Local learning for dynamic ensemble selection

by

Mariana DE ARAUJO SOUZA

MANUSCRIPT-BASED THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE IN PARTIAL FULFILLMENT FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, JULY 28, 2023

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

ACKNOWLEDGEMENTS

Firstly, I would like to express my immense gratitude to my advisor, Prof. Robert Sabourin, and my co-advisors, Prof. George Darmiton and Prof. Rafael Cruz. Their expertise, guidance, and patience throughout the program were invaluable to the completion of this work.

My gratitude also goes to the members of my thesis committee for evaluating this work and providing constructive feedback.

I would also like to thank my colleagues at LIVIA (Laboratoire d'imagerie, de vision et d'intelligence artificielle), for their comradery and their help throughout the research. Special thanks to Le Thanh Nguyen-Meidine and Luiz Gustavo Hafemann for being very supportive friends.

Finally, I am forever grateful to my family for their encouragement and unconditional support throughout the years. I would not have come this far without them.

Apprentissage local pour la sélection dynamique d'ensembles

Mariana DE ARAUJO SOUZA

RÉSUMÉ

Les techniques de sélection dynamique reposent sur l'idée que les classificateurs d'un ensemble sont experts dans différents domaines de l'espace des caractéristiques. En tant que telles, elles tentent d'identifier uniquement le(s) classificateur(s) le(s) plus compétent(s) pour étiqueter un échantillon de requête donné, généralement sur la base de l'hypothèse de localité, c'est-à-dire en supposant que des instances similaires partagent un ensemble similaire de classificateurs capables de les étiqueter correctement. Par conséquent, la réussite de la tâche de sélection dynamique est étroitement liée à la distribution locale des données, car elle établit la qualité de la région définie pour la tâche de sélection dynamique et peut affecter la manière dont l'expertise locale des classificateurs est perçue. Ainsi, des caractéristiques telles que le chevauchement des classes locales et la rareté des données peuvent conduire à une région locale mal définie présentant une hypothèse de localité faible, entravant ainsi la recherche d'un expert local.

Ainsi, dans cette thèse, plusieurs techniques qui intègrent le contexte local dans le système de classificateurs multiples sont proposées pour améliorer la sélection dynamique des classificateurs dans des scénarios difficiles. À cette fin, la définition d'une région locale adéquate est abordée en caractérisant les données locales et en définissant les régions à l'aide de différentes méthodes pour traiter les distributions complexes et avec des échelles multiples pour fournir un contexte ample au système. La présence d'experts locaux est également prise en compte en produisant le pool sur la frontière locale afin d'obtenir des classificateurs plus spécialisés, et en apprenant la tâche de sélection dynamique de bout en bout à partir des interactions des classificateurs et des relations entre les données locales afin de stimuler la recherche d'experts locaux. Ainsi, en exploitant les informations provenant de la distribution des données locales, la capacité des techniques de sélection dynamique à trouver des experts locaux peut être renforcée, ce qui améliore sa robustesse et ses performances sur des problèmes complexes.

Dans le chapitre 2, la technique du pool local en ligne (OLP) est proposée pour résoudre la difficulté que présentent les techniques de sélection dynamique dans la recherche d'experts locaux dans les zones de chevauchement. À cette fin, la technique OLP génère plusieurs modèles linéaires à proximité de l'instance interrogée avec différents degrés de localité pour produire des classificateurs capables de reconnaître la frontière locale. Pour identifier les zones de chevauchement des classes, une mesure de dureté de l'instance est calculée par mémorisation pour tous les échantillons disponibles, et les classificateurs sont conçus de manière à "couvrir" entièrement la région cible. Les résultats expérimentaux montrent que l'utilisation de la réserve locale générée a permis d'améliorer les techniques de sélection de classificateurs dynamiques évaluées par rapport à une réserve générée globalement, ce qui suggère l'avantage d'avoir des classificateurs localement spécialisés dans la réserve pour la tâche de sélection dynamique. L'approche proposée donne également des résultats similaires à ceux de plusieurs méthodes d'apprentissage de pointe.

Au chapitre 3, une méthode d'ensemble local basée sur l'OLP a été proposée, l'OLP++, pour remédier aux limitations que la première méthode présentait sur les données de haute dimension en raison de sa définition de la région locale sensible aux effets de la malédiction de la dimensionnalité. À cette fin, l'approche OLP++ exploite les partitions de données obtenues à partir d'algorithmes basés sur les arbres pour la définition de la localité, puis produit les experts locaux sur les différents nœuds impurs du chemin de décision qu'une instance de requête donnée traverse dans le(s) arbre(s), introduisant ainsi un contexte local de plus en plus large à l'ensemble local. Les résultats expérimentaux montrent que la définition de la région basée sur la partition récursive de l'OLP++ a permis d'identifier les instances limites plus souvent que la définition de la région basée sur les plus proches voisins de l'OLP, ce qui suggère une amélioration de la distribution des données utilisée pour apprendre les règles linéaires locales. La définition de région de l'OLP++ conduit également à un ensemble local plus diversifié et à une performance statistiquement supérieure à celle de l'OLP sur les données de haute dimension. L'OLP++ surpasse également la forêt aléatoire de référence et plusieurs techniques de sélection dynamique locale, ce qui confirme les avantages de l'approche proposée pour le traitement des données de haute dimension dans le contexte de la sélection dynamique.

Enfin, au chapitre 4, un nouveau système dynamique de classification multiple est proposé pour traiter les données éparses et superposées, car l'OLP++ présente une lacune en raison de sa dépendance à l'égard de partitions prédéfinies qui n'ont pas été optimisées pour la tâche de sélection dynamique. La technique proposée de sélection dynamique d'ensemble par réseau neuronal graphique (GNN-DES) résout ce problème en apprenant la tâche de sélection dynamique de bout en bout à l'aide d'un réseau neuronal graphique (GNN) multiétiquettes, qui est responsable de la sélection des experts locaux. En apprenant à partir des relations locales des échantillons, représentées dans un graphe, et des interdépendances des classificateurs, modélisées dans les méta-étiquettes, le GNN peut apprendre implicitement un espace intégré où l'hypothèse de localité est plus forte sans nécessiter une définition explicite de la région locale. Les résultats expérimentaux démontrent que les techniques classiques de sélection dynamique ont généralement du mal à traiter les données éparses et superposées, et que le GNN-DES est plus performant que la sélection statique de base et que plusieurs techniques basées sur les similarités dans l'espace des caractéristiques. Une analyse plus poussée montre également que le GNN-DES gère mieux les données éparses et les données qui se chevauchent. a obtenu de meilleurs résultats que les techniques concurrentes sur les problèmes où l'hypothèse de localité est plus faible en présence d'un chevauchement de classes, ce qui suggère que l'exploitation de la distribution locale des données et des interactions des classificateurs peut faciliter la tâche de sélection dynamique dans des scénarios difficiles.

Mots-clés: Systèmes de classification multiples, Sélection dynamique, Apprentissage local, Chevauchement de classes, Dureté des instances, Rareté des données, Méta-apprentissage, Réseaux de neurones graphiques

Local learning for dynamic ensemble selection

Mariana DE ARAUJO SOUZA

ABSTRACT

Dynamic selection techniques are based on the idea that the classifiers from an ensemble are experts in different areas of the feature space. As such, they attempt to single out only the most competent one(s) to label a given query sample generally based on the locality assumption, i.e., assuming that similar instances share a similar set of classifiers able to correctly label them. Therefore, the success of the dynamic selection task is strongly linked to the local data distribution, as it establishes the quality of the defined region for the dynamic selection task and may affect how the classifiers' local expertise is perceived. As such, characteristics such as local class overlap and data sparsity may lead to a poorly defined local region presenting a weak locality assumption, thus hindering the search for a local expert.

Thus, in this thesis, several techniques that integrate the local context into the multiple classifier system are proposed to improve the dynamic selection of classifiers over challenging scenarios. To that end, the definition of an adequate local region is addressed by characterizing the local data and defining the regions using different methods to tackle complex distributions and with multiple scales to provide ample context to the system. The presence of local experts is also addressed by producing the pool over the local border to yield more specialized classifiers, and by learning the dynamic selection task in an end-to-end manner from the classifiers' interactions and the local data relations to boost the search for local experts. Thus, by leveraging the information from the local data distribution, the dynamic selection techniques' ability to find local experts may be enhanced, improving its robustness and performance over complex problems.

In Chapter 2, the Online Local Pool (OLP) technique is proposed to tackle the difficulty the dynamic selection techniques present in searching for local experts in overlap areas. To that end, the OLP technique generates several linear models in the vicinity of the query instance with different locality degrees to produce classifiers able to recognize the local border. To identify the class overlap areas, an instance hardness measure is computed in memorization for all the available samples, and the classifiers are so that they fully "cover" the target region. Experimental results demonstrate that using the generated local pool provided an improvement to the evaluated dynamic classifier selection techniques compared to a globally generated pool, suggesting an advantage in having locally specialized classifiers in the pool for the dynamic selection task. The proposed approach also performs similarly to several state-of-the-art learning methods.

In Chapter 3, a local ensemble method based on the OLP was proposed, the OLP++, to address the limitations the former presented over high dimensional data due to its local region definition being susceptible to the effects of the curse of dimensionality. To that end, the OLP++ approach leverages the data partitions obtained from tree-based algorithms for the locality definition, and then produces the local experts over the different impure nodes from the decision path that a given query instance traverses in the tree(s), therefore introducing an increasingly wider local context to the local ensemble. Experimental results show that the OLP++'s recursive partition-based region definition successfully identified borderline instances more often than the OLP's nearest neighbors-based region definition, suggesting an improvement in the data distribution used to learn the local linear rules. The OLP++'s region definition also leads to a more diverse local ensemble and a statistically superior performance compared to the OLP over the high dimensional data. The OLP++ also outperforms the random forest baseline and several local-based dynamic selection techniques, further suggesting the advantages of the proposed approach for dealing with high dimensional data in the context of dynamic selection.

Lastly, in Chapter 4, a novel dynamic multiple classifier system is proposed to deal with sparse and overlapped data as the OLP++ presents a shortcoming due to its reliance on pre-defined partitions that were not optimized for the dynamic selection task. The proposed Graph Neural Network Dynamic Ensemble Selection (GNN-DES) technique addresses this issue by learning the dynamic selection task in an end-to-end manner using a multi-label Graph Neural Network (GNN), that is responsible for the selection of the local experts. By learning from the samples' local relationships, represented in a graph, and the classifiers' inter-dependencies, modeled in the meta-labels, the GNN may implicitly learn an embedded space where the locality assumption is stronger without requiring an explicit local region definition. Experimental results demonstrate that the classical dynamic selection techniques generally struggle over sparse and overlapped data, and that the GNN-DES outperforms the static selection baseline and several techniques based on similarities in the feature space. Further analysis also shows the GNN-DES better deals with performed better than the contending techniques over the problems where the locality assumption is weaker in the presence of class overlap, suggesting that leveraging the local data distribution and the classifiers' interactions can aid the dynamic selection task in challenging scenarios.

Keywords: Multiple classifier systems, Dynamic selection, Local learning, Class overlap, Instance hardness, Data sparsity, Meta-learning, Graph neural networks

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

ADA	AdaBoost ensemble
AUC	Area under the ROC curve
BalancedRF	Balanced Random Forest
BR	Binary Relevance
CD	Critical Difference
CHADE	Chained Dynamic Ensemble
DCS	Dynamic Classifier Selection
DES-C	Dynamic Ensemble Selection-Clustering
DES-KNN	Dynamic Ensemble Selection KNN
DES-RRC	Randomized Reference Classifier
DES	Dynamic Ensemble Selection
DESC	Dynamic Ensemble Selection-Clustering
DESP	Dynamic Ensemble Selection Performance
DFP	Dynamic Frienemy Pruning
DKNN	Dynamic Ensemble Selection-KNN
DS	Decision Space
DS	Decision Stump
DS	Dynamic Selection
DSEL	Dynamic Selection Dataset

DT	Decision Tree
ENL	Excess Neighborhood Labelset
ENN	Edited Nearest Neighbors
ETS	École de Technologie Supérieure
FIRE	Frienemy Indecision Region
FLT	Forest of Local Trees ensemble
FS	Feature Space
G-mean	Geometric Mean
GCN	Graph Convolutional Network
GNN-DES	Graph Neural Network Dynamic Ensemble Selection
GNN	Graph Neural Network
GP	Global Pool
HDSSS	High-Dimensional Small Sample-Sized
IR	Imbalance Ratio
K-NN	K-Nearest Neighbors
k-NN	k-Nearest Neighbors
k-NNE	k-Nearest Neighbor Equality
KDN	K-Disagreeing Neighborhood
kDN	k-Disagreeing Neighbors
KDNi	K-Disagreeing Neighbors-imbalance

KNE	K-Nearest Oracles Eliminate
KNN	K-Nearest Neighbors
KNNE	K-Nearest Neighbors Equality
KNOP	K-Nearest Output Profiles
KNORA-B	K-Nearest Oracles Borderline
KNORA-BI	K-Nearest Oracles Borderline-Imbalance
KNORA-E	K-Nearest Oracles Eliminate
KNORA-U	K-Nearest Oracles Union
KNU	K-Nearest Oracles Union
LCA	Local Class Accuracy
LCard	Labelset Cardinality
LP	Local Pool
LSC	Local Set Cardinality
LSCi	Local Set Cardinality-imbalance
MCB	Multiple Classifier Behavior
MCS	Multiple Classifier Systems
META-DES	Meta-learning for Dynamic Ensemble Selection
MLP	Multi-Layer Perceptron
MV	Majority Voting
NLI	Neighborhood Labelset Intersection

OLA	Overall Local Accuracy
OLP	Online Local Pool
OVO	One-Versus-One
PCC-DES	Probabilistic Classifier Chain Dynamic Ensemble Selection
RF	Random Forest
ROC	Receiver Operation Characteristic
RoC	Region of Competence
RRC	Randomized Reference Classifier
SGH	Self-Generating Hyperplanes
SLI	Supervised Labelset Intersection
SS	Static Selection
SVM	Support Vector Machine

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

$\mathcal{T}$	Training set
$\mathcal{V}$	Validation set
C	Pool of classifiers
c_j	*j*-th classifier from the pool
LP	Local pool
Ω	Set of the problem's classes
ω_l	*l*-th class from the problem
$\mathbf{x}_i$	Feature vector of the *i*-th instance
y_i	True label of the *i*-th instance
$\mathbf{x}_q$	Feature vector of the query instance
y_q	True label of the query instance
$\hat{y}_q$	Predicted label of the query instance
θ_q	Region of Competence (RoC) of the query instance
K	RoC size
k	RoC size
α	Significance level
LP	Local Pool
M	Local pool size
H	Instance hardness estimates

k_h	Neighborhood size for instance hardness
k_s	Initial neighborhood size for local pool
$\mathcal{R}$	Set of classes' centroids
$\mathbf{r}_i$	Centroid of i-th class
θ_m	m-th region
C_m	m-th subpool
$c_{m,k}$	k-th classifier from C_m
$\mathcal{A}$	Set of decision trees
A_j	j-th decision tree
T	Number of decision trees
L	Number of tree levels
k	Minimum leaf size
$\eta_{q,j}$	Decision path of query in A_j
$\theta^l_{q,j}$	Data partition at the l-th node in path $\eta_{q,j}$
$\theta^\ell_{q,j}$	Data partition at the leaf node in path $\eta_{q,j}$
G	Graph
V	Set of vertexes
E	Set of edges
v_i	i-th vertex
$\mathbf{e}_{i,j}$	Attributes of the edge between v_i and v_j

$\mathbf{h}_i^l$	Hidden representation of v_i at the l-th layer
$\mathcal{N}(v_i)$	Set of neighboring nodes of v_i in G
$\mathbf{W}^l$	Learnable parameters at the l-th layer
$\sigma()$	Non-linear activation function
$\alpha_{i,j}^l$	Attention coefficient at the l-th layer between v_i and v_j
$G_\mathcal{T}$	Known graph
GNN	Meta-learner
$p_{i,k}$	Output probability from c_k] to $\mathbf{x}_i$
$\mathbf{p}_i$	Output probabilities for $\mathbf{x}_i$
$d_{i,j}$	Normalized *L1* distance between $\mathbf{p}_i$ and $\mathbf{p}_j$
$o_{i,k}$	Value of the network's k-th output node for input $\mathbf{x}_i$
$u_{i,k}$	Indication of (in)correct classification of $\mathbf{x}_i$ from c_k
$\mathbf{u}_i$	Meta-labels of $\mathbf{x}_i$
G_q	Evaluation graph
U_i	Meta-labelset of i-th sample
$\mathbf{v}_i$	Meta-feature vector of i-th sample
t	Output class probability threshold
p_o	Proportion of samples from opposite class
LS	Local set
$\mathbf{x}_{ne}$	Nearest enemy

TPR	True positive rate
TNR	true negative rate
P	Number of positive class instances
N	Number of negative class instances
TP	Number of true positives
TN	Number of true negatives
FP	Number of false positives

INTRODUCTION

Multiple Classifier Systems (MCS) integrate the responses of several classifiers with the purpose of leveraging their complementarity so that the combined system outperforms each individual learner (Kittler, Hatef, Duin & Matas, 1998; Woźniak, Graña & Corchado, 2014). Several studies found in the literature demonstrate the effectiveness of MCS in improving the recognition rates over monolithic classifiers (Kuncheva, 2002, 2014; Zhou, 2012) and solving real-world problems (Gao, Shan, Hu, Niu & Liu, 2019; Goel, Sharma, Khatri & Damodaran, 2020; Cao, Geddes, Yang & Yang, 2020).

MCS are usually divided into three phases (Cruz, Sabourin & Cavalcanti, 2018a): generation, selection and aggregation. In the generation phase, the classifiers that comprise the pool of classifiers are generated. The second phase, called selection, is when the classifiers in the pool are singled out for performing the classification task. Lastly, in aggregation, the responses of all selected classifiers are combined to produce the system's output.

The generation phase of an MCS is responsible for generating the classifiers, with the goal of producing an accurate and diverse pool. Diversity is an intuitive characteristic of an ensemble of classifiers, which is considered *diverse* if the models make different mistakes in different regions of the feature space (Kuncheva & Whitaker, 2003). This characteristic is important to MCS since there is no point in producing and combining identical classifiers. Most generation techniques use one or more of the following approaches for obtaining a diverse pool of classifiers (Duin, 2002): different initializations, different hyperparameters, different architectures, different model types (i.e., producing an *heterogeneous* pool), different training sets (obtained from the original dataset via sampling (Breiman, 1996; Schapire, Freund, Bartlett & Lee, 1997), data partitioning (Kuncheva, 2000), etc.), or different feature representations (e.g, by extracting different features from the same raw data (Bashbaghi, Granger, Sabourin & Bilodeau, 2017), or sampling the original feature space (Ho, 1998)).

In the selection phase of an MCS, the classifiers to be used in the classification task are singled out. This phase is optional, so all classifiers in the pool may be used for labeling the test samples. The classifier selection may be either static or dynamic. In static selection techniques, the classifiers are selected in training, so the same selected ensemble is used to label all query instances. The ensemble selection can be performed using optimization algorithms, such as greedy search (Partalas, Tsoumakas & Vlahavas, 2008) and evolutionary algorithms (Dos Santos & Sabourin, 2011), with the fitness function usually based on accuracy rate (Ruta & Gabrys, 2005; Dos Santos, Sabourin & Maupin, 2009) and/or diversity metrics (Giacinto & Roli, 2001; Dos Santos & Sabourin, 2011). Dynamic selection techniques, on the other hand, perform the selection in generalization, aiming at selecting the classifier(s) that are best fit for labeling each query sample in particular according to its(their) perceived, or estimated, competence in the task. The competence estimation of the classifiers may be based on several criteria, including local accuracy (Woods, Kegelmeyer Jr & Bowyer, 1997), ensemble diversity (Soares, Santana, Canuto & de Souto, 2006), classifier behavior (Giacinto, Roli & Fumera, 2000), probabilistic models (Giacinto & Roli, 2001; Woloszynski & Kurzynski, 2011), ranking (Woods *et al.*, 1997), data complexity (Brun, Britto, Oliveira, Enembreck & Sabourin, 2016), and meta-learning (Cruz *et al.*, 2015a; Pinto, Soares & Mendes-Moreira, 2016), among others.

In the aggregation phase, the responses of the selected classifiers (or all classifiers, if no selection took place) are combined to form the final output of the MCS. The combination can be performed over the classifiers' output labels for the query sample, as in voting schemes, or their output class supports, as in probabilistic and fuzzy-based approaches (Kittler *et al.*, 1998). There are three approaches to combining the responses of the classifiers (Duin, 2002): fixed rules, such as majority voting, mean, product and sum, trained combiners, in which a classifier is trained to learn the aggregation function and combine the responses, as in the Mixture of Experts (ME) paradigm (Jacobs, Jordan, Nowlan & Hinton, 1991; Armano & Hatami, 2010b), and dynamic weighting, in which a weight is assigned to each classifier according to the perceived confidence in their responses.

0.1 Problem statement

The focus of this thesis is on dynamic selection approaches, which were shown to work quite well over a wide range of problems, including class imbalanced (Oliveira, Cavalcanti, Porpino, Cruz & Sabourin, 2018) and ill-defined (Britto, Sabourin & Oliveira, 2014) distributions. They were also shown to outperform static selection schemes in several occasions (Cruz *et al.*, 2015a; Woloszynski & Kurzynski, 2011).

The reasoning behind dynamic selection techniques is that each classifier in the pool may be an expert in different regions of the feature space, so it could be advantageous to single out only the most competent one(s) in the area where the query sample is located. Thus, most techniques rely on the locality assumption to solve the dynamic selection task, that is, it is expected that similar instances are correctly labeled by a similar set of classifiers from the pool. These techniques require defining a local region, called Region of Competence (RoC), using clustering (Soares *et al.*, 2006), nearest neighbors rule (Ko, Sabourin & de Souza Britto Jr, 2007; Cavalin, Sabourin & Suen, 2012), distance-based potential function (Woloszynski & Kurzynski, 2011), recursive partitioning (Souza, Sabourin, Cavalcanti & Cruz, 2023), and/or fuzzy hyperboxes (Davtalab, Cruz & Sabourin, 2022), and then estimating over the RoC the competences of the classifiers in the pool according to some criteria (Cruz *et al.*, 2018a).

So, to properly achieve the dynamic selection of classifiers for a given unknown sample, three tasks need to be accomplished: (a) the RoC must be defined so that the local data aids the search for the local experts, (b) there must be at least one expert in that local region to be found in the pool, and (c) the local experts must be identified and singled out through their estimated competences. On the one hand, while several works in the literature address different selection criteria to tackle (c) (Ko *et al.*, 2007; Soares *et al.*, 2006; Brun *et al.*, 2016; Cruz *et al.*, 2015a), the effectiveness of the competence estimation and classifier selection depends directly on the defined RoC and the models available in the pool. On the other hand, the local data distribution plays an important role in fulfilling (a) and (b), and therefore (c), as the presence of unreliable samples (Pereira, Britto, Oliveira & Sabourin, 2018) and class overlap (Oliveira *et al.*, 2017) affect the quality of the defined RoC and how the classifiers' local expertise is perceived.

As such, the success of the dynamic selection task is strongly linked to the local context of the data. Characteristics such as local class overlap (Cruz, Oliveira, Cavalcanti & Sabourin, 2019a; Cruz, Sabourin & Cavalcanti, 2018b) and noise (Cruz, Sabourin & Cavalcanti, 2015b) were already shown to hinder the techniques' recognition rates. The locality definition can also be negatively affected by high dimensionality and class ambiguity (Zhang, 2022; Vandaele, Kang, De Bie & Saeys, 2022). Moreover, as with other local methods, the dynamic selection techniques may be sensitive to overlap and data sparsity (Sánchez, Mollineda & Sotoca, 2007), with the latter being often associated with increased complexity in the class boundary (Lorena, Costa, Spolaôr & De Souto, 2012; Ho & Basu, 2002). Thus, incorporating local data information into the system could enhance the dynamic selection techniques' ability to find local experts, and thus improve its robustness and performance over challenging scenarios.

0.2 Objectives

The objective of this thesis is to develop techniques that better integrate the local context into the multiple classifier system to improve the dynamic selection of classifiers. As the actual selection of the classifiers requires the definition of a local region and the presence of local experts in the pool, this thesis proposes to embed the local information into the system so that the latter adapts to the surrounding context and approaches the task with a local perspective of the problem from the beginning of the selection process.

The presence of local experts in the pool is addressed in two different ways. In the first, the local context is integrated into the generation step by producing the classifiers on the fly in the local region of each query sample in order to encourage the presence of local experts in the area, if there is any class overlap in the latter. That way, the availability of classifiers that recognize the local border in the pool is increased, which might make it easier for the dynamic selection scheme to select a local expert to label the query. In the second, the local context is incorporated into the system by representing the data in a graph structure which is then used as input to a multi-label graph neural network (GNN) responsible for learning the ensemble's dynamic combination rule in an end-to-end manner. Since the network uses the classifiers' interactions and the instances' relationships to learn the dynamic selection task, it enforces the samples' local relations subject to how similar their set of competent classifiers are, which in turn encourages the search for true experts in each local area.

Moreover, three venues are explored in the local region definition step: the most common nearest neighbors rule, recursive partitioning to deal with sparse data, and an implicit region definition via node representation learning to tackle high dimensional and overlapped data. In all of them, the local information is not limited to a single fixed region size, but rather in multiple scales in order to provide ample context to the system. Furthermore, the class overlap in the local region is identified and characterized so that the system is able to focus on these areas and thus perform better over the ambiguous scenarios known to affect the selection task.

0.3 Contributions

The main contributions of this thesis are the investigation of the role the local data distribution plays in the performance of dynamic selection techniques and the ensuing development of techniques that integrate local information into these systems to enhance their performance over challenging scenarios. As the thesis is manuscript-based, each chapter presents a distinct contribution that works towards achieving the thesis' objective and comprising its main contributions. The chapter-specific contributions are the following:

- In Chapter 2, a novel local ensemble method that locally produces on the fly the pool of classifiers in overlap areas of the feature space is presented. Experiments showed the local pool provided an improvement to several dynamic selection schemes compared to using a globally generated pool, and achieved a statistically similar performance to most evaluated state-of-the-art techniques.
- In Chapter 3, a local ensemble method based on the OLP is presented, the OLP++ framework, to deal with the OLP's shortcomings over high dimensional data. The proposed technique uses recursive partitioning to define the local regions, and generates the local pool on different nodes in ambiguous branches of decision trees. Experiments showed the OLP++ surpassed the OLP and several dynamic techniques over high dimensional small sample sized (HDSSS) datasets.
- In Chapter 4, an evaluation of the performance of several dynamic selection techniques using different ensembles over HDSSS problems is carried out, and a dynamic multiple classifier system is presented to address their limitations associated with a weaker locality assumption in sparse and overlapped data. The instances' relationships are encoded into a graph, and the classifiers' interactions into multiple meta-labels, and both information are leveraged by a multi-label GNN to learn the dynamic combination rule. Experiments showed the proposed approach performed better over locally challenging scenarios.

0.4 Organization of the thesis

Figure 0.1 shows an overview of the proposed organization for the manuscript-based thesis. The boxes in blue represent the main contributions of the thesis, while the black boxes indicate the works associated with the main topic which help contextualize some of the aspects within the proposed approaches.

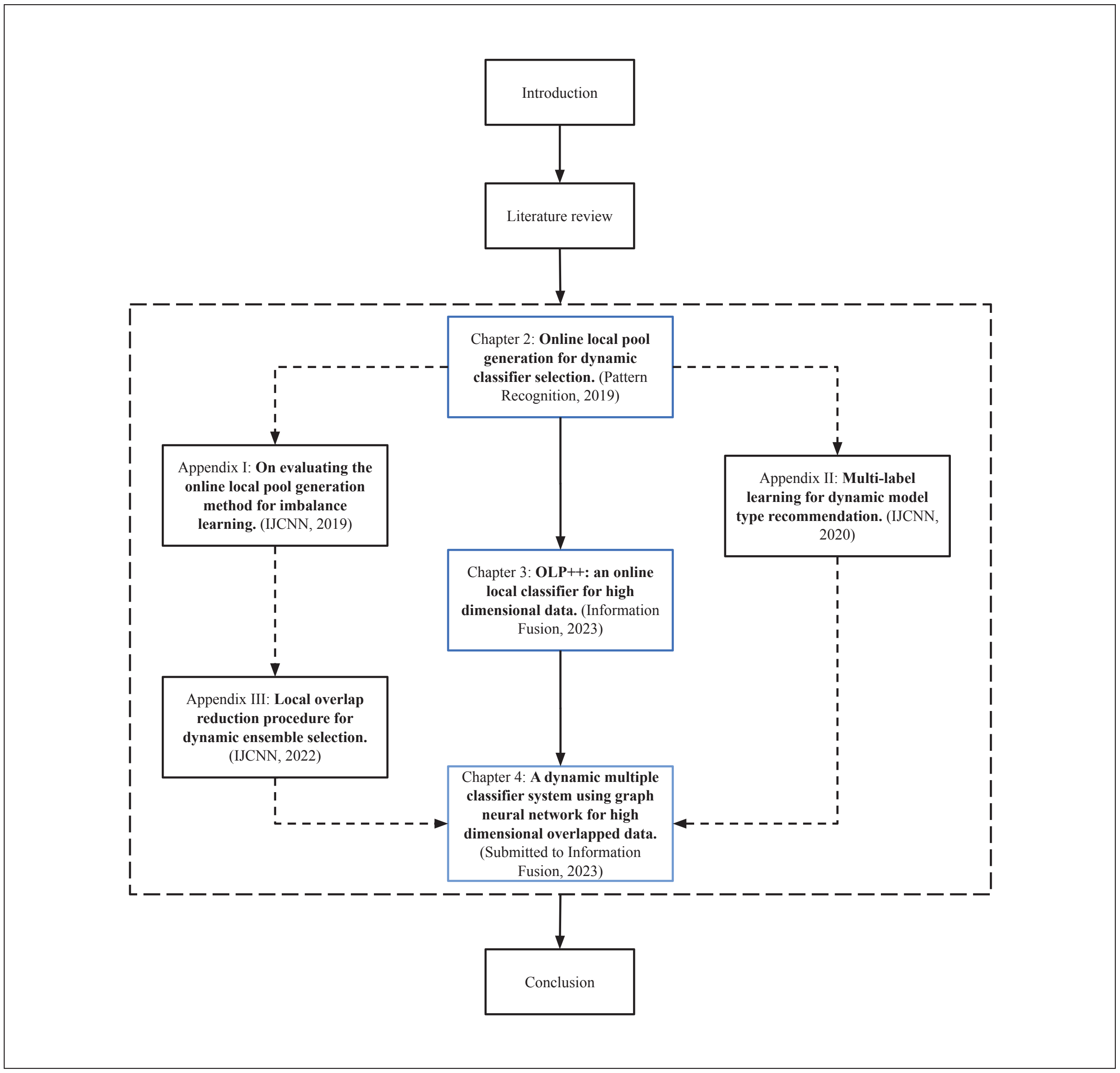

Introduction
Literature review
Chapter 2: **Online local pool generation for dynamic classifier selection.** (Pattern Recognition, 2019)
Appendix I: **On evaluating the online local pool generation method for imbalance learning.** (IJCNN, 2019)
Appendix II: **Multi-label learning for dynamic model type recommendation.** (IJCNN, 2020)
Chapter 3: **OLP++: an online local classifier for high dimensional data.** (Information Fusion, 2023)
Appendix III: **Local overlap reduction procedure for dynamic ensemble selection.** (IJCNN, 2022)
Chapter 4: **A dynamic multiple classifier system using graph neural network for high dimensional overlapped data.** (Submitted to Information Fusion, 2023)
Conclusion

Figure 0.1 Thesis organization. The arrows indicate the reading flow of the thesis. Solid arrows show the chapters' dependencies, and dashed arrows show the suggested additional readings which help the comprehension of the following work.

In the first chapter, a literature review on local ensemble learning is presented, more specifically the approaches that employ dynamic ensemble selection. As this thesis relates to local learning, the review is mainly focused on the local aspects of the techniques.

Chapter 2 presents the first main contribution of the thesis, the Online Local Pool (OLP) technique. Motivated by the dynamic selection techniques' struggle to single out local experts , the OLP technique generates several linear decision borders in the vicinity of each query on the fly, if in a class overlap area. The region over which the local classifiers are generated is defined using the nearest neighbor rule, with an increasing neighborhood size, applied in the feature space. The class overlap detection, which triggers the generation of the local pool, is performed using an instance hardness measure from the literature. Moreover, the linear models are produced using the Self-Generating Hyperplanes (SGH) (Souza, Cavalcanti, Cruz & Sabourin, 2017) technique, which guarantees each known sample in the region is correctly labeled by at least one classifier in the pool. Experiments showed that the performance of three dynamic selection techniques improved when using the local pool instead of a globally generated pool, which suggests an advantage in having highly specialized classifiers in the pool for the dynamic selection task. The OLP also yielded a similar performance to several state-of-the-art methods. The contents of this chapter were published in the Pattern Recognition journal (Souza, Cavalcanti, Cruz & Sabourin, 2019b).

In Appendix I the OLP is then evaluated over imbalanced problems, as its local approach to pool generation suggests a reduced impact from the effects of a global disproportion between the class sizes. Two additions to the OLP technique are also evaluated, both individually and jointly: a class-balanced nearest neighbors rule, and a class-sensitive local overlap-reducing pre-processing technique. Experimental results showed that the OLP alone could yield an improvement over the baseline over imbalanced problems in some cases, but even more so when using the two evaluated modifications, demonstrating the impact the local data distribution and region definition have on the technique's performance. The contents of this appendix were published in the Proceedings of the International Joint Conference on Neural Networks (IJCNN) (Souza, Cavalcanti, Cruz & Sabourin, 2019a).

A problem-independent dynamic model type recommendation framework for the OLP was proposed in Appendix II, with the purpose of choosing the most indicated model type to compose the local pool for each particular local data distribution. To that end, the framework characterizes each sample's local region from a set of different problems using 12 data complexity measures and identifies which classifier models among 5, within the OLP framework, yielded a correct classification. A multi-label classifier is then trained using this meta-data, and in generalization, the same complexity measures are extracted from a given query sample over the target problem so that the meta-learner may recommend the model type to be produced in the region. Experiments showed that always using the best model type yielded a significant improvement to the OLP technique, further suggesting that the local data distribution plays an important role in the production of the local experts. However, the using the meta-learner's recommendations did not significantly outperform the original OLP, which may indicate the meta-features used in the recommender system were insufficient or inadequate for it to properly function in a problem-independent context. The contents of this appendix were published in the Proceedings of the International Joint Conference on Neural Networks (IJCNN) (Souza, Sabourin, Cavalcanti & Cruz, 2020).

In Chapter 3, a novel version of the OLP is proposed for dealing with high dimensional data, motivated by the susceptibility of the OLP's local region definition to high dimensional spaces. Instead of relying on the nearest neighbors rule, which can suffer from the curse of dimensionality, the OLP++ defines the locality using the recursive partitioning procedure from decision trees. Thus, the regions used for the local pool generation are the different impure nodes from the decision path that a given query sample traverses in the tree(s). By using different node levels from the path, each classifier in the local pool has a moderately distinct view of the target region, introducing thus diversity to the pool, without resorting to a dissimilarity metric, which might be susceptible to high dimensional spaces. Experiments showed that the partition-based region definition was more successful in detecting the overlap areas of the feature space in most of the 39 HDSSS problems evaluated, which indicates that it could often obtain a better data distribution for learning the local decision rules compared to the nearest neighbors rule. The OLP++ also statistically outperformed the OLP and the Random Forest (Breiman, 2001) ensemble, and yielded a competitive performance compared to other local-based ensembles, further suggesting the suitability of the proposed approach for dealing with high dimensionality. The contents of this chapter were published in the Information Fusion journal (Souza *et al.*, 2023).

Another supporting work on local instance characterization is presented in Appendix III motivated in part by the analysis of Appendix I with regards to the local class overlap's impact on local methods, especially when dealing with imbalanced classes. A novel dynamic ensemble selection is proposed which attempts to quantify the local unreliability of each instance using two instance hardness measures, adapted or not to class imbalanced data, which convey the degree of local overlap in its vicinity. These measures are then used to choose which samples to remove in the local region during the search for a competent classifier, so that the most reliable ones remain for the competence estimation procedure. Experimental results showed that the instance characterization was successful in boosting the performance of the baseline, with certain measures working well for different types of data distributions, and showed that the local overlap characterization could be successfully used to improve the local data distribution for the dynamic selection task. The contents of this appendix were published in the Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN) (Souza, Sabourin, Cavalcanti & Cruz, 2022).

In Chapter 4, the issue of better defining the locality in sparse (and overlapped) data from Chapter 3 is revisited, as the OLP++ presents a large drawback: while the recursive partitioning region definition was superior to the fixed-sized nearest neighbors rule of the OLP, the quality of the local decision rules depends directly on the pre-defined partitions. Thus, as the partitioning is not optimized for the dynamic selection task, a poorly defined region (e.g. a very large partition, as often occurred in (Souza *et al.*, 2023)) may lead to an equally bad ensemble with a weak locality assumption, meaning that the local samples may not share a similar enough set of competent classifiers. So, in Chapter 4, the Graph Neural Network Dynamic Ensemble Selection (GNN-DES) technique is proposed in which, instead of generating the local experts, a multi-label GNN is trained in an end-to-end manner to dynamically select the local experts from an existing pool using information from both the local data characterization and the classifiers' inter-dependencies. That way, the meta-classifier may define the locality implicitly by learning an embedded space where the locality assumption for the dynamic selection task is stronger, as it can leverage information from two distinct sources. Experiments over 35 HDSSS datasets using different ensemble methods showed the dynamic selection techniques generally struggled over the sparse and overlapped data. However, the GNN-DES was shown to statistically surpass the baseline and most evaluated techniques, including the OLP++. Moreover, further analysis showed that the GNN-DES could better deal with unfavorable local data contexts compared to the contending techniques, which suggests that integrating the instances' local relations and the classifiers' interactions into the dynamic selection pipeline may be advantageous to tackle challenging scenarios.

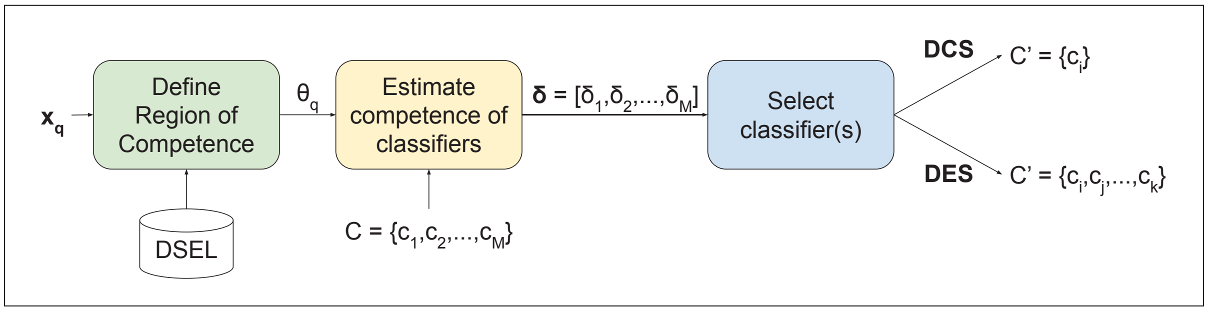
In the last chapter, a general conclusion for this work and possible future works are presented.

CHAPTER 1

LITERATURE REVIEW

1.1 Overview of dynamic selection techniques

Dynamic selection techniques can be usually divided into three steps (Cruz *et al.*, 2018a): the *region of competence definition*, in which the Region of Competence (RoC) of a given query sample is obtained, the *competence estimation*, in which the competence of each classifier in the pool is computed over the RoC, and the *classifier(s) selection*, in which the classifier(s) deemed the most competent one(s) are selected to perform the classification task. This process is illustrated in Figure 1.1, in which $\mathbf{x_q}$ is the query sample, $DSEL$ is the training or validation set, θ_q is the neighborhood of $\mathbf{x}_q$, C is the pool of classifiers, $\boldsymbol{\delta}$ is the vector with the classifiers' competence estimates, and C' is the set of selected classifiers. Techniques that select only one classifier are called Dynamic Classifier Selection (DCS) techniques, while the ones that single out multiple models are called Dynamic Ensemble Selection (DES) techniques.



$\mathbf{x_q}$
Define Region of Competence
DSEL
θ_q
Estimate competence of classifiers
$C = \{c_1, c_2, \ldots, c_M\}$
$\boldsymbol{\delta} = [\delta_1, \delta_2, \ldots, \delta_M]$
Select classifier(s)
DCS
$C' = \{c_i\}$
DES
$C' = \{c_i, c_j, \ldots, c_k\}$

Figure 1.1 Usual steps in a dynamic selection scheme. $\mathbf{x_q}$ is the query sample, $DSEL$ is a labeled dataset (training or validation set), θ_q is the RoC of $\mathbf{x_q}$, C is the pool of classifiers, $\boldsymbol{\delta}$ is the competence vector and C' is the set of selected classifiers.

1.1.1 Region of competence definition

Most dynamic selection techniques are local-based in the sense that they rely on the locality assumption to perform the selection task. As such, the first step in the selection process for these techniques is to define the RoC, a region where the competence of the classifiers in the pool is estimated. As this thesis concerns local learning for dynamic selection techniques, and the RoC definition step determines the local context to be considered in the selection task, the ways in which it can be performed are presented in more detail in this section.

However, it is worth noting that, while crucial for local-based techniques, the RoC definition step is not performed in a few techniques found in the literature because they do not rely on local information to perform the selection task. The Chained Dynamic Ensemble (CHADE) (Pinto *et al.*, 2016) and the Probabilistic Classifier Chain Dynamic Ensemble Selection (PCC-DES) (Narassiguin, Elghazel & Aussem, 2017) techniques define the selection task as a multi-label meta-problem in which the meta-labels indicate the classifiers that are competent to label a given instance. Both techniques then train a multi-label classifier to learn the dynamic ensemble combination rule without explicitly taking into account the samples' local context. While this approach could be interesting in scenarios where the local information does not aid the selection task, these techniques completely disregard the local context and can perform poorly against simple local accuracy-based techniques (Pinto *et al.*, 2016). They can also present a high computational overhead due to the use of a meta-learner ensemble (Pinto *et al.*, 2016) and Monte Carlo sampling (Narassiguin *et al.*, 2017).

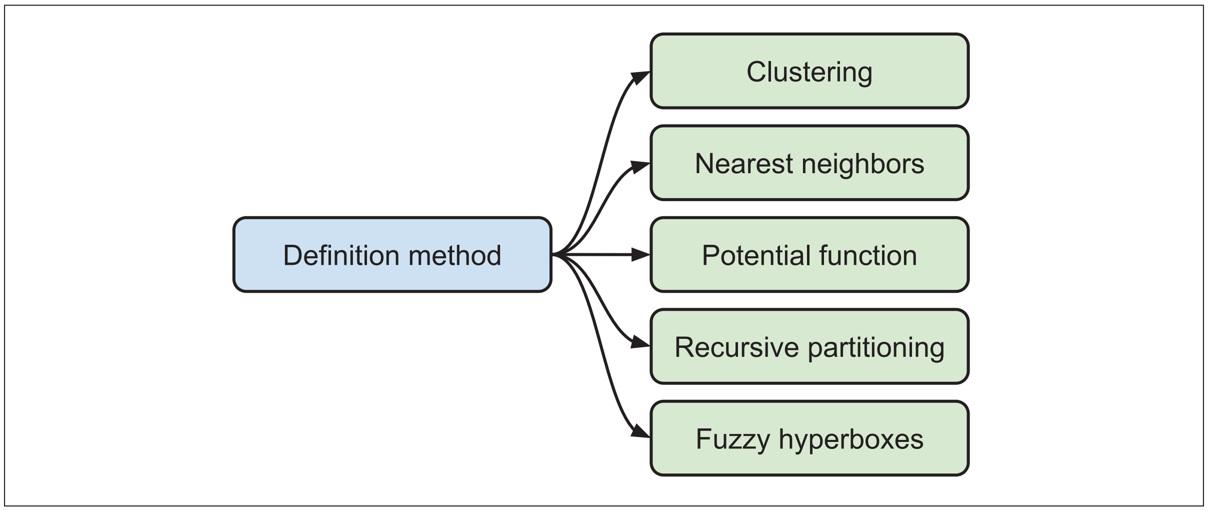


Figure 1.2 Categorization with regards to the definition method applied in the RoC definition step.

To present an overview of the RoC definition procedure, the approaches can be categorized as illustrated in Figure 1.2 with respect to the definition method used for delimiting the target region. The RoC definition methods found in the literature are discussed next.

1.1.1.1 Clustering

Based on the clustering-and-selection approach (Kuncheva, 2000), clustering-based dynamic selection techniques define the RoC using a clustering method over the DSEL in memorization. Then, the classifiers in the pool are evaluated over each cluster and their corresponding competence estimates are computed and stored. In generalization, the cluster whose centroid is closest to the query sample is defined as its RoC , so the classifiers' competencies over that cluster are used for selecting the best one(s).

This approach is applied in the DCS technique proposed in (Kuncheva, 2000) and in the DES technique proposed in (Soares *et al.*, 2006). In both techniques, the K-means algorithm is applied in the feature space to define the local regions in memorization, and in generalization the query instances are assigned to their closest cluster. As such, the size of the RoCs in these techniques may differ depending on which data partition the query falls into. Other hybrid techniques that couple ensemble generation and selection, such as the Cluster-oriented ensemble classifier (Verma & Rahman, 2011) and the One-class clustering-based ensemble (Krawczyk & Cyganek, 2017), define the regions using clustering methods, the former using K-means and the latter using fuzzy clustering, and assign members of the pool to them, which are then selected in generalization according to the query's proximity to the data partitions.

Using clustering methods for RoC defining is generally cost effective since the regions and the classifiers' competences over them are computed offline, and the distance calculation in generalization is performed over the clusters' centroids instead of over all DSEL samples Kuncheva (2014). However, this approach tends to be more limited in the design of the selected ensemble due to the local region granularity (Soares *et al.*, 2006; de Souto, Soares, Santana & Canuto, 2008).

1.1.1.2 Nearest neighbors

A very common way of defining the RoC is using the nearest neighbors rule. In this approach, the distances between the query instance and all known samples in the DSEL are calculated, and the K (previously pre-set) closest ones are singled out as the query's RoC θ_q, which are then used in the competence estimation step. The distance computation may be performed in the feature space (Woods *et al.*, 1997; Smits, 2002; Ko *et al.*, 2007; Soares *et al.*, 2006), in the decision space (Cavalin *et al.*, 2012), where each sample is represented by the classifiers' response to it, or in both (Cruz *et al.*, 2015a).

As the RoC definition can greatly impact the subsequent steps of the selection process, several works propose applying different nearest neighbors rules to improve the performance of the techniques. In (Cruz *et al.*, 2018b) a local adaptive distance function that increases the distance to the border samples is evaluated within the nearest neighbors rule to reduce the incidence of noise in the RoC. An adaptive nearest neighbors rule is also used in (Didaci & Giacinto, 2004) where a distance metric based on the Linear Discriminant Analysis is applied to form neighborhoods of varying sizes that present more homogeneous class posterior probabilities. The K-Nearest Neighbors Equality (KNNE) (Sierra, Lazkano, Irigoien, Jauregi & Mendialdua, 2011), in which the same amount of nearest neighbors from each class are singled out, is used in (Mendialdua, Martínez-Otzeta, Rodriguez-Rodriguez, Ruiz-Vazquez & Sierra, 2015) and (Cruz *et al.*, 2019a) to adapt the RoC definition to One-Versus-One (OVO) approaches for multi-class problems, and to better characterize the local border, respectively.

Other techniques edit the RoC to filter out unwanted samples from the originally defined neighborhood, possibly yielding RoCs of distinct sizes. In (Giacinto *et al.*, 2000) the samples with an output profile, i.e. their decision space representation, further from the query's output profile by more than a pre-set threshold are excluded from the RoC. In (Pereira *et al.*, 2018) the samples in the neighborhood are ranked according to their estimated Item Response Theory's (IRT) discrimination index, and the bottom half of the list is excluded from the RoC. In (Ko *et al.*, 2007) and (Oliveira *et al.*, 2018) the RoC definition and competence estimation steps are coupled, and one sample at a time is removed from the RoC iteratively based on distance alone or considering also the class distribution, respectively, until a competent classifier is found.

Using the nearest neighbors rule may provide a more precise RoC definition due to the controlled degree of locality, which in turn may yield more diverse ensemble compositions at the end of the selection process (Soares *et al.*, 2006). Nevertheless, the computational cost of calculating the distances between the query and all samples in the DSEL can be quite high, especially for large datasets.

1.1.1.3 Potential function

Differently from the clustering and nearest neighbors approaches, which define the RoC as a subset of the DSEL set, potential function approaches use all samples from the DSEL set as the RoC (Woloszynski & Kurzynski, 2011; Woloszynski, Kurzynski, Podsiadlo & Stachowiak, 2012; Antosik & Kurzynski, 2011). However, the influence of each sample in the competence estimation of the classifiers is computed using a weighting scheme based on a potential function model with its distance to the query sample as the argument. That way, the closer to the query instance in the feature space, the higher the sample's importance in the competence estimation step. The Gaussian potential function is usually applied in these techniques.

The advantage of defining the RoC with a potential function is that there is no need to define the size of the neighborhood or the number of partitions in which to divide the data. Nevertheless, the computational cost is increased since the competence estimation is computed using the entire DSEL set instead of only a subset of the dataset.

1.1.1.4 Recursive partitioning

Another way of defining the RoC is by recursively partitioning the feature space. In this approach, a recursive algorithm, e.g. the Classification and Regression Tree (CART) (Breiman, 2017) algorithm, is used to split the data, most commonly with a hyperplane, until a stopping criterion is reached. In generalization, the RoC can then be defined from the data partition(s) the query falls into within the tree-based structure.

A few hybrid approaches use recursive partitioning to define the local regions associated with the members of the ensemble (Zhu, Wang, Li & Du, 2019; Burduk & Biedrzycki, 2022). In Zhu *et al.* (2019), the feature space is recursively partitioned on the axis of the maximum within-class scatter of the majority class to reduce the effects of the local class imbalance, and a classifier is trained at each leaf to label the queries that fall into it. In (Burduk & Biedrzycki, 2022) the feature space is split into disjoint regions using all splits obtained from an ensemble of decision trees. The local rules of the combined ensemble are then defined for each subspace based on the trees' original local rules and the neighboring subspaces' class distribution.

One advantage of recursive partitioning is the computational cost, which similarly to clustering techniques is reduced compared to the nearest neighbors approach as the regions are pre-defined in training time. Moreover, the partitioning process may not require the use of a distance metric, which can be advantageous over sparse data as the latter is more susceptible to the distance concentration phenomenon (François, Wertz & Verleysen, 2007). However, this process provides less control over the degree of locality compared to the nearest neighbors rule, which may lead to a weak locality assumption within the region for the dynamic selection task.

1.1.1.5 Fuzzy hyperboxes

We also find in the dynamic selection literature the RoC definition via fuzzy hyperboxes (Davtalab *et al.*, 2022). Fuzzy hyperboxes (Simpson, 1992) are hyper-rectangles defined in the feature space by two samples that delimit their corners. Their coverage also extends outside the interior structure according to a fuzzy membership function. The RoC definition step via fuzzy hyperboxes works thus by defining the hyperboxes in the feature space during training time, so that in generalization the membership values for the query can indicate which regions to use for competence estimation.

How the hyperboxes are obtained, though, depends on the approach used in the dynamic selection technique. In (Davtalab *et al.*, 2022) two approaches are explored within the same framework that assigns a group of (possibly overlapping) hyperboxes to each classifier according to its performance in the region. So in the first approach, for each classifier, the samples that it correctly labels are partitioned using the hyperbox learning algorithm, which expands the region from one sample until a given maximum size or it is not expandable anymore. In the second approach, the same mechanism is performed but the partitioning occurs over the misclassified samples instead of correctly labeled samples. In generalization, the membership values for the query sample are calculated and two highest associated with each classifier are used to estimate its competence.

Fuzzy hyperboxes have similar advantages to clustering approaches with respect to computational cost and increased (and varying) scope compared to the nearest neighbors rule, while also defining local structures that work well in high dimensional spaces. Nevertheless, in dense distributions where the neighborhood is representative of the unknown instance, the nearest neighbors approach seems to provide a superior local region for dynamic selection schemes (Davtalab *et al.*, 2022).

1.1.2 Competence estimation

In the competence estimation step, the classifiers are evaluated (over the RoC, if defined) and, according to the type of information used in the evaluation and the selection criterion of the technique, a competence level is computed for either each classifier in the pool, in individual-based measures, or certain subsets of the pool, in group-based measures. Individual-based measures extract information regarding the performance of each base-classifier individually, which may be based on one or more of the following criteria: ranking (Woods *et al.*, 1997), local accuracy (Woods *et al.*, 1997; Smits, 2002; Ko *et al.*, 2007), F1-score (Melo Junior, Macedo, Nardini & Renso, 2019), probability (Giacinto & Roli, 1999; Woloszynski & Kurzynski, 2011), classifier behavior (Giacinto *et al.*, 2000), Oracle model (Ko *et al.*, 2007), data complexity (Brun *et al.*, 2016) and meta-learning (Cruz *et al.*, 2015a; Pinto *et al.*, 2016; Narassiguin *et al.*, 2017).

Group-based measures, on the other hand, estimate the competence for a group of classifiers in order to capture their interaction with regards to ambiguity (Dos Santos, Sabourin & Maupin, 2008) and diversity (Soares *et al.*, 2006), among other criteria.

1.1.3 Classifier selection

In the last step, the most competent classifier(s) is/are selected to label the query sample. In DCS techniques, the classifier with the highest competence level is usually selected, though in a few techniques that may only happen if its estimated competence is significantly greater than the other classifiers' (Giacinto *et al.*, 2000; Giacinto & Roli, 1999). On the other hand, in DES techniques a subset of the classifiers in the pool is also selected according to their competence level. In some cases, there may be a competence threshold for selection (Ko *et al.*, 2007; Davtalab *et al.*, 2022) where all classifiers with competence estimates above it are selected, while in others a pre-set number of classifiers is singled out based on a ranked list (Soares *et al.*, 2006).

Another approach that works essentially as dynamic selection in the last step is dynamic weighting, in which the classifiers' estimated competencies are used to compute the weights of their responses to be further used in the aggregation step. A few dynamic selection techniques, such as CHADE (Pinto *et al.*, 2016), apply this approach. However, it is not limited to this literature as other ensemble methods such as the Mixture of Experts (ME) (Jacobs *et al.*, 1991; Armano & Hatami, 2010b) and the Adaptive Splitting and Selection (AdaSS) (Jackowski & Wozniak, 2009; Lopez-Garcia, Masegosa, Osaba, Onieva & Perallos, 2019) paradigms also use the dynamic weighting mechanism though within a different ensemble framework.

1.2 The Oracle model

An important concept in the MCS literature is the Oracle, an abstract model that correctly labels a given test sample if at least one classifier in the pool is able to do so (Kuncheva, 2002).

The Oracle is of particular relevance to dynamic selection techniques as it mimics the perfect selection scheme since it is able to single out the correct classifier for each sample if it exists in the pool. As such, the Oracle accuracy rate is widely regarded as the upper limit for the accuracy of dynamic selection schemes and is used for indicating whether there is still room for improvement or not. However, the gap in performance between the Oracle and dynamic selection techniques observed in the literature (Cruz *et al.*, 2018a, 2015a; Oliveira *et al.*, 2017; Souza *et al.*, 2017) suggests that the Oracle may not be the best indicator of good performance for local-based techniques as there is no guarantee the classifier selected by the Oracle is indeed a local expert in the target region (Oliveira *et al.*, 2017; Souza *et al.*, 2017).

1.3 Local data characterization for dynamic selection

As previously discussed, most techniques rely on the locality assumption to solve the dynamic selection task. While this approach works generally well over a vast array of problems, e.g. class imbalanced distributions (Oliveira *et al.*, 2018; Cruz, Souza, Sabourin & Cavalcanti, 2019b), certain data characteristics were already shown to negatively impact the techniques' performance, such as local class overlap (Cruz *et al.*, 2019a, 2018b) and noise (Cruz *et al.*, 2015b). Furthermore, their RoC definition method may be susceptible to class overlap and data sparsity (Zhang, 2022; Vandaele *et al.*, 2022), the latter of which possibly leading to an increased complexity in the class boundary (Lorena *et al.*, 2012; Ho & Basu, 2002).

As such, certain dynamic selection techniques introduce data characterization into the pipeline to leverage the local information in order to tackle these challenging scenarios. In (Pereira *et al.*, 2018), the instances in the RoC are characterized using IRT coefficients, and the ones with the lowest discrimination indexes are removed with the purpose of improving the local distribution. The RoC is also characterized in (Oliveira *et al.*, 2017), where a dynamic ensemble pruning technique is applied in regions that present a local border, called indecision regions, to avoid the selection of non-local experts. A different approach with a similar goal is proposed in (Oliveira *et al.*, 2018), where the RoC is edited in the search for a local expert without completely removing the local border, characterized by pairs of different classes. In (Li, Wen, Li & Cai, 2019) the

RoC distribution is characterized using a must-link and a cannot-link graph to represent the local class relations and take them into account in the competence estimation step.

While these techniques incorporate some form of data characterization to deal with complex scenarios, they rely on a single pre-defined area of small scope which provides a limited view of the local distribution and may restrict the search for local experts, if they exist, in case the original region is not adequate to the dynamic selection task. Furthermore, albeit focused on borderline regions, these techniques do not address sparse distributions, which were shown to intensify the issues associated with local class overlap (Zhang, 2022; Vandaele *et al.*, 2022; Sánchez *et al.*, 2007) and class boundaries' complexity (Lorena *et al.*, 2012; Ho & Basu, 2002), and comprise several real-world problems such as medical imaging data (El-Sappagh, Saleh, Sahal, Abuhmed, Islam, Ali & Amer, 2021) and DNA microarray data (Lorena *et al.*, 2012), therefore limiting the techniques' application.

In the next chapter, an instance-based analysis of the behavior of three dynamic classifier selection techniques using a globally-generated pool is performed, and the Online Local Pool (OLP) technique is proposed to address the presence of local experts in class-overlapped target regions.

CHAPTER 2

ONLINE LOCAL POOL GENERATION FOR DYNAMIC CLASSIFIER SELECTION

Mariana A. Souza[1] , George D. C. Cavalcanti[2] , Rafael M. O. Cruz[1] , Robert Sabourin[1]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA),
École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Centro de Informática,
Universidade Federal de Pernambuco,
1235 Av. Prof. Moraes Rego, Recife, Pernambuco, Brazil 50670-901

Article published in Elsevier's Pattern Recognition Journal, Volume 85, January 2019

Abstract

Dynamic Classifier Selection (DCS) techniques have difficulty in selecting the most competent classifier in a pool, even when its presence is assured. Since the DCS techniques rely only on local data to estimate a classifier's competence, the manner in which the pool is generated could affect the choice of the best classifier for a given instance. That is, the global perspective in which pools are generated may not help the DCS techniques in selecting a competent classifier for instances that are likely to be misclassified. Thus, it is proposed in this work an online pool generation method that produces a locally accurate pool for test samples in difficult regions of the feature space. The difficulty of a given area is determined by the estimated classification difficulty of the instances in it. That way, by using classifiers that were generated in a local scope, it could be easier for the DCS techniques to select the best one for those instances they would most probably misclassify. For the query samples surrounded by easy instances, a simple nearest neighbors rule is used in the proposed method. In order to identify in which cases the local pool is used in the proposed scheme, an analysis on the correlation between instance hardness and DCS techniques is performed in this work, and it is proposed the use of an instance hardness measure that conveys the degree of local class overlap near a given sample. Experimental results show that the DCS techniques were more able to select the most competent classifier for

difficult instances when using the proposed local pool than when using a globally generated pool. Moreover, the proposed technique yielded significantly greater recognition rates in comparison to a Bagging-generated pool and two other global generation schemes for all DCS techniques evaluated. The performance of the proposed technique was also significantly superior to three state-of-the-art classification models and was statistically equivalent to five of them. [1]

2.1 Introduction

Multiple Classifier Systems (MCS) aim to improve the overall performance of a pattern recognition system by combining numerous base classifiers (Woźniak *et al.*, 2014; Kittler *et al.*, 1998; Kuncheva, 2014). An MCS contains three phases (Britto *et al.*, 2014): (1) Generation, (2) Selection and (3) Integration. In the first phase, a pool of classifiers is generated using the training data. In the second phase, a non-empty subset of classifiers from the pool is selected to perform the classification task. In the third and last phase, the selected classifiers' predictions are combined to form the final system's output. There are two possible approaches in the Selection phase: Static Selection (SS), in which the same set of classifiers is used to label all unknown instances, or Dynamic Selection (DS), which selects certain classifiers from the pool according to each query sample.

[1] The Online Local Pool (OLP) technique proposed in this work is an evolution of the technique proposed in (Souza, 2018). The improvements were motivated by an instance-level data complexity analysis performed afterward to assess the relationship between the performance of dynamic selection techniques and an instance hardness measure found in the literature. The OLP thus improves upon the initial idea in the previous work with the inclusion of local data complexity and instance characterization for the local overlap detection step, which is crucial to the effectiveness of the approach. As such, an offline phase is incorporated into the previous technique where the data is characterized on an instance level using a hardness measure that conveys the degree of local class overlap. During generalization (or the online phase of the OLP), the local class overlap detection is modified to rely on the data characterization previously obtained in the offline phase. This allows for a better detection of the local class overlap compared to the more limited class distribution analysis applied in the previous technique, as experimental results have shown. In addition to the instance hardness analysis and the subsequent proposal of the improved technique, an extensive comparative analysis which included not only the baseline ensembles evaluated in the previous work but also nine state-of-the-art models (among which monolithic classifiers and ensemble methods) was also included in the journal contribution to further validate the OLP's effectiveness over the evaluated problems.

The DS techniques, which have been shown to outperform static ensembles, specially on ill-defined problems (Britto *et al.*, 2014; Cruz *et al.*, 2015a), are based on the idea that the classifiers in the pool are individually competent in different regions of the feature space. The aim of the selection scheme is, then, to choose the classifier(s) that is(are) best fit, according to some criterion, for classifying each unknown instance in particular (Britto *et al.*, 2014). The amount of classifiers singled out to label a given sample separates the DS schemes in two groups (Cruz *et al.*, 2018a): Dynamic Classifier Selection (DCS) techniques, in which the classifier with highest estimated competence in the pool is selected, and Dynamic Ensemble Selection (DES) schemes, in which a locally accurate subset of classifiers from the pool is chosen and combined to label the test sample.

In the context of DCS, the Oracle (Kuncheva, 2002) can be defined as an abstract model that mimics the perfect selection scheme: it always selects the classifier that correctly labels a given instance, if the pool contains such classifier. Thus, the Oracle accuracy rate is the theoretical limit for DCS techniques.

The behavior of the Oracle regarding pool generation for DCS techniques was characterized in a previous work (Souza *et al.*, 2017). It was shown that even though the presence of one competent classifier was assured for a given instance, the DCS techniques still struggled to select it. This analysis was done using a pool generation method that guarantees an Oracle accuracy rate of 100% on the training set. It was reasoned that the nature of the Oracle makes it not very well suited to guide the generation of a pool of classifiers for DCS since the model is performed globally, while DCS techniques use only local data to select the most competent classifier for each instance. Thus, the difference in perspectives between the generation and the selection may hinder the DCS techniques in the selection of a competent classifier, even when the latter is guaranteed to be in the pool.

In addition to that, most works regarding DS use classical generation methods, which were designed for static ensembles (Cruz *et al.*, 2018a) and therefore do not take into account the regional aspect of the competence estimation performed by the DS techniques. Thus, since local information is not considered during the generation process, the presence of local experts is not guaranteed in the final pool.

Based on these observations, it is proposed in this work an online pool generation method which attempts to explore the Oracle's properties on a local scope. Since the Oracle and DCS techniques view the problem from different perspectives, using the Oracle model in a local setting to match these perspectives may help the DCS techniques in the choice of the most competent classifier for a given instance. This work focus only on DCS techniques since their relationship to the Oracle was already characterized in (Souza *et al.*, 2017), and so the results can be further analyzed based on certain aspects presented in the previous work.

Thus, the main idea is to use the Oracle model to guide the generation of a pool specialized on the local region where a given unknown sample is, if that region is deemed difficult. In this context, a region is considered *difficult* if it contains an instance likely to be misclassified, as indicated by an instance hardness measure. Therefore, if a query sample is located in a difficult region of the feature space, a local pool (LP) is generated on the fly so that its classifiers fully cover the surrounding area of that specific instance. Otherwise, a simple k-Nearest Neighbors (k-NN) rule is used to label the query sample, since the classification task is less complex. Hence, whenever an unknown sample is located in a difficult region, the proposed method uses Oracle information in that area to generate locally accurate classifiers for that instance, in hopes that the best classifier among them will be more easily selected by a DCS technique than if the classifiers were generated with a global perspective.

To the best of our knowledge, there is no ensemble method designed to generate local experts for dynamic selection techniques. However, a local learning algorithm with a similar strategy to that of the proposed technique is presented in (Bottou & Vapnik, 1992). The learning algorithm consists of generating a linear classifier for each unknown instance using its surrounding training samples, and then labelling that instance with it. This model was used to analyze the trade-off between capacity and locality of the learning algorithms and its impact on their recognition rates. Although the learning algorithm provides a local perspective on the classification problem, its concept was not used in the context of producing a pool of locally accurate classifiers for DS techniques.

Other related works, such as the Mixture of Random Prototype-based Local Experts (Armano & Hatami, 2010a) and the Forest of Local Trees (Armano & Tamponi, 2018) techniques, explore the divide-to-conquer approach of MCS by locally training their base classifiers in different regions of the feature space and weighting the classifiers' votes based on the distance between the query sample and their assigned region. As opposed to these works, in which the pool generation is paired to a selection based on dynamic distance weighting, our approach consists of producing on the fly a locally accurate pool to be coupled with a DCS technique. Furthermore, the generation process of these approaches do not guarantee the presence of local experts in the vicinity of each borderline unknown sample, as the proposed method does.

Thus, with our proposed approach, we aim to find out in this work whether the presence of locally generated pools is advantageous in DCS context. The research questions we intend to answer are: (1) does the use of locally generated pools aid the DCS techniques in selecting the best classifier for a given instance?, and (2) do the recognition rates improve as a result of this?. To that end, the performances of the proposed scheme and of different ensemble methods that yield globally-generated pools are assessed using DCS techniques over 20 public datasets, and the results compared and analyzed. A comparative study with several state-of-the-art classification models is also performed afterwards.

This work is organized as follows: in Section 2.2, an analysis on instance hardness for DCS techniques is performed in order to observe the correlation between an instance hardness measure and the mistakes made by these techniques. Once this relationship is established, Section 2.3 presents the proposed generation method, which explores the instance hardness information obtained in the previous section. In Section 2.4 the proposed method is evaluated, and it is analyzed whether the use of specialist subpools in difficult regions is beneficial for DCS techniques. A comparative study with state-of-the-art classification models is also performed in Section 2.4. Lastly, in Section 2.5 the results are summarized and future works are suggested.

2.2 Instance Hardness Analysis

Hardness is an aspect inherent to a problem that hinders a classifier, or a set of classifiers, in the classification task. *Instance hardness* is then a characteristic of a sample's problem that conveys the likelihood of such sample being mislabelled by a classifier (Smith, Martinez & Giraud-Carrier, 2014). Hardness measures attempt to quantify this characteristic based on different sources of difficulty associated with data, as well as provide insights as to why some instances or problems are difficult for most learning algorithms. Many data hardness measures were proposed and also used to improve a vast number of methods in the literature (Dong & Kothari, 2003; Smith & Martinez, 2016; Singh, 2003).

In (Ho & Basu, 2002), the authors introduce a set of hardness measures obtained over the entire training set. They also identify aspects that lead to a problem, as a whole, being difficult for a classifier. The authors in (Garcia, de Carvalho & Lorena, 2015) propose a set of hardness measures, also over the whole dataset, and use them, together with the ones from (Ho & Basu, 2002), to identify and remove noisy data in the training set.

In addition to characterizing the hardness of an entire set, efforts have been made to estimate the difficulty of classifying each individual instance from a dataset. In (Smith & Martinez, 2011), the authors propose several instance hardness measures and use a subset of them in the construction of noise filter which removes potentially noisy instances among the hard ones. A hardness analysis on an instance level was done in (Smith *et al.*, 2014), in which the authors identify the most influential causes for an instance to be hard for many diverse classification models. They also introduce more instance hardness measures and show the correlation between them and the misclassification of the classifiers analysed. Moreover, they suggest the integration of the error information of these classifiers in two different scenarios: in the training of a neural network, so that the weight of the hard instances are smaller, and in a noise filter, based on the same idea as the previous one. In (de Melo & Prudêncio, 2014), the authors propose two instance hardness measures that take into account misclassification costs. These measures are further used to define a measure of similarity between algorithms.

Although in (Smith & Martinez, 2011) a set of classifiers was used to evaluate the correlation between the hardness measures and the instance hardness itself, the authors did not investigate it for DCS techniques. Though an analysis on instance hardness regarding DS techniques was performed in (Cruz, Zakane, Sabourin & Cavalcanti, 2017b), its focus was on the comparison between these techniques and the k-NN classifier. Thus, an analysis on instance hardness in DCS context is done in this section. The purpose of such analysis is to understand the correlation between instance hardness measures and the errors made by DCS techniques, in order to identify in which cases the DCS techniques fail to choose a competent classifier for a given instance. This information will later be used to generate subpools specialized in the difficult regions of the training set.

The chosen instance hardness measure to be analyzed is the k-Disagreeing Neighbors (kDN) (Smith *et al.*, 2014), which is defined in Equation 2.1, where $\mathbf{x_i}$ is the instance being evaluated, $\mathcal{T}$ is the dataset that contains it, $kNN()$ is the k-Nearest Neighbors (k-NN) rule and k_h is the neighborhood size of the hardness measure. The kDN measure is the percentage of instances in an example's neighborhood that do not share the same label as itself. Therefore, a high kDN value means the instance is in a local overlap region, making it harder to label it. The reason for using this measure is because it denotes the most relevant source of instance hardness (overlap), according to (Smith *et al.*, 2014). Moreover, the kDN measure was the most correlated with instance hardness according to the same article.

$$kDN(\mathbf{x_i}, \mathcal{T}, k_h) = \frac{|\mathbf{x_j} : \mathbf{x_j} \in kNN(\mathbf{x_i}, \mathcal{T}, k_h) \wedge label(\mathbf{x_j}) \neq label(\mathbf{x_i})|}{k_h} \tag{2.1}$$

The pool generation technique used in this analysis was the Self-generating Hyperplanes (SGH) method (Souza *et al.*, 2017), a simple generation scheme which yielded a similar performance as Bagging (Breiman, 1996) for most DCS techniques. The SGH generation method is described in Algorithm 2.1. The input to the SGH method is only the training set $\mathcal{T}$, and its output is the generated pool of classifiers (C). In each iteration (Step 3 to Step 15), the centroids of all classes in $\mathcal{T}$ are obtained in Step 4 and stored in $\mathcal{R}$. The two centroids in $\mathcal{R}$ most distant from each other, $\mathbf{r_i}$ and $\mathbf{r_j}$, are selected in Step 5. Then, a hyperplane c_m is placed between $\mathbf{r_i}$ and $\mathbf{r_j}$, dividing both points halfway from each other. The two-class linear classifier c_m is then tested over the training set, and the instances it correctly labels are removed from $\mathcal{T}$ (Step 7 to Step 12). Then, c_m is added to C in Step 13, and the loop is repeated until $\mathcal{T}$ is completely empty. That is, the SGH method only stops generating hyperplanes when all training instances are correctly labelled by at least one classifier in C, i.e., the Oracle accuracy rate for the training set is 100%.

Algorithm 2.1 General procedure of the Self-generating Hyperplanes (SGH) method.

```
   input  : $\mathcal{T} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}\}$ ;          ▹ Training dataset
   output : $C$ ;                                     ▹ Final pool
1  $C \leftarrow \{\}$ ;                                  ▹ Pool initially empty
2  $m \leftarrow 1$ ;                                   ▹ Classifier count
3  while $\mathcal{T} \neq \{\}$ do
4      $\mathcal{R} \leftarrow getCentroids(\mathcal{T})$;                ▹ Calculate each class' centroid
5      $\mathbf{r_i}, \mathbf{r_j} \leftarrow selectCentroids(\mathcal{R})$;        ▹ Select the most distant centroids
6      $c_m \leftarrow placeHyperplane(\mathbf{r_i}, \mathbf{r_j})$;  ▹ Generate hyperplane between centroids $\mathbf{r_i}$ and $\mathbf{r_j}$
7      for every $\mathbf{x_n}$ in $\mathcal{T}$ do
8          $\omega \leftarrow c_m(\mathbf{x_n})$ ;                   ▹ Test $c_m$ over training instance
9          if $\omega = y_n$ then
10             $\mathcal{T} \leftarrow \mathcal{T} - \{\mathbf{x_n}\}$ ;      ▹ Remove from $\mathcal{T}$ correctly classified instance
11         end if
12     end for
13     $C \leftarrow C \cup \{c_m\}$;                        ▹ Add $c_m$ to pool
14     $m \leftarrow m + 1$
15 end while
16 return $C$
```

In order to evaluate the correlation between the instance hardness measures and the accuracy of the DCS techniques, the hardness of each instance was computed using the entire dataset, thereby obtaining the *true* hardness value of each instance. Afterwards, the accuracy of the DCS techniques was obtained using 10 times 10-fold cross validation. The previous knowledge regarding each instance's hardness is then used to draw a relationship between this measure and the frequency at which the DCS techniques misclassifies it. That way, an evaluation of the relationship between the kDN measure and the error rate of the DCS techniques can be performed.

The datasets used in this analysis are shown in Table 2.1. All of them are public datasets. Eleven from the UCI machine learning repository (Bache & Lichman, 2013), three from the Ludmila Kuncheva Collection (Kuncheva, 2004) of real medical data, three from the STATLOG project (King, Feng & Sutherland, 1995), two from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository (Alcalá, Fernández, Luengo, Derrac, García, Sánchez & Herrera, 2011) and one from the Enhanced Learning for Evolutive Neural Architectures (ELENA) project (Jutten, 2002). The DCS techniques used in this analysis were Overall Local Accuracy (OLA) (Woods *et al.*, 1997) and Local Class Accuracy (LCA) (Woods *et al.*, 1997), which were the two best performing DCS techniques in a recent survey on dynamic selection of classifiers (Cruz *et al.*, 2018a). For simplicity, the neighborhood sizes of both the kDN measure and the DCS techniques were set to $k_h = k_s = 7$.

Table 2.1 Main characteristics of the datasets used in the experiments.

Dataset	No. of samples	No. of features	No. of classes	Class sizes	Source
Adult	48842	14	2	383;307	UCI
Blood Transfusion	748	4	2	570;178	UCI
Cardiotocography (CTG)	2126	21	3	1655;295;176	UCI
Steel Plate Faults	1941	27	7	158;190;391;72;55;402;673	UCI
German credit	1000	20	2	700;300	STATLOG
Glass	214	9	6	70;76;17;13;9;29	UCI
Haberman's Survival	306	3	2	225;81	UCI
Heart	270	13	2	150;120	STATLOG
Ionosphere	315	34	2	126;225	UCI
Laryngeal1	213	16	2	81;132	LKC
Laryngeal3	353	16	3	53;218;82	LKC
Liver Disorders	345	6	2	145;200	UCI
Mammographic	961	5	2	427;403	KEEL
Monk2	4322	6	2	204;228	KEEL
Phoneme	5404	6	2	3818;1586	ELENA
Pima	768	8	2	500;268	UCI
Sonar	208	60	2	97;111	UCI
Vehicle	846	18	4	199;212;217;218	STATLOG
Vertebral Column	310	6	2	204;96	UCI
Weaning	302	17	2	151;151	LKC

In order to characterize the relationship between the kDN measure and the DCS techniques, all samples from each dataset were grouped by their true hardness value and the mean accuracy rate of both DCS techniques on each group was calculated. The results are summarized in Figure 2.1.

It can be observed in Figure 2.1 that the two DCS techniques misclassify the majority of all instances with kDN above 0.5, on average. It can also be seen a great difference between the accuracy rates of instances with kDN $\in [0.4, 0.5]$ and kDN $\in [0.5, 0.6]$, which is reasonable

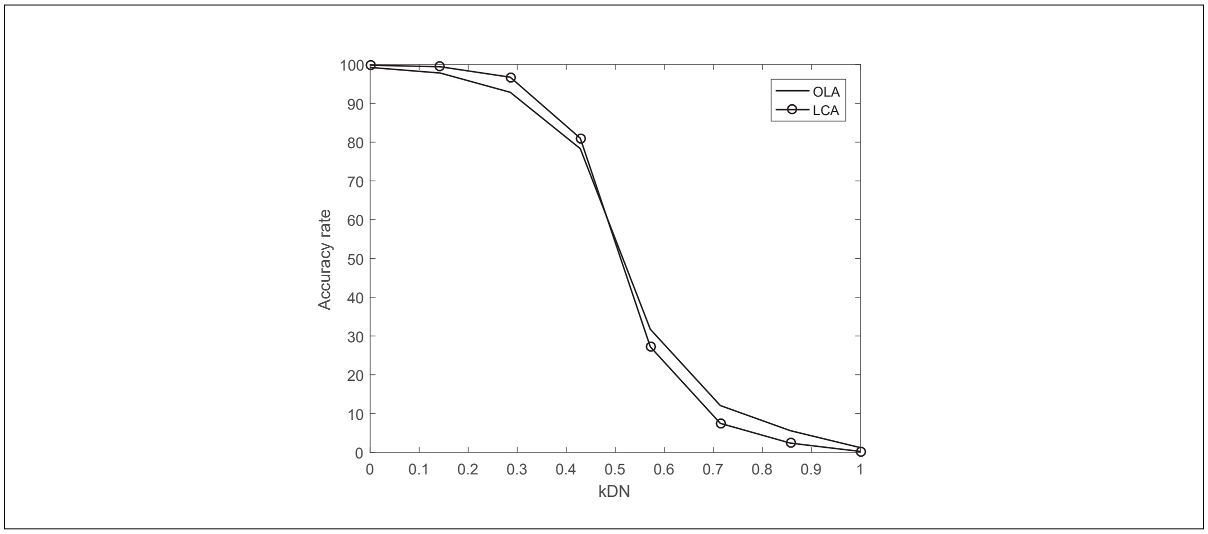


Figure 2.1 Mean accuracy rate of OLA and LCA for each group of kDN value, for all datasets from Table 2.1. The neighborhood sizes of the DCS techniques and the kDN measure are $k_s = k_h = 7$.

since kDN values above 0.5 mean the majority of an instance's neighbors belong to a different class than its own class. This result shows the correlation between the kDN measure and the classification difficulty by the DCS techniques, which was somewhat expected since the measure and the techniques operate locally using the same mechanism of selecting the k nearest neighbors. Moreover, Figure 2.1 shows that LCA correctly classifies a greater amount of the easiest instances ($kDN \leq 0.5$) than OLA, though it struggles more to correctly label the hardest instances ($kDN \geq 0.5$), on average.

2.3 The Proposed Method

In the previous section, it was shown the DCS techniques struggle to select a competent classifier for instances in regions with overlap between the problem's classes. Moreover, since the DCS techniques rely only on a small region, an instance's neighborhood, in order to select the most competent classifier for this instance, a global perspective in the search for a promising pool for DCS could be inadequate in such cases (Souza *et al.*, 2017).

With that in mind, it is proposed the use of an Oracle-guided generation method on a local scope, so that the model's properties may be explored by the DCS techniques. The idea is to use a local pool (LP) consisted of specialized classifiers, each of which selected using a DCS technique from a local subpool that contains at least one competent classifier for each instance in class overlap regions of the feature space. If the unknown instance's Region of Competence (RoC) is located in a difficult region, the LP is generated on the fly using its neighboring instances and then used to label the query sample. However, if the query instance is far from the classes' borders, no pool is generated and the output label is obtained using a simple nearest neighbors rule.

The reasoning behind the proposed approach is that using locally generated classifiers for instances in class overlap areas may be of help to the DCS techniques due to their high accuracy in these regions. Moreover, most works regarding DCS use classical generation methods, which were designed for static ensembles (Cruz *et al.*, 2018a) and therefore do not take into account the regional aspect of the competence estimation performed by the DS techniques. Thus, matching the perspectives of the generation and the selection stages may be advantageous for these techniques.

An overview of the proposed method is described in more detail in Section 2.3.1. Then, a step-by-step analysis of the proposed method is presented using a 2D toy problem in Section 2.3.2.

2.3.1 Overview

The proposed technique is divided into two phases:

1. The *offline* phase, in which the hardness estimation of the training instances is performed. The hardness value of the training samples is used to identify the difficult regions of the feature space.
2. The *online* phase, in which the RoC of the query sample is evaluated using a hardness measure in order to identify if the local area is difficult. If it is not, the sample is labelled

using a nearest neighbors rule. Otherwise, a pool composed of the most locally accurate classifiers in that region, as indicated by a DCS technique, is generated and used to label the unknown instance.

An overview of the proposed techniques' phases is depicted in Figure 2.2, in which $\mathcal{T}$ is the training set, H is the set of hardness estimates, $\mathbf{x_q}$ is the query sample, θ_q is its RoC, k_s is the size of θ_q, LP is the local pool, M is the pool size of LP and ω_l is the output label of $\mathbf{x_q}$. In the offline phase (Figure 2.2a), the hardness of each instance $\mathbf{x_n} \in \mathcal{T}$ is estimated using a hardness measure, and its value is stored in H, to be later used in the online phase. The online phase, in which the query sample $\mathbf{x_q}$ is labelled, is performed in three steps: RoC evaluation, local pool generation and generalization, as shown in Figure 2.2b.

In the RoC evaluation step, the k_s nearest neighbors in the training set $\mathcal{T}$ of the query sample $\mathbf{x_q}$ are selected to form the query sample's RoC θ_q. The dynamic selection dataset (DSEL), which is a set of labelled samples used for RoC definition in DS techniques (Cruz *et al.*, 2018a), was not used in the proposed method because the SGH method did not present overfitting when used for RoC definition (Souza *et al.*, 2017). Then, the instances that compose θ_q are analyzed. If all of them are not in an overlap region, that is, they all have hardness estimate $H = 0$, the method skips the local pool generation and goes directly to the generalization phase. The output class ω_l of $\mathbf{x_q}$ is then obtained using the k-NN rule with parameter k_s. However, if there is at least one instance in θ_q located in an identified class overlap area, the query sample's RoC is considered to be a difficult region. Thus, the local pool LP containing M classifiers is generated in the second step and used to label $\mathbf{x_q}$ in the generalization step via majority voting. The local pool generation step is explained next.

Figure 2.3 shows the generation procedure of the local pool LP. The pool size M of the local pool is an input parameter. The other inputs are the training set ($\mathcal{T}$), the query sample ($\mathbf{x_q}$) and the query sample's RoC size (k_s). The LP is constructed iteratively. In the m-th iteration, the query sample's neighboring instances in the training set are obtained using any nearest neighbors method, with parameter k_m. These neighboring instances form the query sample's neighborhood

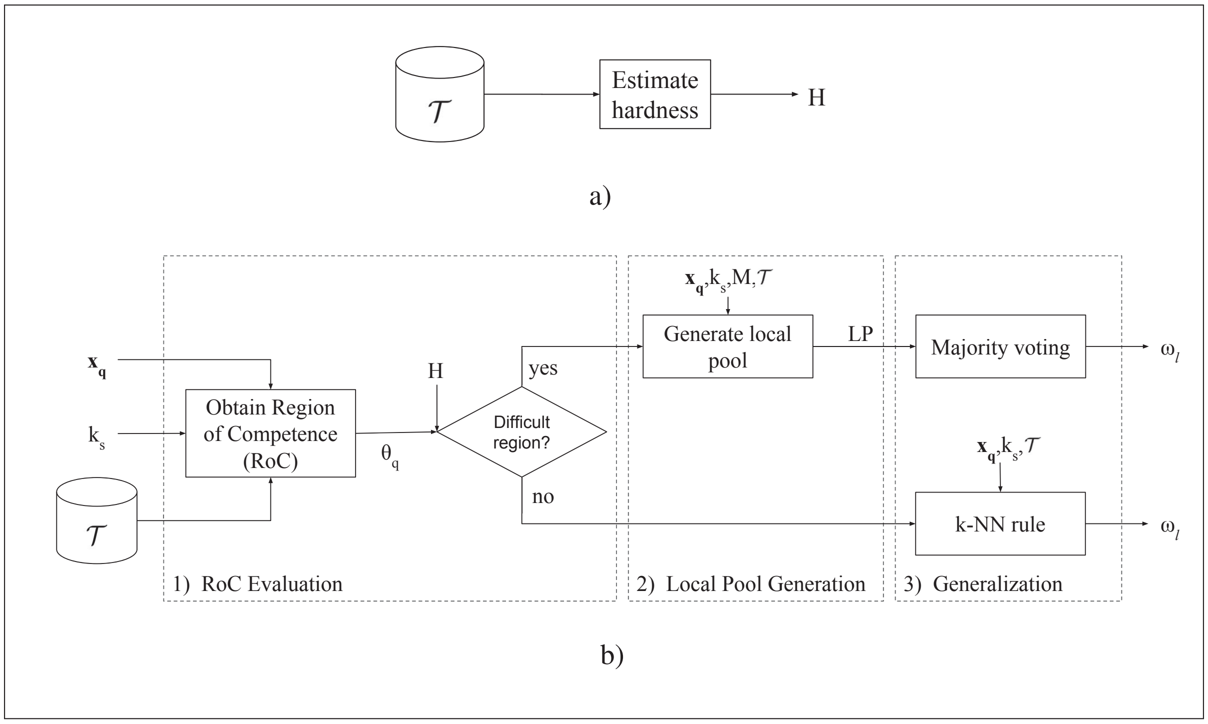


Figure 2.2 Overview of the (a) offline and (b) online phases of the proposed method. $\mathcal{T}$ is the training set, H is the set of hardness estimates, $\mathbf{x_q}$ is the query sample, θ_q is its Region of Competence (RoC), k_s is the size of θ_q, LP is the local pool, M is the pool size of LP and ω_l is the output label of $\mathbf{x_q}$. In the offline phase, the hardness value of all instances in $\mathcal{T}$ is estimated and stored in H. In the online phase, θ_q is first obtained and evaluated based on the hardness values in H. If it only contains easy instances, the k-NN rule is used to label $\mathbf{x_q}$ in the last step. Otherwise, the local pool is generated in the second step, and $\mathbf{x_q}$ is labelled via majority voting of the classifiers in LP in the third step.

θ_m, which is used as input to the SGH method (Algorithm 2.1 from Section 2.2). The SGH method then returns a local subpool C_m that fully covers the neighborhood θ_m. That is, the presence of at least one competent classifier $c_{m,k} \in C_m$ for each instance in θ_m is guaranteed. The indexes in the classifiers' notation indicates that the classifier $c_{m,k}$ is the k-th classifier from the m-th subpool (C_m). Then, the most competent classifier $c_{m,n}$ from C_m in the region delimited by the neighborhood θ_q is selected by a DCS technique and added to the local pool. The same procedure is performed in iteration $m + 1$ with the neighborhood size k_{m+1} increased by 2. This process is then repeated until the local pool contains M locally accurate classifiers.

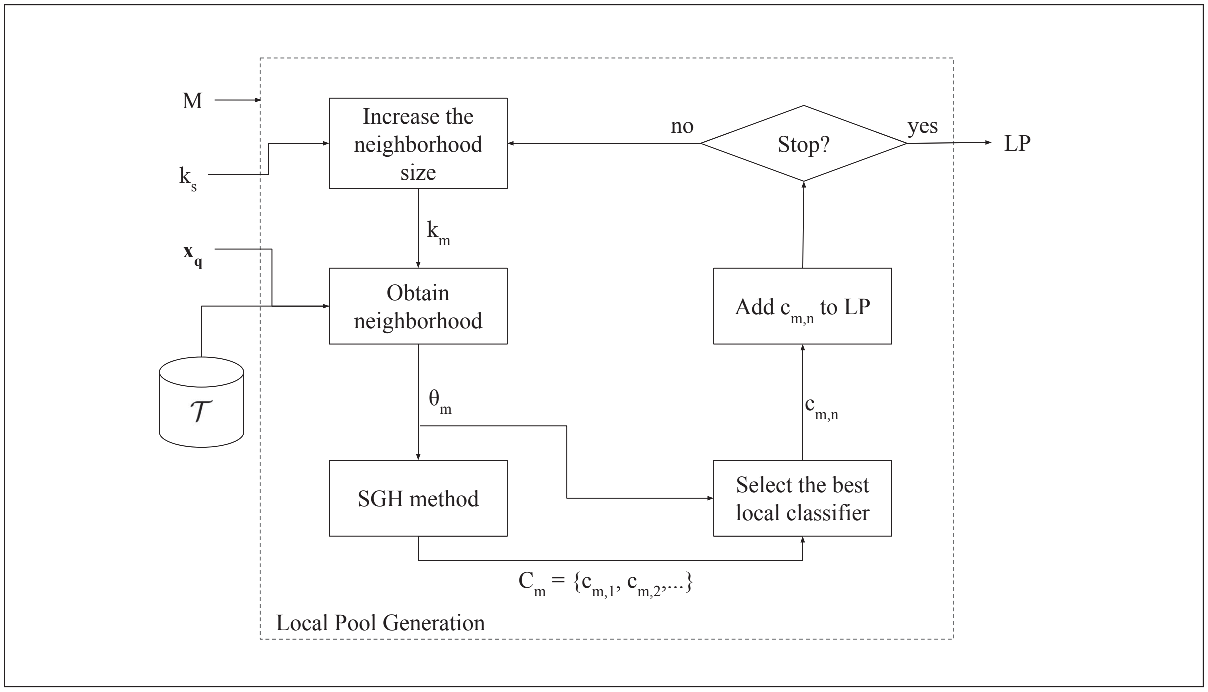


Figure 2.3 Local pool generation step. The inputs to the generation scheme are the training set $\mathcal{T}$, the query sample $\mathbf{x_q}$, the size k_s of the query sample's RoC and the local pool size M. The output is the local pool LP. In the m-th iteration, the query sample's neighborhood θ_m of size k_m is obtained and used as input to the SGH method, which yields the subpool C_m. The classifiers from C_m are then evaluated over θ_m using a DCS technique. The classifiers' notation refers a classifier $c_{m,k}$ as the k-th classifier from the m-th subpool (C_m). The most competent classifier $c_{m,n}$ in subpool C_m is then selected and added to the local pool LP. This process is then repeated until LP contains M locally accurate classifiers.

Algorithm 2.2 Online Local Pool (OLP) technique: offline phase.

```
   input  : $\mathcal{T}, k_h$;                          ▹ Training dataset and kDN neighborhood size
   output : $H$;                                         ▹ Estimated hardness values
1  for every $\mathbf{x_i}$ in $\mathcal{T}$ do
2  |   $H(i) \leftarrow kDN(\mathbf{x_i}, \mathcal{T}, k_h)$ ;        ▹ Calculate hardness (Equation 2.1)
3  end for
4  return $H$
```

The pseudocode of the offline phase of the proposed method is shown in Algorithm 2.2. Its inputs are the training set $\mathcal{T}$ and the kDN parameter k_h, which denotes the neighborhood size of the hardness estimate. From Step 1 to Step 3, the hardness of each instance $\mathbf{x_i} \in \mathcal{T}$ is calculated and stored in H, which is then returned in Step 4.

Algorithm 2.3 Online Local Pool (OLP) technique: online phase.

```
   input  : $\mathbf{x_q}, \mathcal{T}, H$ ;                        ▹ Query sample, training set and hardness estimates
   input  : $k_s, M$ ;                                              ▹ RoC size and pool size of local pool $LP$
   output : $\omega_l$ ;                                            ▹ Output label of $\mathbf{x_q}$
1  $\theta_q \leftarrow obtainRoC(\mathbf{x_q}, k_s, \mathcal{T})$;          } 1) RoC        ▹ Obtain the query instance's RoC
2  if $\{\exists \mathbf{x_i} \in \theta_q | H(i) > 0\}$ then                } Evaluation
3      $LP \leftarrow \{\}$;                                                          ▹ Local pool initially empty
4      for every $m$ in $\{1, 2, ..., M\}$ do                       }
5          $k_m \leftarrow k_s + 2 \times (m-1)$ ;                   }                ▹ Increase neighborhood size by 2
6          $\theta_m \leftarrow obtainNeighborhood(\mathbf{x_q}, k_m, \mathcal{T})$ } 2) Local Pool ▹ Obtain neighborhood of $\mathbf{x_q}$
7          $C_m \leftarrow generatePool(\theta_m)$;                  } Generation     ▹ Generate local subpool $C_m$
8          $c_{m,n} \leftarrow selectClassifier(\mathbf{x_q}, \theta_m, C_m)$ ; }         ▹ Select best classifier in $C_m$
9          $LP \leftarrow LP \cup \{c_{m,n}\}$;                      }                ▹ Add $c_{m,n}$ to $LP$
10     end for
11     $\omega_l \leftarrow majorityVoting(\mathbf{x_q}, LP)$ ;     } 3) Gener-      ▹ Label $\mathbf{x_i}$ with majority voting on $LP$
12 else                                                          } alization
13     $\omega_l \leftarrow kNN(\mathbf{x_q}, k_s, \mathcal{T})$ ;  }                ▹ Label query sample using k-NN rule
14 end if
15 return $\omega_l$
```

The online phase, on the other hand, is described in more detail in Algorithm 2.3. Its inputs are the query sample $\mathbf{x_q}$, training set $\mathcal{T}$, the set of hardness estimates H, the RoC size k_s and the local pool size M. In Step 1, the query sample's RoC θ_q is obtained by selecting the k_s closest samples to $\mathbf{x_q}$ in the training set. The RoC is then evaluated in Step 2. If all instances in θ_q are not located in a difficult region, that is, their hardness value is zero, the method goes straight to Step 13 and the query sample's output label ω_l is obtained using the k-NN rule with parameter k_s and returned in Step 15.

However, if there is one instance $\mathbf{x_i}$ from θ_q whose hardness estimate $H(i)$ is above zero, the region is considered a difficult one and the method proceeds to Step 3. Each classifier in the local pool LP is obtained in the loop that iterates M times (Step 4 to Step 10). In each iteration, the neighborhood size k_m is calculated in Step 5. Then, the query sample's neighborhood θ_m is obtained using a nearest neighbors method in Step 6. The subpool C_m is then generated in Step 7 using the SGH method with θ_m as training set. In Step 8, a DCS technique is then used to select the most competent classifier $c_{m,n}$ in C_m. The classifier $c_{m,n}$ is added to LP in Step 9, and then the loop continues until the local pool is complete. Finally, the query sample's label ω_l is obtained using majority voting over the locally accurate classifiers in LP and returned in Step 15.

2.3.2 Step-by-step Analysis

In order to better understand the generation process by the proposed technique, the latter was executed over a 2D toy problem dataset. The P2 Problem (Valentini, 2005) was chosen for its complex borders. Since the P2 problem has no overlap between the classes, noise was added to the original distribution by randomly changing the labels of the samples near the class borders. The dataset used in this analysis contains 1000 instances, 75% of which were used for training and the rest for test. The parameters used in this demonstration were $k_h = k_s = 7$ and $M = 7$. The method used for selecting the query instance's neighborhood in $getNeighborhood()$ (Step 6 of Algorithm 2.3) for this example was the regular k-NN, and the DCS technique used to select the most competent classifier (Step 8 of Algorithm 2.3) was OLA.

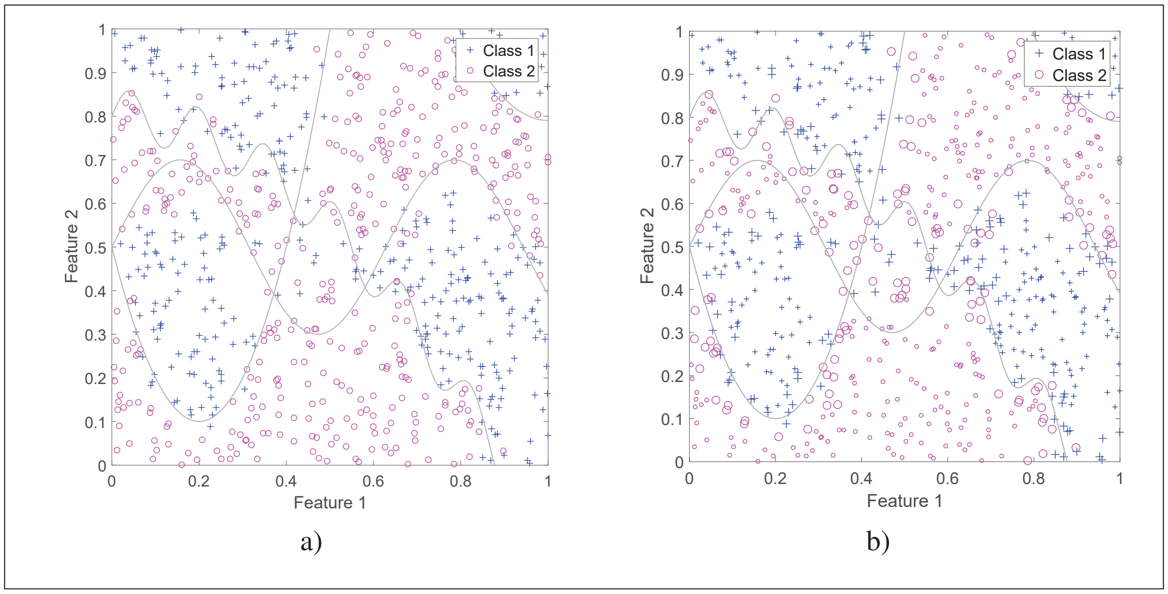


Figure 2.4 P2 Problem dataset, with theoretical decision boundaries in grey. The training set is depicted in (a), and in (b) the same set is shown with hard instances in large markers and easy instances in small ones.

The P2 Problem training set used in this analysis is shown in Figure 2.4a, with its theoretical decision boundaries in grey. The hardness estimation in Step 1 to Step 3 of Algorithm 2.2 separates the training instances with estimated hardness above zero, that is, the instances closer to the border between classes, and the remaining ones. Figure 2.4b shows the training instances closer to the borders (large markers) and the instances with $H(i) = 0$ (small markers).

Two scenarios of the proposed scheme's online phase can be observed in Figure 2.5a and Figure 2.5b. In the first, the input query instance $\mathbf{x_q}$ of Algorithm 2.3 belongs to Class 2. The query sample's RoC θ_q is obtained selecting its k-nearest neighbors over the training set $\mathcal{T}$ in Step 1. In this case, since all instances in θ_q have estimated hardness $H_i = 0$, represented in Figure 2.5a by small markers, $\mathbf{x_q}$ is considered to be in an easy region of the feature space. Therefore, the procedure goes to Step 13, in order to obtain the output label ω_l of $\mathbf{x_q}$ using the k-NN rule over the training set with parameter k_s. Then, the query sample's label is returned in Step 15. In this case $\omega_l = 2$ since all k_s neighbors of $\mathbf{x_q}$ belong to this class.

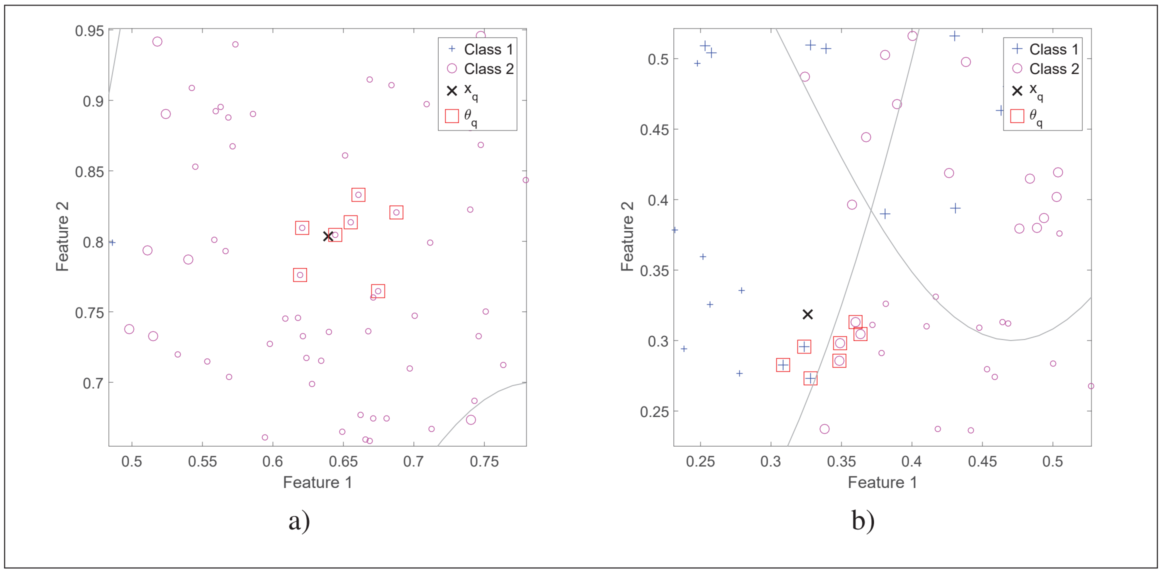


Figure 2.5 Two different scenarios of the online phase. In (a), the query instance $\mathbf{x_q}$ belongs to Class 2. Since all instances in its neighborhood θ_q are easy (small markers), the k-NN rule is used to label $\mathbf{x_q}$. On the other hand, all instances in the query sample's neighborhood θ_q in (b) are deemed hard (large markers). Thus, the local pool LP will label the query instance $\mathbf{x_q}$, which belongs to Class 1.

In the second scenario, shown in Figure 2.5b, the query instance $\mathbf{x_q}$ of Algorithm 2.3 belongs to Class 1. Its RoC θ_q is obtained in Step 1, with more than half of its instances belonging to the opposite class. Thus, a simple k-NN rule would misclassify this query sample. The query instance's RoC θ_q is then analyzed in Step 2. The hardness estimate H_i of each neighbor is verified in Step 2 of Algorithm 2.3, and since at least one of them is above zero, depicted in Figure 2.5b in large markers, the local pool (LP) will be generated and used from this step forward. Starting with an empty set (Step 3), each iteration from Step 4 to Step 10 adds a single classifier to LP.

In the first iteration, the neighborhood size k_1 is set to 7 in Step 5, and then the k_1 nearest neighbors of $\mathbf{x_q}$ are selected to compose the query sample's neighborhood θ_1 in Step 6. The local subpool C_1 is then generated using θ_1 as the input dataset to the SGH method. The resulting pool, which guarantees an Oracle accuracy rate of 100% in θ_1, is shown in Figure 2.6a, containing only one classifier, $c_{1,1}$. Since there is only one classifier in C_1, $c_{1,1}$ is selected to compose LP in Step 8 and Step 9.

In the second iteration, the neighborhood parameter is increased by 2 in Step 5, and the resulting neighborhood θ_2 contains $k_2 = 9$ instances, as shown in Figure 2.6b. Then, the local subpool C_2 is generated in Step 7, with θ_2 as the input parameter of the SGH method. Since only one classifier was able to deliver an Oracle accuracy rate of 100% over θ_2, the resulting pool contains only $c_{2,1}$, which is selected in Step 8 to be added to LP in Step 9.

The neighborhood θ_3, obtained in Step 6 of the third iteration, contains $k_3 = 11$ instances, as Figure 2.6c shows. C_3 is then generated in Step 7 so that it fully covers θ_3, resulting in only one classifier ($c_{3,1}$), which is later added to LP in Step 9.

The fourth local subpool C_4, depicted in Figure 2.6d, is generated in Step 7 of the fourth iteration, with neighborhood θ_4 of size $k_4 = 13$ as input to the SGH method. The only classifier generated, $c_{4,1}$, is then added to LP in Step 9.

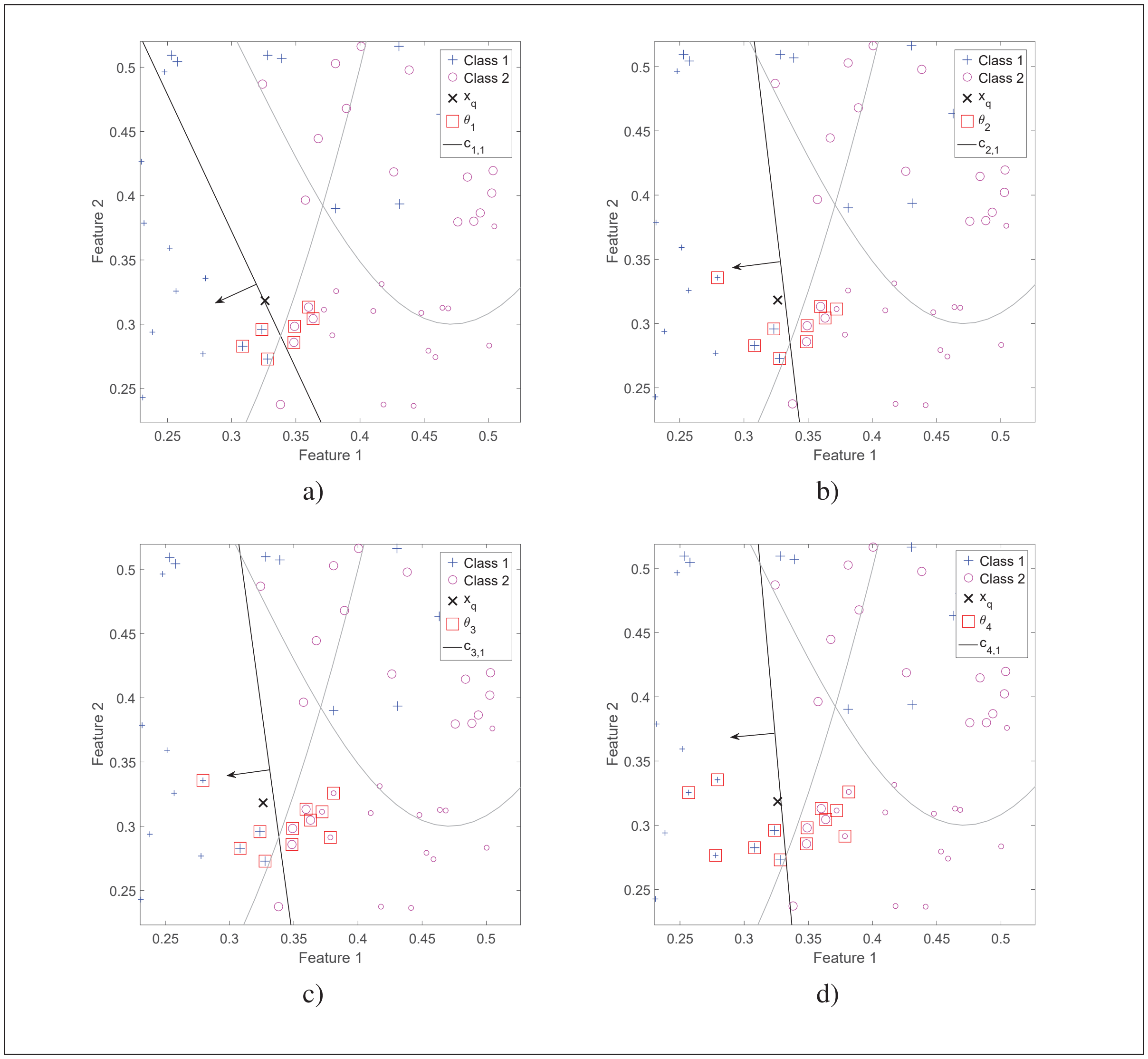


Figure 2.6 Local pool generation. (a) First, (b) second, (c) third, (d) fourth, (e) fifth, (f) sixth and (g) seventh iteration of the method, with its respective neighborhoods (θ_m) and generated local subpools C_m formed by the depicted classifiers ($c_{m,k}$). The arrows indicate in which part of the feature space the classifiers label as Class 1. Each local subpool C_m is obtained using the SGH method with its respective neighborhood θ_m, which increases in each iteration, as input. The final local pool LP, formed by the best classifiers in each subpool C_m, is shown in (h).

In the fifth iteration, the neighborhood θ_5 is obtained with parameter $k_5 = 15$ in Step 6. In Step 7, the SGH method yields the local subpool C_5, depicted in Figure 2.6e. Afterwards, the single classifier $c_{5,1}$ in C_5 is added to LP.

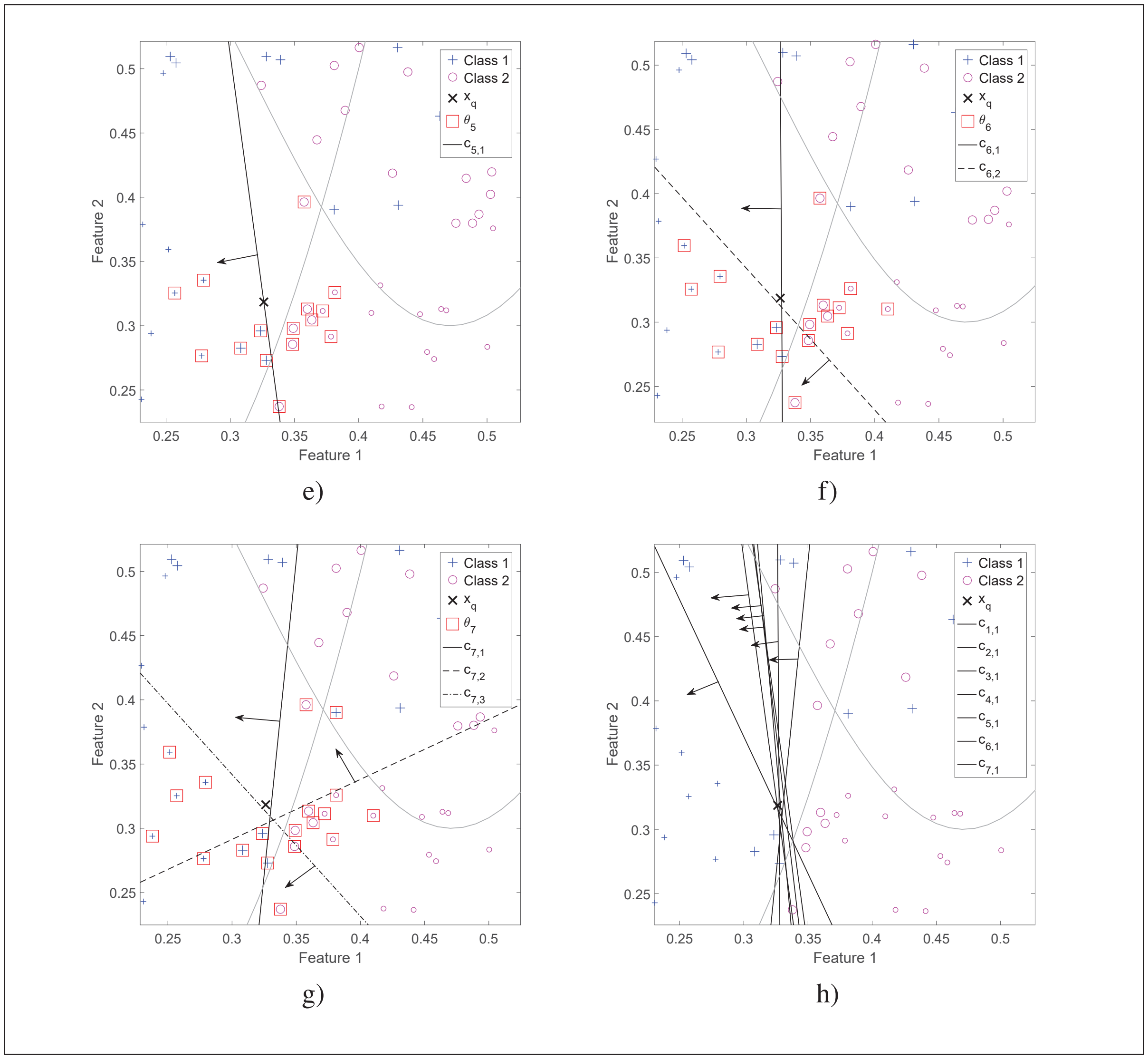


Figure 2.6 Local pool generation. (a) First, (b) second, (c) third, (d) fourth, (e) fifth, (f) sixth and (g) seventh iteration of the method, with its respective neighborhoods (θ_m) and generated local subpools C_m formed by the depicted classifiers ($c_{m,k}$). The arrows indicate in which part of the feature space the classifiers label as Class 1. Each local subpool C_m is obtained using the SGH method with its respective neighborhood θ_m, which increases in each iteration, as input. The final local pool LP, formed by the best classifiers in each subpool C_m, is shown in (h).

The neighborhood θ_6 of the sixth iteration is obtained with $k_6 = 17$ in Step 6. Then, the local subpool C_6 is generated in Step 7, resulting in two classifiers, $c_{6,1}$ and $c_{6,2}$, as shown in Figure

2.6f. In Step 8, both classifiers are evaluated over θ_6 using a DCS technique, OLA in this case. The most accurate one ($c_{6,1}$) in C_6 is returned and added to LP in Step 9.

Table 2.2 Majority voting of the classifiers from LP for the query instance from Figure 2.5b.

	$c_{1,1}$	$c_{2,1}$	$c_{3,1}$	$c_{4,1}$	$c_{5,1}$	$c_{6,1}$	$c_{7,1}$	**Total**
Class 1		x	x	x		x	x	5
Class 2	x				x			2

In the last iteration, the local subpool C_7 is generated in Step 7 using the neighborhood θ_7 with $k_7 = 19$ instances. Then, the local subpool C_7 is generated, yielding three classifiers that fully cover θ_7. Each classifier in C_7, shown in Figure 2.6g, is then evaluated using OLA, and the one that performs best over θ_7 is selected. The selected classifier, $c_{7,1}$ in this case, is then added to the local pool, completing the generation process of LP, depicted in Figure 2.6h.

After the generation process of the local pool, each classifier in it labels the query instance $\mathbf{x_q}$, and the final label is obtained by majority vote in Step 11. Table 2.2 shows the vote of each classifier in LP. The final label returned in Step 11 by the local pool is $\omega_l = 1$, which is the true class of $\mathbf{x_q}$.

2.4 Experiments

In order to analyse and evaluate the performance of the proposed method, experiments were conducted over the 20 datasets described in Table 2.1. All methods in the comparative study were evaluated using 20 replications of each dataset. For the configurations that used pools generated by the SGH method, each replication was randomly split into two parts: 75% for training and 25% for test. Since the SGH method did not present overfitting, both in global (Souza *et al.*, 2017) and local scope, the training set was used as the DSEL set. In the comparative study, however, the methods that use a DS technique were tested using a pool of 100 Perceptrons obtained using Bagging (Breiman, 1996), as it is often done in DS works (Cruz *et al.*, 2015a).

For these configurations, the validation set was used as the DSEL in order to avoid overfitting, so one third of the training set was randomly selected to compose the DSEL set.

This section is organized as follows. A comparative study with regards to DCS techniques is performed in Section 2.4.1, with the purpose of analyzing whether the use of locally generated pools is in fact advantageous in this context. In Section 2.4.2, the performances of the proposed method and state-of-the-art models, including single models, static ensembles and DS techniques, are also compared and analyzed. Lastly, the computational complexity of the proposed method and the compared models are discussed in Section 2.4.3.

2.4.1 Comparison with DCS techniques

In this section, an experimental analysis on the proposed method is performed. The aim of these experiments is to observe whether the DCS techniques are more prone to selecting the best classifier in the pool when said pool is generated locally and whether the use of such pools increase classification rates, in comparison to globally generated pools.

The DCS techniques chosen to evaluate the methods in these experiments were OLA, LCA and MCB, since they outperformed the other evaluated DCS techniques in (Cruz *et al.*, 2018a). The RoC size k_s for each of the DCS techniques is set to 7, since it yielded the best results in (Britto *et al.*, 2014).

The parameters of the proposed method were set to $k_s = k_h = 7$ and $M = 7$. Moreover, the proposed scheme was tested with two neighborhood acquisition methods: the LP configuration and the LP^e configuration. In the first, the $getNeighborhood()$ method from Algorithm 2.3 (Step 6) used was the regular k-NN. In the second configuration, the $getNeighborhood()$ procedure used was a version of the k-Nearest Neighbor Equality (k-NNE) (Sierra *et al.*, 2011) in which the returned neighborhood contains an equal amount of instances from all classes, given that these classes are present in the query instance's RoC θ_q from Algorithm 2.3 (Step 1).

The performance of the proposed method with regards to the DCS techniques is compared to three globally generated pool configurations. The baseline method used in the comparison is a Bagging-generated pool composed of 100 classifiers. The SGH method over the entire training set is also included in the comparative study, since it provides another global approach for generating classifiers. The pool generated by this technique is referenced as the global pool (GP). Lastly, another related method, though it is not a generation one, is used in the comparison with DCS techniques: the Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES) framework (Oliveira *et al.*, 2017).

In the FIRE-DES framework, when a query sample is in an *indecision region*, that is, a neighborhood that contains more than one class, the classifiers that correctly label instances from different classes in the query sample's RoC are pre-selected to form the pool used in the DS technique. That is, if a border is detected in the query sample's RoC, the selection scheme searches only among the classifiers that cross this border. This is performed using the Dynamic Frienemy Pruning (DFP), an online pruning method for DS techniques. The FIRE-DES framework is designed for two-class problems, and it obtained a significant increase in accuracy for most DS techniques, specially for highly imbalanced datasets, in which cases the DFP method provided a considerable improvement in performance for those techniques.

In the FIRE-DES context, an unknown sample in an indecision region has, by definition, a hardness value greater than zero, since at least one of its neighbors belongs to a different class, regardless of its label. In the proposed method, such an instance is labelled using the local pool, which is guaranteed to contain classifiers that cross the query sample's RoC due to its generation procedure (Figure 2.3). Thus, the same idea of using only locally accurate classifiers for instances in overlap regions from the FIRE-DES framework indirectly applies to the proposed method as well. Therefore, the FIRE-DES framework, coupled with the chosen DCS techniques, is also included in the comparative study that follows. The pool used by this framework in the experiments is the same as the one from the Bagging configuration, which contains 100 globally generated classifiers.

The performance of the chosen configurations with regards to DCS techniques is evaluated in memorization, using the hit rate measure, in Section 2.4.1.1, and in generalization, using the accuracy rate over the datasets from Table 2.1, in Section 2.4.1.2.

2.4.1.1 Performance in Memorization

The proposed method was evaluated in memorization using the hit rate (Souza *et al.*, 2017), which is a metric derived from the SGH method that indicates how well the generated pool integrates with the DCS techniques. In the SGH method, since the Oracle accuracy rate over the training set is 100%, each training instance is assigned to a classifier in the pool that correctly labels it. The hit rate is then obtained using the training set as test set, and comparing the chosen classifier to the correct classifier indicated by the SGH method for each training instance. Thus, the hit rate is the rate at which the DCS technique selects the correct classifier for a given known instance.

Since the hit rate is defined specifically for pools generated using the SGH method, the hit rate of the proposed method is only compared with the GP configuration, which uses a pool generated by the SGH method with the entire training set as input. The hit rate of the proposed configurations are calculated the same way as the GP configuration, with the only difference being for instances not in difficult regions. In this case, the accuracy rate of the k-NN rule is used to compute the measure. The comparison between the GP and the LP configurations is relevant because it provides the answer to whether or not the generation over a local region instead of over the entire problem is useful in the selection process of a DCS technique.

Table 2.3 shows the mean hit rate for OLA, LCA and MCB of the three configurations that use the SHG method. In comparison with the GP configuration, both LP and LP^e configurations obtained a greater overall hit rate for both DCS techniques. More specifically, for the LP^e configuration, nearly half of the datasets yielded a hit rate above 90%, whilst for the GP only two of them at most obtained a similar hit rate for the three DCS techniques.

Table 2.3 Mean and standard deviation of the hit rate, i.e., the rate at which the right Perceptron is chosen by (a) OLA, (b) LCA and (c) MCB using the GP, LP and LP^e configurations. The row Wilcoxon shows the result of a Wilcoxon signed rank test over the mean hit rates of the GP configuration and the two proposed configurations. The significance level was $\alpha = 0.05$, and the symbols +, − and ∼ indicate the method is significantly superior, inferior or not significantly different, respectively. Best results are in bold.

(a)

Dataset	**GP**	**LP**	**LP^e**
Adult	86.91 (0.87)	86.10 (1.13)	**91.41 (0.87)**
Blood	79.59 (0.51)	73.11 (1.42)	**80.96 (0.88)**
CTG	92.50 (0.59)	92.66 (0.83)	**92.95 (0.92)**
Faults	**76.88 (1.26)**	74.85 (0.70)	70.92 (0.95)
German	71.05 (1.44)	89.25 (0.56)	**91.86 (0.34)**
Glass	**76.21 (1.98)**	63.07 (0.75)	69.71 (1.74)
Haberman	76.26 (1.10)	69.14 (2.45)	**77.55 (0.93)**
Heart	84.06 (1.92)	90.06 (1.24)	**92.64 (0.54)**
Ionosphere	86.46 (1.48)	87.26 (1.59)	**87.97 (1.19)**
Laryngeal1	84.75 (2.07)	87.16 (0.89)	**89.82 (1.22)**
Laryngeal3	74.81 (2.95)	**79.63 (1.32)**	76.44 (1.20)
Liver	67.22 (1.40)	79.16 (1.28)	**83.92 (0.89)**
Mammographic	82.72 (0.64)	70.32 (1.61)	**84.02 (0.84)**
Monk2	85.77 (3.60)	95.85 (0.32)	**96.47 (0.33)**
Phoneme	87.40 (0.46)	88.56 (0.23)	**90.09 (0.29)**
Pima	75.64 (1.55)	83.00 (0.63)	**87.65 (0.28)**
Sonar	80.00 (3.62)	92.48 (1.11)	**93.81 (1.13)**
Vehicle	76.14 (1.49)	**78.25 (1.00)**	77.32 (0.70)
Vertebral	82.39 (2.14)	87.42 (1.27)	**89.38 (1.02)**
Weaning	83.45 (1.33)	94.82 (0.45)	**94.97 (0.35)**
Average	80.51	83.11	**86.00**
Wilcoxon	n/a	∼	+

(b)

Dataset	**GP**	**LP**	**LP^e**
Adult	86.77 (0.92)	89.99 (0.60)	**91.27 (0.89)**
Blood	80.20 (0.35)	79.43 (1.35)	**80.27 (1.20)**
CTG	92.63 (0.44)	**94.03 (0.27)**	93.30 (0.34)
Faults	**76.84 (1.01)**	74.85 (0.39)	71.25 (0.61)
German	75.75 (1.35)	90.01 (0.63)	**91.90 (0.32)**
Glass	**77.95 (1.92)**	67.73 (1.37)	69.17 (2.00)
Haberman	**76.61 (1.46)**	74.76 (1.84)	76.60 (1.04)
Heart	83.86 (2.40)	91.85 (0.79)	**92.77 (0.68)**
Ionosphere	87.34 (1.53)	**92.32 (0.77)**	92.11 (0.75)
Laryngeal1	84.81 (2.38)	88.55 (0.94)	**89.83 (1.14)**
Laryngeal3	73.98 (1.99)	**80.28 (1.72)**	78.37 (2.01)
Liver	70.62 (2.91)	79.69 (1.26)	**84.10 (1.01)**
Mammographic	**82.83 (1.54)**	80.82 (1.15)	82.07 (0.85)
Monk2	91.82 (3.61)	**96.63 (0.27)**	96.44 (0.32)
Phoneme	89.48 (0.44)	**91.93 (0.32)**	91.31 (0.23)
Pima	76.02 (1.67)	84.07 (0.48)	**87.68 (0.27)**
Sonar	83.46 (3.45)	93.37 (1.08)	**94.27 (0.96)**
Vehicle	**77.98 (1.57)**	77.11 (0.78)	76.60 (0.77)
Vertebral	84.33 (2.32)	**89.87 (1.02)**	89.85 (1.00)
Weaning	84.38 (1.72)	**95.17 (0.51)**	95.07 (0.36)
Average	81.88	85.63	**86.21**
Wilcoxon	n/a	+	+

(c)

Dataset	**GP**	**LP**	**LP^e**
Adult	87.14 (0.73)	87.35 (1.19)	**89.76 (0.59)**
Blood	79.61 (0.51)	74.17 (1.48)	**79.67 (0.81)**
CTG	**92.49 (0.63)**	92.14 (0.44)	92.00 (0.37)
Faults	76.87 (1.26)	**77.16 (0.84)**	76.73 (0.49)
German	71.23 (1.47)	90.90 (0.61)	**91.93 (0.58)**
Glass	**76.27 (1.99)**	66.54 (1.44)	69.98 (1.61)
Haberman	76.35 (1.10)	69.03 (1.65)	**76.77 (1.59)**
Heart	83.96 (1.72)	89.10 (1.31)	**91.85 (1.17)**
Ionosphere	86.43 (1.43)	88.65 (0.77)	**92.14 (0.75)**
Laryngeal1	84.75 (1.93)	86.72 (1.11)	**88.71 (0.90)**
Laryngeal3	74.85 (2.90)	78.97 (1.48)	**80.79 (1.37)**
Liver	67.34 (1.28)	79.60 (1.24)	**83.68 (1.10)**
Mammographic	**82.68 (0.73)**	71.21 (1.48)	82.56 (1.04)
Monk2	86.67 (4.48)	**95.69 (0.28)**	95.35 (0.31)
Phoneme	87.40 (0.47)	89.12 (0.27)	**90.10 (0.18)**
Pima	75.82 (1.83)	83.52 (0.79)	**87.54 (0.31)**
Sonar	80.19 (3.63)	92.51 (0.91)	**93.89 (1.03)**
Vehicle	76.20 (1.51)	**77.90 (0.78)**	76.05 (0.76)
Vertebral	82.39 (2.19)	88.06 (1.27)	**89.63 (1.24)**
Weaning	83.36 (1.20)	**94.06 (0.51)**	93.88 (0.72)
Average	80.60	83.62	**86.15**
Wilcoxon	n/a	∼	+

Moreover, a Wilcoxon signed rank test with a significance level of $\alpha = 0.05$ was performed between the hit rate results for the GP and the two proposed configurations. It can be observed, from the Wilcoxon rows, that the proposed configurations yielded a significantly greater hit rate than the global configuration for LCA, and, in particular, the LP^e configuration obtained a significant increase in the hit rate for OLA and MCB as well. This suggests that the use of the local pools indeed facilitates the DCS technique in choosing the correct classifier for instances in difficult regions.

2.4.1.2 Performance in Generalization

The mean percentage of test instances with true hardness value above zero is depicted in the *True* bars of Figure 2.7 for all datasets. The true hardness value is obtained observing the neighborhood of each test instance over the entire dataset. The mean percentage of test instances deemed hard by the proposed method is also depicted in Figure 2.7 (*Estimated* bars). That is, the *Estimated* bars show the frequency at which the proposed method generated and used local pools, whilst the *True* bars show the actual proportion of instances in difficult regions for each problem. It can be observed that, though the proportion of instances in difficult regions varies greatly from problem to problem, the proposed method was mostly able to identify in which cases the query sample was truly located in a difficult region and thus generated a local pool to handle them.

The averaged value of the true and estimated percentage of hard instances is also indicated in Figure 2.7 by the $true$ and est lines, respectively. It can be observed that the mean percentage of test instances truly located near the borders was 65.04%, while the proposed method generated local pools for 64.46% of the test instances, on average.

We chose the Wilcoxon signed-rank test due to its robustness, as its result do not depend on the algorithms originally included in the comparison (Benavoli, Corani & Mangili, 2016). It can be observed from the Wilcoxon rows of Table 2.5 that both proposed configurations were

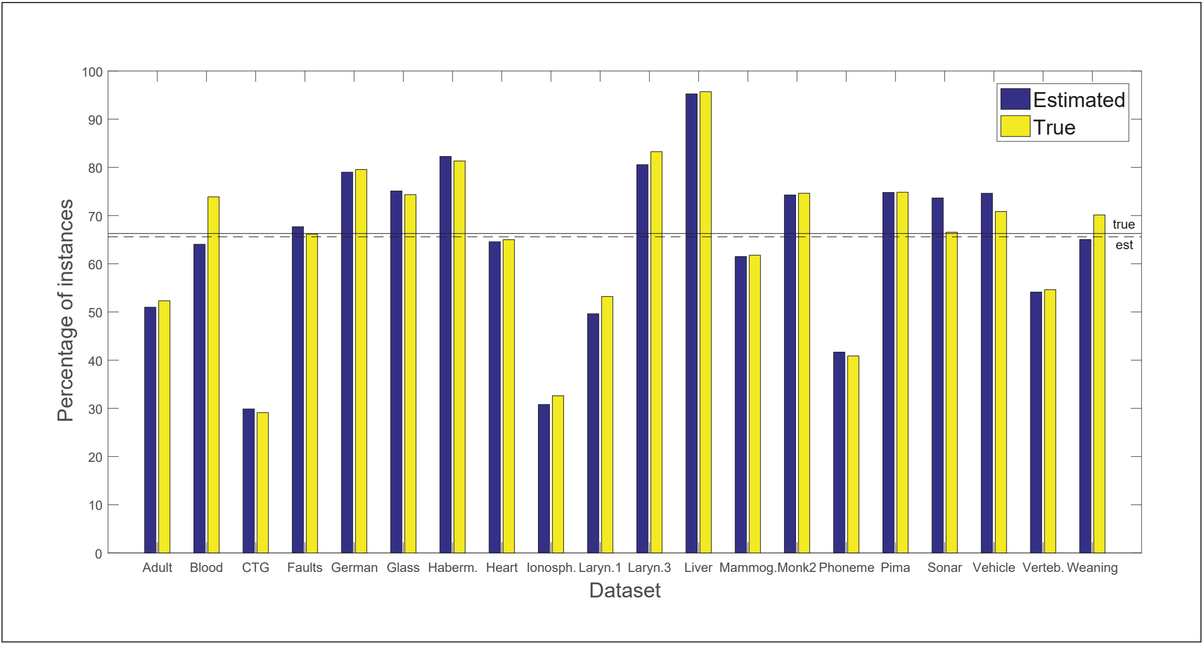


Figure 2.7 Mean percentage of instances in difficult regions for all datasets from Table 2.1. The *Estimated* bar indicates the times the local pool was used to classify an instance, while the *True* bar indicates the percentage of instances with true kDN value above zero. The lines $true$ and est indicate the averaged values of all datasets for the estimated and true percentage of hard instances, respectively.

significantly superior to Bagging, and the LP configuration significantly outperformed the GP configuration using LCA.

The accuracy rate of Bagging, FIRE-DES, the GP configuration and the proposed configurations were evaluated with OLA, LCA and MCB, and the results are presented in Table 2.4, Table 2.5 and Table 2.6. It can be observed that the proposed configurations (LP and LP^e) obtained an average accuracy rate greater than Bagging, FIRE-DES and the GP configuration for all DCS techniques. A Wilcoxon signed-rank test with a significance level of $\alpha = 0.05$ was also performed for comparing the accuracy of the evaluated techniques.

Also, it can be observed in Table 2.4, Table 2.5 and Table 2.6 that, for two-class problems with high percentage of difficult instances such as German, Liver, Monk2, Pima and Sonar (Figure 2.7), the use of local pools fairly increased the accuracy rate in comparison with the other

Table 2.4 Mean and standard deviation of the accuracy rate of using OLA for a pool with 100 Perceptrons generated using Bagging (Breiman, 1996) (column Bagging), a pool of 100 Perceptrons generated using Bagging and pruned with the DFP method (Oliveira *et al.*, 2017) (column FIRE-DES), the GP configuration, the LP configuration and the LP^e configuration. The row *Wilcoxon (Bagging)* shows the result of a Wilcoxon signed rank test over the mean accuracy rates of Bagging and each remaining method. The same test was performed in comparison with the FIRE-DES configuration and the GP configuration (rows *Wilcoxon (FIRE)* and *Wilcoxon (GP)*, respectively). The significance level was $\alpha = 0.05$, and the symbols +, − and ~ indicate the method is significantly superior, inferior or not significantly different, respectively. In LP^{mc}, the LP^e is used for 2-class problems, while the LP is used for multi-class ones. Best results are in bold.

Dataset	**Bagging**	**FIRE-DES**	**GP**	**LP**	**LP^e**	**LP^{mc}**
Adult	84.97 (2.50)	83.84 (3.37)	**88.15 (2.93)**	83.32 (3.63)	87.75 (2.17)	87.75 (2.17)
Blood	75.48 (2.31)	69.28 (3.90)	75.53 (1.14)	73.78 (2.80)	**77.21 (1.57)**	**77.21 (1.57)**
CTG	88.83 (1.26)	88.50 (1.36)	90.24 (0.77)	**92.20 (1.10)**	89.98 (0.80)	**92.20 (1.10)**
Faults	66.52 (1.65)	65.33 (1.95)	71.91 (1.60)	**72.40 (1.29)**	65.93 (1.29)	**72.40 (1.29)**
German	70.34 (1.88)	68.56 (1.89)	70.04 (2.35)	72.16 (2.12)	**74.04 (1.88)**	**74.04 (1.88)**
Glass	61.42 (4.22)	59.43 (5.66)	66.79 (4.17)	**67.83 (3.94)**	60.75 (2.17)	**67.83 (3.94)**
Haberman	70.79 (5.12)	66.78 (4.81)	71.58 (5.24)	68.95 (4.19)	**72.43 (2.24)**	72.43 (2.24)
Heart	82.35 (3.44)	82.21 (4.32)	**86.62 (2.18)**	81.99 (4.79)	83.68 (3.27)	83.68 (3.27)
Ionosphere	86.70 (3.04)	86.53 (2.84)	87.16 (2.76)	91.76 (1.95)	**91.99 (2.16)**	**91.99 (2.16)**
Laryngeal1	**82.92 (3.54)**	81.89 (5.62)	80.38 (4.26)	77.74 (5.53)	80.57 (5.87)	80.57 (5.87)
Laryngeal3	70.73 (5.79)	66.46 (4.73)	**72.25 (1.71)**	71.74 (2.73)	64.66 (1.34)	71.74 (2.73)
Liver	64.59 (4.18)	65.00 (3.85)	58.37 (3.53)	60.12 (4.99)	**67.21 (1.67)**	**67.21 (1.67)**
Mammographic	82.57 (2.02)	79.06 (3.62)	**82.60 (2.47)**	75.53 (2.60)	82.36 (1.81)	82.36 (1.81)
Monk2	87.87 (3.97)	87.92 (3.13)	86.20 (3.74)	**94.91 (0.97)**	94.07 (0.76)	94.07 (0.76)
Phoneme	80.31 (0.68)	76.02 (1.17)	86.74 (0.73)	**88.58 (0.63)**	86.60 (0.67)	86.60 (0.67)
Pima	72.40 (2.73)	68.78 (3.03)	72.29 (2.39)	72.03 (1.68)	**76.77 (2.26)**	**76.77 (2.26)**
Sonar	80.96 (4.04)	79.90 (4.02)	80.00 (3.33)	**83.27 (5.79)**	75.00 (4.14)	75.00 (4.14)
Vehicle	73.61 (2.56)	**74.43 (1.95)**	70.09 (2.57)	74.39 (2.09)	69.74 (1.66)	74.39 (2.09)
Vertebral	85.38 (4.04)	84.49 (4.70)	81.41 (2.06)	85.19 (2.14)	**86.47 (2.65)**	**86.47 (2.65)**
Weaning	77.50 (3.36)	77.57 (3.40)	78.68 (3.71)	**86.05 (1.73)**	85.66 (2.37)	85.66 (2.37)
Average	77.31	75.59	77.85	**78.69**	78.64	80.02
Wilcoxon (Bagging)	n/a	-	~	~	~	+
Wilcoxon (FIRE)	+	n/a	~	+	+	+
Wilcoxon (GP)	~	~	n/a	~	~	+

configurations for the three DCS techniques, further suggesting the advantage of such pools over the global one for instances in difficult regions.

Table 2.5 Mean and standard deviation of the accuracy rate of using LCA for a pool with 100 Perceptrons generated using Bagging (Breiman, 1996) (column Bagging), a pool of 100 Perceptrons generated using Bagging and pruned with the DFP method (Oliveira *et al.*, 2017) (column FIRE-DES), the GP configuration, the LP configuration and the LP^e configuration. The row *Wilcoxon (Bagging)* shows the result of a Wilcoxon signed rank test over the mean accuracy rates of Bagging and each remaining method. The same test was performed in comparison with the FIRE-DES configuration and the GP configuration (rows *Wilcoxon (FIRE)* and *Wilcoxon (GP)*, respectively). The significance level was $\alpha = 0.05$, and the symbols +, − and ~ indicate the method is significantly superior, inferior or not significantly different, respectively. In LP^{mc}, the LP^e is used for 2-class problems, while the LP is used for multi-class ones. Best results are in bold.

Dataset	**Bagging**	**FIRE-DES**	**GP**	**LP**	**LP^e**	**LP^{mc}**
Adult	86.88 (3.17)	85.72 (3.59)	**87.40 (2.82)**	84.71 (3.73)	87.11 (2.40)	87.11 (2.40)
Blood	76.14 (2.24)	71.06 (3.44)	75.74 (1.04)	**77.95 (2.51)**	76.89 (1.67)	76.89 (1.67)
CTG	88.38 (1.37)	88.18 (1.36)	90.30 (0.84)	**92.22 (1.10)**	90.58 (0.39)	**92.22 (1.10)**
Faults	66.00 (1.69)	65.67 (2.23)	71.99 (1.53)	**73.20 (1.22)**	66.28 (1.15)	**73.20 (1.22)**
German	70.66 (2.06)	70.40 (1.22)	70.84 (1.87)	72.88 (2.37)	**74.08 (1.84)**	**74.08 (1.84)**
Glass	56.13 (5.47)	56.04 (5.41)	**69.43 (3.33)**	67.45 (2.73)	62.55 (4.83)	67.45 (2.73)
Haberman	73.03 (3.58)	69.87 (4.87)	71.05 (1.91)	70.79 (3.71)	**72.11 (2.12)**	**72.11 (2.12)**
Heart	82.35 (4.84)	82.21 (4.77)	**86.47 (2.85)**	82.50 (5.54)	83.09 (3.32)	83.09 (3.32)
Ionosphere	86.14 (4.90)	86.19 (4.70)	87.27 (3.21)	91.53 (1.45)	**92.44 (2.56)**	**92.44 (2.56)**
Laryngeal1	81.23 (2.70)	80.09 (3.80)	**80.94 (4.70)**	79.25 (5.05)	80.57 (5.87)	80.57 (5.87)
Laryngeal3	71.57 (5.25)	68.88 (6.19)	72.58 (2.14)	**73.48 (2.48)**	67.42 (1.86)	**73.48 (2.48)**
Liver	64.59 (4.87)	66.74 (2.70)	58.37 (2.81)	62.21 (5.26)	**66.98 (1.79)**	**66.98 (1.79)**
Mammographic	82.00 (3.18)	78.89 (4.09)	81.63 (3.06)	80.05 (1.84)	**82.57 (1.77)**	82.57 (1.77)
Monk2	86.06 (3.06)	85.88 (3.16)	90.28 (2.18)	**94.91 (0.97)**	94.07 (0.76)	94.07 (0.76)
Phoneme	80.78 (0.65)	77.26 (0.89)	87.01 (0.77)	**89.18 (0.50)**	86.62 (0.69)	86.62 (0.69)
Pima	74.66 (2.39)	72.19 (3.63)	73.23 (3.39)	73.46 (1.01)	**76.74 (2.24)**	**76.74 (2.24)**
Sonar	76.35 (5.41)	75.87 (4.93)	78.08 (5.01)	**82.98 (5.27)**	76.35 (3.64)	76.35 (3.64)
Vehicle	72.03 (1.63)	72.41 (1.86)	70.75 (2.22)	**73.51 (1.64)**	71.34 (1.28)	**73.51 (1.64)**
Vertebral	84.55 (3.42)	85.51 (3.30)	82.31 (1.93)	85.32 (2.68)	**86.47 (2.65)**	**86.47 (2.65)**
Weaning	73.88 (2.78)	73.75 (3.51)	78.82 (3.05)	**86.51 (1.90)**	85.66 (2.37)	85.66 (2.37)
Average	76.67	75.64	78.22	**79.70**	78.99	80.08
Wilcoxon (Bagging)	n/a	-	~	+	+	+
Wilcoxon (FIRE)	+	n/a	+	+	+	+
Wilcoxon (GP)	~	-	n/a	+	~	+

2.4.1.3 Discussion

From Table 2.4, Table 2.5 and Table 2.6, it can be observed that the two evaluated configurations of the proposed method yielded quite distinct results: the LP configuration always surpassed, by far most of the times, the LP^e configuration for the multi-class problems for both DCS techniques. The reason for this difference in performance lies in the neighborhood selection

Table 2.6 Mean and standard deviation of the accuracy rate of using MCB for a pool with 100 Perceptrons generated using Bagging (Breiman, 1996) (column Bagging), a pool of 100 Perceptrons generated using Bagging and pruned with the DFP method (Oliveira *et al.*, 2017) (column FIRE-DES), the GP configuration, the LP configuration and the LP^e configuration. The row *Wilcoxon (Bagging)* shows the result of a Wilcoxon signed rank test over the mean accuracy rates of Bagging and each remaining method. The same test was performed in comparison with the FIRE-DES configuration and the GP configuration (rows *Wilcoxon (FIRE)* and *Wilcoxon (GP)*, respectively). The significance level was $\alpha = 0.05$, and the symbols +, − and ~ indicate the method is significantly superior, inferior or not significantly different, respectively. In LP^{mc}, the LP^e is used for 2-class problems, while the LP is used for multi-class ones. Best results are in bold.

Dataset	**Bagging**	**FIRE-DES**	**GP**	**LP**	**LP^e**	**LP^{mc}**
Adult	85.00 (2.53)	83.70 (3.25)	**88.15 (2.93)**	84.45 (3.98)	87.75 (2.18)	87.75 (2.18)
Blood	75.16 (2.07)	68.88 (3.24)	75.53 (1.14)	**76.99 (2.15)**	76.57 (1.90)	76.57 (1.90)
CTG	88.87 (1.24)	88.61 (1.54)	90.24 (0.77)	**92.10 (1.20)**	90.23 (0.36)	**92.10 (1.20)**
Faults	66.58 (1.37)	65.77 (2.32)	71.91 (1.60)	**72.80 (1.29)**	66.24 (0.86)	**72.80 (1.29)**
German	70.16 (2.41)	68.94 (2.72)	70.52 (2.08)	72.84 (2.36)	**74.08 (1.84)**	**74.08 (1.84)**
Glass	60.00 (5.83)	60.19 (6.45)	**66.79 (4.17)**	66.42 (4.27)	61.04 (2.89)	66.42 (4.27)
Haberman	72.17 (5.87)	68.36 (4.95)	71.71 (4.91)	69.80 (2.85)	**72.37 (2.67)**	72.37 (2.67)
Heart	81.10 (4.11)	81.32 (4.82)	**86.18 (2.36)**	82.06 (4.86)	83.09 (3.32)	83.09 (3.32)
Ionosphere	88.24 (2.56)	86.36 (2.55)	87.16 (2.71)	91.48 (1.40)	**92.16 (2.39)**	**92.16 (2.39)**
Laryngeal1	**83.02 (4.29)**	81.51 (6.44)	80.57 (4.59)	78.30 (5.03)	80.47 (5.75)	80.47 (5.75)
Laryngeal3	70.96 (5.62)	67.19 (4.71)	71.80 (1.58)	**72.19 (2.65)**	66.18 (1.41)	**72.19 (2.65)**
Liver	61.98 (4.86)	63.49 (5.01)	58.37 (3.49)	61.34 (4.71)	**67.03 (1.32)**	67.03 (1.32)
Mammographic	82.31 (2.32)	79.01 (3.40)	**82.60 (2.47)**	78.87 (2.55)	82.52 (1.65)	82.52 (1.65)
Monk2	88.06 (4.18)	87.69 (4.29)	87.96 (3.80)	**94.91 (0.97)**	94.07 (0.76)	94.07 (0.76)
Phoneme	80.53 (0.79)	76.12 (1.05)	86.73 (0.73)	**88.98 (0.56)**	86.68 (0.74)	86.68 (0.74)
Pima	72.73 (2.60)	68.36 (2.96)	72.71 (2.67)	72.92 (1.56)	**76.74 (2.28)**	**76.74 (2.28)**
Sonar	80.67 (4.11)	80.29 (4.13)	79.81 (3.09)	**83.08 (5.42)**	76.15 (3.33)	76.15 (3.33)
Vehicle	73.30 (2.54)	74.58 (2.45)	70.14 (2.52)	**74.88 (1.57)**	70.75 (1.65)	**74.88 (1.57)**
Vertebral	84.55 (4.75)	85.38 (4.86)	82.69 (2.22)	85.58 (2.38)	**86.41 (2.71)**	**86.41 (2.71)**
Weaning	76.38 (2.43)	76.05 (3.10)	79.21 (3.30)	**86.38 (1.72)**	85.59 (2.36)	85.59 (2.36)
Average	77.08	75.59	78.03	**79.31**	78.80	80.00
Wilcoxon (Bagging)	n/a	-	~	+	~	+
Wilcoxon (FIRE)	+	n/a	~	+	+	+
Wilcoxon (GP)	~	~	n/a	~	~	+

schemes used in the online phase of the proposed method, as it can be observed in Figure 2.8, in which two multi-class toy problems are depicted.

In Figure 2.8a, the neighborhood θ_1 of the query instance $\mathbf{x_q}$ was obtained using the regular k-NN rule. It can be observed that, since the border contains only two classes (Class 1 and Class 2), this is also the case for all two-class problems. Therefore, the SGH method, which generates only two-class classifiers, returns a pool with only one classifier ($c_{1,1}$) that cover the entire neighborhood θ_1. Figure 2.8b shows the same scenario, but with θ_1 being obtained using the version of k-NNE used in this work, which returns the same amount of neighboring instances for all classes in the original k-NN neighborhood. That is, the instances from classes too far from the query sample are not included in this method, as Figure 2.8b shows. The generated pool also contains only one classifier ($c_{1,1}$) that cover the instances in θ_1. In both presented cases, the DCS technique would select the correct classifier for this query sample, which belongs to Class 1, though the classifier from Figure 2.8b seems better adjusted than the one from Figure 2.8a.

On the other hand, Figure 2.8c shows a similar situation, but with Class 3 much closer to the other two classes. In this case, the neighborhood θ_1 returned by k-NN contains instances from the three classes in the problem. Since the SGH method only generates two-class classifiers, the coverage of θ_1 is incomplete. This is due to the fact that the most distant class in the input set is selected more frequently to draw the hyperplanes. It can be observed in Figure 2.8c that Class 3, which is the farthest class and thus the least relevant one, is much better covered, with all classifiers recognizing it, than the other two classes. In fact, there is not one classifier that separates Class 1 from Class 2 in the generated pool. However, since the DCS technique evaluates the classifiers competence over θ_1 in the proposed technique, Class 3 only possesses one instance, therefore its weight is much smaller than the remaining two classes in the classifiers' score. That way, the classifier $c_{1,3}$ would be selected by OLA, for instance, which would yield the correct label of $\mathbf{x_q}$.

Figure 2.8d depicts the same scenario from Figure 2.8c, but with θ_1 obtained using k-NNE. Since the original k-NN neighborhood already contained an instance from Class 3, this class is also included in θ_1. This leads to the neighborhood containing $k_1 = 7$ instances of each of the three classes of the problem. The SGH method generates then two classifiers ($c_{1,1}$ and $c_{1,2}$), and, as in the previous case, the most distant and least relevant class (Class 3) is favoured by the method, since all classifiers recognize it. The other two classes, which are closer to $\mathbf{x_q}$, do not have a classifier in this subpool to distinguish among themselves. However, as opposed to the previous case, the amount of instances of the farthest class is the same as the other two classes,

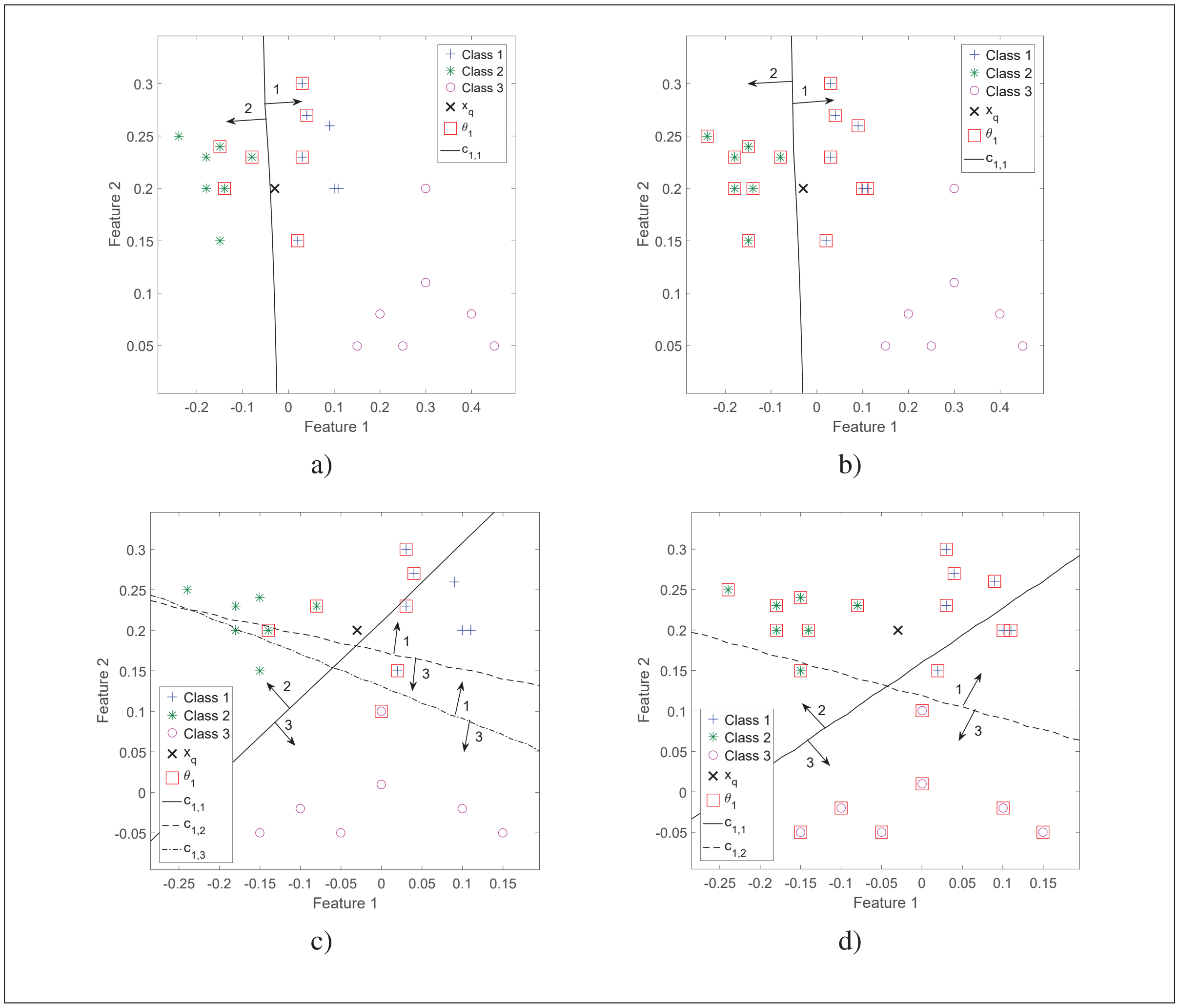


Figure 2.8 Example of pool generation for multi-class problems. In all scenarios, x_q belongs to Class 1. In (a) and (c), the query instance's ($\mathbf{x_q}$) neighborhood θ_1 was obtained using k-NN with $k_1 = 7$. In (b) and (d), θ_1 was obtained using a version of k-NNE with $k_1 = 7$ as well. These neighborhoods were used as input to the SGH method, which yielded the corresponding subpool of classifiers depicted in the images.

which makes its as relevant as the closer classes for the DCS techniques, since the classifiers are evaluated over the entire θ_1. In this example, as both classifiers correctly label two out of three classes in the neighborhood, the DCS technique would choose one of them randomly, which would in turn fairly degrade the performance of the system.

Therefore, a better approach for multi-class problems is to use the LP, which evaluates over the original neighborhood and is likely to give less weight to less relevant classes in the border region. Hence, the LP^{mc} columns in Table 2.4, Table 2.5 and Table 2.6 show the result of the combined LP^{e} and LP configurations, in which the k-NNE is used for 2-class problems and the k-NN for the multi-class problems. It can be observed from the Wilcoxon rows that this scheme is significantly better than Bagging, FIRE-DES and the GP configurations for all DCS techniques.

2.4.2 Comparison with State-of-the-art Models

A comparative study on the performances of the proposed method and nine state-of-the-art models is presented in this section. The purpose of this study is to assess whether the proposed method achieves similar recognition rates to the most well-performing models in the literature, considering single models and other MCS.

Five static state-of-the-art classifiers feature in the comparative study: the Multi-layer Perceptron (MLP) model with the Levenberg-Marquadt algorithm, the Support Vector Machine (SVM) model with a Gaussian Kernel, the Random Forest (RF) (Breiman, 2001) classifier, the AdaBoost (Freund & Schapire, 1997) classifier, and the Oblique Decision Tree (DT) ensemble (Zhang & Suganthan, 2017). These models belong to the best performing families of classifiers, according to (Fernández-Delgado, Cernadas, Barro & Amorim, 2014), and also were among the best performing models in (Zhang & Suganthan, 2017). Since static models do not need a DSEL dataset, it was used as a validation set for the MLP classifier and added to the training set for the remaining static models.

Furthermore, four state-of-the-art DS techniques were also included in this analysis: the Randomized Reference Classifier (RRC) (Woloszynski & Kurzynski, 2011), the META-DES (Cruz *et al.*, 2015a), the META-DES.Oracle (META-DES.O) (Cruz *et al.*, 2017a) and the FIRE-KNORA-U (F-KNU) (Oliveira *et al.*, 2017). The latter consists of the FIRE-DES framework coupled with the K-Nearest Oracles Union (KNORA-U) (Ko *et al.*, 2007) selection technique.

The same Bagging-generated pool of 100 Perceptrons used in the previous section was used for these techniques. The region of competence size was also set to 7, as in the previous experiments.

All classifiers were evaluated using the MATLAB PRTOOLS toolbox (Duin, Juszczak, de Ridder, Paclik, Pekalska & Tax, 2004), and the parameters of the static models were set to the default. [2] Moreover, the proposed method's configuration used for comparison in this analysis was the LP^{mc} with LCA, since it yielded the highest mean accuracy rate in the previous experimental study.

Table 2.7 shows the mean accuracy rate of the static classification models and the proposed method, for all datasets from Table 2.1. It can be observed that the proposed configuration yielded a higher overall accuracy rate than all static models but the Oblique DS ensemble. It is important to remember, though, that no fine tuning of parameters was performed, and that stands for all static models as well as the proposed technique. A Wilcoxon signed-rank test was also performed over the results (row Wilcoxon), and it can be observed that the performance of the proposed configuration was significantly superior to the MLP and SVM models.

Table 2.8 shows the mean accuracy rate of the four state-of-the-art DES techniques and the proposed configuration. It can be observed that the proposed technique obtained a greater mean accuracy rate in comparison with three of the four DES techniques. Moreover, according to a Wilcoxon signed-rank test with significance level of $\alpha = 0.05$ (Wilcoxon row), the proposed configuration obtained a significantly superior performance to the F-KNU technique. In comparison to the remaining DES techniques, the proposed method yielded a statistically similar performance.

[2] Specific details on the hyperparameters:
MLP: Levenberg-Marquardt, hidden layer size: 10, loss: categorical cross entropy
SVM: Gaussian kernel, γ: 1, C: 1
AdaBoost: Classification and Regression Trees (CART), pool size: 100
RF: Pool size: 50, features per split: $\sqrt{\text{\# of features}}$
Oblique DT: Pool size: 50, features per split: $\sqrt{\text{\# of features}}$
DS techniques: Perceptron/Bagging, pool size: 100, K: 7

Table 2.7 Mean and standard deviation of the accuracy rate of MLP, SVM, RF (Breiman, 2001), AdaBoost (Freund & Schapire, 1997), Oblique DT ensemble (Zhang & Suganthan, 2017) and the LP^{mc} configuration. The row *Wilcoxon* shows the result of a Wilcoxon signed rank test over the mean accuracy rates of the proposed configuration and each of the remaining methods. The significance level was $\alpha = 0.05$, and the symbols +, $-$ and ~ indicate if the compared method is significantly superior, inferior or not significantly different from the proposed method, respectively. Best results are in bold.

Dataset	**MLP**	**SVM**	**RF**	**AdaBoost**	**Oblique DT ens.**	**LP^{mc}**
Adult	82.83 (3.61)	73.99 (2.88)	67.83 (8.34)	88.44 (2.05)	**88.76 (1.43)**	87.11 (2.40)
Blood	78.11 (1.63)	**78.14 (1.08)**	72.07 (3.00)	77.29 (1.74)	77.23 (0.95)	76.89 (1.67)
CTG	89.52 (1.46)	84.51 (0.53)	91.20 (0.94)	92.42 (1.69)	**93.96 (0.68)**	92.22 (1.10)
Faults	68.79 (4.45)	49.59 (0.30)	70.86 (2.35)	54.66 (2.43)	**77.04 (2.06)**	73.20 (1.22)
German	70.24 (3.08)	70.12 (0.19)	38.28 (6.78)	74.82 (1.94)	**75.18 (1.88)**	74.08 (1.84)
Glass	61.32 (7.20)	62.83 (3.47)	71.04 (4.59)	48.30 (5.98)	**78.58 (3.14)**	67.45 (2.73)
Haberman	70.72 (2.33)	**74.34 (2.02)**	69.41 (4.24)	68.62 (3.48)	72.04 (3.69)	72.11 (2.12)
Heart	75.15 (5.02)	58.82 (1.79)	62.94 (6.45)	84.26 (4.80)	**86.32 (3.20)**	83.09 (3.32)
Ionosphere	87.84 (4.30)	68.07 (2.24)	93.81 (2.16)	**96.36 (1.50)**	95.17 (1.32)	92.44 (2.56)
Laryngeal1	79.53 (6.16)	76.23 (3.26)	81.79 (4.55)	81.60 (3.29)	**85.94 (2.90)**	80.57 (5.87)
Laryngeal3	67.87 (5.36)	68.88 (2.78)	72.53 (2.93)	70.11 (2.88)	**73.54 (3.25)**	73.48 (2.48)
Liver	68.31 (4.80)	64.53 (3.82)	69.30 (4.52)	69.71 (3.49)	**71.05 (4.11)**	66.98 (1.79)
Mammographic	84.45 (2.94)	83.46 (2.87)	59.28 (8.08)	80.70 (2.25)	**84.57 (1.45)**	82.57 (1.77)
Monk2	99.49 (1.06)	95.28 (1.37)	89.54 (2.93)	**100.0 (0.00)**	96.90 (1.28)	94.07 (0.76)
Phoneme	83.51 (1.07)	87.43 (0.43)	90.34 (0.49)	**90.66 (0.55)**	89.65 (0.52)	86.62 (0.69)
Pima	74.35 (3.63)	71.46 (2.20)	76.25 (2.67)	75.42 (1.94)	**77.21 (1.37)**	76.74 (2.24)
Sonar	78.08 (5.49)	81.15 (4.02)	83.75 (5.38)	**85.38 (5.16)**	83.56 (5.56)	76.35 (3.64)
Vehicle	**76.91 (2.38)**	63.63 (2.85)	73.77 (2.21)	68.94 (3.30)	74.27 (2.43)	73.51 (1.64)
Vertebral	81.03 (4.15)	85.00 (2.76)	85.45 (3.63)	84.04 (2.48)	85.96 (3.80)	**86.47 (2.65)**
Weaning	78.42 (5.20)	69.74 (6.80)	86.97 (2.73)	**87.76 (1.87)**	86.38 (2.06)	85.66 (2.37)
Average	77.82	73.36	75.32	78.97	**82.66**	80.08
Wilcoxon	-	-	~	~	+	n/a

2.4.3 Computational Complexity

An analysis of the complexity of the proposed method versus the complexity of different DS techniques is performed next using the Big-O (O) and Big-Omega (Ω) notations (Knuth, 1976) to represent the worst and best running time scenarios, respectively. This analysis is made taking into account the dataset size n, the classifiers pool size m, the dimensionality of the dataset d and the neighborhood size k. For simplicity, we consider that all base classifiers are from the same model (Perceptron), and the cost of training the base classifier is denoted by l.

Table 2.8 Mean and standard deviation of the accuracy rate of the Randomized Reference Classifier (RRC) (Woloszynski & Kurzynski, 2011), the META-DES (Cruz *et al.*, 2015a), the META-DES.Oracle (META-DES.O) (Cruz *et al.*, 2017a), the FIRE-KNORA-U (F-KNU) (Oliveira *et al.*, 2017) and the LP^{mc} configuration. The row *Wilcoxon* shows the result of a Wilcoxon signed rank test over the mean accuracy rates of the proposed configuration and each of the remaining methods. The significance level was $\alpha = 0.05$, and the symbols +, − and ~ indicate if the compared method is significantly superior, inferior or not significantly different from the proposed method, respectively. Best results are in bold.

Dataset	RRC	META-DES	META-DES.O	F-KNU	$\mathbf{LP^{mc}}$
Adult	**88.87 (2.27)**	84.45 (6.41)	80.78 (7.33)	84.86 (2.85)	87.11 (2.40)
Blood	76.30 (1.41)	**77.98 (1.81)**	**77.98 (1.20)**	64.87 (2.56)	76.89 (1.67)
CTG	89.41 (0.71)	91.49 (0.66)	92.10 (1.12)	88.34 (0.94)	**92.22 (1.10)**
Faults	70.35 (1.06)	**73.58 (1.57)**	73.39 (1.61)	67.44 (1.74)	73.20 (1.22)
German	**76.42 (2.14)**	75.70 (1.69)	74.76 (1.82)	70.30 (1.01)	74.08 (1.84)
Glass	65.19 (4.39)	**70.19 (3.44)**	69.53 (5.17)	65.47 (3.73)	67.45 (2.73)
Haberman	74.08 (1.71)	73.82 (5.79)	**75.26 (2.36)**	57.24 (4.91)	72.11 (2.12)
Heart	86.62 (1.42)	**86.47 (3.53)**	81.10 (4.35)	85.51 (2.35)	83.09 (3.32)
Ionosphere	88.75 (2.24)	88.47 (2.19)	85.17 (5.10)	88.35 (1.91)	**92.44 (2.56)**
Laryngeal1	**85.19 (3.08)**	80.28 (4.95)	78.21 (6.14)	80.94 (5.30)	80.57 (5.87)
Laryngeal3	**74.27 (3.40)**	73.54 (3.31)	73.48 (3.63)	66.24 (4.13)	73.48 (2.48)
Liver	65.81 (4.34)	**68.95 (3.25)**	67.21 (3.57)	60.58 (3.99)	66.98 (1.79)
Mammographic	**85.77 (2.08)**	73.13 (15.9)	72.36 (18.2)	78.53 (2.58)	82.57 (1.77)
Monk2	85.23 (2.71)	**96.76 (1.22)**	**96.76 (1.22)**	85.32 (2.57)	94.07 (0.76)
Phoneme	74.08 (1.57)	87.49 (0.82)	**89.34 (0.69)**	73.89 (1.61)	86.62 (0.69)
Pima	76.95 (2.33)	**77.40 (1.94)**	76.98 (2.49)	67.29 (2.73)	76.74 (2.24)
Sonar	80.96 (2.92)	82.98 (3.38)	**83.75 (2.96)**	81.35 (2.93)	76.35 (3.64)
Vehicle	75.40 (1.97)	75.94 (2.21)	75.12 (2.11)	**76.56 (1.84)**	73.51 (1.64)
Vertebral	85.19 (3.01)	86.22 (3.45)	85.77 (2.79)	**87.12 (3.98)**	86.47 (2.65)
Weaning	81.84 (3.27)	84.28 (3.37)	82.89 (3.28)	81.12 (3.29)	**85.66 (2.37)**
Average	79.33	**80.45**	79.59	75.50	80.08
Wilcoxon	~	~	~	-	n/a

During the offline phase of the proposed method, the hardness value of all instances are estimated, so the final cost of the proposed method in memorization is equal to applying the k-NN rule over the training set, $O(n^2d)$. For the DS techniques, however, the training step includes generating a pool of size m, which yields a cost of $O(ml)$, and pre-processing the base classifiers outputs for each sample in the training set, thus adding a cost of $O(mn)$. As for the computational cost in generalization, the analysis is conducted by dividing each algorithm into the DS techniques's three steps (Cruz *et al.*, 2018a): region of competence definition, competence estimation and classification. Then, the computational complexity of each part is obtained individually.

For the region of competence definition, the proposed method and all DS methods studied in this work are based on the k-NN algorithm. Thus, it is of order $O(nd)$. The two versions of the META-DES framework also require a local competence estimation on the decision space, resulting in a computational complexity of $O(nd+nm)$ for these techniques.

In the competence level estimation, the cost involved to calculate the competence level of m base classifiers is equal to getting their performance on the query sample's neighbors, $O(mk)$. The Randomized Reference Classifier (RRC) algorithm, however, uses the whole dataset to calculate the competence level of the base classifiers, making this step of order $O(nm)$.

In the proposed method, the SGH method is applied in each iteration over a different neighborhood, resulting in m'' classifiers. These classifiers are then evaluated over a neighborhood of size k, so, for each iteration, $O(km''d)$. The best classifier produced in each iteration is added to the local pool, so, for a local pool size of m' hyperplanes, the cost involved in this part is $O(m''km'd)$.

The classification step using the proposed method is performed by applying the majority voting rule over the m' local classifiers, so the computational cost is of order $O(m'd)$. In contrast, the DCS techniques use only one classifier for classification, which yields a computational complexity of $O(d)$. The DES techniques, however, can select an arbitrary number of base classifiers, so in the worst case scenario, when all classifiers in the pool are selected, it has a complexity of order $O(md)$.

This analysis explains why the proposed method was much faster in generalization than the DS techniques using a Bagging-generated pool. In the best case scenario, when the sample is located in an easy region, no local pool is generated, yielding a cost equal to the k-NN classifier's, $\Omega(nd)$. In the worst case, the computational cost is the sum of the three steps: $O(nd+m''dkm'+m'd)$. However, DCS and DES techniques always require the computation of the three steps regardless of whether or not the query sample is located in an easy region. For this reason, the average running time of the proposed method was three times faster than the DCS techniques using Bagging-generated pool, eight times faster than the KNORA-U and also around 14 times faster than the RRC and META-DES techniques. However, the DCS techniques using the GP were 10 times faster than the proposed method, probably due to its reduced pool size (3.80 classifiers, on average).

2.5 Conclusion

In (Souza *et al.*, 2017), it was shown that the DCS techniques had difficulty in selecting a competent classifier even though the presence of such a classifier in the pool was assured. The generation method used in that work guaranteed an Oracle accuracy of 100%. It was concluded that the Oracle model, being performed globally, did not help in the search for a good pool of classifiers for DCS techniques, because the latter use only local data to select a competent classifier for any given instance.

In this work, an instance hardness analysis was performed in order to draw a correlation between hardness measures and the error rates of DCS techniques. Based on that relationship, an online pool generation scheme was proposed with the purpose of increasing the accuracy rates of the instances the DCS techniques had difficulty in labelling. The proposed technique involved generating subpools for each identified difficult region in the feature space, so that another, more locally accurate pool could be used for hard instances, in hopes that, by fully covering these regions with a locally specialist pool, it would be easier for the DS techniques to select the best classifiers for these samples. The instances deemed “easy", however, would be classified using a simple k-NN rule.

Experiments were performed over 20 public datasets, and two configurations of the proposed scheme were analyzed. It was shown that the use of local pools increased the hit rate of the global pool (GP) for most datasets, suggesting the use of such pools indeed helps the DS in selecting the most competent ones for a given query instance. The overall performances of both configurations were compared with a pool of 100 Perceptrons generated using Bagging, with an online pruned pool of originally 100 Bagging-generated Perceptrons and with the GP. It was observed that a combination of both proposed configurations yields a significantly increase in accuracy rate compared to the other tested methods for the three evaluated DCS techniques, suggesting that, not only do the DCS techniques select the best classifier more frequently, but also the recognition rates of the DCS techniques fairly increase when using a local perspective during generation. The choice of which proposed configuration to use is based on the characteristics of each problem, and this selection is necessary due to a limitation in the SGH method. Furthermore, the proposed technique was compared to nine state-of-the-art classification models, including five static and four DES techniques, and it yielded a significant superior performance to three of the models and a statistically similar performance to five of them.

Improvements to the proposed technique may involve developing an automatic scheme for defining the input parameters and an adaptation for better dealing with multi-class problems. Moreover, the impacts of data preprocessing on the performance of DS techniques over imbalanced problems have been analyzed in (Roy, Cruz, Sabourin & Cavalcanti, 2018). Thus, a study on the robustness of the proposed method to class imbalance and the suitability of using data preprocessing techniques for imbalance learning may also be performed in future works.

In the next chapter, a novel version of the OLP, the OLP++ technique, is proposed to address the locality definition in high dimensional spaces for the production and selection of local experts in the presence of class overlap.

CHAPTER 3

OLP++: AN ONLINE LOCAL CLASSIFIER FOR HIGH DIMENSIONAL DATA

Mariana A. Souza[1] , Robert Sabourin[1] , George D. C. Cavalcanti[2] , Rafael M. O. Cruz[1]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA),
École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Centro de Informática,
Universidade Federal de Pernambuco,
1235 Av. Prof. Moraes Rego, Recife, Pernambuco, Brazil 50670-901

Article published in Elsevier's Information Fusion Journal, Volume 90, February 2023

Abstract

Ensemble diversity is an important characteristic of Multiple Classifier Systems (MCS), which aim at improving the overall performance of a classification system by combining the response of several models. While diversity may be introduced through various manipulations at the data level and the model level, some MCSs incorporate local information in order to increase it and/or take advantage of it, based on the idea that the different classifiers in the ensemble may have expertise in distinct areas of the feature space. Following a similar reasoning, we introduced in a previous work an ensemble method which produces in test time a few experts in the local region where each given query sample is located. These local experts, which are generated with slightly differing views of the target area, are then used to label the corresponding unknown instance. While the framework was shown to perform well especially over imbalanced problems, the locality definition in the method is based on the nearest neighbors rule and Euclidian distance, as is the case of various local-based ensembles, which may suffer from the effects of the curse of dimensionality over high dimensional problems. Thus, in this work, we propose a local ensemble method in which we leverage the data partitions given by decision trees for locality definition. More specifically, the partitions defined at different levels of the decision path that a given query instance traverses in the tree(s) are used as the regions over which the local experts are produced.

By using different node levels from the path, each classifier in the local pool has a moderately distinct view of the target region without resorting to a dissimilarity metric, which might be susceptible to high dimensional spaces. Experimental results over 39 high dimensional problems showed that the proposed approach was significantly superior to our previous, distance-based framework in balanced accuracy rate. Compared to other six local-based ensemble methods, including dynamic selection and weighting schemes, the proposed method achieved competitive results, outperforming the random forest baseline and two state-of-the-art dynamic ensemble selection techniques. [3]

3.1 Introduction

Multiple Classifier Systems (MCS) combine several base-classifiers with the purpose of leveraging their complementarity in a way that the final response of the system outperforms each individual model in the ensemble (Kittler *et al.*, 1998). This view is intrinsically associated with the concept of *ensemble diversity*, a desirable characteristic which may be intuitively thought as how distinctly the components of the ensemble respond to samples in different areas of the feature space (Kuncheva & Whitaker, 2003). Diversity may be achieved in multiple ways during the ensemble generation process (Duin, 2002; Kuncheva, 2014), including using different initial model configurations, hyperparameter settings, architectures, learning algorithms, sets of samples and/or sets of features. Another important aspect to ensemble methods design is the output aggregation step, which produces the final response of the system. Aggregation may be done using a fixed combination rule, such as majority voting over the classifiers' output labels or average over the classifiers' output probabilities (Kittler *et al.*, 1998), or using a trained combiner that is adapted to the problem (Duin, 2002), as in the mixture of experts (ME) paradigm (Jacobs *et al.*, 1991). Furthermore, the aggregation approaches may regard the outputs of all classifiers in the ensemble as equally important, or they may make some distinctions among their responses, for instance in the form of a weighting scheme, assuming their diversity entails different levels of confidence in them (Duin, 2002).

[3] Code available at: github.com/marianaasouza/olp_plusplus.

Some approaches do not take into account the different geometrical areas of the feature space when building the MCS. For instance, the classical AdaBoost (Freund & Schapire, 1997) ensemble produces the classifiers iteratively encouraging the specialization of the following models on the samples the previously generated ones misclassify. While the models' distinct strengths are considered in the AdaBoost aggregation rule, they are not associated with any defined geometrical area and thus the behavior of the MCS in generalization does not change according to where a given unknown sample is located. *Local* approaches, on the other hand, assume there are several regions of interest in the feature space (Bischl, Schiffner & Weihs, 2013), and so one can encourage in generation and/or exploit in aggregation the classifiers' distinct strengths over each different region. That is the reasoning behind several local ensemble methods, including Dynamic Selection (DS) techniques, dynamic weighting schemes and many hybrid approaches, which adapt the classifiers' importances for each test sample at hand according to their perceived expertise around it (Armano & Tamponi, 2018; Britto *et al.*, 2014; Cruz *et al.*, 2018a). However, in order to incorporate local information into the MCS, numerous ensemble methods apply a (dis)similarity metric, usually the Euclidean distance, on the feature space, which may introduce some hindering effects to the system when dealing with high dimensional data, as part of the curse of dimensionality (Bellman, 2015).

In a previous work, we proposed a local-based dynamic approach to ensemble generation and selection called the Online Local Pool (OLP) framework (Souza *et al.*, 2019b). In the OLP, several local experts are sequentially produced and singled out during generalization in the area surrounding each query instance, with decreasing locality, if the instance is located near the class borders. Otherwise, the K-Nearest Neighbors (KNN) is used to label the unknown sample. That way, the OLP method attempts to guarantee the presence of highly specialized classifiers over each region of interest, if it presents any label ambiguity. The OLP technique yielded state-of-the-art performance compared to other local-based dynamic approaches and was shown to work well specially in class-imbalanced scenarios (Souza *et al.*, 2019a). However, the region definition performed in both the detection of class overlap and the production of the local pool are based on the nearest neighbors rule and the Euclidean distance which, similarly to other local ensemble approaches, may present the effects of the curse of dimensionality on high dimensional problems.

Thus, in this work, we propose a local ensemble method based on the OLP framework that is better suited for dealing with high dimensional data. Instead of a distance-based region definition procedure, we leverage the data partitions defined at several nodes, including but not restricted to the leaves, of one or more decision trees fitted to the data in order to define the regions of interest in the feature space. So, for each query instance, we use the region definitions from a number of consecutive nodes within its decision path to fit the local pool of linear classifiers used to produce its output label, unless the path contains a pure leaf, in which case the leaf's local rule is used. Hence, the border detection and region definition in the proposed technique are achieved through the recursive partitioning of the feature space given by the tree-based algorithms, which may be more adequate for high dimensional spaces compared to the distance-based alternative. Moreover, using the regions defined at different levels of the trees iteratively reduces the locality and aims at injecting diversity between the classifiers while attempting to preserve some degree of consensus in the pool.

This work is organized as follows. In Section 3.2 we present the motivation for this work, with the help of an illustrative example. In Section 3.3 we introduce the proposed method in detail, including the pseudo-algorithm and a toy example. We discuss the related work found in the literature in Section 3.4. The experimental analysis is conducted in Section 3.5, and our concluding remarks are presented in Section 3.6.

3.2 Problem statement

We start by illustrating the possible issues associated with the local region definition process given by the nearest neighbors rule using the Euclidean distance in high dimensional spaces, which motivated the proposed approach based on the OLP framework for high dimensional data. Figure 3.1 shows a two-class balanced toy problem of 300 points, 80% of which set apart for training and the remaining for test, in a two-dimensional space. Both of the problem's features are informative. The colors orange and green represent the two classes in the problem, and all training instances are depicted in round markers. The diamond shaped marker indicates a test instance from the green class.

Each grey area in Figure 3.1 indicates the local target region defined by (a) the KNN classifier, (b) the decision tree classifier, and (c-d) two decision trees with random feature sampling at each split, as in the Random Forest algorithm, all fitted over the training set. In the case of the trees, we show only the region defined on the leaf level. Moreover, the training instances that compose the local target region are highlighted in red. To compare the regions formed by these four classifiers, we set the minimum number of samples per leaf of all the trees to $K = 7$, which is the number of neighbors (hyperparameter) of the KNN.

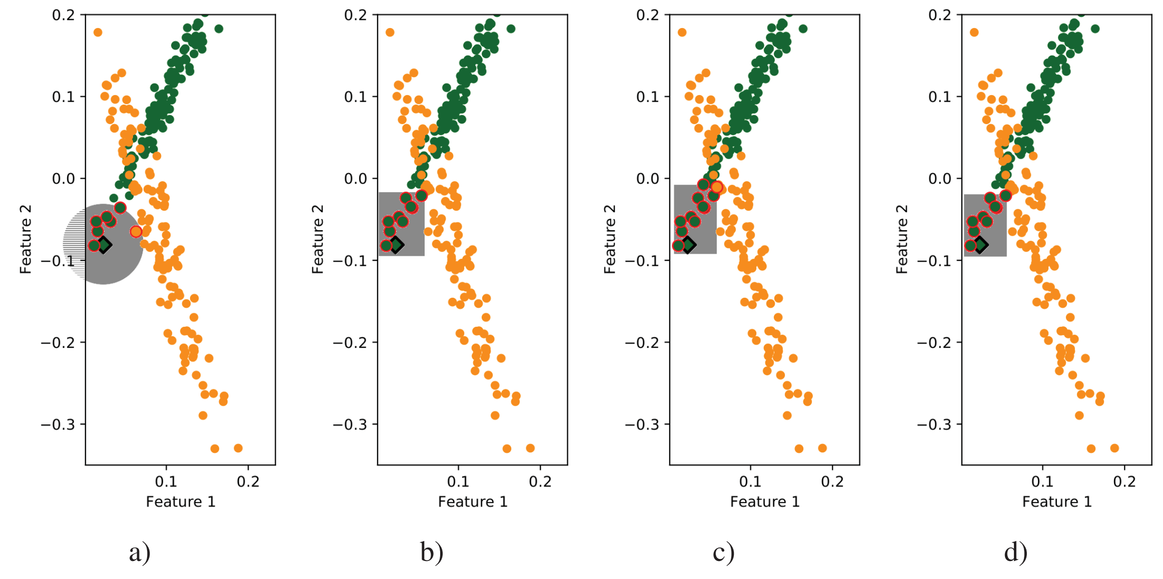


Figure 3.1 Two class two-dimensional toy problem. The training instances are shown in round markers, while the test sample is in a diamond shape. The region in grey was obtained using (a) the KNN classifier, (b) a Decision Tree, and (c-d) a Decision Tree with random feature sampling at each split (similar to the Random Forest), all fitted to the training set. The instances that belong to the target region in grey are shown in larger markers and highlighted in red.

In the 2-dimensional (2D) space, we can first observe the well known geometrical differences between the defined regions. While the KNN with Euclidean distance forms a (hyper-)sphere around the query, the trees partition the space into (hyper-)rectangular regions. The KNN rule also defines fixed-size regions, while the recursive partitions obtained from the trees can contain a variable number of instances, as we can see in Figure 3.1. Moreover, the feature sampling mechanism from the trees used in Figure 3.1(c-d) can yield different regions definitions, which may be useful for providing a distinct view of the target area.

Now we take the toy problem and generate a similar one, taking all the points and the two features, and augment the feature space by adding six redundant features (as random linear combinations of the original two informative features) and two random features (as random noise), yielding a 10-dimensional (10D) space. We use the two (original) features, which are also the only informative features in this 10D problem, to plot the data in Figure 3.2.

We fit the same four classifiers over the new training data, and observe the region definition yielded by each one of them. Because the 10-dimensional data is projected onto the same two informative features from the previous 2D problem, the points are located in the same places as in Figure 3.1, though now the region definitions can not be well depicted as in the 2D case. However, we still highlight in red the samples that belong to the target regions defined in the 10D space. It can be observed that the increase in dimensionality appears to have had a negative impact on the KNN's region definition, with most of the unknown instance's neighbors now being from an area where the opposite class is located, while the recursive partitioning performed by the trees yielded a more adequate target region for the query.

3.3 Proposed method

With that issue in mind, we now present the proposed method based on the OLP framework for high dimensional data. Instead of relying on the nearest neighbors rule, which may suffer from the effects of the curse of dimensionality, we propose in this work to leverage the data partitions

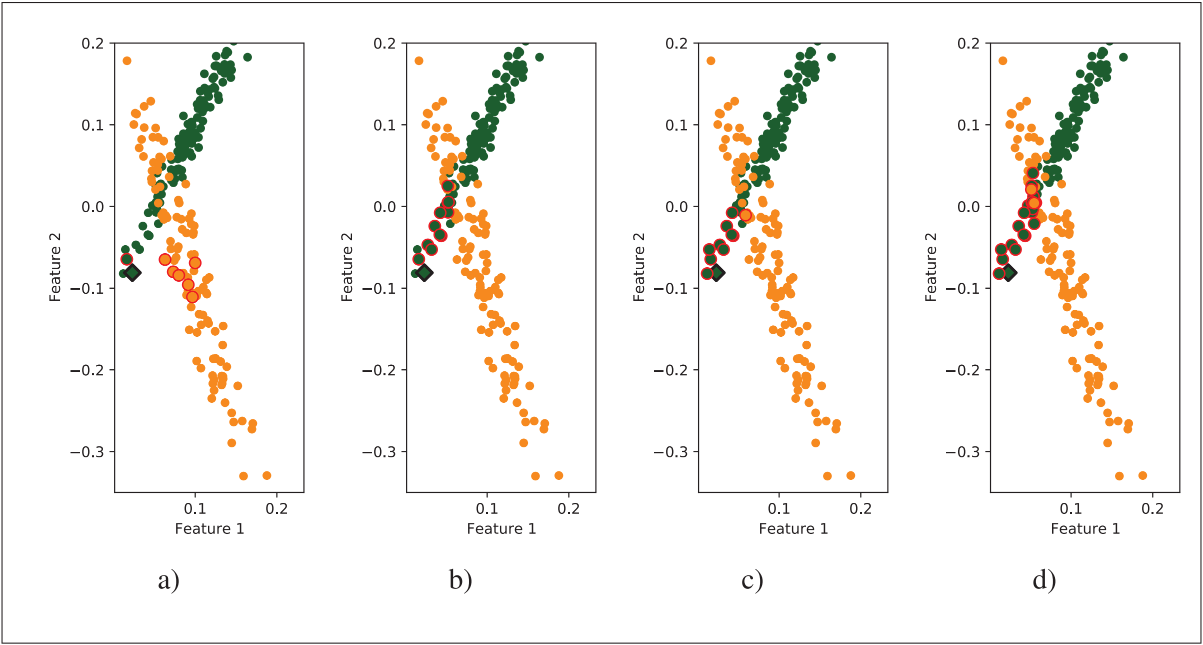


Figure 3.2 Two class 10-dimensional toy problem, projected onto its only two fully informative features. The training instances are shown in round markers, while the test sample is in a diamond shape. The target regions which contain the instances highlighted in red were obtained using (a) the KNN classifier, (b) a Decision Tree, and (c-d) a Decision Tree with random feature sampling at each split (similar to the Random Forest), all fitted to the 10D training set.

yielded by tree-based algorithms to define the regions of interest used for generating diverse local experts.

Table 3.1 Notation used in the proposed method's description.

Symbol	Meaning
$\mathcal{T} = \{(\mathbf{x}_n, y_n) \mid n \in \mathbb{N}, 1 \leq n \leq N\}$	Training set
$\mathcal{A} = \{A_j \mid j \in \mathbb{N}, 1 \leq j \leq T\}$	Set of decision trees
L	Number of tree levels, or local regions
k	Minimum leaf size
$\mathbf{x}_q$	Feature vector of query instance
$y_q, \hat{y}_q$	True and predicted labels of query instance
$\eta_{q,j} = [v_1, v_2, ..., v_\ell]$	Path/ordered list of nodes that $\mathbf{x}_q$ traverses in A_j
$\theta^l_{q,j} \subset \theta^{l-1}_{q,j} \subset ... \subset \theta^1_{q,j} = \mathcal{T}$	Data partition at the l-th node in path $\eta_{q,j}$
LP	Local pool
c	Base-classifier

Table 3.1 indicates the notation used in this work. An overview of the offline and online phases of the proposed technique is depicted in Figure 3.3. The offline phase is responsible for extracting information regarding the local overlap of the data. To that end, we fit a predefined number T of trees over the training set $\mathcal{T}$, as shown in Figure 3.3a, and use their data partitions to define the local regions in the feature space and identify borderline samples. Another hyperparameter of note in this phase is the minimum number of training instances within each leaf node. Since we intend to use the trees' nodes for local region definition, we set a minimum leaf size of k in each tree in order to avoid very small partitions.

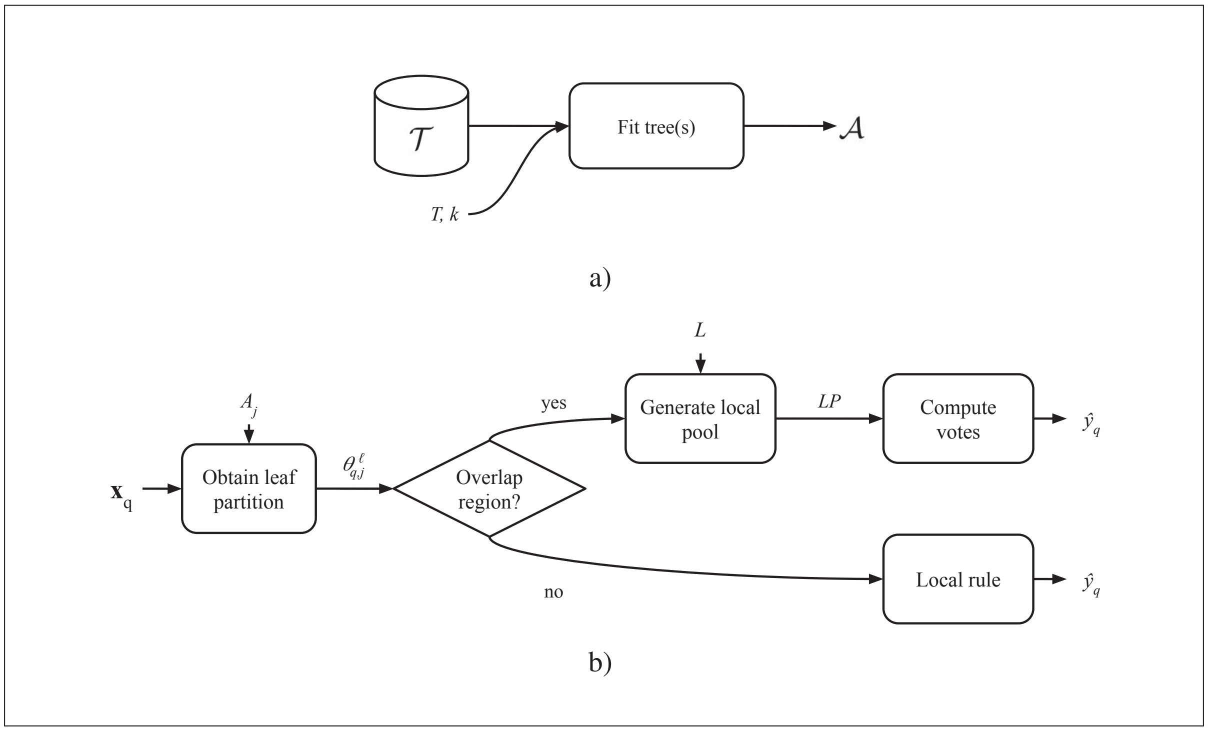


Figure 3.3 Overview of the (a) offline and (b) online phases of the proposed method. $\mathcal{T}$ refers to the training set and $A_j \in \mathcal{A}$ refers to one tree from a set of decision trees fitted to the data. T, k and L are hyperparameters of the technique which indicate the number of trees, minimum region size and number of regions/levels per tree, respectively. $\mathbf{x}_q$ is an unknown sample whose decision path through A_j leads to the data partition $\theta^{\ell}_{q,j}$ at the leaf node. If the partition is pure the system predicts the output $\hat{y}_q$ based on the local class distribution, otherwise the local pool LP is generated to obtain the prediction $\hat{y}_q$.

Figure 3.3b shows the general procedure of labelling a given unknown sample $\mathbf{x}_q$, considering only one tree A_j. For each query instance, a region θ_q around it is defined and analyzed in search of local class overlap. In the proposed technique, $\theta^{\ell}_{q,j}$ is the partition at the last position, or leaf node, of the decision path $\eta_{q,j}$ of $\mathbf{x}_q$ in A_j. Figure 3.4 provides a toy example of the multiple data partitions defined at the nodes within the decision path that an unknown sample traverses through a tree. If the leaf partition is pure, the sample is labelled according to the local rule. If not, however, we generate a set of linear classifiers to label the instance using L data partitions defined in the decision path, backtracking from the leaf node. The local pool (LP) is then used in a majority voting scheme to produce the final label of the sample. With more than one tree ($T > 1$), all generated classifiers and/or votes from the local rules are taken into account at the aggregation step for outputting the predicted label $\hat{y}_q$. The offline and online phases are explained in more detail next.

3.3.1 Offline phase

Algorithm 3.1 describes the offline phase of the technique, responsible for producing the tree structures which are used for the region definition. The inputs to the procedure are the training set $\mathcal{T}$, number of decision trees T and minimum leaf size k, and it returns the set of trees $\mathcal{A} = \{A_1, A_2, ..., A_T\}$ fitted to the training data. From lines 2 to 4, we train T trees over $\mathcal{T}$ with k as the minimum number of samples per leaf. If more than one tree is being generated, we train the T trees with random feature sampling at each node, akin to the Random Forest (RF) ensemble, in order to obtain different feature space partitions. This, together with the use of multiple partitions within the same decision path, introduces diversity into the region definition step. Since we mainly use the trees' partitions to produce the local linear classifiers, we consider these two sources of diversity to be enough for the region definition within the proposed technique and thus do not perform sample bootstrapping, as in the original RF.

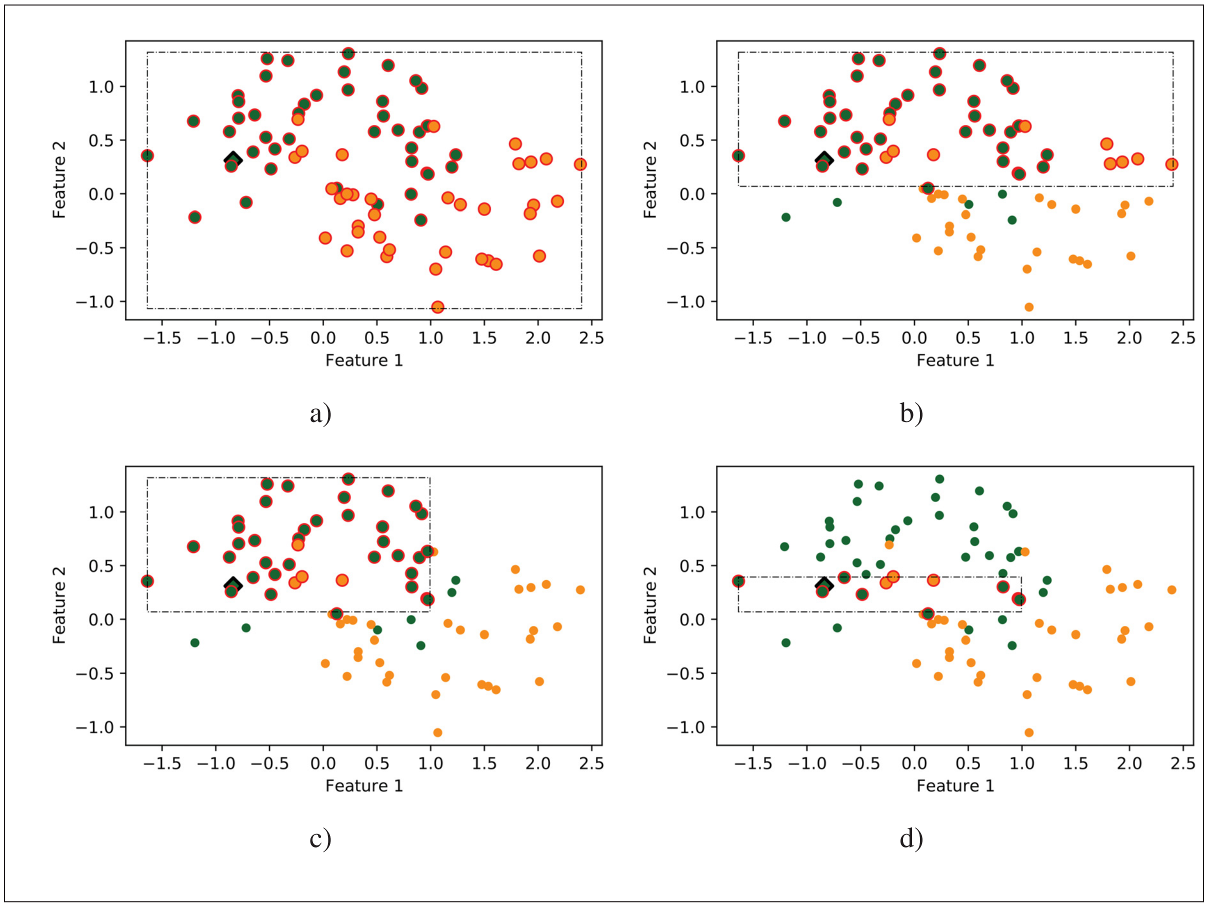


Figure 3.4 Toy example of the data partitions, within the dashed lines, from the decision path a given query sample (in a diamond shape) traverses in a tree A_j. (a) $\theta^1_{q,j} = \mathcal{T}$, from the (root) node in $\eta_{q,j}[1]$, (b) $\theta^2_{q,j}$, from the node in $\eta_{q,j}[2]$, (c) $\theta^3_{q,j}$, from the node in $\eta_{q,j}[3]$, and (d) $\theta^{\ell=4}_{q,j}$, from the (leaf) node in the last position $\eta_{q,j}[4]$.

Algorithm 3.1 Online Local Pool++ (OLP++) technique: offline phase.

```
  input  : $\mathcal{T}, T, k$ ;                    ▹ Training dataset, number of trees and min. leaf size
  output : $\mathcal{A} = \{A_j | 1 \leq j \leq T\}$ ;                    ▹ Set of $T$ trees
1 $\mathcal{A} \leftarrow \{\}$ for $j$ in $\{1, 2, ..., T\}$ do
2     $A_j \leftarrow fit\_tree(k, \mathcal{T})$ ;                    ▹ Fit tree over training set
3 end for
4 return $\mathcal{A}$
```

3.3.2 Online phase

Algorithm 3.2 describes the online phase of the technique. It receives as input the query sample $\mathbf{x}_q$, the set of trees $\mathcal{A}$ and the number of levels L, and it returns the predicted label $\hat{y}_q$ of $\mathbf{x}_q$. The local pool used to label the query sample is initialized empty (Line 1), as well as the predicted labels that will be accounted for in the final voting scheme (Line 2). Then, each tree in $\mathcal{A}$ is visited to obtain the neighborhood definitions and the label votes in the loop from Line 3 to Line 17. For each tree A_j, we first get the sequence of nodes $\eta_{q,j}$ (or decision path) the query takes from the root to the corresponding leaf (Line 4). We then obtain the position ℓ of the leaf node in $\eta_{q,j}$, which should be its last element. Afterwards, the set of training samples $\theta^{\ell}_{q,j}$ that fall at the leaf node ($\eta_{q,j}[\ell]$) of the decision path is obtained.

Algorithm 3.2 Online Local Pool++ (OLP++) technique: online phase.

```
   input  : $\mathbf{x_q}, \mathcal{A}, L$ ;                    ▹ Query sample, tree ensemble, and number of levels
   output : $\hat{y}_q$ ;                                       ▹ Label prediction for $\mathbf{x_q}$
1  $LP \leftarrow \{\}$ ;                                       ▹ Local pool
2  $Votes \leftarrow \{\}$ ;                                    ▹ Label votes
3  for every $A_j$ in $\mathcal{A}$ do
4      $\eta_{q,j} \leftarrow get\_decision\_path(\mathbf{x_q}, A_j)$ ;   ▹ Obtain the list of nodes $\mathbf{x_q}$ traverses in $A_j$
5      $\ell \leftarrow get\_length(\eta_{q,j})$ ;              ▹ Position of the leaf node in $\eta_{q,j}$
6      $\theta^{\ell}_{q,j} \leftarrow get\_partition\_from\_node(\eta_{q,j}[\ell])$ ;   ▹ Leaf node partition
7      for every $i$ in $\{1, 2, ..., L\}$ do
8          if $\{\exists \mathbf{x}_n, \mathbf{x}_m \in \theta^{\ell}_{q,j} | y_n \neq y_m\}$ ;   ▹ Leaf is non-homogeneous
9          then
10             $l \leftarrow \ell - i + 1$ ;                    ▹ Position in $\eta_{q,j}$ of the node $l-1$ levels above the leaf
11             $\theta^{l}_{q,j} \leftarrow get\_partition\_from\_node(\eta_{q,j}[l])$ ;   ▹ Data partition that falls into the $l$-th node
                 in $\eta_{q,j}$
12             $c \leftarrow SGH(\theta^{l}_{q,j})$ ;           ▹ Generate a hyperplane in local partition
13             $LP \leftarrow LP \cup c$ ;                      ▹ Add classifier to the local pool
14             $Votes \leftarrow Votes \cup c(\mathbf{x}_q)$ ;  ▹ Add classifier's prediction to votes
15         else
16             $Votes \leftarrow Votes \cup unique(\{y_n | \mathbf{x}_n \in \theta^{\ell}_{q,j}\})$ ;   ▹ Add label present in leaf partition to votes
17         end if
18     end for
19 end for
20 $\hat{y}_q \leftarrow mode(Votes)$ ;                         ▹ Majority voting
21 return $\hat{y}_q$
```

From Line 7 to Line 18 we follow the decision path level by level, from the leaf node, at position ℓ, until the note at position $\ell - L + 1$ in $\eta_{q,j}$. If the data partition at the leaf level $\theta^{\ell}_{q,j}$ is pure, the classifier generation is skipped and all the L votes from the tree goes to the label present in the leaf (Line 15). If not, we obtain the position l in $\eta_{q,j}$ of the node to be used for generating a classifier from the local pool. The position l is updated in each iteration, backtracking in the decision path to use the different region definitions from the nodes in $\eta_{q,j}$. Then, the region $\theta^{l}_{q,j}$, or data partition, defined at the l-th node in $\eta_{q,j}$ is obtained in Line 10. The Self-generating hyperplanes (SGH) method (Souza *et al.*, 2017) (Line 11) is then used to produce a linear classifier over the region $\theta^{l}_{q,j}$. The SGH is a generation method that produces a set of two-class hyperplanes iteratively so that, for each input training sample, at least one classifier in the final pool is able to correctly label it. We chose the SGH method to generate the classifiers as it produces weakly fitted linear rules in very few iterations, which provides an advantage in terms of time while possibly reducing the chance of overfitting, given the small number of instances in the local region. A more in-depth introduction to the SGH method can be found in the supplementary material. We take the first hyperplane c the SGH generates and add it to the local pool LP (Line 12). The predicted label $\hat{y}$ of $\mathbf{x}_q$ given by c is added to the voting scores in Line 13. Lastly, in lines 19-20, the most frequent label within the predicted labels is returned as the output class of the system.

3.3.3 Illustrative example

To better visualize the proposed technique we use a two-class 2D synthetic dataset, the P2 problem (Valentini, 2005), whose training set with 1600 samples, 800 from each class, is shown in Figure 3.5a. The theoretical border of the problem is depicted with the dashed lines. Moreover, a small percentage (2%) of label noise was added to the training set.

Figure 3.5 2D problem used to illustrate the proposed method. (a) shows the training set, with added noise, and (b) the same set with the decision tree's classification border in purple. The theoretical class borders are dashed.

To simulate the proposed method's steps, we use a single decision tree ($T = 1$), $k = 7$ and $L = 3$. So, in the offline phase, the training set from Figure 3.5a is used as input to a decision tree, which yields the class borders shown in purple in Figure 3.5b.

Table 3.2 Description of the decision paths obtained for the sample depicted in (a) Figure 3.6b and (b) Figure 3.6c

(a)

Decision path : $\mathbf{x}_q$ = [0.0683, 0.0960]
Node 0 : (Feature 2 $\leq$ 0.1826)
Node 1 : (Feature 1 $\leq$ 0.8429)
Node 2 : (Feature 1 $\leq$ 0.2827)
Node 3 : (Feature 2 $\leq$ 0.1246)
Node 4 : (Feature 2 $\leq$ 0.1079)

(b)

Decision path : $\mathbf{x}_q$ = [0.3021, 0.5086]
Node 0 : (Feature 2 > 0.1826)
Node 24 : (Feature 1 $\leq$ 0.3244)
Node 25 : (Feature 1 > 0.0539)
Node 37 : (Feature 2 > 0.5027)
Node 47 : (Feature 2 $\leq$ 0.7947)
Node 48 : (Feature 1 > 0.1727)
Node 56 : (Feature 1 > 0.2291)
Node 60 : (Feature 2 $\leq$ 0.693)
Node 61 : (Feature 1 > 0.2600)
Node 63 : (Feature 2 $\leq$ 0.5772)

In the online phase, we follow the two query samples shown in Figure 3.6a, depicted in a diamond shape with black contour and shown individually in Figure 3.6b-c. The coordinates and the decision paths each of these samples take within the tree are detailed in Table 3.2. The query from Figure 3.6b belongs to the orange class, while the query from Figure 3.6c belongs to the green class. The training samples that are contained in the leaf node where each unknown instance falls is highlighted in red, with the local region itself shown in light grey. As we can see, the query in Figure 3.6b (from the orange class) falls into a pure leaf, thus, all $L = 3$ votes from this tree goes towards the orange class. Since there is only one tree in this example, the output label of the system for that sample is the orange class as well. This behavior would be similar in the OLP, whose region defined with the KNN, shown in dark grey in Figure 3.6b, would also only contain samples from the orange class, though in much smaller numbers. The query in Figure 3.6c, however, falls into a non-homogeneous partition containing 6 samples from the orange class and only one sample from the green class, which is the correct label of this instance. Thus, the local pool will be generated to label it, leveraging the partitions already

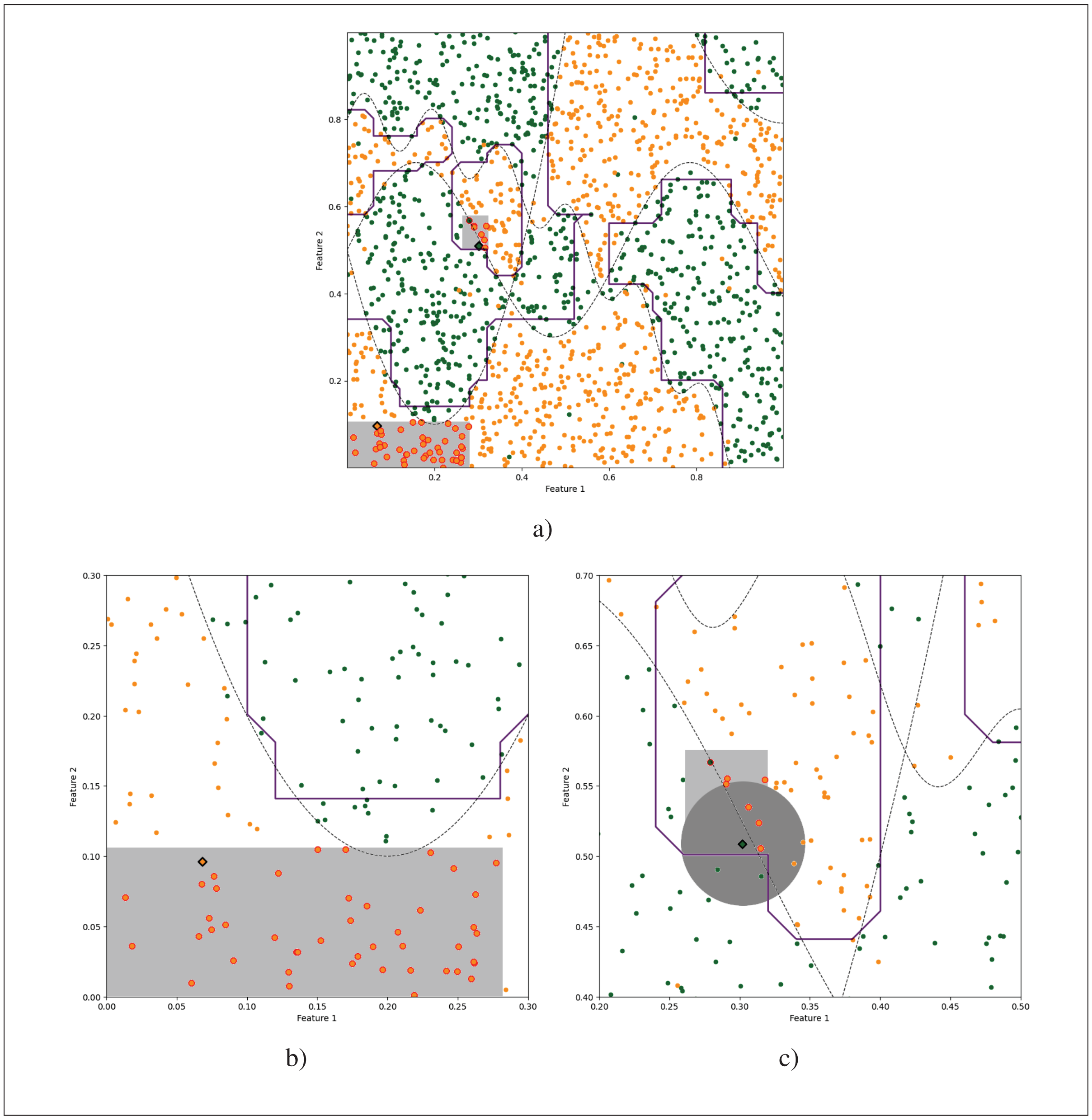


Figure 3.6 Two examples of query instances, depicted in a diamond shape and highlighted in black. The training instances in larger markers and highlighted in red belong to the rectangular partitions defined at the leaf node where the unknown samples fall, shown in light grey. (b) shows a query sample from the orange class in a homogeneous partition, while (c) shows a query sample from the green class in a heterogeneous partition. The circular regions depicted in dark grey in (b-c) show the region definition that would be obtained for the same instances using the KNN, with $K = 7$.

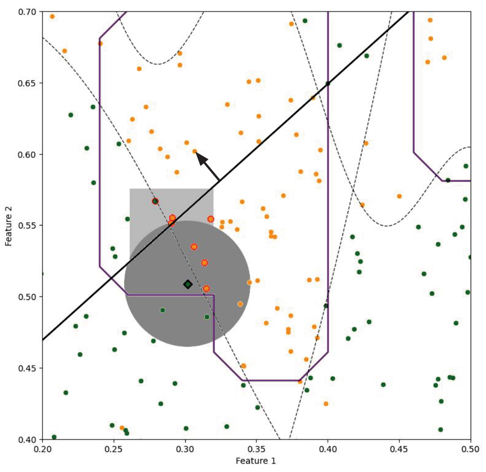
defined in the decision tree starting from the leaf level. The region definition given by the KNN, shown in dark grey, contains the same amount of instances in this case, though with a different class distribution.

Figure 3.7 shows the three classifiers from the local pool (LP) that were generated to label the query instance from Figure 3.6b. The arrows indicate where the classifiers label as the green class. First, the data partition defined at the first visited node, the leaf node, is shown in red over the local region in light grey in Figure 3.7a. These data points are used to generate a linear classifier using the SGH method, depicted in Figure 3.7a, which is added to the local pool. We can see that this first hyperplane is still unable to correctly classify the query.

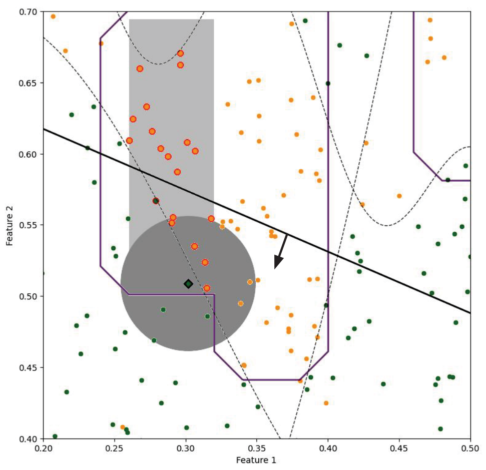
In the second iteration, the samples belonging to the data partition defined at the level immediately above the leaf level, according to the decision path from Table 3.2b, are used as local region as input to the SGH method. These samples are highlighted in red in Figure 3.7b, and the region defined at that level is shown in light grey. We can see that the green class is in a greater numerical disadvantage in this partition, though the locality reduction seem to have provided more context w.r.t. the orange class local distribution. We can also observe that the region that would be used in the original OLP, shown in dark grey, is already much smaller compared to the one in the proposed adaptation. The first hyperplane the SGH produces in the rectangular region is shown in Figure 3.7b, now able to label the query correctly.

At the last visited level, which corresponds to two levels above the leaf level in the decision path (Table 3.2b), the partition defined at that node is shown in Figure 3.7c. We can see that the green class is more well-represented at the local border. This neighborhood is used as input to the SGH method, and the produced classifier, shown in Figure 3.7c, is also able to classify the query instance correctly. The linear classifier is then added to the local pool, and in the aggregation step, the system outputs the green class with two votes to one, which is the sample's correct label.

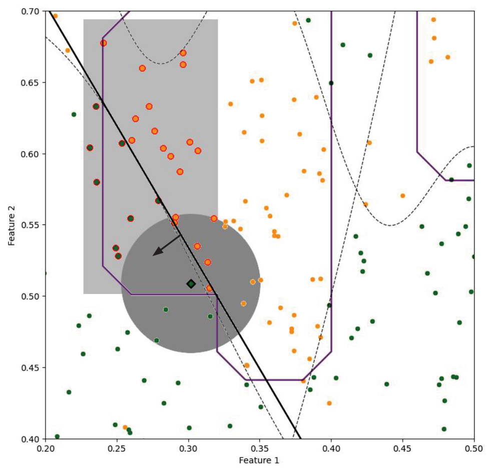
It should be noted that our goal with this example is to help visualize the proposed technique, which is based on the OLP but focused on dealing with high dimensional data, and illustrate its



a)



b)



c)

Figure 3.7 Demonstration of the locality reduction procedure and resulting local linear rules given by the local pool obtained for the query sample from Figure 3.6c. The instances in larger markers and highlighted in red belong to the partitions defined at the (a) leaf level, (b) level immediately above the leaf level, and (c) the level above that in the decision path taken by the query sample in the fitted tree. The regions used for the local pool generation within the proposed adaptation are shown in light grey, while the regions obtained by the KNN within the original OLP are shown in dark grey. The depicted hyperplanes form the local pool and were generated using the SGH method over the rectangular region. The arrows indicate the area of the feature space where the classifiers label as the green class.

procedures. An arguably simpler classification rule, as well as the OLP itself, would probably be able to label correctly the samples shown in Figure 3.6 in this two-dimensional feature space.

3.4 Related work

MCSs are usually divided into three consecutive phases (Cruz *et al.*, 2018a): generation, in which the pool of base-classifiers is produced, selection, an optional step in which a subset of the pool is singled out to perform the classification task, and aggregation, in which the responses of the (selected) ensemble are joined to obtain the final output. Exploiting the local expertise of the base-classifiers in an ensemble is a quite common approach in MCSs, often based on the idea that the models' distinct behaviors are associated with their competence over certain areas of the feature space. Thus, the way the system responds in generalization varies according to the region a given query instance is located, which often leads to their characterization as *dynamic* techniques. In fact, local ensemble methods span several subareas of the MCS literature, including Mixture of Experts (ME) (Jacobs *et al.*, 1991; Armano & Hatami, 2010b), Adaptive Splitting and Selection (AdaSS) (Jackowski & Wozniak, 2009; Lopez-Garcia *et al.*, 2019) and classical Dynamic Selection (DS) techniques (Kuncheva, 2000; Woods *et al.*, 1997), as well as some hybrid approaches, each of which with a distinct way of incorporating the local information into the system. The ME and AdaSS domains are both based on the divide and conquer principle, which attempts to simplify a complex problem via splitting the feature space into easier subproblems. In classical ME, which is defined for neural network ensembles only, the base-classifiers and the gating network are trained together in a way that encourages the localization of the models (Jacobs *et al.*, 1991), while in the AdaSS, the data partitions and their corresponding assignment of base-classifiers are optimized jointly via an evolutionary algorithm (Jackowski & Wozniak, 2009). On the other hand, while classical DS techniques and some hybrid approaches may be based on the divide and conquer idea (Zhu *et al.*, 2019; Armano & Tamponi, 2018; Kuncheva, 2000), the feature space partitioning is performed decoupled from the other steps such as pool generation and selection, in the sense that one can obtain such region definitions individually without changing the final ensemble structure. As the proposed method fits within the latter category, we focus on these techniques in this work while referring the reader to (Masoudnia & Ebrahimpour, 2014; Jackowski & Wozniak, 2009) for more context on the ME and AdaSS approaches.

While the way the local information is embedded into the MCS varies greatly among these techniques, they usually require the definition of one or several regions of interest, which can be done by means of roughly four approaches: clustering (Verma & Rahman, 2011; Soares *et al.*, 2006; Kuncheva, 2000), nearest neighbors rule (Souza *et al.*, 2019b; Woods *et al.*, 1997; Ko *et al.*, 2007), potential function model (Armano & Tamponi, 2018; Woloszynski & Kurzynski, 2011), and recursive partitioning (Zhu *et al.*, 2019; Biedrzycki & Burduk, 2020). With exception of the latter, the use of a dissimilarity metric, usually the Euclidean distance, is quite common in the region definition process, though it is not always applied on the original feature space (Cavalin *et al.*, 2012). As local methods, the region definition process has a considerable impact on the response of these MCS (Cruz *et al.*, 2018b; Soares *et al.*, 2006). However, even though the negative effects of high dimensionality over local-based methods has often been observed in a wide variety of areas and applications, for instance in recent works on local explanation methods over industrial data (Fries & Rydén, 2022) and disease diagnosis over biomedical data (Yan, Li, Ma, Liao, Luo, Wang & Luo, 2022), the possible degradation associated with the region definition step of local ensemble methods on high dimensional spaces has not been investigated yet in this literature.

Another distinction can be done in terms of when the local information is taken into account, which can happen in each one (or several) of the MCS phases. Ensembles based on the divide and conquer principle tend to integrate the region definition in the generation phase (Verma & Rahman, 2011; Armano & Tamponi, 2018; Zhu *et al.*, 2019), so that each member of the pool can specialize over a part of the feature space, therefore encouraging ensemble diversity through localization. They also usually apply a dynamic selection or aggregation rule in order to assign a higher importance to the base-classifiers that were encouraged to learn over a given area. In the Space Partition Tree (SPT) framework, proposed in (Zhu *et al.*, 2019) for dealing with imbalanced problems, the feature space is recursively partitioned on the axis of the maximum within-class scatter of the majority class until one of the stopping criteria is met. Then, a cost-sensitive Support Vector Machine (SVM) is trained over each region defined at the leaves of the tree-based structure, and for a given query instance the single SVM assigned to the leaf it falls in is used to classify it. The Forest of Local Trees ensemble (FLT) (Armano & Tamponi, 2018) localizes the base-classifiers by sequentially selecting random prototypes as centroids to the defined regions, in such a way that the probability of selecting a given instance as centroid is updated according to its distance to the currently selected centroid. Each centroid is then used to define a distance-based weighting function for the training samples during the fitting of the model associated with that centroid. So, by encouraging the distance between the centroids, each classifier specializes in different but overlapping areas of the feature space. In generalization, the classifiers' responses are weighted according to the distance between their designated centroid and the test sample.

Another related work of note, which uses the local information at the generation step and is reminiscent of the proposed method, is (Do, 2015), in which the author proposes to improve the local rules defined at the leaves of a RF ensemble by fitting a linear SVM on them and using the linear rules instead. The OLP (Souza *et al.*, 2019b) also integrates the local information in the generation phase, since all classifiers are produced around the target area, defined using the nearest neighbors rule applied to the query sample with different region sizes. However, as the entire pool is meant to contain only local experts on the region, the consensus is expected to be quite high and so the aggregation is performed using a simple majority voting rule.

If taking into account the local information at the selection phase, dynamic selection techniques assume the base-classifiers in the ensemble are local experts in different areas of the feature space, so they attempt to identify which ones to use in each defined region by estimating their competence there. Each DS technique defines competence according to one or multiple criteria, which may include local accuracy (Woods *et al.*, 1997; Ko *et al.*, 2007), ensemble diversity (Soares *et al.*, 2006), classifier behavior (Giacinto *et al.*, 2000; Cavalin *et al.*, 2012), among others. After assessing the competences of the classifiers in the target region, a subset of the whole ensemble is used to label the given query instance according to a defined aggregation rule. Distance-based dynamic weighting approaches are often applied at the aggregation phase in order to assign different importance levels to the base-classifiers, as in the FLT. A different local aggregation approach was proposed in (Biedrzycki & Burduk, 2020) for an ensemble of decision trees trained over random disjoint samples of the training data. After a feature selection step, the trees are fit and the space is partitioned into regions obtained from each split of every tree in the ensemble. A local rule is then defined for each subspace based on the trees' decisions on both that subspace and its neighboring subspaces, weighted by the distances between the midpoints of the defined regions.

Table 3.3 Summary of the related work cited in this section, including their region definition approach, the amount and size of the defined target region(s), and at which phases the local information is incorporated into the system.

Name	Approach	Target region	Size	Phase of MCS	Reference
Cluster-oriented ensemble classifier	Clustering	Single	Inconstant	Generation, Aggregation	(Verma & Rahman, 2011)
Clustering and selection	Clustering	Single	Inconstant	Selection	(Kuncheva, 2000)
Clustering with accuracy or diversity-based selection	Clustering	Single	Inconstant	Selection	(Soares *et al.*, 2006)
K-nearest oracles union	KNN	Single	Constant	Selection, Aggregation	(Ko *et al.*, 2007)
K-nearest oracles eliminate	KNN	Single	Inconstant	Selection	(Ko *et al.*, 2007)
K-nearest output profiles	KNN	Single	Constant	Selection, Aggregation	(Cavalin *et al.*, 2012)
Online local pool	KNN	Multiple	Multi-scaled constant	Generation, Selection	(Souza *et al.*, 2019b)
Randomized reference classifier DS	Potential function	Single	Constant	Selection	(Woloszynski & Kurzynski, 2011)
Forest of local trees	Potential function	Multiple	Constant	Generation, Aggregation	(Armano & Tamponi, 2018)
Space partition tree	Recursive partitioning	Single	Inconstant	Generation, Selection	(Zhu *et al.*, 2019)
Decision trees integration via dynamic regions	Recursive partitioning	Multiple	Inconstant	Aggregation	(Biedrzycki & Burduk, 2020)
Random forests with local rules	Recursive partitioning	Multiple	Inconstant	Generation	(Do, 2015)
Proposed	Recursive partitioning	Multiple	Multi-scaled inconstant	Generation	

Table 3.3 summarizes some key characteristics of the works on local-based MCS discussed in this section, indicating their mechanism for region definition and the type of target region(s) utilized, as well as the phases at which the local information is incorporated into the system. Techniques that may take into account a single defined region or multiple defined regions in generalization, are shown in the *Target region* column. Moreover, the target regions' size may be fixed or may vary from query instance to query instance, as identified in the *Size* column by

the *Constant* or *Inconstant* characteristics, respectively. In addition to that, the guided target region size increase, characteristic of the OLP and the proposed method, are indicated with *Multi-scaled*. Thus, since the multi-scale region definition in the OLP is fixed for all samples, its target region size is described as *Multi-scale constant*, while the proposed methods' variable partition sizes lead to the *Multi-scale inconstant* characteristic.

Apart from the OLP, whose difference to the proposed technique was already discussed, the two methods that resemble the most the proposed approach are the Random forests with local rules (Do, 2015) and the SPT (Zhu *et al.*, 2019). In the former, the local classifiers trained over the leaves of a RF ensemble to improve their local decision rule. In our case, however, we do not specifically intend to improve the trees' decision rules. Rather, we make use of the partitions defined at several nodes, including but not restricted to the leaves, in order to generate, as in the OLP, a pool of linear classifiers that have a diversely local view of the problem. W.r.t. the SPT framework (Zhu *et al.*, 2019), while the technique uses similar concepts to the proposed method, the way they are applied and their aim are quite different. In the SPT, the splits in the feature space are done in the axis of maximum scatter of the majority class subset, with the purpose of recursively dividing it so that the imbalance ratio in each final region (or leaf) the local class imbalance is smaller than the whole problem's imbalance. The proposed technique, on the other hand, uses the partitions given by a regular tree-based algorithm to obtain regions of increasing size around each unknown sample in order to fit experts in the local area without the use of a distance metric in high dimensional spaces. Moreover, in the SPT the classifiers in the pool are trained in disjoint partitions of the data, one for each leaf level region, and in generalization only the classifier trained in the region that an unknown instance is located labels it. In the proposed method, as the entire local pool is produced in the target area, their responses are combined so that their diversity, which is subject to the local region, can be leveraged in the decision rule.

3.5 Experiments

3.5.1 Experimental setup

3.5.1.1 Datasets

Table 3.4 presents the characteristics of the datasets used in the experiments. All 39 two-class high dimensional datasets, each of which presenting at least 100 features, were obtained from the OpenML repository (Vanschoren *et al.*, 2013), more specifically from the study on the impact of feature selection for classification (StudyID: 15). The column IR in Table 3.4 refers to the problems' imbalance ratio, that is, the ratio between the amount of majority class samples and the amount of minority class samples in the dataset. The performance evaluation was conducted using a 10-fold cross validation procedure, with one fold for test and the remaining for training, using the same partitions provided in the OpenML repository for reproducibility.

Table 3.4 Characteristics of the datasets used in the experiments.

Dataset	# Instances	# Features	IR	Dataset	# Instances	# Features	IR
AP_Endometrium_Breast	405	10935	5.64	OVA_Ovary	1545	10935	6.8
AP_Breast_Omentum	421	10935	4.47	AP_Breast_Prostate	413	10935	4.99
AP_Prostate_Uterus	193	10935	1.8	AP_Omentum_Uterus	201	10935	1.61
AP_Omentum_Lung	203	10935	1.64	leukemia	72	7129	1.88
OVA_Breast	1545	10935	3.49	tumors_C	60	7129	1.86
AP_Colon_Prostate	355	10935	4.14	gina_agnostic	3468	970	1.03
AP_Lung_Uterus	250	10935	1.02	gina_prior	3468	784	1.03
AP_Colon_Kidney	546	10935	1.1	scene	2407	299	4.58
AP_Endometrium_Prostate	130	10935	1.13	mfeat-pixel	2000	240	9
OVA_Uterus	1545	10935	11.46	mfeat-factors	2000	216	9
AP_Prostate_Kidney	329	10935	3.77	musk	6598	168	5.49
AP_Colon_Omentum	363	10935	3.71	tecator	240	124	1.35
AP_Omentum_Kidney	337	10935	3.38	yeast_ml8	2417	116	70.09
OVA_Endometrium	1545	10935	24.33	sylva_prior	14395	108	15.25
AP_Endometrium_Lung	187	10935	2.07	spectrometer	531	101	8.65
AP_Colon_Ovary	484	10935	1.44	fri_c4_250_100	250	100	1.27
AP_Omentum_Ovary	275	10935	2.57	fri_c4_100_100	100	100	1.13
AP_Breast_Kidney	604	10935	1.32	fri_c4_500_100	500	100	1.3
AP_Ovary_Uterus	322	10935	1.6	fri_c4_1000_100	1000	100	1.29
AP_Uterus_Kidney	384	10935	2.1				

3.5.1.2 Performance evaluation

We evaluate the models in this work in terms of the balanced accuracy rate (that is, the macro-averaged recall). The balanced accuracy rate for binary problems is defined as the average between the true positive rate (TPR) and the true negative rate (TNR), as shown in (3.1). As recommended in (Flach, 2019), we focus on one performance metric only, and the choice of the balanced accuracy rate was due to the widely varying imbalance ratios of the problems in the test bed, which range from 1.03 to 70.09.

$$\text{Balanced accuracy rate} = \frac{TPR + TNR}{2} \tag{3.1}$$

For the statistical comparisons between the models' performances over multiple datasets, we also use the pairwise Wilcoxon signed-rank test, as recommended in (Demšar, 2006; Benavoli *et al.*, 2016), since it is non-parametric and its result does not depend on the group of techniques included in the comparative analysis. In addition to that, pairwise comparisons between the performances obtained over the folds of each dataset are done using the Kruskal-Wallis non-parametric test, as in (Cruz *et al.*, 2017a).

3.5.2 Comparison against the OLP

In this section, we perform a comparative analysis between the OLP and the proposed method considering a small local pool size. The purpose of this analysis is to investigate how the different locality definition approaches impact the resulting ensemble, and whether the proposed technique can provide an improvement to the OLP over high dimensional datasets while still yielding some of its desirable characteristics. To that end, we analyze back-to-back several aspects yielded by both techniques, including the borderline sample detection, the local region size, the pool diversity, the average performance and the time complexity.

We evaluated the OLP and the proposed technique with $|LP| = 3$ local classifiers. Within the OLP we used, as in previous works (Souza *et al.*, 2019b,a), the following DCS techniques:

Overall Local Accuracy (OLA) (Woods *et al.*, 1997), Local Class Accuracy (LCA) (Woods *et al.*, 1997) and Multiple Classifier Behavior (MCB) (Giacinto *et al.*, 2000). We chose a small pool size due to several reasons: first, to allow a level-by-level analysis of both techniques; second, to control the locality defined in the tree nodes as it may present impure leaves on shallow branches; and third, to perform a fair comparison against the OLP since its usually recommended number of classifiers is also very small. Due to the chosen pool size, the local regions in the proposed adaptation are obtained from the nodes of a single decision tree ($T = 1$), with the minimum number of samples in each leaf being set to $k = 7$, which is also the initial size of the local region in the OLP, defined using the regular nearest neighbors rule.

3.5.2.1 Borderline sample detection

As the local pool is only generated for queries in class overlap regions, the way the latter are defined has a direct impact on the border detection and response of the system. Figure 3.8 shows the average proportion of test samples considered in borderline regions by the OLP and the proposed method for each dataset, sorted by increasing number of features. It can be observed that except for a few datasets, namely *AP_Breast_Prostate*, *AP_Prostrate_Kidney* and *AP_Colon_Prostate*, the local regions defined by the KNN were much more susceptible to present samples from different classes compared to the DT-defined local regions. While the optimization of the Gini index in the tree's data partitioning could have had a role in the comparatively lower ratio of borderline samples detected, the very high number of samples in regions considered to be close the border as defined by the KNN suggests that the distance-based detection was often negatively impacted due to the high data dimensionality. That is to say, in certain datasets, the KDN measure and the local region definition obtained using the nearest neighbors rule in the original feature space appears to have led the OLP technique to misidentify many safe instances as borderline ones, likely due to the curse of dimensionality effects, such as distance concentration (François *et al.*, 2007; Radovanovic, Nanopoulos & Ivanovic, 2010).

However, as both approaches perceive local class overlap differently, distinct data complexity profiles would yield different behaviors. For instance, we can observe that the DT obtained over the *musk*, *AP_Endometrium_Prostate* and *AP_Prostate_Uterus* datasets had only pure leaves, while the nearest neighbor rule in the *AP_Breast_Prostate* dataset did not identify any borderline query sample. Taking into account both approaches w.r.t. local region definition, though, we can see that some problems do appear to present a comparatively higher proportion of borderline samples, such as the datasets *tumors_C*, *fri_c4_100_100*, *fri_c4_250_100*, *fri_c4_500_100*, *fri_c4_1000_100*, *AP_Omentum_Ovary*, *AP_Ovary_Uterus*, *leukemia*, and *gina_agnostic*.

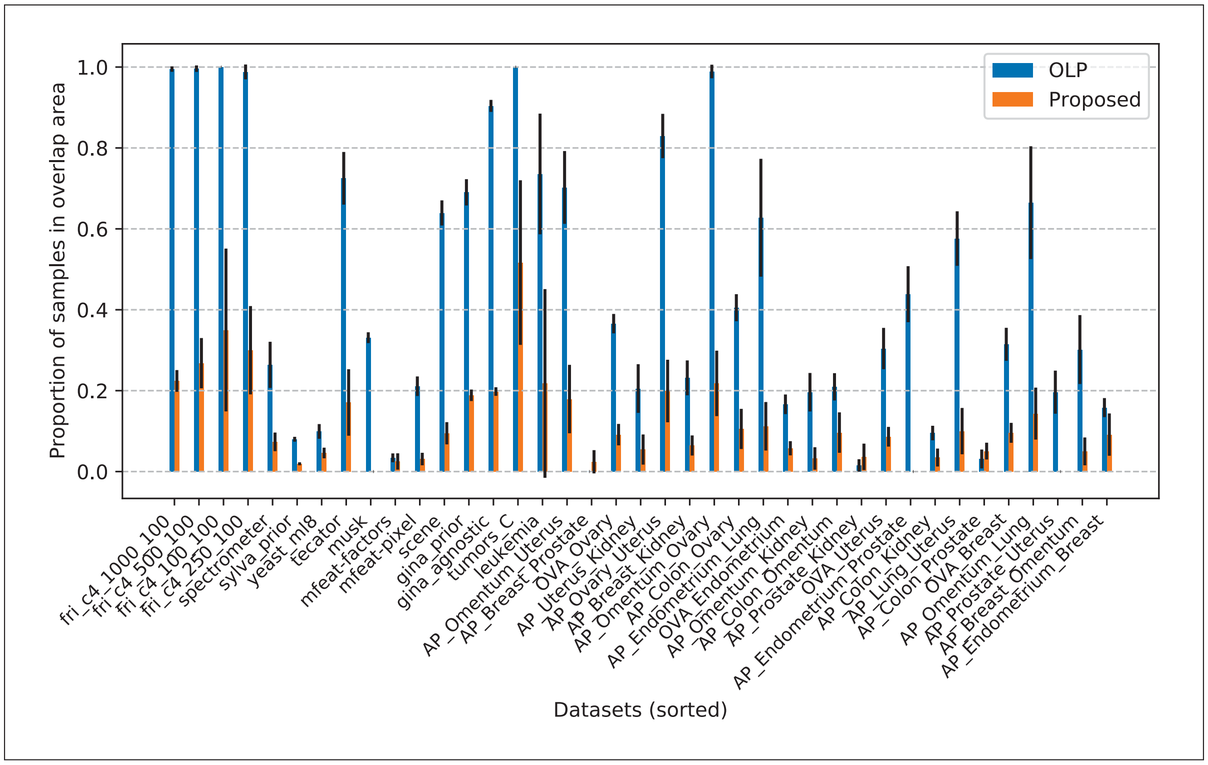


Figure 3.8 Mean and standard deviation of the proportion of test samples considered in borderline areas by the OLP and the proposed method for each dataset from Table 3.4, sorted by increasing number of features.

3.5.2.2 Region size

Figure 3.9 shows the average proportion of training samples in the regions of the detected borderline test instances. We compute the region size per iteration of the local pool generation procedure, with the first one referring to the smallest defined region (with highest locality) and the last one referring to the largest defined region (with lowest locality). We can see that for the regions defined with the highest locality (Figure 3.9(a)), the amount of samples in the neighborhood of the query instances is somewhat similar for both the OLP and the proposed method, likely due to the tree's minimum leaf size which is set to the smallest k value used in the OLP. However, as the OLP's region size is controlled directly in terms of number of instances, the proportion of samples in the local region increases only slightly in the second and third iterations (Figure 3.9(b-c)) whilst the proposed technique's regions increase rapidly with each node jump. In fact, the locality at the third iteration is greatly reduced on the majority of the datasets. In some cases, such as *AP_Breast_Prostate*, *AP_Colon_Prostate* and *AP_Prostate_Kidney*, the average proportion of samples in the region is close to 1.0, which may be an indication that most test instances fall into shallow branches.

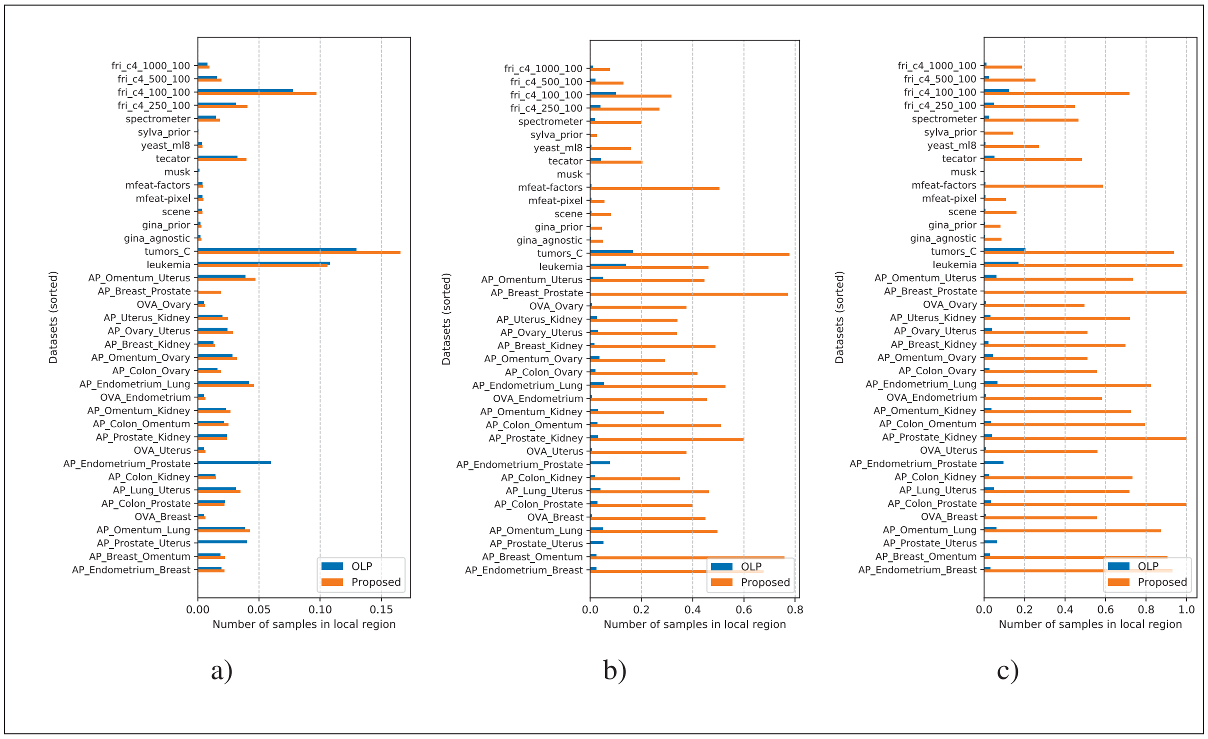


Figure 3.9 Average proportion of training samples in the local region of the test instances considered in class overlap areas at the (a) first, (b) second and (c) third iteration.

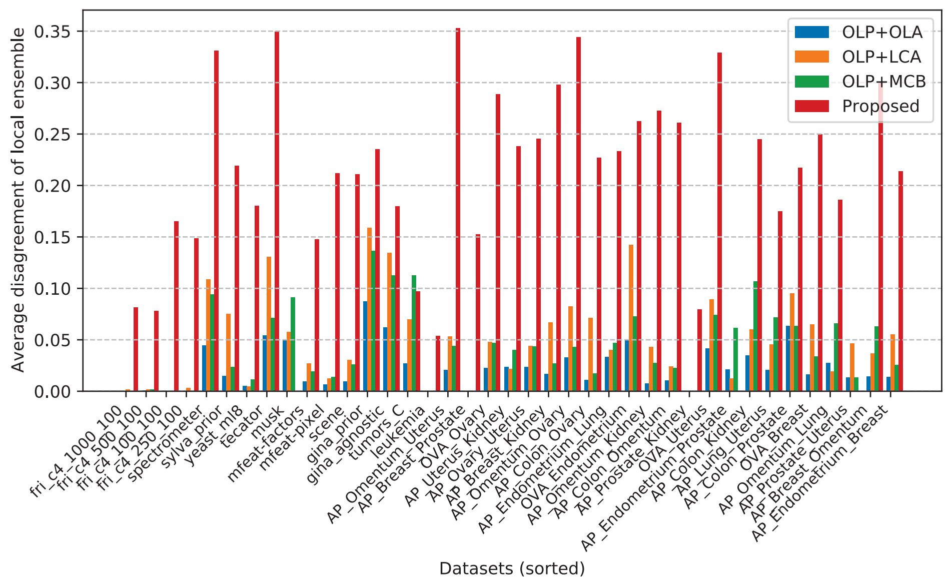
3.5.2.3 Local pool diversity

We now investigate how diverse the classifiers in the local pool are for both techniques. To that end, we calculate the average disagreement (Skalak, 1996) of the classifiers in the local pool used to label each test instance. The disagreement is a pairwise diversity metric defined as the proportion of samples on which only one of the two classifiers are correct. The average disagreement score for binary problems is calculated according to Equation 3.2, in which c_i and c_j are two classifiers and $(\mathbf{x}, y)$ is a pair of features-label of a given sample in the set D. In our analysis, we calculate the disagreement score over the training samples included in the first defined local region (that is, with the highest locality). Since the disagreement score is calculated only with respect to the classifier's outputs, and the local pool is composed of linear classifiers, we also calculate the cosine distance between them in order to improve the characterization of their differences. The cosine distance is computed as $1 - \cos\beta$ where β is the angle between the hyperplanes defined by two linear classifiers. We average the pairwise cosine distance over all pairs of classifiers in the local pool used to label each query instance, and then take the mean over each test fold.

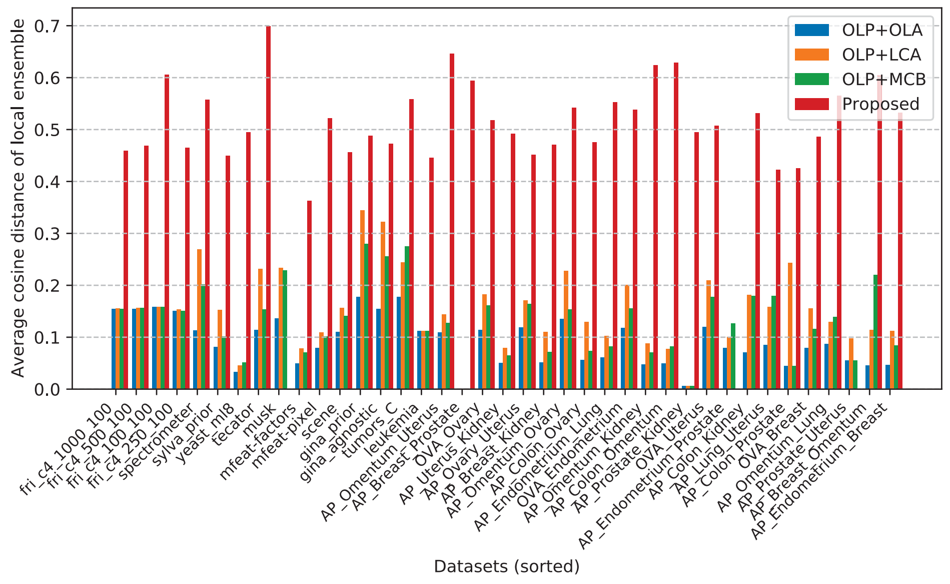
$$\text{disagreement}(c_i, c_j, D) = \frac{|\{(\mathbf{x}, y) \in D : c_i(\mathbf{x}) \neq c_j(\mathbf{x})\}|}{|D|} \tag{3.2}$$

In terms of disagreement, we can observe that the local ensemble defined in the OLP had a high degree of consensus, with the average disagreement score only reaching 0.1 over a few datasets when using LCA. The proposed technique, on the other hand, presented a much higher average disagreement score as the locality reduction is done at an increased rate compared to the OLP, reaching an average score above 0.3 on 5 datasets. Averaged over all datasets, as shown in Table 3.5, the OLP yielded a disagreement score of 0.052, using LCA, while the proposed method obtained an average disagreement of 0.2183.

With regards to the cosine distance, we can see from Figure 3.10b that OLP also presented a quite low average cosine distance between the local linear classifiers, reaching an average above



a)



b)

Figure 3.10 Average (a) disagreement score and (d) cosine distance between the linear classifiers included in the local pool in the OLP, with each of the three DCS techniques, and in the proposed method.

0.3, which represents an angle of $\beta = 45°$, over two datasets only, *gina_agnostic* and *gina_prior*. In the case of the proposed method, it can be observed that the cosine distance between the classifiers in the local pool is much higher, as expected given the larger amount of instances that are included at each tree level. Over all datasets, the average cosine distance from the OLP with LCA (0.1567) represents an angle $\beta = 32.5°$, which intuitively seems as a slight difference in the defined linear border. The proposed method, on the other hand, yielded an average cosine distance of 0.5170, which represents an angle of $\beta = 61.12°$, much higher than the OLP but still far from opposing decision rules.

Table 3.5 Average disagreement score and cosine distance of the local pool obtained from the OLP technique, with different DCS techniques, and the proposed method over all datasets from Table3.4.

Metric	OLP+OLA	OLP+LCA	OLP+MCB	Proposed
Disagreement	0.0235	0.0520	0.0455	0.2183
Cosine distance	0.0946	0.1567	0.1351	0.5170

3.5.2.4 Average performance

We now compare the overall performance of the OLP and the proposed method. Table 3.6 shows the average balanced accuracy rate of the OLP and the proposed method over each dataset from Table 3.4. It can be observed that the proposed method yielded a higher average balanced accuracy rate than the OLP over most datasets, considering all three DCS techniques. In fact, the proposed method outperformed the OLP over the problems shown to have a larger proportion of samples in overlap areas, according to Figure 3.8, with the exception of the *tumors_C* and *leukemia* datasets.

Performing a Wilcoxon signed-rank test, we observe that the proposed method achieved a performance significantly superior to that of the OLP considering all three DCS techniques, with $\alpha = 0.05$. This suggests the local region definition given by a decision tree may be better suited for high dimensional data within the proposed framework. This, however, may stem from several distinct aspects. The feature space partition guided by the Gini index may yield a more

Table 3.6 Average balanced accuracy rate of the OLP with OLA, LCA and MCB, and the proposed method. The row *Win-tie-loss* indicates the amount of datasets over which the proposed technique obtained a higher, equal or lower average performance than the column-wise technique. The row *P-value* shows the p-value of the Wilcoxon signed-rank test between the proposed method and the column-wise technique. Best results are in bold.

Dataset	OLP+OLA	OLP+LCA	OLP+MCB	Proposed
fri_c4_1000_100	0.5948	0.5948	0.5948	**0.8684**
fri_c4_500_100	0.5489	0.5489	0.5489	**0.8444**
fri_c4_100_100	0.6050	0.6050	0.6050	**0.7400**
fri_c4_250_100	0.5994	0.5994	0.5994	**0.7890**
spectrometer	**0.8544**	0.7716	**0.8544**	0.8396
sylva_prior	**0.9715**	0.9300	**0.9715**	0.9573
yeast_ml8	0.5000	0.5000	0.5000	**0.5385**
tecator	**0.9048**	0.7866	0.8998	0.9006
musk	0.9328	0.8917	0.9328	**1.0000**
mfeat-factors	**0.9861**	0.9811	**0.9861**	0.9783
mfeat-pixel	0.9247	0.9225	0.9247	**0.9700**
scene	0.7588	0.7506	0.7588	**0.8819**
gina_prior	0.8198	0.7468	0.8194	**0.8644**
gina_agnostic	0.7143	0.6549	0.7151	**0.8590**
tumors_C	**0.6708**	**0.6708**	**0.6708**	0.6458
leukemia	**0.9050**	**0.9050**	**0.9050**	0.8475
AP_Omentum_Uterus	0.8731	0.8621	0.8793	**0.8837**
AP_Breast_Prostate	0.9702	0.9702	0.9702	**0.9745**
OVA_Ovary	0.7322	0.7027	0.7326	**0.8181**
AP_Uterus_Kidney	0.9263	0.8885	0.9263	**0.9708**
AP_Ovary_Uterus	0.7992	0.7904	0.8097	**0.8373**
AP_Breast_Kidney	**0.9577**	0.9490	**0.9577**	0.9465
AP_Omentum_Ovary	0.6220	0.5647	0.6220	**0.6731**
AP_Colon_Ovary	0.8772	0.8404	0.8808	**0.9172**
AP_Endometrium_Lung	**0.9221**	0.8705	**0.9221**	0.8731
OVA_Endometrium	0.6124	**0.6594**	0.6044	0.5985
AP_Omentum_Kidney	0.9170	0.9065	0.9170	**0.9443**
AP_Colon_Omentum	0.8497	0.8406	0.8497	**0.9141**
AP_Prostate_Kidney	**0.9747**	**0.9747**	**0.9747**	**0.9747**
OVA_Uterus	0.6944	0.6739	0.6910	**0.6978**
AP_Endometrium_Prostate	0.9512	0.9595	0.9512	**0.9929**
AP_Colon_Kidney	**0.9743**	0.9663	0.9741	0.9595
AP_Lung_Uterus	0.8689	0.8606	0.8651	**0.9401**
AP_Colon_Prostate	0.9370	0.9156	0.9370	**0.9471**
OVA_Breast	0.9161	0.9073	**0.9190**	0.9140
AP_Omentum_Lung	0.9008	**0.9126**	0.9046	0.8841
AP_Prostate_Uterus	0.9521	0.9310	0.9521	**0.9819**
AP_Breast_Omentum	0.8743	0.8459	0.8695	**0.9015**
AP_Endometrium_Breast	0.8983	0.7995	0.8983	**0.9236**
Average	0.8280	0.8065	0.8281	**0.8716**
Average rank	2.359	3.4231	2.4103	**1.8077**
Win-tie-loss	26-1-12	31-1-7	27-1-11	n/a
P-value	0.0009	$< 10^{-4}$	0.0008	n/a

adequate border detection, as in the problems whose trees did not present any impure leaves while the KNN found borderline instances such as in the *musk* dataset. Another important aspect is the region size, which in the OLP is only slightly changed while in the proposed method it may be drastically modified. This increases the diversity of the local pool and may produce classifiers with a broader view of the problem, which is not likely to happen in the OLP. Both of these characteristics may have a positive impact, with strictly local algorithms presenting limitations (Zakai & Ritov, 2008), though with regards to the diversity of a localized pool, the extent of its effect on the ensemble's performance is yet to be investigated in the literature.

3.5.2.5 Time complexity

We now estimate the time complexity in Big-O notation of the proposed method and compare it to the OLP technique, in memorization and generalization. To do so, we assume the corresponding hyperparameters of both methods are the same, namely the local pool size and the neighborhood size/minimum number of samples per leaf. Table 3.7 shows the symbols and the time complexity of both algorithms. For simplicity, we also disregard the classifier selection scheme that may be adopted in either technique. Moreover, since the number of classifiers is very small in the OLP framework, for the proposed method we will assume only one tree, so $T = 1$ and $L = |LP| = p$ (Table 3.7a).

We start with the offline phase. In the OLP, the instance hardness measure K-Disagreeing Neighbors (KDN) (Smith *et al.*, 2014) is used to identify the class borders. The KDN is based on the KNN, and its leave-one-sample-out calculation requires computing a pairwise distance between all samples from the training set. Thus, depending on the nearest-neighbor implementation, the worst case scenario would be n times the cost for each sample, which is ndk, so $O(n^2dk)$. In the proposed method, a single tree is fit over the training set, so the computational cost would come entirely from that. The worst case scenario in the production of the tree would entail obtaining the maximum amount of leaves, and therefore splits. With k as the minimum leaf size, the maximum amount of split nodes would be $\lfloor n/k \rfloor 1$. Thus, the cost of growing the tree in the worst case scenario would be $(\lfloor n/k \rfloor 1) \times dc = O(\frac{ndc}{k})$.

In the online phase, the worst case scenario for both techniques is when the query instance is assumed to be close to the class borders. For the OLP, the KNN is used to retrieve the sample's local region. Though it is used repeatedly, the computations can be performed only once. At each local region, the SGH ($O(dc)$) is applied and one classifier is chosen to be added to the pool. Thus, the cost at the online phase is $ndk \times p \times dc = O(ndkpc)$. In the proposed method, the worst case scenario for the local region definition would be using a tree with maximum depth, which would also mean the maximum number of leaves, each of which with k training samples, in an unbalanced structure. So the maximum depth would also be the maximum number of split nodes, $\lfloor n/k \rfloor 1$. As in the case of the KNN in the OLP, the path needs only to be obtained once. Then, at each level (p times), the SGH is applied, so the cost of the proposed method in the worst case scenario would be $(\lfloor n/k \rfloor 1) \times p \times dc = O(\frac{npdc}{k})$.

Table 3.7 (a) Description of the symbols and (b) complexity analysis of the OLP and the proposed adaptation, in training (offline phase) and test (online phase) time.

(a)

Symbol	Meaning
n	Training set size
d	Problem dimensionality
c	Number of classes
k	Neighborhood size
p	Local pool size

(b)

Phase	OLP	Proposed
Offline	$O(n^2dk)$	$O(\frac{ndc}{k})$
Online	$O(ndkpc)$	$O(\frac{npdc}{k})$

Of course, some details of implementation in both techniques can help reducing the computational cost. The nearest neighbors can be obtained using more efficient algorithms instead of the brute force method. Moreover, Table 3.7 shows the cost in the worst case scenario. In both techniques, the local pool is only used to label instances that are identified as borderline, which are generally much less numerous than safe samples. In fact, in (Souza *et al.*, 2019b) we observed empirically that the total execution time of the OLP was much lower than of several DS techniques. The proposed method has the added advantage that in a scenario with a balanced tree, the decision path is obtained with $O(log(n/k))$ instead of $O(n/k)$.

3.5.3 Comparison against other local ensemble methods

We now evaluate how the proposed technique behaves when scaling up the local pool size with the use of a RF ensemble and compare it to other local ensembles. Due to the much larger number of classifiers, we take a different approach in this analysis compared to the previous one (Section 3.5.2). Instead of a white-box analysis, we perform a comparative analysis based on performance between the proposed method and a few baselines, as well as some state-of-the-art ensembles, in order to assess (a) whether the local decision rules already present in the trees can be improved with the proposed pool of linear classifiers (LP), and whether the proposed method presents some competitiveness w.r.t. high dimensional problems against (b) several local ensembles with varying definitions of locality, and (c) the best result found in the OpenML database. We investigate (a) in Section 3.5.3.1, (b) in Section 3.5.3.2 and (c) in Section 3.5.3.3.

3.5.3.1 Impact of the local pool in the decision rule

As opposed to the single DT case, using multiple trees within the proposed method introduces a second source of diversity in the local region definition in addition to the locality reduction: the random feature sampling at each node. Thus, with more diverse partitions being used to generate the local pool, we evaluate the impact of its linear decision rules compared to the different local rules already defined in the trees' nodes used to produce the hyperplanes. That is, considering the (singled-out) nodes of the trees as weak classifiers, we compare in terms of performance the aggregated response given by their classification rules against the aggregated response of the local pool, generated over the same partitions. That way, we can observe the effect in performance the introduction of the local pool provides over the different set of partitions we use in its creation. The comparative analysis is done so that the total number of weak classifiers, either the nodes of the trees or the linear models in the LP, is set to ≈ 100. The evaluated configurations are then, in number of trees (T)/ number of levels (L): ($T = 100, L = 1$), ($T = 50, L = 2$) and ($T = 33, L = 3$).

Each point in Figure 3.11 represents the average balanced accuracy rate of a given (T, L) configuration over a dataset, with the x-axis indicating the performance obtained by the proposed technique (that is, using the local pool generated over the trees' partitions) and the y-axis indicating the performance obtained from the local decision rules given by the (same) partitions themselves. It can be observed that, in most cases, using the LP yielded a higher average performance compared to the trees' original local rules, specially for the problems over which the performance was on the lower range.

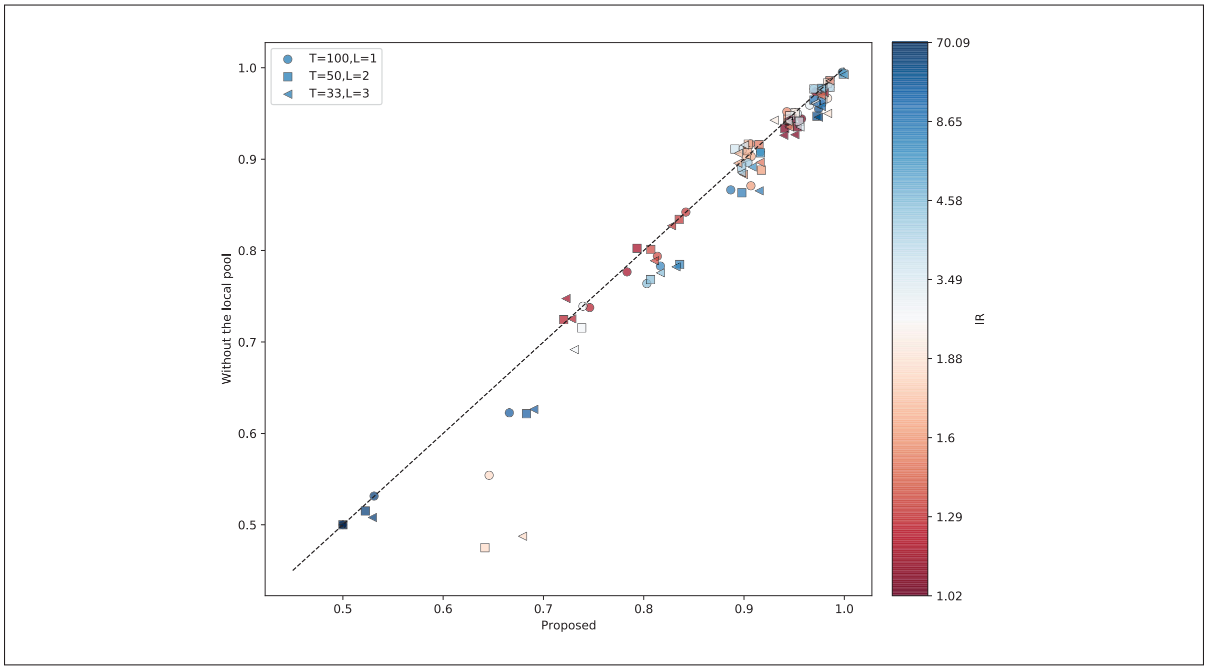


Figure 3.11 Average balanced accuracy rate of the proposed method (with the decision rules given by the local pool, LP), x-axis, and the aggregated decision rules from the corresponding trees' partitions only, y-axis, over all datasets from Table 3.4 and with $(T = 100, L = 1)$, $(T = 50, L = 2)$ and $(T = 33, L = 3)$.

Table 3.8 summarizes the pairwise comparison between the proposed method and the corresponding tree-based ensembles without the linear models. We can see that, for the same number of trees/levels, using the local pool yielded a significantly superior performance, with $\alpha = 0.05$, considering all three evaluated configurations. This suggests that the linear decision rules given by the local pool can provide an improvement over the local rules defined in the trees' partitions used for the pool generation procedure, even when taking into account only the leaf nodes ($T = 100, L = 1$). Considering all configurations, we observe that using the local pool outperforms the trees' local decision rules over the majority of the datasets in all cases, further suggesting an advantage in using the linear classifiers over the trees' partitions.

Table 3.8 Summary of the pairwise comparison between the performances of the proposed method (with the decision rules given by the local pool, LP), row-wise, and the aggregated decision rules from the corresponding trees' partitions only, column-wise, in terms of balanced accuracy rate. The row *P-value* indicates the p-value of a Wilcoxon signed rank test, with the results below $\alpha = 0.05$ in bold. The *Win-tie-loss* row shows the amount of datasets over which the proposed technique yielded a higher, similar or lower average performance than the column-wise configuration.

Proposed		$T = 100, L = 1$	$T = 50, L = 2$	$T = 33, L = 3$
$T = 100, L = 1$	P-value	**0.0004**	**0.0008**	$< \mathbf{10^{-4}}$
	Win-tie-loss	26-6-7	24-6-9	30-5-4
$T = 50, L = 2$	P-value	**0.0082**	**0.0019**	$< \mathbf{10^{-4}}$
	Win-tie-loss	24-6-9	24-6-9	31-4-4
$T = 33, L = 3$	P-value	0.1700	**0.0087**	**0.0001**
	Win-tie-loss	20-7-12	24-8-7	28-6-5

3.5.3.2 Comparison against the state-of-the-art

We now evaluate the performance of the proposed framework and of several ensemble methods in order to assess whether there is any advantage in using the proposed local approach for high dimensional data compared to some already existent in the literature. We chose the Random Forest as our baseline, as although it does not explicitly enforce the locality aspect, the decision rule of each tree in the ensemble is local, that is, mostly influenced by the close points to the query (Bischl *et al.*, 2013). The exact same RF ensemble is used as the pool of classifiers in four of the DS techniques included in the comparative analysis, namely Dynamic Ensemble Selection-Clustering (DES-C) (Soares *et al.*, 2006), K-Nearest Oracles Union (KNU) (Ko *et al.*, 2007), K-Nearest Oracles Eliminate (KNE) (Ko *et al.*, 2007), K-Nearest Output Profiles (KNOP) (Cavalin *et al.*, 2012), and Randomized Reference Classifier (RRC) (Woloszynski & Kurzynski, 2011). Because of that, and due to the very limited number of instances in some problems (such as *leukemia* and *tumors_C*), we use the whole training set as the dynamic selection set (DSEL) in the experiments. The DSEL is a set of labelled data used in dynamic selection techniques for region of competence definition (Cruz *et al.*, 2018a). We chose these techniques since they define locality differently, as indicated in Table 3.3. We also include in the analysis the Forest of Local Trees (FLT) ensemble (Armano & Tamponi, 2018) and the OLP technique. The latter is used in this analysis as a reference for a non-tree based local ensemble method.

The hyperparameters of the local ensemble methods, shown in Table 3.9, were set according to their default configuration in the Python libraries scikit-learn (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg et al., 2011) and DESLib (Cruz, Hafemann, Sabourin & Cavalcanti, 2020) except for the FLT and the OLP. In the case of the former we used the hyperparameters recommended in the original paper, while in the case of the latter we picked the configuration with the highest overall balanced accuracy rate from Table 3.6. All ensembles but the OLP's contain 100 trees as base-classifiers. On our side, we include in this analysis all configurations of number of trees/number of levels evaluated in Section 3.5.3.1.

Table 3.10 shows the average balanced accuracy rate of the evaluated techniques. It can be observed that all DS techniques presented a general improvement, though a slight one in some cases, over the original RF ensemble, which suggests an overall benefit in explicitly exploiting the information around the target area. However, we can see from the performance of the OLP+MCB that the relying heavily on the nearest neighbors information was detrimental in most of the high dimensional problems.

With regards to the proposed method, we can observe that it obtained the highest overall balanced accuracy rate and average rank with ($T = 100, L = 1$), with ($T = 50, L = 2$) yielding the second highest overall performance and fourth highest rank, after the FLT. These two versions also outperformed all tree-based ensembles on five of the nine problems identified in Section 3.5.2 as having a comparatively higher proportion of samples in overlap regions. The version with fewer trees, ($T = 33, L = 3$), still yielded a better overall performance and higher rank than the RF baseline.

Table 3.9 Hyperparameter setting of the local ensembles included in the comparative analysis. The pool of classifiers used in all DS techniques but the FLT and OLP+MCB is the RF baseline.

Local ensemble methods	Hyperparameters

Random Forest (RF)	max_depth=$None$
	min_samples_split=2
	max_features=$\sqrt{\#features}$
	bootstrap=$True$
	max_samples=1.0
	max_leaf_nodes=$None$

Dynamic Ensemble Selection-Clustering (DES-C)	clustering=K-means($K = 5$)
	pct_accuracy=0.5
	pct_diversity=0.33
	metric_performance=$accuracy\_score$
	metric_diversity=df

K-Nearest Oracles Union (KNU)	K=7
K-Nearest Oracles Eliminate (KNE)	K=7
K-Nearest Output Profiles (KNOP)	K=7
Randomized Reference Classifier (RRC)	mode=$selection$

Forest of Local Trees (FLT)	max_depth=$None$
	min_samples_split=2
	max_features=0.3
	bootstrap=$True$
	max_samples=1.0
	max_leaf_nodes=$None$

Online Local Pool with Multiple Classifier Behavior (OLP+MCB)	K=7 $\|LP\| = 3$

Comparing the individual pairs of results, Table 3.11 shows the resulting p-values of the Wilcoxon signed rank test over the average performances, as well as their win-tie-loss summary. First, we can see that two of the three versions of the proposed technique, namely ($T = 100, L = 1$) and ($T = 50, L = 2$), significantly outperform the RF baseline with significance level $\alpha = 0.05$. The version with fewer trees, ($T = 33, L = 3$), yielded a statistically similar performance to the RF baseline with $\alpha = 0.05$.

With respect to the other techniques, the ($T = 100, L = 1$) version of the proposed method obtained a statistically superior performance compared to the KNORAU, KNORAE, KNOP and RRC with $\alpha = 0.05$, while the ($T = 50, L = 2$) version significantly outperformed the KNORAU and RRC with $\alpha = 0.05$. The KNOP presenting a comparatively better performance with respect to the KNORAU and RRC could be due to the distance computations being performed in the

Table 3.10 Mean balanced accuracy rate of the evaluated local ensembles over all folds for each dataset, ordered by increasing dimensionality. Best results are in bold.

Dataset	Local ensembles								Proposed		
	RF	DES-C	KNORAU	KNORAE	KNOP	RRC	FLT	OLP+MCB	T=100,L=1	T=50,L=2	T=33,L=3
fri_c4_1000_100	0.8455	0.8634	0.8501	0.8428	0.8529	0.8455	**0.9020**	0.5948	0.8420	0.8354	0.8277
fri_c4_500_100	0.7993	0.8243	0.8084	0.7974	0.8136	0.8114	**0.8771**	0.5489	0.8136	0.8070	0.8107
fri_c4_100_100	0.7483	0.6958	0.7633	0.6842	0.7833	0.7267	**0.7958**	0.6050	0.7833	0.7933	0.7225
fri_c4_250_100	0.7114	0.7279	0.7260	0.7234	0.7123	0.7396	**0.8432**	0.5994	0.7461	0.7201	0.7282
spectrometer	0.8470	0.8522	0.8470	0.8554	0.8543	0.8554	0.8743	0.8544	0.8867	0.8978	**0.9151**
sylva_prior	0.9655	0.9705	0.9660	0.9722	0.9694	0.9666	**0.9797**	0.9715	0.9743	0.9726	0.9742
yeast_ml8	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	**0.5119**	0.5000	0.5000	0.5000	0.5000
tecator	0.9323	0.9299	0.9323	0.9311	0.9323	0.9223	**0.9455**	0.8998	0.9064	0.9149	0.9161
musk	0.9995	**1.0000**	0.9995	**1.0000**	**1.0000**	0.9995	**1.0000**	0.9328	0.9980	0.9990	0.9999
mfeat-factors	0.9700	0.9697	0.9700	0.9725	0.9700	0.9700	0.9847	**0.9861**	0.9775	0.9772	0.9772
mfeat-pixel	0.9722	0.9747	0.9722	0.9722	0.9722	0.9722	**0.9819**	0.9247	0.9772	0.9697	0.9722
scene	0.7590	0.8105	0.7659	0.7881	0.7764	0.7602	**0.8822**	0.7588	0.8031	0.8068	0.8167
gina_prior	0.9470	0.9439	0.9453	0.9442	0.9473	0.9470	0.9526	0.8194	**0.9574**	0.9533	0.9507
gina_agnostic	0.9401	0.9370	0.9393	0.9343	0.9393	0.9401	0.9179	0.7151	**0.9446**	0.9406	0.9397
tumors_C	0.4667	0.5667	0.4542	0.6333	0.4542	0.4667	0.5167	0.6708	0.6458	0.6417	**0.6792**
leukemia	0.9500	0.9667	0.9500	0.9500	0.9667	0.9500	0.9567	0.9050	**0.9833**	**0.9833**	**0.9833**
AP_Omentum_Uterus	0.8965	**0.9203**	0.8965	0.9027	0.8965	0.8965	0.8884	0.8793	0.9072	0.9033	0.8938
AP_Breast_Prostate	0.9774	0.9774	0.9774	0.9774	0.9774	0.9774	**0.9845**	0.9702	0.9774	0.9774	0.9774
OVA_Ovary	0.7578	0.7757	0.7674	0.7664	0.7738	0.7578	0.5000	0.7326	0.8166	**0.8358**	0.8322
AP_Uterus_Kidney	0.9689	0.9785	0.9689	0.9689	0.9689	0.9689	**0.9808**	0.9263	0.9788	0.9785	0.9708
AP_Ovary_Uterus	0.8797	0.8904	0.8915	0.8889	0.8929	0.8797	0.8645	0.8097	0.9068	**0.9174**	0.8994
AP_Breast_Kidney	0.9755	0.9755	**0.9774**	**0.9774**	**0.9774**	0.9755	0.9692	0.9577	0.9769	0.9769	0.9769
AP_Omentum_Ovary	0.6811	0.6943	0.6749	0.6542	0.6802	0.6811	0.6836	0.6220	**0.7394**	0.7381	0.7308
AP_Colon_Ovary	0.9494	**0.9554**	0.9494	0.9451	0.9494	0.9494	**0.9554**	0.8808	0.9427	0.9434	0.9452
AP_Endometrium_Lung	0.9510	**0.9593**	0.9510	**0.9593**	0.9510	0.9510	0.9051	0.9221	0.9510	0.9510	0.9301
OVA_Endometrium	0.5071	0.5141	0.5071	0.5315	0.5071	0.5071	0.5000	**0.6044**	0.5311	0.5225	0.5294
AP_Omentum_Kidney	0.9501	0.9654	0.9501	0.9520	0.9563	0.9501	**0.9731**	0.9170	0.9654	0.9457	0.9438
AP_Colon_Omentum	0.9022	**0.9147**	0.9022	0.9039	0.9022	0.9022	0.9102	0.8497	0.8988	0.8909	0.9006
AP_Prostate_Kidney	**0.9857**	0.9786	**0.9857**	**0.9857**	**0.9857**	**0.9857**	0.9838	0.9747	**0.9857**	**0.9857**	0.9786
OVA_Uterus	0.6141	0.6332	0.6099	0.6441	0.6259	0.6141	0.5000	0.6910	0.6660	0.6831	**0.6904**
AP_Endometrium_Prostate	0.9857	**0.9929**	**0.9929**	**0.9929**	**0.9929**	0.9857	0.9786	0.9512	0.9857	0.9857	0.9857
AP_Colon_Kidney	0.9740	0.9740	0.9740	0.9759	0.9740	0.9740	0.9777	0.9741	0.9710	0.9785	**0.9800**
AP_Lung_Uterus	0.9327	0.9321	0.9327	0.9362	0.9285	0.9327	0.9446	0.8651	0.9439	**0.9481**	0.9442
AP_Colon_Prostate	0.9768	0.9697	0.9768	0.9697	0.9768	0.9768	**0.9823**	0.9370	0.9697	0.9697	0.9768
OVA_Breast	0.9430	0.9373	0.9416	**0.9430**	0.9430	0.9430	0.5000	0.9190	0.9533	0.9551	**0.9557**
AP_Omentum_Lung	0.9189	0.9174	0.9189	0.9260	0.9147	0.9189	**0.9266**	0.9046	0.9022	0.9043	0.8942
AP_Prostate_Uterus	**0.9857**	**0.9857**	**0.9857**	**0.9857**	**0.9857**	**0.9857**	0.9786	0.9521	**0.9857**	**0.9857**	**0.9857**
AP_Breast_Omentum	0.9039	**0.9275**	0.9039	0.9212	0.9039	0.9039	0.9245	0.8695	0.9039	0.8976	0.8976
AP_Endometrium_Breast	0.9164	**0.9248**	0.9164	**0.9248**	0.9164	0.9164	0.9135	0.8983	0.9164	0.9164	0.9081
Average	0.8741	0.8776	0.8748	0.8752	0.8724	0.8694	0.8627	0.8281	**0.8850**	0.8847	0.8831
Average rank	6.8718	5.2821	6.4744	5.6282	5.8205	6.4872	4.7564	9.3462	**4.7436**	5.1923	5.3974

decision space instead of the feature space, so the technique may be less affected by the high dimensionality of the problems.

Lastly, we can observe that all versions of the technique were statistically similar to the DES-C and FLT methods. The former is set apart from the other techniques as it takes into account the diversity of the classifiers as well as their local accuracy to assign an ensemble to a given defined region. The FLT method is also set apart from the other techniques, as it uses a different

Table 3.11 Summary of the pairwise comparison between the performances of the proposed method and the evaluated local ensembles in terms of balanced accuracy rate over all datasets. The row *P-value* indicates the p-value of a Wilcoxon signed rank test, with the results below $\alpha = 0.05$ in bold. The *Win-tie-loss* row shows the amount of datasets over which the proposed technique yielded a higher, similar or lower average performance than the column-wise technique.

Proposed		RF	DES-C	KNU	KNE	KNOP	RRC	FLT	OLP+MCB
T=100,L=1	P-value	**0.0009**	0.1798	**0.0031**	**0.0249**	**0.0125**	**0.0013**	0.7855	$< \mathbf{10^{-4}}$
	Win-tie-loss	23-8-8	20-5-14	22-7-10	21-5-13	20-8-11	23-8-8	18-0-21	32-1-6
T=50,L=2	P-value	**0.0078**	0.5158	**0.0494**	0.1602	0.0545	**0.0345**	0.7641	$< \mathbf{10^{-4}}$
	Win-tie-loss	22-7-10	18-5-16	19-6-14	19-5-15	20-6-13	20-7-12	20-0-19	33-1-5
T=33,L=3	P-value	0.0919	0.5435	0.201	0.1625	0.3528	0.2056	0.8071	$< \mathbf{10^{-4}}$
	Win-tie-loss	20-6-13	20-4-15	20-5-14	21-4-14	18-5-16	18-6-15	16-0-23	34-1-4

ensemble of trees that are optimized to fit well over different areas of the feature space. In generalization, the closer to the query, the more important the classifier is considered, though all trees contribute to the final response. So, through different means, both techniques actively attempt to provide a certain degree of diversity, subject to the local accuracy, to the ensemble responsible for labelling each sample, which could have had a positive impact on the performance over the high dimensional data.

3.5.3.3 Comparison against the best result from OpenML

Lastly, we perform a comparative analysis against the results obtained from the best performing execution, in terms of balanced accuracy rate, registered in the OpenML database (Vanschoren *et al.*, 2013) for each dataset in particular. That is, using the OpenML Python API (Feurer, Van Rijn, Kadra, Gijsbers, Mallik, Ravi, Müller, Vanschoren & Hutter, 2021) we fetched, for each dataset, the execution (or *run*) which yielded the highest macro-averaged recall, as well as its label outputs in order to calculate the performance on each one of the folds. A run in the OpenML database consists of the record of an execution of a given machine learning pipeline over a given task, in this case the classification task over a dataset. The pipeline can include any sort of model, pre-processing and hyperparameter optimization steps. Thus, it is reasonable to expect that the best performance among all registered runs for each dataset were obtained by tuning the pipeline to that individual problem. Our reason for comparing our technique against the best result registered in the OpenML database is to use it as a reference for the upper limit of performance in order to assess how the chosen local ensembles fare against it over each high dimensional dataset, taking into consideration they were not specifically tuned. The dataset-by-dataset highest average balanced accuracy rate retrieved from the OpenML database, as well as its corresponding run ID, can be found in the supplementary material.

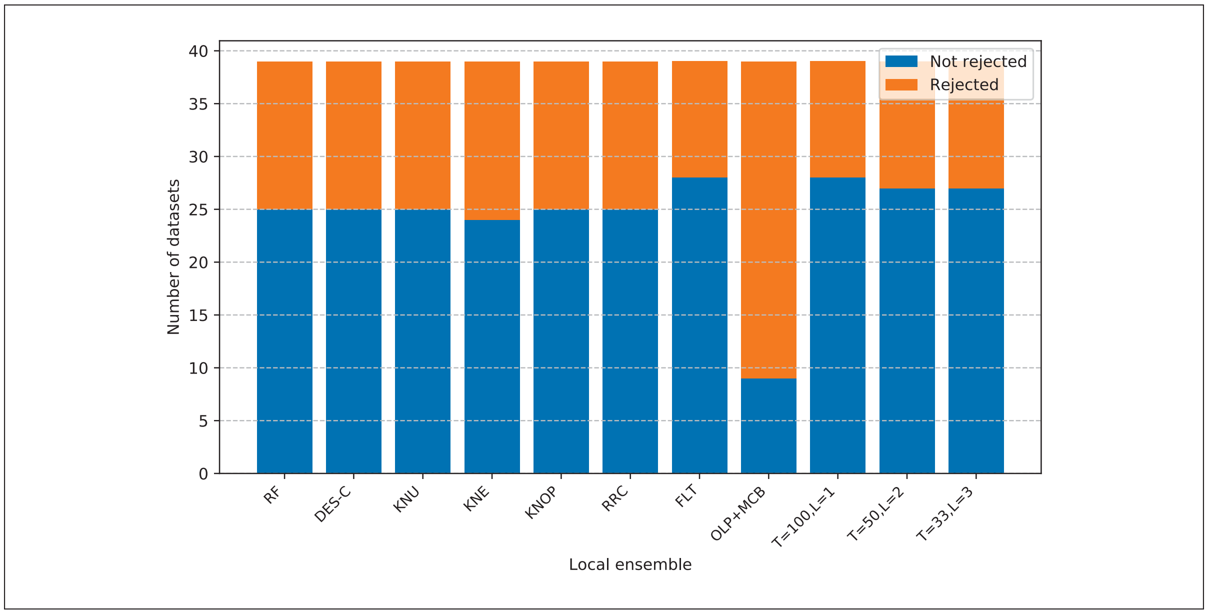


Figure 3.12 Number of datasets over which the null hypothesis, that the median balanced accuracy rate of the indicated local ensemble method and the best performing execution in the OpenML database are the same, was rejected or not (with $\alpha = 0.05$).

Since each performance result for each dataset may have been obtained from very different classification pipelines, we perform now a dataset-by-dataset comparison taking into account the balanced accuracy rate obtained over each fold. We performed a pairwise Kruskal-Wallis test with a 95% confidence interval, between the best execution from the OpenML and each local ensemble method, on the performances over the folds for each dataset individually. Figure 3.12 shows the number of problems over which the null hypothesis, meaning the median of the per-fold balanced accuracy rates of the best execution and the local ensemble method are equal, was rejected or not. It can be observed that, with the exception of the OLP+MCB, the local ensemble methods obtained a statistically equivalent balanced accuracy rate over at least 60%

of the datasets, with the FLT and the proposed technique ($T = 100, L = 1$) yielding a similar performance to the best OpenML execution over 28 datasets, or approximately 72% of the test bed. This suggests that, while the tree-based local ensemble methods, including the proposed technique, were not tuned to each individual problem, they can often achieve a comparable performance to the state-of-the-art on high dimensional problems.

3.6 Conclusions

In this work, we proposed a local ensemble method adapted for dealing with high dimensional data. Instead of relying on the Euclidean distance and the nearest neighbors rule, which may suffer from the effects of the curse of dimensionality, the proposed technique leverages the data partitions obtained from tree-based algorithms for the locality definition. That way, in the case an unknown instance falls into an impure leaf, the regions defined at multiple levels of its decision path are used to produce the local pool responsible for labelling it. By using different partitions from the same decision path, we introduce diversity to the pool subject to the locality of the query in order to promote consensus between the classifiers.

The experimental analysis was conducted over 39 high dimensional problems. Results showed that the proposed partition-based approach to region definition was often more successful in identifying borderline instances compared to the nearest neighbors-based approach used in the original Online Local Pool framework (OLP). We also observed that the local pool generated in the proposed method presented a more distinct set of classifiers in comparison with the OLP. A time complexity analysis also demonstrated that the proposed technique is less computationally expensive than the OLP. Moreover, the proposed method significantly outperformed the OLP using the same pool size.

Investigating the impact of the local pool's decision rule compared to the trees' already defined local rules themselves showed that the former was able to significantly improve the average performance of the latter. Furthermore, a comparative analysis against six state-of-the-art local-based ensembles demonstrated the competitiveness of the proposed approach, which was able to significantly outperform the random forest baseline in most cases, as well as two DS techniques, in terms of balanced accuracy rate. The experimental results thus suggest that leveraging the data partitions from decision trees for building local classifiers is advantageous when dealing with high dimensional data.

In the current framework, since the partitions are defined in memorization time and there is little control over their size, the classifier generated over it, which can be preprocessed offline, may still not be an adequate local expert for all samples that fall into the region. Thus, in future works, we aim on further adapting the local classifiers over the region according to each query instance via local regularization, as a way to counter the rapid loss of locality observed in the experiments. In addition, we may improve the aggregation step of the method, either using a geometrical integration (Ksieniewicz, Zyblewski & Burduk, 2021) or using a dynamic weighting scheme based on local data characterization (Campos, Morell & Ferri, 2012). In addition to that, an adaptation of the proposed method for multi-class problems using a local-based classifier combination technique for the one-versus-one (OVO) decomposition strategy (Galar, Fernández, Barrenechea & Herrera, 2015) may be investigated in the future. Different recursive partitioning mechanisms may also be investigated for improving the local region definition step, including different split strategies within the trees' construction. Lastly, the relationship between diversity and consensus w.r.t. a locally generated ensemble may also be investigated in the future, with the aim of optimizing the local data partitions to present a certain degree of both (Zhou, Wang, Du & Li, 2022).

In the next chapter, a further analysis on the performance of several dynamic selection techniques using different ensembles over HDSSS problems is carried out, and a dynamic multiple classifier system is proposed to address their limitations associated with a weaker locality assumption in pre-defined sparse and overlapped regions.

3.7 Supplementary material

3.7.1 Self-generating Hyperplanes (SGH) method

The SGH (Souza *et al.*, 2017) is a pool generation method which produces linear decision rules iteratively over a given input set, so that the generated ensemble contains at least one classifier able to correctly label each known instance. The two-class linear rules are obtained using an heuristic, which speeds up the generation process. Figure 3.13 illustrates the pool generation procedure of the SGH method. The two-class dataset input to the technique is depicted in Figure 3.13a. In the first iteration of the method, the centroids of each class, considering all instances from the dataset (highlighted in red), are obtained, and a linear classifier is placed in the midpoint that connects them. This classifier is illustrated in Figure 3.13b, with the arrow indicating where it labels as the green class. Afterwards, the instances that are misclassified by the classifier in Figure 3.13b are removed from the set.

In the second iteration, the remaining instances in the set, highlighted in red in Figure 3.13c, are then used to generate a second linear classifier, which is placed in the midpoint between the current classes' centroids. Then, the instances that the recently generated classifier, shown in Figure 3.13c, cannot label correctly are removed from the set. Since it is able to correctly classify all remaining instances, the method stops and returns the pool shown in Figure 3.13d, comprised of two binary classifiers that together have a theoretical upper limit on the accuracy rate of 100% over the input set.

3.7.2 OpenML results

Table 3.12 presents additional information regarding the best execution in terms of balanced accuracy rate found in the OpenML repository for each dataset, including the run ID, the balanced accuracy rate averaged over all folds, and the resulting p-value of the Kruskall-Wallis test between the per-fold balanced accuracy rates of the indicated (best) run and the best performing configuration of the proposed method ($T = 100, L = 1$). We also include in the table

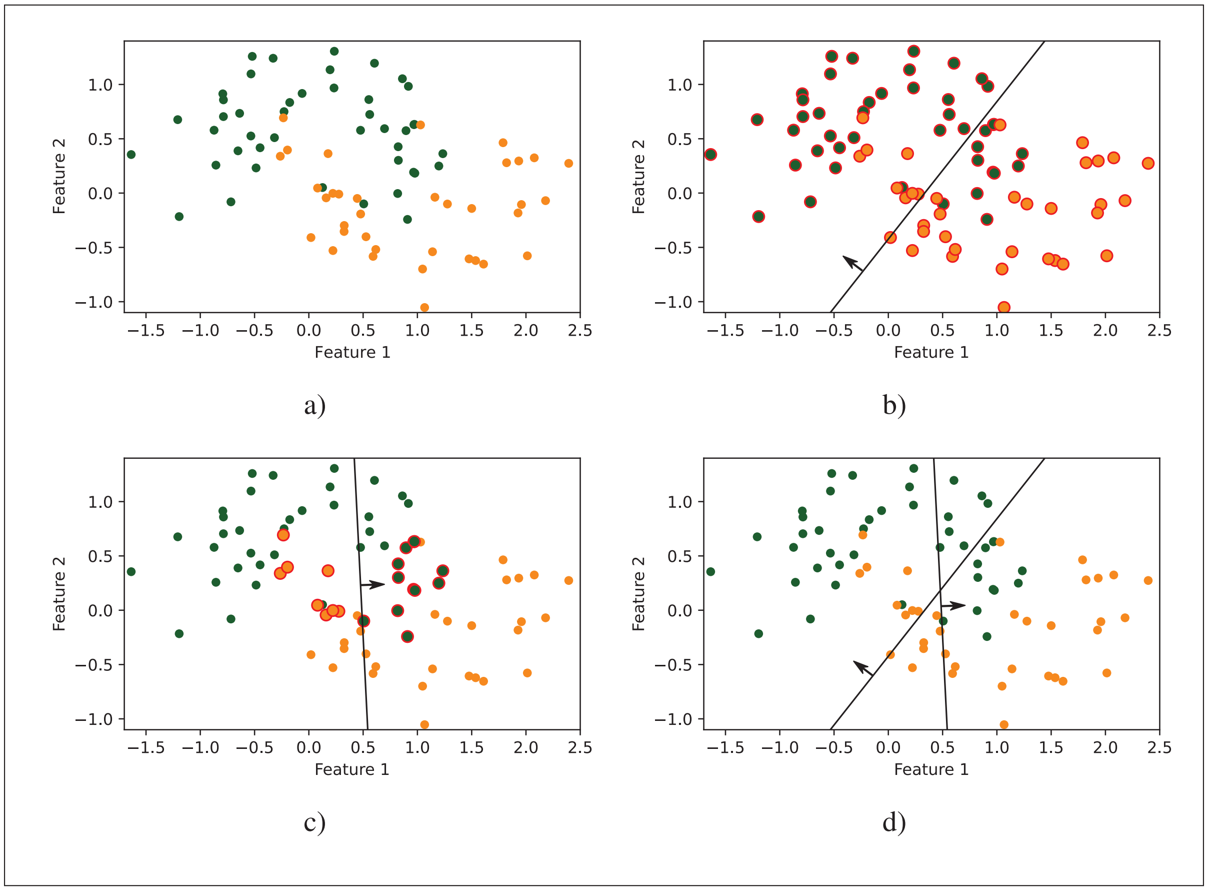


Figure 3.13 Toy example illustrating the SGH method. (a) shows the data input to the technique. (b) shows the first iteration of the technique, with the samples used to generate the indicated linear classifier highlighted in red, and the arrow indicating where the latter labels as the green class. (c) illustrates the second iteration of the technique, also with the linear classifier produced using the samples highlighted in red. The final pool yielded by the SGH method is shown in (d).

a few general characteristics of each dataset, namely the number of instances, the number of features and the imbalance ratio (IR), for context.

Table 3.12 Information regarding the best performing execution retrieved from the OpenML database (Vanschoren *et al.*, 2013) for each dataset. *# Instances* and *# Features* indicate the amount of samples and dimensionality of the datasets, respectively. The *IR* column shows the imbalance ratio, or the ratio between the majority class and the minority class sample counts, of each problem. From the OpenML registered executions over each dataset, column *Run ID* indicates the identifier of the best performing one in terms of balanced accuracy rate. The *Avg. perf.* column shows the balanced accuracy rate obtained in the indicated run, averaged over all folds. The *P. value* column indicates the resulting p-value obtained from the Kruskal-Wallis test with 95% confidence on the per-fold performances of the (indicated) best run and the proposed technique's best performing configuration ($T = 100, L = 1$).

Dataset	Dataset characteristics			Best execution		
	# Instances	# Features	IR	Run ID	Avg. perf.	P-value
fri_c4_1000_100	1000	100	1.29	8878606	0.9187	0.0004
fri_c4_500_100	500	100	1.3	8878857	0.8982	0.0015
fri_c4_100_100	100	100	1.13	515054	0.8133	0.5645
fri_c4_250_100	250	100	1.27	8871859	0.8857	0.0006
spectrometer	531	101	8.65	593342	0.9482	0.1955
sylva_prior	14395	108	15.25	146301	0.9899	0.0041
yeast_ml8	2417	116	70.09	8866903	0.5990	0.3935
tecator	240	124	1.35	46665	0.9610	0.0431
musk	6598	168	5.49	1640440	1.0000	0.1468
mfeat-factors	2000	216	9	523441	0.9967	0.0327
mfeat-pixel	2000	240	9	123100	0.9944	0.0298
scene	2407	299	4.58	2105396	0.9710	0.0002
gina_prior	3468	784	1.03	147370	0.9614	0.6230
gina_agnostic	3468	970	1.03	6008253	0.9627	0.0191
tumors_C	60	7129	1.86	9201347	0.7216	0.4427
leukemia	72	7129	1.88	560987	1.0000	0.3173
AP_Omentum_Uterus	201	10935	1.61	9200768	0.9198	0.5677
AP_Breast_Prostate	413	10935	4.99	563992	0.9884	0.8278
OVA_Ovary	1545	10935	6.8	564743	0.8513	0.1508
AP_Uterus_Kidney	384	10935	2.1	554299	0.9883	0.3625
AP_Ovary_Uterus	322	10935	1.6	560900	0.9188	0.6768
AP_Breast_Kidney	604	10935	1.32	1782939	0.9793	0.6362
AP_Omentum_Ovary	275	10935	2.57	553590	0.7554	0.5961
AP_Colon_Ovary	484	10935	1.44	553594	0.9563	0.1795
AP_Endometrium_Lung	187	10935	2.07	554809	0.9677	0.5068
OVA_Endometrium	1545	10935	24.33	553263	0.8759	0.0001
AP_Omentum_Kidney	337	10935	3.38	564011	0.9839	0.2912
AP_Colon_Omentum	363	10935	3.71	549825	0.9356	0.2703
AP_Prostate_Kidney	329	10935	3.77	550689	0.9908	0.9136
OVA_Uterus	1545	10935	11.46	1672695	0.8430	0.0000
AP_Endometrium_Prostate	130	10935	1.13	550686	1.0000	0.1462
AP_Colon_Kidney	546	10935	1.1	555372	0.9853	0.1006
AP_Lung_Uterus	250	10935	1.02	592461	0.9441	0.9693
AP_Colon_Prostate	355	10935	4.14	566406	0.9875	0.3784
OVA_Breast	1545	10935	3.49	555253	0.9603	0.5959
AP_Omentum_Lung	203	10935	1.64	555388	0.9398	0.1477
AP_Prostate_Uterus	193	10935	1.8	563063	1.0000	0.1462
AP_Breast_Omentum	421	10935	4.47	555260	0.9538	0.1331
AP_Endometrium_Breast	405	10935	5.64	554691	0.9792	0.1453

CHAPTER 4

A DYNAMIC MULTIPLE CLASSIFIER SYSTEM USING GRAPH NEURAL NETWORK FOR HIGH DIMENSIONAL OVERLAPPED DATA

Mariana A. Souza[1] , Robert Sabourin[1] , George D. C. Cavalcanti[2] , Rafael M. O. Cruz[1]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA),
École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Centro de Informática,
Universidade Federal de Pernambuco,
1235 Av. Prof. Moraes Rego, Recife, Pernambuco, Brazil 50670-901

Article submitted to Elsevier's Information Fusion Journal, March 2023

Abstract

Dynamic selection techniques select a subset of the classifiers from a pool according to their perceived competence in labeling each given query instance in particular. To do so, most techniques rely on the locality assumption for the selection task, meaning that similar instances should share a set of adequate classifiers, so their competencies are usually estimated over a local region surrounding the query. However, as the local distribution is crucial to these techniques, a poor region definition due to the presence of high dimensionality and class overlap can have a negative impact on their performance, thus limiting their application. Thus, we propose in this work a dynamic selection technique to better deal with sparse and overlapped data in which the instance-instance and the classifier-classifier relationships are leveraged to learn the dynamic classifier combination rule. The proposed technique uses a multi-label graph neural network as a meta-learner, so both the data modeled as a graph, without directly defining the local region, and the classifiers' inter-dependencies modeled in the meta-labels are used to learn an embedded space where the dynamic selection task is more straightforward. Experimental results over 35 high dimensional datasets show that the proposed method significantly outperforms the static selection baseline and most evaluated dynamic selection techniques when using a diverse ensemble. Moreover, the proposed technique surpassed the contending state-of-the-art techniques over the problems with the highest excess of incompetent classifiers in overlap regions, further suggesting its suitability to deal with challenging local distributions. Code available at: github.com/marianaasouza/gnn_des.

4.1 Introduction

Multiple classifier systems (MCS) attempt to improve the recognition rates of a set of individual classifiers by combining their responses and capitalizing on their complementarity (Kittler *et al.*, 1998). How complementary they are can characterize the *diversity* of the ensemble, an intuitively desirable aspect when composing an MCS. MCSs have been widely applied to solve real-world problems such as intrusion detection (Gao *et al.*, 2019; Gormez, Aydin, Karademir & Gungor, 2020), software defect prediction (Goel *et al.*, 2020), and protein function prediction (Hakala, Kaewphan, Björne, Mehryary, Moen, Tolvanen, Salakoski & Ginter, 2020; Cao *et al.*, 2020), among many others.

An MCS usually has three phases: generation, in which the pool of classifiers is produced, selection, an optional step where a subset of the pool is singled out, and integration, in which the selected classifiers are combined to yield the response of the system (Britto *et al.*, 2014; Cruz *et al.*, 2018a). The selection may be done statically, meaning the same ensemble is used to label all query samples, or dynamically, in which a different subset of classifiers may be used for each individual test sample. Dynamic selection techniques were often shown to outperform static selection (SS) techniques over various classification problems (Cruz *et al.*, 2015a; Woloszynski & Kurzynski, 2011).

The reasoning behind dynamic selection techniques is that, as the classifiers make different mistakes in distinct areas of the feature space, the techniques usually estimate their competence to label a given query according to some criteria evaluated in the local area around the sample (Cruz *et al.*, 2018a), sometimes taking special care for borderline or ambiguous regions (Oliveira *et al.*, 2018; Souza *et al.*, 2022). Most dynamic selection techniques thus rely on the locality assumption to solve the dynamic selection task, meaning that a similar set of classifiers should be able to correctly label similar instances. However, this assumption may fail more often in scenarios expected to be challenging to local methods, as in the presence of ambiguity and distance concentration caused by class overlap and high dimensionality (Zhang, 2022; Vandaele *et al.*, 2022; Costa, Lorena, Peres & de Souto, 2009; Sánchez *et al.*, 2007). This may limit the application of the dynamic selection techniques over the problems that present these issues, such as medical imaging data (El-Sappagh *et al.*, 2021) and DNA microarray data (Osama, Shaban & Ali, 2023; Lorena *et al.*, 2012; Costa *et al.*, 2009) used for disease detection.

We illustrate the matter in Figure 4.1, in which two toy problems are depicted. The first toy problem, shown in Figure 4.1a and Figure 4.1c, presents two classes and two features (both informative), while the second toy problem, shown in Figure 4.1b and Figure 4.1d, is a 10-dimensional toy problem obtained from the first problem by adding six redundant features (using linear combinations of the two informative ones) and two random features (using random noise). We show only the two informative features of the 10-dimensional problem in Figure 4.1b and Figure 4.1d. In all images, the samples in circles belong to the training set, and the samples in a diamond shape are query instances whose k-neighbors in the feature space are highlighted in red and within the region defined with the dashed lines. Moreover, the query instances in Figure 4.1a-b and Figure 4.1c-d are matched, meaning they present the same description considering the two informative features.

In the first pair (Figure 4.1a-b) we can see that the inclusion of the 8 non-informative features increased the class overlap in the k-neighborhood of the 10-dimensional space compared to the 2-dimensional space, though it still remained favorable to the correct (green) class. Considering the effect of the local region definition on the dynamic selection task, an AdaBoost (Freund & Schapire, 1997) ensemble of 10 decision stumps was generated for each problem, and the average ratio of competent classifiers in common between the queries and their neighbors was computed. For the first pair, the ratio went from 0.67 in the 2-dimensional problem to 0.51 in the 10-dimensional problem, a reduction that may not have a great impact on the dynamic

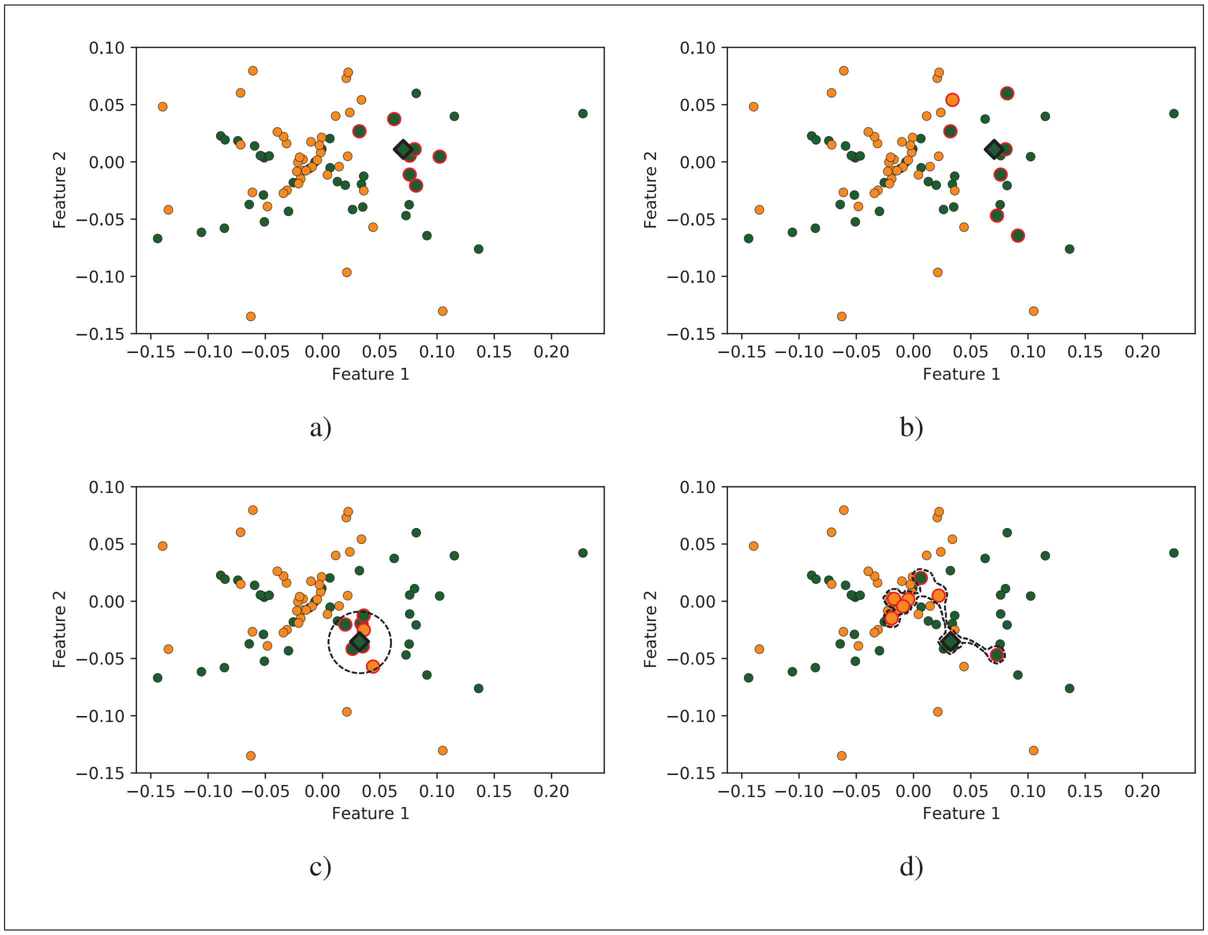


Figure 4.1 Illustrative two-class toy problems. Circles indicate the training samples and diamonds the query instances, whose k-neighbors ($k = 7$) are highlighted in red and within the region defined with the dashed lines. (a,c) is a 2-dimensional problem with two informative features, and (b,d) is a 10-dimensional problem obtained from the 2-dimensional one by adding six redundant features and two random features. We show only the two informative features in (b,d).

selection task as the locality information still seems quite valuable for the instance in Figure 4.1b.

In the second pair of query instances (Figure 4.1c-d), we see that the region in the 2-dimensional space already presents class overlap. The latter is also intensified in the corresponding 10-dimensional space to the point of breaking the k-nearest neighbors classification rule for the depicted query instance. For the second pair, the average ratio of competent classifiers in common between the queries and their neighbors went from 0.44 in the 2-dimensional problem to less than half of that (0.20) in the 10-dimensional problem. This reduction could certainly hinder the dynamic selection techniques as the information in the local region obtained in the 10-dimensional space seems to be insufficient to properly choose a set of classifiers for the query in Figure 4.1d. In such cases, we believe another source of information in addition to the locality could be valuable to perform the dynamic selection task well. Thus, it is our intuition that not only the instances' but also the classifiers' interactions could be used for that aim, and that learning both jointly in an end-to-end manner could provide the system with the necessary information to tackle these scenarios where the locality assumption is not reliable enough for the task.

Thus, we propose in this work a dynamic selection technique that leverages the instance-instance and classifier-classifier relationships to tackle the class overlap and high dimensionality jointly while learning the dynamic selection rule. To do so, we encode the instance-instance relationships in a graph structure where the edges connect the samples (nodes) with a similar ensemble response, taking into account the class overlap, in order to model how much useful information for the dynamic selection task the samples may share. The classifier-classifier relationships, on the other hand, are encoded as multi-labels that indicate which classifiers correctly label each given sample. The dynamic selection meta-problem is then learned subject to these relationships using a graph neural network (GNN) with a multi-label output layer, providing an end-to-end solution to the dynamic selection task instead depending on hand-crafted meta-features. Implicitly, the meta-learner learns an embedded space where the instances with a similar set of competent classifiers are close-by and is responsible for indicating the combination of the classifiers to be used in the integration phase. That way, the framework exploits both the samples' local class connections and the classifiers' inter-dependencies to learn the dynamic classifier combination rule, which may improve its robustness to the challenges high dimensionality and class overlap can pose for the dynamic selection task.

The main contributions of this work are then:

- A novel dynamic selection technique that uses a GNN multi-label meta-learner that takes advantage of the samples' relations, modeled as a graph, and classifiers' relations, modeled as meta-labels, to learn the dynamic selection task to better deal with high dimensionality and class overlap;
- A comprehensive experimental analysis including 35 high dimensional datasets, two ensemble methods of differing characteristics, and 10 dynamic selection techniques with different types of locality definition;
- An investigation on the limitations of the state-of-the-art dynamic selection methods in the presence of high dimensionality and class overlap and the scenarios in which the proposed modeling of the samples' and classifiers' relations help overcome these issues.

This work is organized as follows. We present a brief background on graph convolutional networks in Section 4.2. The proposed method is then introduced in Section 4.3. Section 4.4 presents the related work in the field of dynamic ensemble selection. In Section 4.5 and Section 4.6 we report and analyze the experiments conducted in this work. Lastly, our conclusions are summarized in Section 4.7.

4.2 Background

Graphs are structures able to represent the different interactions between the individual data elements, usually as a set of edges and nodes, respectively. Besides their high expressive power, graphs are also quite prevalent as they can naturally model several real-world problems, such as product recommendation (Berg, Kipf & Welling, 2017), traffic prediction (Zhang, Shi, Xie, Ma, King & Yeung, 2018) and text classification (Kipf & Welling, 2017), among many others. Because of that, tasks such as node and graph classification and link prediction have attracted a lot of attention in the machine learning community (Xia, Sun, Yu, Aziz, Wan, Pan & Liu, 2021).

Nevertheless, learning from graph-structured data may be challenging as, in order to exploit the complex relations encoded in the graph, its non-Euclidean structural information needs to be incorporated into a machine learning pipeline. Graph embedding methods (Cai, Zheng & Chang, 2018; Hamilton, 2020) provide a solution to the issue by learning the graphs' representation so that the data information is encoded into a low-dimensional Euclidean space. This can be quite advantageous as it allows the application of classical learning methods to extract information from the complex structure of the graph and make predictions.

Traditional embedding methods, however, rely on shallow learning which, by means of matrix factorization or random walks statistics (Cai *et al.*, 2018), learn an embedding matrix that functions as a "lookup" encoder for the nodes of the graph. Shallow methods present some limitations including scalability, as the parameters of the encoder are learned for each node in the graph, and the inability to produce embeddings for unseen nodes (Hamilton, 2020). On the other hand, embedding methods based on deep learning models were shown to overcome most or all of the limitations of the shallow approaches (Zhang, Cui & Zhu, 2022).

A subset of such models, graph convolutional networks define the convolution operation on graphs, which combines node information and produces high-level node representations (Xia *et al.*, 2021). The convolution operation can be defined in the *spectral* domain, based on the application of filters over the transformed graph signal, and on the *spatial* domain, based on the aggregation of the neighboring nodes' representation. The convolution operation in the spectral domain, however, depends on the eigenfunctions of the adjacency matrix's Laplacian, which may limit the application of the model over a graph whose eigenfunctions are different from the graph used for training. Spatial-based graph convolutional networks, on the other hand, perform the graph signal filtering in the vertex domain instead of the frequency domain, which allows the convolution operation to be generalized as certain aggregation functions applied to a node and its neighbors. Thus, due to their flexibility and efficiency, the spatial-based models have become quite popular (Zhang, Tong, Xu & Maciejewski, 2019), with recent works proposing an ensemble of graph convolutional networks (Nagarajan, Stevens & Raghunathan, 2022) and a message-passing mechanism for dealing with multi-relational graphs (Wang, Xu, Yu, Zhang, Li, Yang & Wu, 2022). These techniques fuse graph information, in the form of multiple GNNs as base-classifiers in the former and multiple node relations in the latter, differing from our proposed approach which uses a GNN to learn a dynamic fusion rule for the outputs of a given ensemble of classifiers. We briefly present a few spatial-based convolutional layers that are relevant to this work next, using the notation presented in Table 4.1.

Table 4.1 Notation pertaining to representation learning on graphs.

Symbol	Meaning
G	Graph
V	Set of vertexes
E	Set of edges
v_i	i-th vertex
$\mathbf{x}_i$	Attributes of v_i
$\mathbf{e}_{i,j}$	Attributes of the edge between v_i and v_j
$\mathbf{h}_i^l$	Hidden representation of v_i at the l-th layer
$\mathcal{N}(v_i)$	Set of neighboring nodes of v_i in G
$\mathbf{W}^l$	Learnable parameters at the l-th layer
$\sigma()$	Non-linear activation function

Graph Convolutional Network (GCN) (Kipf & Welling, 2017)

The GCN model proposed in (Kipf & Welling, 2017) uses a localized first-order approximation of spectral graph convolution, which in practice yields an information diffusion procedure performed in each layer of the network. As such, the model can be interpreted as a spatial-based graph convolutional network, with its convolution step shown in Equation 4.1, in which $L_{i,j}$ is a scalar calculated as a function of the adjacency matrix with self-loops and the degrees of nodes v_i and v_j (Gilmer, Schoenholz, Riley, Vinyals & Dahl, 2017). Moreover, the model originally uses the Rectified Linear Unit function as activation function $\sigma()$. The model was proposed for semi-supervised learning and performs node classification on graph data, a task at which it was shown to be quite powerful despite its simplicity (Kipf & Welling, 2017; Xu, Hu, Leskovec & Jegelka, 2019). However, the model in principle works only for transductive learning, so it cannot produce embeddings for unseen nodes.

$$\mathbf{h_i}^{l+1} = \sigma(\mathbf{W}^l \cdot \sum_{v_j \in \mathcal{N}(v_i)} L_{i,j}\mathbf{h}_j^l) \tag{4.1}$$

GraphSAGE (Hamilton *et al.*, 2017)

Another spatial-based GCN is the GraphSAGE model (Hamilton *et al.*, 2017), which can also be seen as an extension of the GCN for inductive learning. The model learns a set of functions that aggregate the features from neighboring nodes to produce the node embeddings.

Equation 4.2 shows the convolution operation applied at the l-th layer in the GraphSAGE model. The concatenation operation is referenced as $CCT()$, and the function $AGG^l()$ is the aggregation function at layer l, with the mean, max-pooling, and long short-term memory (LSTM) functions among the options.

$$\mathbf{h_i}^{l+1} = \sigma(\mathbf{W}^l \cdot CCT(\mathbf{h}_i^l, AGG^l(\{\mathbf{h}_j^l, \forall v_j \in \mathcal{N}(v_i)\}))) \tag{4.2}$$

During the forward propagation, the model aggregates, at each layer, information from nodes from an increasing hop neighborhood. Algorithm 4.1 (adapted from (Hamilton *et al.*, 2017)) describes the forward propagation procedure, where the representation of all nodes $v_i \in V$ is computed from the first layer $l = 1$ to the last layer L sequentially, so that the representations of all vertexes at layer l are obtained after their representations at layer $l - 1$ are obtained. The learnable parameters $\mathbf{W}^l$ (and aggregator functions parameters, if trainable) are then optimized using stochastic gradient descent using a proposed unsupervised loss or a regular supervised loss. The training of the network can also be done in mini-batches by sampling and producing the embeddings of the required nodes from each batch, making the model suitable to address large graphs.

The network is thus optimized to learn a general mapping function between the nodes' original feature space and the embedded space so that the latter contains the structural information from the original graph. In generalization, the model is then able to apply the trained aggregation functions over unseen nodes, either from an expanded version of the original graph or an entirely new graph, and produce their embeddings in the new feature space.

Algorithm 4.1 GraphSAGE forward propagation (Adapted from (Hamilton *et al.*, 2017)).

```
    input  : $G = (V, E)$ ;                                   ▹ Graph with its set of vertexes and edges
    input  : $X = \{\mathbf{x}_i, \forall v_i \in V\}$ ;      ▹ Input representation of the vertexes
    input  : $L, \{\mathbf{W}^l, \forall l \in [1, L]\}$ ;    ▹ Number of layers and their weight matrices
    input  : $\{AGG^l, \forall l \in [1, L]\}$ ;              ▹ Aggregation functions
    output : $H^L = \{\mathbf{h}_i^L, \forall v_i \in V\}$ ;  ▹ Nodes representation in the embedded space
 1  $\mathbf{h}_i^0 \leftarrow \mathbf{x}_i, \forall v_i \in V$ ;   ▹ Initial representation is the input features
 2  for every $l$ in $\{1, ..., L\}$ do
 3      for every $v_i$ in $V$ do
 4          $\mathbf{h}_{\mathcal{N}(v_i)}^l \leftarrow AGG^l(\mathbf{h}_j^{l-1}, \forall v_j \in \mathcal{N}(v_i))$ ;   ▹ Aggregate the neighbors' representations from layer $l - 1$
 5          $\mathbf{h}_i^l \leftarrow \sigma(\mathbf{W}^l \cdot CCT(\mathbf{h}_i^{l-1}, \mathbf{h}_{\mathcal{N}(v_i)}^l))$ ;   ▹ Obtain the representation at layer $l$
 6      end for
 7      Normalize $\mathbf{h}_i^l, \forall v_i \in V$
 8  end for
 9  $H^L \leftarrow \mathbf{h}_i^L, \forall v_i \in V$ ;   ▹ Representation of all vertexes at the last layer
10  return $H^L$
```

Graph Attention Network (Veličković, Cucurull, Casanova, Romero, Liò & Bengio, 2018)

In (Veličković *et al.*, 2018) a GNN architecture with a self-attention mechanism was proposed to naturally deal with graphs that present variable node degrees by allowing the assignment of different weights to the neighbors. The network applies a shared attention mechanism that integrates the graph structure through masked attention, so that the layer produces the nodes' hidden representations by attending over their first-order neighbors. This allows the model to not only be applicable in an inductive scenario, due to its shared parameters, but also to learn and apply different importances to the nodes from a given neighborhood, as opposed to the previously presented models. The latter characteristic is especially interesting from a local learning general standpoint as the network is able to focus on the connections that are more relevant to the task being learned, meaning the local operation is adaptively applied. It also provides an increase in the model capacity (Veličković *et al.*, 2018), which could be advantageous when dealing with complex local distributions.

The attention layer operation is described in Equation (4.3), in which **a** indicates the parameters of the attention mechanism, defined as a single layer feed-forward neural network, and α

(Equation (4.4)) represents the attention coefficients used to aggregate the neighbors' hidden representations. The model also originally uses the Leaky Rectified Linear Unit function as activation function $\sigma()$.

$$\mathbf{h_i}^{l+1} = \sigma(\sum_{v_j \in \mathcal{N}(v_i)} \alpha_{i,j}^l \mathbf{W}^l \cdot \mathbf{h}_j^l) \tag{4.3}$$

$$\alpha_{i,j}^l = \frac{\exp(\sigma(\mathbf{a}^\intercal \cdot CCT(\mathbf{W}^l \cdot \mathbf{h}_i^l, \mathbf{W}^l \cdot \mathbf{h}_j^l)))}{\sum_{v_m \in \mathcal{N}(v_i)} \exp(\sigma(\mathbf{a}^\intercal \cdot CCT(\mathbf{W}^l \cdot \mathbf{h}_i^l, \mathbf{W}^l \cdot \mathbf{h}_m^l)))} \tag{4.4}$$

4.3 Graph Neural Network Dynamic Ensemble Selection technique

We propose in this work the Graph Neural Network Dynamic Ensemble Selection (GNN-DES) technique, which exploits both the local information shared between the instances and the classifiers' inter-dependencies so that the learned dynamic classifier combination rule can better deal with class overlap and high dimensionality. To that end, we encode the relationships between the instances, derived from the ensemble behavior towards each pair of samples, into a graph. This allows mapping not only their similarities but also their class relations, thus outlining the existing overlap. In addition to that, the relationships between the classifiers are encoded in the problem's meta-labels. We then train a multi-label GNN which learns the dynamic classifier combination rule from the classifiers' inter-dependencies and local class structure of the data jointly. Internally, the GNN is expected to use both information to learn an embedded space where the dynamic selection task is potentially easier, in the sense of the locality assumption.

Thus, in order to characterize the local relations in the data, we define two types of links between the instances: *weak links* and *strong links*. Generally speaking, samples connected by weak links are the ones for which the ensemble produces a similar response even though they belong to different classes. These samples are weakly linked as they may share a subset of competent classifiers but they may not be relied upon given the indicated class overlap. Strong link connections, on the other hand, are formed between samples that belong to the same class and have a similar ensemble response. These samples are the most indicated for drawing information from as they may share a subset of competent classifiers without the issue of ambiguity. Each sample has at least one strong link with an instance from its own class, the one with the most similar ensemble response (or its *nearest friend*), and the maximum margin for the sample's connections is a function of this link.

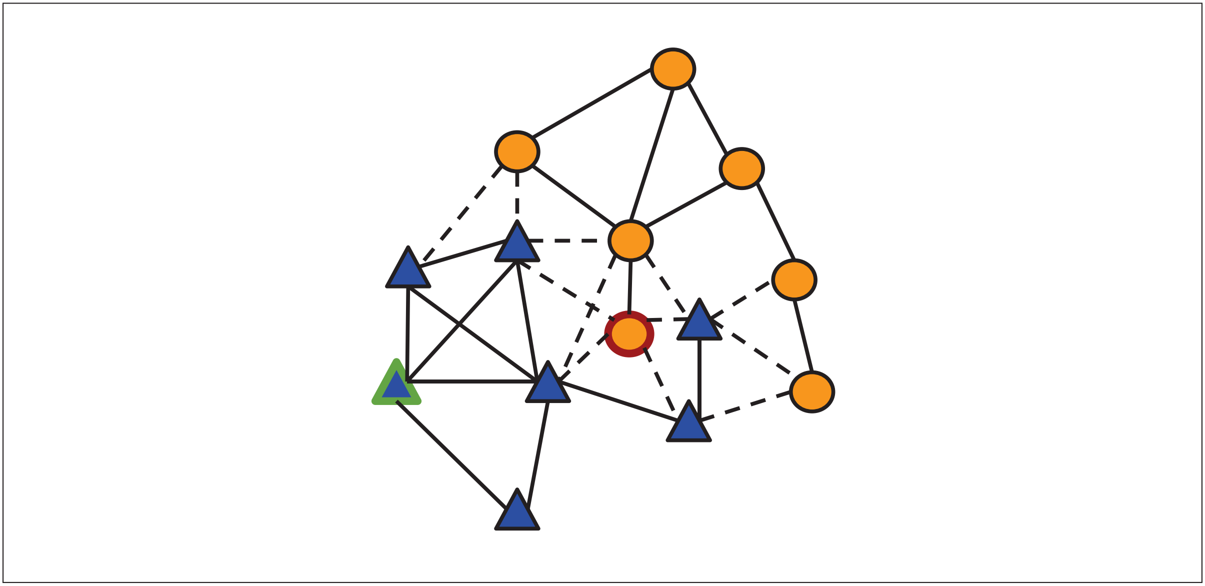


Figure 4.2 Example of weak and strong linkage applied in the proposed method. Weak links (dashed) are built between close samples from different classes while strong links are built between close samples from the same class. The instance highlighted in red presents mostly weak links, indicating an ambiguous decision region. The instance highlighted in green presents only strong links, suggesting a safe decision region.

Figure 4.2 illustrates the linkage concepts used in the proposed method, with the dashed and solid lines representing weak and strong links, respectively. The sample highlighted in red is an example of instances in an ambiguous continuous decision space, that is, the classifiers' responses for it differ little from the ones given to its nearest friend as well as four other samples from the opposite class, thus yielding one strong link and four weak links. The instance highlighted in green, on the other hand, has three strong connections in addition to the one with its nearest friend, and no weak connection as all instances from the opposite class are much further from it than its nearest friend.

With the meta-data thus encoded in the graph form following the basic principles described above, a network with a GNN core and a multi-label output layer is trained as our meta-learner for the dynamic selection task. Thus, given an unknown instance added to the original graph used during training, the meta-learner outputs the weights of each classifier's decision to produce the instance's predicted label.

We introduce the proposed technique in detail in Section 4.3.1. Then, in Section 4.3.2, we present an illustrative example of the method using a synthetic dataset. Throughout this section, we use the notation from Table 4.2 as well as from Table 4.1.

Table 4.2 Notation pertaining to the GNN-DES technique.

Symbol	Meaning
$\mathcal{T}$	Training set
Ω	Set of the problem's labels
C	Pool of classifiers
$G_{\mathcal{T}}$	Known graph
GNN	Meta-learner
y_i	True label of $\mathbf{x}_i$
c_k	k-th classifier from C
$p_{i,k}$	Output probability from c_k to $\mathbf{x}_i$
$\mathbf{p}_i$	Output probabilities for $\mathbf{x}_i$
$d_{i,j}$	Normalized *L1* distance between $\mathbf{p}_i$ and $\mathbf{p}_j$
$o_{i,k}$	Value of the network's k-th output node for input $\mathbf{x}_i$
$u_{i,k}$	Indication of (in)correct classification of $\mathbf{x}_i$ from c_k
$\mathbf{u}_i$	Meta-labels of $\mathbf{x}_i$
$\mathbf{x}_q$	Query/unknown instance
G_q	Evaluation graph
$\hat{y}_q$	Predicted label for $\mathbf{x}_q$

4.3.1 Description

Figure 4.3 describes the general steps of the proposed technique. In memorization, with a pool of classifiers C and the training set $\mathcal{T}$ as input, we first assign the meta-labels to the labeled instances according to the correct or incorrect classification from each classifier in the pool (Step 1). As multiple classifiers can label correctly the same instance, the meta-problem is designed as a multi-label problem.

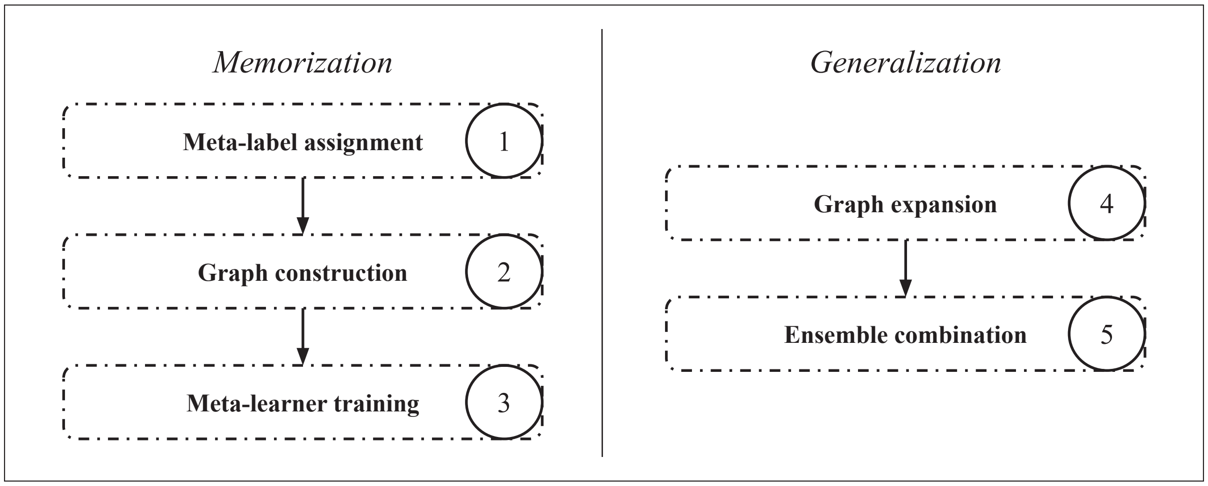


Figure 4.3 General steps of the GNN-DES technique.

In Step 2, we encode the meta-data associated with the responses of the classifiers over the labeled data into a graph representation ($G_{\mathcal{T}}$). The nodes represent the instances, described by their representation in the original feature space, and the edges represent their relationship, defined using the posterior probabilities yielded by the classifiers and, in the case of the training data alone, the available class labels. Then, in Step 3, the GNN is trained in a supervised manner using the known graph $G_{\mathcal{T}}$, the meta-labels obtained from Step 1, and the original representation of the instances.

In generalization, we add the unknown sample to the known graph to form the evaluation graph (G_q) in Step 4. Then, in Step 5, we induce the GNN using the query instance's original representation and G_q to yield its output, which in turn indicates the weights to be used for each classifier for the query sample's prediction.

Figure 4.4 outlines all steps of the proposed framework. We refer to it as we describe in detail the technique in Section 4.3.1.1 and Section 4.3.1.2.

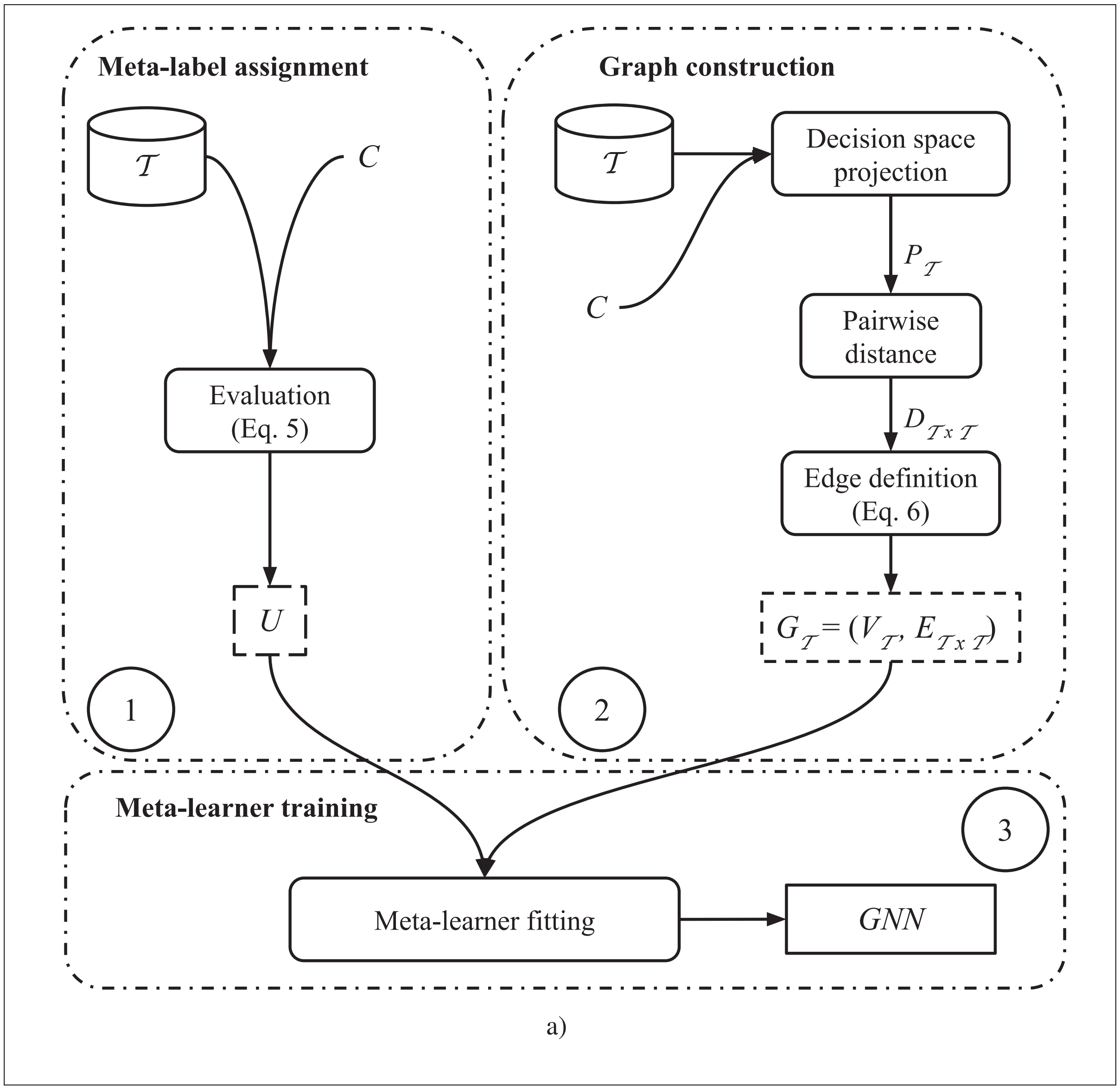


Figure 4.4 Description of the GNN-DES in (a) memorization and (b) generalization.

4.3.1.1 Memorization

Meta-label assignment

In Step 1 of Figure 4.4 we obtain the meta-labels of the known data. As we wish to learn how to select a subset of competent classifiers to label each query sample in particular, the meta-labels

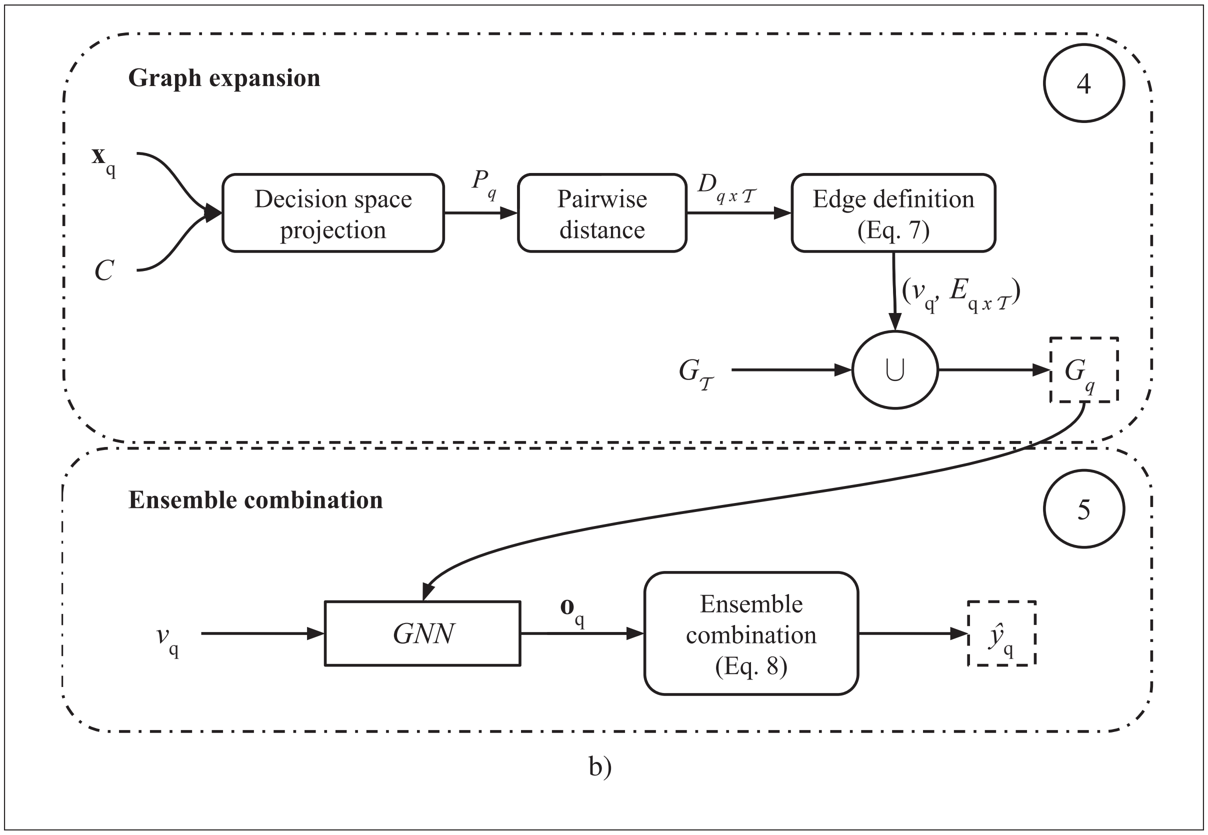


Figure 4.4 Description of the GNN-DES in (a) memorization and (b) generalization.

in our framework represent the ability of a given classifier to correctly label a given sample. The meta-labels are then obtained by evaluating the known data using the input set of classifiers C. We encode the meta-labels of each sample $\mathbf{x}_i$ in a vector form $\mathbf{u}_i$, obtained according to Equation (4.5). Thus, the meta-label assignment step yields a meta-label matrix U of size $|\mathcal{T}| \times |C|$, in which each row represents a sample and each column the indication of (in)correct classification of a classifier from the pool.

$$u_{i,k} = \begin{cases} 1, & if \, c_k(\mathbf{x}_i) = y_i \\ 0, & otherwise. \end{cases} \tag{4.5}$$

Algorithm 4.2 describes in more detail the meta-label assignment step of the proposed method. Taking the training set, the validation set and the pool of classifiers, we start with the meta-labels matrix empty. We then evaluate each labeled instance over each classifier in the pool from Step 2 to Step 6 of Algorithm 4.2 and return the meta-labels matrix U in Step 7.

Algorithm 4.2 Graph Neural Network Dynamic Ensemble Selection (GNN-DES) technique - Step 1: meta-label assignment.

```
   input  : $\mathcal{T}, C$ ;                                   ▹ Training set, pool of classifiers
   output : $U$ ;                                                ▹ Meta-labels in matrix form
1  $U \leftarrow []$ ;                                           ▹ Empty matrix
2  for every $\mathbf{x}_i$ in $\mathcal{T}$ do
3      for every $c_k$ in $C$ do
4          Set $u_{i,k}$ from $c_k, y_i$ according to Eq. (4.5)
5      end for
6  end for
7  return $U$ ;                                                  ▹ Meta-labels matrix
```

Graph construction

In Step 2 of the proposed method, we construct the graph which encodes the local information used as input to the meta-learner. In the graph, each vertex represents an instance, which is described by its feature vector, and the edges connect the samples that are closely located in the *decision* space, where the axes represent the responses of the set of classifiers.

Thus, to construct the graph we first project the labeled data into the continuous decision space, meaning we obtain, for each instance, the output probabilities given by all classifiers, yielding $P_{\mathcal{T}} = [\mathbf{p}_i, \forall \mathbf{x}_i \in \mathcal{T}]$, where $\mathbf{p}_i = [p_{i,1}, p_{i,2}, ..., p_{i,|C|}]$, as shown in Figure 4.4 (Step 2). For binary problems, which is the case presented in this section, only the output probabilities for the positive class (represented here as $p_{i,k}$ from classifier c_k for sample $\mathbf{x}_i$) are used to represent the samples in the decision space, whereas for multi-class problems, the output probabilities for all classes are used for representing the instances. We then calculate the pairwise L1 distance between the representations in $P_{\mathcal{T}}$, normalized by the ensemble size, and also by the number of labels in the multi-class scenario. We chose the L1 distance as it conveys how much the ensemble collectively disagrees in its response to two samples, as this determines their linkage based on our assumption w.r.t. their shared subset of competent classifiers. The normalized pairwise distances then yield the distance matrix $D_{\mathcal{T}\times\mathcal{T}}$, which is used to build the strong and weak links within the known graph $G_{\mathcal{T}}$ according to Equation (4.6).

The edges in the known graph are drawn so that each instance has at least one strong link, meaning they are connected to their nearest friend (i.e. sample from the same class in the continuous decision space), and its maximum margin for connection d^{max} is defined as the distance to the nearest friend plus a preset threshold τ (Equation (4.6)). For the strong links, the closer the ensemble response, the more reliable the connection between the instances, thus their edge weights are defined as the samples' similarity in the continuous decision space. An instance's weak links are formed with the samples from the opposite class that are also located within its maximum margin for connection. While they may share a subset of competent classifiers, justifying their linkage, the more similar the ensemble responses, the more ambiguous the area is, thus, we penalize the connection accordingly. Thus, the edge weight of a strong link follows a monotonically decrescent linear function with the distance until the maximum margin d^{max}, aiming at rewarding the similar ensemble behavior over similar instances from the same class. On the other hand, the weak link edge weight follows a monotonically crescent quadratic function with the distance until the same threshold to highly penalize the local class overlap so that the more similar the classifiers' responses, the weaker the link. Both edge weight functions are illustrated in Figure 4.5.

$$\begin{aligned} e_{i,j} &= \begin{cases} 1 - d_{i,j}, \mathit{if}\ (d_{i,j} \leq d_i^{max} \vee d_{i,j} \leq d_j^{max}) \wedge y_i = y_j, \\ d_{i,j}^2, \mathit{if}\ (d_{i,j} \leq d_i^{max} \vee d_{i,j} \leq d_j^{max}) \wedge y_i \neq y_j, \\ 0, otherwise, \end{cases} \\ d_i^{max} &= min(d_{i,k}, \forall \mathbf{x}_k \in \mathcal{T} | y_k = y_i) + \tau \end{aligned} \tag{4.6}$$

Algorithm 4.3 shows the graph construction step performed in memorization (Step 2 of Figure 4.4), which results in the known graph $G_{\mathcal{T}}$. The vertexes and edges sets start empty (Steps 1-2). Then, the training data is projected to the decision space (Step 3), and in Step 4 we calculate

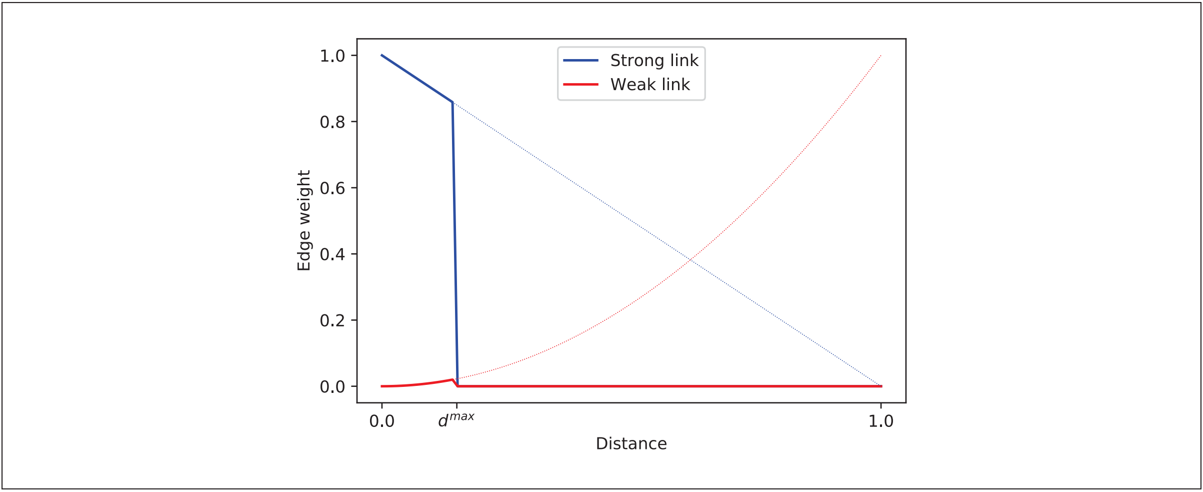


Figure 4.5 Edge weight function for strong and weak links (Eq. (4.6)). The strong and dotted lines indicate the edge weight values for distances below and above the d^{max} cut-off point, respectively.

Algorithm 4.3 Graph Neural Network Dynamic Ensemble Selection (GNN-DES) technique - Step 2: graph construction.

```
   input  : $\mathcal{T}, C, \tau$ ;                         ▷ Training set, pool of classifiers, preset threshold
   output : $G_{\mathcal{T}}$ ;                                                          ▷ Known graph
 1 $V \leftarrow \{\}$ ;                                                                   ▷ Vertexes set
 2 $E \leftarrow \{\}$ ;                                                                     ▷ Edges set
 3 $P_{\mathcal{T}} \leftarrow to\_decision\_space(\mathcal{T}, C)$ ;                 ▷ Project $\mathcal{T}$ to the decision space
 4 $D_{\mathcal{T}\times\mathcal{T}} \leftarrow pairwise\_distance(P_{\mathcal{T}}, P_{\mathcal{T}})$ ;         ▷ Obtain pairwise distance matrix
 5 for every $\mathbf{x}_i$ in $\mathcal{T}$ do
 6     $V \leftarrow V \cup v_i(\mathbf{x}_i)$ ;                                ▷ Add $v_i$ with its feature vector $\mathbf{x}_i$ to $V$
 7     for every $\mathbf{x}_j$ in $\mathcal{T}$ do
 8         Calculate $e_{i,j}$ from $D_{\mathcal{T}\times\mathcal{T}}, \tau, y_i, y_j$ according to Eq. (4.6) if $e_{i,j} > 0$ then
 9             $E \leftarrow E \cup e_{i,j}$ ;                                              ▷ Add edge to graph
10         end if
11     end for
12 end for
13 $G_{\mathcal{T}} \leftarrow (V, E)$ ;                                       ▷ Add vertexes and edges to $G_{\mathcal{T}}$
14 return $G_{\mathcal{T}}$
```

their pairwise distances in the projected space. From Step 5 to Step 13 we add to the vertexes set V all training instances (Step 6), and compute their pairwise edge weight (Step 8), adding the non-zero weighted edges to the edge set E (Step 10). The known graph $G_{\mathcal{T}}$ is then defined by V and E in Steps 14-15 and returned.

Meta-learner training

With the known graph $G_{\mathcal{T}}$ and the meta-labels U we fit the model in the meta-learner training step of the proposed framework, depicted in Figure 4.4 as Step 3. We train the GraphSAGE network in a supervised manner. We use the GraphSAGE model as it provides an efficient general framework for inductive learning on graphs (Zhang *et al.*, 2022; Hamilton *et al.*, 2017). As our meta-problem is multi-label, we set a dense layer as the output layer of the network with the sigmoid function as activation, and each output node is associated with a given classifier combination weight. The loss function optimized during training is the binary cross-entropy loss, weighted so that the harder to classify the sample, measured in the number of classifiers in the pool able to label it correctly, the higher its weight. The weights are included to encourage the network to focus on the more difficult samples in cases where a large number of classifiers can correctly label most of the samples. Equation (4.7) presents the loss function used to optimize the meta-learner, where O is the set of outputs, U is the set of meta-labels, $\mathcal{B}$ is the training batch, w_i the weight for the i-th sample, $\mathcal{L}_{BCE}(\mathbf{u}_i, \mathbf{o}_i)$ is the binary cross-entropy loss between the meta-labels and the network's output for the i-th sample, and ϵ a small value used for numerical stability.

$$\begin{aligned}\mathcal{L}(O,U) =& \frac{1}{\sum_{i=1}^{|\mathcal{B}|} w_i(\mathbf{u}_i)} \sum_{i=1}^{|\mathcal{B}|} w_i(\mathbf{u}_i) \times \mathcal{L}_{BCE}(\mathbf{u}_i, \mathbf{o}_i) \\ w_i(\mathbf{u}_i) =& |C| - \sum_{k=1}^{|C|} u_{i,k} + \epsilon\end{aligned} \tag{4.7}$$

Figure 4.6 broadly illustrates the procedure the meta-learner performs during the training step. The vector $\mathbf{x}_i$ indicate the i-th sample's representation in the feature space and the vector $\mathbf{u}_i$ its meta-labels, as described in Table 4.2. Suppose the sample $\mathbf{x}_1$ is input to the network with the GraphSAGE core of $L = 2$ layers. For the first layer, the instances from $\mathbf{x}_1$'s 1-hop neighborhood ($l = 1$, in green) are sampled, proportionally to their edge weights with $\mathbf{x}_1$. Then, the hidden representation of $\mathbf{x}_1$, $\mathbf{h}_1^1$ is obtained using the aggregated representation of its sampled neighbors (shown with green connections) and its original representation, so $\mathbf{h}_1^1 = f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5)$, with $f()$ described in more detail in Equation (4.2).

In the second layer of the GraphSAGE core, another set of instances are sampled, now in the 2-hop neighborhood ($l = 2$, in purple). The process repeats, with the hidden representations of the sampled instances (shown with purple connections) being aggregated and used to obtain the hidden representation of $\mathbf{x}_1$ in the second layer, $\mathbf{h}_1^2$, so $\mathbf{h}_1^2 = f(\mathbf{h}_5, \mathbf{h}_6, \mathbf{h}_7, \mathbf{h}_8, \mathbf{h}_9)$, $f()$ described in Equation (4.2). Then, in the last, dense output layer, the hidden representation of $\mathbf{x}_1$, $\mathbf{h}_1^2$, is used as input to produce the predicted meta-label vector $\hat{\mathbf{u}}_1$. The latter is used with its ground truth meta-label vector $\mathbf{u}_1$ to calculate the weighted binary cross-entropy for the sample and adjust the network weights using gradient descent.

Thus, the meta-learner is optimized using both the samples relationships, provided by the graph, and the classifiers' relationships, provided by the meta-labels. We then expect that, internally, it learns an embedded space where the locality assumption for the dynamic selection task is improved, so that the dynamic combination rule can be yielded in a straightforward manner at the output of the network.

4.3.1.2 Generalization

Graph expansion

The first step in generalization is to expand the known graph $G_{\mathcal{T}}$ by adding the query instance $\mathbf{x}_q$, forming the evaluation graph G_q (Step 4 of Figure 4.4). To that end, we project the unknown instance to the continuous decision space using the pool of classifiers C. Then, we obtain the distances between the projected sample and the labeled instances, building its (strong) links according to Equation (4.8). The non-zero weighted edges, as well as the query vertex and its description $\mathbf{x}_q$ are added to $G_{\mathcal{T}}$ to form the query graph G_q.

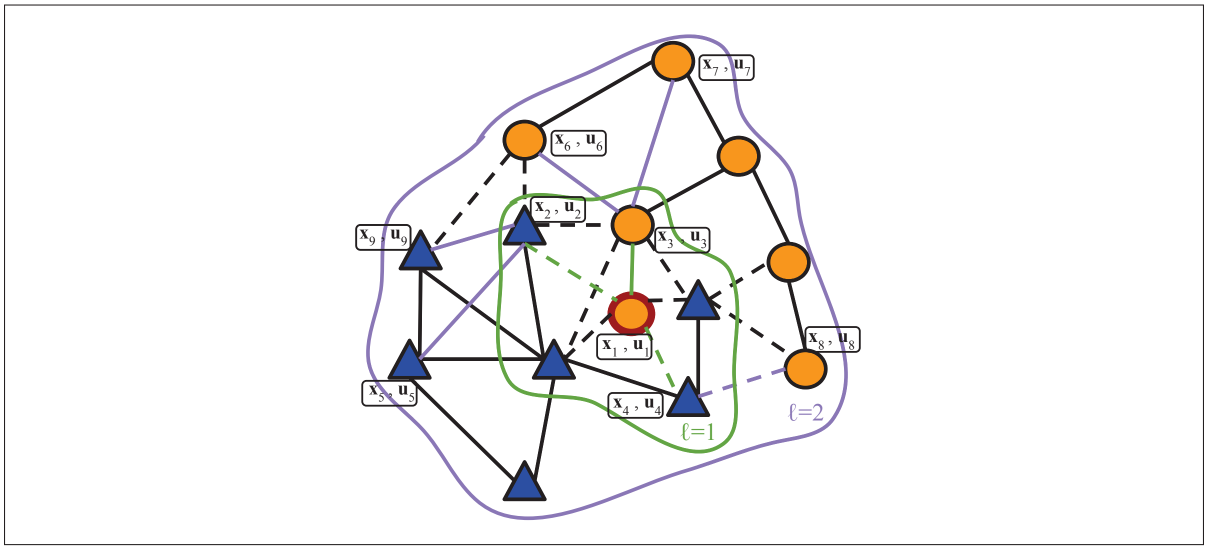

$\mathbf{x}_7$, $\mathbf{u}_7$
$\mathbf{x}_6$, $\mathbf{u}_6$
$\mathbf{x}_2$, $\mathbf{u}_2$
$\mathbf{x}_9$, $\mathbf{u}_9$
$\mathbf{x}_3$, $\mathbf{u}_3$
$\mathbf{x}_1$, $\mathbf{u}_1$
$\mathbf{x}_8$, $\mathbf{u}_8$
$\mathbf{x}_5$, $\mathbf{u}_5$
$\mathbf{x}_4$, $\mathbf{u}_4$
ℓ=2
ℓ=1

Figure 4.6 Illustrative example of the procedure the meta-learner performs during the training step. The vector $\mathbf{x}_i$ indicate the i-th sample's representation in the feature space and the vector $\mathbf{u}_i$ its meta-labels. The samples from $\mathbf{x}_1$ 1-hop neighborhood ($l = 1$) are circled in green, and the sampled instances among them have a green connection. Similarly, the samples from $\mathbf{x}_1$ 2-hop neighborhood ($l = 2$) are circled in purple, and the sampled instances among them have a purple connection.

$$e_{q,j} = \begin{cases} 1 - d_{q,j}, & \textit{if } d_{q,j} \leq d_q^{max}, \\ 0, & otherwise, \end{cases} \qquad (4.8)$$
$$d_q^{max} = \min(d_{q,k}, \forall \mathbf{x}_k \in \mathcal{T}) + \tau$$

Algorithm 4.4 describes the graph expansion procedure. The new set of edges E starts empty (Step 1). Then, the query instance is projected to the continuous decision space in Step 2, and its distance to the labeled data, also in the continuous decision space, is calculated in Step 3. From Step 4 to Step 9, the edge weights between the query instance and the known data are calculated, with the non-zero ones being added to the edge set (Step 7). Lastly, in Step 10, the query vertex and its edges are added to the known graph to form the evaluation graph G_q.

Algorithm 4.4 Graph Neural Network Dynamic Ensemble Selection (GNN-DES) technique - Step 5: graph expansion.

```
    input  : $\mathbf{x}_q, C, \tau, G_\mathcal{T}$ ;     ▷ Query instance, pool of classifiers, preset threshold, known graph
    input  : $P_\mathcal{T}$ ;     ▷ Training data in the decision space representation
    output : $G_q$ ;     ▷ Evaluation graph
 1  $E \leftarrow \{\}$ ;     ▷ Edges set
 2  $\mathbf{p}_q \leftarrow to\_decision\_space(\mathbf{x}_q, C)$ ;     ▷ Project $\mathbf{x_q}$ to the decision space
 3  $D_{q\times\mathcal{T}} \leftarrow pairwise\_distance(\mathbf{p}_q, P_\mathcal{T})$ ;     ▷ Obtain pairwise distance between $\mathbf{p}_q$ and $P_\mathcal{T}$
 4  for $every$ $\mathbf{x}_j$ $in$ $\mathcal{T}$ do
 5      Calculate $e_{q,j}$ from $D_{q\times\mathcal{T}}, \tau$ according to Eq. (4.8) if $e_{q,j} > 0$ then
 6          $E \leftarrow E \cup e_{i,j}$ ;     ▷ Add edge to graph
 7      end if
 8  end for
 9  $G_q \leftarrow G_\mathcal{T} \cup (v_q(\mathbf{x}_q), E)$ ;     ▷ Add query vertex and edges to the known graph
10  return $G_q$
```

Ensemble combination

The last step of the proposed technique, the ensemble combination, is depicted as Step 5 in Figure 4.4. In this step, the evaluation graph is used as input to the meta-learner yielding the output $\mathbf{o}_q$ for the vertex v_q. Then, the output of the network is used to weigh the votes of their respective classifiers, which are then aggregated to yield the predicted label y_q, according to Equation (4.9), where $\mathbb{1}()$ is the indicator function.

$$\hat{y}_q = argmax_{\omega\in\Omega} \sum_{k}^{|C|} o_{q,k} \times \mathbb{1}(c_k(\mathbf{x}_q) = \omega) \tag{4.9}$$

4.3.2 Illustrative example

To illustrate the proposed method, we use a balanced 2-dimensional two-class toy problem with a total of 50 instances. The dataset was split firstly into two sets, with 33% separated as test data and the remaining 66% for training/validation. We then generate an ensemble C with $|C| = 6$ classifiers, in this case a set of Decision Stumps obtained using AdaBoost (Freund & Schapire, 1997). Figure 4.7a depicts the training data and the classifiers' decision borders (in blue).

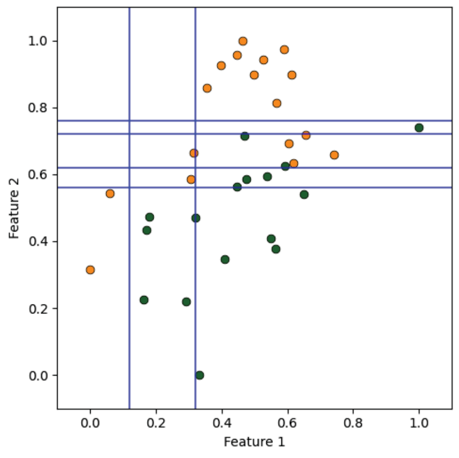
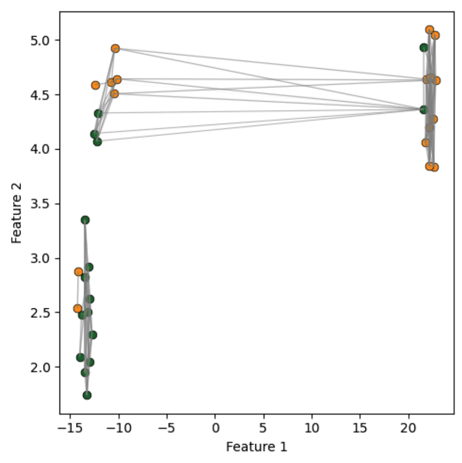


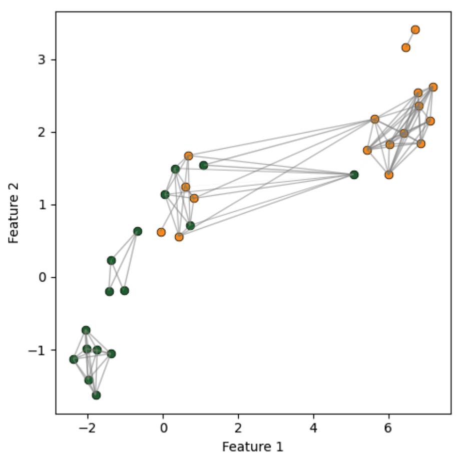
Figure 4.7 Training data and the ensemble of Decision Stumps (with decision borders in blue) in the original feature space.

In Step 1 of the proposed method (Figure 4.4a) we perform the meta-label assignment procedure, thus identifying which classifiers are able to label each instance correctly. That is, we obtain the Oracle information of the ensemble with respect to (w.r.t.) the training data, as the Oracle is the abstract model that always selects the correct classifiers for each instance (Kuncheva, 2002). Considering the input ensemble C, the Oracle accuracy rate, regarded as the theoretical upper limit on the performance of the ensemble (Didaci, Giacinto, Roli & Marcialis, 2005) over the toy's test set was 100%.

Then, in Step 2 of the proposed method, we project the known data to the decision space using the classifiers' output probabilities. Then, we connect the samples over which the ensemble yielded a similar response to obtain the edges of the known graph $G_{\mathcal{T}}$. The threshold used in this example was $\tau = 0.05$. Figure 4.8a shows the Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) (McInnes, Healy & Melville, 2018) feature reduction of the samples' decision space representation and the known graph's edges, obtained using Equation (4.6) with $\tau = 0.05$.



a)



b)

Figure 4.8 2-dimensional representation (obtained using UMAP) of the training data and the known graph in (a) the decision space, and (b) the embedded space.

With the known graph $G_{\mathcal{T}}$ and the corresponding meta-labels U at hand, we train the multi-label GNN, as shown in Step 3 of Figure 4.4a. In this toy experiment, we trained the GraphSAGE with two convolutional layers of size 16, neighborhood size for sampling of 5 in each layer, and attentional aggregator function. Figure 4.8b shows the UMAP feature reduction of the learned embedded representation of the known data, where we observe not only a greater separation between the problem's classes, but also between the samples from the same class itself. Since the network is optimized in a supervised way, these small clusters may indicate a certain similarity in the samples' labelsets.

In generalization, we first expand the known graph with the query sample node, according to the ensemble response (Step 4 of Figure 4.4b). Figure 4.9 shows the 2-dimensional representation of the evaluation graph in the decision space and in the embedded space, with the query sample indicated with a diamond marker. We can further observe the improvement in the embedded space's distribution compared to the decision space's, as the depicted query appears to be placed in a much better position in the former than in the latter. Using the evaluation graph as input to the network in Step 5 of the proposed method we obtain the weights of each classifier to obtain

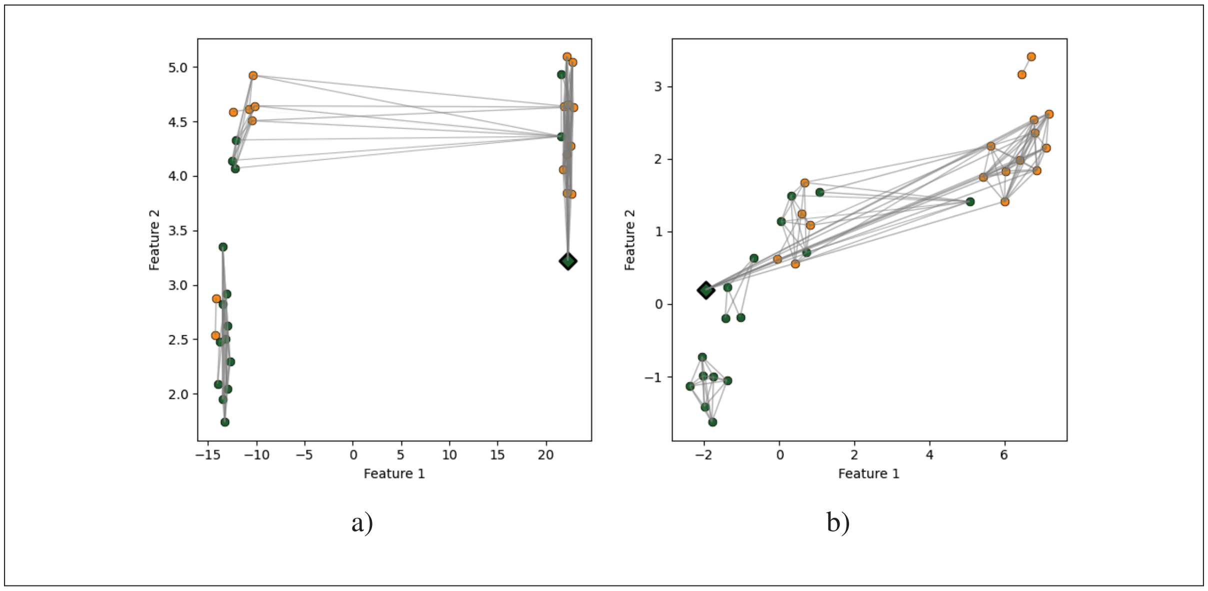


Figure 4.9 2-dimensional representation (obtained using UMAP) of the evaluation graph in (a) the decision space, and (b) the embedded space. The query sample is depicted with a diamond marker.

the output label of the query instance according to Equation (4.9). In terms of performance, the proposed method correctly labeled 15 of the 17 test samples, surpassing the AdaBoost combination rule (labeling correctly 11 test samples), the feature space-based DES technique K-Nearest Oracles Union (Ko *et al.*, 2007) (labeling correctly 12 test samples), and the decision space-based DES techniques K-Nearest Output Profiles (Cavalin *et al.*, 2012) (labeling correctly 11 test samples), both with neighborhood size $K = 7$.

To further visualize and assess the quality of the neighborhood produced over different feature spaces for the dynamic selection task over the depicted test sample, we compute the average intersection between the meta-labelset of the query and its neighbors, which indicates the degree of support the neighborhood provides for the true meta-labels of the query. Figure 4.10 shows the query instance (in diamond) and its neighbors (in squares) obtained in the feature space, decision space, and embedded space, all represented in the original feature space. We can see from Figure 4.10 that the neighborhoods obtained over the feature space and the decision space contain at most one instance from the query's class, while the one obtained over the embedded space yielded by the GNN is composed of samples from the green class only. Computing the average meta-label support over the three neighborhoods, the neighborhood obtained in the original feature space has an average meta-label intersection of 0.09, the one obtained in the decision space has an average meta-label intersection of 0.00, and the one obtained in the embedded space has an average meta-label intersection of 0.67. This suggests that the neighborhood in the embedded space optimized for the dynamic selection task is more favorable to the latter than the original feature space and decision space.

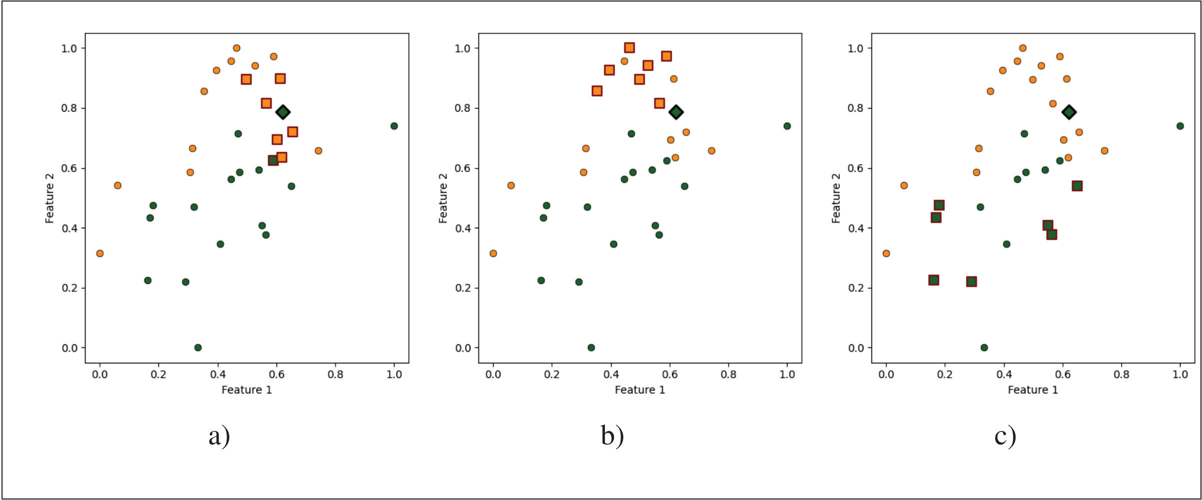


Figure 4.10 Query instance (diamond) and respective neighbors (square) obtained using the KNN ($K = 7$) over the (a) feature space, (b) decision space, and (c) embedded space, shown in the original feature space.

Figure 4.11 depicts the same data on the embedded space after reducing its dimensionality from 16 to 2 using UMAP. We can see that the network clustered the data mostly by class but not necessarily to one cluster each, indicating that other sources of information are taken into account in the optimization process, as intended. We can also observe the neighbors from Figure 4.10c in the vicinity of the query instance in the embedded space.

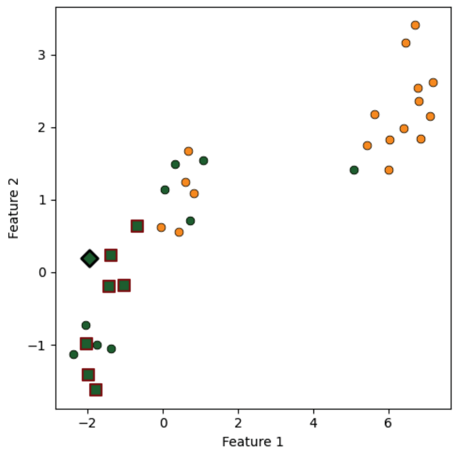


Figure 4.11 Query instance (diamond) and respective neighbors (square) obtained using the KNN ($K = 7$) over the embedded space (with graph edges), shown in the embedded space in two dimensions obtained with UMAP.

4.4 Related work

Dynamic selection techniques single out a subset of classifiers from a pool of classifiers to label each query instance in particular, with the purpose of using only the ones deemed competent enough for the task. The selection process is usually performed in three steps: region of competence (RoC) definition, competence estimation and classifier selection (Cruz *et al.*, 2018a). In the first step, a local region around the query sample, called the region of competence, is defined using clustering methods (Soares *et al.*, 2006), nearest neighbors rule (Cruz *et al.*, 2015a; Ko *et al.*, 2007; Cavalin *et al.*, 2012; Souza *et al.*, 2019b), potential function model (Woloszynski & Kurzynski, 2011), recursive partitioning (Souza *et al.*, 2023; Biedrzycki & Burduk, 2020), and/or fuzzy hyperboxes (Davtalab *et al.*, 2022) over a labeled dataset, referred to as the dynamic selection set (or DSEL set). In the second step, the competence of the classifiers in the pool is estimated in the RoC based on one or multiple criteria, including local accuracy (Ko *et al.*, 2007), classifier behavior (Cavalin *et al.*, 2012), and ensemble diversity (Soares *et al.*, 2006), meta-learning (Cruz *et al.*, 2015a; Pinto *et al.*, 2016), among others. In the last step, one or several classifiers are selected to perform the query instance's classification. If the former, the technique is referred to as a dynamic classifier selection (DCS) technique. If the latter, we consider it a dynamic ensemble selection (DES) technique.

Local-based DES techniques were often shown to perform well over problems that present challenging characteristics, including class imbalance, compared to static selection schemes (Oliveira *et al.*, 2017). However, the success of these techniques is strongly linked to the distribution of the DSEL, in a global view, and the RoC, in an individual view. Characteristics such as the size of the DSEL (Cruz *et al.*, 2015b) and local class overlap in the RoC (Cruz *et al.*, 2018b; Souza *et al.*, 2019a) were already shown to impact the performance of the local-based dynamic selection techniques. This is somewhat expected as in general the RoC is defined using a dissimilarity metric applied in a given space, most often the Euclidean distance applied in the original feature space. Thus, these techniques assume that the samples' similarity in the space indicates they share a similar set of competent classifiers. Nevertheless, this assumption may fail more frequently in complex distributions, as class overlap, data dimensionality and sparsity were shown to negatively affect the performance of local methods (Costa *et al.*, 2009; Sánchez *et al.*, 2007), and can impact the quality of the acquired region for the dynamic selection task.

Several local-based DES techniques which directly address certain issues, especially class overlap, in the RoC data distribution were proposed in the literature. In (Pereira *et al.*, 2018), the instances from the RoC are filtered out according to their discrimination index, obtained by applying Item Response Theory (IRT) over the pool of classifiers, with the purpose of using only the most discriminative ones in the competence estimation step. The RoC is also edited in (Oliveira *et al.*, 2018) and (Souza *et al.*, 2022), the former based on the local class distribution and the latter based on instance characterization, to reduce the effects of the class imbalance in overlap regions. In (Li *et al.*, 2019), a must link and a cannot link graph are constructed in the RoC to map the relationships between the instances, taking into account the class overlap in the area. Each classifier's competence is then estimated as a function of its overall local accuracy and a score dependent on the RoC distribution and its response to it, which is maximized when the classifier gives different responses to samples from different classes and similar responses to samples from the same class or close to one another. While these local-based DES techniques attempt to reduce the impact a poorly defined or complex RoC has on the selection task, they still heavily (or in most cases, solely) depend on the previously defined local distribution and disregard the relationship between the classifiers, which might be helpful in difficult cases where the locality assumption fails.

We can also find in the literature a class of DES techniques that completely skips the RoC definition step. Methods such as the Chained Dynamic Ensemble (CHADE) (Pinto *et al.*, 2016) and the Probabilistic Classifier Chain Dynamic Ensemble Selection (PCC-DES) (Narassiguin *et al.*, 2017) define the DES task as a multi-label problem in the meta-level, in which the meta-labelset indicate the classifiers that are competent to label a given instance. Both techniques make use of a multi-label classifier as the meta-learner which is responsible for yielding the ensemble combination rule for each input query sample, without explicitly taking into account its local context. In CHADE, a Classifier Chain ensemble (Read, Pfahringer, Holmes & Frank, 2011) is used as the meta-learner, while in the PCC-DES a Probabilistic Classifier Chain ensemble (Cheng, Hüllermeier & Dembczynski, 2010) and Monte-Carlo sampling are applied. Thus, these techniques learn the inter-dependencies between the classifiers to solve the DES meta-problem instead of relying on the locality assumption, which could be useful in situations where the RoC presents a complex or unreliable distribution, though possibly lacking otherwise. However, the order of the classifiers in the chain greatly affects the performance of the learner (Read *et al.*, 2011; Pinto *et al.*, 2016). This issue can be reduced by using an ensemble of classifier chains, though it increases significantly its cost. The PCC-DES technique has a particularly high computational overhead in generalization as the DES task loss minimization requires a search with a quadratic time cost on the sample size (Narassiguin *et al.*, 2017).

Table 4.3 presents a non-exhaustive list of DES techniques and their characteristics w.r.t. their use of local and classifiers' inter-dependencies information. We can see that, as discussed previously, defining the local region can be performed using several procedures, or can simply not be performed at all, as is the case of CHADE and PCC-DES. Building more complex structures to characterize the local relationships between the instances can also be performed, as the GDEP does by building graphs in the already defined RoC, and the FH-DES does by

Table 4.3 Characteristics of several DES techniques. The column *Acronym* indicates the acronym adopted in this work.

Name	Acronym	RoC definition	RoC definition space	Multilabel learning
DES-Clustering (Soares *et al.*, 2006)	DES-KMEANS	Clustering	Feature space	No
K-Nearest Oracles Eliminate (Ko *et al.*, 2007)	KNE	Nearest neighbors	Feature space	No
K-Nearest Oracles Union (Ko *et al.*, 2007)	KNU	Nearest neighbors	Feature space	No
Dynamic Ensemble Selection-KNN (Soares *et al.*, 2006)	DKNN	Nearest neighbors	Feature space	No
Online Local Pool (Souza *et al.*, 2019b)	OLP	Nearest neighbors	Feature space	No
Randomized Reference Classifier (Woloszynski & Kurzynski, 2011)	RRC	Potential function	Feature space	No
Online Local Pool++ (Souza *et al.*, 2023)	OLP++	Recursive partitioning	Feature space	No
Fuzzy Hyperboxes DES (Davtalab *et al.*, 2022)	FH-DES	Fuzzy hyperboxes	Feature space	No
K-Nearest Output Profiles (Cavalin *et al.*, 2012)	KNOP	Nearest neighbors	Decision space	No
META-DES (Cruz *et al.*, 2015a)	META-DES	Nearest neighbors	Feature & decision spaces	No
Graph-based Dynamic Ensemble Pruning (Li *et al.*, 2019)	GDEP	Nearest neighbors (Graph)	Decision space	No
Chained Dynamic Ensemble (Pinto *et al.*, 2016)	CHADE	-	-	Yes
Probabilistic Classifier Chain DES (Narassiguin *et al.*, 2017)	PCC-DES	-	-	Yes
Proposed	GNN-DES	Radius neighbors (Graph)	Decision space	Yes

defining fuzzy hyperboxes associated with the classifiers' competences/incompetences. For the local-based DES techniques, the most common space where the local region is defined is the original feature space, though several techniques perform the RoC definition (additionally or not) in the decision space. Lastly, the techniques that learn the inter-dependencies between the classifiers define the DES task as a multi-label problem and use a meta-classifier that allows the learning of the meta-labels jointly.

It can be observed that the proposed method, described in the last row of Table 4.3, combines the local and the multi-label approaches to provide information regarding samples' and classifiers' relationships to the meta-learner and thus learn the dynamic classifier combination rule. Moreover, the local information is encoded in a graph structure, similar to the GDEP, so that the local class distribution is considered when dealing with overlapped areas. However, we encode all of the data into one graph, as opposed to GDEP which characterizes each RoC individually with two graphs, and introduce it to a GNN meta-learner which, as will be presented next, performs convolutions with sampled neighbors from different depths in the graph. That way, the RoC in the proposed method is not explicitly defined as a fixed region and is thus not constrained to the same set of instances as in the GDEP. Another difference between the proposed method and the local-based techniques is that the former is trained end-to-end, meaning that the system learns a dynamic selection rule from the graph and meta-labels information alone, while the other techniques require the definition of classifier competence criteria to be used directly or as handcrafted meta-features (as in the META-DES technique).

4.5 Experimental protocol

4.5.1 Research questions

We assess in this work through an experimental analysis the following research questions:

- **RQ1**: How do the DES techniques perform in the presence of high dimensionality and class overlap?
- **RQ2**: Does the DES techniques' behavior relates to the ensemble characteristics and/or their RoC definition procedure?
- **RQ3**: Does the proposed method surpass the static selection baseline and the different dynamic selection methods?
- **RQ4**: Does learning from the interactions between the classifiers and between the instances aid in overcoming the limitations of the current state-of-the-art techniques?

4.5.2 Datasets

Table 4.4 presents the characteristics of the datasets used in the experiments. We use the same test bed from (Souza *et al.*, 2023), excluding only the four datasets over which the AdaBoost ensemble method generated fewer than the set amount of classifiers in the pool. The chosen testbed contains two-class high dimensional small sample sized (HDSSS) datasets (with at least 100 features) taken from the OpenML repository (Vanschoren *et al.*, 2013). The column IR in Table 4.4 refers to the problems' imbalance ratio, that is, the ratio between the amount of majority class samples and the amount of minority class samples in the dataset. The column F/N indicates the ratio between the datasets' number of features and number of samples. The F/N ratio, which is also a data complexity measure (Lorena, Garcia, Lehmann, Souto & Ho, 2019), conveys a problem's data sparsity, which is associated with a higher classification difficulty as low density areas can make the model induction harder (Lorena *et al.*, 2019; Pascual-Triana, Charte, Andrés Arroyo, Fernández & Herrera, 2021). High data sparsity was also linked to the size and complexity of the class boundaries (Costa *et al.*, 2009; Ho & Basu, 2002), which may present a further challenge to the RoC definition for the DES techniques.

The performance evaluation was conducted using a 10-fold cross-validation procedure, with one fold for test and the remaining for training, using the same partitions provided in the OpenML repository for reproducibility. Due to the very limited number of instances in some problems (such as *leukemia* and *tumors_C*), we use the whole training set to generate the pool of classifiers and as the dynamic selection set (DSEL) in the experiments, similarly to (Cruz *et al.*, 2019a; Souza *et al.*, 2023). The DSEL is a set of labeled data used in dynamic selection techniques for the region of competence definition (Cruz *et al.*, 2018a).

4.5.3 Performance measures

We evaluate the models over the binary problems from Table 4.4 in terms of the balanced accuracy rate, or macro-averaged recall. The balanced accuracy rate for binary problems is defined as the average between the true positive rate (TPR) and the true negative rate (TNR), as shown in Equation (4.10). As recommended in (Flach, 2019), we focus on one performance metric only. Moreover, we chose the balanced accuracy rate as the testbed contains problems with widely varying imbalance ratios, ranging from 1.03 to 70.09, so the accuracy rate would not be adequate the assess the performance. However, since the focus of this work is not to investigate the effects of class imbalance, we found that the balanced accuracy rate is an adequate metric as it is sensitive to the class imbalance but does not focus only on one class nor penalizes too much a poor performance over one or the other. For the statistical comparisons between the models' performances over multiple datasets, we also use the pairwise Wilcoxon signed-rank test, as recommended in (Demšar, 2006; Benavoli *et al.*, 2016), since it is non-parametric and its result does not depend on the group of techniques included in the comparative analysis.

Table 4.4 Characteristics of the datasets used in the experiments.

Ref. #	Dataset	Instances (N)	Features (F)	IR	F / N
1	tumors_C	60	7129	1.86	118.82
2	leukemia	72	7129	1.88	99.01
3	AP_Endometrium_Lung	187	10935	2.07	58.48
4	AP_Omentum_Uterus	201	10935	1.61	54.4
5	AP_Omentum_Lung	203	10935	1.64	53.87
6	AP_Lung_Uterus	250	10935	1.02	43.74
7	AP_Omentum_Ovary	275	10935	2.57	39.76
8	AP_Ovary_Uterus	322	10935	1.6	33.96
9	AP_Omentum_Kidney	337	10935	3.38	32.45
10	AP_Colon_Prostate	355	10935	4.14	30.8
11	AP_Colon_Omentum	363	10935	3.71	30.12
12	AP_Uterus_Kidney	384	10935	2.1	28.48
13	AP_Endometrium_Breast	405	10935	5.64	27
14	AP_Breast_Prostate	413	10935	4.99	26.48
15	AP_Breast_Omentum	421	10935	4.47	25.97
16	AP_Colon_Ovary	484	10935	1.44	22.59
17	AP_Colon_Kidney	546	10935	1.1	20.03
18	AP_Breast_Kidney	604	10935	1.32	18.1
19	OVA_Endometrium	1545	10935	24.33	7.08
20	OVA_Uterus	1545	10935	11.46	7.08
21	OVA_Ovary	1545	10935	6.8	7.08
22	OVA_Breast	1545	10935	3.49	7.08
23	fri_c4_100_100	100	100	1.13	1
24	tecator	240	124	1.35	0.52
25	fri_c4_250_100	250	100	1.27	0.4
26	gina_agnostic	3468	970	1.03	0.28
27	gina_prior	3468	784	1.03	0.23
28	fri_c4_500_100	500	100	1.3	0.2
29	spectrometer	531	101	8.65	0.19
30	scene	2407	299	4.58	0.12
31	mfeat-pixel	2000	240	9	0.12
32	mfeat-factors	2000	216	9	0.11
33	fri_c4_1000_100	1000	100	1.29	0.1
34	yeast_ml8	2417	116	70.09	0.05
35	sylva_prior	14395	108	15.25	0.01

$$\text{Balanced accuracy rate} = \frac{TPR + TNR}{2} \tag{4.10}$$

4.5.4 Classifier models and hyperparameters

We evaluate the DES techniques using two ensembles: 100 Perceptrons generated using Bagging (Breiman, 1996), and 100 Decision Stumps generated using AdaBoost (Freund & Schapire, 1997). We evaluate the two different ensemble methods with different base-classifier models so that they yield two pools with distinct characteristics, in terms of how strong and/or redundant each set of classifiers is, in order to analyze how they impact the techniques' behavior over the HDSSS datasets.

The DES techniques used in the comparative analysis which use the two linear ensembles were the K-Nearest Oracles Union (KNU) (Ko *et al.*, 2007), K-Nearest Oracles Eliminate (KNE) (Ko *et al.*, 2007), Dynamic Ensemble Selection-KNN (DKNN) (Soares *et al.*, 2006), K-Nearest Output Profiles (KNOP) (Cavalin *et al.*, 2012), META-DES (Cruz *et al.*, 2015a), Randomized Reference Classifier (RRC) (Woloszynski & Kurzynski, 2011), Chained Dynamic Ensemble (CHADE) (Pinto *et al.*, 2016). These techniques have different RoC definition procedures and selection criteria and are often found among the best-performing DES techniques in the literature (Cruz *et al.*, 2018a). Moreover, given the extensive list of evaluated techniques and ensemble methods, we do not include any clustering-based method in the experiments as they are generally outperformed by the KNN-based and potential function-based methods (Cruz *et al.*, 2018a) and are more limited in the design of the selected ensemble due to the local region granularity (Soares *et al.*, 2006; de Souto *et al.*, 2008). We also include in the experimental analysis three local ensemble methods that generate their own pool of classifiers, namely the Online Local Pool generation technique (OLP) (Souza *et al.*, 2019b), and the tree-based OLP++ (Souza *et al.*, 2023) and the Forest of Local Trees (FLT)(Armano & Tamponi, 2018), as they integrate the local information in both the generation and the integration part of the MCS.

Table 4.5 shows the hyperparameters of the DES techniques used in the comparative analysis. The hyperparameters were set as recommended in their papers if no implementation is available in the DESLib (Cruz *et al.*, 2020) library, or otherwise to their default value. In addition to those, we include in the analysis the Oracle model, an abstract selector which always chooses the correct classifier for each instance, and we use as baselines two static combination approaches, the majority voting (MV) for the Bagging ensemble, and the AdaBoost combiner for the AdaBoost ensemble (ADA). The experiment was implemented in *Python*, using the libraries *TensorFlow*, *DESLib* (Cruz *et al.*, 2020), *scikit-multilearn* (Szymański & Kajdanowicz, 2019), *scikit-learn* (Pedregosa *et al.*, 2011) and *StellarGraph* (Data61, 2018).

Table 4.5 Hyperparameter setting of the local ensembles included in the comparative analysis. The pool of classifiers used in all DS techniques but the FLT, OLP and OLP++ is the same used as in the static selection baselines.

Acronym	Hyperparameters
KNU	K=7
KNE	K=7

DKNN	K=7
	pct_accuracy=0.5
	pct_diversity=0.3

KNOP	K=7

META-DES	K=7
	$K_p = 5$
	meta_classifier=$NaiveBayes$
	mode=$selection$
	h_c=1.0

RRC	mode=$selection$
CHADE	meta_classifier=$DecisionTree$

FLT	max_features=0.3
	sample_percent=100
	max_number_leaves=∞

OLP	K=7
	$\lvert LP\rvert = 5$
	selection=$MultipleClassifierBehavior$ (Giacinto *et al.*, 2000)

OLP++	T=100 $L = 1$

GNN-DES

The technique is evaluated with $\tau = 0.05$, which was chosen arbitrarily as it would amount to a disagreement of a very small part of the ensemble (5% of the classifiers) over two samples. For the training of the meta-classifier, a stratified random sample of 20% of the training set is used for validation/early stopping. The validation nodes are connected to the known graph $G_{\mathcal{T}}$ as if unknown samples, using Equation (4.8). The procedure for including the validation data is described in more detail in the supplementary material.

The GNN core contains two graph convolutional layers of 512 units, as in (Salehi & Davulcu, 2020), with attention aggregation function from (Veličković *et al.*, 2018), as the local samples may have distinct importances for the DES task, and one dense output layer, as in the fully supervised model in (Hamilton *et al.*, 2017), with sigmoid activation to produce the multi-label output. A few adaptations were necessary to cope with the little amount of data in the HDSSS problems: we sample only 5 instances at each convolutional layer, as opposed to the 25 and 10 applied in (Hamilton *et al.*, 2017) over large graphs, and apply L_2 regularization with $\lambda = 0.01$, which we empirically found to aid the training process according to the validation loss curves. The network is trained over 150 epochs, with a patience of 30 epochs, a batch size of 300, and the adaptive learning rate is initially set to 0.005, as in (Veličković *et al.*, 2018). We also perform a sweep on the drop-out rate in the set $\{0.0, 0.2, 0.5\}$ as in (Salehi & Davulcu, 2020), and the model with the best micro-averaged multi-label precision score over the validation data is used as the meta-learner in generalization.

4.6 Experimental results

4.6.1 Ensemble and graph characteristics

We start by analyzing the characteristics of the two evaluated ensembles and the resulting graphs' degrees. Figure 4.12 shows the average labelset cardinality of instances of each dataset with respect to both ensembles. The labelset cardinality (LCard) of an instance $\mathbf{x}_i$ in our multi-label meta-problem is the number of classifiers in the pool that correctly labels it, or in other words, $LCard_i = \sum_{k=1}^{|C|} u_{i,k}$. We can observe that the bagged ensemble yielded a much higher average labelset cardinality compared to the AdaBoost ensemble in most datasets. This is expected as not only the base-classifier used in Bagging is stronger, but also the generation procedure in AdaBoost enforces classifier specialization directly. This can be further observed in Figure 4.13, which shows the average correlation coefficient measure (Kuncheva, 2014) of the two ensembles.

The correlation coefficient is a pairwise diversity metric that is defined for two-class classifiers as shown in Equation (4.11), where c_k and c_n are two binary models and $\mathbb{1}()$ is the indicator function. We can see that the AdaBoost technique yielded a much more diverse ensemble compared to the Bagging technique, with very few exceptions.

$$
\begin{aligned}
\rho_{k,n} &= \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}}, \\
a &= \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \mathbb{1}(c_k(\mathbf{x}_i) = y_i \wedge c_n(\mathbf{x}_i) = y_i), \\
b &= \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \mathbb{1}(c_k(\mathbf{x}_i) = y_i \wedge c_n(\mathbf{x}_i) \neq y_i), \\
c &= \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \mathbb{1}(c_k(\mathbf{x}_i) \neq y_i \wedge c_n(\mathbf{x}_i) = y_i), \\
d &= \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \mathbb{1}(c_k(\mathbf{x}_i) \neq y_i \wedge c_n(\mathbf{x}_i) \neq y_i).
\end{aligned}
\tag{4.11}
$$

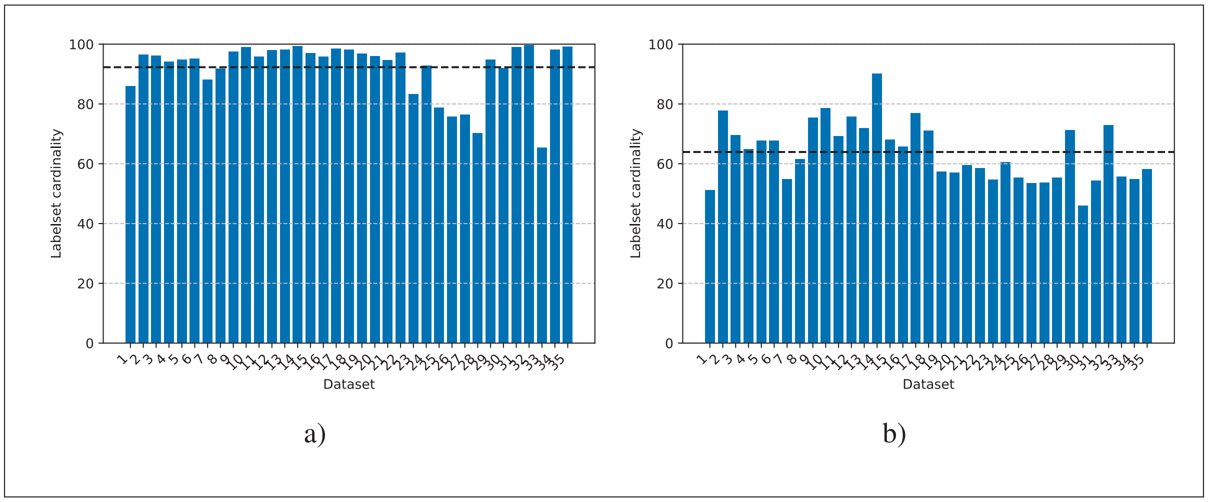


Figure 4.12 Labelset cardinality of the instances with respect to the (a) Bagging ensemble and the (b) AdaBoost ensemble. The dashed line indicates the average over all datasets.

One consequence of the classifiers' correlated responses is the degree of the nodes in the known graph $G_{\mathcal{T}}$ in the proposed method. Figure 4.14 shows the average node degree in $G_{\mathcal{T}}$ obtained from both ensembles. We can see that the ensemble generated using Bagging yielded a

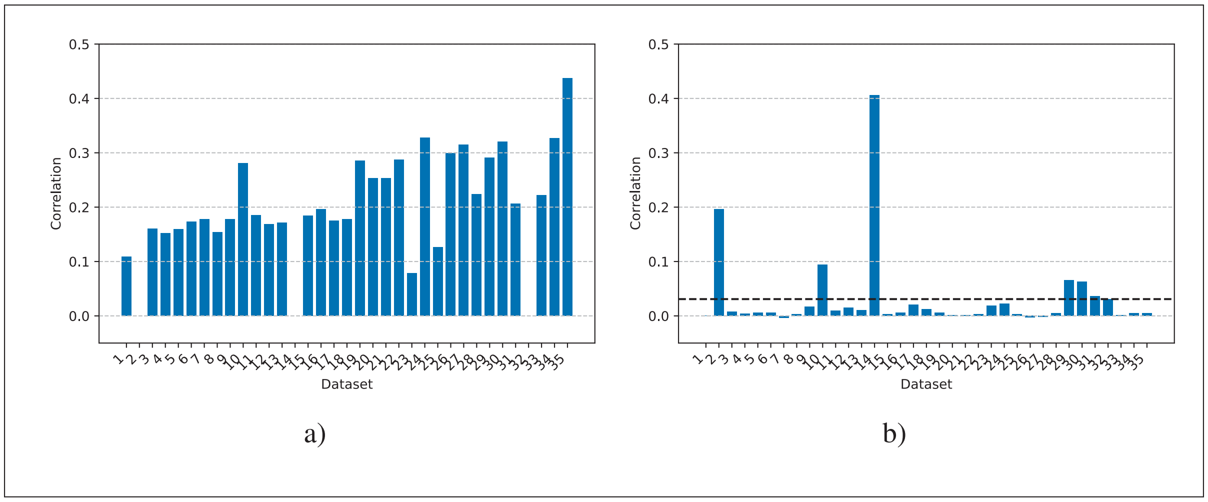


Figure 4.13 Average correlation coefficient of the (a) Bagging ensemble and (b) AdaBoost ensemble. The dashed line indicates the average over all datasets.

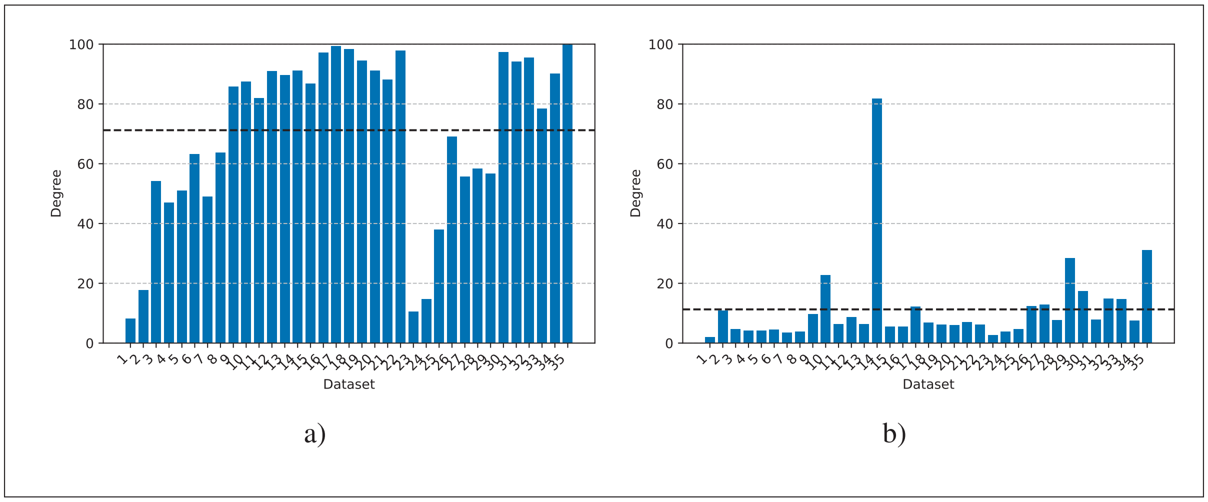


Figure 4.14 Average nodes' degree in the known graph $G_{\mathcal{T}}$ obtained using the (a) Bagging ensemble and (b) AdaBoost ensemble. The dashed line indicates the average over all datasets.

graph much more connected than the AdaBoost ensemble. This could be an indication that the distances are quite close to one another in the decision space of the bagged ensemble, which is expected if the classifiers tend to respond similarly to the same instances. Moreover, having an average degree that is too high reduces the locality information encoded in the graph which could negatively impact the learning of the dynamic selection rule.

4.6.2 Overall performance

In terms of performance, Table 4.6 and Table 4.7 show the average balanced accuracy rate of the techniques using both ensembles (with the caveat that the FLT, OLP and OLP++ techniques use a different ensemble altogether). It can be observed that using the bagged ensemble, the performances of the techniques were quite similar, except for the FLT, OLP and OLP++. In fact, observing the win-tie-loss distribution of the proposed method using the bagged ensemble in Figure 4.15a, we see a large number of ties with the techniques that use the same ensemble, with the exception of the DKNN. Performing a pairwise Wilcoxon-signed rank test over the results for all techniques, we can see in Table 4.8a that, using the bagged ensemble, all techniques except for the OLP yield a statistically similar performance to the majority voting combiner, including the proposed method, with $\alpha = 0.05$.

As the bagged ensemble presents a high degree of correlation between the classifiers, as shown in Figure 4.13, several issues may be at play to obtain such results. First, distinguishing between very similar classifiers might be in general more difficult during the selection process. We can also see that in only 10 datasets did the Oracle accuracy rate reach a perfect score, suggesting that the bagged ensemble was not able to fully cover the data, possibly as a consequence of its lack of diversity. Moreover, we already expected the local region definition in the feature space to be affected by the high dimensionality, as the comparative results of the feature space-based techniques and the majority voting combiner in Table 4.8a suggest. However, a less diverse ensemble also provides a decision space with correlated features, which in turn hinders the techniques that rely on the classifiers' responses for the local region definition. In the case of the OLP, though it does not use the correlated ensemble, its poor performance can be explained as it relies solely on the local information for producing the pool, so the local region definition in the original feature space, which can be challenging in HDSSS problems, greatly affects the method's behavior.

Table 4.6 Average balanced accuracy rate of the techniques over each dataset, using the Bagging ensemble.

Ref. #	Oracle	MV	KNU	KNE	DKNN	KNOP	META-DES	RRC	CHADE	FLT	OLP	OLP++	GNN-DES
1	1.0000	0.6208	0.6208	0.6333	0.6583	0.6333	0.5958	0.6208	0.6583	0.5167	**0.6833**	0.5542	0.6208
2	1.0000	**0.9667**	**0.9667**	**0.9667**	**0.9667**	**0.9667**	**0.9667**	**0.9667**	**0.9667**	0.9567	0.9300	**0.9667**	**0.9667**
3	0.9962	0.9022	0.9022	0.8984	0.9067	0.9022	0.9022	0.9022	0.9022	0.9051	0.9141	**0.9510**	0.9022
4	0.9958	0.8748	0.8787	0.8707	0.8835	0.8787	0.8748	0.8748	0.8748	0.8884	0.8609	**0.9030**	0.8748
5	0.9962	0.8883	0.8883	0.8927	0.8990	0.8844	0.8883	0.8883	0.8921	**0.9266**	0.8990	0.9022	0.8883
6	0.9962	0.8965	0.8965	0.8965	0.8926	0.8965	0.8965	0.8965	0.8926	**0.9446**	0.8644	0.9401	0.8965
7	0.9752	0.6631	0.6506	0.6122	0.6156	0.6577	0.6702	0.6702	0.6727	0.6836	0.6386	**0.7392**	0.6568
8	0.9912	0.8221	0.8263	0.8261	0.8239	0.8263	0.8261	0.8221	0.8266	0.8645	0.8049	**0.8709**	0.8221
9	1.0000	0.9285	0.9285	0.9223	0.9179	0.9285	0.9285	0.9285	0.9285	**0.9731**	0.9190	0.9591	0.9285
10	0.9983	**0.9823**	**0.9823**	0.9668	0.9668	**0.9823**	**0.9823**	**0.9823**	**0.9823**	**0.9823**	0.9525	0.9768	**0.9823**
11	0.9903	0.8765	0.8765	0.8719	0.8774	0.8765	0.8765	0.8765	0.8765	0.9102	0.8417	**0.9110**	0.8765
12	0.9981	0.9567	0.9567	0.9567	0.9609	0.9567	0.9567	0.9567	0.9567	**0.9808**	0.9301	0.9689	0.9567
13	1.0000	**0.9581**	**0.9581**	**0.9581**	0.9566	**0.9581**	**0.9581**	**0.9581**	**0.9581**	0.9135	0.8882	0.9081	**0.9581**
14	1.0000	0.9801	0.9801	0.9801	0.9801	0.9801	0.9801	0.9801	0.9801	**0.9845**	0.9702	0.9774	0.9801
15	0.9813	0.9216	0.9216	0.9216	0.9230	0.9216	0.9216	0.9216	0.9187	**0.9245**	0.8636	0.8962	0.9216
16	0.9871	0.9169	0.9169	0.9158	0.9102	0.9169	0.9169	0.9169	0.9169	**0.9554**	0.8815	0.9519	0.9169
17	1.0000	0.9720	0.9739	0.9739	0.9759	0.9720	0.9720	0.9720	0.9739	**0.9777**	0.9704	0.9740	0.9739
18	0.9918	0.9701	0.9701	0.9706	0.9701	0.9687	0.9701	0.9701	0.9687	0.9692	0.9576	**0.9755**	0.9701
19	0.9679	0.5798	0.5900	0.5967	0.5816	**0.5968**	0.5957	0.5889	0.5734	0.5000	0.5784	0.5315	0.5882
20	0.9753	0.7099	0.7130	**0.7335**	0.7276	0.7130	0.7085	0.7060	0.7088	0.5000	0.6823	0.6225	0.7126
21	0.9730	0.7426	0.7493	0.7447	0.7488	0.7555	0.7511	0.7454	0.7470	0.5000	0.7301	**0.7831**	0.7527
22	0.9859	0.9471	0.9460	0.9440	0.9440	0.9490	0.9471	0.9471	0.9442	0.5000	0.9200	**0.9498**	0.9475
23	1.0000	0.5408	0.5408	0.4983	0.4883	0.5592	0.5267	0.5408	0.5100	**0.7958**	0.5950	0.7767	0.5408
24	0.9964	0.9409	0.9409	0.9449	0.9313	0.9409	0.9378	0.9409	0.9409	**0.9455**	0.9120	0.9168	0.9409
25	1.0000	0.5740	0.5740	0.5643	0.5497	0.5740	0.5558	0.5760	0.5695	**0.8432**	0.5903	0.7377	0.5740
26	0.9954	0.7650	0.7694	0.7493	0.7659	0.7692	0.7584	0.7707	0.7593	0.9179	0.7198	**0.9369**	0.7648
27	0.9963	0.8114	0.8232	0.8207	0.8324	0.8285	0.8266	0.8247	0.8094	**0.9526**	0.8214	0.9441	0.8160
28	0.9895	0.6167	0.6183	0.5925	0.6171	0.6137	0.6117	0.6182	0.6107	**0.8771**	0.5695	0.7938	0.6155
29	1.0000	0.9141	0.8895	0.8968	0.8905	0.8711	0.8868	0.9130	**0.9224**	0.8743	0.8545	0.8664	0.9141
30	0.9919	0.8425	0.8472	0.8567	0.8517	0.8517	0.8515	0.8510	0.8423	**0.8822**	0.7566	0.7639	0.8446
31	0.9950	0.9761	0.9761	0.9789	0.9783	0.9764	0.9761	0.9761	0.9758	**0.9819**	0.9192	0.9697	0.9761
32	1.0000	0.9917	0.9917	**0.9919**	**0.9919**	0.9917	0.9917	0.9917	0.9917	0.9847	0.9836	0.9625	0.9917
33	0.9969	0.6490	0.6471	0.5997	0.6447	0.6466	0.6470	0.6524	0.6318	**0.9020**	0.6017	0.8421	0.6471
34	0.9208	0.5158	0.5158	**0.5319**	**0.5319**	0.5158	0.5158	0.5156	0.5158	0.5119	0.5000	0.5000	0.5158
35	0.9993	0.9857	0.9862	0.9805	**0.9896**	0.9857	0.9857	0.9868	0.9857	0.9797	0.9687	0.9550	0.9857
Avg.	0.9909	0.8343	0.8347	0.8303	0.8329	0.8356	0.8331	0.8357	0.8339	0.8488	0.8135	**0.8622**	0.8349
Avg. rank	n/a	6.6714	5.8857	7.0857	5.9429	5.8429	6.7286	6.0857	7.1429	**4.6714**	10.1429	5.4857	6.3143

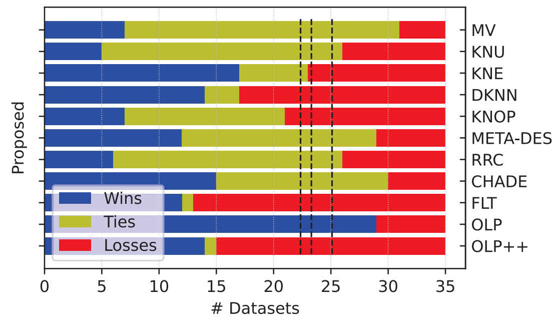
Using the AdaBoost ensemble, on the other hand, yielded more diverse results. Table 4.7 shows the average balanced accuracy rate obtained by the techniques over each dataset. For a complementary analysis, the results regarding the performance over the minority class alone can be found in the supplementary material. We can see that using the more diverse AdaBoost ensemble the overall performance of all techniques was larger compared to the bagged ensemble, though not for every problem. The Oracle's performance was also better, obtaining a perfect score in all but two datasets. Moreover, using the AdaBoost ensemble, the techniques based on the similarities in the decision space representation (GNN-DES, META-DES, and KNOP) obtained the three highest average ranks, with the proposed method in the first place. The

Table 4.7 Average balanced accuracy rate of the techniques over each dataset, using the AdaBoost ensemble.

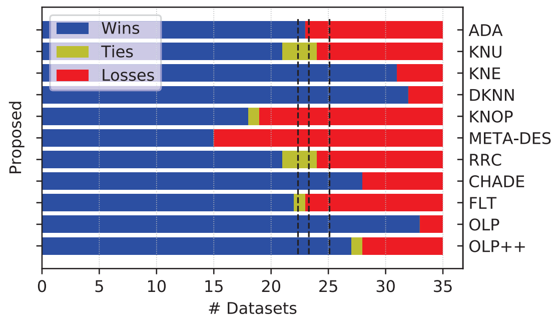
Ref. #	Oracle	ADA	KNU	KNE	DKNN	KNOP	META-DES	RRC	CHADE	FLT	OLP	OLP++	GNN-DES
1	1.0000	0.4833	0.5208	0.5667	0.5292	0.4958	0.4375	0.5208	0.4667	0.5167	**0.6833**	0.5542	0.5333
2	1.0000	0.9442	0.9342	0.9075	0.8950	0.9442	0.9442	0.9075	0.9217	0.9567	0.9300	**0.9667**	0.9342
3	1.0000	0.9394	0.9478	**0.9593**	0.9471	0.9522	0.9561	0.9522	0.9481	0.9051	0.9141	0.9510	0.9522
4	1.0000	0.9072	0.9131	0.8914	0.8962	0.8822	0.8926	**0.9173**	0.9010	0.8884	0.8609	0.9030	0.9030
5	1.0000	0.9213	0.9314	**0.9385**	0.9361	0.9046	0.9008	0.9162	0.9091	0.9266	0.8990	0.9022	0.9109
6	1.0000	0.9356	0.9397	0.9128	0.9244	0.9317	0.9359	0.9196	0.9196	**0.9446**	0.8644	0.9401	0.9401
7	1.0000	0.7557	0.7557	0.6734	0.6974	**0.7622**	0.7521	0.6799	0.7445	0.6836	0.6386	0.7392	0.7480
8	1.0000	0.9340	0.9258	0.8676	0.8971	0.9416	**0.9519**	0.9095	0.8942	0.8645	0.8049	0.8709	0.9374
9	1.0000	0.9707	0.9707	0.9520	0.9457	0.9645	0.9645	0.9727	0.9630	**0.9731**	0.9190	0.9591	0.9626
10	0.9983	0.9733	0.9859	0.9733	0.9733	0.9788	0.9805	0.9788	**0.9913**	0.9823	0.9525	0.9768	0.9859
11	1.0000	**0.9236**	0.9067	0.8860	0.8808	0.9147	0.9201	0.9129	0.9050	0.9102	0.8417	0.9110	0.9229
12	1.0000	0.9865	0.9824	0.9763	0.9782	0.9843	0.9843	0.9824	**0.9885**	0.9808	0.9301	0.9689	0.9824
13	1.0000	0.9540	0.9609	0.9397	0.9469	0.9609	**0.9623**	0.9594	0.9580	0.9135	0.8882	0.9081	0.9594
14	0.9917	0.9816	0.9774	0.9774	0.9774	0.9831	0.9759	0.9774	0.8393	**0.9845**	0.9702	0.9774	**0.9845**
15	1.0000	0.9342	0.9398	0.9404	0.9278	0.9490	0.9476	0.9398	**0.9509**	0.9245	0.8636	0.8962	0.9460
16	1.0000	0.9556	0.9417	0.9274	0.9230	0.9574	0.9574	0.9457	0.9406	0.9554	0.8815	0.9519	**0.9599**
17	1.0000	0.9763	0.9762	0.9683	0.9684	0.9761	0.9798	0.9744	0.9759	0.9777	0.9704	0.9740	**0.9816**
18	1.0000	0.9682	0.9740	0.9774	**0.9778**	0.9706	0.9711	0.9730	0.9629	0.9692	0.9576	0.9755	0.9725
19	1.0000	0.6271	0.7585	0.6371	0.5784	0.7137	0.7134	0.6965	**0.8088**	0.5000	0.5784	0.5315	0.7370
20	1.0000	0.7510	0.8288	0.7024	0.7411	0.8163	0.8096	**0.8980**	0.8376	0.5000	0.6823	0.6225	0.8128
21	1.0000	0.8252	0.8252	0.7370	0.7645	0.8405	0.8433	**0.8621**	0.8240	0.5000	0.7301	0.7831	0.8255
22	1.0000	0.9473	0.9617	0.9522	0.9551	0.9622	0.9607	0.9539	0.9070	0.5000	0.9200	0.9498	**0.9632**
23	1.0000	0.7667	0.6675	0.5792	0.7183	0.7458	0.7442	0.6908	0.6067	**0.7958**	0.5950	0.7767	0.7400
24	1.0000	0.9223	0.8961	0.9008	0.9175	0.9166	0.9371	0.9315	0.8290	**0.9455**	0.9120	0.9168	0.9271
25	1.0000	0.8078	0.7104	0.6792	0.7019	0.8179	0.8052	0.6458	0.7422	**0.8432**	0.5903	0.7377	0.8094
26	1.0000	0.8607	0.8039	0.7725	0.7647	0.8980	0.9000	0.7028	0.6892	0.9179	0.7198	**0.9369**	0.8969
27	1.0000	0.8649	0.8563	0.8441	0.8363	0.9032	0.9107	0.8140	0.7165	**0.9526**	0.8214	0.9441	0.9104
28	1.0000	0.8520	0.7591	0.6371	0.6868	0.8640	0.8639	0.6159	0.7733	**0.8771**	0.5695	0.7938	0.8609
29	1.0000	0.8649	**0.8964**	0.8535	0.8639	0.8759	0.8938	0.8921	0.8826	0.8743	0.8545	0.8664	0.8853
30	1.0000	**0.9648**	0.8563	0.8215	0.8178	0.9531	0.9560	0.7926	0.5776	0.8822	0.7566	0.7639	0.9300
31	1.0000	0.9861	0.9786	0.9753	0.9542	0.9856	**0.9878**	0.9519	0.8017	0.9819	0.9192	0.9697	0.9825
32	1.0000	0.9850	0.9867	0.9867	0.9894	0.9800	0.9850	**0.9933**	0.9761	0.9847	0.9836	0.9625	0.9900
33	1.0000	0.8856	0.7396	0.6492	0.6644	0.9015	**0.9077**	0.5822	0.7967	0.9020	0.6017	0.8421	0.8946
34	1.0000	0.5146	0.5000	0.4975	0.5000	0.4996	0.4998	0.5102	**0.5665**	0.5119	0.5000	0.5000	0.4996
35	1.0000	0.9801	0.9827	0.9664	0.9785	0.9797	**0.9834**	0.9814	0.8610	0.9797	0.9687	0.9550	0.9800
Avg.	0.9997	0.8814	0.8712	0.8408	0.8473	0.8888	0.8890	0.8530	0.8393	0.8488	0.8135	0.8622	**0.8903**
Avg. rank	n/a	5.3143	5.3571	8.3857	8.0143	4.9000	4.2857	6.3286	7.6857	5.9000	10.5857	7.1143	**4.1286**

AdaBoost combiner (ADA) also yielded the fourth highest rank followed by the KNU and the other techniques that perform the local region definition in the feature space, in addition to CHADE. The surpassing of the decision space-based techniques might be due to the improvement in the data representation with the increase in diversity, while the worse performance of the feature space-based compared to the static selection baseline suggests the high dimensional representation still poses a challenge to these local-based techniques.

The win-tie-loss summary of the results with respect to the proposed method, shown in Figure 4.15b, confirms the trend, with the proposed method only losing more often than winning against the META-DES. Comparing the techniques two at a time, we can see in Table 4.8b that the proposed technique yielded a statistically superior performance to all techniques except the KNOP, META-DES and FLT, being the only one to surpass the AdaBoost combiner with $\alpha = 0.05$.



a)



b)

Figure 4.15 Win-tie-loss distribution of the average balanced accuracy rate of the proposed GNN-DES technique vs. the DES techniques using the (a) Bagging ensemble, and (b) AdaBoost ensemble. The dashed lines indicate the critical values of the sign test considering a sample size of $n = 35$ for $\alpha = 0.1$, $\alpha = 0.05$ and $\alpha = 0.01$ from left to right, respectively.

Thus, so far we could observe a few points. First, the behavior of the DES techniques over HDSSS when using ensembles that present different degrees of diversity. The more diverse pool was able to boost the techniques' performances in most cases, especially for the ones based on similarities in the decision space. This suggests that using a diverse ensemble and the local information in the decision space can help the DES technique deal somewhat better with the high dimensionality, which answers our **RQ2**. However, and answering our **RQ1**, the techniques still struggled compared to the static baseline using either pool of classifiers, with most of them obtaining a statistically similar performance. This further indicates the challenge they face over HDSSS distributions and points to some limitations in their applicability.

W.r.t. our **RQ3**, we observed that the proposed method was the only one to significantly surpass the AdaBoost static combiner with $\alpha = 0.05$, suggesting that using another source of information,

Table 4.8 P-value of the pairwise Wilcoxon signed-rank test between the techniques, using the (a) Bagging ensemble, and (b) AdaBoost ensemble. Values below $\alpha = 0.05$ are in bold. The symbols $-$ and $+$ indicate whether the column-wise technique yielded a statistically inferior or superior performance to the row-wise technique.

(a)

	MV	KNU	KNE	DKNN	KNOP	META-DES	RRC	CHADE	FLT	OLP	OLP++	GNN-DES
MV	n/a	0.1152	0.3988	0.6406	0.1434	0.5215	0.0597	0.2379	0.1518	**0.0002**(-)	0.1714	0.3753
KNU		n/a	0.1817	0.7869	0.4959	0.1316	0.7580	**0.0296**(-)	0.1340	**0.0001**(-)	0.1472	0.3506
KNE			n/a	0.4811	0.0809	0.4173	0.1612	0.1791	0.1110	**0.0010**(-)	0.0964	0.2040
DKNN				n/a	0.8186	0.7370	0.6642	0.3943	0.1442	**0.0005**(-)	0.1427	0.9543
KNOP					n/a	0.0515	0.5469	0.0977	0.0900	$<$ **10^{-4}**(-)	0.1383	0.1058
META-DES						n/a	0.3383	0.5103	0.1140	**0.0003**(-)	0.1383	0.1981
RRC							n/a	0.2051	0.1518	**0.0001**(-)	0.1819	0.3836
CHADE								n/a	0.1472	**0.0001**(-)	0.1614	0.0664
FLT									n/a	**0.0051**(-)	0.4809	0.1565
OLP										n/a	**0.0003**(+)	**0.0001**(+)
OLP++											n/a	0.1819
GNN-DES												n/a

(b)

	ADA	KNU	KNE	DKNN	KNOP	META-DES	RRC	CHADE	FLT	OLP	OLP++	GNN-DES
ADA	n/a	0.3853	**0.0002**(-)	**0.0001**(-)	0.0964	0.0997	0.2193	**0.0048**(-)	0.5124	$<$ **10^{-4}**(-)	**0.0125**(-)	**0.0423**(+)
KNU		n/a	$<$ **10^{-4}**(-)	**0.0001**(-)	0.1140	0.0825	0.0574	**0.0092**(-)	0.8570	$<$ **10^{-4}**(-)	0.3720	**0.0332**(+)
KNE			n/a	0.2619	**0.0001**(+)	**0.0001**(+)	0.2415	0.6583	0.1495	**0.0001**(-)	0.1518	$<$ **10^{-4}**(+)
DKNN				n/a	**0.0001**(+)	**0.0001**(+)	0.4175	0.9478	0.3021	$<$ **10^{-4}**(-)	0.2102	$<$ **10^{-4}**(+)
KNOP					n/a	0.1177	0.0607	**0.0019**(-)	0.6465	$<$ **10^{-4}**(-)	**0.0097**(-)	0.6174
META-DES						n/a	**0.0168**(-)	**0.0020**(-)	0.2725	$<$ **10^{-4}**(-)	**0.0064**(-)	0.5888
RRC							n/a	0.1085	0.6702	$<$ **10^{-4}**(-)	0.8506	**0.0069** (+)
CHADE								n/a	0.3505	0.0948	0.2516	**0.0004**(+)
FLT									n/a	**0.0059**(-)	0.4711	0.1340
OLP										n/a	**0.0003**(+)	$<$ **10^{-4}**(+)
OLP++											n/a	**0.0015**(+)
GNN-DES												n/a

the interactions between the classifiers, in addition to the local information can be advantageous when performing the dynamic selection task over sparse data. However, it yielded a statistically similar performance to the two other techniques that carry out a dissimilarity search in the decision space, KNOP and META-DES.

Thus, in the following section, we investigate the local distribution with respect to the overlap in the problem and in the meta-problem considering both the feature and decision spaces that could explain the differences in the performance of the DES techniques. We also assess whether the proposed method was more able to overcome the locally difficult scenarios compared to the other techniques due to its joint learning of the classifiers' inter-dependencies and samples' interactions.

4.6.3 Further analysis

We now investigate the problems' local characteristics in the feature space and in the decision space which can help understand the behavior of some of the evaluated techniques. Since local methods can be strongly sensitive to class overlap and data sparsity (Sánchez *et al.*, 2007), and the latter is often associated with an increased class boundary complexity (Lorena *et al.*, 2012; Ho & Basu, 2002), we expect the DES techniques to behave differently depending on the local difficulty present in the problem (and the meta-problem) for each HDSSS dataset. This analysis is done over the results using the AdaBoost ensemble as it yielded a comparatively more diverse performance of the DES techniques.

Thus, we further analyze the experimental results using the following local measures: the K-disagreeing neighbors (KDN) (Smith *et al.*, 2014), the supervised labelset intersection (SLI), and the neighborhood labelset intersection (NLI). The first is defined as the proportion of samples in a given instance's neighborhood of size K that does not share its label, conveying the degree of local class overlap for that instance. The KDN measure is shown in Equation (4.12), where $KNN()$ is the nearest neighbors function and $\mathbb{1}()$ is the indicator function.

The second measure, called the supervised labelset intersection, we define as the average pairwise intersection between a given query sample's labelset and each of its neighbors' labelsets, normalized by the number of classifiers $|C|$. The measure is shown in Equation (4.13), where $u_{q,k}$ and $u_{i,k}$ are the meta-label representation of the query's and one of its neighbors' labelset w.r.t. the competence of classifier c_k. The SLI then describes how good of an indication the neighborhood provides of the query instance's competent classifiers in the meta-problem.

Lastly, the third measure, the neighborhood labelset intersection, we define as the average intersection between the labelsets of each pair of instances from a given neighborhood of size K, also normalized by the number of classifiers $|C|$. The neighborhood labelset intersenction is described in Equation (4.14), where $u_{i,k}$ and $u_{j,k}$ are the meta-label representation of two samples' labelsets w.r.t. the competence of classifier c_k, with both of them belonging to the neighborhood of the query instance. Thus, the NLI conveys the degree of consensus in the neighborhood w.r.t. the ability of the classifiers, disregarding the query instance's labelset. These measures are used to assess the difficulty of the ensemble selection task based on the surrounding context in the feature and decision spaces. In all cases, we extract these measures with $K = 7$, as it was used for several DES techniques in the previous section.

$$KDN(\mathbf{x}_q) = \frac{1}{K} \sum_{i} \mathbb{1}(y_q = y_i), \forall \mathbf{x}_i \in KNN(\mathbf{x}_q) \tag{4.12}$$

$$SLI(\mathbf{x}_q) = \frac{1}{K|C|} \sum_{i} \sum_{k=1}^{|C|} \mathbb{1}(u_{q,k} = 1 \wedge u_{i,k} = 1), \forall \mathbf{x}_i \in KNN(\mathbf{x}_q) \tag{4.13}$$

$$NLI(\mathbf{x}_q) = \frac{2}{K(K-1)|C|} \sum_{i} \sum_{j, i \neq j} \sum_{k=1}^{|C|} \mathbb{1}(u_{i,k} = 1 \wedge u_{j,k} = 1), \forall \mathbf{x}_i, \mathbf{x}_j \in KNN(\mathbf{x}_q) \tag{4.14}$$

4.6.3.1 Feature space *vs.* decision space

We can see in Figure 4.16a that the decision space neighborhood provides, on average, a higher supervised labelset intersection for the test instances, meaning the query samples and their corresponding neighborhoods share on average a larger number of competent classifiers compared to the neighborhood obtained in the feature space. This might help explain why the decision space-based DES techniques outperformed on average the feature space-based DES techniques over the HDSSS datasets, as in general the similarities in the former provided a better indication of the competence of the classifiers compared to the latter.

We now take a look at the impact the problems' class overlap has on both spaces. Figure 4.16b shows the proportion of test samples in overlap regions for each dataset in the feature space and the decision space. A sample is considered in an overlap region if its neighborhood contains more than one class, i.e., its KDN score is between zero and one ($0.0 < KDN < 1.0$). We can observe that the neighborhoods obtained in the feature space present much more class overlap compared to its counterpart in the decision space, on average. Moreover, Figure 4.16c shows the average supervised labelset intersection of the instances in overlap regions in both spaces. We can observe that, even though the decision space presented fewer overlap regions, they tend to yield a lower average supervised labelset intersection than in the feature space. This suggests the class overlap in the decision space, while less frequent, might be more hindering to the DES task compared to the overlap in the feature space.

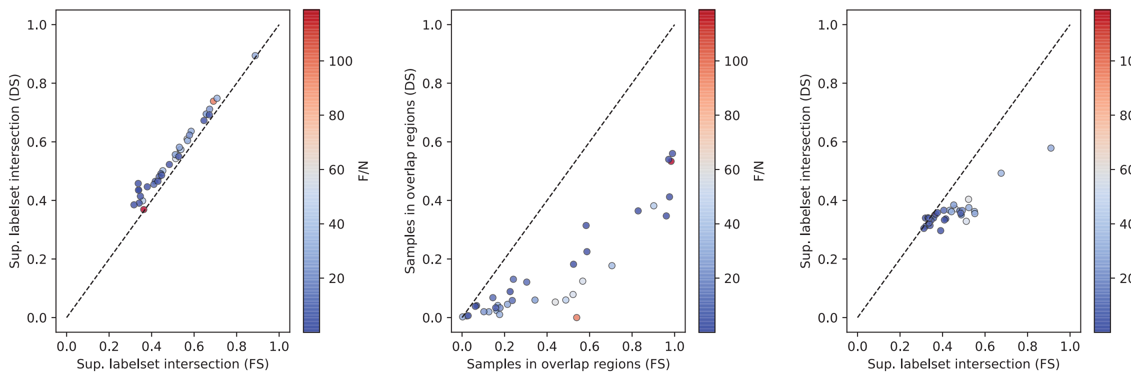


Figure 4.16 Overall comparison between the feature and decision spaces (FS and DS, respectively) in each dataset (circle). (a) shows the overall average supervised labelset intersection in both spaces. (b) shows the average proportion of test instances in overlap regions, and (c) the average supervised labelset intersection of the samples in overlap regions. The F/N indicates the ratio between the number of features and samples of each dataset.

4.6.3.2 Neighborhood labelset intersection *vs.* supervised labelset intersection

We further analyze the datasets' local characteristics w.r.t. the DES meta-problem to understand the aspects which can make the task more difficult. Figure 4.17a-b shows the neighborhood labelset intersection against the supervised labelset intersection in both spaces, averaged over all samples from each dataset. Observing the two measures together may indicate whether

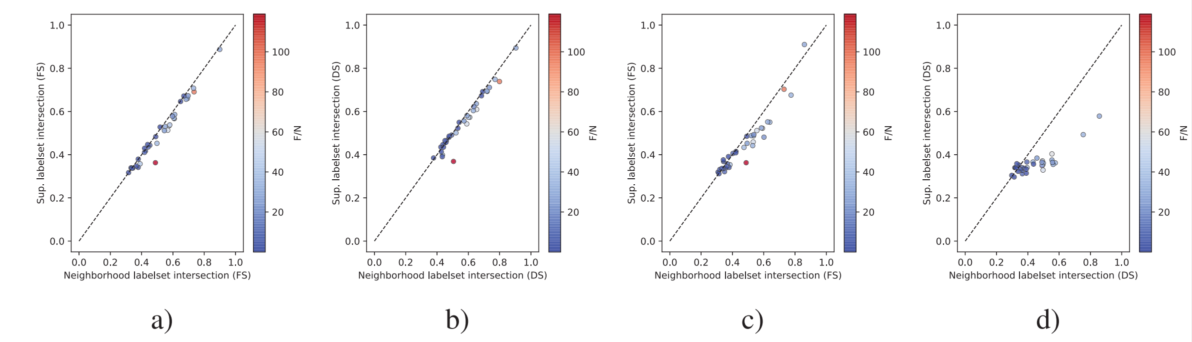


Figure 4.17 Neighborhood labelset intersection (Eq. (4.14)) *vs.* the supervised labelset intersection (Eq. (4.13)) in the (a,c) feature space (FS) and (b,d) decision space (DS), averaged over (a-b) all instances and (c-d) samples in overlap regions, for each dataset. The F/N indicates the ratio between the number of features and samples of each dataset.

the local regions present an excess labelset that does not belong to their corresponding query samples' labelset. That is, we can see how much the neighborhood consensus exceeds the useful information it provides in the search for competent classifiers. Similarly to a single-label classification case, in which the excess support for an opposite class in the neighborhood may lead to misclassification based on a local rule, in the case of the local-based DES techniques the excess support for other meta-labels in the neighborhood may lead to the selection of incompetent classifiers for the query. We thus define the excess neighborhood labelset (ENL) (Equation (4.15)), i.e. the difference between the average neighborhood labelset intersection and the average supervised labelset intersection, as a measure that conveys how much the neighborhood consensus is helpful for choosing the correct classifiers for a given query instance. It can be observed that, in both spaces, the neighborhood labelset intersection is quite close to the supervised labelset intersection, except for a few datasets, showing that the excess neighborhood labelset is not a great issue considering the collective of all instances at once for most evaluated problems. This suggests that in most cases the neighborhood consensus provides a good indication of which classifiers to select for a given query.

$$ENL = NLI - SLI \tag{4.15}$$

However, considering only the samples in overlap regions, we can see from Figure 4.17c-d that for a large amount of the datasets, the neighborhood labelset intersection exceeds on average the supervised labelset intersection, in both spaces. This suggests that in ambiguous regions the excess neighborhood labelset problem is more prevalent, which in turn may make the DES task more difficult for the samples located in those regions as it would incentivize the selection of incompetent classifiers. We can also observe from Figure 4.17c-d that in the decision space the excess neighborhood labelset in overlap regions is generally greater compared to the feature space for most datasets. Thus, it seems that in the decision space regions, the class overlap may lead to not only a small supervised labelset intersection but also a comparatively large neighborhood labelset intersection, which can hinder the performance of the DES techniques that rely on similarities in the decision space for RoC definition.

4.6.3.3 Comparison against decision space-based techniques

With the characterization of the local data in both spaces done, and the issues associated with the class overlap and meta-problem shown, we now compare the performance of the proposed method to its similar counterparts: the DES techniques that rely on the classifiers' responses as input to the system, namely the KNOP, META-DES, and CHADE. We focus this analysis on the overlap instances in the decision space, as they present a difficult scenario to perform the DES task. It is worth noting that the META-DES and KNOP yielded a statistically similar performance to the proposed method over the whole testbed (Section 4.6.2), and that only CHADE does not apply any dissimilarity metric within the decision space, though the proposed method does not define a fixed RoC as META-DES and KNOP does. Moreover, the META-DES is the only one among them that defines the RoC in both the feature and decision spaces.

Figure 4.18 shows the average excess neighborhood labelset in overlap regions in the feature and decision spaces for each dataset. The marker colors indicate the win-tie-loss result for each

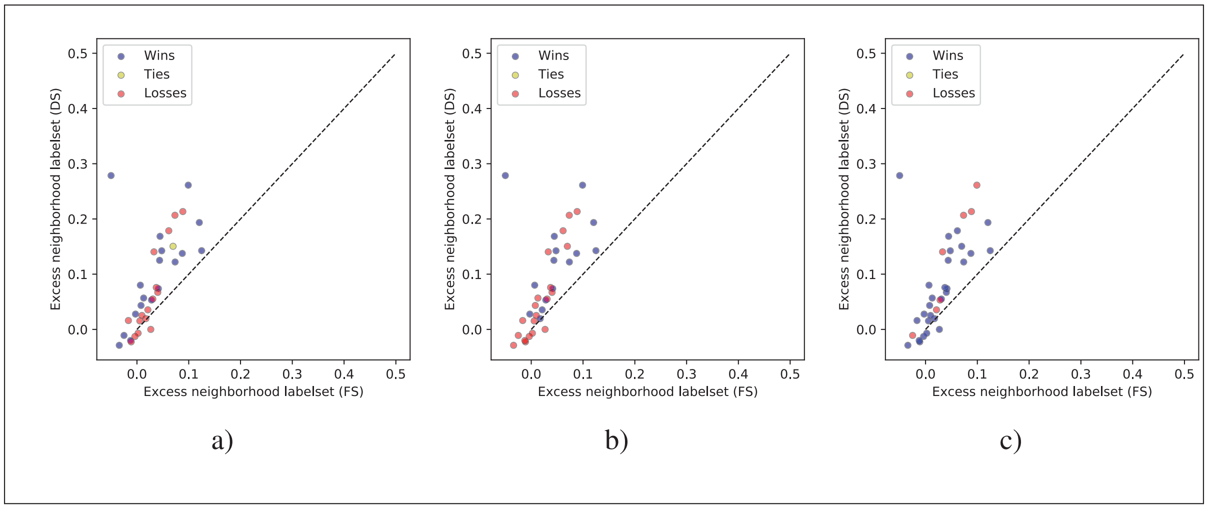


Figure 4.18 Average excess neighborhood labelset (Eq. (4.15)) in overlap regions in the feature and decision spaces for each dataset. The win-tie-loss summary is done w.r.t. the proposed GNN-DES technique and the (a) KNOP, (b) META-DES and (c) CHADE.

dataset considering the proposed method against the KNOP, META-DES and CHADE. It can be observed that the proposed method outperformed more often the first two techniques over the datasets with higher excess neighborhood labelset in overlap regions in the decision space. This suggests that the proposed method might be more competent in dealing with the problems in which the local overlap poses a greater challenge to the DES task than the other two local-based techniques.

Compared to the CHADE, we see that the latter outperformed the proposed method only over a few datasets, most of them among the ones with high excess neighborhood labelset. As opposed to the other three techniques, which take into account similarities in the feature and/or decision space (a generally useful approach for the DES task, as seen in Figure 4.16a), CHADE learns the inter-dependencies between the meta-labels to recommend an ensemble. Thus, in the cases where the former fails, CHADE may still find a competent ensemble for a given query. However, CHADE depends heavily on the order of the classifier chain to learn properly the inter-dependencies, which may be one of the reasons it was not so competitive compared to the other three techniques based on the decision space.

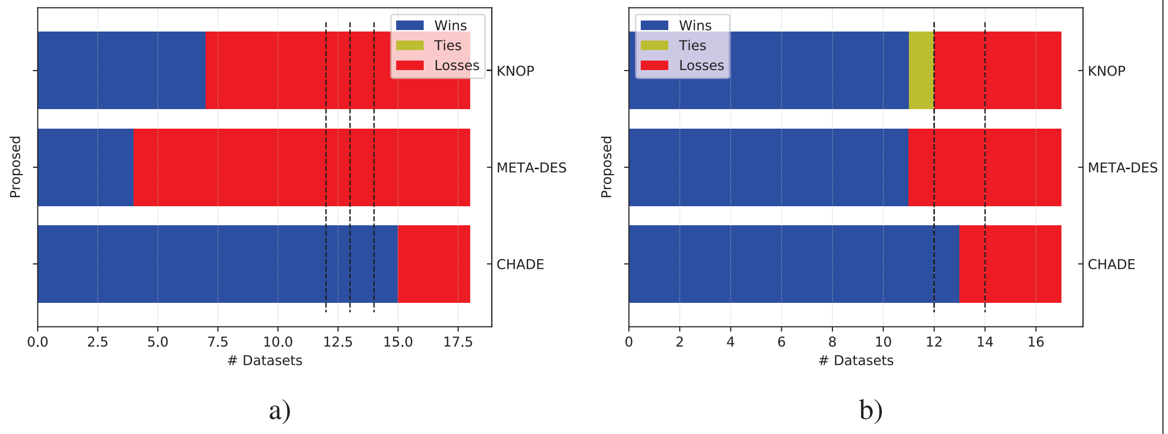


Figure 4.19 Win-tie-loss distribution of the average balanced accuracy rate of the proposed GNN-DES technique vs. the three DES techniques using the AdaBoost ensemble considering the (a) bottom and (b) top half of the datasets ordered by the average excess neighborhood labelset in overlap regions. The dashed lines indicate the critical values of the sign test considering a sample size of (a) $n = 18$ for $\alpha = 0.1$, $\alpha = 0.05$ and $\alpha = 0.01$, and (b) $n = 17$ for $\alpha = \{0.1, 0.05\}$ and $\alpha = 0.01$, from left to right.

Table 4.9 P-value of the pairwise Wilcoxon signed-rank test between the proposed method and the three decision space-based techniques over the datasets with the lowest excess neighborhood labelset and the highest excess neighborhood labelset. Values below $\alpha = 0.05$ are in bold. The symbols - and + indicate whether the proposed technique yielded a statistically inferior or superior performance to the other technique, respectively.

	Low excess neighb. labelset	High excess neighb. labelset
GNN-DES *vs.* KNOP	0.2837	**0.0331**(+)
GNN-DES *vs.* META-DES	**0.0268**(-)	0.0714
GNN-DES *vs.* CHADE	**0.0047**(+)	**0.0093**(+)

Splitting the datasets in half by the average excess neighborhood labelset in overlap regions and obtaining the win-tie-loss distribution for each part in Figure 4.19, we can observe that, over the datasets with higher ENL (Figure 4.19b), the proposed method obtains more wins than losses compared to the three techniques. The Wilcoxon signed-rank test (Table 4.9) shows that over these datasets the proposed technique significantly outperforms the KNOP and CHADE. For the datasets on the bottom half in terms of excess neighborhood labelset, in Figure 4.19a, we can see that the proposed technique loses more than wins against the KNOP and META-DES, performing statistically worse than the latter, as shown in Table 4.9.

Thus, the results suggest that the proposed method can better deal with the datasets whose overlap regions yield a larger excess of incompetent classifiers than the other local-based techniques, answering our **RQ4**. This might be due to its use of not only local information but also the classifiers' relationship information, as the latter might be useful for learning to select the classifiers when the former is not reliable enough.

4.6.4 Lessons learned

We summarize our findings from the experimental analysis in the following comments.

- In the first set of experiments (Sections 4.6.2 and 4.6.1) we evaluated the DES techniques over several HDSSS datasets using two different ensemble methods. The ensembles were characterized and also used to obtain the performance of a static baseline. We observed that, though the DES techniques were shown in the literature to work well over ill-defined problems (Britto *et al.*, 2014), most of them performed similarly to the static baseline over the HDSSS problems when given a highly correlated ensemble to select from. Thus, we recommend analyzing the diversity of the ensemble before choosing to apply these techniques over such problems;
- Also from the first set of experiments, we observed that when using a more diverse ensemble the three techniques based on similarities in the decision space obtained an overall better performance compared to the techniques based solely on the similarities in the feature space and to the static baseline. Among them, the proposed method was the only DES technique to significantly surpass the latter over the HDSSS testbed. Thus, when choosing a DES technique to deal with an HDSSS problem, we suggest evaluating this family of techniques first and picking the best from them, as they performed similarly overall, instead of testing all available techniques as most of them belong to the family of feature space-based techniques and those were shown to perform generally worse over the sparse data;

- In the second set of experiments (Section 4.6.3) we characterized the local regions in the feature and decision spaces to assess how the class overlap and data sparsity affected the meta-problem (i.e. the DES task) on a local level, and how the decision space-based techniques performed over the problems with the least and the most unfavorable local contexts. We observed that the proposed method outperformed the local-based techniques over the problems where the overlap areas were the most misleading, as in suggesting the selection of several incompetent classifiers. We can thus conclude that leveraging both the instances' and classifiers' interactions in an end-to-end manner aided the proposed method in overcoming the limitations of the classical techniques over the scenarios where the locality assumption for the dynamic selection meta-problem is weaker, at the cost of not performing as well as the other two techniques when the locality assumption is stronger. Thus, we recommend analyzing the local characteristics to assess whether applying the proposed approach instead of the other decision space-based techniques could be advantageous based on how unfavorable to the DES task the local overlap regions are in the HDSSS problem;
- Lastly, and considering the results of the two sets of experiments, we can conclude that, as other local methods are prone to do (Sánchez *et al.*, 2007), the DES techniques generally struggled over the evaluated high dimensional overlapped data. However, some approaches stood out as positive to include in the dynamic selection pipeline when dealing with such problems: using a diverse ensemble, leveraging the local information from the decision space, and in some cases, the inter-dependencies between the classifiers. Thus, based on the experimental results obtained over the HDSSS datasets, we believe these are interesting approaches to be considered going forward in the research area of the DES techniques to further improve their performance over high dimensional overlapped data.

4.7 Conclusion

We proposed in this work a dynamic selection technique that learns from the instance-instance and classifier-classifier relationships to better deal with high dimensionality and class overlap, which may hinder the local region definition for the DES task. The proposed technique uses a multi-label GNN as a meta-learner, exploiting not only the local data structure in the form of a graph, without directly defining the RoC, but also the classifiers' inter-dependencies modeled in the meta-labels to learn the dynamic classifier combination rule. Thus, the GNN may learn internally an embedded space where the DES task is potentially easier, especially for the samples that present a poor local distribution, so that it can still yield a good combination rule when the locality assumption fails.

Experimental results over 35 high dimensional datasets with 10 DES techniques, in addition to the proposed method, and a static selection baseline showed the former struggled to surpass the latter when using a highly correlated ensemble. Using a more diverse ensemble, however, the proposed method significantly outperformed the static baseline and most of the evaluated DES techniques. Further analyzing the results showed that the feature space presented more class overlap and less intersection of competent classifiers locally than the decision space, which could justify the overall superior performance of the techniques based on similarities in the latter against the former. Nevertheless, the local class overlap present in the decision space was shown to provide more challenge to the DES task compared to the feature space, with the regions presenting a higher excess of incompetent classifiers. The proposed method was able to outperform the decision space-based techniques over the datasets the most afflicted by such scenarios.

As future work, we may perform a sensitivity analysis on the proposed approach to investigate the influence of hyperparameter variation and different linkage strategies in the graph design. Moreover, we may further explore the impact that diversity has on the proposed method using ensembles that directly optimize it, such as the Local Independence Training ensemble (Ross, Pan, Celi & Doshi-Velez, 2020), as well as the effects of class imbalance on the performance of the technique. We may also investigate the application of the proposed method in the context of multi-view learning (Gupta, Khan, Singh, Tanveer, Kumar, Chakraborti & Pachori, 2020; Du, Wang, Hu, Li & Chen, 2022), since the different views may provide enough ensemble diversity, and the proposed technique presents the possible advantage of using the graph representation, which allows a unified description of the samples' relationships in all views.

4.8 Supplementary material

4.8.1 Including validation data into the known graph $G_{\mathcal{T}}$

Though not an essential part of the GNN-DES technique, validation data was used in the experimental analysis of this work during the training of the GNN meta-learner, as stated in the experimental protocol. This was done so that the training procedure could be stopped early to avoid overfitting, as well as to be able to choose the version of the GNN with the highest micro-averaged multi-label precision over an unlabeled set of nodes.

Thus, in this section, we provide the pseudocode (Algorithm 4.5) which describes in more detail the procedure for including the validation set $\mathcal{V}$ into the known graph $G_{\mathcal{T}}$. With the validation data included in the known graph $G_{\mathcal{T}}$, the validation nodes can be used to compute the validation loss of the network during the fitting of the meta-learner, which allows a training procedure based on early stopping. Moreover, in this work, we also computed the average multi-label precision over the validation nodes at the end of each epoch and kept the parameters of the version which yielded the highest score to be used as our meta-learner.

$$\begin{aligned} e_{i,j} &= \begin{cases} 1 - d_{i,j}, & if\, d_{i,j} \leq d_i^{max}, \\ 0, & otherwise, \end{cases} \\ d_i^{max} &= \min(d_{i,k}, \forall \mathbf{x}_k \in \mathcal{T}) + \tau \end{aligned} \tag{4.16}$$

4.8.2 Additional results

Table 4.10 presents the average performance of the evaluated techniques using the AdaBoost ensemble over each dataset in terms of the F1 score, defined for binary problems as the harmonic mean between the precision and the recall over the minority class (**?**). The win-tie-loss summary for the GNN-DES with respect to the other techniques in terms of average F1 score is shown on

Algorithm 4.5 Procedure for including validation data into the known graph

```
   input  : $G_\mathcal{T}, C, \tau$ ;                       ▷ Known graph, pool of classifiers, preset threshold
   input  : $\mathcal{T}, \mathcal{V}$ ;                     ▷ Training and validation sets
   output : $G_\mathcal{T}$ ;                                ▷ Known graph with validation data
1  $V \leftarrow \{\}$ ;                                      ▷ Validation vertexes set
2  $E \leftarrow \{\}$ ;                                      ▷ Validation edges set
3  $P_\mathcal{V} \leftarrow to\_decision\_space(\mathcal{V}, C)$ ▷ Project $\mathcal{V}$ to the decision space
4  $D_{\mathcal{V}\times\mathcal{T}} \leftarrow pairwise\_distance(P_\mathcal{V}, P_\mathcal{T})$ ;   ▷ Obtain pairwise distance matrix
5  for every $\mathbf{x}_i$ in $\mathcal{V}$ do
6      $V \leftarrow V \cup v_i(\mathbf{x}_i)$ ;              ▷ Add $v_i$ with its feature vector $\mathbf{x}_i$ to $V$
7      for every $\mathbf{x}_j$ in $\mathcal{T}$ do
8          Calculate $e_{i,j}$ from $D_{\mathcal{V}\times\mathcal{T}}, \tau$ according to Eq. (4.16) if $e_{i,j} > 0$ then
9              $E \leftarrow E \cup e_{i,j}$ ;                ▷ Add edge to graph
10         end if
11     end for
12 end for
13 $G_\mathcal{T} \leftarrow G_\mathcal{T} \cup (V, E)$ ;      ▷ Add validation vertexes and edges to $G_\mathcal{T}$
14 return $G_\mathcal{T}$
```

the *Win-tie-loss* row. The row *P-value* shows the resulting p-value of the Wilcoxon signed-rank test between the GNN-DES and each column-wise technique.

It can be observed that, similarly to the performance over both classes (in terms of balanced accuracy rate), the proposed technique also achieves the highest average rank in terms of F1 score among the evaluated techniques, suggesting it yields a generally good performance over the minority class as well. It also statistically outperforms CHADE and the feature space-based only dynamic selection techniques according to the pairwise Wilcoxon signed-rank tests. This demonstrates the GNN-DES's effectiveness in the presence of class imbalance in sparse and overlapped data compared to several techniques that were often shown to work fairly well over low-dimensional class imbalanced problems, such as KNU, RRC, DKNN and OLP (Oliveira *et al.*, 2017; Souza *et al.*, 2019a, 2022). However, the proposed technique yields a statistically similar performance ($\alpha = 0.05$) to the static baseline (ADA), the decision space-based techniques (KNOP and META-DES), and the tree-based dynamic ensembles (FLT and OLP++) in terms of F1, indicating a competitive overall performance over the minority class samples in high dimensional overlapped data compared to these techniques. The overall results thus suggest a similar trend to the results reported in the main manuscript in terms of balanced accuracy rate.

Table 4.10 Average F1 score of the techniques over each dataset, using the AdaBoost ensemble. Best results are in bold. The row *Win-tie-loss* indicates the amount of datasets over which the GNN-DES obtained a higher, equal, and lower, respectively, average F1 score than the column-wise technique. The row *P-value* indicates the resulting p-value of the Wilcoxon signed-rank test between the GNN-DES and the column-wise techniques, and values below $\alpha = 0.05$ are in bold.

Ref. #	ORACLE	ADA	KNU	KNE	DKNN	KNOP	META-DES	RRC	CHADE	FLT	OLP	OLP++	GNN-DES
1	1.0000	0.3333	0.3900	0.3633	0.3800	0.3000	0.2467	0.4233	0.2333	0.2067	**0.5138**	0.2500	0.3571
2	1.0000	0.9257	0.9057	0.8557	0.8414	0.9257	0.9257	0.8557	0.8850	0.9400	0.9067	**0.9600**	0.9057
3	1.0000	0.9548	0.9584	**0.9723**	0.9639	0.9577	0.9621	0.9577	0.9534	0.9456	0.9438	0.9682	0.9577
4	1.0000	0.9191	0.9292	0.9128	0.9174	0.9048	0.9136	**0.9339**	0.9152	0.9202	0.8843	0.9269	0.9217
5	1.0000	0.8990	0.9096	**0.9163**	0.9156	0.8799	0.8742	0.8904	0.8822	0.9087	0.8699	0.8789	0.8865
6	1.0000	0.9382	0.9429	0.9153	0.9273	0.9342	0.9385	0.9240	0.9217	**0.9477**	0.8635	0.9391	0.9404
7	1.0000	0.8697	0.8568	0.7717	0.8568	0.8714	0.8699	0.8571	0.8123	0.8754	0.8105	**0.8821**	0.8648
8	1.0000	0.9200	0.9002	0.8319	0.8707	0.9262	**0.9398**	0.8795	0.8653	0.8311	0.7581	0.8402	0.9216
9	1.0000	0.9495	0.9480	0.9271	0.9205	0.9428	0.9428	**0.9547**	0.9253	0.9451	0.8897	0.9341	0.9361
10	0.9933	0.9569	0.9665	0.9569	0.9578	0.9588	0.9646	0.9588	0.9683	**0.9713**	0.9392	0.9703	0.9665
11	1.0000	**0.8756**	0.8484	0.8290	0.8139	0.8599	0.8646	0.8568	0.8421	0.8611	0.7545	0.8628	0.8615
12	1.0000	0.9732	0.9723	0.9639	0.9679	0.9726	0.9726	0.9723	**0.9769**	0.9658	0.9121	0.9560	0.9689
13	1.0000	0.9330	0.9344	0.9152	0.9241	0.9344	**0.9408**	0.9268	0.9191	0.8825	0.8509	0.8841	0.9268
14	0.9909	0.9707	0.9755	0.9755	0.9755	0.9766	0.9689	0.9755	0.7457	**0.9832**	0.9678	0.9755	**0.9832**
15	1.0000	0.9014	0.9032	0.9088	0.8953	**0.9209**	0.9158	0.9032	0.9092	0.8987	0.7850	0.8655	0.9099
16	1.0000	0.9480	0.9334	0.9165	0.9115	0.9500	0.9500	0.9368	0.9309	0.9458	0.8622	0.9423	**0.9527**
17	1.0000	0.9749	0.9748	0.9668	0.9667	0.9748	0.9787	0.9729	0.9748	0.9765	0.9690	0.9728	**0.9805**
18	1.0000	0.9635	0.9709	0.9747	**0.9749**	0.9672	0.9672	0.9691	0.9587	0.9666	0.9517	0.9728	0.9691
19	1.0000	0.9800	0.9763	0.9667	0.9794	0.9785	0.9781	0.9785	0.8987	0.9799	0.9711	**0.9805**	0.9777
20	1.0000	0.5848	0.6183	0.4455	0.5437	**0.6424**	0.6340	0.6301	0.4764	0.0000	0.4219	0.3790	0.6349
21	1.0000	0.7290	0.7085	0.5718	0.6296	0.7600	**0.7646**	0.6883	0.6396	0.0000	0.5515	0.6902	0.7317
22	1.0000	0.9784	0.9803	0.9792	0.9800	**0.9829**	0.9825	0.9809	0.9157	0.8747	0.9655	0.9810	0.9829
23	1.0000	0.7832	0.5993	0.5917	0.7156	0.7072	0.7260	0.6964	0.4183	**0.8033**	0.6210	0.7945	0.7232
24	1.0000	0.9116	0.8809	0.8856	0.9062	0.9044	0.9275	0.9193	0.7864	**0.9342**	0.9006	0.9016	0.9175
25	1.0000	0.7827	0.6410	0.6115	0.6512	0.7965	0.7820	0.6125	0.6793	**0.8222**	0.4750	0.6796	0.7831
26	1.0000	0.8585	0.7877	0.7520	0.7251	0.8968	0.8989	0.7560	0.7144	0.9195	0.7074	**0.9359**	0.8942
27	1.0000	0.8618	0.8509	0.8350	0.8237	0.9024	0.9094	0.8302	0.7400	**0.9522**	0.8181	0.9432	0.9077
28	1.0000	0.8317	0.7246	0.5842	0.6180	0.8479	0.8473	0.6022	0.7556	**0.8603**	0.4480	0.7560	0.8425
29	1.0000	0.7893	0.7972	0.7595	0.7831	0.7645	0.7739	0.7374	0.6836	0.8067	0.7629	**0.8091**	0.7822
30	1.0000	**0.9581**	0.6507	0.6791	0.7002	0.9086	0.9129	0.5930	0.3410	0.8632	0.6188	0.6881	0.8797
31	1.0000	0.9753	0.8980	0.9389	0.9270	0.9521	0.9525	0.8760	0.3668	**0.9769**	0.8865	0.9663	0.9630
32	1.0000	0.9846	0.9797	0.9799	0.9847	0.9795	0.9845	0.9804	0.9639	0.9820	0.9723	0.9600	**0.9897**
33	1.0000	0.8707	0.6933	0.5955	0.5965	0.8893	**0.8959**	0.5805	0.7700	0.8902	0.5198	0.8186	0.8814
34	1.0000	0.0400	0.0000	0.0000	0.0000	0.0000	0.0000	0.0400	**0.0459**	0.0400	0.0000	0.0000	0.0000
35	1.0000	0.9538	0.9387	0.9305	0.9325	0.9489	0.9494	0.8950	0.5844	**0.9569**	0.9335	0.9384	0.9525
Avg.	0.9995	**0.8594**	0.8270	0.7994	0.8137	0.8577	0.8587	0.8156	0.7543	0.8181	0.7716	0.8344	0.8587
Avg. rank	n/a	4.8714	6.2571	8.6143	7.4714	5.0286	4.4286	6.8714	8.9429	4.8714	10.5286	5.7000	**4.4143**
Win-tie-loss	n/a	21-0-14	22-3-10	30-1-4	28-1-6	17-2-16	16-1-18	24-2-9	32-0-3	16-1-18	32-1-2	21-1-13	n/a
P-value	n/a	0.6942	**0.0044**	**$< 10^{-4}$**	**0.0001**	0.8186	0.3298	**0.0011**	**$< 10^{-4}$**	0.6882	**$< 10^{-4}$**	0.0564	n/a

We now analyze the results by Imbalance Ratio (IR) range according to the categorization used in (Fernández, García, del Jesus & Herrera, 2008), where a dataset has low imbalance if its IR is below 3, medium imbalance if its IR is between 3 and 9 (both inclusive), and high imbalance if its IR is above 9. Because of the number of datasets in the high imbalance category was too low (only 4), we grouped the medium and high datasets into one group (medium-high) in order to obtain more robust statistical results. Table 4.11 shows the summary of the comparative results of the proposed technique against the evaluated methods in terms of average F1 score. It can be observed that the proposed technique was among the top 3 techniques with the highest average rank over the two groups of dataset, obtaining the highest rank over the datasets with medium-high imbalance. This suggests the proposed method is an overall competitive approach for dealing with the minority class in problems of varying imbalance levels.

Compared to the feature space-based only techniques (KNU, KNE, DKNN, RRC, OLP) and CHADE, the proposed method statistically outperformed them in both groups of low and medium-high class imbalance, further indicating its comparative effectiveness over the minority class in high dimensional overlapped data. Compared to the remaining techniques, the GNN-DES obtained a statistically similar performance over both groups of datasets.

All in all, the results in terms of F1 follow a similar trend to the one observed in the experimental analysis presented in terms of balanced accuracy rate in this work, with the GNN-DES obtaining an overall improved performance compared to the feature space-based methods and CHADE, and an overall similar performance to the KNOP, META-DES and FLT, differing only against the static baseline and the OLP++. We also observed that the degree of the problems' imbalance did not affect this trend, further supporting the experimental analysis presented in this work.

Table 4.11 Summary of the comparative analysis over the average F1 score of the proposed method against the evaluated techniques for each group of datasets using the AdaBoost ensemble. *Low* is the group of datasets that have $IR < 3$, and *Medium-high* is the group of datasets that datasets have $IR \geq 3$. Best results are in bold. The row *Win-tie-loss* indicates the amount of datasets over which the GNN-DES obtained a higher, equal, and lower, respectively, average F1 score than the column-wise technique. The row *P-value* indicates the resulting p-value of the Wilcoxon signed-rank test between the GNN-DES and the column-wise techniques, and values below $\alpha = 0.05$ are in bold.

Dataset group		ADA	KNU	KNE	DKNN	KNOP	META-DES	RRC	CHADE	FLT	OLP	OLP++	GNN-DES
	Avg. rank	5.2631	6.3158	8.6053	7.4737	5.3158	4.4211	7.0000	8.8421	**4.3684**	10.5789	5.2105	4.6053
Low	Win-tie-loss	14-0-5	11-1-7	15-0-4	15-0-4	10-1-8	8-0-11	12-1-6	18-0-1	8-0-11	17-0-2	11-0-8	n/a
	P-value	0.1956	**0.0421**	**0.0012**	**0.0053**	0.5328	0.2579	**0.0186**	$< \mathbf{10^{-4}}$	0.1819	**0.0005**	0.3955	n/a
	Avg. rank	4.40625	6.1875	8.625	7.46875	4.6875	4.4375	6.71875	9.0625	5.46875	10.46875	6.28125	**4.1875**
Medium-high	Win-tie-loss	7-0-9	11-2-3	15-1-0	13-1-1	7-1-8	8-1-7	12-1-3	14-0-2	8-1-7	15-1-0	10-1-5	n/a
	P-value	0.5282	**0.0411**	**0.0005**	**0.0012**	0.7960	0.8564	**0.0214**	**0.0010**	0.3936	**0.0005**	0.0525	n/a

CONCLUSION AND RECOMMENDATIONS

In this thesis, different ensemble frameworks were proposed to improve the dynamic selection of classifiers by integrating local context information into the system. The importance of the local distribution to the dynamic selection task was discussed, and key challenges directly associated with this aspect were identified, namely the definition of an adequate local region and the presence of local experts in the pool. A literature review on dynamic selection techniques was presented in Chapter 1, focusing on the different RoC definition methods and the application of local data characterization into the selection process, and the limitations of the current approaches with regard to the key challenges were established. Different techniques that incorporate local data information into the ensemble pipeline were then proposed in this thesis to address the identified challenges. On the presence of local experts in the pool, producing the pool over the local border was proposed to yield more specialized classifiers. Learning the dynamic selection task in an end-to-end manner from the classifiers' interactions and the local data relations was also proposed to boost the search for local experts. On the definition of an adequate local region, three approaches using different definition methods were proposed to tackle scenarios that affect the selection task, including local class overlap and data sparsity. In all of them, the local data is characterized and the region is defined with multiple scales to provide ample context to the system.

In Chapter 2, an instance hardness analysis was performed to investigate the impact the local class overlap has on the performance of several dynamic selection schemes, and based on that analysis, the Online Local Pool (OLP) technique was proposed. The OLP generates linear classifiers on the fly to produce locally specialized models able to recognize the local border if the latter is detected in the target region. This is achieved by first characterizing the samples using an instance hardness measure to identify the presence of local class overlap in different areas of the feature space. Then, in generalization, the local regions over which the local classifiers are generated are defined using the nearest neighbor rule with an increasing neighborhood size to incorporate the local context to the ensemble with different locality degrees. Furthermore, the linear models are produced using the SGH technique, which yields an Oracle accuracy rate of 100% over the input data, in order to fully "cover" the area surrounding the query instance. Experiments showed the local pool provided an improvement to several dynamic selection schemes compared to using a globally generated pool, suggesting that using locally specialized classifiers instead of a globally generated pool can be quite advantageous over class-overlapped regions. The OLP also yielded a statistically similar performance to several state-of-the-art methods, including ensemble methods and monolithic classifiers, further suggesting the competitiveness of the proposed approach.

In Chapter 3, another local ensemble method was proposed based on the OLP, called the OLP++, to address the limitations the previous technique presented over high dimensional data. Instead of relying on the nearest neighbors rule, which may be susceptible to the effects of the curse of dimensionality, the proposed approach leverages the data partitions obtained from tree-based algorithms for the locality definition. The OLP++ then produces the local experts over the different impure nodes from the decision path that a given query sample traverses in the tree(s), therefore introducing an increasingly wider local context to the ensemble. Experiments showed that the OLP++'s recursive partition-based region definition identified borderline instances with more success than the OLP's nearest neighbors-based region definition, suggesting an improvement in the local distribution over which the local linear rules are learned. The OLP++ also yielded a more diverse set of local classifiers and obtained a statistically superior performance compared to the OLP with the same pool size. Moreover, the OLP++ significantly outperformed the random forest baseline in most cases as well as several dynamic selection techniques, using the random forest ensemble, and obtained a similar performance to the other local-based ensemble methods, further suggesting its applicability for local ensemble learning in high dimensional spaces.

Lastly, in Chapter 4, the issue of local learning in sparse data is revisited, as the OLP++ presents a considerable shortcoming: the quality of the local decision rules depends directly on the pre-defined partitions, which may not provide regions where the locality assumption is strong for the dynamic selection task, especially in the presence of local class overlap. Thus, the Graph Neural Network Dynamic Ensemble Selection (GNN-DES) technique is proposed to address local learning for dynamic selection in sparse and overlapped distributions. In the GNN-DES, a multi-label GNN is fit in an end-to-end manner to select the local experts from an existing pool using information from the data's local relationships, characterized in a graph structure, and the classifiers' interactions, encoded in the meta-labels. By optimizing the network directly to the dynamic selection task using these two distinct sources of information, the meta-classifier may implicitly learn an embedded space where the locality assumption is stronger compared to the original feature space without requiring an explicit RoC definition. Experiments showed the dynamic selection techniques generally struggled over the sparse and overlapped data, though the GNN-DES and the other techniques that rely on the decision space representation fared better. However, the GNN-DES was the only one to statistically surpass the static selection baseline, which suggests that it might behave somewhat differently from the other local-based schemes possibly due to the classifiers' inter-dependencies learned by the multi-label meta-classifier. It also statistically outperformed several other dynamic schemes, including the OLP++, further suggesting its suitability to deal with sparse overlapped data. Further analysis of the local region distribution showed that the GNN-DES performed better than the contending techniques over the problems where the locality assumption was weaker in the presence of class overlap, suggesting that leveraging the local data distribution and the classifiers' interactions can aid the dynamic selection task in challenging scenarios.

Overall, the contributions of this thesis have focused on improving the performance of dynamic selection approaches by leveraging the local data distribution. This was achieved through data characterization and improvements in the local region definition and the search for local experts.

These advancements may further stimulate the application of dynamic selection schemes to real-world problems that present complex characteristics such as class overlap and data sparsity. Thus, from this thesis, several future directions may be considered:

- Increasing the characterization of the instances' local relationships to improve the graph representation for the GNN-DES may be an interesting research direction. The samples' relations in the GNN-DES were derived from similarities in the decision space, which the experimental results from Chapter 4 were overall more adequate to be used in HDSSS problems. However, it was observed that the locality assumption within a region defined in the feature space was quite suitable in some cases, using the nearest neighbors rule in the OLP and recursive partitioning in the OLP++, as well as the other evaluated techniques, even over the same test bed. So it seems that different local methods may be able to describe more representative relationships over different problems, or over distinct instances within the same problem. Thus, there is an argument for modeling the samples' local relationships in multiple, and hopefully complementary, ways and encoding them in the graph, as the latter's high expressive power was not fully explored in this thesis.
- Another interesting research direction would be the development of a representation learning framework where the embedded space is optimized for the OLP. It was observed in Chapter 2 and Chapter 3 that the presence of local experts could boost the performance of local dynamic schemes. However, the OLP and OLP++ rely on region definition methods that were not optimized for the task, and as such may be susceptible to poor local distribution. The GNN-DES, on the other hand, is optimized for the dynamic selection task and may therefore learn an embedded space where the local distribution is more favorable. Nevertheless, the GNN-DES only works for a previously generated ensemble, which is usually produced with a global perspective of the problem and may not guarantee a good availability of local experts in all regions. Thus, developing a framework that learns an embedded space for producing the local rules might be a worthwhile endeavor for future research.

APPENDIX I

ON EVALUATING THE ONLINE LOCAL POOL GENERATION METHOD FOR IMBALANCE LEARNING

Mariana A. Souza[1] , George D. C. Cavalcanti[2] , Rafael M. O. Cruz[1] , Robert Sabourin[1]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA),
École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Centro de Informática,
Universidade Federal de Pernambuco,
1235 Av. Prof. Moraes Rego, Recife, Pernambuco, Brazil 50670-901

Article published in the Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2019

Abstract

Imbalanced problems are characterized by a disproportion between the number of samples from the classes in a classification problem. This difference in amount of examples may lead to a bias toward the majority class, hindering the recognition of the underrepresented minority class. Ensemble methods have been widely used for dealing with such problems, and have been shown to perform well on them. In this context, Dynamic Selection (DS) approaches, which perform the classification task on a local level, have been receiving some attention for their promising results. More specifically, the Frienemy Indecision Region Dynamic Ensemble Selection++ (FIRE-DES++) framework, which has yielded state-of-the-art results on imbalanced problems, use a data preprocessing technique for noise removal and a class-balanced neighborhood definition for coping with imbalanced datasets. A different DS-based approach proposed in a previous work, an online local pool generation method, generates on the fly locally accurate classifiers for labelling samples near the class borders. Though the local generation of the classifiers may reduce the impact of class imbalance on the performance of the technique, its suitability for imbalance learning was not yet evaluated. Thus, in this work we evaluate how well the online local pool generation method deals with imbalanced problems. We perform a comparative analysis with a baseline technique using three Dynamic Classifier Selection (DCS) techniques over 64 imbalanced datasets and four performance measures. We also evaluate the use of the preprocessing and balanced neighborhood definition steps from the FIRE-DES++ on the online scheme to assess their impact on the performance of the method. Moreover, we evaluate the online technique and its variants against seven state-of-the-art ensemble methods, including both static and DS approaches. Experimental results show that the approach of locally generating the classifiers is advantageous for imbalance learning, providing an improvement to the DCS techniques and yielding state-of-the-art results. Furthermore, the addition of the noise removal and the balanced neighborhood definition steps to the online scheme improved the overall results of the technique, which indicates the advantage of including such steps in DS-based techniques.

1. Introduction

Imbalanced distributions, in which a given class is underrepresented in comparison with the remaining ones in the problem, may be encountered in many real-world classification problems, such as biomedical diagnosis (Mazurowski, Habas, Zurada, Lo, Baker & Tourassi, 2008), detection of fraudulent bank account transactions (Wei, Li, Cao, Ou & Chen, 2013), image retrieval (Piras & Giacinto, 2012), text classification (Zheng, Wu & Srihari, 2004), and so on. For two-class problems, the class with fewer instances is referenced as the minority or positive class, while the class that contains the most instances is called the majority or negative class. Since one class outnumbers the other, many traditional classification models may end up biased in favor of the majority class, leading to a poor recognition rate of the examples from the minority class (Prati, Batista & Silva, 2015).

Several techniques have been proposed to deal with imbalance learning, and they can be categorized into four groups (Galar, Fernandez, Barrenechea, Bustince & Herrera, 2012; Fernández, García, Galar, Prati, Krawczyk & Herrera, 2018): algorithm-level approaches, data-level approaches, cost-sensitive learning frameworks, and ensemble based approaches. Algorithm-level approaches adapt the existing learning algorithms so as to adjust their bias toward the minority class. Data-level approaches, on the other hand, make use of resampling techniques to rebalance the class distribution of the problem. Cost-sensitive frameworks combine both data-level and algorithm-level approaches by adding cost to instances and modifying the learning procedure to incorporate those costs. Finally, the ensemble-based approaches make use of one of the previous approaches, usually data preprocessing, and an ensemble learning algorithm.

Ensemble methods have been widely used for handling imbalanced problems. Classical exemples are the cost-sensitive-based AdaCost (Fan, Stolfo, Zhang & Chan, 1999) and RareBoost (Joshi, Kumar & Agarwal, 2001), and the preprocessing-based SMOTE-Boost (Chawla, Lazarevic, Hall & Bowyer, 2003), RUSBoost (Seiffert, Khoshgoftaar, Van Hulse & Napolitano, 2010), OverBagging (Wang & Yao, 2009) and UnderBagging (Barandela, Valdovinos & Sánchez, 2003), among others. These techniques are based on Static Selection (SS), which defines the same ensemble to be used in the labelling of all query instances of a problem. Dynamic Selection (DS) approaches, on the other hand, single out the classifiers better fit for labelling each query sample in particular, usually by evaluating the local competence of each classifier in the neighborhood of that sample. Since they do not take the entire dataset into account when performing the classification task, which may reduce the impact of the disproportion between the classes, DS techniques are seen as a promising alternative for imbalance learning (Cruz *et al.*, 2018a).

A few recent works on ensemble-based approaches apply DS for dealing with imbalanced problems. The effect of using data preprocessing techniques combined with DS approaches for imbalance learning has been studied in (Roy *et al.*, 2018). In (Xiao, Xie, He & Jiang, 2012), a DS technique featuring cost-sensitive selection criteria was proposed for dealing with imbalanced data applied to the customer classification problem. The K-Nearest Oracles Borderline-Imbalance (KNORA-BI) (Oliveira *et al.*, 2018) technique was also proposed as an adaptation of the K-Nearest Oracles Eliminate (KNORA-E) Dynamic Ensemble Selection (DES) technique for imbalanced distributions. The Frienemy Indecision Region Dynamic Ensemble Selection++ (FIRE-DES++) framework (Cruz *et al.*, 2019a), which yielded state-of-the-art results over several public imbalanced datasets, makes use of a data preprocessing technique for noise removal and a class-balanced neighborhood definition in order to better select the classifiers in imbalanced local regions.

In a previous work (Souza *et al.*, 2019b), an online local pool generation method was proposed for dealing with samples in local class overlap regions of the feature space. The method consists of iteratively generating hyperplanes on the fly using the instances in the neighborhood of these samples, and then selecting the most competent ones using a Dynamic Classifier Selection (DCS) technique. This online scheme yielded a statistically equivalent performance to several classification models, including four state-of-the-art DES techniques (Souza *et al.*, 2019b). Though the online technique was not evaluated in the context of imbalance learning, the class-balanced local generation of the classifiers combined with their local evaluation and selection via DCS technique suggests that it may be somewhat robust to class imbalance.

So, in this work, we perform an evaluation of the online local pool generation technique from (Souza *et al.*, 2019b) with regards to imbalance learning. To that end, we compare its performance with a baseline pool generation technique using three DCS techniques over 64 two-class datasets, in order to evaluate whether generating the classifiers locally provide any advantage in imbalanced scenarios compared to only locally selecting them. The online scheme is also evaluated using the two additional modules included in the FIRE-DES++ framework, namely the DSEL pre-processing and the balanced region of competence definition, in order to assess whether these modules have a positive impact on the performance of the online method over imbalanced problems. Finally, the online local pool generation method and its variants including the additional modules are evaluated against seven state-of-the-art ensemble methods, four static and three DS-based, to assess its suitability for dealing with imbalance learning.

The rest of this work is organized as follows. In Section 2, the online pool generation method is briefly introduced. The comparative analysis between the online method and its variants and the FIRE-DES++ is then performed in Section 3. Lastly, the conclusions derived from this work are summarized and future works are suggested in Section 4.

2. Online local pool generation technique

The online local pool generation technique proposed in (Souza *et al.*, 2019b) is divided into two phases: an offline phase, in which the borderline training samples are identified, and an online phase, in which a pool of locally accurate classifiers is generated for labelling the query instances located on overlap regions of the feature space. The offline and the online phases of the method are described in Section 2.1 and Section 2.2, respectively.

2.1 Offline phase

In the offline phase, shown in Figure I-1, the borderline samples in the training set $\mathcal{T}$ are singled out using the K-Disagreeing Neighborhood (KDN) measure, shown in Equation A I-1, where $\mathcal{T}$ is the training dataset, $\mathbf{x_i}$ and $\mathbf{x_j}$ are training samples, y_i and y_j are their respective labels and k_h is the neighborhood size. The KDN measure calculates the proportion of samples from a different class in the neighborhood of a given sample. This measure indicates the degree of class overlap in the region where the sample is located, and it is used in the online phase for deciding whether to generate the local pool on the fly or not.

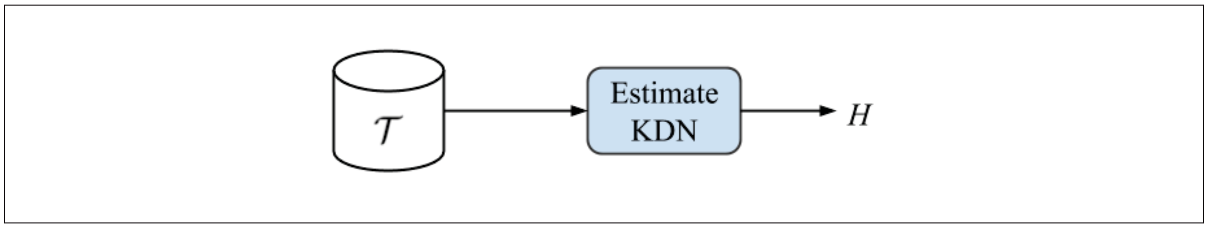


Figure-A I-1 Overview of the offline phase of the online local pool generation method. $\mathcal{T}$ is the training set and H is the set of KDN estimates, containing the KDN score (Equation A I-1) of all instances in $\mathcal{T}$.

$$\text{KDN}(\mathbf{x_i}, \mathcal{T}, k_h) = \frac{|\{\mathbf{x_j} | \mathbf{x_j} \in \text{KNN}(\mathbf{x_i}, \mathcal{T}, k_h) \wedge y_i \neq y_j\}|}{k_h} \tag{A I-1}$$

Algorithm overview

Algorithm-A I-1 Offline phase.

```
  input  : $\mathcal{T}, k_h$;                         ▷ Training dataset and KDN parameter
  output : $H$ ;                                       ▷ Estimated KDN scores
1 for every $\mathbf{x_i}$ in $\mathcal{T}$ do
2     $H(i) \leftarrow KDN(\mathbf{x_i}, \mathcal{T}, k_h)$ ;    ▷ Calculate KDN score (Eq. A I-1)
3 end for
4 return $H$
```

The pseudocode of the offline phase of the online local pool generation scheme is shown in Algorithm I-1. Its inputs are the training set $\mathcal{T}$ and the KDN neighborhood size k_h. From Step 1 to Step 3, the KDN score of each instance $\mathbf{x_i} \in \mathcal{T}$ is calculated and stored in H, which is then returned in Step 4.

2.2 Online phase

The online phase of the method, described in Figure I-2, is divided in three steps: region of competence estimation, local pool generation and generalization. In the first step, the region of competence θ_q of the query sample $\mathbf{x_q}$ is first obtained using regular KNN with a neighborhood size of k_s over the training set $\mathcal{T}$.

The region of competence θ_q is then evaluated based on the KDN scores stored in H, which was obtained in the offline phase. If none of the sample's neighbors are borderline samples, that is, if their KDN score is zero, then the procedure goes directly to the last step, generalization, and the KNN classifier yields the output label ω_l of $\mathbf{x_q}$. If, however, any of the neighbors $\mathbf{x_i} \in \theta_q$ is a borderline sample, the region is identified as an overlap region and the local pool (LP) is generated in the second step. The generation procedure of the local pool is explained next. Then, the generated locally accurate classifiers in LP are combined using the majority voting rule to obtain the output label ω_l in the generalization step.

Figure I-3 shows the generation procedure of the local pool (LP). The LP is constructed iteratively, and in each iteration the most locally accurate classifier produced in that iteration is added to

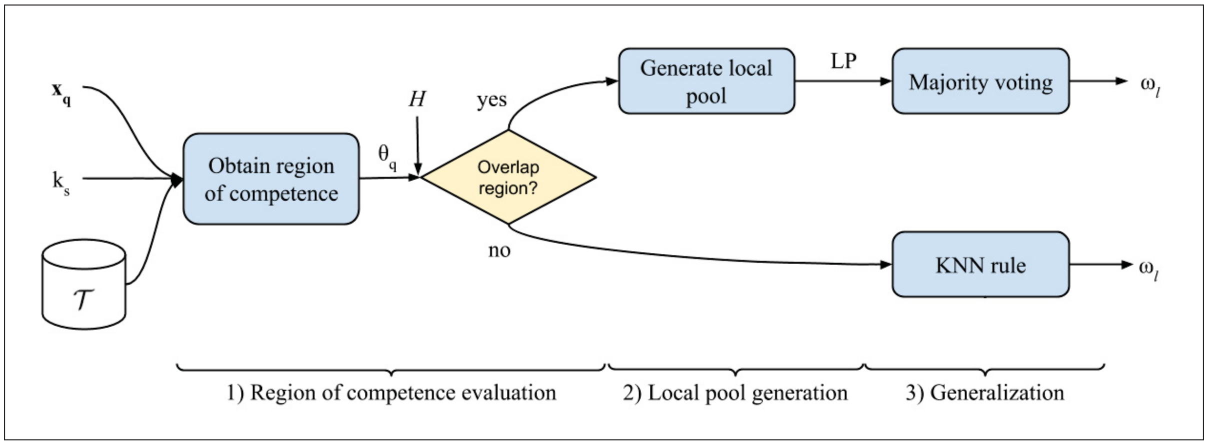


Figure-A I-2 Overview of the online phase of the online local pool generation method. $\mathcal{T}$ is the training set, $\mathbf{x_q}$ is the query sample, θ_q is its region of competence, k_s is the neighborhood size, H is the set of KDN estimates, LP is the local pool and ω_l is the output label of $\mathbf{x_q}$. In the online phase, the region of competence θ_q is first defined and evaluated based on the KDN scores in H, obtained in the offline phase. If θ_q does not contain borderline samples, that is, it is not an overlap region, the KNN rule is used to label $\mathbf{x_q}$ in the last step. Otherwise, the local pool is generated in the second step, and $\mathbf{x_q}$ is labelled via majority voting of the classifiers in LP in the third step.

LP. In a given m-th iteration, the query sample's neighboring instances in the training set $\mathcal{T}$ are obtained using a nearest neighbors procedure with neighborhood size k_m, calculated based on the region of competence size k_s. For two-class problems, the K-Nearest Neighbors Equality (KNNE) (Sierra *et al.*, 2011) is used. The KNNE is a variation of the regular KNN that selects the same amount of samples from each of the classes in the problem. Figure I-4 illustrates this procedure.

The query sample's neighborhood θ_m is then used as input to the Self-Generating Hyperplanes (SGH) method (Souza *et al.*, 2017), a pool generation method that yields an Oracle (Kuncheva, 2002) accuracy rate of 100% over the input dataset. The SGH method then returns a local subpool C_m that fully covers the class-balanced neighborhood θ_m. That is, the presence of at least one competent classifier $c_{m,k} \in C_m$ for each instance in θ_m is guaranteed. The indexes in the classifiers' notation indicates that the classifier $c_{m,k}$ is the k-th classifier from the m-th subpool (C_m). Then, the most competent classifier $c_{m,n}$ from C_m in the region delimited by

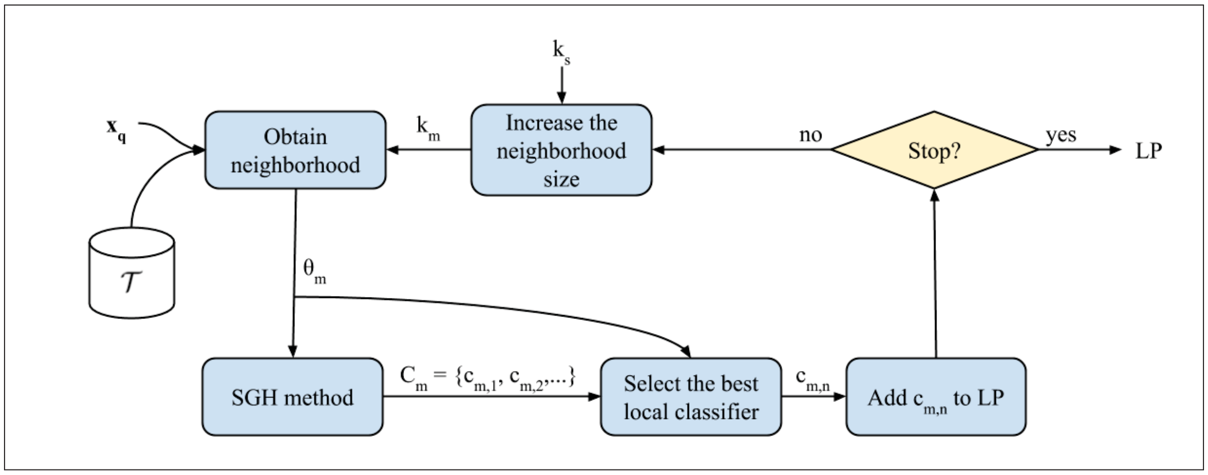


Figure-A I-3 Local pool generation step. $\mathcal{T}$ is the training step, $\mathbf{x_q}$ is the query sample and k_s is the size of the query sample's region of competence and LP is the final local pool, which is obtained iteratively. In the m-th iteration, the query sample's neighborhood θ_m of size k_m is obtained and used as input to the Self-Generating Hyperplanes (SGH) method (Souza *et al.*, 2017), which yields the subpool C_m. The classifiers from C_m are then evaluated over θ_m using a DCS technique. The classifiers' notation refers a classifier $c_{m,k}$ as the k-th classifier from the m-th subpool (C_m). The most competent classifier $c_{m,n}$ in subpool C_m is then selected and added to the local pool LP. This process is then repeated until LP contains a pre-defined amount of locally accurate classifiers.

the neighborhood θ_q is selected by a DCS technique and added to the local pool. The same procedure is performed in iteration $m + 1$ with the neighborhood size k_{m+1} increased by 2. This process is then repeated until the local pool contains a pre-defined amount of locally accurate classifiers.

Algorithm overview

The online phase of the technique is described in more detail in Algorithm I-2. Its inputs are the query sample $\mathbf{x_q}$, training set $\mathcal{T}$, the set of KDN scores H, the region of competence size k_s and the local pool size M.

Step 1 to Step 2 represent the region of competence evaluation step of the online phase (Figure I-2). In Step 1, the query sample's region of competence θ_q is obtained by selecting the k_s

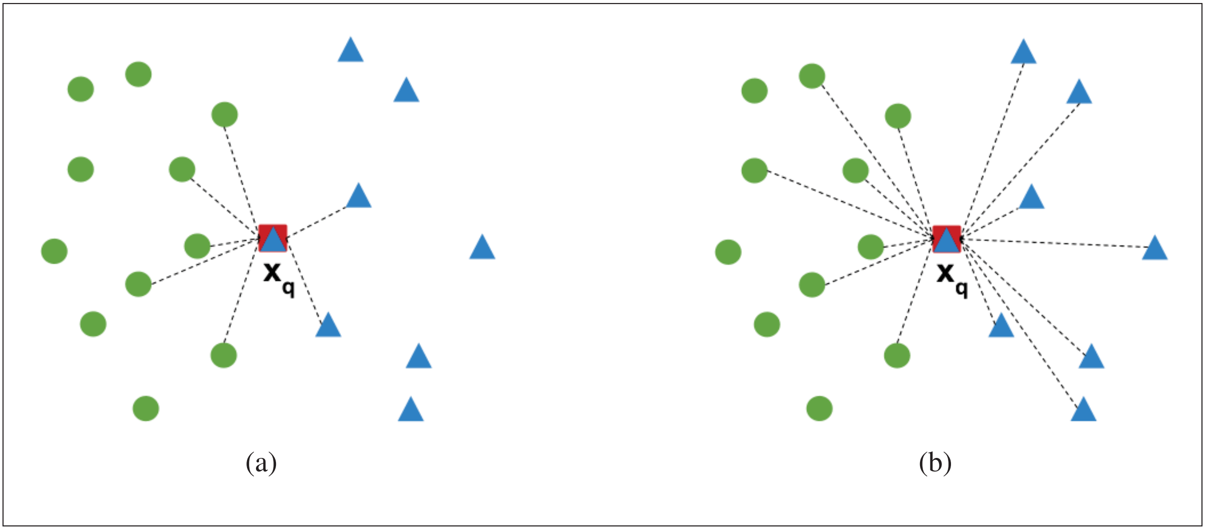


Figure-A I-4 Neighborhood definition of (a) KNN and (b) KNNE, with neighborhood size $K = 7$. In (a) the K closest examples to the query sample $\mathbf{x_q}$ are selected, while in (b) the K closest examples from each class are singled out.

Algorithm-A I-2 Online phase.

```
input  : $\mathbf{x_q}, \mathcal{T}, H$ ;                      ▷ Query sample, training set and KDN estimates
input  : $k_s, M$ ;                      ▷ Neighborhood size and pool size of local pool (LP)
output : $\omega_l$ ;                      ▷ Output label of $\mathbf{x_q}$
$\theta_q \leftarrow obtainRoC(\mathbf{x_q}, k_s, \mathcal{T})$ ;      ▷ Obtain the query instance's region of competence
if $\{\exists \mathbf{x_i} \in \theta_q | H(i) > 0\}$ then      1) Region of competence evaluation
    $LP \leftarrow \{\}$ ;                      ▷ Local pool initially empty
    for every $m$ in $\{1, 2, ..., M\}$ do      2) Local pool generation
        $k_m \leftarrow k_s + 2 \times (m - 1)$ ;      ▷ Increase neighborhood size by 2
        $\theta_m \leftarrow obtainNeighborhood(\mathbf{x_q}, k_m, \mathcal{T})$ ;      ▷ Obtain neighborhood of $\mathbf{x_q}$
        $C_m \leftarrow generatePool(\theta_m)$ ;      ▷ Generate local subpool $C_m$
        $c_{m,n} \leftarrow selectClassifier(\mathbf{x_q}, \theta_m, C_m)$ ;      ▷ Select best classifier in $C_m$
        $LP \leftarrow LP \cup \{c_{m,n}\}$ ;      ▷ Add $c_{m,n}$ to $LP$

    end for
    $\omega_l \leftarrow majorityVoting(\mathbf{x_q}, LP)$ ;      ▷ Label $\mathbf{x_i}$ with majority voting on LP
else      3) Generalization
    $\omega_l \leftarrow KNN(\mathbf{x_q}, k_s, \mathcal{T})$ ;      ▷ Label query sample using KNN rule

end if
return $\omega_l$
```

closest samples to $\mathbf{x_q}$ in the training set. The region of competence is then evaluated in Step 2. If all instances in θ_q are not borderline samples, that is, their KDN value is zero, the method goes straight to Step 13 (generalization step of Figure I-2) and the query sample's output label ω_l is obtained using the KNN rule with parameter k_s and returned in Step 15.

However, if there is one instance $\mathbf{x_i}$ from θ_q whose KDN estimate $H(i)$ is above zero, the region is considered an overlap one and the method proceeds to Step 3. Each classifier in the local pool (LP) is obtained in the loop that iterates M times, so Step 4 to Step 10 describe the local pool generation procedure from Figure I-3. In each iteration, the neighborhood size k_m is calculated in Step 5. Then, the query sample's neighborhood θ_m is obtained using a nearest neighbors method in Step 6. For two class problems, the KNNE is used in this step.

The subpool C_m is then generated in Step 7 using the SGH method with θ_m as training set. In Step 8, a DCS technique is then used to select the most competent classifier $c_{m,n}$ in C_m. The classifier $c_{m,n}$ is added to LP in Step 9, and then the loop continues until the local pool is complete. Finally, the query sample's label ω_l is obtained using majority voting over the locally accurate classifiers in LP in Step 11 and it is then returned in Step 15.

3. Experiments

Experiments were conducted over the 64 two-class datasets presented in Table I-1 from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository (Alcalá *et al.*, 2011). Each dataset was evaluated using a stratified 5-fold cross validation, one fold for test and the remaining for training, with the partitions already provided in the KEEL website. Since the number of samples in each dataset is quite small, the training set was also used as the dynamic selection dataset (DSEL) (Cruz *et al.*, 2018a) in the experiments for all evaluated DS techniques, as in (Cruz *et al.*, 2019a). The DSEL is a set of labelled samples used for region of competence definition in DS techniques.

The measures used to evaluate the performance of the techniques were the mean accuracy rate and three frequently used measures for imbalance learning: the area under the Receiver

Table-A I-1 Main characteristics of the datasets used in the experiments.

Ref.	Dataset	# Feat.	# Samples	IR	Ref.	Dataset	# Feat.	# Samples	IR
1	glass1	9	214	1.82	33	ecoli-0-2-6-7vs3-5	7	224	9.18
2	ecoli0vs1	7	220	1.86	34	glass-0-4vs5	9	92	9.22
3	wisconsin	9	683	1.86	35	ecoli-0-3-4-6vs5	7	205	9.25
4	pima	8	768	1.87	36	ecoli-0-3-4-7vs5-6	7	257	9.28
5	iris0	4	150	2	37	yeast-05679vs4	8	528	9.35
6	glass0	9	214	2.06	38	vowel0	13	988	9.98
7	yeast1	8	1484	2.46	39	ecoli-0-6-7vs5	6	220	10
8	haberman	3	306	2.78	40	glass-016vs2	9	192	10.29
9	vehicle2	18	846	2.88	41	ecoli-0-1-4-7vs2-3-5-6	7	336	10.59
10	vehicle1	18	846	2.9	42	led7digit-0-2-4-5-6-7-8-9vs1	7	443	10.97
11	vehicle3	18	846	2.99	43	glass-0-6vs5	9	205	11
12	glass0123vs456	9	214	3.2	44	ecoli-0-1vs5	6	240	11
13	vehicle0	18	846	3.25	45	glass-0-1-4-6vs2	9	205	11.06
14	ecoli1	7	336	3.36	46	glass2	9	214	11.59
15	new-thyroid1	5	215	5.14	47	ecoli-0-1-4-7vs5-6	6	332	12.28
16	new-thyroid2	5	215	5.14	48	ecoli-0-1-4-6vs5	6	280	13
17	ecoli2	7	336	5.46	49	cleveland-0vs4	13	177	12.62
18	segment0	19	2308	6	50	shuttle-c0vsc4	9	1829	13.87
19	glass6	9	214	6.38	51	yeast-1vs7	7	459	14.3
20	yeast3	8	1484	8.1	52	glass4	9	214	15.47
21	ecoli3	7	336	8.6	53	ecoli4	7	336	15.8
22	page-blocks0	10	5472	8.79	54	page-blocks-13vs4	10	472	15.86
23	ecoli-0-3-4vs5	7	200	9	55	glass-0-1-6vs5	9	184	19.44
24	yeast-2vs4	8	514	9.08	56	shuttle-c2-vs-c4	9	129	20.5
25	ecoli-0-6-7vs3-5	7	202	9.09	57	yeast-1458vs7	8	693	22.1
26	ecoli-0-2-3-4vs5	7	222	9.1	58	glass5	9	214	22.78
27	yeast-0-3-5-9vs7-8	8	506	9.12	59	yeast-2vs8	8	482	23.1
28	glass-0-1-5vs2	9	172	9.12	60	yeast4	8	1484	28.1
29	yeast-0-2-5-7-9vs3-6-8	8	1004	9.14	61	yeast-1289vs7	8	947	30.57
30	yeast-0-2-5-6vs3-7-8-9	8	1004	9.14	62	yeast5	8	1484	32.73
31	ecoli-0-4-6vs5	6	203	9.15	63	ecoli-0137vs26	7	281	39.14
32	ecoli-0-1vs2-3-5	7	224	9.17	64	yeast6	8	1484	41.4

Operation Characteristic (ROC) curve (AUC) (Fawcett, 2006), the F-measure (van Rijsbergen, 1979) and the Geometric Mean (G-mean) (Kubat, Matwin et al., 1997). The F-measure and the G-mean are described in Equation A I-2 and Equation A I-3, respectively, in which P is the number of test samples from the minority class (positive class), N is the number of test samples from the majority class (negative class), TP is the number of true positives, TN is the number of true negatives, TN is the number of true negatives and FP is the number of false positives.

$$F\text{-}measure = 2 \times \frac{TP}{P + TP + FP} \tag{A I-2}$$

$$G\text{-}mean = \sqrt{\frac{TP}{P} \times \frac{TN}{N}} \tag{A I-3}$$

The experiments were performed using the libraries scikit-learn (Pedregosa *et al.*, 2011), imbalanced-learn (Lemaître, Nogueira & Aridas, 2017) and DESLib (Cruz *et al.*, 2020), which provide implementations of several classification models in the Python 3.5 language.

The experimental analysis is divided into two parts. In Section 3.1, the online scheme and its variants are evaluated and compared to the baseline technique using three DCS techniques. In Section 3.2, the online technique and its variants are compared to seven state-of-the-art ensemble methods, four of which being static ensembles and three being dynamic ensemble selection frameworks.

3.1 Comparison with baseline technique

In this section, the online local pool generation technique is evaluated against the baseline technique, Bagging (Breiman, 1996), with a pool size of 100 classifiers, with the Perceptron as the base classifier (learning rate $\alpha = 0.001$ and number of iterations $n_{iter} = 100$), over three DCS techniques. The DCS techniques used in the experiments were the Overall Local Accuracy (OLA) (Woods *et al.*, 1997), Local Class Accuracy (LCA) (Woods *et al.*, 1997) and Multiple Classifier Behavior (MCB) (Giacinto *et al.*, 2000), all with a region of competence size of $K = 7$. No Dynamic Ensemble Selection (DES) techniques were evaluated since the generation procedure of the online scheme is not fit for selecting more than one classifier at a time (Souza *et al.*, 2019b).

The inclusion of the DSEL preprocessing and the balanced region of competence definition from the FIRE-DES++, namely using the Edited Nearest Neighbors (ENN) (Wilson, 1972) with $K = 3$, and the KNNE with $K = 7$, respectively, to the online scheme is also evaluated in order to assess its impact on the method's performance. Thus, the online local pool generation technique is evaluated in four versions: the original one (referenced as OLP), which defines the region of competence using KNN and generates the classifiers using KNNE, the OLP+KNNE, which uses KNNE for region of competence definition *and* for pool generation, the OLP+ENN, which is the original OLP with the DSEL set pre-processed using ENN, and the OLP+KNNE+ENN, which includes the two modifications previously mentioned. The pool size of all versions of the online technique was set to $M = 5$.

Table I-2 shows the mean performance of the online schemes and the baseline technique using the three evaluated DCS techniques. It can be observed that the OLP and its variants obtained a greater overall accuracy rate in comparison with Bagging using OLA and MCB. They also statistically outperformed the baseline technique in this scenario according to a Wilcoxon signed-rank test with a significance level of $\alpha = 0.05$.

In terms of AUC and G-mean, it can be observed that the baseline technique obtained a greater performance overall using OLA and MCB, though its performance was statistically similar to all versions of the online scheme. Using LCA, though, all versions of the OLP statistically outperformed Bagging considering these performance measures.

As for the F-measure, the OLP+KNNE+ENN obtained the highest mean performance using OLA and MCB, while using LCA the OLP+KNNE yielded the greater overall F-measure. The two versions of the OLP coupled with the ENN also obtained a statistically superior performance to Bagging using MCB. As in the previous case, all versions of the online scheme obtained a statistically similar performance to Bagging using OLA, and a significantly superior performance using LCA.

Thus, the results suggest that based on the same local evaluation by the DCS techniques, the approach of locally generating the classifiers on the fly is at least as effective for imbalance learning as dynamically selecting the classifiers from a globally generated pool obtained offline, and at most a more advantageous approach (as in the case of using LCA).

It can also be observed from Table I-2 that the inclusion of a balanced region of competence and a DSEL pre-processing improved the overall performance of the original online scheme. More specifically, the addition of the KNNE alone improved significantly the OLP using LCA and MCB in terms of accuracy, while using the ENN yielded a significant improvement using MCB in terms of AUC and F-measure and using OLA and MCB in terms of G-mean.

Table-A I-2 Mean (a) accuracy rate, (b) AUC, (c) F-measure and (d) G-mean of the evaluated techniques. Best results are in bold. The rows *p-value* show the resulting p-value of a Wilcoxon signed-rank test with $\alpha = 0.05$ between the performance of the indicated technique (Bagging or OLP) and the remaining methods. The p-values below α are underlined.

(a)

Technique		Bagging	OLP	OLP+KNNE	OLP+ENN	OLP+KNNE+ENN
	Avg.	92.81	93.95	**93.98**	93.62	93.66
OLA	p-value (Bag)	n/a	0.000	0.000	0.000	0.000
	p-value (OLP)	-	n/a	0.534	0.030	0.076
	Avg.	92.55	92.63	**92.90**	92.18	92.47
LCA	p-value (Bag)	n/a	0.802	0.273	0.169	0.692
	p-value (OLP)	-	n/a	0.001	0.001	0.553
	Avg.	92.95	94.00	**94.11**	93.84	93.84
MCB	p-value (Bag)	n/a	0.000	0.000	0.000	0.000
	p-value (OLP)	-	n/a	0.012	0.549	0.627

(b)

Technique		Bagging	OLP	OLP+KNNE	OLP+ENN	OLP+KNNE+ENN
	Avg.	**0.819**	0.808	0.809	0.815	0.817
OLA	p-value (Bag)	n/a	0.470	0.583	0.849	0.700
	p-value (OLP)	-	n/a	0.125	0.012	0.006
	Avg.	0.771	0.797	**0.798**	0.795	0.794
LCA	p-value (Bag)	n/a	0.000	0.000	0.003	0.004
	p-value (OLP)	-	n/a	0.193	0.495	0.494
	Avg.	**0.823**	0.808	0.810	0.821	**0.823**
MCB	p-value (Bag)	n/a	0.174	0.405	0.553	0.354
	p-value (OLP)	-	n/a	0.141	0.000	0.000

(c)

Technique		Bagging	OLP	OLP+KNNE	OLP+ENN	OLP+KNNE+ENN
	Avg.	0.672	0.673	0.675	0.682	**0.684**
OLA	p-value (Bag)	n/a	0.241	0.209	0.104	0.057
	p-value (OLP)	-	n/a	0.301	0.091	0.044
	Avg.	0.606	0.634	**0.640**	0.630	0.633
LCA	p-value (Bag)	n/a	0.011	0.003	0.025	0.010
	p-value (OLP)	-	n/a	0.002	0.747	0.377
	Avg.	0.680	0.678	0.681	0.693	**0.694**
MCB	p-value (Bag)	n/a	0.093	0.051	0.005	0.006
	p-value (OLP)	-	n/a	0.144	0.002	0.006

(d)

Technique		Bagging	OLP	OLP+KNNE	OLP+ENN	OLP+KNNE+ENN
OLA	Avg.	**0.773**	0.738	0.739	0.753	0.755
	p-value (Bag)	n/a	0.297	0.416	0.935	0.858
	p-value (OLP)	-	n/a	0.125	0.006	0.002
LCA	Avg.	0.680	**0.732**	0.730	0.730	0.725
	p-value (Bag)	n/a	0.000	0.000	0.001	0.002
	p-value (OLP)	-	n/a	0.213	0.427	0.543
MCB	Avg.	**0.781**	0.738	0.740	0.759	0.762
	p-value (Bag)	n/a	0.161	0.399	0.540	0.353
	p-value (OLP)	-	n/a	0.226	0.000	0.000

3.2 Comparison with the state-of-the-art

In this section, we compare the performance of the online local pool scheme and its variants evaluated in the previous section with seven state-of-the-art ensemble techniques. The static ensembles evaluated were AdaBoost (Freund & Schapire, 1997), RUSBoost (Seiffert *et al.*, 2010), Random Forest (RF) (Breiman, 2001) and Balanced Random Forest (BalancedRF) (Chen, Liaw & Breiman, 2004), all with a pool of 100 decision trees and with the hyperparameters set to the values suggested in (Fernández-Delgado, Cernadas, Barro & Amorim, 2014). The DES techniques evaluated were the FIRE-KNORA-E++ (FKNE++) (Cruz *et al.*, 2019a), the META-DES (Cruz *et al.*, 2015a) and the Randomized Reference Classifier (RRC) (Woloszynski & Kurzynski, 2011), also with a pool size of 100 classifiers, but with Perceptrons as base-classifiers. For simplicity, the online techniques used in the comparative analysis were coupled with MCB, since they obtained a greater overall performance in comparison with the other two DCS techniques evaluated in the previous section.

Table I-3 shows the mean performance of the evaluated techniques with regards to accuracy rate, AUC, F-measure and G-mean. It can be observed that the four versions of the OLP obtained the greatest mean accuracy rates save for the RF. They also yielded the highest average ranks after the RF according to a Friedman test.

In terms of AUC and F-measure, the BalancedRF obtained the highest average rank, while the OLP+KNNE+ENN and the OLP+ENN yielded the second and third highest mean ranks, respectively. The three alternate versions of the OLP also obtained the highest mean ranks in terms of F-measure, with the META-DES and the OLP right after in the ranking.

Since the p-values indicated in Table I-3 were below the confidence level of $\alpha = 0.05$, a post-hoc Bonferroni test with the same α was performed and the results are shown in the critical difference diagrams in Figure I-5. It can be observed that the OLP and its variants were significantly superior to FKNE++, RUSBoost and BalancedRF and statistically equivalent to the remaining state-of-the-art ensemble methods in terms of accuracy.

Table-A I-3 Mean performance of the evaluated state-of-the-art ensemble methods, the online local pool scheme and its variants, the latter of which coupled with MCB. Best results are in bold. The row *Avg. rank* indicates the average rank of each technique according to a Friedman test over the evaluated techniques, and the *p-value* row indicates the resulting p-value of the test.

Measure		AdaBoost	RUSBoost	RF	BalancedRF	FKNE++	META-DES	RRC	OLP	OLP+KNNE	OLP+ENN	OLP+KNNE+ENN
Accuracy	Avg.	93.37	92.57	**94.30**	86.87	92.79	93.71	93.77	94.00	94.11	93.84	93.84
	Avg. rank	6.492	8.367	**3.968**	9.757	7.640	5.414	5.296	4.804	4.414	4.953	4.890
	p-value											9.635×10^{-39}
AUC	Avg.	0.805	0.815	0.797	**0.880**	0.829	0.825	0.811	0.808	0.810	0.821	0.823
	Avg. rank	7.015	6.843	7.359	**2.492**	6.054	5.656	6.625	6.828	6.585	5.335	5.203
	p-value											6.471×10^{-20}
F-measure	Avg.	0.659	0.647	0.661	0.624	0.682	**0.700**	0.674	0.678	0.681	0.693	0.694
	Avg. rank	6.835	7.632	5.945	7.179	6.6719	5.304	6.281	5.703	5.265	**4.585**	4.593
	p-value											5.856×10^{-10}
G-mean	Avg.	0.740	0.760	0.711	**0.876**	0.791	0.777	0.745	0.738	0.740	0.759	0.762
	Avg. rank	7.101	6.656	7.679	**2.304**	6.00	5.664	6.687	6.937	6.671	5.226	5.070
	p-value											6.909×10^{-24}

In terms of AUC and G-mean, the BalancedRF significantly outperformed all evaluated techniques. The two versions of the online scheme that use ENN were significantly superior to RF only, while the other two had a similar performance to all techniques but the BalancedRF. In terms of F-measure, the OLP+ENN and the OLP+KNNE+ENN significantly outperformed RUSBoost, BalancedRF, AdaBoost and FKNE++, while the other two versions (OLP and OLP+KNNE) were significantly superior to RUSBoost only.

Thus, the results from Figure I-5 suggest that the online local pool technique yields a statistically similar performance to most of the evaluated state-of-the-art ensemble methods with regards to imbalance learning. The addition of the ENN and the KNNE to the online scheme also provided enough improvement to outperform few state-of-the-art techniques.

4. Conclusions and future work

In this work, we conducted an evaluation of the online local pool generation method from (Souza *et al.*, 2019b) with regards to imbalance learning, with the aim of assessing its suitability for handling imbalanced problems. A total of 64 datasets of varying imbalance degrees and four performance measures were used in the experiments. The comparative analysis featured three variations of the online scheme incorporating the data pre-processing and the balanced region of competence definition from the FIRE-DES++ (Cruz *et al.*, 2019a) technique, in order to evaluate

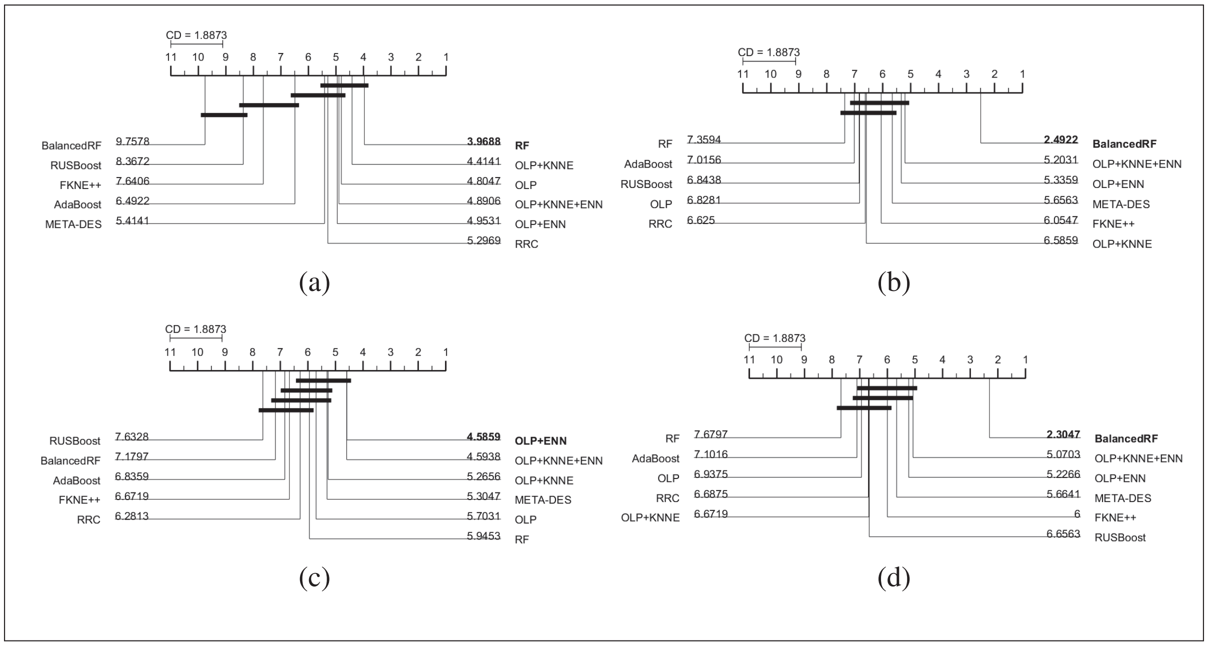


Figure-A I-5 Critical difference diagram of Bonferroni-Dunn post-hoc test on the results from Table I-3 in terms of (a) accuracy rate, (b) AUC, (c) F-measure and (d) G-mean. The critical value (CD) was computed with a confidence level pf $\alpha = 0.05$.

the impact of these additional steps on the performance of the method. The online scheme was also evaluated against four state-of-the-art static ensembles and three state-of-the-art dynamic selection frameworks.

Experimental results show that the online pool generation method and its alternate approaches performed generally better in terms of accuracy in comparison with the baseline technique, Bagging, using three DCS techniques. In terms of AUC, F-measure and G-mean, the online scheme was mostly statistically equivalent to Bagging when using OLA and MCB, and significantly superior to it when using LCA, which suggests generating the classifiers locally is an advantageous approach for dealing with imbalanced problems. Moreover, the inclusion of a balanced region of competence definition to the online scheme improved its accuracy rate, while the addition of a pre-processing step using the ENN improved the results in terms of AUC, F-measure and G-mean.These results indicate that the noise removal provided by the ENN and the use of balanced local regions is quite advantageous for local learning algorithms when dealing with imbalanced problems.

The comparative analysis with the state-of-the-art showed that the online local pool technique obtained a statistically similar performance to most evaluated state-of-the-art techniques, which suggests it is suitable for dealing with imbalanced problems. Furthermore, the online technique coupled with the additional modules of the FIRE-DES++ yielded a significantly superior performance in comparison with a few of the evaluated techniques.

Future works may include the design of a dynamic local noise removal or prototype generation procedure for better dealing with local imbalanced regions during the generation and the selection of the local classifiers.

APPENDIX II

MULTI-LABEL LEARNING FOR DYNAMIC MODEL TYPE RECOMMENDATION

Mariana A. Souza[1] , Robert Sabourin[1] , George D. C. Cavalcanti[2] , Rafael M. O. Cruz[1]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA),
École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Centro de Informática,
Universidade Federal de Pernambuco,
1235 Av. Prof. Moraes Rego, Recife, Pernambuco, Brazil 50670-901

Article published in the Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2020

Abstract

Dynamic selection techniques aim at selecting the local experts around each test sample in particular for performing its classification. While generating the classifier on a local scope may make it easier for singling out the locally competent ones, as in the online local pool (OLP) technique, using the same base-classifier model in uneven distributions may restrict the local level of competence, since each region may have a data distribution that favors one model over the others. Thus, we propose in this work a problem-independent dynamic base-classifier model recommendation for the OLP technique, which uses information regarding the behavior of a portfolio of models over the samples of different problems to recommend one (or several) of them in a per-instance manner. Our proposed framework builds a multi-label meta-classifier responsible for recommending a set of relevant base-classifier models based on the local data complexity of the region surrounding each test sample. The OLP technique then produces a local pool with the model that yields the highest probability score of the meta-classifier. Experimental results show that different data distributions favored different model types on a local scope. Moreover, based on the performance of an ideal model type selector, it was observed that there is a clear advantage in choosing a relevant base-classifier model for each

test instance in particular. Overall, the proposed model type recommender system yielded a statistically similar performance to the original OLP with fixed base-classifier model. However, the proposed framework struggled to recommend at least one relevant model type specially for the samples with low labelset cardinality. Given the novelty of the approach and the gap in performance between the proposed framework and the ideal selector, we regard this as a promising research direction.
Code available at github.com/marianaasouza/dynamic-model-recommender.

1. Introduction

Multiple Classifier Systems (MCS) combine the responses of several classifiers in the hopes that the combined system outperforms each individual base-classifier (Kittler *et al.*, 1998; Woźniak *et al.*, 2014). MCS are usually divided into three phases (Cruz *et al.*, 2018a): generation, in which the base-classifiers from the pool are generated, selection, in which a subset of the classifiers may be selected to perform the classification, and aggregation, in which the responses of the selected base-classifiers are combined. The classifier selection may be either static or dynamic, with the former being performed during training and the latter during generalization. The reasoning behind dynamic selection techniques is that each classifier in the pool may be a local expert in different regions of the feature space, so the dynamic selection schemes aim at selecting the base-classifier(s) that are best fit for labelling each test instance in particular. Dynamic selection techniques were shown to outperform static selection approaches specially on ill-defined problems (Britto *et al.*, 2014).

Yet since most pool generation methods used in dynamic selection schemes are classical techniques designed for static selection (Cruz *et al.*, 2018a), such as Bagging (Breiman, 1996) and Boosting (Schapire *et al.*, 1997), they generate the base-classifiers with a global perspective of the problem, so producing a local expert in the vicinity of all test instances is not guaranteed. Dynamic selection techniques were also shown to have difficulty in selecting a locally competent classifier even when it exists in the pool (Oliveira *et al.*, 2017; Souza *et al.*, 2017). In a previous work (Souza *et al.*, 2019b), it was proposed an online local pool generation method (OLP).

The OLP produces hyperplanes on the fly in the area surrounding each test instance near class borders, so as to guarantee the presence of locally accurate classifiers in the region and thus facilitate their selection by the dynamic selection techniques. Using the locally generated pool was shown to increase the frequency at which the most competent classifier is selected by the evaluated dynamic selection schemes in comparison to using a globally generated pool (Souza *et al.*, 2019b), and also to work well over imbalanced problems (Souza *et al.*, 2019a).

However, the production of local experts by the OLP technique was restricted by the base-classifier model used in the pool, which in this case are only two class Perceptrons. To the best of our knowledge, the choice of base-classifier model used in pool generation techniques is always done a priori in the literature regarding ensemble methods. However, in uneven distributions, each local region in the feature space may have different characteristics, such as data topology, class balance, class overlap and data density, among others. Since the idea is to select the base-classifier(s) that are experts in a given region, it would follow that the best model to learn the data from that region depends on its local data distribution. For instance, using the Perceptron as base-classifier may be far from ideal when trying to label a query sample located near a non-linearly separable local class border, while it would make sense to use it when labelling an instance near a linearly separable one. So, our hypothesis is that, by choosing a relevant base-classifier model for each instance in particular, we may be able to produce more locally competent base-classifiers in that region, and thus yield an improvement in performance compared to using a fixed base-classifier model for all samples.

Thus, we propose a framework for base-classifier model recommendation in order to, given a local data distribution, indicate which model is most suitable to be used for each test sample, with the purpose of yielding a pool of local experts for the online local pool scheme. We wish to answer the following research questions with our model type recommender system: (a) which data characteristics affect the performance of each base-classifier model on a local scope?, and (b) can we use this information to recommend a suitable model for each given query sample and improve the online method's performance? To do so, we use a problem-independent multi-label meta-classifier, which is obtained using information from the behavior of the base-classifier models over instances from previous datasets. We construct the multi-label set of the meta-problem using the class probabilities of each base-classifier model over several problems. We then associate its responses for each sample to its meta-features, which are comprised of local complexity measures obtained over the neighborhood of the corresponding sample in the training set, and train the multi-label meta-classifier afterwards. In generalization, the meta-features of each query instance are first extracted using its neighboring samples in the training set, which are then fed into the meta-classifier, who outputs the relevant base-classifier models for that instance.

Our proposed framework is closely related to the algorithm recommendation area. Several recommender systems for algorithm selection based on data complexity measures were proposed in the literature. In (Garcia, Lorena, de Souto & Ho, 2018), the authors use a meta-regressor, built with the extracted data complexity measures of several datasets, to select the best-performing classification model for a given unknown problem. A somewhat similar framework is proposed in (Deng, Chen & Pan, 2018), in which the authors propose a set of meta-features based on data complexity extracted from a kernel matrix generated from the data, in order to recommend the most suitable classification model using an NN rule. Another algorithm recommender system was proposed in (das Dôres, Alves, Ruiz & Barros, 2016) for fault prediction in software projects. The meta-data was obtained by extracting a set of meta-features, including simple, statistical and data complexity measures, from the data of previous software projects and assigning which model in the portfolio yielded the best performance, according to the balance criterion (Menzies, Greenwald & Frank, 2006), over each project as the meta-label. However, none of these recommender systems base their model recommendation procedure on the local data characteristics within a given dataset. Moreover, they recommend the models for an entire problem, not for each instance in particular. To the best of our knowledge, *per-sample* meta-learning for algorithm recommendation was only explored in certain fields, such as combinatorial search problems (Kotthoff, 2016) and collaborative filtering (Collins, Tkaczyk & Beel, 2018).

This work is divided as follows. Section 2 describes our proposed base-classifier model recommender system. Experimental results are presented in Section 3. Lastly, we present our concluding remarks in Section 4.

2. Proposed framework

2.1 Online local pool method

Before delving into the proposed model type recommender system, we briefly present the OLP technique next. The OLP technique attempts to exploit the properties of the Oracle (Kuncheva, 2002) on a local scope in order to guide the generation of the local base-classifiers. The Oracle is an ideal selector which always chooses the base-classifier in the pool that labels a given test sample correctly, if such classifier exists. The OLP technique was shown to yield a similar performance to state-of-the-art ensemble methods (Souza *et al.*, 2019b), and also to perform quite well on imbalanced distributions (Souza *et al.*, 2019a).

In the offline phase of the OLP, the K-Disagreeing Neighbors (KDN) (Smith *et al.*, 2014) estimates of each training sample is computed, in order to identify which areas of the feature space present a local class border. The KDN measure calculates the proportion of samples from a different class in the neighborhood of a given sample.

The online phase of the OLP is described in Figure II-1, in which k_s is the dynamic selection neighborhood size, H is the set of KDN estimates and LP is the local pool. It is divided in three steps: region of competence estimation, local pool generation and generalization. In the first step, the neighborhood θ_q of the query sample is first obtained using regular K-NN, with size k_s over the training set, and then evaluated based on the KDN scores stored in H, obtained in the offline phase. If none of the sample's neighbors are borderline samples, that is, if their KDN score is zero, then the procedure goes directly to the third and last step, generalization, and the K-NN classifier yields the predicted label $\hat{y}_q$. If, however, any of the neighbors $\mathbf{x_i} \in \theta_q$ is a borderline sample, the region is identified as a borderline area and the local pool (LP) is generated in the next step.

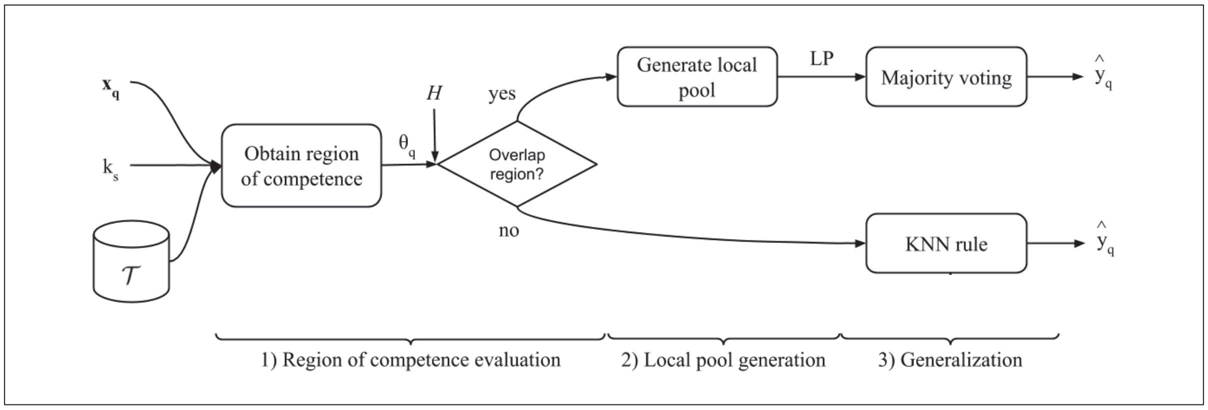


Figure-A II-1 Overview of the online phase of the online local pool generation method (from (Souza *et al.*, 2019a)). The symbols k_s, H and LP mean the dynamic selection technique's neighborhood size, the set of KDN estimates, and the local pool, respectively.

In the second step, the LP is produced iteratively, and in each iteration the most locally competent classifier produced in that iteration is added to the final pool (Figure II-2). In a given m-th iteration, the query sample's neighboring instances in the training set $\mathcal{T}$ are obtained using a neighborhood size of k_m, calculated based on k_s. For two-class problems, the K-Nearest Neighbors Equality (K-NNE) (Sierra *et al.*, 2011), which selects the same amount of neighbors from each class, is used in this step.

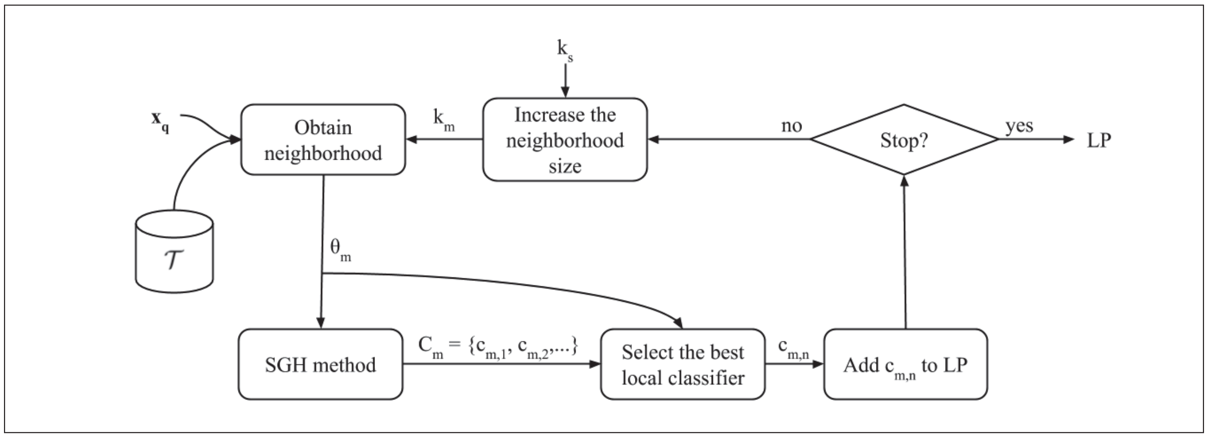


Figure-A II-2 Local pool generation step (from (Souza *et al.*, 2019a)). The symbols k_s and LP mean the dynamic selection technique's neighborhood size and the local pool, respectively.

The query sample's neighborhood θ_m is then used as input to the Self-generating Hyperplanes (SGH) method (Souza *et al.*, 2017), a pool generation method that yields an Oracle accuracy rate of 100% over the input dataset. The SGH then produces a local subpool C_m in which the presence of at least one competent classifier $c_{m,k} \in C_m$ for each instance in θ_m is guaranteed. The indexes in the classifiers' notation indicates that the classifier $c_{m,k}$ is the k-th classifier from the m-th subpool.

Then, the most competent classifier $c_{m,n}$ from C_m in the region delimited by the neighborhood θ_q is selected by a DCS technique and added to the local pool. The selection by a DCS technique is performed at this stage, and not after the LP is completed, because the subpool generation by the SGH method yields too diverse classifiers (Souza *et al.*, 2017), so not only is it not fit for DES techniques, but also it may generate classifiers that are near opposite to the local border (Souza *et al.*, 2019b). Thus, the dynamic selection is performed concurrently with the generation. The same procedure using the SGH method is performed in iteration $m + 1$ with the neighborhood size k_{m+1} increased by 2, in order to not only adjust the locality of the classifiers but also to provide a different set of training samples and produce slightly diverse base-classifiers. This process is then repeated until the local pool contains a predefined amount (M) of locally accurate classifiers.

In the last step, generalization, the predicted label $\hat{y}_q$ of the query sample $\mathbf{x_q}$ is produced (Figure II-1). If the LP was generated, the responses of the base-classifiers in LP are combined using the majority voting rule. Otherwise, the K-NN is used to obtain $\hat{y}_q$.

2.2 Dynamic model type recommendation system

Though the OLP yielded promising results reported in previous works, it presents several limitations, one of the greatest being its local pool generation procedure based on the SGH method. Although the latter presents interesting qualities, the base-classifier generation is done using an heuristic and, for this reason, it can only produce two-class Perceptrons, yielding hyperplanes that are not always well adjusted to the border depending on the local data distribution around the query sample. Since different classification model types fit the data in different ways, our hypothesis is that locally training a suitable base-classifier model according to the local data distribution may be advantageous for producing a more competent set of classifiers for the OLP. Thus, we propose a dynamic base-classifier model recommender system which indicates, for each test instance, the *relevant* base-classifier models considering the data complexity around the sample.

The choice of a suitable base-classifier model to be trained for each instance can be formulated as a meta-learning problem, in which:

- The meta-classes correspond to the model types that are suitable for a particular instance $\mathbf{x_i}$. Since more than one base-classifier model may be suitable for each given sample $\mathbf{x_i}$, it is associated with a meta-labelset U_i, or its corresponding meta-label vector $\mathbf{u_i}$, a binary vector indicating the relevant models.
- Each element $v_{i,j}$ of the meta-feature vector $\mathbf{v_i}$ corresponds to a different complexity measure extracted in the neighborhood of the sample $\mathbf{x_i}$.
- A multi-label meta-classifier is trained on the meta-dataset to predict which base-classifier models are relevant for a given query sample $\mathbf{x_q}$, according to its meta-feature vector $\mathbf{v_q}$ extracted in its neighborhood.

2.3 Meta-training step

In the meta-training step (Figure II-3), we obtain our meta-data by evaluating the OLP technique with different model types and associating, for each instance, the information regarding which ones were successful (meta-labels) to the complexity measures extracted in the sample's neighborhood (meta-features). We then join the meta-examples obtained from several problems to form the meta-training set, which is used to train our meta-classifier. It is important to note that, since for each instance in a given problem there may be more than one suitable base-classifier model, our meta-learning problem is also a multi-label one. Thus, our meta-classifier must be able to deal with this scenario. Figure II-4 shows in more detail the meta-feature extraction and the algorithm evaluation for each dataset from Figure II-3 individually.

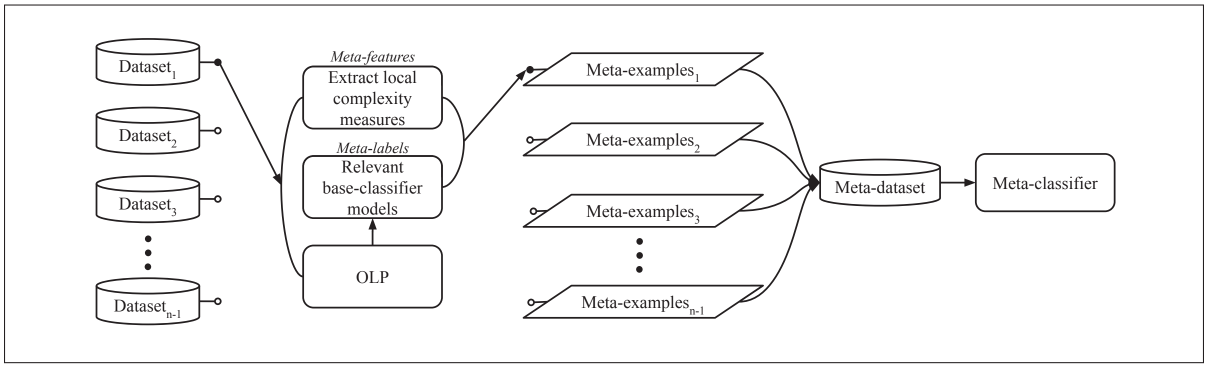


Figure-A II-3 Overview of the training phase of the proposed framework.

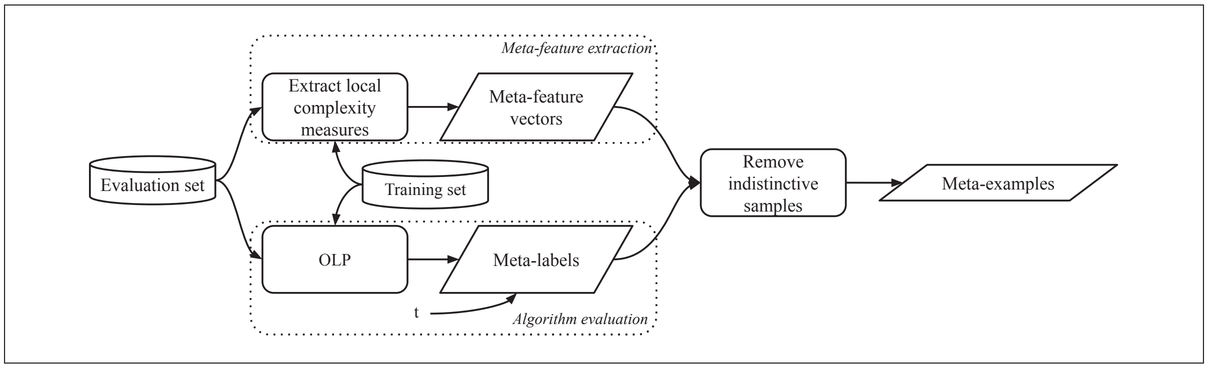


Figure-A II-4 Meta-feature extraction and algorithm evaluation for each dataset in the meta-training phase. The threshold *t* is used to define the relevance of each model for a given sample.

2.3.1 Meta-feature extraction

In order to characterize the local data complexity of each sample in the evaluation set (Figure II-4), we use 12 data complexity measures described in (Lorena *et al.*, 2019). Our reasoning for choosing these measures was based on the sort of information each of them brings for characterizing a local data distribution. We selected a subset of measures that cover all data complexity aspects described in (Lorena *et al.*, 2019) with the exception of dimensionality, which is problem-dependent and would not help characterizing a local area in a problem-independent manner. Most of the chosen complexity measures were also shown to have a good discriminating power for predicting algorithm performance on a global scope (Garcia *et al.*, 2018; Muñoz, Villanova, Baatar & Smith-Miles, 2018), specially the distance-based measures. The complexity measures selected as meta-features for the proposed framework are:

- Maximum Individual Feature Efficiency (F3): This measure assesses the degree of ambiguity of the feature which presents the smallest overlap of values between samples from different classes.
- Collective Feature Efficiency (F4): The F4 measure gives an insight on the degree of efficiency provided by a given set of features, and is calculated using the F3 measure iteratively over a given set of points.
- Error Rate of Linear Classifier (L2): The L2 measure is defined as the error rate of a linear SVM trained over the input dataset.
- Non-Linearity of a Linear Classifier (L3): The L3 measure tries to quantify the degree of linearity of a problem's class borders, and is defined as the error rate of a linear classifier obtained with the original training set over prototypes generated via interpolation of the training points.
- Fraction of Borderline Points (N1): The N1 measure is obtained by generating a minimum spanning tree (MST) using the distance matrix from all points of the input set and then calculating the proportion of samples that are connected to a sample from a different class, thus conveying the size and degree of complexity of the decision boundary.
- Ratio of Intra/Extra Class Nearest Neighbor Distance (N2): The N2 measure estimates the iter/intra class relationship of the data by computing the ratio between the sum of the distances between the nearest neighbors (1-NN) of the same class and the sum of the distances between the nearest neighbors that possess different labels, for the entire input set.
- Error Rate of the Nearest Neighbor Classifier (N3): The N3 measure is defined as the error rate of the 1-NN classifier over the input dataset, computed using a leave-one-out procedure.
- Non-Linearity of the Nearest Neighbor Classifier (N4): The N4 measure is similar to the L3 measure, in that it generates a few prototypes via interpolation and evaluates a classifier, trained with the original training samples, over the newly generated prototypes. In the case of the N4, the classifier used is the 1-NN.

- Local Set Average Cardinality (LSC): The LSC is defined as the average number of samples within the local set (LS) of each instance in the input set. The LS of a given instance is the set of samples that share the same label as the target instance while being closer to it than its nearest enemy.
- Average density of the network (Den): The Density measure is computed using a graph constructed so that each node is a training instance and each edge is a distance-based weighted connection between them, with the edges with distance greater than a given threshold are discarded, as well as all edges connecting samples from different classes. The measure is then calculated as the normalized number of edges in the graph.
- Entropy of class proportions (C1): The C1 measure estimates the normalized entropy of the class sizes, giving an insight into the class imbalance of the data.
- Imbalance ratio (C2): Also referred to as IR, the imbalance ratio of binary problems is defined as the ratio between the number of samples in the majority class and the number of samples from the minority class.

In the meta-feature extraction step, we calculate for each sample $\mathbf{x_i}$ in the evaluation set (Figure II-4) its neighborhood over the training set using the K-NN, with neighborhood size k'. We then extract the 12 complexity measures over the neighborhood of $\mathbf{x_i}$, yielding the meta-feature vector $\mathbf{v_i}$ depicted in Figure II-5.

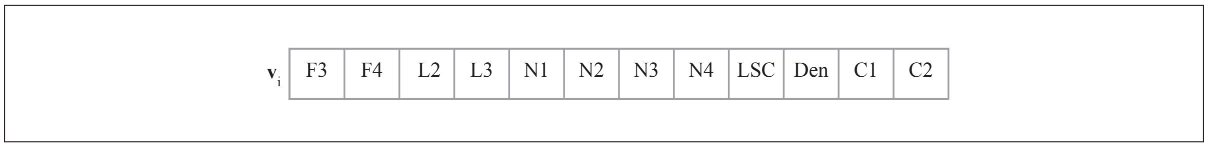


Figure-A II-5 Example of meta-feature vector.

2.3.2 Algorithm evaluation

All samples in the evaluation set are tested using the OLP m times, with m being the number of model types considered in the meta-learning framework's portfolio. So, for each sample $\mathbf{x_i}$, the base-classifiers used in the executions that yielded the correct label y_i with class probability above a threshold t are referenced as relevant for that sample. The vector of meta-labels $\mathbf{u_i}$ corresponding to the sample $\mathbf{x_i}$ is illustrated in Figure II-6. Each column indicates the relevance of the base-classifier model for that sample. A relevant model for a given sample $\mathbf{x_i}$ is one with which the OLP technique was able to correctly label with output class probability above a threshold t. If the OLP with the base-classifier model was unable to classify the sample correctly, or the class probability was below t, the model is deemed non-relevant to that sample and its corresponding value in $\mathbf{u_i}$ is 0. That way, we indicate to the meta-classifier which base-classifier models are indeed more likely to successfully learn the local data distribution.

We then remove the samples for which all base-classifier models yielded the same response, referred to in Figure II-4 as *indistinctive* samples. Since for these samples any model will produce the same output, they do not help in discriminating between the base-classifier models. Thus, similarly to (Cruz *et al.*, 2015a), we remove the indistinctive samples in order for the meta-classifier to focus on the distinctive ones.

	Model 1	Model 2	Model 3	Model 4	Model 5		Model *m*
$\mathbf{u}_i$	0	0	1	0	1	...	1

Figure-A II-6 Example of meta-label vector. The assigned value is 1 if the model correctly classified the sample with output class probability above a threshold t, or 0 otherwise.

2.3.3 Meta-classifier training

Lastly, the meta-classifier is trained with the meta-data. Since our meta-problem is a multi-label one (Figure II-6), we need a multi-label learning method for dealing with it. For simplicity, we chose to use the Binary Relevance (BR) method (Boutell, Luo, Shen & Brown, 2004), a problem transformation approach in which we transform the problem into several binary datasets, one for each label. Though limited, in the sense that it assumes the labels are independent, this approach is simple and easily adapted to our problem. Thus, our meta-classifier is a set of m classifiers, each one trained to indicate whether its corresponding model is relevant for a given input sample.

However, since using the BR method may yield highly imbalanced binary problems, we train the meta-classifier applying class weights, with the weights being adjusted inversely proportional to the class frequencies in order to reduce the impact of the class imbalance.

2.4 Generalization step

In the generalization phase (Figure II-7), the unseen training data is first analyzed and the training samples qualified into borderline (hard) or not (easy) using its KDN estimate. Then, in generalization, if the unknown sample $\mathbf{x_q}$ is considered easy, it is labelled by the K-NN, as in the original OLP technique. Otherwise, the local complexity measures are extracted over the neighborhood of the sample θ'_q with size k', the same as in the meta-training step. The meta-feature vector $\mathbf{v_q}$ is then used as input to the meta-classifier, which returns which base-classifier models are relevant for the sample $\mathbf{x_q}$. Among the recommended models, the one whose meta-classifier (within the BR ensemble) outputs the highest class probability is chosen to be used. The OLP technique is then applied to the query sample with the chosen base-classifier model and yields the predicted label $\hat{y}_q$. If no model is recommended, the class probability rule is applied to all models (as in the T-Criterion rule for the BR method (Zhang & Zhou, 2013)).

3. Experiments

3.1 Experimental protocol

The impact on the performance of the OLP using our meta-learning framework for choosing a relevant base-classifier model according to the local data complexity is assessed using a leave-one-dataset-out procedure, in order to achieve a problem-independent approach to our model recommendation scheme. That is, we use one dataset from our testbed in the generalization step and the remaining ones in the meta-training step for obtaining the meta-data. Within each dataset, we use a 5-fold cross validation procedure, both for algorithm evaluation and meta-feature extraction in the meta-training step and for evaluating the performance of the method in test in generalization.

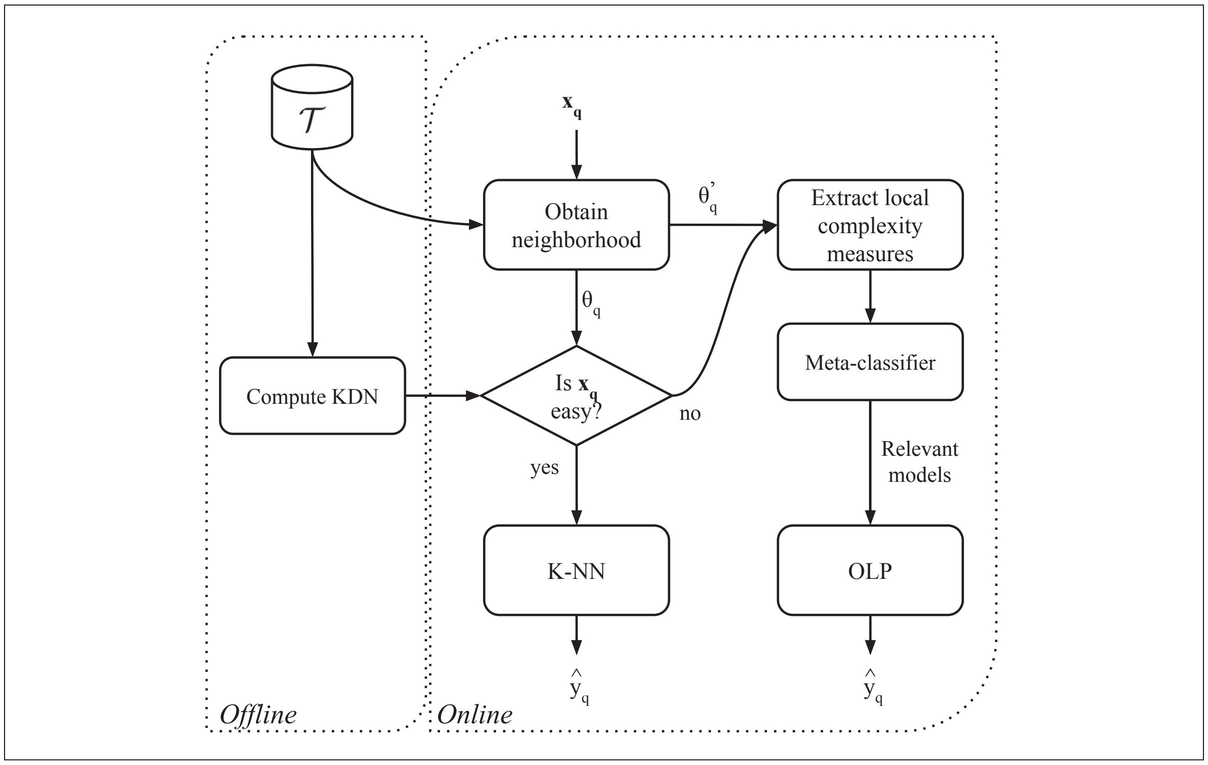


Figure-A II-7 Overview of the generalization phase of the proposed framework.

3.1.1 Datasets

For facilitating the comparison with previous works, we use the same testbed of 64 two-class datasets from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository (Alcalá *et al.*, 2011), presented in Table II-1. Each dataset was evaluated using a stratified 5-fold cross validation procedure, one fold for test and the remaining for training, using the same partitions provided in the KEEL website for reproducibility. Due to the small-sized datasets, we use the training set as the DSEL set for the dynamic selection techniques evaluated, as in (Cruz *et al.*, 2019a; Souza *et al.*, 2019a).

3.1.2 Classifier models

We consider 5 base-classifier models to be chosen by the meta-learning framework: Perceptron, Decision Stump (DS), Decision Tree (DT), linear SVM (LSVM) and SVM with Gaussian

Table-A II-1 Main characteristics of the datasets used in the experiments.

Ref.	Dataset	# Feat.	# Samples	IR	Ref.	Dataset	# Feat.	# Samples	IR
1	glass1	9	214	1.82	33	ecoli-0-2-6-7vs3-5	7	224	9.18
2	ecoli0vs1	7	220	1.86	34	glass-0-4vs5	9	92	9.22
3	wisconsin	9	683	1.86	35	ecoli-0-3-4-6vs5	7	205	9.25
4	pima	8	768	1.87	36	ecoli-0-3-4-7vs5-6	7	257	9.28
5	iris0	4	150	2	37	yeast-05679vs4	8	528	9.35
6	glass0	9	214	2.06	38	vowel0	13	988	9.98
7	yeast1	8	1484	2.46	39	ecoli-0-6-7vs5	6	220	10
8	haberman	3	306	2.78	40	glass-016vs2	9	192	10.29
9	vehicle2	18	846	2.88	41	ecoli-0-1-4-7vs2-3-5-6	7	336	10.59
10	vehicle1	18	846	2.9	42	led7digit-0-2-4-5-6-7-8-9vs1	7	443	10.97
11	vehicle3	18	846	2.99	43	glass-0-6vs5	9	205	11
12	glass0123vs456	9	214	3.2	44	ecoli-0-1vs5	6	240	11
13	vehicle0	18	846	3.25	45	glass-0-1-4-6vs2	9	205	11.06
14	ecoli1	7	336	3.36	46	glass2	9	214	11.59
15	new-thyroid1	5	215	5.14	47	ecoli-0-1-4-7vs5-6	6	332	12.28
16	new-thyroid2	5	215	5.14	48	ecoli-0-1-4-6vs5	6	280	13
17	ecoli2	7	336	5.46	49	cleveland-0vs4	13	177	12.62
18	segment0	19	2308	6	50	shuttle-c0vsc4	9	1829	13.87
19	glass6	9	214	6.38	51	yeast-1vs7	7	459	14.3
20	yeast3	8	1484	8.1	52	glass4	9	214	15.47
21	ecoli3	7	336	8.6	53	ecoli4	7	336	15.8
22	page-blocks0	10	5472	8.79	54	page-blocks-13vs4	10	472	15.86
23	ecoli-0-3-4vs5	7	200	9	55	glass-0-1-6vs5	9	184	19.44
24	yeast-2vs4	8	514	9.08	56	shuttle-c2-vs-c4	9	129	20.5
25	ecoli-0-6-7vs3-5	7	202	9.09	57	yeast-1458vs7	8	693	22.1
26	ecoli-0-2-3-4vs5	7	222	9.1	58	glass5	9	214	22.78
27	yeast-0-3-5-9vs7-8	8	506	9.12	59	yeast-2vs8	8	482	23.1
28	glass-0-1-5vs2	9	172	9.12	60	yeast4	8	1484	28.1
29	yeast-0-2-5-7-9vs3-6-8	8	1004	9.14	61	yeast-1289vs7	8	947	30.57
30	yeast-0-2-5-6vs3-7-8-9	8	1004	9.14	62	yeast5	8	1484	32.73
31	ecoli-0-4-6vs5	6	203	9.15	63	ecoli-0137vs26	7	281	39.14
32	ecoli-0-1vs2-3-5	7	224	9.17	64	yeast6	8	1484	41.4

kernel (GSVM). We compare the proposed framework with dynamic model type selection against the original OLP method, which generates the base-classifiers using the Self-Generating Hyperplanes (SGH) (Souza *et al.*, 2017) technique and uses a dynamic classifier selection technique embedded (we chose the Multiple Classifier Behavior (MCB) (Giacinto *et al.*, 2000) due to its superior performance in previous experiments). Moreover, we use the Decision Tree as our BR meta-learner in the multi-label framework. We chose this classifier because of its embedded feature selection, which allows us to analyze how correlated the local characteristics (meta-features) are with the relevance of each base-classifier model.

3.1.3 Parameter setting

The pool size of the OLP, regardless of the base classifier used, is set to $M = 5$. The neighborhood size for the KDN and the neighborhood definitions within the OLP framework are set to $k_h = k_s = 7$. For the version that uses the SGH and MCB, the similarity and competence threshold of the latter are set to 0.7 and 0.1, respectively.

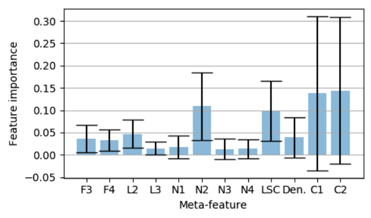
For the meta-learning framework, we fix the class probabilities threshold t at 0.7 in order to regard a base-classifier model as relevant or not for a given sample. Moreover, the neighborhood size for the meta-feature extraction is set to $k' = 50$, providing enough samples for reliably estimating the measures with a local scope (Kotsiantis, Kanellopoulos & Pintelas, 2006). The meta-learner hyperparameters (maximum depth, minimum impurity decrease, minimum samples per leaf) are obtained using a grid search in a 10-fold cross-validation procedure over the meta-training set.

3.1.4 Performance measures

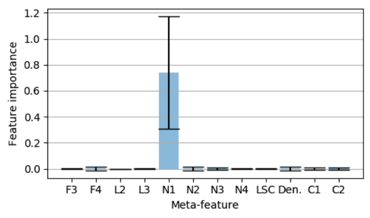
In order to evaluate the impact of the automatic choice of base-classifier model on the performance of the OLP, we use the accuracy rate, the area under the Receiver Operating Characteristic (ROC) curve (AUC) (Fawcett, 2006), the F-measure (van Rijsbergen, 1979) and the Geometric Mean (G-mean) (Kubat *et al.*, 1997), the latter three being measures frequently used for performance evaluation in imbalanced scenarios. Moreover, we use the Precision measure (Zhang & Zhou, 2013) for evaluating the performance of our multi-label meta-classifier, since for our scenario, it is more important that the set of recommended base-models is mostly comprised of suitable base-models than it contains a high proportion of unsuitable base-models, albeit including all suitable ones. In this sense, the precision of the meta-learner is the lower bound of our framework accuracy rate.

3.2 Multi-label meta-classifier analysis

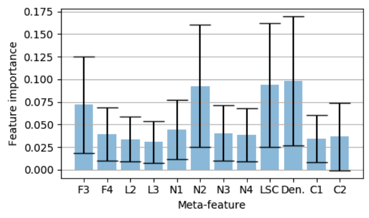
We first analyze which characteristics (complexity measures) are more pertinent for recommending each base-classifier model. Figure II-8 shows the mean meta-feature importances obtained from the multi-label meta-classifier (Decision tree). The importance of a feature in a DT is the normalized total reduction of the Gini impurity given by that feature. Since we used the BR method, the recommendation of each base-classifier model is given by an individual DT, so the feature importances for each model are shown in Figure II-8.



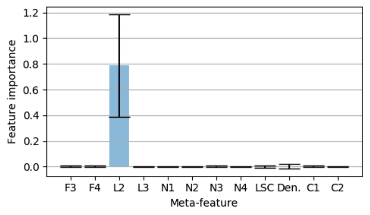
(a) Perceptron



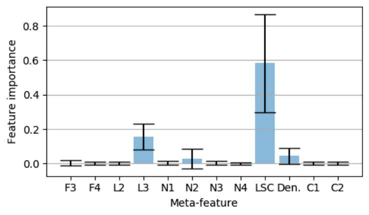
(b) DS



(c) DT



(d) LSVM



(e) GSVM

Figure-A II-8 Mean feature importances of each component of the multi-label meta-classifier over all datasets from from Table II-1.

Unsurprisingly, the most important meta-feature for recommending the LSVM is the L2 measure, which computes the error rate of a linear SVM on the local region. For the DS, the most important meta-feature appears to be the N1, which indicates the size of the local border. The recommendation of the Perceptron, on the other hand, was highly influenced by the local class balance and level of overlap in the target region. The level of class overlap and the linearity of the border is mostly regarded for recommending the GSVM. The recommendation of the DT, on the other hand, is quite different, with most meta-features being almost equally important.

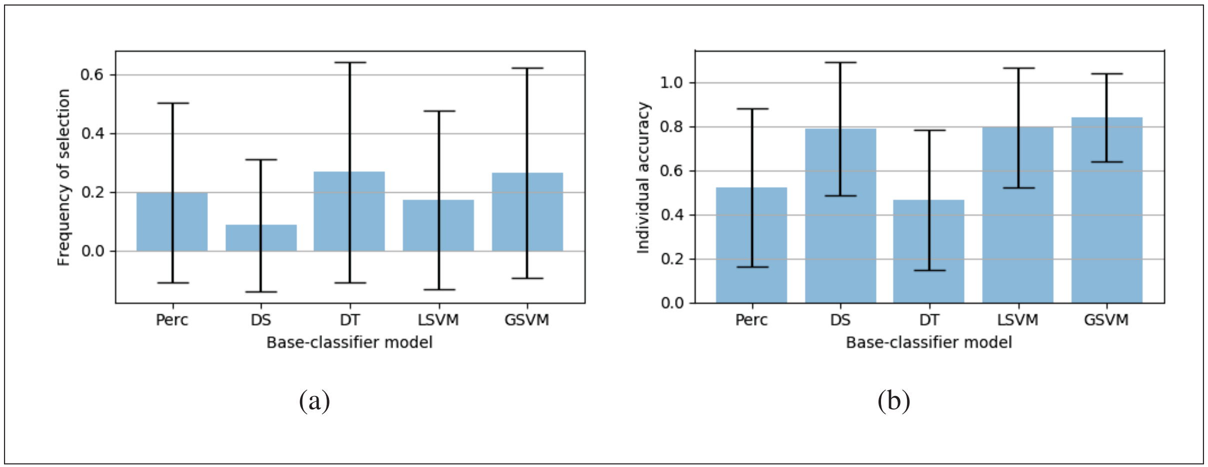


Figure-A II-9 (a) Mean frequency of selection of the base-classifier models and (b) mean individual accuracy of each component of the BR meta-classifier over all datasets from Table II-1.

Figure II-9(a) shows the mean frequency of selection for each base-classifier model by the multi-label meta-classifier in generalization. It can be observed that the DS was overall selected less often, with the most frequently selected model being the DT. The individual mean accuracy rate of each component of the BR meta-classifier is shown in Figure II-9(b). It can be observed that, while the DT was the most recommended base-classifier model, its meta-classifier yielded the poorest accuracy rate, wrongly recommending the model half of the time, on average. The recommendation of the Perceptron was also generally quite inaccurate. For the remaining models, the mean accuracy rate was around 0.8, which suggests that the meta-features used for characterizing their relevance in a subproblem are indeed important.

The performance of the multi-label meta-classifier in generalization is depicted in Figure II-10, which indicates the mean precision of the meta-classifier per dataset. It can be observed that the precision is quite high, especially for the highly imbalanced datasets (large reference number). However, for the first few datasets, the precision is quite low, reaching below 0.5 for the *glass0*, *yeast1* and *haberman* (ref. 6, 7 and 8) datasets. This may be explained by the labelset cardinality, that is, the average number of relevant classifiers per sample, which for these datasets, over which the meta-classifier yielded a poor precision score, is very low. This suggests that the multi-label meta-classifier struggles to recommend at least one relevant base-classifier model for the samples with low labelset cardinality.

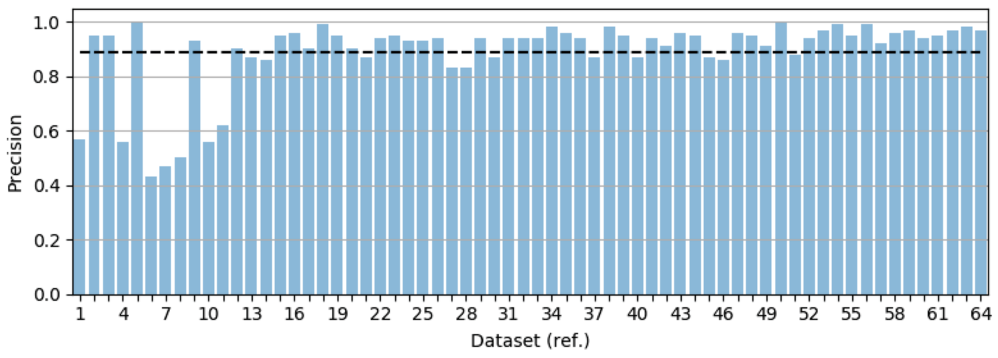


Figure-A II-10 Mean precision of meta-classifier for each dataset in Table II-1. The horizontal dashed line indicates the average performance over all datasets.

3.2.1 Framework performance

We now analyze the performance of the framework as a whole. Table II-2 shows the mean performance of the proposed framework (Proposed) and the online scheme using its original fixed individual base-classifier model (SGH+MCB) and an ideal base-classifier model for each sample. The results per dataset can be found in the Appendix. First, we can see that there is a significant improvement in selecting an ideal base-classifier model for each instance in particular in comparison to using the fixed model strategy for all instances, considering all performance metrics used in this work. Thus, we confirm one of our hypotheses: that each local region may favor certain types of classifiers and choosing the ideal one for each test sample is advantageous for local ensembles. When we analyze the performance of the proposed technique, though, we see that the selection of such ideal base-classifier model per instance is not so trivial. In terms of accuracy, the proposed technique yielded a significantly inferior performance to using the fixed model strategy. However, in terms of AUC, F-measure and G-mean, the performance was statistically similar.

Table-A II-2 Average performance of the proposed framework (Proposed) and the online scheme using the fixed individual base-classifier model (SGH+MCB) and an ideal base-classifier model for each sample. Best results excluding the ideal selector ones are in bold. The row *W-T-L* shows the number of wins, ties and losses of the proposed framework compared to using the column-wise strategy. The rows *p-value* show the result of a Wilcoxon signed rank test with $\alpha = 0.05$ between the indicated strategy (row-wise: proposed and ideal selector) and the column-wise strategy, with the symbols (+) and (-) indicating whether the former is significantly superior or inferior to the latter.

Performance metric		**Proposed**	**SGH+MCB**	**Ideal**
Accuracy	Mean	0.936	**0.941**	<u>0.971</u>
	W-T-L	n/a	14-20-30	0-5-59
	p-value (proposed)	n/a	**0.008 (-)**	**0.000 (-)**
	p-value (ideal sel.)	-	**0.000 (+)**	n/a
AUC	Mean	0.805	**0.810**	<u>0.882</u>
	W-T-L	n/a	19-12-33	0-6-58
	p-value (proposed)	n/a	0.179	**0.000 (-)**
	p-value (ideal sel.)	-	**0.000 (+)**	n/a
F-measure	Mean	0.674	**0.682**	<u>0.825</u>
	W-T-L	n/a	21-8-35	0-3-61
	p-value (proposed)	n/a	0.265	**0.000 (-)**
	p-value (ideal sel.)	-	**0.000 (+)**	n/a
G-mean	Mean	**0.740**	**0.740**	<u>0.851</u>
	W-T-L	n/a	20-12-32	0-4-60
	p-value (proposed)	n/a	0.506	**0.000 (-)**
	p-value (ideal sel.)	-	**0.000 (+)**	n/a

4. Conclusion

In this work, we presented a novel algorithm recommendation framework which dynamically suggests a set of relevant model types for each instance in particular in a problem-independent manner. Since each model learns differently from a given set of points, our recommender system makes use of meta-learning and multi-label learning for recommending the models according to the local data complexity surrounding each test sample. We then integrated the algorithm recommender system to our online local pool generation technique (Souza *et al.*, 2019b) and evaluated the proposed framework's performance over 64 binary problems.

Experiments showed that the local data characteristics affect the performance of each model type differently on a local scope. Moreover, it was shown that it is highly advantageous to use a suitable model type for each instance in particular, since the ideal model type selector yielded a statistically superior performance compared to the fixed model type approach considering all evaluated performance metrics. However, almost half of the components of our multi-label meta-classifier were not very well fit to the data, which may explain why its overall precision was high though it still struggled on harder recommendation scenarios. Overall, the performance of the proposed model type recommendation framework was statistically similar to using the original state-of-the-art online method, which uses a fixed base-classifier model for all test samples. Given the upper limit provided by the ideal selector's performance, and thus the margin for improvement of the framework, we believe this to be a promising line of research.

Since this is a novel approach to model type recommendation, in the sense that it is done dynamically according to the local structure of the data, there are many open challenges and improvements to be made. Future works may involve using a more powerful multi-label classifier that takes into account the label correlations of the meta-problem, as well as using a broader, more descriptive set of meta-features and applying a meta-feature selection procedure that is more suitable for multi-label learning.

Appendix: Detailed performance results

Table-A II-3 Mean and standard deviation of the accuracy rate of the evaluated techniques over each dataset from Table II-1. Best results are in bold.

Ref.	Proposed	SGH+MCB	Ideal	Ref.	Proposed	SGH+MCB	Ideal
1	0.76 (0.08)	**0.77** (0.07)	0.91 (0.05)	33	**0.96** (0.01)	**0.96** (0.01)	0.98 (0.01)
2	**0.98** (0.01)	0.97 (0.02)	0.99 (0.01)	34	**1.00** (0.00)	0.99 (0.02)	1.00 (0.00)
3	0.96 (0.01)	**0.97** (0.01)	0.98 (0.01)	35	**0.97** (0.02)	**0.97** (0.02)	0.99 (0.01)
4	0.73 (0.02)	**0.76** (0.04)	0.86 (0.02)	36	**0.97** (0.02)	0.96 (0.02)	0.98 (0.03)
5	**1.00** (0.00)	**1.00** (0.00)	1.00 (0.00)	37	**0.91** (0.01)	**0.91** (0.01)	0.95 (0.01)
6	0.79 (0.02)	**0.84** (0.06)	0.94 (0.04)	38	**0.99** (0.01)	**0.99** (0.00)	1.00 (0.00)
7	0.72 (0.02)	**0.75** (0.03)	0.90 (0.01)	39	**0.97** (0.01)	**0.97** (0.02)	0.98 (0.02)
8	0.69 (0.03)	**0.71** (0.05)	0.87 (0.03)	40	0.90 (0.02)	**0.91** (0.03)	0.94 (0.03)
9	**0.98** (0.01)	0.97 (0.01)	0.99 (0.00)	41	0.95 (0.02)	**0.97** (0.02)	0.98 (0.01)
10	0.75 (0.01)	**0.79** (0.02)	0.92 (0.02)	42	**0.97** (0.01)	0.95 (0.01)	0.97 (0.02)
11	0.79 (0.01)	**0.80** (0.02)	0.93 (0.01)	43	**0.99** (0.01)	**0.99** (0.02)	0.99 (0.02)
12	**0.93** (0.02)	**0.93** (0.01)	0.98 (0.01)	44	**0.98** (0.01)	0.97 (0.02)	0.98 (0.02)
13	0.95 (0.02)	**0.96** (0.02)	1.00 (0.01)	45	0.90 (0.01)	**0.92** (0.02)	0.96 (0.02)
14	0.91 (0.02)	**0.92** (0.03)	0.96 (0.02)	46	**0.91** (0.03)	0.90 (0.02)	0.95 (0.03)
15	0.98 (0.01)	**0.99** (0.01)	1.00 (0.01)	47	**0.98** (0.01)	0.97 (0.02)	0.99 (0.01)
16	0.98 (0.01)	**0.99** (0.02)	1.00 (0.00)	48	**0.98** (0.01)	0.97 (0.02)	0.99 (0.01)
17	0.95 (0.02)	**0.96** (0.03)	0.98 (0.02)	49	0.93 (0.02)	**0.94** (0.02)	0.98 (0.02)
18	**0.99** (0.00)	**0.99** (0.00)	1.00 (0.00)	50	**1.00** (0.00)	**1.00** (0.00)	1.00 (0.00)
19	**0.97** (0.01)	0.96 (0.01)	0.97 (0.02)	51	0.93 (0.01)	**0.94** (0.01)	0.96 (0.01)
20	0.93 (0.01)	**0.95** (0.01)	0.97 (0.01)	52	0.96 (0.03)	**0.97** (0.03)	0.98 (0.02)
21	0.90 (0.01)	**0.92** (0.03)	0.96 (0.02)	53	0.98 (0.01)	**0.99** (0.01)	0.99 (0.01)
22	**0.97** (0.01)	**0.97** (0.00)	0.99 (0.00)	54	**1.00** (0.00)	0.99 (0.01)	1.00 (0.00)
23	**0.97** (0.02)	**0.97** (0.03)	0.97 (0.02)	55	**0.98** (0.02)	0.97 (0.03)	0.99 (0.01)
24	0.94 (0.01)	**0.96** (0.01)	0.98 (0.02)	56	**0.99** (0.01)	**0.99** (0.02)	0.99 (0.02)
25	**0.96** (0.03)	0.94 (0.03)	0.98 (0.03)	57	0.94 (0.01)	**0.95** (0.01)	0.96 (0.01)
26	**0.97** (0.02)	**0.97** (0.03)	0.98 (0.02)	58	**0.98** (0.02)	**0.98** (0.02)	1.00 (0.01)
27	**0.91** (0.01)	0.90 (0.01)	0.95 (0.01)	59	0.97 (0.01)	**0.98** (0.01)	0.98 (0.01)
28	**0.87** (0.03)	**0.87** (0.07)	0.92 (0.03)	60	0.96 (0.01)	**0.97** (0.01)	0.98 (0.01)
29	0.96 (0.01)	**0.97** (0.01)	0.98 (0.01)	61	0.96 (0.01)	**0.97** (0.01)	0.98 (0.00)
30	**0.93** (0.01)	**0.93** (0.02)	0.96 (0.01)	62	**0.98** (0.01)	**0.98** (0.00)	0.99 (0.00)
31	0.96 (0.02)	**0.97** (0.04)	0.99 (0.02)	63	**0.99** (0.01)	**0.99** (0.01)	0.99 (0.01)
32	0.95 (0.02)	**0.97** (0.03)	0.98 (0.02)	64	**0.98** (0.01)	**0.98** (0.00)	0.99 (0.01)

Table-A II-4 Mean and standard deviation of the AUC of the evaluated techniques over each dataset from Table II-1. Best results are in bold.

Ref.	Proposed	SGH+MCB	Ideal	Ref.	Proposed	SGH+MCB	Ideal
1	0.73 (0.09)	**0.74** (0.08)	0.89 (0.06)	33	0.83 (0.09)	**0.85** (0.09)	0.90 (0.09)
2	**0.97** (0.02)	0.96 (0.03)	0.99 (0.02)	34	**1.00** (0.00)	0.95 (0.10)	1.00 (0.00)
3	0.95 (0.01)	**0.97** (0.01)	0.98 (0.01)	35	**0.89** (0.09)	**0.89** (0.09)	0.93 (0.06)
4	0.69 (0.04)	**0.74** (0.04)	0.84 (0.02)	36	**0.89** (0.11)	**0.89** (0.09)	0.92 (0.12)
5	**1.00** (0.00)	**1.00** (0.00)	1.00 (0.00)	37	**0.67** (0.08)	0.63 (0.09)	0.81 (0.05)
6	0.77 (0.03)	**0.82** (0.08)	0.94 (0.05)	38	0.95 (0.04)	**0.98** (0.01)	1.00 (0.00)
7	0.67 (0.04)	**0.69** (0.04)	0.86 (0.02)	39	**0.87** (0.07)	**0.87** (0.08)	0.92 (0.06)
8	0.57 (0.05)	**0.58** (0.05)	0.79 (0.05)	40	0.52 (0.07)	**0.53** (0.07)	0.68 (0.13)
9	**0.96** (0.01)	0.95 (0.01)	0.99 (0.01)	41	0.80 (0.05)	**0.87** (0.08)	0.90 (0.06)
10	0.66 (0.03)	**0.68** (0.01)	0.87 (0.03)	42	**0.88** (0.06)	0.78 (0.06)	0.90 (0.08)
11	0.70 (0.03)	**0.73** (0.02)	0.89 (0.02)	43	**0.95** (0.10)	**0.95** (0.10)	0.95 (0.10)
12	0.88 (0.06)	**0.90** (0.03)	0.96 (0.03)	44	**0.90** (0.09)	0.89 (0.09)	0.90 (0.09)
13	**0.94** (0.03)	**0.94** (0.02)	0.99 (0.01)	45	**0.55** (0.06)	0.54 (0.10)	0.72 (0.14)
14	**0.87** (0.04)	**0.87** (0.04)	0.94 (0.04)	46	**0.55** (0.09)	0.49 (0.01)	0.69 (0.13)
15	0.95 (0.03)	**0.97** (0.03)	0.99 (0.03)	47	**0.88** (0.07)	0.86 (0.08)	0.92 (0.04)
16	0.94 (0.05)	**0.97** (0.06)	1.00 (0.00)	48	**0.90** (0.12)	0.87 (0.15)	0.93 (0.10)
17	0.87 (0.04)	**0.90** (0.03)	0.94 (0.04)	49	0.58 (0.11)	**0.68** (0.11)	0.85 (0.13)
18	**0.99** (0.01)	0.98 (0.01)	0.99 (0.01)	50	**1.00** (0.01)	**1.00** (0.01)	1.00 (0.01)
19	**0.89** (0.07)	**0.89** (0.04)	0.91 (0.06)	51	0.62 (0.03)	**0.64** (0.06)	0.70 (0.08)
20	**0.82** (0.01)	**0.82** (0.02)	0.91 (0.03)	52	0.74 (0.13)	**0.84** (0.13)	0.88 (0.14)
21	0.73 (0.07)	**0.77** (0.10)	0.86 (0.09)	53	0.89 (0.04)	**0.90** (0.05)	0.90 (0.05)
22	**0.92** (0.01)	**0.92** (0.01)	0.96 (0.01)	54	**0.97** (0.04)	0.95 (0.07)	1.00 (0.00)
23	0.87 (0.12)	**0.89** (0.10)	0.88 (0.11)	55	**0.85** (0.20)	0.79 (0.19)	0.95 (0.10)
24	0.82 (0.03)	**0.87** (0.03)	0.92 (0.06)	56	**0.95** (0.10)	**0.95** (0.10)	0.95 (0.10)
25	**0.86** (0.17)	0.80 (0.16)	0.92 (0.12)	57	**0.51** (0.03)	0.49 (0.00)	0.55 (0.07)
26	0.85 (0.09)	**0.87** (0.12)	0.88 (0.11)	58	**0.80** (0.24)	0.75 (0.22)	0.95 (0.10)
27	0.60 (0.04)	**0.61** (0.04)	0.75 (0.05)	59	0.70 (0.05)	**0.74** (0.10)	0.77 (0.09)
28	**0.54** (0.08)	0.48 (0.03)	0.60 (0.13)	60	0.61 (0.07)	**0.63** (0.07)	0.73 (0.08)
29	0.88 (0.04)	**0.89** (0.03)	0.91 (0.03)	61	0.59 (0.03)	**0.63** (0.08)	0.65 (0.06)
30	0.72 (0.03)	**0.75** (0.04)	0.81 (0.04)	62	**0.84** (0.09)	0.74 (0.09)	0.94 (0.04)
31	0.87 (0.13)	**0.89** (0.15)	0.93 (0.10)	63	0.80 (0.19)	**0.85** (0.20)	0.85 (0.20)
32	0.80 (0.10)	**0.86** (0.14)	0.86 (0.14)	64	**0.73** (0.13)	0.67 (0.07)	0.83 (0.11)

Table-A II-5 Mean and standard deviation of the F-measure of the evaluated techniques over each dataset from Table II-1. Best results are in bold.

Ref.	Proposed	SGH+MCB	Ideal	Ref.	Proposed	SGH+MCB	Ideal
1	0.64 (0.12)	**0.66** (0.12)	0.87 (0.08)	33	0.75 (0.12)	**0.78** (0.10)	0.86 (0.11)
2	0.97 (0.03)	**0.98** (0.01)	0.99 (0.01)	34	**1.00** (0.00)	0.93 (0.13)	1.00 (0.00)
3	0.94 (0.01)	**0.96** (0.01)	0.97 (0.01)	35	**0.84** (0.15)	0.81 (0.12)	0.91 (0.07)
4	0.60 (0.06)	**0.65** (0.06)	0.79 (0.03)	36	0.80 (0.17)	**0.81** (0.13)	0.88 (0.19)
5	**1.00** (0.00)	**1.00** (0.00)	1.00 (0.00)	37	**0.42** (0.16)	0.34 (0.20)	0.71 (0.06)
6	0.69 (0.04)	**0.75** (0.11)	0.91 (0.06)	38	0.93 (0.07)	**0.97** (0.01)	1.00 (0.00)
7	0.53 (0.06)	**0.56** (0.05)	0.81 (0.03)	39	**0.83** (0.11)	**0.83** (0.11)	0.89 (0.10)
8	0.35 (0.09)	**0.36** (0.09)	0.70 (0.08)	40	**0.10** (0.20)	**0.10** (0.20)	0.48 (0.30)
9	**0.95** (0.02)	0.94 (0.02)	0.99 (0.01)	41	0.67 (0.08)	**0.81** (0.13)	0.87 (0.05)
10	0.50 (0.05)	**0.53** (0.02)	0.82 (0.05)	42	**0.79** (0.10)	0.63 (0.10)	0.80 (0.11)
11	0.56 (0.05)	**0.60** (0.03)	0.85 (0.02)	43	**0.93** (0.13)	**0.93** (0.13)	0.93 (0.13)
12	0.83 (0.07)	**0.85** (0.02)	0.95 (0.04)	44	**0.85** (0.11)	0.81 (0.12)	0.88 (0.12)
13	0.90 (0.05)	**0.92** (0.03)	0.99 (0.01)	45	**0.14** (0.17)	0.11 (0.23)	0.54 (0.31)
14	0.81 (0.05)	**0.82** (0.07)	0.92 (0.05)	46	**0.20** (0.24)	0.00 (0.00)	0.50 (0.30)
15	0.94 (0.03)	**0.96** (0.04)	0.98 (0.03)	47	**0.82** (0.11)	0.78 (0.12)	0.91 (0.04)
16	0.94 (0.06)	**0.97** (0.07)	1.00 (0.00)	48	**0.84** (0.15)	0.75 (0.21)	0.90 (0.13)
17	0.81 (0.07)	**0.85** (0.08)	0.92 (0.05)	49	0.23 (0.29)	**0.46** (0.26)	0.79 (0.19)
18	**0.98** (0.01)	**0.98** (0.01)	0.99 (0.01)	50	**1.00** (0.01)	**1.00** (0.01)	1.00 (0.01)
19	**0.86** (0.09)	0.85 (0.06)	0.88 (0.08)	51	0.32 (0.06)	**0.38** (0.13)	0.55 (0.17)
20	0.69 (0.02)	**0.73** (0.02)	0.87 (0.04)	52	0.60 (0.23)	**0.75** (0.21)	0.80 (0.20)
21	0.51 (0.10)	**0.59** (0.18)	0.79 (0.13)	53	0.80 (0.05)	**0.86** (0.08)	0.89 (0.06)
22	**0.85** (0.02)	0.84 (0.01)	0.94 (0.01)	54	**0.96** (0.04)	0.94 (0.08)	1.00 (0.00)
23	0.82 (0.17)	**0.84** (0.16)	0.84 (0.15)	55	**0.69** (0.37)	0.60 (0.37)	0.93 (0.13)
24	0.69 (0.02)	**0.79** (0.03)	0.88 (0.09)	56	**0.93** (0.13)	**0.93** (0.13)	0.93 (0.13)
25	**0.76** (0.26)	0.64 (0.21)	0.89 (0.17)	57	**0.06** (0.10)	0.00 (0.00)	0.16 (0.20)
26	0.79 (0.15)	**0.82** (0.17)	0.84 (0.15)	58	**0.60** (0.49)	0.53 (0.45)	0.93 (0.13)
27	**0.31** (0.10)	**0.31** (0.08)	0.65 (0.10)	59	0.51 (0.10)	**0.65** (0.17)	0.67 (0.15)
28	**0.15** (0.18)	0.00 (0.00)	0.26 (0.33)	60	0.27 (0.15)	**0.36** (0.16)	0.61 (0.16)
29	0.79 (0.05)	**0.83** (0.06)	0.89 (0.05)	61	0.25 (0.06)	**0.34** (0.20)	0.45 (0.15)
30	0.55 (0.07)	**0.60** (0.08)	0.74 (0.05)	62	**0.69** (0.15)	0.57 (0.14)	0.90 (0.04)
31	0.76 (0.20)	**0.82** (0.25)	0.90 (0.13)	63	**0.67** (0.37)	0.63 (0.37)	0.73 (0.39)
32	0.67 (0.18)	**0.77** (0.23)	0.81 (0.22)	64	**0.51** (0.26)	0.45 (0.14)	0.76 (0.15)

Table-A II-6 Mean and standard deviation of the geometric mean of the evaluated techniques over each dataset from Table II-1. Best results are in bold.

Ref.	**Proposed**	**SGH+MCB**	**Ideal**	**Ref.**	**Proposed**	**SGH+MCB**	**Ideal**
1	0.72 (0.10)	**0.73** (0.09)	0.89 (0.07)	33	0.81 (0.11)	**0.83** (0.11)	0.89 (0.11)
2	**0.97** (0.03)	0.96 (0.03)	0.99 (0.02)	34	**1.00** (0.00)	0.94 (0.12)	1.00 (0.00)
3	0.95 (0.01)	**0.97** (0.01)	0.98 (0.01)	35	**0.88** (0.11)	**0.88** (0.11)	0.92 (0.07)
4	0.68 (0.05)	**0.73** (0.05)	0.83 (0.02)	36	**0.88** (0.13)	**0.88** (0.10)	0.90 (0.15)
5	**1.00** (0.00)	**1.00** (0.00)	1.00 (0.00)	37	**0.58** (0.15)	0.45 (0.25)	0.78 (0.06)
6	0.77 (0.04)	**0.81** (0.09)	0.94 (0.05)	38	0.95 (0.04)	**0.98** (0.01)	1.00 (0.00)
7	0.65 (0.05)	**0.67** (0.04)	0.85 (0.02)	39	**0.86** (0.09)	**0.86** (0.09)	0.92 (0.07)
8	**0.51** (0.07)	**0.51** (0.07)	0.76 (0.07)	40	**0.12** (0.23)	**0.12** (0.23)	0.53 (0.30)
9	**0.96** (0.01)	0.95 (0.01)	0.99 (0.01)	41	0.78 (0.07)	**0.86** (0.08)	0.89 (0.07)
10	0.63 (0.04)	**0.64** (0.01)	0.86 (0.03)	42	**0.87** (0.07)	0.74 (0.08)	0.89 (0.09)
11	0.68 (0.05)	**0.72** (0.02)	0.89 (0.02)	43	**0.94** (0.12)	**0.94** (0.12)	0.94 (0.12)
12	0.87 (0.07)	**0.90** (0.03)	0.96 (0.04)	44	**0.89** (0.11)	0.88 (0.11)	0.89 (0.11)
13	0.93 (0.03)	**0.94** (0.02)	0.99 (0.01)	45	**0.21** (0.26)	0.14 (0.28)	0.58 (0.32)
14	**0.87** (0.04)	**0.87** (0.04)	0.94 (0.04)	46	**0.23** (0.28)	0.00 (0.00)	0.54 (0.30)
15	0.95 (0.03)	**0.97** (0.03)	0.99 (0.03)	47	**0.86** (0.09)	0.84 (0.09)	0.92 (0.04)
16	0.94 (0.06)	**0.97** (0.06)	1.00 (0.00)	48	**0.88** (0.14)	0.84 (0.20)	0.91 (0.12)
17	0.86 (0.04)	**0.89** (0.04)	0.94 (0.04)	49	0.26 (0.32)	**0.53** (0.28)	0.82 (0.17)
18	**0.99** (0.01)	0.98 (0.01)	0.99 (0.01)	50	**1.00** (0.01)	**1.00** (0.01)	1.00 (0.01)
19	**0.88** (0.08)	**0.88** (0.05)	0.90 (0.07)	51	0.50 (0.08)	**0.53** (0.11)	0.62 (0.14)
20	**0.81** (0.01)	0.80 (0.02)	0.90 (0.04)	52	0.68 (0.17)	**0.82** (0.17)	0.85 (0.18)
21	0.69 (0.10)	**0.73** (0.14)	0.85 (0.11)	53	**0.89** (0.05)	**0.89** (0.05)	0.89 (0.05)
22	0.91 (0.01)	**0.92** (0.01)	0.96 (0.01)	54	**0.97** (0.04)	0.94 (0.07)	1.00 (0.00)
23	0.85 (0.13)	**0.88** (0.11)	0.86 (0.13)	55	**0.74** (0.39)	0.68 (0.37)	0.94 (0.12)
24	0.80 (0.05)	**0.86** (0.04)	0.91 (0.07)	56	**0.94** (0.12)	**0.94** (0.12)	0.94 (0.12)
25	**0.81** (0.23)	0.75 (0.21)	0.91 (0.14)	57	**0.08** (0.16)	0.00 (0.00)	0.20 (0.25)
26	0.83 (0.11)	**0.85** (0.13)	0.86 (0.13)	58	**0.60** (0.49)	0.54 (0.45)	0.94 (0.12)
27	0.45 (0.10)	**0.47** (0.10)	0.70 (0.08)	59	0.62 (0.10)	**0.70** (0.14)	0.73 (0.13)
28	**0.23** (0.28)	0.00 (0.00)	0.28 (0.35)	60	0.46 (0.14)	**0.49** (0.13)	0.66 (0.12)
29	0.87 (0.05)	**0.88** (0.04)	0.90 (0.04)	61	0.44 (0.07)	**0.45** (0.25)	0.54 (0.11)
30	0.66 (0.05)	**0.71** (0.05)	0.79 (0.05)	62	**0.81** (0.11)	0.69 (0.13)	0.94 (0.04)
31	0.84 (0.18)	**0.87** (0.19)	0.91 (0.12)	63	0.68 (0.37)	**0.74** (0.39)	0.74 (0.39)
32	0.76 (0.14)	**0.83** (0.19)	0.83 (0.19)	64	**0.64** (0.22)	0.57 (0.13)	0.80 (0.13)

APPENDIX III

LOCAL OVERLAP REDUCTION PROCEDURE FOR DYNAMIC ENSEMBLE SELECTION

Mariana A. Souza[1] , Robert Sabourin[1] , George D. C. Cavalcanti[2] , Rafael M. O. Cruz[1]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA),
École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Centro de Informática,
Universidade Federal de Pernambuco,
1235 Av. Prof. Moraes Rego, Recife, Pernambuco, Brazil 50670-901

Article published in the Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2022

Abstract

Class imbalance is a characteristic known for making learning more challenging for classification models as they may end up biased towards the majority class. A promising approach among the ensemble-based methods in the context of imbalance learning is Dynamic Selection (DS). DS techniques single out a subset of the classifiers in the ensemble to label each given unknown sample according to their estimated competence in the area surrounding the query. Because only a small region is taken into account in the selection scheme, the global class disproportion may have less impact over the system's performance. However, the presence of local class overlap may severely hinder the DS techniques' performance over imbalanced distributions as it not only exacerbates the effects of the under-representation but also introduces ambiguous and possibly unreliable samples to the competence estimation process. Thus, in this work, we propose a DS technique which attempts to minimize the effects of the local class overlap during the classifier selection procedure. The proposed method iteratively removes from the target region the instance perceived as the hardest to classify until a classifier is deemed competent to label the query sample. The known samples are characterized using instance hardness measures that quantify the local class overlap. Experimental results show that the proposed technique can significantly outperform the baseline as well as several other DS techniques, suggesting its suitability for dealing with class under-representation and overlap. Furthermore, the proposed technique still yielded competitive results when using an under-sampled, less overlapped version of the labelled sets, specially over the problems with a high proportion of minority class samples in overlap areas. Code available at https://github.com/marianaasouza/lords.

1. Introduction

Imbalanced classification problems are characterized by a disproportion in the number of instances from the problems' classes. Because one class is underrepresented compared to the remaining one(s), some traditional classification models may become biased towards the more well-represented labels (Prati *et al.*, 2015). This bias may hinder the performance over the rarer label, which is often also the most relevant class in real-world imbalanced problems, such as fraudulent transactions detection (Wei *et al.*, 2013) and biomedical diagnosis (Mazurowski *et al.*, 2008).

Methods specifically tailored to deal with imbalanced datasets may fall into one of the following four categories (Fernández *et al.*, 2018): algorithm-level approaches, which adapt traditional classifiers to deal with the class disproportion; data-level approaches, which resample the data to balance the distribution; cost-sensitive learning frameworks, which apply and incorporate class-based costs to the learning procedure; and ensemble based approaches, which usually couple ensemble methods with other methods, most commonly data-level and cost-sensitive approaches.

Among the ensemble-based approaches, dynamic selection (DS) schemes were often shown to perform rather well over imbalanced distributions (Oliveira *et al.*, 2017), specially in combination with pre-processing methods (Roy *et al.*, 2018). DS techniques select a subset of the classifiers in the ensemble for labelling each query sample in particular, with the aim of using only the members that are perceived as competent in the area where the target instance is located. Their local approach in generalization can be an advantage in imbalanced scenarios as the global class disproportion may have a limited impact over the performance, similarly to other local methods (García, Mollineda & Sánchez, 2008).

Also due to their local approach, however, the presence of local class overlap can degrade their recognition rates not only over the positive (minority) class but also the negative (majority) class (Prati, Batista & Monard, 2004; García, Sánchez & Mollineda, 2007). In fact, the use of local adaptive distance and/or prototype selection methods that aim at minimizing the class overlap in the target region, called the Region of Competence (RoC) in the DS literature, were shown to improve their performance, including over highly imbalanced distributions (Cruz *et al.*, 2019a; Souza *et al.*, 2019a).

Thus, in this work, we propose a dynamic selection technique which dynamically edits the target region taking into account the instances' characteristics with respect to (w.r.t.) the class overlap surrounding them. Based on the K-Nearest-Oracles Eliminate (KNORA-E) (Ko *et al.*, 2007), the proposed method also searches for local oracles, that is, classifiers in the pool that can correctly label all instances in the RoC. Until a competent model is found, the region is iteratively reduced by removing from it the sample with the highest estimated classification difficulty. Thus, the instances considered relatively more unreliable due to their ambiguity are increasingly disregarded for estimating the competence of the classifiers.

The contributions of this work are then:

- A novel DS technique which integrates instance characterization, including two proposed adaptations for imbalanced distributions, into the definition of the RoC for dynamically reducing the class overlap in it.
- An experimental analysis over 64 imbalanced datasets in which we assess the performance of the proposed technique against 11 DS techniques, as well as the impact of the proposed dynamic overlap-reducing scheme compared to using a pre-processing technique with the same goal.

This work is organized as follows. Section 2 presents a brief background on DS techniques. Then, in Section 3 we lay out the problem statement. The proposed method is introduced in

Section 4. The experimental analysis is presented in Section 5. Lastly, we summarize our conclusions in Section 6.

2. Background

Dynamic selection techniques select a subset of a pool of base-classifiers to label each given query sample according to their perceived competence in the task, estimated over a region around the target instance. DS techniques are usually performed in three steps: RoC definition, competence estimation and classifier selection (Cruz *et al.*, 2018a). In the first step, the RoC is defined over a labelled set, called the *DSEL* set, using the nearest neighbors rule (Ko *et al.*, 2007), clustering methods (Soares *et al.*, 2006), distance-based potential functions (Woloszynski & Kurzynski, 2011), among others. Then, the competence of the classifiers in the pool is estimated using the samples in the RoC according to some criteria, for instance local accuracy (Ko *et al.*, 2007). Lastly, the classifier(s) deemed competent is(are) selected to label the query. If only one the most competent one is selected, the method is a Dynamic Classifier Selection (DCS) technique while if more than one can be selected the method is a Dynamic Ensemble Selection (DES) technique, which requires a combination scheme to join the selected classifiers' responses.

It has been observed in the literature that the RoC definition has a large impact over the performance of the DS techniques (Cruz *et al.*, 2018b), as the classifiers' competence is estimated as a function of the instances included in it. In fact, a few DS techniques present a RoC editing scheme with the purpose of improving the classifiers' competence estimation. The Multiple Classifier Behavior (Giacinto *et al.*, 2000) uses only the subset of instances from the RoC that have a similar output profile, that is, the aggregate of classifiers' responses, as the query's. In (Pereira *et al.*, 2018), Item Response Theory (IRT) is applied over the ensemble to filter out from a larger RoC the samples with low discrimination index. Lastly, the KNORA-E (and several methods based on it (Oliveira *et al.*, 2018)) also removes instances from the RoC, but in an iterative way. We describe the KNORA-E technique next.

3. Problem statement

The KNORA-E technique attempts to find in the pool a classifier that can correctly label all instances in the RoC, referred to as a local oracle. Fig. III-1 shows a toy example which we use to illustrate the technique. It depicts the initial RoC $\theta_q = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_7\}$ obtained using the K-Nearest Neighbors (KNN), the query sample $\mathbf{x}_q$ and the pool of linear classifiers $C = \{c_1, c_2\}$. The larger the index i in $\mathbf{x}_i$, the furthest from the query the sample $\mathbf{x}_i$ is. The KNORA-E tries to find a classifier in C that can label all $\mathbf{x}_i \in \theta_q$ correctly. If there is none, it removes from θ_q the instance that is the most distant from the query $\mathbf{x}_q$, and repeats the search for a local oracle. This process is repeated until at least one classifier is found to correctly label all remaining instances in the RoC. In the example from Fig. III-1, the RoC is edited until the removal of $\mathbf{x}_5$, as with $\theta_q = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_4\}$ the classifier c_2 would be considered a local oracle and selected to label $\mathbf{x}_q$.

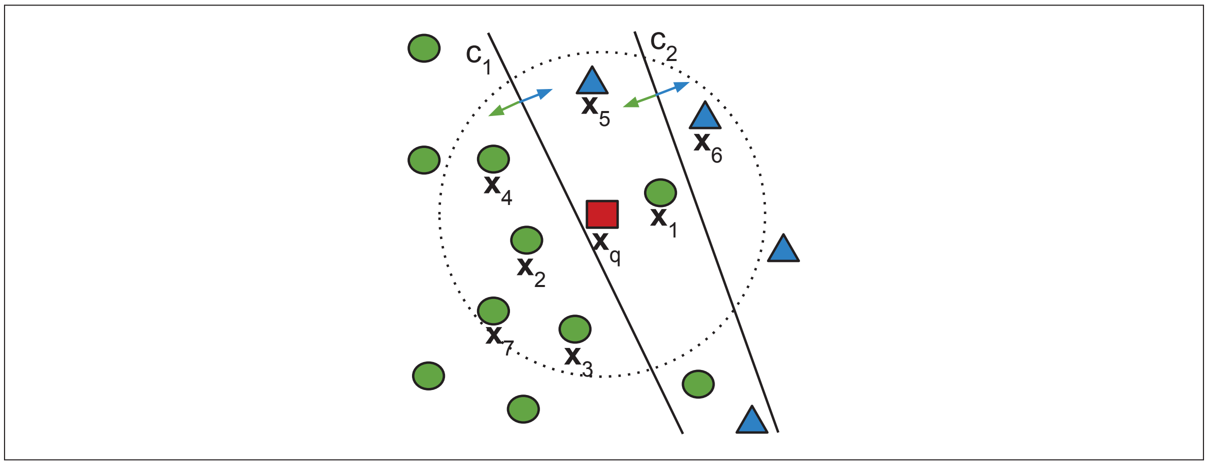


Figure-A III-1 Illustrative example. $\mathbf{x}_q$ is a query instance that belongs to the locally underrepresented blue class. The dashed line delimits its RoC $\theta_q = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_7\}$, with the distances to the query $d(\mathbf{x}_q, \mathbf{x}_1) < d(\mathbf{x}_q, \mathbf{x}_2) < d(\mathbf{x}_q, \mathbf{x}_3) < d(\mathbf{x}_q, \mathbf{x}_4) < d(\mathbf{x}_q, \mathbf{x}_5) < d(\mathbf{x}_q, \mathbf{x}_6) < d(\mathbf{x}_q, \mathbf{x}_7)$. The Perceptrons c_1 and c_2 form the pool of classifiers and label blue to their right and green to their left, as indicated by the arrows.

As shown in this example, the RoC editing procedure from the KNORA-E may lead to the entire elimination of one of the classes in the region, which can bias the selection towards the remaining class, often the less sparse or under-represented one. Two methods based on the KNORA-E, the K-Nearest Oracles-Borderline (KNORA-B) and K-Nearest Oracles-Borderline-Imbalanced (KNORA-BI) propose to fix this issue by forcing the presence of all classes in the RoC. However, the three methods still assume the samples that are the closest to the query are more relevant for estimating the classifiers' competences. This may not be true if the RoC presents a certain degree of class overlap, as in the example from Fig. III-1. In fact, by disregarding the query's closest neighbor ($\mathbf{x}_1$), which is located in the most overlapped area of the RoC, the classifier c_1 would be recognized as a local oracle and would correctly label $\mathbf{x}_q$.

4. Proposed technique

We propose a dynamic selection technique based on the KNORA-E which performs the RoC editing based on the samples' estimated classification difficulty. The classification difficulty is estimated using an instance hardness measure (Smith *et al.*, 2014; Arruda, Prudêncio & Lorena, 2020) which attempts to capture the degree of class overlap where the instance is located. That is, for a given unknown sample, we remove from the RoC the neighbors with higher instance hardness first, instead of the ones with largest distance to the query, in the search for the nearest local oracles. That way, the samples which are perceived as more reliable (that is, that are easier to classify) have a larger impact on the classifiers' competence estimation compared to the more unreliable ones in the RoC. Thus, the proposed neighborhood editing procedure functions similarly to a dynamic instance selection over the RoC in which the samples with higher class ambiguity are sequentially removed until a competent classifier is found.

Algorithms III-1 and III-2 present in more detail the proposed dynamic selection scheme. The instance hardness estimation step described in Algorithm III-1 occurs in memorization, while the selection of the ensemble of classifiers (Algorithm III-2) happens in generalization.

4.1 Instance hardness estimation

With regards to the hardness estimation procedure, Algorithm III-1 requires the DSEL set and returns the hardness estimates of each sample in the DSEL. The hardness estimate consists of the score obtained from an instance hardness measure computed over the DSEL set. Thus, from Line 1 to Line 4, the instance hardness of each sample $\mathbf{x}_i$ in the DSEL ($\mathcal{V}$) is estimated and stored. The instance hardness measures that are computed in $estimate\_hardness()$ are explained next. The hardness estimates of all instances in the DSEL are then returned in Line 5.

Algorithm-A III-1 Instance hardness estimation.

```
  input  : $\mathcal{V} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$ ;          ▹ DSEL set
  output : $H = \{h_1, h_2, ..., h_N\}$ ;                                   ▹ Hardness estimates
1 for every $(\mathbf{x}_i, y_i)$ in $\mathcal{V}$ do
2     $h_i \leftarrow estimate\_hardness((\mathbf{x}_i, y_i), \mathcal{V})$ ;            ▹ Estimate instance hardness
3     $H \leftarrow H \cup h_i$ return $H$
4 end for
```

In order to characterize the hardness of the samples in the DSEL, we make use of instance hardness measures found in the literature. We have selected two measures to investigate in this work: the K-Disagreeing Neighbors (KDN) (Smith *et al.*, 2014) and the Local Set Cardinality (Arruda *et al.*, 2020). We chose these measures because they attempt to convey the classification difficulty associated with the local class overlap of the region where the sample is located. Since the dynamic selection technique is based on the concept of local oracles, the information regarding how ambiguous that region is may be valuable to assess the instance's reliability for the competence estimation step. Moreover, local class overlap was shown to correlate the most with classification difficulty on the instance level in (Smith *et al.*, 2014). Lastly, since both measures are based on sample counts, their adaptation for class imbalanced scenarios can be quite straightforward.

We describe next the two chosen measures, as well as their proposed adaptation for imbalanced datasets. We also illustrate the hardness estimation using the example from Fig. III-1, and to calculate the scores of the instances within the RoC we assume the total number of samples in the DSEL set is $|\mathcal{V}| = 100$ and its imbalance ratio (IR), that is, the ratio between the majority and minority class sizes, is $IR = 3.0$.

4.1.1 K-Disagreeing Neighbors (KDN)

The KDN score indicates the proportion of neighbors of a given sample which belong to a different class. The computation of the KDN measure is shown in (A III-1), in which the k-neighborhood θ_i of the target instance $\mathbf{x}_i$ is obtained over the remaining samples in the labelled set $\mathcal{V}$. Fig. III-2a shows the KDN scores, with $k = 3$ and rounded to two decimal points, of the instances from the toy example.

$$KDN((\mathbf{x}_i, y_i), \mathcal{V}, k) = \frac{|\{(\mathbf{x}_j, y_j) \in \theta_i : y_i \neq y_j\}|}{k} \tag{A III-1}$$

Since the KDN measure disregards the differences between the classes' frequencies, in highly imbalanced scenarios the scores of the positive instances in overlap regions may be estimated much higher than the negative samples. This could negatively impact their recognition rates, as they would be removed first in the RoC editing procedure of the proposed technique. Thus, we propose and evaluate an adaptation of the KDN measure which attempts to even out the scores according to the classes' frequencies. The adaptation, K-Disagreeing Neighbors-imbalance (KDNi) is shown in (A III-2). It consists of the regular KDN score of the sample, according to (A III-1), divided by the proportion of samples in the dataset that belong to the opposite class of the target instance (p_o). Since the KDN score includes the value 0.0, we add a very small number $e = 10^{-3}$ to it before computing the KDNi. Moreover, as the KDNi score can reach quite large values due to the division, we apply the function $f(x) = 1 - 1/(1 + x)$ to bound the base score from (A III-2) and keep it in the range (0.0, 1.0), with the higher values for harder instances and lower values for easier samples.

$$KDNi((\mathbf{x}_i, y_i), \mathcal{V}, k) = \frac{KDN((\mathbf{x}_i, y_i), \mathcal{V}, k)}{p_o} \tag{A III-2}$$

The KDNi scores of the example from Fig. III-1 are shown in Fig. III-2b. We can see, comparing the KDNi scores to the original KDN scores, that the instances very close to the border had their estimated hardness changed, while the ones further from the border changed little to nothing.

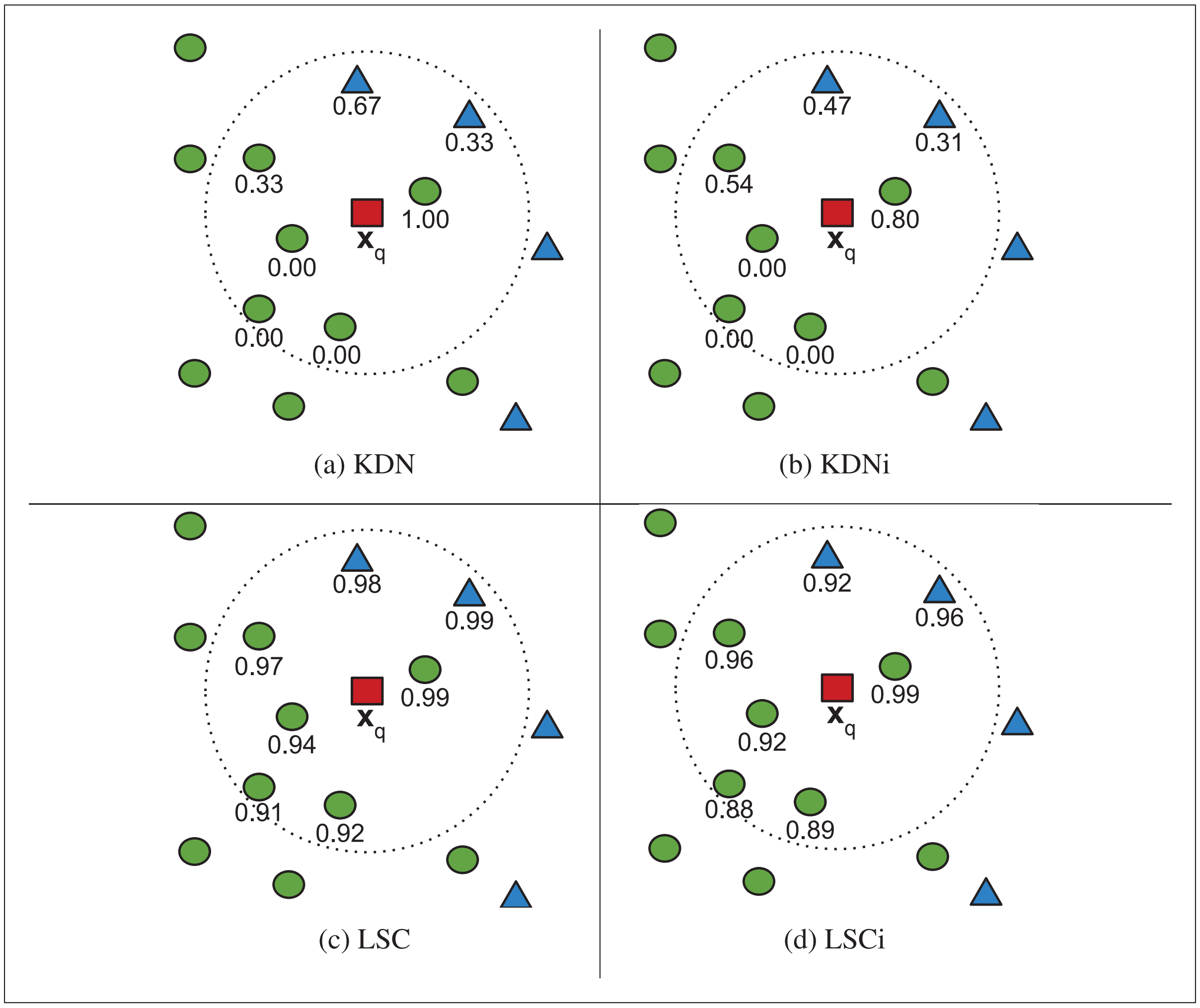


Figure-A III-2 Rounded instance hardness estimates of the samples from Fig. III-1, according to each indicated measure.

4.1.2 Local Set Cardinality (LSC)

The LSC is an instance hardness measure based on the Local Set (LS) concept (Leyva, González & Perez, 2014). The LS of a given sample is comprised of the instances whose distance to it is smaller than the distance between the target sample and its nearest enemy, that is, the closest instance from the opposite class. The LSC measure of an instance is then the cardinality of its LS averaged by the number of samples in the dataset, as shown in (A III-3), where $d()$ is the Euclidean distance and $\mathbf{x}_{ne}$ is the nearest enemy of $\mathbf{x}_i$. In order to yield a higher score to a harder to classify instance, we do as in (Lorena *et al.*, 2019) and calculate the measure as one minus the score shown in (A III-3). The LSC scores of the instances from Fig. III-1 are shown in Fig. III-2c.

$$LSC((\mathbf{x}_i, y_i), \mathcal{V}) = \frac{|\{(\mathbf{x}_j, y_j) \in \mathcal{V} : d(\mathbf{x}_i, \mathbf{x}_j) < d(\mathbf{x}_i, \mathbf{x}_{ne})\}|}{|\mathcal{V}|} \tag{A III-3}$$

As the LSC measure also does not take into account the disproportion between the classes' sizes, we adapt it to imbalanced distributions. The Local Set Cardinality-imbalance is a straightforward adaptation in which instead of dividing the size of the local set by the total number of instances in the dataset $|\mathcal{V}|$, we divide it by the number of samples that share the same label as the target instance, as shown in (A III-4). As in the LSC score used in this work, we also compute the LSCi as one minus the score shown in the equation for it to have a higher value for harder samples.

$$LSCi((\mathbf{x}_i, y_i), \mathcal{V}) = \frac{|\{(\mathbf{x}_j, y_j) \in \mathcal{V} : d(\mathbf{x}_i, \mathbf{x}_j) < d(\mathbf{x}_i, \mathbf{x}_{ne})\}|}{|\{(\mathbf{x}_j, y_j) \in \mathcal{V} : y_i = y_j)\}|} \tag{A III-4}$$

Fig. III-2d shows the computed scores of the LSCi. We can see that the adaptation for imbalanced scenarios with this measure affected more instances than in the KDN adaptation case.

4.2 Ensemble selection

Algorithm III-2 describes the dynamic ensemble of classifiers selection procedure. It takes as input the query sample $\mathbf{x}_q$, the region of competence (RoC) size k, the DSEL set, the pool of classifiers C and the hardness estimates H obtained in memorization, and returns the selected ensemble of classifiers $C' \subseteq C$. First, the RoC θ_q of size k is obtained using the nearest neighbors rule in Line 1. Then, in Line 2, the subset of classifiers in C which correctly label all instances in the RoC are singled out. If there are local oracles with regards to the original RoC θ_q, they are returned in Line 9. Otherwise, the procedure enters the loop from Line 4 to Line 8 until there can be found a classifier able to correctly label all instances in the RoC. In Line 5, the hardness estimates of the instances in the current RoC θ'_q, which starts as the original

θ_q, are obtained. The size of the current RoC θ'_q is then reduced by one in Line 6 by removing from it the instance with highest hardness estimate. If there are more than one instance with the maximum hardness score in the current RoC, the furthest one from the query sample is removed. Then, we update the subset C' with the classifiers in C which have 100% accuracy over the reduced RoC in Line 7. If at least one classifier is in C', then the loop is exited and the procedure returns the ensemble C', which is aggregated using majority voting to produce the predicted label of the query. However, if no local oracle is found at that iteration, the RoC editing process is repeated with the current RoC θ'_q.

Algorithm-A III-2 Ensemble of classifiers selection.

```
   input  : $\mathbf{x}_q, k$ ;                                                   ▹ Query instance, RoC size
   input  : $\mathcal{V} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$ ;  ▹ DSEL set
   input  : $C = \{c_1, c_2, ..., c_M\}$ ;                                        ▹ Pool of classifiers
   input  : $H = \{h_1, h_2, ..., h_N\}$ ;                                        ▹ Hardness estimates
   output : $C'$ ;                                                                ▹ Selected ensemble of classifiers
1  $\theta_q \leftarrow \{(\mathbf{x}_i, y_i) \in KNN(\mathbf{x}_q, \mathcal{V}, k)\}$ ;                 ▹ Obtain RoC
2  $C' \leftarrow \{c_j \in C : \forall(\mathbf{x}_i, y_i) \in \theta_q, c_j(\mathbf{x}_i) = y_i\}$ ;     ▹ Select local oracles
3  $\theta'_q \leftarrow \theta_q$ ;                                                         ▹ Current RoC
4  while $C' = \emptyset$ do
5      $H' \leftarrow \{h_i \in H : (\mathbf{x}_i, y_i) \in \theta'_q\}$ ;                      ▹ Obtain hardness of current RoC
6      $\theta'_q \leftarrow \theta'_q - \{\mathbf{x}_i : h_i = max(H')\}$;                   ▹ Remove hardest instance
7      $C' \leftarrow \{c_j \in C : \forall(\mathbf{x}_i, y_i) \in \theta'_q, c_j(\mathbf{x}_i) = y_i\}$ ; ▹ Select local oracles
8  end while
9  return $C'$
```

Now, going back to the example from Fig. III-1, we can see that based on the neighboring samples' instance hardness scores obtained in memorization (Fig. III-2), the proposed technique would remove the sample $\mathbf{x}_1$ from the RoC (Line 6) in the first iteration of the loop from Line 4 to Line 8, if using the KDN, KDNi or LSCi measures. In this case, the edited RoC θ' obtained after the first RoC edit is shown in Fig. III-3a. If using the LSC measure, the first sample to be removed from the original RoC would be the $\mathbf{x}_6$, as it presents the same (highest) hardness estimate as $\mathbf{x}_1$ but it is further away from the query. In the second iteration, however, $\mathbf{x}_1$ would be removed as well based on the LSC scores of the remaining instances in the RoC. The RoC configuration obtained after the removal of $\mathbf{x}_1$ in the case of using the LSC measure in shown in Fig. III-3b. Using any of the presented instance hardness measures, we can see from Fig. III-3 that, after the removal of $\mathbf{x}_1$, the method reaches its stop criteria as in Line 7 the classifier c_1 is identified as a local oracle in C, since it labels all instances in the current RoC correctly. Thus, the ensemble C' which is returned in Line 9 contains the classifier c_1, which is then used to label the query instance.

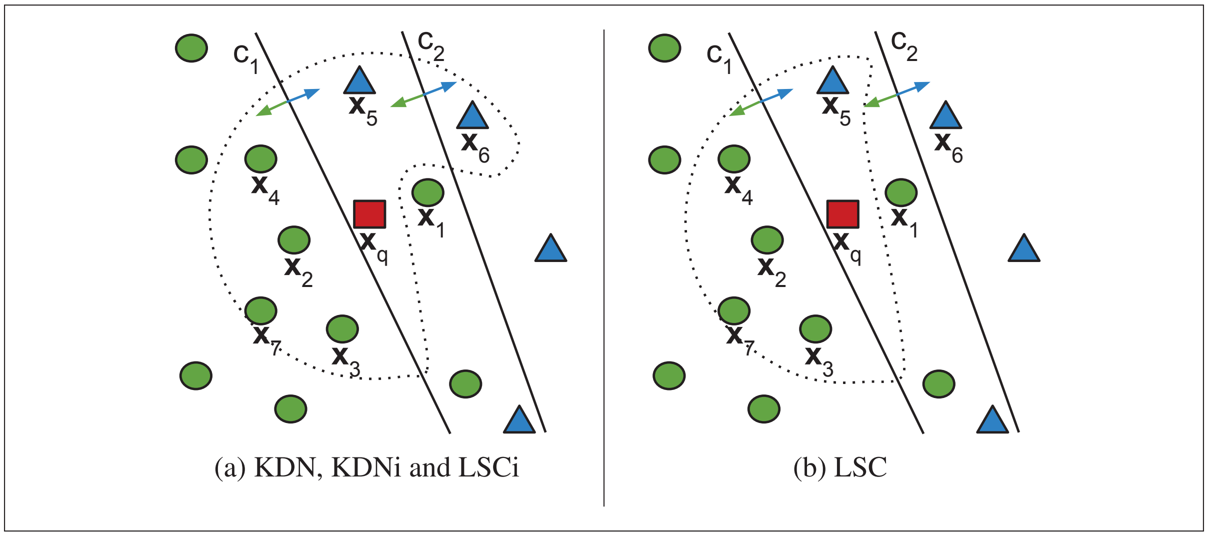


Figure-A III-3 Example from Fig. III-1 after (a) one and (b) two iteration(s) of the RoC editing loop (Line 4 to Line 8 of Algorithm III-2) using the indicated measures. The dotted area indicates the current RoC configuration at the time where a local oracle (c_1) is finally found, right after the sample $\mathbf{x}_1$ is removed.

5. Experiments

5.1 Experimental protocol

5.1.1 Datasets

In order to evaluate the performance of the proposed technique over class imbalanced scenarios, we use the 64 two-class datasets from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository (Alcalá *et al.*, 2011) presented in Table III-1. The datasets present an IR ranging from 1.82 to 41.40. Moreover, we obtained for each dataset the proportion of safe instances according to the categorization proposed in (Napierala & Stefanowski, 2016) which is based on the local class distribution of the positive class samples. A minority class sample is considered safe if at least 4 of its 5 neighbors also belong to the minority class. The datasets in Table III-1 are sorted in descending order of S (%), that is, the percentage of safe minority class instances. We refer to the first 36 problems in Table III-1 as safe datasets, since at least half of its positive class samples are safe ($S(\%) \geq 50$), and the remaining problems as unsafe, as in (García, Marqués & Sánchez, 2019).

For reproducibility, each dataset was evaluated using a stratified 5-fold cross validation procedure, one fold for test and the remaining for training, using the same partitions provided in the KEEL website. Due to the small-sized datasets and their high imbalance ratio, we use the training set as the DSEL set for all dynamic selection techniques evaluated, as in (Cruz *et al.*, 2019a; Souza *et al.*, 2019a).

5.1.2 Performance measures

We evaluate the models in this work in terms of two frequently used measures for imbalance learning: the F_1 score (van Rijsbergen, 1979) and the Geometric Mean (G-mean) (Kubat *et al.*, 1997). For the statistical comparisons between the techniques' performances over multiple datasets, we use the pairwise Wilcoxon signed-rank test, as recommended in (Demšar, 2006).

5.1.3 Dynamic Selection techniques

We compare our proposed technique against 11 DES techniques, including the baseline KNORA-E. The chosen techniques are shown in Table III-2. We use their implementation from the Python library DESLib (Cruz *et al.*, 2020), and evaluate them with the RoC size $k = 7$, and their remaining hyperparameters as the default. W.r.t. the proposed method, we use the same RoC size, and evaluate it using each of the hardness measures presented in Section 4.1. The only extra hyperparameter necessary is the neighborhood size of the KDN (as in (A III-1) and (A III-2)), which we set to $k = 5$ as recommended in (Smith *et al.*, 2014).

Table-A III-1 Main characteristics of the datasets used in the experiments. Ref. is the reference number used in this work for each dataset in the table. I and F refer to the number of instances and features, respectively. IR refers to the imbalance ratio, and S indicates the percentage of safe minority class instances in the whole dataset.

Ref.	Dataset	I	F	IR	S	Ref.	Dataset	I	F	IR	S
1	ecoli-0_vs_1	220	7	1.86	99.30	33	yeast-2_vs_8	482	8	23.10	55.00
2	shuttle-c0-vs-c4	1829	9	13.87	99.19	34	yeast-2_vs_4	514	8	9.08	54.90
3	vowel0	988	13	9.98	98.89	35	led7digit-0-2-4-5-6-7-8-9_vs_1	443	7	10.97	51.35
4	iris0	150	4	2.00	98.00	36	shuttle-c2-vs-c4	129	9	20.50	50.00
5	segment0	2308	19	6.02	96.35	37	glass1	214	9	1.82	48.68
6	wisconsin	683	9	1.86	91.21	38	ecoli-0-2-6-7_vs_3-5	224	7	9.18	45.45
7	vehicle2	846	18	2.88	89.45	39	glass-0-1-6_vs_5	184	9	19.44	44.44
8	page-blocks-1-3_vs_4	472	10	15.86	82.14	40	glass-0-4_vs_5	92	9	9.22	44.44
9	ecoli2	336	7	5.46	76.92	41	ecoli-0-6-7_vs_3-5	222	7	9.09	40.91
10	vehicle0	846	18	3.25	75.38	42	ecoli-0-6-7_vs_5	220	6	10.00	40.00
11	ecoli-0-3-4-6_vs_5	205	7	9.25	75.00	43	yeast6	1484	8	41.40	37.14
12	ecoli-0-4-6_vs_5	203	6	9.15	75.00	44	yeast-0-2-5-6_vs_3-7-8-9	1004	8	9.14	34.34
13	ecoli-0-3-4_vs_5	200	7	9.00	75.00	45	yeast5	1484	8	32.73	34.09
14	newthyroid2	215	5	5.14	74.29	46	ecoli3	336	7	8.60	31.43
15	glass6	214	9	6.38	72.41	47	glass4	214	9	15.46	30.77
16	ecoli-0-1-4-7_vs_5-6	332	6	12.28	72.00	48	pima	768	8	1.87	28.73
17	ecoli-0-1-3-7_vs_2-6	281	7	39.14	71.43	49	vehicle1	846	18	2.90	23.04
18	page-blocks0	5472	10	8.79	70.30	50	glass5	214	9	22.78	22.22
19	ecoli-0-2-3-4_vs_5	202	7	9.10	70.00	51	yeast1	1484	8	2.46	22.14
20	ecoli4	336	7	15.80	70.00	52	vehicle3	846	18	2.99	17.45
21	ecoli-0-1-4-6_vs_5	280	6	13.00	70.00	53	yeast-0-3-5-9_vs_7-8	506	8	9.12	16.00
22	ecoli-0-1_vs_5	240	6	11.00	70.00	54	cleveland-0_vs_4	173	13	12.31	15.38
23	new-thyroid1	215	5	5.14	68.57	55	yeast-0-5-6-7-9_vs_4	528	8	9.35	7.84
24	yeast-0-2-5-7-9_vs_3-6-8	1004	8	9.14	67.68	56	yeast4	1484	8	28.10	7.84
25	glass-0-1-2-3_vs_4-5-6	214	9	3.20	66.67	57	yeast-1_vs_7	459	7	14.30	6.67
26	ecoli-0-1-4-7_vs_2-3-5-6	336	7	10.59	65.52	58	haberman	306	3	2.78	6.17
27	ecoli-0-3-4-7_vs_5-6	257	7	9.28	64.00	59	yeast-1-2-8-9_vs_7	947	8	30.57	3.33
28	glass0	214	9	2.06	58.57	60	glass-0-1-5_vs_2	172	9	9.12	0.00
29	ecoli-0-1_vs_2-3-5	244	7	9.17	58.33	61	glass-0-1-6_vs_2	192	9	10.29	0.00
30	ecoli1	336	7	3.36	57.14	62	glass2	214	9	11.59	0.00
31	yeast3	1484	8	8.10	55.83	63	glass-0-1-4-6_vs_2	205	9	11.06	0.00
32	glass-0-6_vs_5	108	9	11.00	55.56	64	yeast-1-4-5-8_vs_7	693	8	22.10	0.00

All techniques except for the OLP use the same pool of classifiers containing 100 Perceptrons (learning rate $\alpha = 0.001$ and number of iterations $n_{iter} = 100$) that was generated using Bagging (Breiman, 1996), as in (Souza *et al.*, 2019a). To obtain the output class probabilities of the base-classifiers, we apply the sigmoid function over the hyperplanes' decision function.

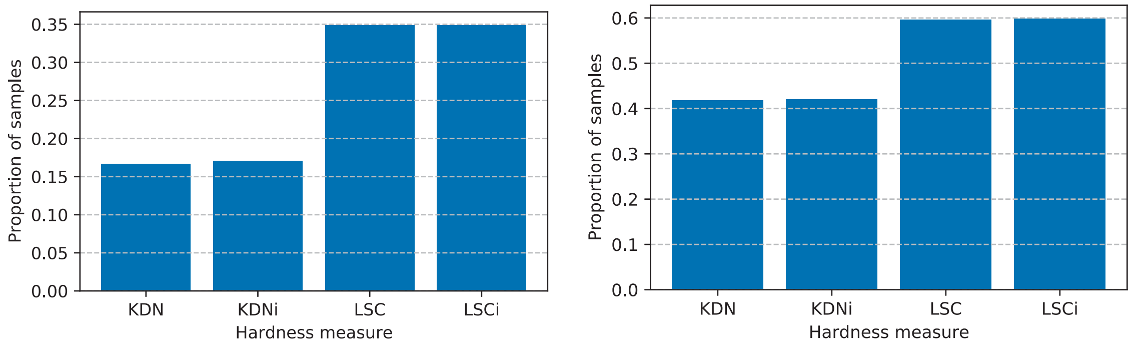
Lastly, since the proposed technique performs a dynamic RoC definition that takes into account the degree of class overlap associated with the location of each instance, the end result may be similar to performing an instance selection/noise removal over the DSEL set. Thus, we also include in the analysis the use of the Edited Nearest Neighbors (ENN) (Wilson, 1972) pre-processing method, available in the Python library imbalanced-learn (Lemaître *et al.*, 2017), over the DSEL set only. The ENN, in this case, removes from the DSEL set the negative class samples which contain a majority of its $k = 3$ neighbors from the positive class.

Table-A III-2 Dynamic Ensemble Selection (DES) techniques included in the comparative analysis.

Dynamic Ensemble Selection technique	Reference
K-Nearest Oracles Eliminate (KNORA-E)	(Ko *et al.*, 2007)
K-Nearest Oracles Union (KNORA-U)	(Ko *et al.*, 2007)
Dynamic Ensemble Selection Performance (DESP)	(Woloszynski & Kurzynski, 2011)
Dynamic Ensemble Selection Clustering (DESC)	(Soares *et al.*, 2006)
K-Nearest Output Profiles (KNOP)	(Cavalin *et al.*, 2012)
Dynamic Ensemble Selection KNN (DES-KNN)	(Soares *et al.*, 2006)
Meta-learning for Dynamic Ensemble Selection (META-DES)	(Cruz *et al.*, 2015a)
Randomized Reference Classifier (DES-RRC)	(Woloszynski & Kurzynski, 2011)
Online Local Pool (OLP)	(Souza *et al.*, 2019b)
K-Nearest Oracles-Borderline (KNORA-B)	(Oliveira *et al.*, 2018)
K-Nearest Oracles-Borderline-Imbalanced (KNORA-BI)	(Oliveira *et al.*, 2018)

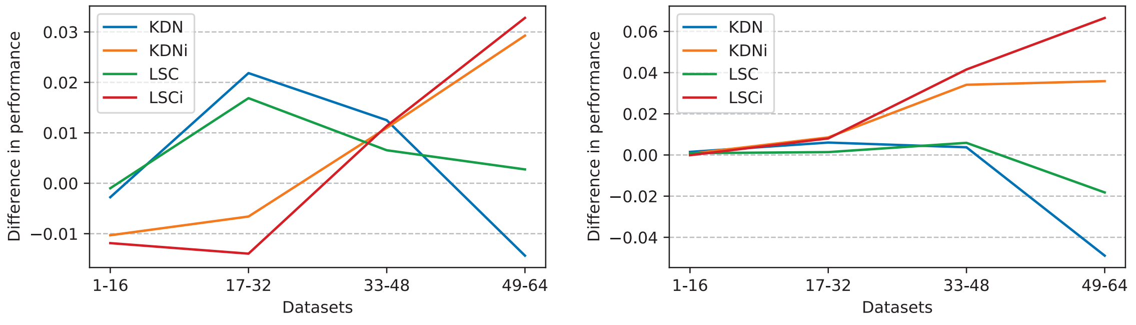
5.2 Experimental results

We start by observing the difference in the behavior of the baseline KNORA-E and the proposed technique. Fig. III-4 shows the average proportion of test instances whose sample removal order in the RoC editing scheme was different from the distance-based order from the KNORA-E, over the safe and unsafe datasets. Of course, this does not necessarily mean the selected ensemble of classifiers was different than the one KNORA-E would select. As expected due to the higher local class overlap, changes in the sample removal order were much more frequent over the unsafe datasets compared to the safe ones, considering all measures. Another trend we can observe in Fig. III-4 is that the LSC-based measures have a larger effect on the ranking of the RoC samples compared to the KDN-based. This is likely due to the local set size, which can vary on a greater scale among close-by instances compared to the amount of disagreeing neighbors of a small fixed k-neighborhood. More score variation in a small area leads to falling fewer times on the tie rule, which is the distance-based ranking from the KNORA-E.



(a) Safe (b) Unsafe

Figure-A III-4 Average proportion of instances with a different sample removal order from the baseline's (KNORA-E) over the safe and unsafe datasets.



(a) F_1 (b) G-mean

Figure-A III-5 Difference in performance between the proposed method, using the indicated hardness measure, and the baseline technique (KNORA-E), averaged over the indicated datasets (Table III-1).

As to whether the proposed RoC editing scheme was able to outperform the KNORA-E's distance-based procedure, Fig. III-5 shows the difference in average performance between the proposed method, using the indicated instance hardness measure, to the baseline (KNORA-E), averaged over each range of datasets. It can be observed that, in terms of F_1, the proposed method generally outperformed the baseline using the original hardness measures (KDN and LSC) , while using the adapted measures (KDNi and LSCi) yielded an overall superior performance over the KNORA-E over the unsafest problems . This is understandable as the original measures are expected to favor the majority class instances if the minority class is more spread out. The adapted measures, on the other hand, favor the minority class regardless, which could explain the lower F_1 over the safest datasets. W.r.t. the G-mean, however, the overall performance of the proposed method using the original measures is similar or slightly better compared to the baseline until the last, hardest group , at which point its performance drops relative to the KNORA-E, probably due to their bias towards the majority class in such problems. The gain in performance when using the adapted measures is visible for most groups.

Table-A III-3 Average performance and mean rank of the DES techniques over the groups of safe and unsafe datasets. *Wins* shows the total number of first positions. Solo wins count as 1 while ties in the first position count as 1/# tied techniques. Best results are in bold.

(a) Safe

Measure	F1		G-mean		Wins
Technique	Mean	Rank	Mean	Rank	
Prop. (KDN)	0.835	6.903	0.891	7.083	4.115
Prop. (KDNi)	0.823	9.194	0.897	7.458	2.726
Prop. (LSC)	0.834	7.097	0.889	7.250	2.448
Prop. (LSCi)	0.821	9.708	**0.899**	7.694	2.726
KNORA-E	0.824	7.917	0.886	7.958	1.226
KNORA-U	0.831	7.778	0.884	7.931	1.572
DESP	0.833	7.403	0.884	7.889	3.572
DESC	0.828	8.472	0.881	9.069	7.418
KNOP	0.833	8.306	0.887	8.681	1.572
DES-KNN	0.834	6.278	0.884	**6.542**	8.190
META-DES	0.844	7.861	0.897	7.542	3.972
DES-RRC	0.836	8.361	0.889	8.514	7.172
OLP	**0.852**	**6.194**	0.886	9.444	**19.200**
KNORA-B	0.825	9.111	0.892	8.819	2.044
KNORA-BI	0.818	9.417	0.895	8.125	4.044

(b) Unsafe

Measure	F1		G-mean		Wins
Technique	Mean	Rank	Mean	Rank	
Prop. (KDN)	0.503	7.268	0.609	7.857	0.365
Prop. (KDNi)	**0.523**	**5.732**	0.667	**4.000**	9.865
Prop. (LSC)	0.509	7.393	0.626	7.839	1.365
Prop. (LSCi)	**0.523**	6.250	**0.685**	4.161	**11.865**
KNORA-E	0.508	7.250	0.636	6.964	4.143
KNORA-U	0.454	8.929	0.534	9.839	1.098
DESP	0.455	9.250	0.536	10.196	2.765
DESC	0.426	11.232	0.507	11.446	4.143
KNOP	0.444	9.571	0.533	9.768	0.765
DES-KNN	0.484	7.500	0.575	8.518	2.143
META-DES	0.507	6.482	0.613	6.893	7.098
DES-RRC	0.456	9.179	0.540	9.857	3.098
OLP	0.460	7.804	0.533	8.946	7.000
KNORA-B	0.500	8.018	0.635	7.589	0.143
KNORA-BI	0.496	8.143	0.653	6.125	0.143

We now include all other DES techniques from Table III-2 to the comparative analysis. Table III-3 shows the average F_1 and G-mean of each technique, as well as their average rank and total number of wins, over the safe and unsafe datasets. It can be observed that, over the safe datasets, the OLP yielded the highest overall F_1 score and average rank, with the proposed technique using the KDN and LSC obtaining the third and fourth highest scores, respectively, right after the DES-KNN. In terms of G-mean, the DES-KNN obtained the highest rank, followed again by the proposed method using the KDN, LSC and KDNi. Considering both performance measures, the OLP obtained the highest number of wins over the safe datasets, though the proposed method still obtained more wins than the baseline. Over the unsafe datasets, however, the proposed method obtained the two highest mean performance and average ranks in both F_1 and G-mean when using the adapted hardness measures, as well as the two highest number of wins.

Table III-4 shows the p-values obtained from the pairwise Wilcoxon signed-rank test with significance $\alpha = 0.05$, performed over the average performances of the DS techniques. First, we can see that compared to the baseline, the proposed technique yielded a statistically similar F_1 score over the safe datasets except when using the LSCi measure, which obtained a significantly worse performance. As observed previously, the LSCi prompts a change in the RoC editing order much more often than the KDN-based measures, all the while favoring the positive class, so over the safe problems the precision may have been hindered. In fact, both adapted measures obtained a poorer F_1 score compared to the DES-KNN, META-DES and OLP over the safe datasets. However, the proposed method using the original measures was significantly better over the safe datasets than the KNORA-B and KNORA-BI. Moreover, in terms of G-mean, the proposed technique significantly outperformed both the baseline and KNORA-B over the safe datasets, using the KDN and LSC as hardness estimates.

Over the unsafe datasets, the proposed technique using the adapted measures obtained a statistically similar F_1 and a significantly better G-mean compared to the baseline, META-DES and DES-KNN. It also significantly surpassed the OLP, KNORA-B and KNORA-BI when using the LSCi measure in both F_1 and G-mean. Compared to the remaining techniques, the proposed method significantly outperformed them using all of the hardness measures investigated. This

Table-A III-4 P-value obtained from the Wilcoxon signed-rank test between the average performances of the row-wise and column-wise techniques over the groups of safe and unsafe datasets. Values below the significance $\alpha = 0.05$ are in bold. The symbols (+) and (-) indicate whether the column-wise (proposed) method was statistically superior or inferior, respectively, to the row-wise technique.

(a) Safe

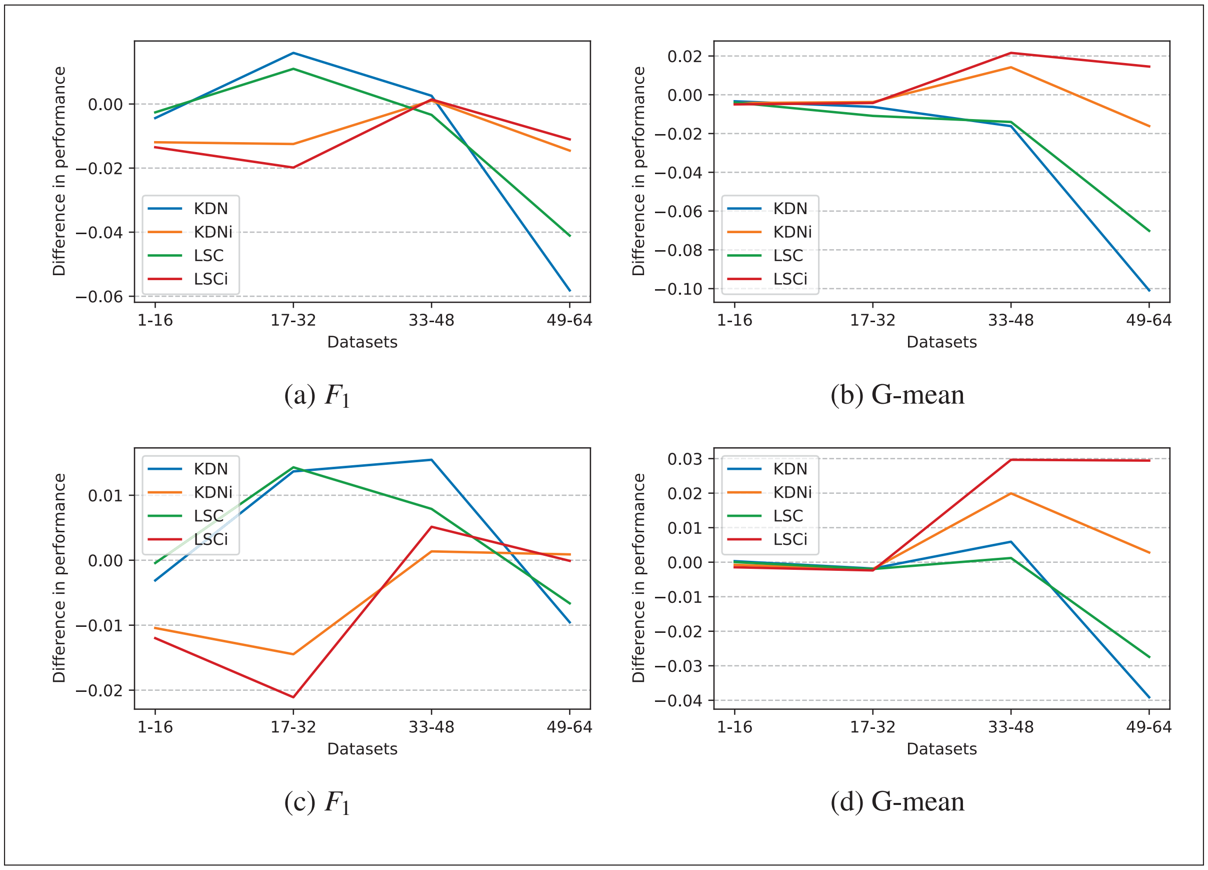
Measure	F_1				G-mean			
Technique	Prop. (KDN)	Prop. (KDNi)	Prop. (LSC)	Prop. (LSCi)	Prop. (KDN)	Prop. (KDNi)	Prop. (LSC)	Prop. (LSCi)
KNORA-E	0.066	0.265	0.099	**0.025** (-)	**0.028** (+)	0.172	**0.047** (+)	0.364
KNORA-U	0.875	0.296	0.850	0.235	0.671	0.282	0.765	0.326
DESP	0.648	0.056	0.850	**0.039** (-)	0.354	0.342	0.718	0.409
DESC	0.289	0.753	0.275	0.561	0.056	**0.043** (+)	0.074	**0.041** (+)
KNOP	0.220	0.303	0.299	0.174	0.112	0.129	0.214	0.129
DES-KNN	0.284	**0.004** (-)	0.357	**0.004** (-)	0.747	0.632	0.524	0.455
META-DES	0.987	**0.007** (-)	0.838	**0.005** (-)	0.789	0.747	0.717	0.712
DES-RRC	0.582	0.155	0.937	0.094	0.469	0.455	0.826	0.474
OLP	0.311	**0.002** (-)	0.146	**0.003** (-)	0.164	0.056	0.248	**0.031**
KNORA-B	**0.027** (+)	0.862	**0.005** (+)	0.249	**0.035** (+)	0.170	**0.025** (+)	0.352
KNORA-BI	**0.028** (+)	0.352	**0.006** (+)	0.925	0.250	0.143	0.239	0.180

(b) Unsafe

Measure	F_1				G-mean			
Technique	Prop. (KDN)	Prop. (KDNi)	Prop. (LSC)	Prop. (LSCi)	Prop. (KDN)	Prop. (KDNi)	Prop. (LSC)	Prop. (LSCi)
KNORA-E	0.882	0.151	0.546	0.198	0.094	**0.008** (+)	0.657	**0.001** (+)
KNORA-U	**0.034** (+)	**0.010** (+)	**0.040** (+)	**0.021** (+)	**0.004** (+)	**<0.001** (+)	**0.005** (+)	**<0.001** (+)
DESP	**0.019** (+)	**0.016** (+)	**0.043** (+)	**0.038** (+)	**0.001** (+)	**<0.001** (+)	**0.002** (+)	**<0.001** (+)
DESC	**0.001** (+)	**0.001** (+)	**0.002** (+)	**0.001** (+)	**0.001** (+)	**<0.001** (+)	**0.001** (+)	**<0.001** (+)
KNOP	**0.012** (+)	**0.001** (+)	**0.023** (+)	**0.002** (+)	**0.007** (+)	**<0.001** (+)	**0.003** (+)	**<0.001** (+)
DES-KNN	0.172	0.116	0.452	0.264	**0.029** (+)	**<0.001** (+)	**0.019** (+)	**<0.001** (+)
META-DES	0.927	0.322	0.838	0.682	0.733	**0.001** (+)	0.767	**0.001** (+)
DES-RRC	**0.016** (+)	**0.010** (+)	**0.031** (+)	**0.022** (+)	**0.003** (+)	**<0.001** (+)	**0.004** (+)	**<0.001** (+)
OLP	0.202	**0.024** (+)	0.096	**0.048** (+)	**0.024** (+)	**<0.001** (+)	**0.007** (+)	**<0.001** (+)
KNORA-B	0.802	0.062	0.412	**0.038** (+)	0.227	**0.004** (+)	0.909	**0.001** (+)
KNORA-BI	0.793	**0.026** (+)	0.339	**0.012** (+)	**0.005** (+)	**0.006** (+)	**0.006** (+)	**0.001** (+)

suggests that reducing the local overlap in the RoC in generalization may be advantageous for the classifier selection step, specially over the unsafe datasets.

We now analyze the performance of the proposed technique relative to the use of the ENN. In order to observe the impact of such procedure over the DS techniques, we only apply it to the DSEL set, thus the pool of classifiers is the same as in the previous analysis. Fig. III-6a and Fig. III-6b show the difference in average F_1 and G-mean between the proposed method *without* pre-processing and the baseline *with* pre-processing (KNORA-E + ENN). We can observe that



(a) F_1 (b) G-mean

(c) F_1 (d) G-mean

Figure-A III-6 Difference in performance between the proposed method using the indicated hardness measure, (a-b) without and (b-c) with pre-processing (ENN), and the baseline technique with pre-processing (KNORA-E + ENN), averaged over the indicated datasets (Table III-1).

applying the ENN over the DSEL yielded a greater improvement to the KNORA-E method compared to the proposed hardness-based RoC editing procedure. This may be due to the fact that, after applying the ENN over the DSEL, the initial RoC contains more reliable samples since the most unreliable ones in that area were already removed. Therefore, even if the proposed technique eventually removes such samples, comparatively fewer instances will remain in the RoC which may negatively impact the competence estimation. Thus, starting off the ensemble selection procedure with a less overlapped RoC seems to provide a better overall performance for the KNORA-E. However, for some datasets the dynamic sample removal used in the proposed method still yielded slightly better average results, which may motivate the design of an approach that can combine this desirable characteristic from a pre-processing procedure for an online RoC editing scheme.

Fig. III-6c and Fig. III-6d show the difference in the mean performances of the proposed method and the baseline, both using the ENN over the DSEL set now. We can see that the difference in performance presents similar characteristics to the results obtained for both techniques without the pre-processing (Fig. III-5), but with less improvement in general. With less local overlap in the DSEL, it can be expected that the proposed RoC editing scheme will have a reduced impact in performance.

Table-A III-5 Average performance and mean rank of the DES techniques with the ENN pre-processing method over the groups of safe and unsafe datasets. *Wins* shows the total number of first positions. Solo wins count as 1 while ties in the first position count as 1/# tied techniques. Best results are in bold.

(a) Safe

Measure	F1		G-mean		Wins
Technique	Mean	Rank	Mean	Rank	
Prop. (KDN)	0.838	6.639	0.897	**6.847**	3.218
Prop. (KDNi)	0.823	9.639	0.900	7.931	1.468
Prop. (LSC)	0.837	6.750	0.895	**6.847**	3.318
Prop. (LSCi)	0.822	10.167	**0.903**	8.125	3.068
KNORA-E	0.828	7.292	0.894	**6.847**	5.468
KNORA-U	0.832	8.306	0.886	8.875	2.092
DESP	0.832	8.042	0.886	8.681	4.092
DESC	0.825	9.347	0.885	9.542	2.105
KNOP	0.836	8.139	0.891	8.708	4.592
DES-KNN	0.830	6.611	0.885	7.236	5.175
META-DES	0.845	8.083	0.900	8.014	7.890
DES-RRC	0.840	7.681	0.894	8.139	7.425
OLP	**0.852**	**5.500**	0.890	8.792	**17.486**
KNORA-B	0.832	8.319	**0.903**	7.611	1.801
KNORA-BI	0.823	9.486	0.902	7.806	2.801

(b) Unsafe

Measure	F1		G-mean		Wins
Technique	Mean	Rank	Mean	Rank	
Prop. (KDN)	0.535	**6.250**	0.653	6.464	1.254
Prop. (KDNi)	0.531	6.714	0.680	**4.464**	9.254
Prop. (LSC)	0.535	6.518	0.659	6.482	1.254
Prop. (LSCi)	0.530	7.321	**0.697**	4.500	6.254
KNORA-E	**0.537**	6.839	0.676	5.839	4.487
KNORA-U	0.465	10.107	0.549	11.518	0.654
DESP	0.478	8.982	0.564	10.250	0.654
DESC	0.462	10.339	0.558	10.768	5.000
KNOP	0.458	10.054	0.551	10.875	0.654
DES-KNN	0.520	7.107	0.623	8.232	1.154
META-DES	0.526	7.304	0.633	8.089	8.654
DES-RRC	0.472	9.500	0.559	10.946	2.654
OLP	0.498	6.661	0.579	8.464	**11.100**
KNORA-B	0.529	7.893	0.673	7.000	1.487
KNORA-BI	0.518	8.411	0.684	6.107	1.487

Applying the ENN over the DSEL for all DS techniques investigated in this work, we obtain the results from Table III-5. We can see that the OLP still yielded the best average rank over the safe datasets in terms of F_1, as well as the highest number of wins considering both performance measures, but the proposed method (with the KDN and LSC) and the baseline KNORA-E obtained the highest rank in terms of G-mean. Over the unsafe datasets, the proposed method obtained the two highest average ranks w.r.t. F_1 using the original measures, and w.r.t. G-mean using the adapted measures. The second highest number of wins was also achieved by the proposed technique, using the KDNi. Interestingly using the original measures yielded a better F_1 than using the adapted measures over the unsafe datasets, as opposed to the previous results without the pre-processing, possibly due to the slightly less biased hardness estimation after removing the most overlapped majority class samples.

Table-A III-6 P-value obtained from the Wilcoxon signed-rank test between the average performances of the column-wise and row-wise techniques, with the ENN pre-processing method, over the groups of safe and unsafe datasets. Values below the significance $\alpha = 0.05$ are in bold. The symbols (+) and (-) indicate whether the column-wise (proposed) method was statistically superior or inferior, respectively, to the row-wise technique.

(a) Safe

Measure	F_1				G-mean			
Technique	Prop. (KDN)	Prop. (KDNi)	Prop. (LSC)	Prop. (LSCi)	Prop. (KDN)	Prop. (KDNi)	Prop. (LSC)	Prop. (LSCi)
KNORA-E	0.404	**0.006** (-)	0.297	**<0.001** (-)	0.860	0.057	0.974	0.061
KNORA-U	0.366	0.195	0.535	0.133	**0.039** (+)	0.072	0.133	0.091
DESP	0.582	0.052	0.620	**0.036** (-)	0.107	0.248	0.142	0.342
DESC	**0.049** (+)	0.912	**0.046** (+)	0.683	**0.011** (+)	**0.034** (+)	**0.010** (+)	**0.026** (+)
KNOP	0.370	0.111	0.268	0.072	0.076	0.133	0.063	0.155
DES-KNN	0.740	**0.003** (-)	0.918	**0.003** (-)	0.249	0.887	0.138	0.813
META-DES	0.706	**0.007** (-)	0.274	**0.003** (-)	0.409	0.556	0.404	0.545
DES-RRC	0.881	**0.049** (-)	0.994	**0.030** (-)	0.465	0.666	0.620	0.700
OLP	0.104	**0.001** (-)	0.072	**0.002** (-)	0.164	0.134	0.212	0.126
KNORA-B	**0.033** (+)	**0.012** (-)	**0.022** (+)	**<0.001** (-)	0.318	0.623	0.358	0.358
KNORA-BI	**0.018** (+)	0.893	**0.013** (+)	0.492	0.303	0.663	0.281	0.281

(b) Unsafe

Measure	F_1				G-mean			
Technique	Prop. (KDN)	Prop. (KDNi)	Prop. (LSC)	Prop. (LSCi)	Prop. (KDN)	Prop. (KDNi)	Prop. (LSC)	Prop. (LSCi)
KNORA-E	0.793	0.501	0.546	0.130	0.161	0.269	0.168	**0.029** (+)
KNORA-U	**0.001** (+)	**0.011** (+)	**0.001** (+)	**0.026** (+)	**<0.001** (+)	**<0.001** (+)	**<0.001** (+)	**<0.001** (+)
DESP	**0.005** (+)	**0.029** (+)	**0.010** (+)	0.056	**<0.001** (+)	**<0.001** (+)	**<0.001** (+)	**<0.001** (+)
DESC	**0.001** (+)	**0.003** (+)	**0.001** (+)	**0.009** (+)	**<0.001** (+)	**<0.001** (+)	**<0.001** (+)	**<0.001** (+)
KNOP	**0.001** (+)	**0.002** (+)	**0.003** (+)	**0.008** (+)	**<0.001** (+)	**<0.001** (+)	**<0.001** (+)	**<0.001** (+)
DES-KNN	0.190	0.445	0.241	0.707	**0.006** (+)	**<0.001** (+)	**0.003** (+)	**<0.001** (+)
META-DES	0.439	0.982	0.600	0.820	0.106	**0.008** (+)	0.106	**0.002** (+)
DES-RRC	**0.003** (+)	0.053	**0.005** (+)	0.068	**<0.001** (+)	**<0.001** (+)	**<0.001** (+)	**<0.001** (+)
OLP	0.285	0.316	0.412	0.750	**0.006** (+)	**0.001** (+)	**0.005** (+)	**0.001** (+)
KNORA-B	0.305	0.927	0.316	0.733	0.716	0.179	0.733	**0.029** (+)
KNORA-BI	0.136	0.206	0.136	0.259	0.101	0.592	0.127	0.094

Table III-6 shows the resulting p-values of the Wilcoxon signed-rank test on the average performances of the techniques with the ENN over the safe and unsafe datasets. We can observe that, over the safe datasets, the proposed method with the adapted measures was statistically worse than several methods including the baseline while with the original measures it surpassed the KNORA-B and KNORA-BI in terms of F_1. Over the unsafe datasets, however, the proposed method with the LSCi statistically outperformed all techniques except for the KNORA-BI, in terms of G-mean.

5.3 Lessons learned

All in all, the results suggest that there is a performance improvement to be had by characterizing the hardness of the instances in the RoC, and prioritizing the ones that seem more reliable for the classifiers' competence estimation. Using an overlap reducing pre-processing technique tailored for imbalanced data seems to have an overall better impact over the classifier selection procedure compared to the proposed dynamic RoC editing scheme alone, possibly due to the region being less ambiguous from the start. However, we observed an advantage in not outright removing the instances in memorization for some datasets. Moreover, the proposed technique still presented a performance improvement after reducing the local class overlap in the data using the pre-processing technique, which further supports the integration of instance characterization into the RoC definition for DS techniques.

Furthermore, we observed that the instance characterization within the proposed method had a large effect on performance, so choosing which hardness measure to use should be based on the distribution of the positive class in the problem. For a minority class composed of mostly safe samples, the original measures seemed to provide a good estimate of the instances' reliability in the region, even in the presence of high global imbalance ratios. On the other hand, over the unsafe datasets, the adapted measures were a much more effective guide in the proposed RoC editing scheme. Lastly, while the corresponding KDN-based and LS-based measures yielded somewhat similar performances, likely because they both attempt to quantify the local class overlap using a concept of neighborhood, we would recommend using the LS-based measures as they not only presented slightly better overall results but also do not require any hyperparameter.

6. Conclusion

In this work, we proposed a Dynamic Selection technique which dynamically edits the target region in the search for a local oracle. Motivated by the observation that a class-overlapped region can hinder the system's recognition rates, specially over the locally under-represented class, the proposed method removes the samples perceived as most unreliable from the RoC one by one until at least one classifier can label all remaining instances in it. For characterizing the known instances in the problem, we use two instance hardness measures that convey the degree of local overlap in the area. We also propose and evaluate an adapted version of these measures as they can often be biased towards the majority class.

Experiments were conducted over 64 imbalanced datasets, which were split into two groups according to the percentage of safe positive class instances in the problem. The proposed method yielded very competitive results against a baseline and 10 other DS techniques, specially over the unsafe datasets, which suggests that the overlap-reducing procedure on the RoC can improve the competence estimation and thus the selection of the classifiers. Moreover, the most adequate instance characterization to use within the proposed technique appears to depend on the positive class distribution, as the original instance hardness measures present a high bias towards the majority class on the unsafe problems.

Future work in this line of research may involve a further investigation on the impact of using pre-processing methods to remove the local overlap for DS techniques, as opposed to dynamically editing the RoC. This may lead to the study of a scheme which can provide some advantages from both approaches, in order to improve the competence estimation of the classifiers and thus the DS techniques' recognition rates over all classes.

BIBLIOGRAPHY

Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L. & Herrera, F. (2011). KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2-3), 255–287.

Antosik, B. & Kurzynski, M. (2011). New measures of classifier competence-heuristics and application to the design of multiple classifier systems. *Computer Recognition Systems 4*, pp. 197–206.

Armano, G. & Hatami, N. (2010a). Mixture of Random Prototype-Based Local Experts. *International Conference on Hybrid Artificial Intelligence Systems*, (Lecture Notes in Computer Science), 548–556.

Armano, G. & Hatami, N. (2010b). Mixture of Random Prototype-Based Local Experts. *Hybrid Artificial Intelligence Systems*, (Lecture Notes in Computer Science), 548–556.

Armano, G. & Tamponi, E. (2018). Building forests of local trees. *Pattern Recognition*, 76, 380–390.

Arruda, J. L., Prudêncio, R. B. & Lorena, A. C. (2020). Measuring Instance Hardness Using Data Complexity Measures. *Brazilian Conference on Intelligent Systems*, pp. 483–497.

Bache, K. & Lichman, M. (2013). UCI machine learning repository. [Online].

Barandela, R., Valdovinos, R. M. & Sánchez, J. S. (2003). New applications of ensembles of classifiers. *Pattern Analysis & Applications*, 6(3), 245–256.

Bashbaghi, S., Granger, E., Sabourin, R. & Bilodeau, G.-A. (2017). Dynamic Selection of Exemplar-SVMs for Watch-list Screening through Domain Adaptation. *ICPRAM*, pp. 738–745.

Bellman, R. E. (2015). *Adaptive control processes*. Princeton university press.

Benavoli, A., Corani, G. & Mangili, F. (2016). Should we really use post-hoc tests based on mean-ranks? *The Journal of Machine Learning Research*, 17(1), 152–161.

Berg, R. v. d., Kipf, T. N. & Welling, M. (2017). Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*.

Biedrzycki, J. & Burduk, R. (2020). Decision Tree Integration Using Dynamic Regions of Competence. *Entropy*, 22(10), 1129.

Bischl, B., Schiffner, J. & Weihs, C. (2013). Benchmarking local classification methods. *Computational Statistics*, 28(6), 2599–2619.

Bottou, L. & Vapnik, V. (1992). Local learning algorithms. *Neural computation*, 4(6), 888–900.

Boutell, M. R., Luo, J., Shen, X. & Brown, C. M. (2004). Learning multi-label scene classification. *Pattern recognition*, 37(9), 1757–1771.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.

Breiman, L. (2017). *Classification and regression trees*. Routledge.

Britto, A., Sabourin, R. & Oliveira, L. (2014). Dynamic selection of classifiers - A comprehensive review. *Pattern Recognition*, 47(11), 3665–3680.

Brun, A. L., Britto, A. S., Oliveira, L. S., Enembreck, F. & Sabourin, R. (2016). Contribution of data complexity features on dynamic classifier selection. *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 4396–4403.

Burduk, R. & Biedrzycki, J. (2022). Subspace-based decision trees integration. *Information Sciences*, 592, 215–226.

Cai, H., Zheng, V. W. & Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9), 1616–1637.

Campos, Y., Morell, C. & Ferri, F. J. (2012). A local complexity based combination method for decision forests trained with high-dimensional data. *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*, pp. 194–199.

Cao, Y., Geddes, T. A., Yang, J. Y. H. & Yang, P. (2020). Ensemble deep learning in bioinformatics. *Nature Machine Intelligence*, 2(9), 500–508.

Cavalin, P. R., Sabourin, R. & Suen, C. Y. (2012). LoGID: An adaptive framework combining local and global incremental learning for dynamic selection of ensembles of HMMs. *Pattern Recognition*, 45(9), 3544–3556.

Chawla, N. V., Lazarevic, A., Hall, L. O. & Bowyer, K. W. (2003). SMOTEBoost: Improving prediction of the minority class in boosting. *European conference on principles of data mining and knowledge discovery*, pp. 107–119.

Chen, C., Liaw, A. & Breiman, L. (2004). Using random forest to learn imbalanced data. *University of California, Berkeley*, 110, 1–12.

Cheng, W., Hüllermeier, E. & Dembczynski, K. J. (2010). Bayes optimal multilabel classification via probabilistic classifier chains. *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 279–286.

Collins, A., Tkaczyk, D. & Beel, J. (2018). A Novel Approach to Recommendation Algorithm Selection using Meta-Learning. *AICS*, pp. 210–219.

Costa, I. G., Lorena, A. C., Peres, L. R. & de Souto, M. C. (2009). Using supervised complexity measures in the analysis of cancer gene expression data sets. *Brazilian Symposium on Bioinformatics*, pp. 48–59.

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

Cruz, R. M. O., Hafemann, L. G., Sabourin, R. & Cavalcanti, G. D. C. (2020). DESlib: A Dynamic ensemble selection library in Python. *Journal of Machine Learning Research*, 21(8), 1–5.

Cruz, R. M., Sabourin, R. & Cavalcanti, G. D. (2015b). A DEEP analysis of the META-DES framework for dynamic selection of ensemble of classifiers. *arXiv preprint arXiv:1509.00825*.

Cruz, R. M., Sabourin, R. & Cavalcanti, G. D. (2017a). META-DES. Oracle: Meta-learning and feature selection for dynamic ensemble selection. *Information fusion*, 38, 84–103.

Cruz, R. M., Zakane, H. H., Sabourin, R. & Cavalcanti, G. D. (2017b). Dynamic Ensemble Selection VS K-NN: why and when Dynamic Selection obtains higher classification performance? *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*.

Cruz, R. M., Sabourin, R. & Cavalcanti, G. D. (2018a). Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41, 195–216.

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

Cruz, R. M., Oliveira, D. V., Cavalcanti, G. D. & Sabourin, R. (2019a). FIRE-DES++: Enhanced online pruning of base classifiers for dynamic ensemble selection. *Pattern Recognition*, 85, 149–160.

Cruz, R. M., Souza, M. A., Sabourin, R. & Cavalcanti, G. D. (2019b). Dynamic ensemble selection and data preprocessing for multi-class imbalance learning. *International Journal of Pattern Recognition and Artificial Intelligence*, 33(11), 1940009.

das Dôres, S. N., Alves, L., Ruiz, D. D. & Barros, R. C. (2016). A meta-learning framework for algorithm recommendation in software fault prediction. *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pp. 1486–1491.

Data61, C. (2018). StellarGraph Machine Learning Library. GitHub.

Davtalab, R., Cruz, R. M. & Sabourin, R. (2022). Dynamic Ensemble Selection Using Fuzzy Hyperboxes. *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-9. doi: 10.1109/IJCNN55064.2022.9892635.

de Melo, C. E. C. & Prudêncio, R. B. C. (2014). Cost-Sensitive Measures of Algorithm Similarity for Meta-learning. *Intelligent Systems (BRACIS), 2014 Brazilian Conference on*, pp. 7–12.

de Souto, M. C., Soares, R. G., Santana, A. & Canuto, A. M. (2008). Empirical comparison of dynamic classifier selection methods based on diversity and accuracy for building ensembles. *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pp. 1480–1487.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7, 1–30.

Deng, L., Chen, W.-S. & Pan, B. (2018). Automatic Classifier Selection Based on Classification Complexity. *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pp. 292–303.

Didaci, L. & Giacinto, G. (2004). Dynamic classifier selection by adaptive k-nearest-neighbourhood rule. *International Workshop on Multiple Classifier Systems*, pp. 174–183.

Didaci, L., Giacinto, G., Roli, F. & Marcialis, G. L. (2005). A study on the performances of dynamic classifier selection based on local accuracy estimation. *Pattern Recognition*, 38(11), 2188–2191.

Do, T.-N. (2015). Using local rules in random forests of decision trees. *International Conference on Future Data and Security Engineering*, pp. 32–45.

Dong, M. & Kothari, R. (2003). Feature subset selection using a new definition of classifiability. *Pattern Recognition Letters*, 24(9), 1215 - 1225.

Dos Santos, E. M. & Sabourin, R. (2011). Classifier ensembles optimization guided by population oracle. *2011 IEEE Congress of Evolutionary Computation (CEC)*, pp. 693–698.

Dos Santos, E. M., Sabourin, R. & Maupin, P. (2008). A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. *Pattern recognition*, 41(10), 2993–3009.

Dos Santos, E. M., Sabourin, R. & Maupin, P. (2009). Overfitting cautious selection of classifier ensembles with genetic algorithms. *Information Fusion*, 10(2), 150–162.

Du, Y., Wang, Y., Hu, J., Li, X. & Chen, X. (2022). An emotion role mining approach based on multiview ensemble learning in social networks. *Information Fusion*, 88, 100–114.

Duin, R. P. W., Juszczak, P., de Ridder, D., Paclik, P., Pekalska, E. & Tax, D. M. (2004). Prtools, a matlab toolbox for pattern recognition.

Duin, R. P. (2002). The combining classifier: to train or not to train? *Object recognition supported by user interaction for service robots*, 2, 765–770.

El-Sappagh, S., Saleh, H., Sahal, R., Abuhmed, T., Islam, S. R., Ali, F. & Amer, E. (2021). Alzheimer's disease progression detection model based on an early fusion of cost-effective multimodal data. *Future Generation Computer Systems*, 115, 680–699.

Fan, W., Stolfo, S. J., Zhang, J. & Chan, P. K. (1999). AdaCost: misclassification cost-sensitive boosting. *6th International Conference on Machine Learning*, pp. 97–105.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, 27(8), 861–874.

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

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

Fernández-Delgado, M., Cernadas, E., Barro, S. & Amorim, D. (2014). Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research*, 15, 3133-3181.

Fernández-Delgado, M., Cernadas, E., Barro, S. & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems. *J. Mach. Learn. Res*, 15(1), 3133–3181.

Feurer, M., Van Rijn, J. N., Kadra, A., Gijsbers, P., Mallik, N., Ravi, S., Müller, A., Vanschoren, J. & Hutter, F. (2021). Openml-python: an extensible python api for openml. *The Journal of Machine Learning Research*, 22(1), 4573–4577.

Flach, P. (2019). Performance evaluation in machine learning: the good, the bad, the ugly, and the way forward. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, 9808–9814.

François, D., Wertz, V. & Verleysen, M. (2007). The concentration of fractional distances. *IEEE Transactions on Knowledge and Data Engineering*, 19(7), 873–886.

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55, 119–139.

Fries, N. & Rydén, P. (2022). A comparison of local explanation methods for high-dimensional industrial data: A simulation study. *Expert Systems with Applications*, 207, 117918.

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

Galar, M., Fernández, A., Barrenechea, E. & Herrera, F. (2015). DRCW-OVO: distance-based relative competence weighting combination for one-vs-one strategy in multi-class problems. *Pattern recognition*, 48(1), 28–42.

Gao, X., Shan, C., Hu, C., Niu, Z. & Liu, Z. (2019). An adaptive ensemble machine learning model for intrusion detection. *IEEE Access*, 7, 82512–82521.

Garcia, L. P., Lorena, A. C., de Souto, M. C. & Ho, T. K. (2018). Classifier Recommendation Using Data Complexity Measures. *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 874–879.

Garcia, L. P., de Carvalho, A. C. & Lorena, A. C. (2015). Effect of label noise in the complexity of classification problems. *Neurocomputing*, 160, 108 - 119.

García, V., Sánchez, J. & Mollineda, R. (2007). An empirical study of the behavior of classifiers on imbalanced and overlapped data sets. *Iberoamerican Congress on Pattern Recognition*, pp. 397–406.

García, V., Mollineda, R. A. & Sánchez, J. S. (2008). On the k-NN performance in a challenging scenario of imbalance and overlapping. *Pattern Analysis and Applications*, 11(3-4), 269–280.

García, V., Marqués, A. I. & Sánchez, J. S. (2019). Exploring the synergetic effects of sample types on the performance of ensembles for credit risk and corporate bankruptcy prediction. *Information Fusion*, 47, 88–101.

Giacinto, G. & Roli, F. (1999). Methods for dynamic classifier selection. *Proceedings 10th International Conference on Image Analysis and Processing*, pp. 659–664.

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

Giacinto, G., Roli, F. & Fumera, G. (2000). Selection of classifiers based on multiple classifier behaviour. *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition*, pp. 87–93.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. (2017). Neural message passing for quantum chemistry. *International conference on machine learning*, pp. 1263–1272.

Goel, L., Sharma, M., Khatri, S. K. & Damodaran, D. (2020). Defect Prediction of Cross Projects Using PCA and Ensemble Learning Approach. In *Micro-Electronics and Telecommunication Engineering* (pp. 307–315). Springer.

Gormez, Y., Aydin, Z., Karademir, R. & Gungor, V. C. (2020). A deep learning approach with Bayesian optimization and ensemble classifiers for detecting denial of service attacks. *International Journal of Communication Systems*, 33(11), e4401.

Gupta, A., Khan, R. U., Singh, V. K., Tanveer, M., Kumar, D., Chakraborti, A. & Pachori, R. B. (2020). A novel approach for classification of mental tasks using multiview ensemble learning (MEL). *Neurocomputing*, 417, 558–584.

Hakala, K., Kaewphan, S., Björne, J., Mehryary, F., Moen, H., Tolvanen, M., Salakoski, T. & Ginter, F. (2020). Neural network and random forest models in protein function prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(3), 1772–1781.

Hamilton, W., Ying, Z. & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, pp. 1024–1034.

Hamilton, W. L. (2020). Graph Representation Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3), 1-159.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8), 832–844.

Ho, T. K. & Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE transactions on pattern analysis and machine intelligence*, 24(3), 289–300.

Jackowski, K. & Wozniak, M. (2009). Algorithm of designing compound recognition system on the basis of combining classifiers with simultaneous splitting feature space into competence areas. *Pattern Analysis and Applications*, 12(4), 415–425.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J. & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1), 79–87.

Joshi, M. V., Kumar, V. & Agarwal, R. C. (2001). Evaluating boosting algorithms to classify rare classes: Comparison and improvements. *Proceedings IEEE International Conference on Data Mining*, pp. 257–264.

Jutten, C. (2002). The Enhanced Learning for Evolutive Neural Architectures Project. [Online].

King, R. D., Feng, C. & Sutherland, A. (1995). Statlog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, 9(3), 289–333.

Kipf, T. N. & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *International Conference on Learning Representations (ICLR).*

Kittler, J., Hatef, M., Duin, R. P. W. & Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20, 226–239.

Knuth, D. E. (1976). Big omicron and big omega and big theta. *ACM Sigact News*, 8(2), 18–24.

Ko, A. H.-R., Sabourin, R. & de Souza Britto Jr, A. (2007). A new dynamic ensemble selection method for numeral recognition. *7th International Conference on Multiple Classifier Systems*, pp. 431–439.

Kotsiantis, S. B., Kanellopoulos, D. N. & Pintelas, P. E. (2006). Local Boosting of Decision Stumps for Regression and Classification Problems. *Journal of Computers*, 1, 30-37.

Kotthoff, L. (2016). Algorithm selection for combinatorial search problems: A survey. In *Data Mining and Constraint Programming* (pp. 149–190). Springer.

Krawczyk, B. & Cyganek, B. (2017). Selecting locally specialised classifiers for one-class classification ensembles. *Pattern Analysis and Applications*, 20(2), 427–439.

Ksieniewicz, P., Zyblewski, P. & Burduk, R. (2021). Fusion of linear base classifiers in geometric space. *Knowledge-Based Systems*, 227, 107231.

Kubat, M., Matwin, S. et al. (1997). Addressing the curse of imbalanced training sets: one-sided selection. *Icml*, 97, 179–186.

Kuncheva, L. (2004). Ludmila Kuncheva Collection. [Online].

Kuncheva, L. (2014). *Combining pattern classifiers: methods and algorithms*. J. Wiley.

Kuncheva, L. I. (2000). Clustering-and-selection model for classifier combination. *KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No.00TH8516)*, 1, 185–188 vol.1.

Kuncheva, L. I. (2002). A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2), 281–286.

Kuncheva, L. I. & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2), 181–207.

Lemaître, G., Nogueira, F. & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, 18(17), 1-5.

Leyva, E., González, A. & Perez, R. (2014). A set of complexity measures designed for applying meta-learning to instance selection. *IEEE Transactions on Knowledge and Data Engineering*, 27(2), 354–367.

Li, D., Wen, G., Li, X. & Cai, X. (2019). Graph-based dynamic ensemble pruning for facial expression recognition. *Applied Intelligence*, 49(9), 3188–3206.

Lopez-Garcia, P., Masegosa, A. D., Osaba, E., Onieva, E. & Perallos, A. (2019). Ensemble classification for imbalanced data based on feature space partitioning and hybrid metaheuristics. *Applied Intelligence*, 49(8), 2807–2822.

Lorena, A. C., Costa, I. G., Spolaôr, N. & De Souto, M. C. (2012). Analysis of complexity indices for classification problems: Cancer gene expression data. *Neurocomputing*, 75(1), 33–42.

Lorena, A. C., Garcia, L. P., Lehmann, J., Souto, M. C. & Ho, T. K. (2019). How complex is your classification problem? a survey on measuring classification complexity. *ACM Computing Surveys (CSUR)*, 52(5), 1–34.

Masoudnia, S. & Ebrahimpour, R. (2014). Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2), 275–293.

Mazurowski, M. A., Habas, P. A., Zurada, J. M., Lo, J. Y., Baker, J. A. & Tourassi, G. D. (2008). Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, 21(2-3), 427–436.

McInnes, L., Healy, J. & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv preprint arXiv:1802.03426*.

Melo Junior, L., Macedo, J. F., Nardini, F. M. & Renso, C. (2019). KNORA-IU: Improving the Dynamic Selection Prediction in Imbalanced Credit Scoring Problems. *Proceedings of the 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 424–431.

Mendialdua, I., Martínez-Otzeta, J. M., Rodriguez-Rodriguez, I., Ruiz-Vazquez, T. & Sierra, B. (2015). Dynamic selection of the best base classifier in One versus One. *Knowledge-Based Systems*, 85, 298–306.

Menzies, T., Greenwald, J. & Frank, A. (2006). Data mining static code attributes to learn defect predictors. *IEEE transactions on software engineering*, 33(1), 2–13.

Muñoz, M. A., Villanova, L., Baatar, D. & Smith-Miles, K. (2018). Instance spaces for machine learning classification. *Machine Learning*, 107(1), 109–147.

Nagarajan, A., Stevens, J. R. & Raghunathan, A. (2022). Efficient ensembles of graph neural networks. *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 187–192.

Napierala, K. & Stefanowski, J. (2016). Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, 46(3), 563–597.

Narassiguin, A., Elghazel, H. & Aussem, A. (2017). Dynamic ensemble selection with probabilistic classifier chains. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 169–186.

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

Oliveira, D. V., Cavalcanti, G. D., Porpino, T. N., Cruz, R. M. & Sabourin, R. (2018). K-nearest oracles borderline dynamic classifier ensemble selection. *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8.

Osama, S., Shaban, H. & Ali, A. A. (2023). Gene reduction and machine learning algorithms for cancer classification based on microarray gene expression data: A comprehensive review. *Expert Systems with Applications*, 213, 118946.

Partalas, I., Tsoumakas, G. & Vlahavas, I. P. (2008). Focused Ensemble Selection: A Diversity-Based Method for Greedy Ensemble Selection. *ECAI*, pp. 117–121.

Pascual-Triana, J. D., Charte, D., Andrés Arroyo, M., Fernández, A. & Herrera, F. (2021). Revisiting data complexity metrics based on morphology for overlap and imbalance: snapshot, new overlap number of balls metrics and singular problems prospect. *Knowledge and Information Systems*, 63(7), 1961–1989.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct), 2825–2830.

Pereira, M., Britto, A., Oliveira, L. & Sabourin, R. (2018). Dynamic ensemble selection by k-nearest local oracles with discrimination index. *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 765–771.

Pinto, F., Soares, C. & Mendes-Moreira, J. (2016). Chade: Metalearning with classifier chains for dynamic combination of classifiers. *Joint european conference on machine learning and knowledge discovery in databases*, pp. 410–425.

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

Prati, R. C., Batista, G. E. & Monard, M. C. (2004). Class imbalances versus class overlapping: an analysis of a learning system behavior. *Mexican international conference on artificial intelligence*, pp. 312–321.

Prati, R. C., Batista, G. E. & Silva, D. F. (2015). Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. *Knowledge and Information Systems*, 45(1), 247–270.

Radovanovic, M., Nanopoulos, A. & Ivanovic, M. (2010). Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(sept), 2487–2531.

Read, J., Pfahringer, B., Holmes, G. & Frank, E. (2011). Classifier chains for multi-label classification. *Machine learning*, 85(3), 333.

Ross, A., Pan, W., Celi, L. & Doshi-Velez, F. (2020). Ensembles of Locally Independent Prediction Models. 3, 1-11.

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

Ruta, D. & Gabrys, B. (2005). Classifier selection for majority voting. *Information fusion*, 6(1), 63–81.

Salehi, A. & Davulcu, H. (2020). Graph Attention Auto-Encoders. *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 989-996.

Sánchez, J. S., Mollineda, R. A. & Sotoca, J. M. (2007). An analysis of how training data complexity affects the nearest neighbor classifiers. *Pattern Analysis and Applications*, 10(3), 189–201.

Schapire, R. E., Freund, Y., Bartlett, P. & Lee, W. S. (1997). Boosting the margin: a new explanation for the effectiveness of voting methods. *14th International Conference on Machine Learning*, pp. 322–330.

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J. & Napolitano, A. (2010). RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(1), 185–197.

Sierra, B., Lazkano, E., Irigoien, I., Jauregi, E. & Mendialdua, I. (2011). K Nearest Neighbor Equality: Giving equal chance to all existing classes. *Information Sciences*, 181(23), 5158–5168.

Simpson, P. K. (1992). Fuzzy Min—MaX Neural NetWorks—Part 1: Classification. *IEEE Trans. on Neural Networks*, 3(5), 776–786.

Singh, S. (2003). Multiresolution estimates of classification complexity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12), 1534–1539.

Skalak, D. B. (1996). The Sources of Increased Accuracy for Two Proposed Boosting Algorithms. *In Proc. American Association for Arti Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop*, pp. 120–125.

Smith, M. R. & Martinez, T. (2011). Improving classification accuracy by identifying and removing instances that should be misclassified. *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pp. 2690–2697.

Smith, M. R. & Martinez, T. (2016). A comparative evaluation of curriculum learning with filtering and boosting in supervised classification problems. *Computational Intelligence*, 32(2), 167–195.

Smith, M. R., Martinez, T. & Giraud-Carrier, C. (2014). An instance level analysis of data complexity. *Machine Learning*, 95(2), 225–256.

Smits, P. C. (2002). Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection. *IEEE Transactions on Geoscience and Remote Sensing*, 40(4), 801–813.

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

Souza, M. A. (2018). *An online local pool generation method for dynamic classifier selection*. (Master's thesis, Universidade Federal de Pernambuco).

Souza, M. A., Cavalcanti, G. D., Cruz, R. M. & Sabourin, R. (2017). On the characterization of the Oracle for dynamic classifier selection. *International Joint Conference on Neural Networks (IJCNN), 2017*, pp. 332–339.

Souza, M. A., Cavalcanti, G. D., Cruz, R. M. & Sabourin, R. (2019a). On evaluating the online local pool generation method for imbalance learning. *International Joint Conference on Neural Networks (IJCNN), 2019*, pp. 1–8.

Souza, M. A., Cavalcanti, G. D., Cruz, R. M. & Sabourin, R. (2019b). Online local pool generation for dynamic classifier selection. *Pattern Recognition*, 85, 132–148.

Souza, M. A., Sabourin, R., Cavalcanti, G. D. & Cruz, R. M. (2020). Multi-label learning for dynamic model type recommendation. *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10.

Souza, M. A., Sabourin, R., Cavalcanti, G. D. C. & Cruz, R. M. O. (2022). Local overlap reduction procedure for dynamic ensemble selection. *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-9. doi: 10.1109/IJCNN55064.2022.9892846.

Souza, M. A., Sabourin, R., Cavalcanti, G. D. & Cruz, R. M. (2023). OLP++: An online local classifier for high dimensional data. *Information Fusion*, 90, 120-137.

Szymański, P. & Kajdanowicz, T. (2019). Scikit-multilearn: a scikit-based Python environment for performing multi-label classification. *Journal of Machine Learning Research*, 20, 209-230.

Valentini, G. (2005). An experimental bias-variance analysis of svm ensembles based on resampling techniques. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B 35, 1252–1271.

van Rijsbergen, C. (1979). *Information Retrieval*. Butterworth.

Vandaele, R., Kang, B., De Bie, T. & Saeys, Y. (2022). The Curse Revisited: When are Distances Informative for the Ground Truth in Noisy High-Dimensional Data? *International Conference on Artificial Intelligence and Statistics*, pp. 2158–2172.

Vanschoren, J., van Rijn, J. N., Bischl, B. & Torgo, L. (2013). OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, 15(2), 49–60.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. & Bengio, Y. (2018). Graph Attention Networks. *International Conference on Learning Representations*, 1–12.

Verma, B. & Rahman, A. (2011). Cluster-oriented ensemble classifier: Impact of multicluster characterization on ensemble classifier learning. *IEEE Transactions on Knowledge and Data Engineering*, 24(4), 605–618.

Wang, S. & Yao, X. (2009). Diversity analysis on imbalanced data sets by using ensemble models. *IEEE Symposium on Computational Intelligence and Data Mining*, pp. 324–331.

Wang, Y., Xu, H., Yu, Y., Zhang, M., Li, Z., Yang, Y. & Wu, W. (2022). Ensemble Multi-Relational Graph Neural Networks. 2298–2304.

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

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3), 408–421.

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

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

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

Woźniak, M., Graña, M. & Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16, 3–17.

Xia, F., Sun, K., Yu, S., Aziz, A., Wan, L., Pan, S. & Liu, H. (2021). Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(2), 109–127.

Xiao, J., Xie, L., He, C. & Jiang, X. (2012). Dynamic classifier ensemble model for customer classification with imbalanced class distribution. *Expert Systems with Applications*, 39(3), 3668–3675.

Xu, K., Hu, W., Leskovec, J. & Jegelka, S. (2019). How Powerful are Graph Neural Networks? 1–17.

Yan, C., Li, M., Ma, J., Liao, Y., Luo, H., Wang, J. & Luo, J. (2022). A Novel Feature Selection Method Based on MRMR and Enhanced Flower Pollination Algorithm for High Dimensional Biomedical Data. *Current Bioinformatics*, 17(2), 133–149.

Zakai, A. & Ritov, Y. (2008). How Local Should a Learning Method Be? *COLT*.

Zhang, J., Shi, X., Xie, J., Ma, H., King, I. & Yeung, D. Y. (2018). GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. 339–349.

Zhang, L. & Suganthan, P. N. (2017). Benchmarking ensemble classifiers with novel co-trained kernel ridge regression and random vector functional link ensembles [research frontier]. *IEEE Computational Intelligence Magazine*, 12(4), 61–72.

Zhang, M.-L. & Zhou, Z.-H. (2013). A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8), 1819–1837.

Zhang, S. (2022). Challenges in KNN Classification. *IEEE Transactions on Knowledge and Data Engineering*, 34(10), 4663-4675.

Zhang, S., Tong, H., Xu, J. & Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1), 1–23.

Zhang, Z., Cui, P. & Zhu, W. (2022). Deep Learning on Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1), 249-270.

Zheng, Z., Wu, X. & Srihari, R. (2004). Feature selection for text categorization on imbalanced data. *ACM Sigkdd Explorations Newsletter*, 6(1), 80–89.

Zhou, P., Wang, X., Du, L. & Li, X. (2022). Clustering ensemble via structured hypergraph learning. *Information Fusion*, 78, 171–179.

Zhou, Z. (2012). *Ensemble methods*. Taylor & Francis.

Zhu, Z., Wang, Z., Li, D. & Du, W. (2019). Tree-based space partition and merging ensemble learning framework for imbalanced problems. *Information Sciences*, 503, 1–22.