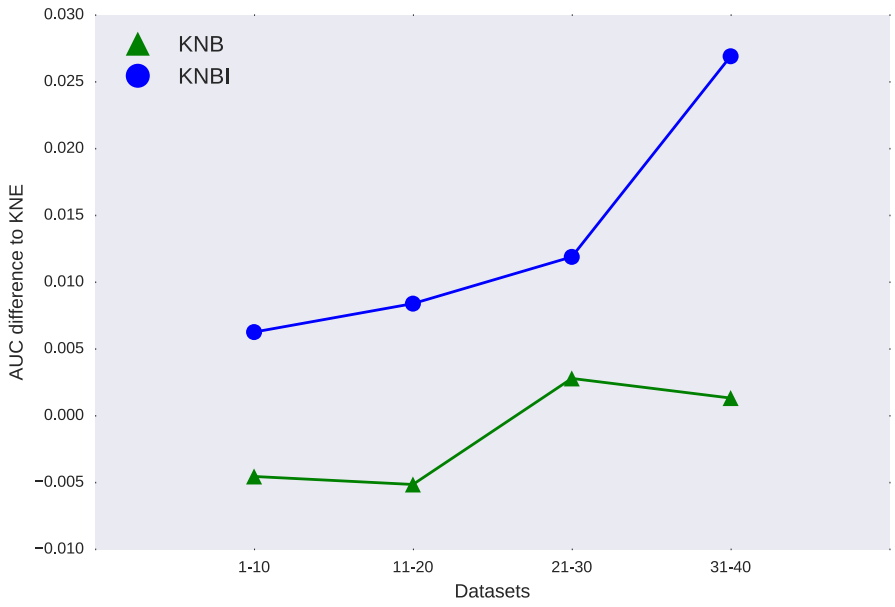


Figure 33 – AUC difference between KNORA-B and KNORA-E, and between KNORA-BI and KNORA-E.

Figure 33 presents the average AUC difference from KNORA-B and KNORA-BI to KNORA-E considering the datasets 1-10, 11-20, 21-30, and 31-40. This figure shows that KNORA-B is better than KNORA-E optimally in 21-30 and slightly better in 31-40, while KNORA-BI was always better than KNORA-E (the more the imbalance ratio the higher

the difference). This confirms that for all imbalance levels KNORA-BI was better than KNORA-E, while KNORA-B was only better than KNORA-B for high imbalance level.

Based on the results, we can state with confidence that KNORA-B achieved statistically equivalent performance of KNORA-E, and KNORA-BI statistically outperformed KNORA-E and all other techniques considered in this experiment.

A.6 Conclusion

In this paper, we proposed two DES techniques: K-Nearest Oracles Borderline (KNORA-B) and K-Nearest Oracles Borderline Imbalanced (KNORA-BI). KNORA-B is a DES technique based on KNORA-E that selects all classifiers that correctly classify all samples in the region of competence. If no classifier is selected, KNORA-B reduces the region of competence maintaining the at least one sample from each class in the original region of competence. KNORA-BI is a variation of KNORA-B that reduces the region of competence, but maintaining at least one sample of the minority class (if such sample exists in the original region of competence).

We conducted experiments using 40 datasets from the KEEL software (ALCALÁ et al., 2010) with different levels of imbalance, and compared KNORA-B and KNORA-BI with 8 regular DES techniques, 8 FIRE-DES techniques, and 3 state-of-the-art DES techniques.

Results showed that KNORA-B had statistically equivalent performance of KNORA-E, and KNORA-BI statistically outperformed KNORA-E in classification performance. In fact, KNORA-BI achieved the best average classification performance over all DES techniques considered in our experiments, including the state-of-the-art DES techniques.

APPENDIX B – A BOOTSTRAP-BASED ITERATIVE SELECTION FOR ENSEMBLE GENERATION

Abstract

We propose a bootstrap-based iterative method for generating classifier ensembles called *Iterative Classifier Selection Bagging* (ICS-Bagging). Each iteration of ICS-Bagging has two phases: i) bootstrap sampling to generate a pool of classifiers; and, ii) selection of the best classifier of the pool using a fitness function based on the ensemble accuracy and diversity. The selected classifier is added to the final ensemble. The bootstrap sampling runs on each iteration and updates the probability of sampling per class based on the class accuracy. This process is repeated until the number of classifiers in the final ensemble is reached. For the specific case of imbalanced datasets, we also propose the SMOTE-ICS-Bagging, a variation of the ICS-Bagging that runs SMOTE at the beginning of each iteration in order to reduce the class imbalance before data sampling. We compared the proposed techniques with Bagging, Random Subspace and SMOTEBagging, using 15 imbalanced datasets from KEEL. The results show the proposed techniques outperform all other techniques in accuracy. Ranking diagrams revealed that the proposed algorithms achieved the highest rankings in accuracy, outperforming SMOTEBagging, a renowned ensemble generation method for imbalanced datasets.

B.1 Introduction

In many real-world classification datasets, the instances of one class are greatly outnumbered by the instances of the other classes. This imbalance gives rise to the so called class imbalance problem, which is the problem of learning a class that has a small set of examples when compared to the other classes. In recent years, the class imbalance problem has emerged as one of the great challenges of data mining (YANG; WU, 2006), and it is very common in practice, being present in areas such as medical diagnosis (MAZUROWSKI et al., 2008), fraud detection (CIESLAK; CHAWLA; STRIEGEL, 2006), and face recognition (LIU; CHEN, 2005).

On imbalanced datasets, the underrepresented class, called the minority or positive class, is usually the class of interest (i.e. higher misclassification cost), which makes it essential that the few instances available for this class be appropriately learned by the model. This is not usually the case for standard classification learning algorithms, as most of them do not take class distribution into account and have a strong bias towards the majority class. This results in poor learning of the minority class. As this class is so important in these types of problems, it makes sense to explicitly deal with the class imbalance problem.

Numerous methods exist to treat imbalanced datasets (HULSE; KHOSHGOFTAAR; NAPOLITANO, 2007) (HE; GARCIA, 2009) (LÓPEZ et al., 2013) (GALAR et al., 2012), (LÓPEZ et al., 2013). These methods can be categorized into 3 major groups: (1) data sampling, in which the dataset is preprocessed so that a standard learning method can be used (AKBANI; KWEK; JAPKOWICZ, 2004); (2) algorithmic modification, which involves adapting standard learning methods to take class distribution into account; (3) cost-senstive learning (ELKAN, 2001), that uses approaches at the data level, at the algorithmic level, or both.

Ensemble methods (DIETTERICH, 2000) (KUNCHEVA, 2004) (WOŹNIAK; GRAÑA; CORCHADO, 2014) (also known as Multiple Classifier Systems) are an important area of research in machine learning, and have been proven in practice to increase the accuracy of classifier systems (WOŹNIAK; GRAÑA; CORCHADO, 2014). These methods consist of training not just one classifier (i.e. expert), but several, and combining their outputs (i.e. expert opinions) in the hope that the results will be better than any classifier, or at least trying to avoid the choice of the worst classifier.

The process to generate good classifiers for an ensemble is not trivial, as the criteria for a "good" ensemble is not only the accuracy of its classifiers, but that they are as uncorrelated as possible. To heuristically measure this degree of "uncorrelatedness", the concept of diversity (KUNCHEVA; WHITAKER, 2003) (TANG; SUGANTHAN; YAO, 2006) is used. Since the training of each classifier already optimizes for accuracy, the ensemble

designer must choose how to improve diversity in the ensemble. There are several methods that do this, and they can be divided in: (1) training data manipulation (BREIMAN, 1996) (SCHAPIRE, 1990) (HO, 1998), (2) randomization (HANSEN; SALAMON, 1990) and (3) different models/architectures (CARUANA et al., 2004). Ensemble methods are also frequently adapted to imbalanced domains (GALAR et al., 2012), either by preprocessing the data before training each classifier (WANG; YAO, 2009)(CHAWLA et al., 2003) or by using a cost-sensitive framework.

In this paper, we proposed a technique for generating classifier ensembles called Iterative Classifier Selection Bagging (ICS-Bagging). ICS-Bagging is a bootstrap-based iterative method composed of two steps: first, a bootstrap sampling generates a pool of classifiers, and, after, the best classifier of the pool is selected using a fitness function based on the ensemble accuracy and diversity.

The main contributions of this paper are two new proposed methods: (1) ICS-Bagging: an ensemble generation method that chooses what classifiers should be added in the final ensemble by using ensemble accuracy and diversity; (2) SMOTE-ICS-Bagging (SICS-Bagging): a variant of ICS-Bagging that explicitly deals with the class imbalance problem. Both methods are shown to beat the state-of-the-art according to a recent survey (GALAR et al., 2012).

This paper is organized as follows. Section 2 presents the background for the proposed methods, going into more detail about imbalanced datasets solutions and ensemble methods. Section 3 presents ICS-Bagging and SICS-Bagging. Next, in Section 4, we detail the experiments, and in Section 5 we present the conclusions.

B.2 Background

In the field of classification, imbalanced datasets are a common occurrence and dealing with such datasets is difficult problem (OLIVEIRA et al., 2012) (YANG; WU, 2006). The main characteristic of such datasets is the fact that it has an imbalance in the class distribution.

Most of the standard learning algorithms expect a balanced training set. Therefore, good models for standard classification are not necessarily the best for imbalanced datasets. Some of the reasons for this are (LÓPEZ et al., 2013):

- The use of performance metrics that do not take class imbalance into account, such as the accuracy rate, may provide an advantage to the majority class.
- Classification rules that predict the minority class may be too specific, and so they are discarded in favor of more general rules.

- Very small clusters of minority class examples can be identified as noise, and therefore they could be wrongly discarded by the classifier. Also, a few noisy examples can degrade the identification of the minority class, since it has fewer examples to train with.

Multiple methods exist to deal with the class imbalance problem, (LÓPEZ et al., 2013) categorizes these methods into 3 major groups:

- **Data sampling**: training data is modified in such a way to produce a more or less balanced class distribution that allow classifiers to perform in a similar manner to standard classification.
- **Algorithmic modification**: adapts standard learning algorithms to be more attuned to class imbalance issues.
- **Cost-sensitive learning** (ELKAN, 2001): This type of solutions incorporate approaches at the data level, at the algorithmic level, or at both levels combined, considering higher costs for the misclassification of examples of the positive class with respect to the negative class, and therefore, trying to minimize higher cost errors.

B.2.1 Data sampling

One of the simpler alternatives (listed above) for dealing with imbalanced datasets, is preprocessing the data so as to diminish or eliminate the imbalance. In this vein, there are 3 options: (1) oversampling the minority class (CHAWLA et al., 2002), (2) undersampling the majority class (KUBAT; MATWIN, 1997) (BATISTA; CARVALHO; MONARD, 2000), (3) combining oversampling and undersampling (STEFANOWSKI; WILK, 2008). In this paper we focus on SMOTE (CHAWLA et al., 2002), which is an oversampling technique that was shown to be very effective as a preprocessing step for dealing with the class imbalance problem (GALAR et al., 2012).

SMOTE (CHAWLA et al., 2002) is an oversampling technique that creates synthetic examples by interpolating between minority instances that are close to each other. Synthetic instances are created along the line segment between each minority class instance and one of its k nearest neighbors (from the same class). The procedure is basically this: for each minority class example, randomly select one of its k nearest neighbors (from the same class), take the difference between the two vectors, multiply the difference by a random number (between 0 and 1) and add it to the minority class example.

The intuition on why SMOTE improves performance on imbalanced datasets is that it provides more related minority class samples to learn from, thus allowing a learning

algorithm to create broader decision regions, leading to more coverage of the minority class. This of course, has an implicit locality bias, that is, it assumes that interpolating two nearby samples of the same class, will generate a point in the same class. That is not always the case, so one of the problems of SMOTE is its sensitivity to the complexity of the dataset.

B.2.2 Ensemble generation

B.2.2.1 *Bagging*

Bagging (BREIMAN, 1996) is an ensemble meta-algorithm that was designed to improve the accuracy and stability of supervised machine learning models. Its main concept is bootstrap aggregation, where the training set for each classifier is constructed by random uniform sampling (with replacement) instances from the original training set (usually keeping the size of the original data) (BŁASZCZYŃSKI; STEFANOWSKI; IDKOWIAK, 2013). The classifiers output is then combined in some way (for classification problems, majority vote is generally used). The bootstrap sampling process generates a different training set for each classifier, which naturally increases ensemble diversity.

B.2.2.2 *Boosting*

Boosting (SCHAPIRE, 1990) (FREUND, 1995) is a general method for improving the accuracy of any given learning algorithm (SCHAPIRE, 2003). The objective of boosting methods is to produce a very accurate (i.e. "strong") classifier by combining rough and somewhat inaccurate (i.e. "weak") classifiers. The boosting method works by iteratively training each classifier, by feeding it a weighted training set. The weights in each instance of the training set are higher for instances often misclassified by the preceding classifiers. This effectively forces each classifier to focus on the current "hardest" examples. After the ensemble is generated, a simple combination scheme may be used (e.g. majority voting).

B.2.3 Ensemble Generation for Imbalanced Datasets

The use of ensembles for dealing with imbalanced datasets is one common solution (LÓPEZ et al., 2013), and (GALAR et al., 2012) roughly categorizes these techniques into: (1) bagging-based; (2) boosting-based; (3) hybrid; and (4) cost-sensitive ensembles. Out of these types, we're interested in comparing our proposed technique with the bagging-based SMOTEBagging, since it obtained slightly better results than the best (more robust) techniques tested in (GALAR et al., 2012) and (LÓPEZ et al., 2013).

In SMOTEBagging (WANG; YAO, 2009), the bootstrapping procedure used in the original Bagging, is modified so that even more diversity is introduced in the ensemble, this is done by using SMOTE. This technique, at first, uses SMOTE for oversampling

the minority class, and then its resampling rate is updated in each iteration (i.e. for each classifier that is trained), from lower to higher values (e.g. 10% - 100%). This ratio defines the number of minority instances to be additionally resampled in each iteration (BŁASZCZYŃSKI; STEFANOWSKI; IDKOWIAK, 2013).

B.3 Proposed Algorithms

This section presents the *Iterative Classifier Selection Bagging* (ICS-Bagging) and the *SMOTE Iterative Classifier Selection Bagging* (SICS-Bagging).

B.3.1 ICS-Bagging

ICS-Bagging is a bootstrap-based iterative methods for generating classifier ensembles. Each iteration generates a set of classifiers and selects the best classifier to the ensemble. The bootstrap sampling uses a probability of sampling from each class, with this probability being derived from the class accuracy. Figure 34 presents the architecture of the ICS-Bagging algorithm.

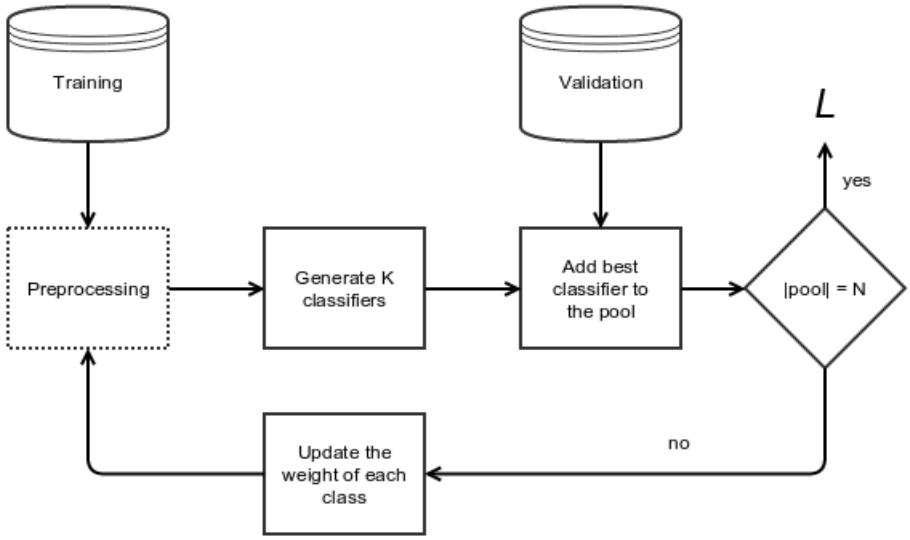


Figure 34 – Architecture of ICS-Bagging and SICS-Bagging. Where L is the final ensemble, and Preprocessing is a step from the SICS-Bagging

Follows the explanation of each step:

Preprocessing: Before generating the classifiers, a preprocessing technique might be applied to the training set. This preprocessing can consist of removing or generating features, removing outliers, noisy and redundant data, or generating new data. ICS-Bagging does not have this step, it is only used in the SICS-Bagging and is mentioned later in this section.

Generate K classifiers: This step generates K classifiers using bootstrap sampling (with replacement). In the first step, the weights are the same for all classes, after the first step the weights are updated to prioritize the class that has a lower accuracy.

The motivation for using the weights to guide the bootstrap process is that the new generated classifiers are trained with instances that increase the accuracy of the class with the lowest accuracy.

The motivation for generating $K > 1$ classifiers is to expand a region of search, increasing the probability of finding a classifier that considerably increase the classification accuracy and diversity.

Add the best classifier to the pool: For each of the K generated classifiers the following steps are performed to find the best classifier. Algorithm 14 presents the mechanism to find the classifier to be inserted into the pool.

Algorithm 14 Find Best Classifier

Require: $\mathcal{V}$: validation set
Require: $\mathcal{C}$: list of classifiers
Require: $\mathcal{P}$: pool
Require: *fitness*: a fitness function
1: $best_{index} \leftarrow -1$
2: $best_{value} \leftarrow -1$
3: **for all** $i \in SizeOf(C)$ **do**
4: Add C_i to $\mathcal{P}$
5: $fit_i \leftarrow fitness(\mathcal{P})$
6: **if** $fit_i > best_{value}$ **then**
7: $best_{value} \leftarrow fit_i$
8: $best_{index} \leftarrow i$
9: **end if**
10: Remove C_i from $\mathcal{P}$
11: **end for**
12: **return** $\mathcal{C}_{best_{index}}$

In Algorithm 14, C is the list of K generated classifiers, $\mathcal{V}$ is the validation set (in this work, we used the traning set as the validation set) and $\mathcal{P}$ is the current pool of classifiers.

For all classifiers in C, the classifier is added to the pool (Line 4), and the the fitness of the pool is calculated (Line 5). The *fitness* is given by

$$fitness = \alpha \times ACC + (1 - \alpha) \times DIV \tag{B.1}$$

where ACC is the classification accuracy of the pool, DIV is the diversity metric, and α is the balance parameter that regulates the objective function between high classification accuracy and high diversity, and has a range of 0.51 to 0.99.

If the pool achieves the highest fitness with this classifier, the index of this classifier is saved in $best_{index}$ (Line 6 - 9). The classifier is removed from the pool (Line 10) and the process starts again with another classifier, until the best classifier is returned (Line 12).

For the classification of a test sample, any combination rule could be used. In this paper, we used the majority vote rule (KUNCHEVA, 2004) to combine the outputs of the classifiers in the pool.

Any classification and diversity metric can be used in Equation B.1, but both need to be normalized (between 0 and 1). In this paper we used the AUC as classification accuracy because of the imbalanced datasets issue, and the Entropy Measure E (KUNCHEVA; WHITAKER, 2003) as the diversity metric because of the simplicity and running speed.

The Entropy Measure E is a non-pairwise diversity measure that has it's highest value when half classifiers correctly classify a pattern and the other half doesn't. If all classifiers have the same agree on a classification, the ensemble is not considered diverse. The Entropy Measure E is described as

$$E = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{(L - \lceil \frac{L}{2} \rceil)} min\{l(z_j), L - l(z_j)\} \tag{B.2}$$

where L is the number of classifiers of the ensemble, N is the number of samples to be classified, $l(z_i)$ is a function that returns the number of classifiers that correctly classify the sample z_i. This diversity metric varies from 0 to 1, where 1 is the highest diversity and 0 is the lowest, therefore, there is no need for normalization when using this metric.

The motivation for adding only one of the K generated classifiers is because the accuracy of each class might change when the best classifier is inserted in the pool, which means, the pool now has different samples to prioritize in order to increase classification accuracy and diversity.

$|\mathbf{pool}| = \mathbf{N}$: If the pool already contains the desired number of classifiers the pool is returned.

Update the weight of each class: Since the accuracy of each class might have changed after inserting the new classifier in the pool, the weights need to be updated using Equation B.3,

$$weight_{class} = 1.0 - \frac{accuracy_{class}}{\sum_{c \in classes} accuracy_c} \tag{B.3}$$

where $weight_{class}$ is the weight of the class, $accuracy_{class}$ is the classification rate of the class, and $\sum_{c \in classes} accuracy_c$ is the sum of the classification rate of all classes.

As previously stated, the motivation for updating the weights is to increase the probability of training the new K classifiers with samples from the class with lower accuracy of the pool.

Return the pool: The final pool of classifiers L is returned.

B.3.2 SMOTE-ICS-Bagging

SMOTE-ICS-Bagging (SICS-Bagging) is a variation of ICS-Bagging in which the Synthetic Minority Over-sampling Technique (SMOTE) is used as a preprocessing phase before generating the K classifiers. This step is performed to increase diversity and to reduce the imbalance ratio when performing bootstrap sampling.

B.4 Experiments

This section presents the methodology used in the experiments, and the results of ICS-Bagging and SICS-Bagging.

B.4.1 Methodology

The ICS-Bagging and SICS-Bagging methods are evaluated using 15 imbalanced datasets from KEEL (ALCALÁ et al., 2010). The datasets are binary and have incremental imbalance ratio (IR), given by the number of samples of the majority class divided by the number of samples of the minority class. The datasets used in this study are summarised in Table 13 that shows the number of samples, the number of attributes, the classes distribution and the IR.

The datasets are partitioned using the *5-fold cross-validation* procedure, which means that the datasets are divided in 5 folds (each one with 20% of the samples) and the experiments are performed 5 times, each time with one of the folds as the test set and the remaining four folds as the training set. This partitioning is performed respecting class proportion.

The evaluation metrics are: classification accuracy and diversity. For the classification accuracy, the metric used was the Area Under the ROC Curve (AUC). This metric was chosen because it is one of the most suitable metrics when dealing with imbalanced datasets. For the diversity, the Entropy Measure E was used. This non-paired diversity metric was used because it is not biased by the AUC (already considered in the fitness function), and for simplicity, because it has the ideal range (from 0 to 1), where 1 is the highest diversity possible.

Table 13 – Datasets characteristics

Label	Name	Patterns	Features	% (min., maj.)	IR
1	Glass1	214	9	(35.51, 64.49)	1.82
2	Pima	768	8	(34.84, 66.16)	1.90
3	Iris0	150	4	(33.33, 66.67)	2.00
4	Yeast1	1484	8	(28.91, 71.09)	2.46
5	Vehicle2	846	18	(28.37, 71.63)	2.52
6	Vehicle3	846	18	(28.37, 71.63)	2.52
7	Ecoli1	336	7	(22.92, 77.08)	3.36
8	Ecoli2	336	7	(15.48, 84.52)	5.46
9	Glass6	214	9	(13.55, 86.45)	6.38
10	Yeast3	1484	8	(10.98, 89.02)	8.11
11	Ecoli3	336	7	(10.88, 89.12)	8.19
12	Vowel0	998	13	(9.10, 90.9)	9.98
13	Glass4	214	9	(6.07, 93.93)	15.47
14	Ecoli4	336	7	(5.95, 94.05)	15.8
15	Page-blocks13vs4	472	10	(5.93, 94.07)	15.85

In order to evaluate the results, we used the statistical paired test *One Sided Wilcoxon Signed Rank Test* (WILCOXON, 1945), comparing ICS-Bagging and SICS-Bagging with the other techniques in this experiment. The level of significance used was $\alpha = 0.05$.

The techniques and parameters used in this experiments are presented in Table 14.

Table 14 – Algorithms and parameters

Algorithm	Parameters
ICS-Bagging	$N = 40$, *repetition* = *True*
	$K = 5$, $\alpha = 0.75$
SICS-Bagging	$N = 40$, *repetition* = *True*
	$K = 5$, $\alpha = 0.75$, $K_{smote} = 5$
SMOTEBagging	$N = 40$, *repetition* = *True*
	$K_{smote} = 5$
Bagging	$N = 40$, *repetition* = *True*
RandomSubspace	$N = 40$, $features_{max} = 0.3$

The base classifier used in the experiments was the Decision Tree Classifier with maximum depth of 9, and minimum number of samples required to be at a leaf node of 1. The combination scheme used is a simple majority vote.

B.4.2 Results

Tables 15 and 16 present the average and standard deviation for, respectively, the AUC and Entropy Measure E. The best result in each dataset is highlighted in bold. The last lines present the results of the Wilcoxon Test, the approximated *p-value* and the result when compared with ICS-Bagging and SICS-Bagging. The symbol "+" when a proposed technique outperformed, the symbol "−" when it was outperformed, and "=" when nothing can be concluded with statistical support (*NA* means *not applicable*).

Table 15 – Average, standard deviation AUC, and Wilcoxon Signed Rank Test

Dataset	ICS-Bagging-40		SICS-Bagging-40		SMOTEBagging-40		Bagging-40		RandomSubspace-40	
glass1	**0.7923**	**0.0500**	0.7829	0.0193	0.7816	0.0770	0.7521	0.0529	0.7576	0.0816
pima	**0.7390**	**0.0194**	0.7377	0.0189	0.6988	0.0307	0.7215	0.0227	0.6328	0.0295
iris0	**1.0000**	**0.0000**	**1.0000**	**0.0000**	**1.0000**	**0.0000**	**1.0000**	**0.0000**	**1.0000**	**0.0000**
yeast1	0.7309	0.0343	0.7204	0.0237	**0.7314**	**0.0282**	0.6810	0.0187	0.5325	0.0148
vehicle2	0.9592	0.0168	0.9630	0.0159	**0.9669**	**0.0127**	0.9594	0.0225	0.9633	0.0129
vehicle3	0.7368	0.0188	**0.7566**	**0.0162**	0.7300	0.0083	0.6600	0.0226	0.6327	0.0278
ecoli1	0.8722	0.0497	**0.8795**	**0.0444**	0.8642	0.0414	0.8480	0.0454	0.7672	0.0887
ecoli2	0.8855	0.0495	**0.8861**	**0.0439**	0.8670	0.0529	0.8714	0.0464	0.7283	0.0357
glass6	0.8917	0.0190	**0.8965**	**0.0643**	0.8886	0.0361	0.8865	0.0624	0.8640	0.0836
yeast3	**0.9089**	**0.0273**	0.9045	0.0248	0.9055	0.0095	0.8434	0.0337	0.5030	0.0061
ecoli3	0.8047	0.0967	**0.8223**	**0.0590**	0.7525	0.0827	0.7724	0.0677	0.6279	0.0708
vowel0	0.9483	0.0573	**0.9594**	**0.0546**	0.9455	0.0547	0.9589	0.0540	0.9278	0.0572
glass4	0.8750	0.1877	**0.8825**	**0.1915**	0.8708	0.0982	0.7467	0.1654	0.6475	0.1362
ecoli4	0.8405	0.1143	0.8655	0.0721	0.8155	0.1194	**0.8671**	**0.1326**	0.7000	0.1000
page-blocks-1-3_vs_4	**0.9978**	**0.0045**	**0.9978**	**0.0045**	**0.9978**	**0.0045**	0.9766	0.0414	0.8733	0.0712
Average	0.8655	0.0497	**0.8703**	**0.0435**	0.8544	0.0438	0.8363	0.0526	0.7439	0.0544
P-value (ICS-Bagging)	NA		0.9421		0.0044		0.0055		0.0006	
Result (ICS-Bagging)	NA		=		+		+		+	
P-value (SICS-Bagging)	0.0579		NA		0.0054		0.0008		0.0006	
result (SICS-Bagging)	=		NA		+		+		+	

Table 16 – Average, standard deviation Entropy Measure E, and Wilcoxon Signed Rank Test

Dataset	ICS-Bagging-40		SICS-Bagging-40		SMOTEBagging-40		Bagging-40		RandomSubspace-40	
glass1	0.4445	0.0560	0.4314	0.0570	0.4177	0.0464	0.3786	0.0531	**0.4604**	**0.0347**
pima	0.4284	0.0133	0.4130	0.0197	0.4303	0.0174	0.4131	0.0196	**0.4966**	**0.0216**
iris0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**0.0073**	**0.0060**
yeast1	**0.4342**	**0.0230**	0.3926	0.0256	0.3779	0.0200	0.3410	0.0029	0.2727	0.0222
vehicle2	0.1006	0.0148	0.0809	0.0039	0.0849	0.0059	0.0886	0.0126	**0.1699**	**0.0204**
vehicle3	**0.3900**	**0.0219**	0.3774	0.0248	0.3757	0.0239	0.3661	0.0177	0.3349	0.0382
ecoli1	0.1542	0.0361	0.1555	0.0479	0.1474	0.0340	0.1456	0.0337	**0.2965**	**0.0475**
ecoli2	0.1556	0.0440	0.1377	0.0347	0.1174	0.0203	0.1216	0.0286	**0.2174**	**0.0281**
glass6	0.1108	0.0550	0.0894	0.0235	0.0708	0.0305	0.0613	0.0148	**0.1265**	**0.0313**
yeast3	0.0767	0.0076	0.0717	0.0110	0.0619	0.0086	0.0825	0.0086	**0.1158**	**0.0119**
ecoli3	0.1187	0.0364	0.1200	0.0339	0.1089	0.0361	0.1246	0.0290	**0.1710**	**0.0476**
vowel0	0.0346	0.0176	0.0309	0.0148	0.0237	0.0084	0.0299	0.0069	**0.0994**	**0.0093**
glass4	0.0684	0.0299	0.0623	0.0288	0.0557	0.0289	0.0944	0.0312	**0.1026**	**0.0363**
ecoli4	0.0460	0.0228	0.0476	0.0049	0.0253	0.0125	0.0333	0.0151	**0.0984**	**0.0284**
page-blocks-1-3_vs_4	0.0052	0.0065	0.0020	0.0035	0.0012	0.0013	0.0269	0.0090	**0.0524**	**0.0152**
Average	0.1712	0.0257	0.1608	0.0223	0.1533	0.0196	0.1538	0.0189	**0.2015**	**0.0266**
P-value (ICS-Bagging)	NA		0.0018		0.0006		0.0320		0.9795	
Result (ICS-Bagging)	NA		+		+		+		−	
P-value (SICS-Bagging)	0.9982		NA		0.0078		0.1501		0.9900	
Result (SICS-Bagging)	−		NA		+		=		−	

B.4.2.1 *Classification Accuracy*

Table 15 shows that ICS-Bagging and SICS-Bagging outperformed all techniques in classification accuracy. ICS-Bagging outperformed Bagging, Random Subspace, and one of the top ensemble techniques for imbalanced datasets, SMOTEBagging. SICS-Bagging also outperformed all the other techniques, but was statistically equivalent with ICS-Bagging.

SICS-Bagging was designed as an improvement of ICS-Bagging, with SMOTE being applied before each iteration in order to increase the likelihood of creating balanced classifiers (following the SMOTEBagging approach). The objective was achieved, and SICS-Bagging was the best technique in 9 out of 15 datasets, and achieved the highest average AUC (0.8703), followed by ICS-Bagging (0.8655).

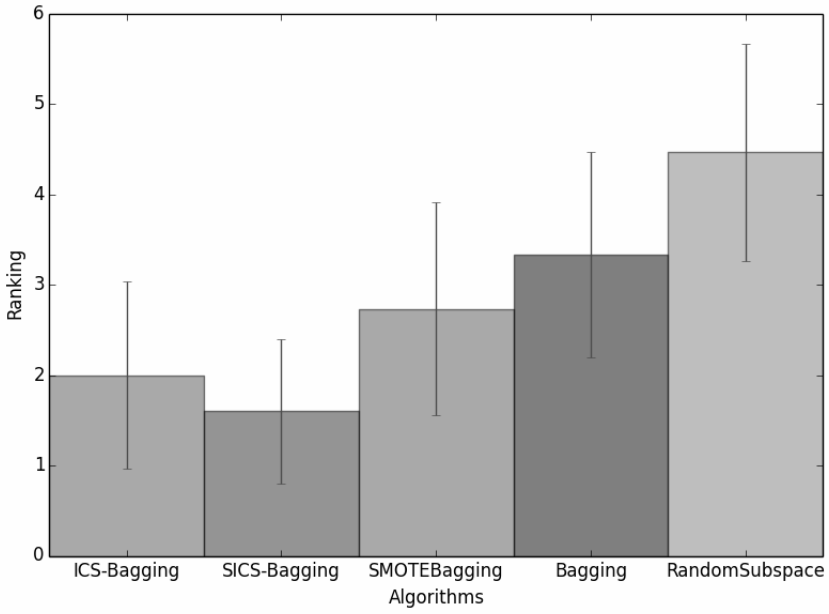


Figure 35 – Average AUC ranking of the ensemble techniques

Figure 35 and Table 17 show the average and standard deviation AUC ranking of the ensemble techniques in the experiments. The two techniques proposed in this paper achieved the two highest performances. SICS-Bagging achieved the first place, and ICS-Bagging the second place.

Table 15 and Figure 35 show that ICS-Bagging and SICS-Bagging had an excelent performance in classification accuracy with imbalanced datasets, outperforming SMOTE-Bagging, which was considered one of the best ensemble techniques for imbalanced domains in (GALAR et al., 2012).

B.4.2.2 *Diversity*

Table 16 shows that ICS-Bagging outperformed all techniques in diversity, except for the Random Subspace. Random Subspace achieved a high diversity because it only

selects a subset of the features for each classifier, and it was confirmed to be a more effective diversity generator than selecting a subset of samples for each classifier. This can be confirmed when Random Subspace is compared with Bagging (with no preprocessing).

Statistically, SICS-Bagging outperformed only SMOTEBagging, but, on average, it achieved the third highest diversity, losing only to Random Subspace and ICS-Bagging. SICS-Bagging was outperformed by ICS-Bagging in diversity because the SMOTE preprocessing improves the AUC of all classifiers, but does not necessarily generate diverse classifiers. This can be confirmed because SMOTEBagging did not improve diversity over Bagging (on average).

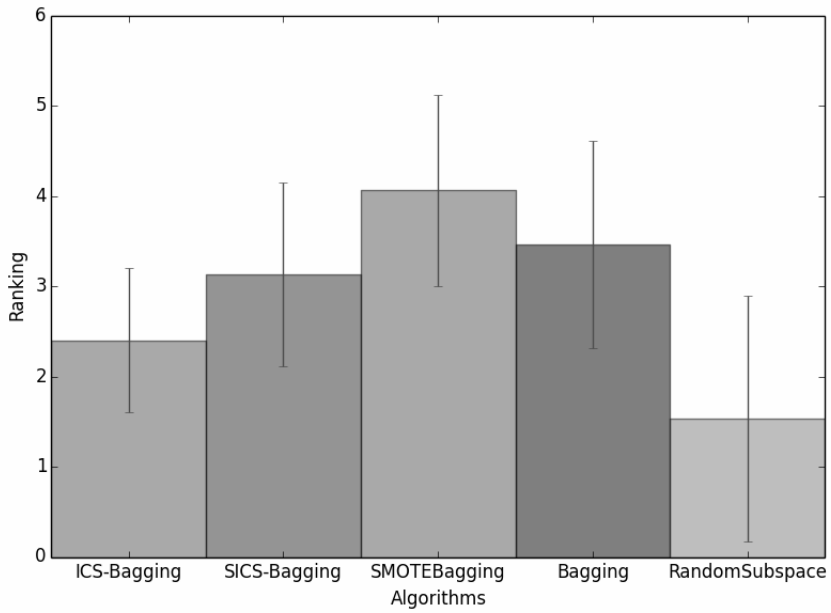


Figure 36 – Average Entropy Measure E ranking of the ensemble techniques

Figure 36 and Table 18 show the average and standard deviation ranking of the ensemble techniques in the experiments. ICS-Bagging achieved the second highest diversity, and SICS-Bagging the third highest. The highest diversity ranking was achieved by Random Subspace, which achieved a low classification accuracy. This indicates that a high Entropy E value does not necessarily result in a high classification accuracy.

The purpose of a diversity measure is to predict the accuracy of the ensemble on new data. These results show that the Entropy E was not strongly correlated to classification accuracy when using simple majority vote, this scenario could be different if another combination rule was used, or if we used dynamic ensemble selection.

B.4.2.3 *Diversity vs. Classification*

Figure 37 presents the dispersion graph (Diversity vs. AUC) of the ensemble techniques used in this experiment. This figure shows that, on average, SICS-Bagging

Table 17 – Ranking of AUC classification accuracy

Algorithm	Average	Std	Ranking
ICS-Bagging	2.00	1.03	2
SICS-Bagging	1.60	0.80	1
SMOTEBagging	2.73	1.18	3
Bagging	3.33	1.13	4
Random Subspace	4.46	1.20	5

Table 18 – Ranking of Entropy Measure E diversity

Algorithm	Average	Std	Ranking
ICS-Bagging	2.40	0.80	2
SICS-Bagging	3.13	1.02	3
SMOTEBagging	4.07	1.06	5
Bagging	3.47	1.15	4
Random Subspace	1.53	1.36	1

achieved the highest classification accuracy and the third highest diversity. ICS-Bagging had the second highest classification accuracy and the second highest diversity.

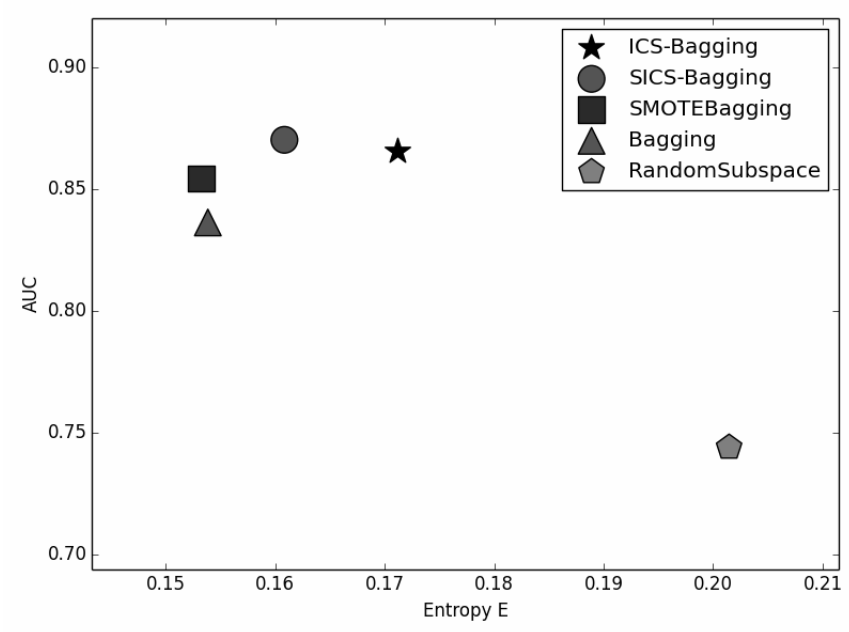


Figure 37 – Dispersion (Diversity vs. AUC) of the ensemble algorithms used in the experiment.

B.5 Conclusion

This paper proposed a new method of generating ensembles called Iterative Classifier Selection Bagging (ICS-Bagging), and its extension for imbalanced datasets SMOTE Iterative Classifier Selection Bagging (SICS-Bagging). An experimental study concluded

that both ICS-Bagging and SICS-Bagging obtain state-of-the-art results in the datasets tested (according with (GALAR et al., 2012)). Future works include: (1) comparing the proposed techniques with other methods; (2) testing other preprocessing techniques and (3) using other diversity metrics.

APPENDIX C – EVOLUTIONARY ADAPTIVE SELF-GENERATING PROTOTYPE FOR IMBALANCED DATASETS

Abstract

The nearest neighbor (NN) is one of the most well known classifiers in pattern recognition. Despite the high classification accuracy, the NN has several drawbacks: high storage requirements, bad time of response, and high noise sensitivity. Prototype Generation (PG) is one of the most well-known solutions to tackle these shortcomings. In supervised classification, many real world datasets do not have an equitable distribution among the different classes, these are called imbalanced datasets. Many PG techniques that have a high classification accuracy in regular datasets, have a poor performance when dealing with imbalanced datasets. The *Self-Generating Prototypes* (SGP) is one of these techniques. The *Adaptive Self-Generating Prototypes* was proposed to tackle the SGP problem with imbalanced datasets, but, in doing so, the reduction rate is compromised. This paper proposes the *Evolutionary Adaptive Self-Generating Prototypes* (EASGP), a SGP based technique with iterative merging and evolutionary pruning to help find the optimal solution. An experimental analysis is performed with datasets of different levels of imbalance ratio and statistical tests are used to evaluate the proposed technique. The results obtained show that EASGP outperforms previous SGP based algorithms in classification accuracy and reduction.

C.1 Introduction

The nearest neighbor (NN) rule (COVER; HART, 1967) is one of the most well known supervised learning techniques. The general idea is to classify a new instance as being of the same class as its nearest instance from the training set. The K-nearest neighbor rule (KNN) (PATRICK; FISCHER, 1969) is a generalization of the NN rule that considers the label of the K nearest instances of an instance to make it's classification. This technique is very simple, yet is one of the most interesting and useful algorithms in Pattern Recognition (SHAKHNAROVICH; DARRELL; INDYK, 2005b).

Despite the high classification accuracy, the KNN has several drawbacks (KONONENKO; KUKAR, 2007) (GARCIA et al., 2012) (TRIGUERO et al., 2012), the three most relevant are:

1. **high storage requirements**: it needs to store all training set, because the decision rule is defined by all training instances.
2. **bad response time**: for every new classification, the KNN needs to visit all instances from the training set ($\mathcal{O}(n)$ complexity, where n is the size of the traning set).
3. **low tolerance to noise**: it considers all data from the training set to be relevant to the classification task. Because of that, noisy data might compromise the classification accuracy.

In the literature, many approaches have been proposed to solve these issues (FERNÁNDEZ; ISASI, 2008). One of these approaches is the use of instance reduction techniques (WILSON; MARTINEZ, 2000). Techniques of instance reduction aim to select only relevant instances from the training set, creating a smaller training set without compromising the classification accuracy (WILSON; MARTINEZ, 2000). In doing so, the storage requirements and time of response are reduced, because less instances are saved and visited in each classification, and also removes noisy data, improving classification and solving the noise sensitivity problem.

Prototype selection (PS) is a process of instance reduction that removes instances that are redundant or irrelevant to the classification, and selects the representative ones. Prototype generation (PG) is the process of instance reduction that generates artificial instances in order to achieve a higher generalization and create a more suitable training set. Because of the limitations of the search space of PS techniques, PG techniques have the potential of achieve a higher reduction rate than PS techniques.

Imbalanced datasets have many instances of one class, the majority class, and only a few of the other, the minority class. Learning from such datasets is a difficult task, being considered an important problem in data mining and pattern recognition (YANG; WU,

2006) (LÓPEZ et al., 2013). Despite the high performance in improving classification, instance reduction techniques do not cope with imbalanced datasets, for they cannot discriminate noisy instances from the minority class.

The *Self-Generating Prototypes* (SGP) (FAYED; HASHEM; ATIYA, 2007) is a centroid-based prototype generation technique, it was even being combined with other techniques to improve classification (PEREIRA; CAVALCANTI, 2008), but does not work well with imbalanced datasets. The same thing happens with the *Self-Generating Prototypes 2* (SGP2).

Experiments have shown that, in some datasets, SGP and SGP2 consider all minority class instances as noise or outliers that need to be removed in order to improve generalization (OLIVEIRA et al., 2012). In order to solve this issue, in (OLIVEIRA et al., 2012), the authors proposed the *Adaptive Self-Generating Prototypes* (ASGP) a SGP based technique that is adaptive to different levels of imbalance ratio. This technique achieved interesting results, and the empirical experiments have shown that ASGP is a better solution for imbalanced datasets (OLIVEIRA et al., 2012).

ASGP outperforms SGP2 in classification accuracy, but it is outperformed in reduction rate because ASGP generates more minority class prototypes than needed (increasing false positives in classification).

To solve the issues of SGP, SGP2 and ASGP, in this paper, we propose the *Evolutionary Adaptive Self-Generating Prototypes* (EASGP), a PG technique composed by a main loop and two new steps: incremental merging and evolutionary pruning. The incremental merging expands the search space inserting new generalized samples that might better represent the distribution of the classes. The evolutionary pruning uses a memetic algorithm (MA) based on the *Steady-State Memetic Algorithm* (SSMA) (GARCÍA; CANO; HERRERA, 2008) to explore the search space and find the optimal solution. The evolutionary pruning uses the original training set as a validation set, instead of using only the new generated samples.

The experiments showed that EASGP outperforms SGP, SGP2 and ASGP in classification accuracy and reduction rate for datasets with different levels of imbalance. This result was confirmed with the One-Sided Wilcoxon Signed Rank Test (WILCOXON, 1945).

This paper is organized as follows: Section II gives a brief review of PG techniques and presents the SGP, SGP2 and ASGP techniques. Section III presents the EASGP technique. Section IV presents the experiments and results. Finally, Section V concludes the paper.

C.2 Background

This section presents the main concepts of prototype selection (PS) and prototype generation (PG), including the Self-Generating Prototypes (SGP), Self-Generating Prototypes 2 (SGP2) and Adaptive Self-Generating Prototypes (ASGP).

C.2.1 Prototype Selection and Generation

PS methods are instance selection methods that, in order to improve the NN rule, attempt to find the smallest subset of the training set that enable the KNN to correctly classify a test sample (LIU; MOTODA, 2002).

The PS problem can be defined as follows: Let TR be the training set, and $S \subseteq TR$ be the subset of instances selected by a PS technique, where S has less noise or redundant instances. The classification of a test sample x_i is made using the KNN rule over S instead of TR.

Since PS problems can be reduced to a combinatorial problem, it is possible to use Evolutionary Algorithms (EAs) to solve them. In fact, a high number of the PS algorithms recently proposed are based on EAs. Those methods were called evolutionary prototype selection (EPS) methods (CANO; HERRERA; LOZANO, 2003). The Steady-State Memetic Algorithm (SSMA) (GARCÍA; CANO; HERRERA, 2008) is a Memetic Algorithm (MA) applied to PS. Studies have shown that this is one of the most successful PS techniques (GARCIA et al., 2012). The use of stratified PS based on SSMA is an alternative to solve the scalability issue, the problem of increasing the running time when the number of instances increase (DERRAC; GARCÍA; HERRERA, 2010).

There are over 50 PS methods proposed in the literature. A complete study of PS, including taxonomy, can be found in (GARCIA et al., 2012).

PG methods are instance reduction methods that attempt to find the smallest set of artificial generated instances that improves the accuracy of the NN rule based on the training set.

A PG problem can be defined as follows: Let TR be the training set, and TG a set of prototypes generated or selected by a PG method based on TR, where *Size(TG)* $<$ *Size(TR)*. The classification of a test sample x_i is made using the KNN rule over TG, instead of TR.

Most PG techniques that uses EAs are based on positioning adjustment, that means that they move the prototypes around the m-dimensional space, adjusting them until it finds an optimal solution (TRIGUERO et al., 2012). Examples of interesting positioning adjustment PG techniques are the *Evolutionary Nearest Prototype Classifier* (ENPC) (FERNÁNDEZ; ISASI, 2004) and the *Particle Swarm Optimization* (PSO) (NANNI;

LUMINI, 2009). Other techniques such as *Prototype Selection Clonal Selection Algorithm* (PSCSA) (GARAIN, 2008), and *Differential Evolution* (DE) (TRIGUERO; GARCÍA; HERRERA, 2011) have also achieved interesting results.

For imbalanced datasets, the use of selection of evolutionary selection of generalized examples (DERRAC et al., 2012) has achieved interesting results.

There are over 25 PG methods proposed in the literature. A complete study of PG, including taxonomy, can be found in (TRIGUERO et al., 2012).

C.2.2 Self-Generating Prototypes Based Algorithms

The *Self-Generating Prototypes* (SGP) (FAYED; HASHEM; ATIYA, 2007) is an interesting PG technique (PEREIRA; CAVALCANTI, 2008). The SGP method generates prototypes using a combination of centroid based and space splitting mechanisms. The SGP creates groups of instances generates a single prototype (the representant) for each group.

In the beginning of the process, for each class, the SGP generates one group containing all instances of that class. Then, the following steps are performed the until the solution converge:

1. If, for all instances of a group, the closest prototype is the representant of the group itself, then no modification is performed.

2. If, for all instances of a group, the closest prototype is from a different class, then the group is divided in two new subgroups. The separation is made by a hyperplane that passes through the representant of the original group and is perpendicular to the first principal component of the instances in the original group.

3. If, for some instances of a group, the closest prototype is a representant of a different group of the same class, these instances are moved from the original group to the group of that closest prototype.

4. If, for some instances in a group, the closest prototype is the representant of a group with different class, the misclassified instances are removed from the original group and form a new group.

After any of these procedures, each group representant is updated.

In order to improve the generation capability, the SGP implements a trade-off between training error and the model complexity using two parameters: R_{min} and R_{mis}. If the number of instances in a group divided by the number of instances of the largest group is less than a threshold R_{min}, the group is discarded. Also, along the SGP algorithm, if

the number of misclassified instances in a group divided by the number of instances in that group is less than a threshold R_{mis}, no modification is performed. This step is called *generalization step*.

The SGP2 introduces two steps in order to reduce even more the number of prototypes: A merging step and a pruning step. In the merging step, two groups A and B are merged if both A and B are from the same class and the second closest prototype to all instances in A is the representant of B, and the second closest prototype to all instances in B is the representant of A. The pruning step removes redundant prototypes using the following rule: if all instances of a group is correctly classified by their second closest prototype, the group is discarded.

In (OLIVEIRA et al., 2012), the authors analyzed the SGP behavior when trained with imbalanced datasets. Sometimes, the SGP returned no prototypes at all of the minority class. The authors concluded that one of the major issues happens in the generalization step, so they proposed a different approach in the use of the generalization factor R_{min}.

SGP eliminates the groups that have less than R_{min} times L instances, L being the size of the largest group in the dataset. The *Adaptive Self-Generating Prototypes* (ASGP) suggests that this elimination is not fair with the minority class groups. To solve this issue, the same R_{min} is used for all groups, but L is the size of the largest group of the same class, instead of the size of the largest group of all classes. This step is detailed in Algorithm 15.

Algorithm 15 Generalization Step

Require: GP: a set of groups of instances
Require: H: a hashtable
Require: CS: a list of classes of the groups
1: **for all** Class $C \in CS$ **do**
2: $L \leftarrow -1$
3: **for all** Group G in GS **do**
4: **if** $SizeOf(G) > L$ **then**
5: $L \leftarrow SizeOf(G)$
6: **end if**
7: **end for**
8: $H(C) \leftarrow L$
9: **end for**
10: **for all** Group G in GS **do**
11: $C \leftarrow ClassOf(G)$
12: $L \leftarrow H(C)$
13: **if** $\frac{SizeOf(G)}{L} \leq R_{min}$ **then**
14: Remove G from GS
15: **end if**
16: **end for**
17: **return** GS

ASGP also proposes that the merge and pruning steps should be performed as usual, but after both procedures, all prototypes of the minority class should be included again, in case they were lost in the generalization procedure.

The ASGP method achieved interesting results with imbalanced datasets, but there are other drawbacks to be tackled. When performing the merging step, the order in which the merging occurs affects the final solution, making possible the algorithm to find sub-optimal solutions. The same thing happens with the pruning step. Another issue with the pruning step is that it removes a group only if all instances in that group are well classified without the group representant. A higher generalization might be achieved if there was a threshold that evaluated if a removal is an advantage or not.

When inserting all prototypes of the minority class, after the pruning step, the ASGP method might also insert not needed prototypes, and even, prototypes that does not fit the new data, generating an overlap between the prototypes of the minority and majority classes.

The next section presents the proposed technique that handles all the mentioned issues.

C.3 Evolutionary Adaptive Self-Generating Prototypes

This section presents the Evolutionary Adaptive Self-Generating Prototypes (EASGP), a prototype generation (PG) that uses an iterative merge and evolutionary pruning.

C.3.1 Motivation

The Self-Generating Prototypes (SGP) has a poor performance when trained with imbalanced datasets. The Adaptive Self-Generating Prototypes (ASGP) is an improved SGP that handles imbalanced datasets. Despite the fact that ASGP achieved better results than SGP2, ASGP has a lower reduction rate than SGP2. This behavior is acceptable, because of high cost of misclassifying the minority class (ELKAN, 2001), but it is not desired.

The following three flaws were found in the SGP2 and ASGP algorithms:

1. In the merging step, the order in which the groups are merged affects the resulting prototypes. If two groups are merged, another important merge might not take place, and a sub-optimal solution might be returned.

2. In the pruning step, the already reduced search space is not fully explored. Also, when the order in which the groups are visited is changed, the resulting prototypes also change. Because of that, a sub-optimal solution might be returned.

3. After the merging and pruning steps, when introducing back the prototypes of the minority class, the ASGP algorithm does not consider that the new set of majority class prototypes were generated considering another group of the minority class prototypes. This might cause overlapping between prototypes of different classes.

In order to solve these issues, this paper proposes the EASGP method, an SGP based PG technique that implements an incremental merging and evolutionary pruning steps.

C.3.2 Architecture

Figure 38 shows the architecture of EASGP. First, ASGP is used to generate the initial prototypes. These prototypes are used by the iterative merging algorithm, generating new prototypes and expanding the search space. Finally, the evolutionary pruning is applied to find the optimal subset of prototypes, and the best subset is returned. The evolutionary pruning uses the original training set as a validation set to find the best solution.

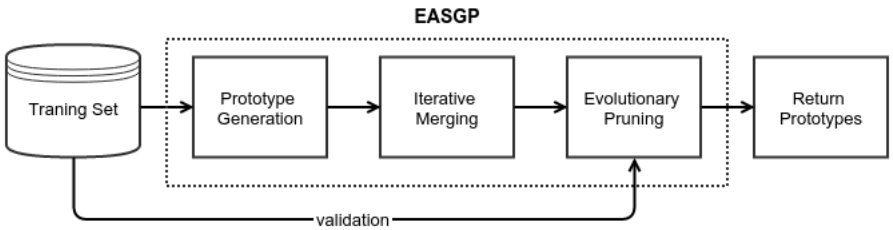


Figure 38 – Architecture of EASGP.

The EASGP steps are explained in the next subsections.

C.3.3 EASGP Initial Prototype Generation

Following the same approach of the SGP algorithm, In the beginning of the process, a group containing all the instances within each class is created for all class labels, and the mean of each group is elected representant. The ASGP main loop is executed once the initial groups are formed until no changes in the groups occurs.

The generalization step follows the ASGP approach, removing a group only if it is considerably smaller than the larger group of the same class.

The major inovations of EASGP are the iterative merging and evolutionary pruning, presented in the next subsections.

C.3.4 Iterative Merging

Differently than the merging step of ASGP, the iterative merging uses only the generated prototypes to perform the merge between two groups, instead of using all instances within the groups. This approach has a high performance effect on the algorithm, since the number of representants is usually significantly smaller than the number of instances in the training set.

In order to understand the iterative merging, the *merging-links* concept must be defined. A pair of prototypes A and B is called a *merging-link* if they meet the 3 following conditions:

- A and B belong to the same class.
- The closest prototype of A is B.
- The closest prototype of B is A.

The iterative merging also uses two lists: *search list* and *final list*. The *search list* is the list of prototypes where the procedure searches for *merging-links*. The *final list* is the list of prototypes to be returned by the procedure.

First, the iterative merging insert all prototypes from the EASGP initial prototypes procedure in the *search list* and then finds all *merging-links* prototypes in that list. If at least one *merging-link* is found, than this is not the last iteration. Each link is merged, and a new prototype is generated (the mean of the each link). The *merging-links* are now removed from *search list* and saved into the *final list*. The process is repeated until the last generation (no *merging-link* is found), when all remaining prototypes are included in the *final list*. Finally, all prototypes in *final list* are returned.

The iterative merging is detailed in Algorithm 16.

Algorithm 16 Iterative Merging

Require: PS: a set of prototypes
Require: *search list*: a list to search for links
Require: *final list*: a list to save the prototypes
save all PS in *search list*
merging-links ← all merging links in *search list*
while *merging-links* is not empty **do**
  for all *link* ∈ *merging-links* **do**
    $P \leftarrow mean(link)$
    insert P in the *search list*
    remove the *link* prototypes from *search list*
    insert the *link* prototypes in *final list*
  end for
  merging-links ← all merging links in *search list*
end while
insert all prototypes in *search list* in *final list*
return the prototypes in *final list*

With this approach, the EASGP iterative merging expands the region of search that was reduced by the initial prototypes procedure. The result of the iterative merging is not affected by the order in which the prototypes are merged (differently than the ASGP merge step).

C.3.5 Evolutionary Pruning

The pruning step aims to remove redundant prototypes without compromising the classification accuracy. The problem of pruning can be considered a problem of prototype selection (PS), which is to find the best subset of instances that better represent a training set. As mentioned in the previous section, the problem of PS is a combinatorial problem, and the current best approaches to solve this problem is to use evolutionary prototype selection (EPS), especially memetic algorithms (MA).

The evolutionary pruning uses the Steady-State Memetic Algorithm (SSMA) (GARCÍA; CANO; HERRERA, 2008) concept to find the best subset of generated prototypes, the difference is the use of the whole traning set as a reference set. First, a population of solutions is created using the chromosome representation. For each interection, two parent chromosomes are selected and the genetic operators are applied to generate an offspring. A local search is used to optimize each solution with a given probability. The evolutionary

pruning uses the fitness function detailed the Equation C.1.

$$Fitness(S) = \alpha \times \text{AUC}_{rate} + (1 - \alpha) \times \text{reduction}_{rate} \tag{C.1}$$

The value of the parameter α is within the interval $[0.51, 0.99]$. In the evolutionary pruning, the classification rate used in the fitness when handling imbalanced dataset is the Area Under the ROC Curve (AUC), avoiding over generalization and the possibility of removing of all prototypes of the minority class.

Also, the fitness is estimated using the original training set as a validation set, not only the prototypes from the merging step. Using this approach, the evolutionary pruning finds better solutions and avoids the cost of misclassification of already removed samples in the previous steps of the algorithm.

The Algorithm 17 presents the evolutionary pruning algorithm.

Algorithm 17 Evolutionary Pruning

```
1: Initialize Population.
2: while not termination-condition do
3:    parents ← Parent Selection (binary tournament)
4:    Off_1, Off_2 ← Crossover(parents)
5:    Mutation(Off_1, Off_2)
6:    for all Off_i do
7:       if local search decision then
8:          local-search(Off_i)
9:       end if
10:   end for
11:   Standard Replacement for Off_1 and Off_2
12: end while
13: return The best chromosome
```

The evolutionary pruning, which is based on the SSMA, steps are detailed as follows:

- **Population Initialization**: Each chromosome represents a subset of prototypes, a gene is '1' when the prototype is in the subset and '0' when it is not. All chromosomes reference the same set of prototypes returned by the iterative merging. In the population inicialization, the chromosomes are inicialized randomly, and each chromosome is evaluated using the fitness function. Differently than other approaches, the evaluation is performed using all instances in the training set as a validation set, and not only the prototypes referenced by the chromosomes.

- **Parent Selection**: In order to select two parents, a binary tournament selection is employed. For each parent, two random candidates are selected from the population of chromosomes, and the best one (the one with the higher fitness) is elected parent.

- **Crossover**: The parents are combined to generate the offspring, two new individuals with half of the genes from each parent.

- **Mutation**: The mutation changes each gene of the offspring with a probability $P = 1/N$, where N is the size of the chromosome.

- **Offspring Evaluation**: The offspring is evaluated using the fitness function (Equation C.1) and the validation set (the original training set).

- **Local Search Decision**: This step decides if a local search is applied to an individual in the offspring. The local search is performed with a probability P_{ls} which is detailed in the Equation C.2.

$$P_{ls}(S) = \begin{cases} 1 & \text{if fitness}(S) > \text{fitness}(S_{worse}) \\ 0.0625 & \text{otherwise} \end{cases} \tag{C.2}$$

  If the offspring is better than the worst solution of the population of chromosomes (has a higher fitness than the solution with lower fitness in the population), the local search is performed, otherwise, the local search is performed with a probability $P_{ls} = 0.0625$. This value was found empirically in (GARCÍA; CANO; HERRERA, 2008).

- **Local Search**: For a given chromosome, it considers neighborhood solutions by removing an instance from the current solution. A change is maintained if it improves the classification accuracy of the current solution, otherwise, the change is reverted. If a removal becomes permanent, the whole neighborhood is considered again. To avoid local optimum, the local search also accepts solutions that decreases the classification accuracy but increase the fitness.

- **Standard Replacement**: If the offspring is better than the worst solution in the population (has a higher fitness than the solution with lower fitness in the population), the offspring replaces the worst solution.

- **Termination Condition**: The algorithm stops when a convergence of the solutions occurs, or when a number of evaluations NE (passed as parameter) is reached. In other evolutionary algorithms, usually $NE = 10000$. Because of the high reduction power of ASGP, $NE = 100$ is enough for EASGP.

The evolutionary pruning procedure works as an optimization algorithm. Other EPS can be used, but the local search of SSMA makes it possible to find optimal solutions without reach the number of evaluations that a brute-force algorithm requires.

Compared to SSMA alone, one advantage of EASGP is that EASGP makes possible a higher generalization and requires fewer evaluations than other evolutionary algorithms, since the chromosomes only represent the previously generated prototypes and not the whole original training set. In classification accuracy, EASGP pruning also works as a fixer for the previous steps, removing any residual noisy data generated by the initial generation (ASGP main procedure) and the iterative merging.

C.4 Experiments

This section presents the methodology used in the experiments, and the results of the Evolutionary Adaptive Self-Generating Prototypes (EASGP).

C.4.1 Methodology

The EASGP method is evaluated using 15 imbalanced datasets from KEEL (ALCALÁ et al., 2010). The datasets are binary (2 classes) and have different levels of imbalance. Table 19 summarises the datasets used in this study, detailing the number of samples, the number of attributes, the class distribution and the imbalance ratio (IR). The datasets are partitioned using the *five fold cross-validation* procedure, respecting the classes proportions.

Table 19 – Datasets characteristics

Label	**Dataset**	**#Attributes**	**#Instances**	**IR**
1	pima	8	768	1.87
2	yeast1	8	1484	2.46
3	vehicle2	18	846	2.88
4	vehicle1	18	846	2.9
5	vehicle3	18	846	2.99
6	vehicle0	18	846	3.25
7	ecoli2	7	336	5.46
8	segment0	19	2308	6.02
9	ecoli3	7	336	8.6
10	yeast05679vs4	8	528	9.35
11	vowel0	13	988	9.98
12	glass016vs2	9	192	10.29
13	glass2	9	214	11.59
14	shuttlec0vsc4	9	1829	13.87
15	yeast1vs7	7	459	14.3

The evaluation metrics are: Area Under the ROC Curve (AUC) and reduction rate. To compare the results, we use the *One Sided Wilcoxon Rank Test* (WILCOXON, 1945), with significance level $\alpha = 0.1$. Table 20 presents the techniques and parameters used in this experiment. The 1NN was the base classifier used for all techniques.

Table 20 – Parameters used in the experiments.

Algorithm	Parameters
SGP	$R_{min} = 0.05, R_{mis} = 0.04$
SGP2	$R_{min} = 0.05, R_{mis} = 0.04$
ASGP	$R_{min} = 0.05, R_{mis} = 0.04$
EASGP	$R_{min} = 0.05, R_{mis} = 0.04, \alpha = 0.7$

C.4.2 Results

Table 21 and Table 22 are grouped in columns by algorithms, and the best result of each dataset is highlighted in bold. The last lines presents the results of the Wicoxon Test, considering $\alpha_{wilcoxon} = 0.1$, the symbol "+" is used when EASGP outperforms the algorithm in that column (NA means *not applicable*).

C.4.2.1 *AUC*

Table 21 shows the average and standard deviation of the classification accuracy (given by the AUC). The results show that EASGP outperformed all previous version of SGP with statistical confidence: SGP (p-*value* $= 0.0004908$), SGP2 (p-*value* $= 0.001755$) and ASGP (p-*value* $= 0.0659$).

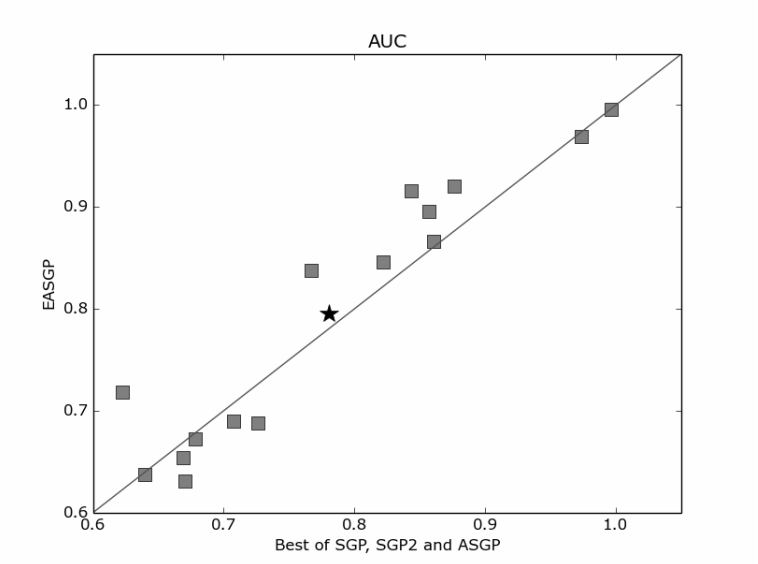


Figure 39 – Best of SGP, SGP2 and ASGP × EASGP AUC rate graph, where the squares are the datasets and the star is the average.

Table 21 – Average, standard deviation and Wilcoxon Signed Rank Test p-value and result of the SGP, SGP2, ASGP and EASGP AUC rate

Label	SGP	SGP2	ASGP	EASGP
1	0.6503(0.0348)	**0.6780(0.0507)**	0.6548(0.0191)	0.6721(0.0107)
2	0.6329(0.0081)	**0.6399(0.0219)**	0.6316(0.0318)	0.6371(0.0265)
3	0.8508(0.0636)	0.8448(0.0705)	0.8606(0.0459)	**0.8662(0.0174)**
4	0.6364(0.0478)	0.6336(0.0472)	**0.6687(0.0377)**	0.6541(0.0339)
5	0.6403(0.0377)	0.6398(0.0177)	**0.7076(0.0277)**	0.6901(0.0161)
6	0.7702(0.1111)	0.7728(0.1085)	0.8570(0.0578)	**0.8952(0.0342)**
7	0.8761(0.0533)	0.8455(0.0212)	0.8443(0.0310)	**0.9203(0.0386)**
8	0.9672(0.0253)	**0.9735(0.0226)**	0.9684(0.0098)	0.9687(0.0114)
9	0.8189(0.0690)	0.8215(0.0504)	0.8169(0.0311)	**0.8461(0.0965)**
10	0.5864(0.1062)	0.5922(0.1135)	0.7667(0.0438)	**0.8375(0.0215)**
11	0.6828(0.2253)	0.6817(0.2238)	0.8431(0.1264)	**0.9154(0.0753)**
12	0.5710(0.0954)	0.6179(0.0664)	**0.6702(0.1180)**	0.6312(0.1436)
13	0.6356(0.1126)	0.6204(0.1083)	**0.7260(0.1256)**	0.6878(0.1879)
14	**0.9960(0.0080)**	**0.9960(0.0080)**	**0.9960(0.0080)**	**0.9960(0.0080)**
15	0.5744(0.0579)	0.5996(0.0808)	0.6224(0.0638)	**0.7185(0.0929)**
Mean	0.7260(0.0704)	0.7305(0.0674)	0.7756(0.0518)	**0.7958(0.0543)**
p-value	0.0004908	0.001755	0.0659	NA
Result	+	+	+	NA

Figure 39 shows EASGP compared with the best of the other techniques. This figure shows that EASGP outperforms the best of SGP, SGP2 and ASGP in classification accuracy, achieving the highest ranking in 8 out of the 15 datasets. Clearly, EASGP achieved the best performance in classification accuracy (AUC).

C.4.2.2 *Reduction*

Table 22 shows the average and standard deviation of the reduction rate. The results show that EASGP outperformed all previous versions of SGP with statistical confidence: SGP (*p-value* = 0.0004908), SGP2 (*p-value* = 0.007827), and ASGP (*p-value* = 0.0004908).

The impressive result is that EASGP outperformed SGP2 in reduction rate. Based on previous studies (OLIVEIRA et al., 2012), we concluded that SGP2 have a high reduction power, but sometimes it removes all samples of the minority class, which allows the algorithm to leave only a prototype of the majority class. Despite of that, EASGP was able to outperform SGP2 in reduction, without compromising classification accuracy.

Figure 40 shows EASGP compared with the best of the other techniques. This figure shows that EASGP outperforms the best of SGP, SGP2 and ASGP in reduction rate, losing in only 3 datasets. Even in the few datasets where EASGP was outperformed,

we can see that the points are very close to the line, which means the difference is very small.

We can conclude with confidence that EASGP achieved the best peformance in reduction rate.

Table 22 – Average, standard deviation and Wilcoxon Signed Rank Test p-value and result of the SGP, SGP2, ASGP and EASGP reduction rate

Dataset	SGP	SGP2	ASGP	EASGP
1	0.8645(0.1893)	0.9127(0.1478)	0.7204(0.0815)	**0.9505(0.0305)**
2	0.7867(0.1586)	0.8760(0.1192)	0.7010(0.0604)	**0.9212(0.0308)**
3	0.9232(0.0432)	0.9548(0.0275)	0.9134(0.0198)	**0.9749(0.0116)**
4	0.7843(0.1549)	0.8502(0.1362)	0.7355(0.0623)	**0.9250(0.0390)**
5	0.7884(0.1055)	0.8623(0.0884)	0.7376(0.0440)	**0.9241(0.0295)**
6	0.9554(0.0451)	**0.9734(0.0254)**	0.9081(0.0253)	0.9722(0.0258)
7	0.8438(0.0514)	0.9301(0.0424)	0.8765(0.0224)	**0.9769(0.0059)**
8	0.9828(0.0033)	0.9916(0.0017)	0.9880(0.0020)	**0.9927(0.0045)**
9	0.9092(0.0409)	**0.9680(0.0130)**	0.8936(0.0178)	0.9524(0.0242)
10	0.9115(0.0952)	0.9597(0.0569)	0.8911(0.0397)	**0.9654(0.0120)**
11	0.9818(0.0195)	0.9899(0.0109)	0.9790(0.0046)	**0.9901(0.0073)**
12	0.7538(0.0646)	0.8632(0.0565)	0.8151(0.0187)	**0.8867(0.0203)**
13	0.7745(0.0835)	0.8890(0.0646)	0.8446(0.0192)	**0.9054(0.0288)**
14	**0.9986(0.0000)**	**0.9986(0.0000)**	**0.9986(0.0000)**	**0.9986(0.0000)**
15	0.7898(0.1006)	**0.8791(0.0592)**	0.8660(0.0399)	0.8726(0.0556)
Mean	0.8699(0.0770)	0.9266(0.0566)	0.8579(0.0305)	**0.9472(0.0217)**
p-value	0.0004908	0.007827	0.0004908	NA
Result	+	+	+	NA

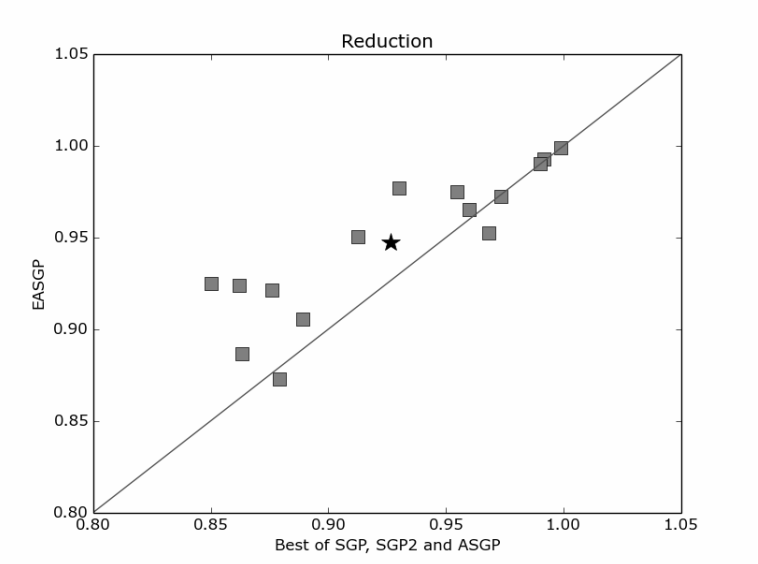


Figure 40 – Best of SGP, SGP2 and ASGP × EASGP reduction rate graph, the squares are the datasets and the star is the average.

C.4.2.3 *Reduction vs. Classification*

Figure 41 shows the dispersion graph (Reduction vs. AUC) of SGP, SGP2, ASGP and EASGP. This figure shows that EASGP achieved the best classification accuracy and reduction rate.

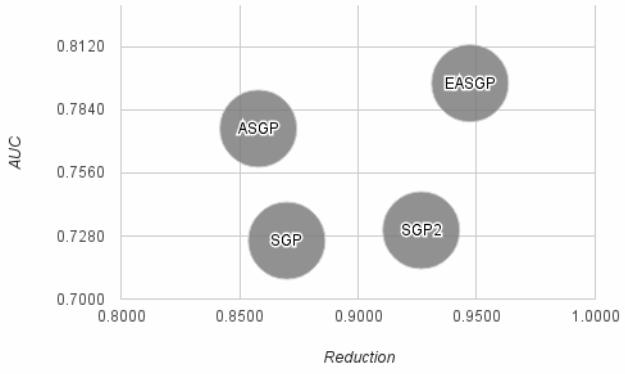


Figure 41 – Dispersion (Reduction vs. AUC) of EASGP, ASGP, SGP and SGP2.

We can state with confidence that EASGP outperformed all previous versions of SGP (SGP, SGP2 and ASGP) in both, classification accuracy and reduction rate.

C.5 Conclusion

This paper presented the Evolutionary Adaptive Self-Generating Prototypes (EASGP), a centroid based prototype generation (PG) algorithm for imbalanced datasets. EASGP

uses an iterative merging to expand the search space, and evolutionary pruning to find the optimal solution.

An experimental study was carried out to compare EASGP and the previous versions of the Self-Generationg Prototupes (SGP). The main conclusions reached were:

1. EASGP outperformed all previous versions of SGP in classification accuracy on imbalanced datasets.
2. EASGP outperformed all previous versions of the SGP in reduction rate on imbalanced datasets.
3. Differently than previous versions of SGP, EASGP can be adjusted to give preference to the classification or reduction with the α parameter.

The use of an iterative merging and evolutionary pruning achieved excelent results. Future works include the use of these algorithms with other PG techniques.