UNIVERSIDAD DE DEUSTO

# ENHANCEMENT OF ENSEMBLE DATA MINING TECHNIQUES VIA SOFT COMPUTING

by

Amgad Monir Mohamed Elsayed

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, within the PhD Program in Engineering for the Information Society and Sustainable Development

Supervised by Prof. Enrique Onieva Caracuel and Prof. Michał Woźniak

UNIVERSIDAD DE DEUSTO

# ENHANCEMENT OF ENSEMBLE DATA MINING TECHNIQUES VIA SOFT COMPUTING

by

Amgad Monir Mohamed Elsayed

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, within the PhD Program in Engineering for the Information Society and Sustainable Development

Supervised by Prof. Enrique Onieva Caracuel and Prof. Michał Woźniak

The candidate The supervisor

Bilbao, February 2021

*Enhancement of ensemble data mining techniques via soft computing*

Author: Amgad Monir Mohamed Elsayed

Supervisor: Prof. Enrique Onieva Caracuel and Prof. Michał Woźniak

Text printed in Bilbao

First edition, February 2021

*Dedicated to everyone who wish for a better life!*

# Abstract

Machine learning (ML) is the area of study that gives computers the ability to learn without being explicitly programmed. Sometimes this will reveal unsuspected correlations and lead to a deeper understanding of the problem. The magic is to *learn from data*, as we are surrounded by data everywhere (user logs, financial data, production data, medical records, etc.). Machine learning is great for complex problems for which there is no good solution at all. Furthermore, ML is suitable for fluctuating environments as it can adapt to new data. While data mining is a related field that aims to discover patterns that were not immediately apparent. There are two important factors that drive this area: usage of effective models that capture the complex data, and design of scalable learning systems that learn from massive datasets.

While it has been extensively reported in the literature that pooling together learning models is a desirable strategy to construct robust data mining systems. This is recognized as ensemble data mining. Ensemble systems for pattern classification have been expanded in the literature under the name of multiple classifier system (MCS). In classification tasks, various challenges are encountered, e.g., in terms of the data size, the number of classes, the dimensionality of the feature space, the overlap between instances, the balance between class categories, and the nonlinear complexity of the true unknown hypotheses. Those challenges cause the perfect solutions to be difficult to obtain. A promising solution is to train a set of diverse and accurate base classifiers and to combine them.

A primary drawback of classifiers ensemble, despite its remarkable performance, is that it is necessary to combine a large number of classifiers to ensure that the error converges to its asymptotic value. This

brings on high computational requirements, including the cost of training, large space of memory, and large time for prediction. In addition, when classifiers are spread over a network, high communication costs are needed. To alleviate these drawbacks, various strategies will be proposed in this thesis. In particular, how soft computing techniques can be incorporated in MCS.

Soft computing methods are pioneer computing paradigms that parallel the extraordinary ability of the human mind to reason and learn. Soft computing methods, computational intelligence, use approximate calculations to provide imprecise but usable solutions to unsolvable or just too time-consuming problems. From the literature in MCS, at most, soft computing methods were proposed either to optimize the classifiers' combination function or to select a subset of classifiers instead of aggregating all. However, the efficiency and efficacy of MCS can be still improved through our contributions in this thesis.

The efficiency of MCS concerns; fast training, lower storage requirements, higher classification speed, lower communication cost between distributed models. Two directions were followed to achieve that. First, for data level, we apply instance selection (IS) methods as a preprocessing mechanism to decrease the training data-size. This could fast the training of MCS, and the accuracy of models could be increased through focusing on informative samples. Related to this part, we evaluate the interconnection between IS and MCS. Second, for the ensemble level, ensemble pruning is a strategy by which a subset of classifiers can be selected while maintaining, even improving, the performance of the original ensemble. For that, we propose a guided search pruning method to combine multiple pruning metrics while retaining their performance. In addition, the simultaneous effect of downsizing the number of samples and downsizing the number of classifiers is analyzed. Furthermore, we analyze recent reordering-based MCS pruning metrics that are recognized as accurate and fast strategies to identify a subset of classifiers.

The efficacy of MCS concerns the predictive performance, to go beyond what can be achieved from the state-of-art ensemble algorithms. Related to this part, we apply swarm intelligence (SI) algorithms, soft computing techniques, to integrate multiple classifier decisions. In connection with that, a framework was proposed to combine three computational intelligence paradigms IS, MCS, and SI algorithms. The objective is to build a more diverse and highly accurate MCS, only from a reduced portion of the available data.

In summary, this research introduces novel and improved strategies to increase the efficiency and the efficacy of MCS. Soft computing is applied to optimize the integration of classifiers and to identify the best classifier subsets. The results obtained throughout the thesis can boost the efficiency of ensemble systems by applying IS methods as a kind of data preprocessing technique. The application of SI algorithms or hybrid versions can be more promising to effectively integrate individuals' decisions. Furthermore, small-size ensembles with training on fewer samples could significantly outperform large-size ensembles that use whole training data. Finally, an analysis of recent heuristic metrics to prune bagging ensembles has been conducted.

# Acknowledgements

First of all, I am deeply grateful to Allah. I want to thank my supervisors Enrique Onieva and Michał Woźniak, whose contribution has raised the quality of this thesis. They have always supported me and have given me enthusiasm for science. They have patiently guided me. I am very grateful for their supervision and I owe the greatest degree of appreciation. Thank you, Enrique, you always listened to me during my worst time, you taught me several things: how to be patient, to work intelligently, to work daily, and to be more organized. Thank you, Michał, for your valuable feedback and guidance, suggestions, encouragement, trusts in me, continuous and endless support during my research stay.

Thanks for the fund from the European Commission that allowed me to pursue a doctorate. With the great influence on my personality, to give back to the society that helped me to develop my skills, and shaping my career. Thanks to Asier Perallos, Pili Elejoste, Alfonso Bahillo, Osane Uriarte, and the DIRS-COFUND office for allowing me to be part of Deusto Smart Mobility, DeustoTech, and the University of Deusto.

I also want to thank my workmates, friends and former colleagues (Aimar, Alex, Asier, Alejo, Antonio, Florian, Galder, Hugo, Iker, Idoia, Itziar, Jenny, Juan, Laura, Leire, Luis, Nacho, Naia, Natasa, Pablo, Pedro, Timothy, Ruben, Alvaro, Sofia, Zabit, M. Al-Rashaida). The working environment of Deusto, the coffee room, the restaurant, the garden, all of this will be hard to forget.

Many thanks to my Egyptian colleagues (H. Zahera, I. Elgendy, M. Elgendy, A. Samy, M. Dahi, M. Hammad, A. Abd El-Latif, K. Sallam, A. Hagras) for their support and contact during my hard times. This is where I express my gratitude to many people, including professor WF Abd El-wahed who had profoundly shaped the way I think and learn, who had died years ago, God's blessings and peace. Many thanks to my previous supervisors during my Master degree, Prof. Mahmoud M. El-Sherbiny, and Prof. Nancy ELhefnawy.

Of course, none of this would have been possible without my family. My dad, you are not around, but I didn't forget you. You are the best thing in my life. Thanks for your care and teaching me during my childhood. Daily, I ask God to rest you in heaven. Many thanks to my family (Mom, Ahmed, Ayman, Akram) for their endless support. I'm touched beyond words to thank my wife, Faten, and her family for their infinite love and support. To my wife, I love you and I know you have sacrificed a lot to see this dream real, my heart is still smiling. To my kids (MOHAMED, MOAZ, TAMIM) I love all of you, I am sorry for not being with you, but one day all of you will be proud of what has been achieved.

*Thank you everyone,*

Amgad Monir

February 2021

# Contents

# List of Figures

# List of Tables

# Acronyms

**AdaB** AdaBoost

**AI** Artificial Intelligence

**ANNs** Artificial Neural Networks

**AR-Friedman** Average Rank of Friedman Test

**Bagging** Boostrap AGGregatING

**BGWO** Binary Grey Wolf Optimizer

**BS** Backward Search

**CatBoost** Gradient Boosting with Categorical features

**CBCD** Cluster Based Concurrent Decomposition

**CC** Complementariness measure

**CPU** Central Processing Unit

**CW-NN** Class Weight via Neural Network

**DC** Distributed Computing

**DISC** Discriminant classifiers pruning

**DL** Deep Learning

**DM** Data Mining

## ACRONYMS

**DT** Decision Tree

**EFB** Exclusive Feature Bundling

**EPIC** Ensemble Pruning via Individual Contributions

**ES** Ensemble Selection

**ET** Extra-Trees

**FS** Forward Search

**G-REX** Generic Rule EXtraction methods

**GA** Genetic Algorithm

**GASEN** Genetic Algorithm based Selective ENsemble

**GBM** Gradient Boosting Machine

**GOSS** Gradient-based One-Side Sampling

**GP** GPU Computing

**GWO** Grey Wolf Optimizer

**HC** Hill Climbing

**ICE** Individual Conditional Expectation plots

**IS** Instance Selection

**JRIP** Java Repeated Incremental Pruning rule learner

**KEEL** Knowledge Extraction based on Evolutionary Learning

**kNN** k-Nearest Neighbor

**LB** Logit Boost

**LightGBM** Light Gradient Boosting Machine

**LIME** Local Interpretable Model-agnostic Explanations

**LR** Logistic Regression

**MA** Metaheuristic Algorithms

**MCC** Matthews Correlation Coefficient

**MCM** Minimal Classification Method

**MCS** Multiple Classifier Systems

**MDEP** Margin and Diversity-based Ensemble Pruning

**MDSQ** Margin Distance minimization

**ME** Mixture of Experts

**MFO** Moth-Flame Optimization algorithm

**ML** Machine Learning

**MRMR** Maximum Relevance Minimum Redundancy pruning

**Multinom** Multinomial logistic regression

**NB** Naïve Bayes

**NRD-NSE** Non-Reduced Data, Non-Selected Ensembles

**OOB** Out-Of-Bag

**OpenML** Open Machine Learning repository

**OVA** One Versus All

**OVO** One Versus One

**PCC** Parallel CPU Computing

**RD-NSE** Reduced Data, Non-Selected Ensembles

**RD-SE** Reduced Data, Selected Ensembles

**RE** Reduced Error Pruning

**RF** Random Forest

**RFCOM** Random Forest trained on COMplete dataset

**RFSM** Random Forest trained on SMall dataset

**RotF** Rotation Forest

**RP** Random Patches

**RR** Random Rotation

**SA** Simulated Annealing

**SAMME** Stagewise Additive Modeling using a Multi-class Exponential function

**SBM** Single Best Model

**SC** Sequential CPU Computing

**SHAP** SHapley Additive exPlanations

**SI** Swarm Intelligence

**SVM** Support Vector Machine

**UMEP** Unsupervised Margin based Ensemble Pruning

**UNP** UNPruned ensemble

**WOA** Whale Optimization Algorithm

**XGBoost** eXtreme Gradient Boosting

*Success is the sum of small efforts,*
*repeated day-in and day-out.*

Robert Collier

CHAPTER

# 1

# Introduction

Machine learning (ML) allows developing effective data-driven solutions that allow making smart decisions. The rapid growth of data volumes and sources has increased the efficiency of data-based solutions, making this area more and more important. A closely related field to ML is data mining (DM), *¨to do with the discovery of useful, valid, unexpected, and understandable knowledge from data¨* [1]. However, with the plethora of statistical learning methods, the explored pattern is usually different and hence, the final decision could be affected [2]. A promising data-driven solution has been recognized via combining or weighting several data analytical methods [3–5]. This is accepted and promoted by computational intelligence community with the name of ensemble systems [6–12]. Specific ensemble systems for classification tasks are known as Multiple Classifier Systems (MCSs) [13–16], with the concept of designing, implementing, and validating many classifiers to cope with uncertainty, ambiguity, and complex problems. Classifier ensembles reside at the intersection of engineering, computing, and mathematics [15].

MCS is the methodology where many classifiers are generated and their decisions are combined to get more accurate and stable decisions. Regarding that, MCS should be designed efficiently in all of their stages, from data preprocessing to multioutput decision fusion. Many variants exist either to generate or to combine the group of classifiers, while the majority of the articles and the designed algorithms

consider the whole dataset to train each individual. This slows the classification system by adding more computations to train several classifiers from complete data. In addition, ambiguity and uncertainty will be increased. Furthermore, the test time will be more consumed by the individual classifiers to get the final decision.

Therefore, this thesis aims to propose the development of MCS, to design more efficient and effective classifier ensembles, via incorporating the following strategies:

- Intelligent data sampling via applying instance selection (IS) techniques.
- Building a group of heterogeneous classifiers to increase the diversity inside the decision space.
- Applying swarm intelligence (SI) as a soft computing technique to combine the classifier's outputs properly.
- Analysis of popular and recent ensemble pruning metrics to get thinner/small-size ensembles, with the significant impact to increase the efficiency and the predictive performance.

In this chapter, the motivation for this research along with the research questions that naturally arise are discussed in Section 1.1. After this, the objectives and contributions are presented in Sections 1.2 and 1.3, respectively. Next, the research methodology is summarised in Section 1.4. Finally, the research context and the outline of this thesis are presented in Sections 1.5 and 1.6, respectively.

## 1.1 Motivation and Scope

MCS performs well since classifiers differ in terms of their inductive biases [17]. With the classical approach of focusing on a highly optimized classifier, various challenges are encountered, e.g., in terms of the data size, the number of classes, the dimensionality of the feature space, the overlap between instances, the balance between class categories, and the nonlinear complexity of the true unknown hypotheses [18]. A promising alternative is to use a group of classifiers. The true unknown hypothesis can be approximated by searching in various regions. This is one of the main supports of the working mechanisms of MCS, as each classifier exploits the search space differently by using different feature subsets, data samples, or learning mechanisms. The general structure of MCS is depicted in Figure 1.1

following the classical pattern recognition structure. The set of features describing the objects are input to the classifier ensemble, formed by a set of diverse classifiers [16]. Then, an appropriate combination rule fuses the individual classifier outputs to provide the system decision.

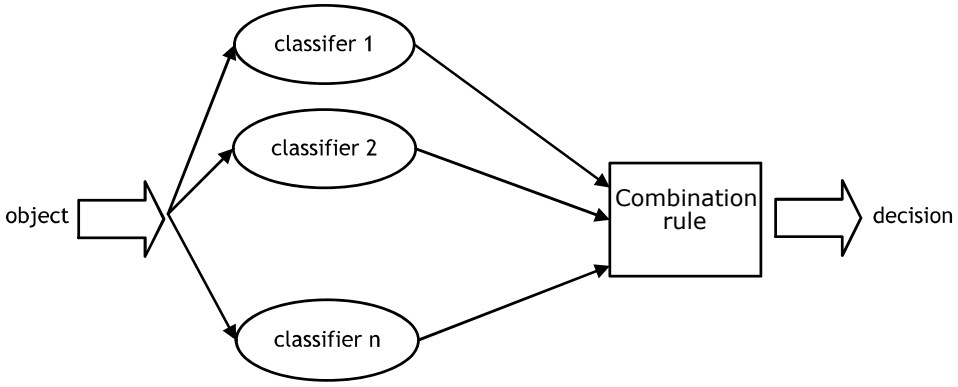


**Figure 1.1:** The canonical topology of MCS, Taken from [16].

A vital phase in building a group of classifiers is to use a suitable fusion strategy to aggregate response decisions [15, 18]. Regarding that, the general accuracy of the ensemble model could be improved by properly tuning the decision weights of individual classifiers. However, the following challenges are encountered in the application of a weight-based combination rule: (1) If the amount of data increases or the number of classifiers becomes large, the search space becomes more complex to determine the proper weight. (2) The search space may be flat or multimodal, namely, there are multiple solutions (weights) that all provide approximately the same accuracy. (3) The assignment of a general weight for each classifier in consideration of its overall accuracy is not efficient for decision fusion because each classifier has different performance capabilities for predicting per-class instances. Answering to that, swarm intelligence (SI) can be used to identify a pattern or even to tune the weight parameters for multiclassifier decision fusion. In addition, SI handles the multimodal problem by stochastically applying exploration to avoid stagnation at local solutions.

*In addition*, reducing the training data leads to reducing the in-between classifiers conflict, with the aim to decrease the uncertainty and ambiguity in MCS. However, reducing the training data could affect the learning negatively. Scientific

researchers have focused on the selection of the most relevant instances via data preprocessing [19] to feed the classification model. Regarding that, instance selection (IS) techniques could be applied in advance before training MCS. Hence, the training and the testing times will be further reduced. The combination method should exploit the strengths of individual classifiers' over each class (class-specific weights). For that, SI algorithms can be used to tune the weights of each classifier based on its accuracy to predict a specified class. Furthermore, SI algorithms could compensate the loss in the accuracy of the applied IS techniques.

A Current subject of interest in MCS is to downsize the ensemble. Downsizing the ensemble (ensemble pruning, ensemble selection, and ensemble forming) is a strategic process by which a subset of ensemble members can be selected while maintaining, even improving, the performance of the original ensemble [20]. Ensemble selection can be embedded in the combination method, as in weighted voting where classifiers with lower weights or less than a predefined threshold can be removed [21, 22]. The ensemble size has an important inconvenience according to the following: *Memory requirements* to store the parameters of all the base learning models, *classification speed* which deteriorates with large ensemble sizes, and the *predictive performance metric* that can be improved by merging a subset of models instead of depending on the whole ensemble members [23–25]. A possible solution to alleviate those shortcomings can be broadly divided into the following five solutions [26, 27]:

- Exhaustive search.
- Optimization-based search.
- Sequential search (greedy methods).
- Clustering-based pruning.
- Ranking-based pruning (ordering-based pruning).

Regarding that, a promising MCS could be designed to:

- Benefit from the downsized data and the downsized classifiers simultaneously.
- Form ensemble models quickly, especially when training complex and a large number of individual classifiers.
- Improve the size and overall accuracy of the ensemble systems.

- Get out more accuracy from the reduced data in comparison with state-of-art ensembles which are trained from non reduced data.

Multiple classifier systems are composed of three stages: (A) Forming, (B) Selection, and (C) Aggregation [28]. The selection process is optional as it is not embedded in many ensemble systems. However, it has been proved that the generalization performance of a subensemble reports superior results over the traditional combination approaches, such as majority voting of the whole ensemble [23, 29]. In addition, pruning down the redundant models reduces the memory burden [30]. Furthermore, ensemble selection (ES) is a proven mechanism to enhance the efficiency and elevate the efficacy of classification ensemble systems [23, 24, 31, 32]. However, it is not trivial to find the optimal subset of classifiers from a large ensemble as the complexity grows exponentially with the size of the pool. Researchers agreed in common that ES is a combinatorial search problem with $2^T - 1$ nonempty subsets to be evaluated from pool size, $T$, to find the best subset [33, 34].

Greedy algorithms and heuristic metrics have been proved to be convenient techniques that return near-optimal subsets in fast time. Virtually, an ordered list from the generated classifier pool is formed according to an evaluation function, followed by the selection of models according to this fixed order [31]. Those techniques comprise dissimilar heuristic measures such as: ensemble diversity [32], ensemble margin [24, 35], margin hybrid diversity [36], discriminating classifiers [31], ensemble error [37], complementary of misclassification [35], and relative accuracy with minimum redundancy [31]. In [38], since no widely accepted definition to measure the ensemble diversity exists, five pairwise diversity measures are combined to obtain an efficient pruned ensembles. In the literature, those techniques are popular and recognized under the name of ordering-based ensemble pruning with the following merits:

- The ordering strategies return subsets that are close to the optimal solution (*Efficacy*) [23].
- The time complexity of those strategies is low, in comparison with exhaustive or optimization-based search methods (*Efficiency*) [23].
- Pruning strategies based on base classifier reordering can be easily adjusted to adapt to any given storage and computational restrictions [31, 36].

Till now and related to our best knowledge, no recent article has considered the analysis of all those promising metrics together. This research covers that gap by comparing all those new techniques with the best performing techniques found in [23], and against other popular baseline metrics.

Summarising, the following research questions are stated based on the above motivations:

- $Q_1$. What is the impact of reduced and consistent data on the performance of ensemble learning?
- $Q_2$. Is it possible with the search capability of swarm intelligence to enhance the combination of classifiers?
- $Q_3$. What is the effect of combining multiple pruning metrics together?
- $Q_4$. What is the effect of downsizing data and downsizing the number of classifiers simultaneously?
- $Q_5$. How the initial classifier pool size and the required subensemble size affect on the performance of heuristic pruning metrics?
- $Q_6$. How the heuristic pruning metrics are affected by the individual classifier type?
- $Q_7$. How the efficacy of the pruning metrics could be affected by binary and multi-class tasks?
- $Q_8$. How the pruning metrics are effective to reduce the performance variance?
- $Q_9$. How the efficiency of the heuristic pruning metrics differs in terms of time and space complexities?

## 1.2 Objectives

As it has been aforementioned, the contribution of this thesis is related to the design of MCS. To make an intersection between IS, MCS, and SI, with the aim to increase the efficiency and the efficacy of classifier ensemble systems. Furthermore, to analyze the popular and the recent classifier ensemble pruning metrics. Based on the formulated research questions, the following objectives should be achieved.

- **Objective 1**. To build more diverse and highly accurate MCS, only from a reduced portion of the available data. *This objective corresponds to research questions $Q_1$ and $Q_2$.*

- **Objective 2**. Increasing the efficiency of MCS and going beyond what can be achieved from ensemble pruning methods. *This objective corresponds to research questions $Q_3$ and $Q_4$.*

- **Objective 3**. Grouping and analyzing fast and accurate heuristic metrics for MCS pruning. *This objective corresponds to research questions $Q_5$ to $Q_9$*

## 1.3 Contributions

The research carried out during this thesis aims to contribute to the scientific community by designing heuristic and optimization-based ensembles. Consequently, small-size and effective ensembles could be designed. Below, the main contributions of this thesis are briefly discussed:

- **Training set selection and swarm intelligence:** To alleviate the computation complexity associated with training large-size ensembles. In this proposal, we prove the possibility of building MCS from a reduced portion of training data. While to reduce the randomness of data sampling, intelligent data sampling in the form of instance selection is used. Regarding that, the ensemble members could focus deeply on the most informative data samples during the train. Furthermore, the combination function could be optimized via the search capabilities of SI algorithms. In this case, a class-specific weight is assigned to each classifier based on its accuracy to predict different samples from each class.

- **A guided search for ensemble pruning:** The small-size ensembles have benefits; save the testing time, reduce memory burden, reduce the communication costs of distributed models, and improve the prediction accuracy. While it is not easy to prune from overproduced ensembles. In this proposal, we discuss efficient and critical heuristic metrics for ensemble pruning. In addition, a guided search is presented for handling the conflicting of those metrics to enhance the general accuracy of subset integration. The pruning process starts after the individual's accuracy and the ensemble diversity are

captured in a preliminary search set. In connection with our first contribution, the efficiency of ensemble systems can be further improved during the learning phase, by fast learning from reduced samples, and during the testing phase, by reducing the ensemble size via guided search. For that, our contribution is to downsize the training data and to downsize the classifiers simultaneously without affecting the performance of MCS.

- **An analysis of heuristic metrics for classifier ensemble pruning:** MCSs are superior to any random single classifier. However, three main defects are reported for those systems; (1) A large pool of classifiers should be built, (2) A sufficient memory space should be available to store those models, and (3) A large classification time will be consumed for combining multi-decisions. To alleviate these drawbacks, we discuss the concept and the benefits of thinning/pruning ensemble of classifiers. An effective, fast, and implementable heuristic metrics are analyzed to reorder the classifier's position in the generated random bagging. The investigated metrics are based on modifying the order of the classifiers in the bagging algorithm with the selection of the first set in the queue. Some of these criteria include general accuracy, complementary decisions, ensemble diversity, the margin of samples, minimum redundancy, discriminant classifiers, and margin hybrid diversity. The efficacy of those metrics is affected by the original ensemble size, the required subensemble size, the kind of individual classifiers, and the number of classes. While the efficiency is measured in terms of the computational cost and the memory space requirements. The separate performance of those metrics is assessed over binary and multiclass benchmark classification tasks, respectively. In addition, the behavior of those metrics against randomness is measured in terms of the distribution of their accuracy around the median.

**Publications:** During the research activities of this thesis, several international peer-reviewed journal and conference articles were published to disseminate the obtained results. The publications can be found in Table 1.1.

**Table 1.1:** Publications in journals and conferences conducted during this thesis.

| |
|---|
| **Title:** Vertical and Horizontal Data Partitioning for Classifier Ensemble Learning.<br>**Authors:** AM Mohammed, E. Onieva, M. Woźniak.<br>**Congress:** The 11th International Conference on Computer Recognition Systems, 2019, Poland. |
| **Title:** Training set selection and swarm intelligence for enhanced integration in multiple classifier systems.<br>**Authors:** AM Mohammed, E. Onieva, M. Woźniak.<br>**Journal:** Applied Soft Computing (Impact Factor = 5.472 $\rightarrow$ Q1).<br>**Status:** Published. |
| **Title:** Selective Ensemble of Classifiers Trained on Selective Samples.<br>**Authors:** AM Mohammed, E. Onieva, M. Woźniak.<br>**Journal:** Neurocomputing (Impact Factor = 4.438 $\rightarrow$ Q1).<br>**Status:** Under review. |
| **Title:** An Analysis of Heuristic Metrics For Classifier Ensemble Pruning Based on Ordered Aggregation.<br>**Authors:** AM Mohammed, E. Onieva, M. Woźniak, G Martínez-Muñoz.<br>**Journal:** Pattern Recognition (Impact Factor = 7.196 $\rightarrow$ Q1).<br>**Status:** Under review. |

## 1.4 Research Methodology

The research field of this thesis is moving fast due to technological advances and the continuous generation of new contributions in ML. Consequently, an iterative research methodology was followed. The main idea of this cyclical process is that the knowledge acquired in its initial phase helps to design an increasingly promising technique, either in terms of accurate results, or in terms of concept,

offering originality and remarkable contributions. Figure 1.1 shows the different phases of this research methodology. These phases are briefly described below:

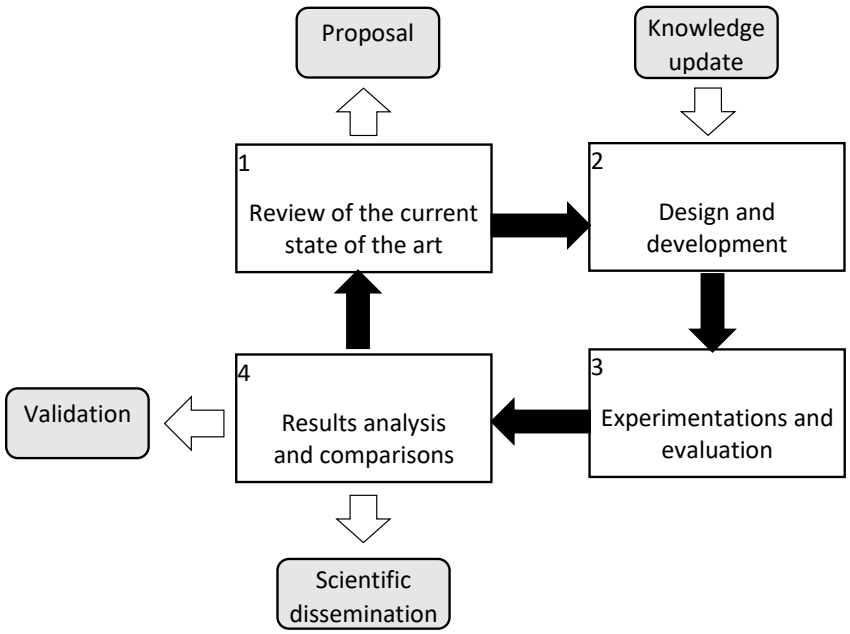


**Figure 1.2:** The research methodology of the thesis.

1. **Review of the current state-of-the-art:** The main objective of this phase is to investigate the state of the art related to the field under consideration to identify problems and/or challenges. To achieve this, the related bibliography is used, reviewing publications from the scientific community published in journals, and proceedings of international congresses. The knowledge acquired in this phase should lead to a proposal to address the identified challenges.

2. **Design and development:** In this phase, a novel proposal to solve the identified challenges is designed and developed. To this end, previously acquired or updated knowledge is used to ensure that the solution is always up-to-date with the current state of the art.

3. **Experimentation and evaluation:** The goal of this phase is to test the proposals resulting from the previous step to a process of experimentation. To

carry out this procedure, it is crucial to provide some criteria and evaluation methods with which the results will be compared in the subsequent phase. All these criteria and methods must be built using the knowledge acquired in the first stage of the methodology.

4. **Results analysis and comparison:** After carrying out experimentation, results must be analyzed and compared with those obtained in the state-of-the-art. At this point, it is needed to check if the results obtained are enough to address the challenges identified in the first phase. In such a case, another methodological cycle begins to approach the following challenge identified or to keep working with the challenge under consideration if it was not still solved. In this stage, conclusions must be drawn from analyses of results, and knowledge obtained must be materialized in scientific dissemination, either through journals, or conferences.

## 1.5 Research Context

This research has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement Nº 665959. This research has also been funded and supported by Deusto Smart Mobility Research Group, DeustoTech-Fundaci´on Deusto, the Faculty of Engineering at the University of Deusto, (Spain). In addition, the Department of Systems and Computer Networks, Faculty of Electronics, Wrocław University of Science and Technology, (Poland).

During the second year of the Ph.D., an international research stay was made as part of the research activities. The research stay was carried out at Wrocław University of Science and Technology (WUST) within the Department of Systems and Computer Networks. The stay lasted three months, from April to July 2019, under the supervision of Dr. Michal Woźniak. He is a supervisor of this research and a full professor at Faculty of Electronics. The objective of the research stay was to collaborate with international experts in the field of ensemble learning to share knowledge and get feedback from them. Therefore, it was possible to add

more to the experiment design through several suggestions that later end with the dissemination of a scientific journal article.

In addition, participating in the 11th International Conference on Computer Recognition Systems (CORES), 2019, Poland, with a presentation of some results. Besides, participating in the 7th international student workshop, June 2019, Ladek Zdroj, Poland.

## 1.6 Structure of the Dissertation

The structure of the remainder of this thesis dissertation is outlined below.

**Chapter 2** reviews background and related work about ML, supervised data mining, and ensemble data mining. In addition, we present a taxonomy of MCS. Furthermore, we discuss the importance of diversity and how to measure it. Finally, closing the chapter with the importance of soft computing.

**Chapter 3** reviews state-of-the-art ensemble algorithms. This includes bagging-like, boosting, and gradient boosting ensembles. The revision includes different mechanisms to promote diversity in each algorithm. Following that, we group all the ensemble methods in a comparison table to show their properties. Finally, we spotlight on the role of metaheuristic algorithms (MA) to enhance the performance of MCS.

**Chapter 4** presents our first proposal to build an effective MCS from a reduced portion of the training data, and how to combine multi-decisions via SI algorithms. This chapter is therefore aligned with *specific objective 1*.

**Chapter 5** provides our second proposal to prune MCS via guided search. We discuss how to merge ordering-based pruning metrics to gain what cannot be reached from any of them. The work presented in this chapter is therefore directly related to *specific objective 2*.

**Chapter 6** presents an analysis of fast and accurate heuristic metrics for MCS pruning. The analysis shows how the performance of ensemble pruning metrics could be affected by the original ensemble size, the required subensem-

ble size, the kind of individuals, and the number of classes. The work is directly related to *specific objective 3*.

**Chapter 7** revisits the main goal and specific objectives posted earlier. In this chapter, we summarise the main contributions of this thesis and outline possible future research.

*The good thing about science is that it's true whether or not you believe in it.*

Neil deGrasse Tyson

CHAPTER

# 2

# Background

This chapter is intended to introduce the basics of machine learning, data mining, classifier ensemble learning, and soft computing. First, machine learning is defined, why it is important, how the machines extract the pattern, and research ethics when dealing with automated algorithms. Then, the importance of data preprocessing is highlighted to prepare and reduce the training data. While, the next section is dedicated to differentiate between data mining and machine learning, and to introduce supervised learning algorithms. Ensemble data mining, the benefits of ensemble learning, the taxonomy of MCS, and diversity measuring metrics will be presented next. Finally, the importance of soft computing, and how soft computing techniques can be incorporated in ensemble learning is to be presented in the last part of this chapter.

## 2.1 Machine Learning

The invention of artificial intelligence (AI) enabled machines to outperform humans in specific scenarios. The main goal is not to create an artificial brain, but to assist us to understand the world's massive data [39]. Today we are surrounded by a vast amount of data that is intractable for a human to understand [19]. The revolution in the information field, due to powerful storage and communication, and

the invention of electronic sensors increased the volume of recorded data. Our lives are recorded in databases; (e.g. weather conditions, traffic status, human behavior, bank transactions, medical diagnosis, network communications, and more). With this data, it became necessary to find a systematic way to get potential and important actions. This was the start of Machine Learning, as it is defined in [39] as "*The development of computer algorithms to transform data into intelligent action*". Figure 2.1, shows the three main components that form the cycle of advancement in this field. The growth of data pushed the development of computing and storage power, which in turn spurred the development of statistical and computing algorithms. In the literature, there is an agreement that even with the capabilities of computers to find patterns in large databases, their power is limited to the novelty and the motivation of the analysis which is directed by humans [39–41].

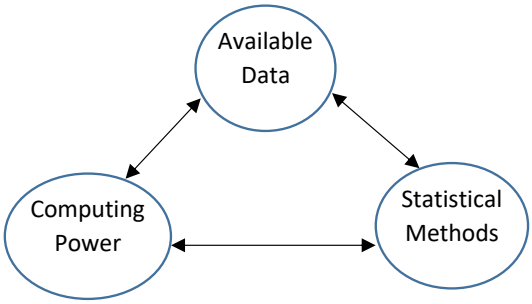


**Figure 2.1:** The development cycle of Machine Learning.

ML is more successful when it can be used to assist rather than replacing human experts; e.g. assist programmers to identify spam messages, assist engineers to optimize energy usage in homes, assist politicians to predict election outcomes, assist doctors to eradicate cancer, assist biologists to discover genetic sequences linked to the disease, assist policymakers to reduce the fraudulent credit card transactions, assist engineers to design self-driving cars, and more. Contrary and as part of their limits, computers are less flexible to extrapolate beyond the strict criteria learned. In addition, the ML algorithm is only as good as the data it learns from. If the input data does not contain an implicit context, the behavior of computers will be like humans; best guess.

The main purpose is to benefit from the computer experience to solve similar experiences in the future. Figure 2.2 shows the four interconnected parts of any

learning process to answer the following questions: how the experience can be formed?, how learning can be transferred and understood by computers?

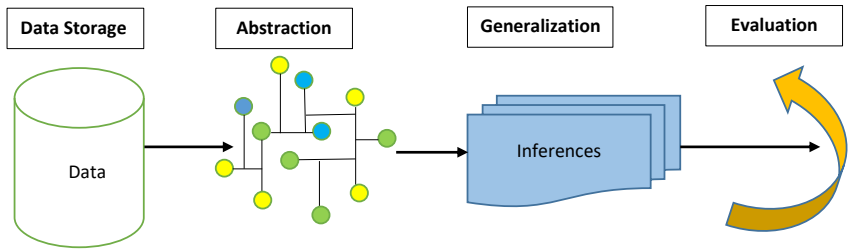


**Figure 2.2:** The four components of how to learn.

1. **Data Storage:** The main place to store data for the purpose of short-and long-term recall. Those data are represented as ones and zeros, with no meaning, on disks or random access memory (RAM). Memorizing the data can not be sufficient for the learning process, but it can be useful for reasoning; based on similar stored cases.

2. **Abstraction:** This phase is known as the knowledge representation or model training to assign meaning to stored data. The explicit description of the patterns within the data can be represented by different forms; e.g.(mathematical equations, interconnected diagrams like trees and graphs, logical IF-THEN rules, data clusters). For that, the original information can be summarized in a simple form with a discovery of unseen relations among data. No new data will be generated, only the original information can be seen from different perspectives of the representation form.

3. **Generalization:** The process of using the abstracted model for future actions on tasks which are similar to what has been seen before. Many discovered patterns can be noticed during the training phase, the most relevant pattern among data will be promoted as the most useful inference. For that, the model bias, which is the wrong in the prediction, usually will exist; as humans who are biased to take decisions based on past learned information. The uncorrelated bias (error) is a big benefit in ensemble learning, as we will discuss later.

4. **Evaluation:** Each model has its own bias which is inherited form the learned pattern. According to that, each model has its strengths and weaknesses. The final phase in learning, is to measure the model success in terms of bias to check if further training is needed. The evaluation is done to judge on the generalization capability for new, unseen, dataset. Finally, a model with a good performance during training, but with a poor evaluation, is said to be overfitted model.

It is worth mentioning the ethics of machine learning and artificial intelligence. An automated algorithm working on emotionless devices can cause unintended consequences [39]. The inference of those tools could be biased to racial, ethnic, and religious information in the dataset [39]. For that and according to the application, that information should be blinded in data before training. It is important to have a clear understanding of; what we are doing?, why it is necessary to do it in an automatic way?, and how the implemented algorithm works?. In addition, some tools could help; like "FairML" [42] to know which algorithmic inputs could cause harm or bias. Finally, the user privacy and the behavior which is recorded through the web cookies should be respected. The availability of data does not give us the right to analyze it.

In addition, the prediction result of an AI system could be explained in terms of the input data. The model-agnostic, post-hoc explainable technique, can be applied to any ML model regardless of its internal representation or process. Basically relies on the following sorted techniques according to their popularity:

- *Explanation by simplification:* A new simplified model can be generated, keeping the similar performance score, while reducing the complexity. Popular techniques are local interpretable model-agnostic explanations (LIME [43]), generic rule extraction methods (G-REX [44]).
- *Feature relevance explanation:* The opaque model can be explained by measuring the influence of each internal feature on the predicted output. Profitable contributions include: Shapley additive explanations (SHAP [45]) as a coherent method for describing the performance of any type of machine learning. As well, sensitivity analysis [46] to quantify the importance of the

input variable based on partial derivative, but maybe sub-optimal for explaining AI and with several drawbacks [47].

- *Visual explanation:* Portfolio of visualization techniques to explain the opaque model can be founded here [48]. Furthermore, the use of individual conditional expectation plots (ICE [49]) for visualizing the estimated model by any supervised learning algorithm.

## 2.2 Data Preprocessing

Data preprocessing [19] includes data preparation; (e.g., integration, cleaning, normalization, transformation) and data reduction; (e.g., instance selection, feature selection, discretization). The desired result is to get a cleaned, relevant, manageable, and meaningful dataset ready for analysis. Usually, there is a trade-off between time-complexity and accuracy to get the prepared data, which keeps the research ongoing for this area.

**Data Preparation:** Refers to a set of techniques to prepare data as an input for a certain DM algorithm. Usually, it is ignored by inexperienced practitioners, which may cause the model runtime crash. Even if the algorithm works, the expected results will not be optimistic. A set of preliminary steps can be followed before model training as described in [19]:

- **Data Cleaning:** The process of detecting and correcting (or removing) inaccurate records from data. Other tasks could be to detect irrelevant data fragments that do not make sense. The result will be a consistent, accurate, meaningful dataset [50].

- **Data Transformation and Data Integration:** Data transformation is the process to convert and consolidate the data to another format to improve model efficiency. This process is composed of sub-tasks; feature construction, feature aggregation, normalization, and more. While data integration is recommended for merging data that come from multiple data sources. The aim is to detect conflicts and to remove redundant and inconsistent data [51, 52].

- **Data Normalization:** This process is dedicated to unify the measurement unit to all the attributes. Under this schema, all the attributes are equally weighted. Statistically, to align the entire probability distribution of the adjusted values, to reduce the effects of certain gross influences [53, 54].

- **Missing Data Imputation and Noise Identification:** Training a model with a dataset that has many missing values can have a dramatic effect on the quality of the machine learning model. The imputation strategies can be (Mean/Median values, most frequent value, k-nearest neighbors, using deep learning, multivariate imputation by Chained Equation) [55]. Noise identification is known as smoothing, to detect variances or random errors in a measured variable [56].

**Data Reduction:** Not like data preparation, data reduction is an optional step. It provides a set of methods for obtaining a reduced version of the original data. It is the process of downsizing the data while maintaining the integrity of the complete dataset. However, it could be a crucial step as data preparation, to enable the DM algorithm when the data size exceeds. Following are some representative approaches:

- **Feature Selection:** Data reduction can be accomplished by removing irrelevant or redundant attributes. The aim is to use the least number of features while keeping the output of the classification as similar as possible as if we were using the full feature set. Regarding that, the training speed can be boosted and the model performance can be elevated [57, 58].

- **Feature Extraction:** Lessening the amount of resources required for the representation of a large array of data. Analysis with a wide range of variables usually involves a large amount of memory and computational resources. In addition, the classification algorithm could generalize to new samples badly. Feature extraction is a set of techniques that create combinations of variables to fix these issues, but also reflecting the data with sufficient precision. Many machine learning specialists believe that the properly configured extraction of features is the key to a successful model building [59].

- **Instance Selection:** The process of reducing or eliminating samples intelligently without affecting the DM application. This process can be guided by heuristic rules to select horizontal subsets of data [60, 61]. In addition, the process can be applied to adapt with a particular DM algorithm; like "selection of support vectors for support vector machine algorithm" [62].

- **Discretization:** The mechanism by which quantitative data is converted into qualitative data; via converting numerical variables into discrete or nominal variables. For that, a huge spectrum of numeric values can be compacted or reduced into a subset of discrete values [63].

Finally, the benefits of data preprocessing can be among the following; (1) Adaptation to a particular machine learning algorithm. (2) Increasing predictive accuracy. (3) Enabling: data mining algorithms are negatively affected by the data size, and data reduction provides a solution for data choking. (4) Cleaning noisy, missing, and redundant data to improve data quality. (5) Focusing: to focus on relevant data instead of all available information.

## 2.3 Data Mining

A closely related field to machine learning is Data Mining, which is defined in [39] as, "*the generation of novel insights from large databases*". While in [19]; data mining is defined as, "*solving problems by analyzing data present in real databases*". While others [40, 41, 64, 65] view DM as, "*the main steps of knowledge discovery in databases*". The main distinction between ML and DM is that; ML is dedicated to teach computers how to use data to solve problems, while DM is dedicated to teach computers how to identify patterns like a human to solve problems. It is interesting to mention that every DM involves the use of ML, but not all ML involves DM. Next, a brief description of data mining categories is presented:

- **Unsupervised Learning**: Those techniques deal with an unlabeled dataset, with a minimum of human intervention [66]. Data with no pre-existing labels at our disposal and the purpose is to find associations, relationships,

regularities, and similarities in the data. Cluster analysis is among the common models used in unsupervised learning. Cluster analysis is the process to segment/group datasets with shared attributes [67]. Unlabeled examples are given an implied cluster label from the relationships within the data entirely. Sometimes, the clustering task is referred to as "unsupervised classification", because it classifies unlabeled examples.

- **Supervised Learning:** Supervised learning is popular in the field of DM, commonly knows as prediction methods. In supervised learning, a relationship is to be learned between input space and target space. Based on the predicted target type, there are two common tasks; regression and classification. In regression, the numerical output to be predicted falls in a certain interval. Contrary to classification, the domain of the target is finite and categorical.

Next, all the subsequent sections will be dedicated to the classification task in matching with the core of this thesis. In the classification problem, the input attributes (features) and the target attribute (class) are transparent. The aim is to learn a function that maps inputs to outputs. The learned function is called a model, and it is inferred from labeled training data. Let,

$$\mathbf{x} = \left[x^{(1)}, ..., x^{(d)}\right]^T, \text{ and } \mathbf{x} \in \mathcal{X} = \mathcal{X}^{(1)} \times ... \times \mathcal{X}^{(d)} \tag{2.1}$$

where $\mathcal{X}$ denotes feature space and $\mathbf{x}$ is the sample, i.e., $\mathbf{x}$ is the so-called feature vector which informs about attribute values. We will assume that we have $d$ attributes at our disposal. The supervised classification model will assign a given object described by its features, $\mathbf{x}$, into one of the predefined categories, also called labels. Let $\mathcal{M} = \{1, ..., M\}$ stands for the set of class labels (decision regions). The classification algorithm (discrimination algorithm) is any learned function $\Psi$ with domain $\mathcal{X}$ and codomain $\mathcal{M}$ as clarified in Equation (2.2). Where the target values in codomain are finite and categorical.

$$\Psi \ : \ \mathcal{X} \rightarrow \mathcal{M}. \tag{2.2}$$

There is a large number of models with different inferring strategies to discover the hidden patterns from data. Next, I provide a short review of popular classification algorithms according to the division in Figure 2.3, inspired by [19].

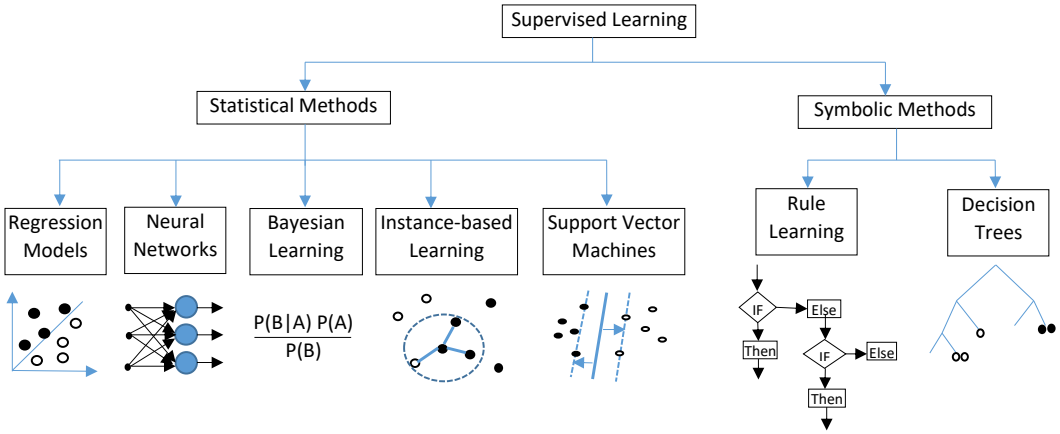


**Figure 2.3:** Supervised classification algorithms.

- Regression Models: Logistic Regression (LR) is a statistical model, that returns a probability for each class level. The cutoff value is embedded to separate the upper and the lower probabilities to work as a binary classifier (binomial LR). While multinomial LR deals with more than two classes. In order to be enabled, the missing values should be handled. In addition, the high correlation among the predictors (variables) should be minimized. LR has been praised for its robustness and simplicity [68].

- Artificial Neural Networks (ANNs): The excessive and growing formulations of ANNs from the theoretical and algorithmic depth [69], made them more influence on the field of pattern recognition. ANNs handle large and complex tasks due to their nested non-linear structure. While non-explanation is one of the pitfalls of those black-box models. The computations of those models are based on the definition of neurons. ANNs are unstable and more sensitive to small changes in training data. Similar to LR, they require no missing values.

- Bayesian Learning: Based on the probability theory to get rational decisions [70]. The Naïve Bayes is the most popular algorithm in this category. The posterior probability for each class label is calculated, then the decision is promoted upon the maximum probability returned. Those methods only work with categorical attributes, cannot work with missing values, and are

very sensitive to redundancy. The ”Naïvety” comes from the assumption of the conditional independence between features. It can be viewed as an explainable model to give reasoning about the decisions.

- Instance-based Learning: The prediction of a new unknown sample is based on a distance function with the past stored samples. Also called memorization techniques and lazy learners [71]. The performance of those methods is affected by the used distance function, neighborhood size, and decision aggregation mechanism. K-Nearest Neighbor (KNN) is the most popular method in this category. Pitfalls of those methods can be mentioned as: high memory space for storage, delayed prediction response, sensitivity to noise.

- Support Vector Machines (SVM): Learning algorithms which are based on maximizing gab separation (margin) between different class samples to get correct decisions. They are suitable to work as linear and non-linear data separation. Only the class borders (support vectors) are important to optimize the margin where internal points can be removed to improve the efficiency [72]. They require no missing values and are commonly robust against noise.

- Rule Learning: Called divide-conquer algorithms [73]. The data parts are divided based on one rule, then recursively conquer the divided parts. Those models are transparent or explainable to nonexperts in the form of logical structures. Available features are analyzed to find homogeneous groups, then an additional rule is built to drill down more. Small changes in the training data results in decision change. Rule learning techniques are affected by missing values, noisy samples, and outliers.

- Decision Trees (DT): This is a kind of indirect rule learning. Uses structure branching decisions to model the relationships among the features and the predicted class value. They are widely used and can model any type of data. The human-readable model is appropriate in applications where legal reasoning is required. Those models are vulnerable to overfitting, and the internal parameters should be tuned [59]. Unstable like ANNs and sensitive to change in the training data. Usually, they are biased towards the splits on features.

For unseen pattern **x**, a class membership values are calculated as in Equation (2.3), then the labeling choice will be connected to the highest score.

$$\Psi(\mathbf{x}) = \text{Max}\{g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_M(\mathbf{x})\} \tag{2.3}$$

The decision region $R_1$ for the $1^{st}$ class, is the set of points for which $g_1(\mathbf{x})$ has the highest score [15]. While the *classification boundaries* contain data points for which the membership values tie. If the decision region $R_1$ contains data points from the $2^{nd}$ class, then we have overlapped classes. From Figure 2.4. (c) and as shown in [74], the regions are nonoverlapping as the model learns all the details about the data. This case is known as overfitting, and the model will not perform properly to predict unseen samples. While Figure 2.4. (a) shows the optimal class separation boundary that guarantees minimum possible error with the future samples. Finally, Figure 2.4. (b) shows the the underfitting case when the model fails to capture relationships between a dataset's features and a target variable during training.

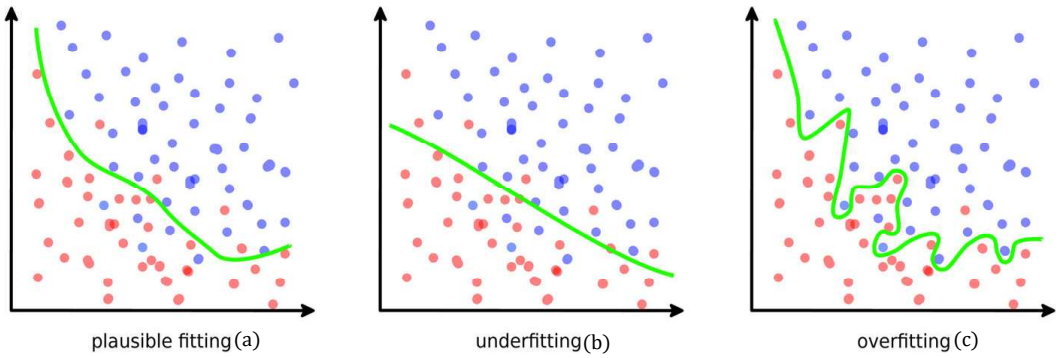


**Figure 2.4:** Trade-off between overlapping and overfitting, taken from [74].

Each model has an accompanied error, we need to understand the different sources that cause this error. Equation (2.4) represents the compound generalization error $E_G$ of a classifier $\Psi$ that is trained on dataset $D$.

$$E_G(\Psi, D) = E_A(D) + E_M + E_B \tag{2.4}$$

where $E_A(D)$ is the "approximation error" represents the variance due to using different training data, or non-deterministic training algorithm. Clarified as, the

hyper-parameters of the model that affect its performance. The second term $E_M$ is the "model error" which represents the bias due to selecting a model in preference of another. The last term $E_B$ is the "irreducible error" coming from the insufficient representation of the data. This is commonly known as Bias/Variance Tradeoff [15, 59]. Increasing a model's complexity will typically increase its variance and reduce its bias. Conversely, reducing a model's complexity increases its bias and reduces its variance. This is why it is called a tradeoff.

## 2.4 Ensemble Data Mining

Ensemble learning is the strategy of using multiple learning algorithms in order to obtain greater predictive precision than all of the constituent learning algorithms alone [6–14]. In addition, Wozniak. et al. defined ensemble models as hybrid intelligent systems [16] with the potentiality to cope with ambiguity, uncertainty, and complex problems. Thanks to their capabilities, ensembles received great attention in the applications related to data mining. For unsupervised learning, clustering performance could be significantly improved by ensemble methods [75–77]. Furthermore, ensembles are employed for unsupervised anomaly detection [78]. While, for supervised learning, those systems are widely popular for regression tasks [27, 79] and for classification tasks [9, 16].

Ensemble systems for pattern classification have been expanded in the literature under creative names as: consensus aggregation [80], stacked generalization [81], committees of neural networks [82], mixture of experts [83, 84], classifier ensembles [85, 86], classifier selection [28, 87], multiple classifier systems [16, 88], classifier fusion [89] and more. Theoretical and empirical studies prove that ensemble systems are more accurate than any random classifier [13, 88, 90, 91]. The final decision is the accumulative decisions of all the classifier set. Let $\Pi$ denotes a pool of $T$ base classifiers $\Pi = \{\Psi_1, \Psi_2, \Psi_k..., \Psi_T\}$ to be grouped by a combination function. The ensemble output $\hat{\Psi}$ is determined on the basis of the outputs of the base classifiers, i.e.,

$$\hat{\Psi}(\mathbf{x}) = F(\Psi_1(\mathbf{x}), \Psi_2(\mathbf{x}), ..., \Psi_T(\mathbf{x})) \tag{2.5}$$

*Intuitively, any classifier ensemble is in fact a classifier* (L. Kuncheva [15]).

Each learning algorithm, Section 2.3, has a limit to discovering the hidden pattern. According to Wolpert's *no free lunch* theorem [92], there is no best classifier suitable for all problems, but each model has its own area of competence giving the design assumptions. For that, a set of learning models solving the same problem can be consolidated to generate a better composite global model [93]. L. Kuncheva [15] stated *"the improvement of the ensemble over the single best classifier or even on the average of individual classifier accuracies is not guaranteed"*. From our perspective, the proper design of ensemble is conditioned by outperformance over the best individual classifier in the group.

### 2.4.1 Are we pursuing complexity?

Why we accept complex systems instead of depending on a single classification algorithm?. The answer to this question is highlighted in the following points.

- Ensemble models are the solution to deal with uncertainty. A solution from a single classifier can be boosted and trusted by aggregating a group of predictors (*wisdom of crowds* [14]).
- There is no guide to design universal approximators, perfect model, i.e. it is difficult to set up ANNs or reaching their optimistic parameters.
- Ensemble selection or pruning is an interesting research topic that aims to reduce ensemble complexity without deterioration in the performance.

In addition, a promising solution via ensemble learning can be achieved for the following scenarios:

- Imperfect Learning: Non-deterministic classifier can be considered as a local optimizer in terms of the training error, a safer option is to group several models that cover the solution space properly.
- Too much data: We are surrounded by too much data. In this case, data is split into chunks where similar or different learning algorithms can be trained on each part independently. Ensemble learning support parallelization and distributed computing for handling this scenario efficiently.
- Small-size data: In data shortage, stratified sampling with replacement can be applied, where several data replications will be obtained to train individual classifiers inside the ensemble.

- Data fusion: The data pattern can be identified differently based on the data source. The availability of sensors strengthens decision making by analyzing different features. Instead of fusing all features and building a single classifier, it could be better to build a single classifier for each feature space and combine their decisions.
- Complex hypothesis: Complex classification boundary can be approximated by combining several base classifiers.

### 2.4.2 A Taxonomy of Classifier Ensemble Methods

A comprehensive review of the classifier ensemble methods, thorough discussion, and the development of further knowledge in this area was the core of many articles [13, 28, 94–97], with a proposed taxonomy by L. Rokach [98] who indicated the five dimensions to design this kind of powerful models. Figure 2.5 shows our perspective for the Multiple Classifier System (MCS) taxonomy as it has been proposed by L. Kuncheva [15], R. Cruz et al. [28], and L. Rokach [98]. The two main phases in classifier ensembles are; Generation and Integration, while the selection is an intermediate/optional phase.

#### 2.4.2.1 Generation

The goal in this phase is to generate a pool of classifiers that are both diverse and accurate. This phase discusses strategies to handle data horizontally/vertically. In addition, what classifier type to accommodate, how to build the classifiers: dependent/independent manner, and how many classifiers to train (ensemble pool size).

**1) Diversity:** Diversity is one of the main reasons for the effectiveness of ensemble methods [18, 94]. Diversified classifiers cause uncorrelated errors, which lead to improved classification accuracy [99]. In general, this can be achieved through the following six wide subcategories to promote diversity during the generation phase.

- **Different Parameters/Initialization:** *Algorithm-level diversity*; via different parameters, the base classifiers can be generated by modifying the hyperparameters of the learning algorithm; for example, controlling the number of

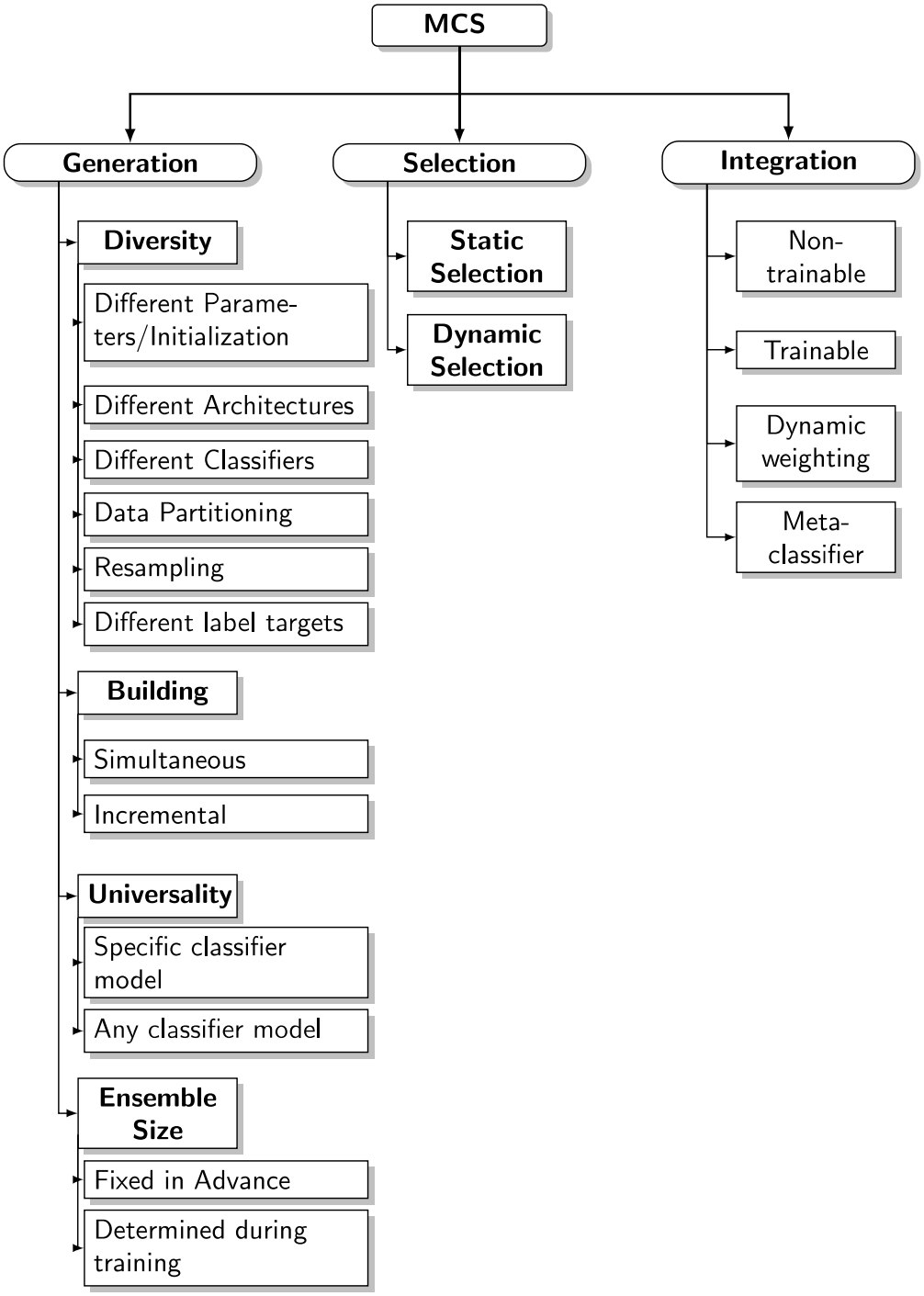


**Figure 2.5:** The Taxonomy of Multiple Classifier System (MCS).

$k$ neighbors, distance function in KNN model, and controlling the confidence level parameter in the decision tree. While, via different initialization and if the training process is initialization dependent, the model will be sensitive. In neural networks, different initial configurations of weights result in different decisions [100].

- **Different Architectures:** *Algorithm-level diversity*; this strategy is more suitable to multi-layer perceptron neural networks, where the number of hidden layers, the number of neurons, and the network topology affect the classifier domain space. For example; in the Addemup algorithm [101], the genetic algorithm is used to choose the required network topology to compose the ensemble according to a measure of diversity.

- **Different Classifier types:** *Algorithm-level diversity*; each classifier model build its inference with a capability to discover a hidden pattern differently. Each model, Section 2.3, contains explicit or implicit bias that leads to a better generalization accuracy through combination. In addition, as in [102], the divide and conquer mechanism is best implemented by calculating the local accuracy in the feature space to choose between four different classifier types.

- **Data Partitioning:** *Data-level diversity*; Partitioning means the division into smaller disjoint components of the initial training. A different classifier will be trained on each part to gain different and accurate decision [103]. This mechanism enables data mining algorithms to handle massive datasets by managing memory size and computational resources perfectly. The data can be partitioned horizontally or vertically. *In horizontal partitioning*, several parts will be formed; each part will contain sub-samples while sharing the complete feature set. Sub-samples can reflect the entire dataset by selecting the instances from all the formed clusters of the dataset; known as cluster-based concurrent decomposition (CBCD) [104]. The mixture of experts (ME) [105] splits the input space into several subspaces and assign an expert, classifier, to each subspace. In [106] a decision tree framework is employed to divide the input space into mutual exclusive partitions, then a

new unseen pattern will be classified by a dedicated classifier that is learned from the space to which the instance belongs. *In vertical partitioning*, The feature space is partitioned into several subsets keeping the same number of samples, then each classifier can be trained on a different projection. This mechanism is more suitable for a high-dimensional dataset without affection by the feature selection drawbacks [107], and the accuracy could be improved by the less correlation among classifiers. This division results in a high-speed classification algorithm [108], and solves the problem of class under-representation that exist with instance-based sampling.

- **Resampling:** *Data-level diversity*; the diversity could be promoted by manipulating the training set. Each base model is trained over a different sample from the training. Bagging [3, 109] and Boosting [5, 110] are the popular strategies in that paradigm. Bagging ensembles achieve a reasonable diversity level by creating different bootstrap samples to train each base model independently, then the final decision is adopted by a simple majority voting-based aggregation. Moreover, the non-sensitivity of bagging and robustness under diverse noise conditions makes it more attractive [111]. Contrary to sequential ensembles, *Boosting*, the individual members are generated in the sequential schema by the learning algorithm [5]. The sequential mechanism of boosting encourages the complementariness between ensemble members, by focusing on previously misclassified samples. However, the performance in boosting is more sensitive to noisy samples [111, 112] and sometimes overfitting can be observed for large pool size [113].

- **Differnet label targets:** *Data-level diversity*; manipulating the target attribute is very interesting mechanism to promote diversity. Instead of building complex classifier, several classifiers with usually simpler representations, about the target attribute, will be trained. For instance, to handle the multi-class dataset, the original target attribute can be replaced by a simpler and smaller target domain. Among those strategies, One versus all (OVA) [114] which divides the $M$-class classification problem into $M$ two-class classification tasks. While one versus one (OVO) [115] divides the $M$-class classification problem into $M(M-1)/2$ two-class classification tasks, so

the complex decision boundary can be simplified. Minimal classification method (MCM) [116] converts the $M$-class classification task to the minimal binary classification tasks. MCM requires $\log_2(M)$ classification tasks in the form of a separation between groups of multiple classes. Furthermore, the error-correcting output coding (ECOC) algorithm, uses a code matrix to decompose the multiclass problem into multiple binary problems [117]. In addition, label switching algorithms [118, 119] change the labels of samples picked randomly.

**2) Building:** This part discusses the building schema to generate a pool of classifiers, whether to be dependent or independent. In the dependent framework (sequential/incremental), the performance of a classifier affects the creation of the next classifier in the chain. Instead, in the independent framework (simultaneous), the classifiers are generated independently.

- **Simultaneous:** The independent methods to generate a set of classifiers mainly depends on forming different training samples from the original training set. The training samples could be mutually exclusive (disjointed) or overlapping. The reason behind preferring this schema is to improve the predictive accuracy or to speed up the generation step as this schema supports parallel implementation. *Bagging* (Bootstrap AGGregatING) [3] is the popular method in this category, with a replacement from the original training set, each classifier is trained on data samples. The sample size will be equivalent to the original training size; randomly some samples will be duplicated and some samples will be ignored.

- **Incremental:** Sometimes called dependent, there is a kind of interconnection during the learning process. The model generation depends on the accuracy of the previous model/committee. The training process is done in iterative form, where the learning process will be directed to focus on the previously misclassified samples. *Boosting* [120] (also known as arcing- Adaptive Resampling and Combining) is the popular method in this category. The training process mainly depends on assigning weights to all the training samples. In the beginning, all the samples will be equally weighted (having the same

importance), but throughout the iterations, the weights of correctly classified samples are lowered while the weights of misclassified samples are raised. As a consequence, the base model is directed to focus on the hard samples in the training set.

**3) Universality:** This part discusses the universality of the ensemble model. Some ensembles techniques could be designed to work for any classifier, while other ensembles have been designed to work with specific classifiers. In other meaning, the relation between the ensemble technique and the used classifier type.

- **Specific Inducer:** Known as inducer-dependent ensembles, where the effectiveness of the ensemble could be degraded if applied for other classifier types. For example, [100, 115] those ensembles are explicitly designed for neural networks. Additional schemas [121] are perfectly suited for SVM.

- **Any Inducer:** Known as inducer-independent ensembles, Those implementations can be extended to a wide variety of classifier types without affecting the generalization accuracy.

**4) Ensemble Size:** This part discusses the aspect of pool size. How many classifiers should be trained?. The main four factors that concern the ensemble size as determined by L. Rokach [98] are; (A) *Sufficient Accuracy*; the appropriate accuracy of the ensemble could be reached by aggregating 10 models [100], while other empirical studies show that this level of accuracy is correlated with large-size ensembles containing 25 models [8]. (B) *Computational Cost*; the more classifiers are generated the more computational resources are consumed. As a consequence, the user may predetermine the pool size to match the computational cost limits. (C) *Nature of Task*; the ensemble size could be problem-dependent, as we stated before with the ensemble size of OVO and OVA strategies for handling multi-class classification tasks. (D) *Number of processor cores*; for independent ensembles, the number of internal cores can be the upper bound to control how many classifiers can be trained in parallel mode.

- **Fixed in Advance:** This is the simple form to predetermine how many iterations should be considered. Most Bagging and Boosting software packages give the flexibility to control the number of iterations.

- **Determined during Training:** The best ensemble size can be determined during train time. The contribution of a new classifier to the ensemble performance is to be checked if it is significant or not. To estimate the unbiased error of the test sample, Random Forests uses the out-of-bag (OOB) procedure. The OOB error estimation is used in [122] to determine the sufficient number of classifiers. As the maximum accuracy no longer increases, the training procedure stops.

#### 2.4.2.2 Selection

There is no value from combining similar classifiers' decisions. The effectiveness of the ensemble is conditioned by the diversity and the correctness of the base classifiers. Ensemble Selection is one of the strategies that can be used to handle this challenge. Ensemble Selection has been known in the literature as ensemble pruning [32, 123, 124], ensemble thinning [125] and ensemble reduction [30, 126]. ES can be considered as an intermediate process between building the ensemble and aggregating the decisions. Specifically, ES is the strategy of optimizing and selecting the number and the type of individual classifiers in-advance. Collecting the decisions from a reduced number of models speeds up the classification systems and relieves memory storage. In the literature, the selection process can be performed offline, static selection [34], or online, dynamic selection [28].

- **Static Ensemble Selection:** Known in the literature review as ensemble pruning. This process is done during the training and before the real test. A subset of classifiers that optimize a predefined function/metric are selected, and the estimation of that metric is done over a pruning set. The most common metrics/selection criteria are; diversity [127, 128], classification accuracy [87, 129], instance margin [24, 130, 131]. However, it is not trivial to find the optimal subset of classifiers from a large ensemble as the complexity increases exponentially with the pool size. Researchers agreed in common that ES is a combinatorial search problem with $2^T - 1$ nonempty subsets to be evaluated from pool size, $T$, to find the best subset [33, 34]. To handle this complexity, several attempts ranging from optimized search [30, 132, 133], clustering techniques [26, 134], and greedy algorithms [23, 31, 37, 135, 136]

have been applied over decades. More details about ordering-based pruning will be discussed in Section 5.2.

- **Dynamic Ensemble Selection:** This process is done during the test time. A single classifier or subset of classifiers are selected based on the unseen sample. In addition, dynamic weights are assigned according to the competition among individual classifiers over the local region of an unknown sample **x**. As a consequence, the selected subset is changeable for each test pattern [137]. This process consists of three steps; (1) Definition of the local region surrounding the query sample **x**, (2) Determination of the selection criteria to estimate the competence level of the classifiers, and (3) Determination of the selection mechanism, whether to select a single classifier or ensemble of classifiers. Without debate, dynamic selection techniques can outperform the static selection methods as experimentally been proved in [28] since the selection is optimized for each test sample independently. The rationale in dynamic selection is that each classifier is an expert in a different local region within the feature space. However, in the dynamic selection, there is a computational overhead for selecting subensemble for each test sample. Besides, those techniques flood the memory space as all individual classifiers have to be retained in memory. Additionally, the dynamic selection is affected by the outlier instances around the query sample in the feature space [138].

Cluster and pick [139] is a poor variant of dynamic selection. Initially, the input space is split into disjoint regions via clustering the training samples. The best classifier is then defined and chosen for each cluster. Cluster and select is in-between static and dynamic strategies; the classifiers are selected dynamically depending on which region the input sample belongs to, but the regions are static, determined in advance during the training [87].

#### 2.4.2.3 **Integration**

Also known as the combiner function. The combiner balances the deviation between the diversity and the bias, also alleviates the errors that certain models have made [140]. This process concerns the methodology by which the outputs of the

base classifiers will be aggregated. A vital phase in building a group of classifiers is the use of a suitable fusion strategy to aggregate response decisions [15, 18]. The responses of individual classifiers restrict the fusion method and enhance or degrade the composite prediction. Those responses may be on the *abstract level* [141], where each classifier specifies a class name as a decision. Additionally, the responses may be *ranked levels* [142], where each classifier outputs a ranked subset of class labels, or even *measurement values* [143, 144], where each classifier specifies a posterior probability.

- **Non-trainable:** The combination function does not require any training from the classifiers' decision space. Many combination rules have been proposed; for measurement values (Sum, Product, Maximum, Minimum, Median, and Decision Templates [145]), for abstract level (Majority voting [146], Behavior Knowledge Space [147], and Naive Bayes [141]), and for ranked levels (Borda count [148]). The majority voting, Equation(2.6), will be effective if the base classifiers are independent. While, the weighted sum rule, Equation(2.7), produces good results if the classifiers perform the same task and have comparable success or when we would like to avoid over-fitting or long training time [98]. The weights, $w_k$, are calculated to be proportional to the individuals' performance over a validation dataset. In addition, weighted voting is preferred for highly imbalanced datasets [143, 149].

$$\hat{\Psi}(\mathbf{x}) = \underset{c \in \mathcal{M}}{\arg\max} \sum_{k=1}^{T} [\Psi_k(\mathbf{x}) = c] \tag{2.6}$$

$$\hat{\Psi}(\mathbf{x}) = \underset{c \in \mathcal{M}}{\arg\max} \sum_{k=1}^{T} [\Psi_k(\mathbf{x}) = c]\, w_k \tag{2.7}$$

- **Trainable:** The combination function is to be configured specifically to the classification task. The aggregation function will be trained over a validation dataset from the domain, *base classifiers outputs*, and co-domain, *real attribute output*. For example; the classifiers' fusion weights can be optimized by evolutionary algorithms (EA). In [149], the authors tuned the weights

for the selection and fusion of multiple cost-sensitive decision trees to handle imbalanced data using the evolutionary genetic algorithm (GA) [150]. Each chromosome from the genetic population simulates a weighted ensemble as $[w_1, w_2, \ldots, w_k, \ldots w_T]^T$, $\forall w_k \in [0, 1]$. The ensemble performance is estimated for each chromosome, and genetic operators evolve the next generations. In addition, GA has been applied in [151] to tune the weights of heterogeneous classifiers in consideration of the above chromosome encoding. While, a bird-flock based optimization algorithm has been applied in [152, 153] to enhance the fusion process.

- **Meta-classifier:** Meta-learner is another important fusion mechanism. It is a trainable method, where the aggregation function is to be learned based on the base classifiers outputs. *Stacking* [154] is the most popular meta-learning method, where the predictions from the base classifiers become meta-features/inputs to a new classifier. The original target attribute from the training set remains as it is. Stacking is generally used to combine heterogeneous models, the term refers to stacking layers of classifiers. In stacking, the correctness of the base classifiers will be learned indirectly through the meat-learner. In *Grading* [155], the predictions of base classifiers are transformed into true or false. After that, one meta-classifier will be trained on the transformed decisions of one base model to learn when it errors. Therefore, grading can be seen as a generalization of cross-validation selection, by using only those classifiers that correctly predict specific instances.

- **Dynamic weighting:** Similar to dynamic ensemble selection. The classifiers' fusion weights are determined dynamically based on the local competence of the classifier in the region where the unknown **x** is located. A higher weight value is assigned to the most competent classifier [28]. For example, dynamic integration of classifiers in [156]. In addition, the dynamic ensemble selection and the dynamic ensemble weighting can be hybrid as in [137].

Finally, Figure 2.6 shows the questions that should be answered during the four levels of constructing MCS [15].

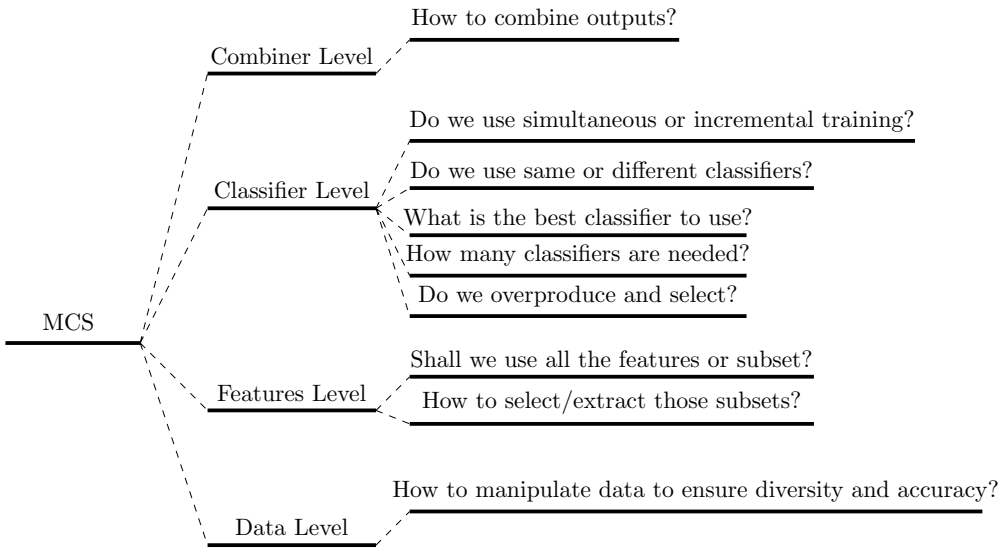


**Figure 2.6:** Four level questions while building MCS.

### 2.4.3 Diversity and Uncorrelated Errors

Indeed the two basic facets of enhancing the efficacy of the committee are often referred to as decision optimization and coverage optimization [157]. *Decision optimization* refers to methods for choosing and optimizing the combination method. *Coverage optimization* refers to the techniques used to construct a diverse classifier set, assuming a fixed combiner. In Section 2.4.2.1, the prospective methods to generate diverse base models were reviewed. If a classifier makes errors on some objects, then it is better to complement it with another classifier that compensates that error. The ensemble performance is restricted by a compromise between an individual's accuracy and group diversity. Confirmed as, neither the individual's accuracy [158] nor the diversity [159] on their own provide reliable ensemble to outperform the best individual classifier. Here, we agree with L. Kuncheva [15], "*good estimate can be obtained from arbitrarily inaccurate estimators as long as their deviations cancel one another*". However, it is so difficult to engineer this ensemble design.

There is no benefit from combining redundant classifiers, and the system will be more complex. A large number of classifiers increase the ambiguity, risk, and add

more complexity to the selection procedure which could lead to weak generalization [87]. While for critical systems the useful evidence is so important and should not be wasted. Regarding that, more attempts have been made to select classifiers based on diversity measures. In [160], the authors used a double fault measure to cluster classifier outputs, then they picked a single classifier from each cluster. While in [161] a similarity measure is used to pick a classifier from a pool of five classifiers. Many diversity measures have been presented [159, 162] with the conclusion that diversity guided search could be invaluable. In addition, the diversity has been intensively used with the combiner performance [87, 128, 163] to improve the efficiency. The system design does not end with the selection process, as the selected classifiers should be next combined. For that, there should be a correlation between diversity measure value and committee performance. Diversity initiatives that represent at least some clear association trends, with generalization, have the potential to become appropriate criteria for selection. Diversity measurement is not a trivial task in ensemble methods.

Matti et.al [164] stated that diversity can be measured from two perspectives, Figure 2.7, based on the population. (a) *The data-based approach:* we have $N$ populations with $T$ objects. The ensemble diversity of the $T$ classifiers is calculated for each sample $\mathbf{x}_i$, then the overall average is considered over $N$ samples. Here the diversity of all member classifiers is evaluated simultaneously (Non-Pairwise measurement). (b) *The classifier-based approach:* in this case we have $T$ populations each contain $N$ objects. The diversity is to be calculated based on the classifiers' outputs for all input data. Each pair of classifiers are considered for measuring the diversity, then the overall diversity is computed by averaging over the number of pairs (Pairwise measurement).

Next, a set of popular measures are presented based on the correct/incorrect outputs. First, the output of $\Psi_k$ can be represented as N-dimensional binary vector $y_k = [y_{1k}, y_{2k}, \ldots, y_{Nk}]^T$, such that $y_{ik}$=1, if $\Psi_k$ recognizes $\mathbf{x}_i$ correctly, and 0 otherwise, $k = 1, 2, \ldots, T$. Moreover, the relationship between a pair of classifiers ( $\Psi_j$, $\Psi_k$ ) can be represented as in Table 2.1.

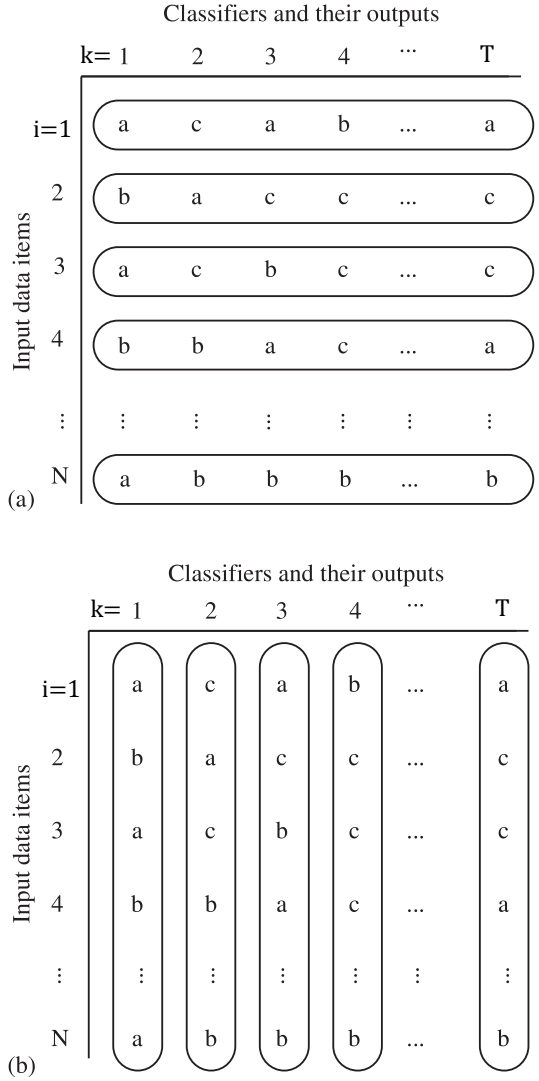


**Figure 2.7:** Diversity measure approaches for MCS: (a) data-based, and (b) classifier-based, taken from [164].

**Table 2.1:** $2 \times 2$ table of the relationship between a pair of classifiers.

| | $\Psi_k$ correct (1) | $\Psi_k$ wrong (0) . |
|---|---|---|
| $\Psi_j$ correct (1) | $N^{11}$ | $N^{10}$ |
| $\Psi_j$ wrong (0) | $N^{01}$ | $N^{00}$ |
| Total, $N = N^{00} + N^{01} + N^{10} + N^{11}$ | | |

#### 2.4.3.1 Pairwise diversity measures

1. *The Q statistics :* Is a various statistic measure to measure the similarity of two classifier outputs. The $Q$ statistics for two classifiers, $\Psi_j$ and $\Psi_k$ is:

$$Q_{j,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} \tag{2.8}$$

Symbols are explained in Table 2.1. $Q$ varies between -1 and 1. Classifiers that agree on the same objects will have positive $Q$ values, and those that make mistakes on different objects will have negative $Q$ values. The averaged statics over all pairs of classifiers is:

$$Q_{avg} = \frac{2}{T(T-1)} \sum_{j=1}^{T-1} \sum_{k=j+1}^{T} Q_{j,k} \tag{2.9}$$

2. *The correlation coefficient $\rho$ :* The correlation between two binary classifier outputs can be calculated as:

$$\rho_{j,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{\sqrt{(N^{11} + N^{10})(N^{01} + N^{00})(N^{11} + N^{01})(N^{10} + N^{00})}} \tag{2.10}$$

For any two classifiers, $Q$ and $\rho$ have the same sign, and it can be proved that $|\rho| \leq |Q|$.

3. *The disagreement measure :* This measure has been used by HO [165] to measure the diversity in decision forests. It is the ratio between the number of samples on which one classifier disagree with another, to the total number of samples.

$$Dis_{j,k} = \frac{N^{01} + N^{10}}{N^{11} + N^{10} + N^{01} + N^{00}} \tag{2.11}$$

4. *The double-fault measure :* It is defined as the proportion of samples on which both classifiers makes error, i.e.,

$$DF_{j,k} = \frac{N^{00}}{N^{11} + N^{10} + N^{01} + N^{00}} \tag{2.12}$$

For all pairwise measures, the averaged values are calculated similarly to Equation (2.9).

2.4.3.2 **Non-Pairwise diversity measures**

1. *The entropy measure E :* The highest diversity among classifiers for a particular $\mathbf{x}_i \in \mathbf{X}$ is proved by $\lfloor T/2 \rfloor$ of the votes for $\mathbf{x}_i$ with the same value (0 or 1) and the other $T - \lfloor T/2 \rfloor$ with the alternative value. If all decisions are 0's or all are 1's, then there is no diverse. Denote by $t(\mathbf{x}_i)$ the number of classifiers that correctly classify $\mathbf{x}_i$, i.e, $t(\mathbf{x}_i)$=$\sum_{k=1}^{T} \Psi_{ik}$. On the basis of this concept, the diversity can be measured as:

$$E = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{(T - \lceil T/2 \rceil)} \min\{t(\mathbf{x}_i), T - t(\mathbf{x}_i)\} \tag{2.13}$$

   $E$ ranges between 0 and 1, where 0 implies no difference, whereas high diversity is measured by 1.

2. *Kohavi-Wolpert variance :* It measures the average variance between the binomial distributions of the outputs of each classifier. This measure can be simply calculated by :

$$KW = \frac{1}{NT^2} \sum_{i=1}^{N} t(\mathbf{x}_i)(T - t(\mathbf{x}_i)) \tag{2.14}$$

3. *Measurement of interrater agreement $\kappa$ :* It is used to measure the level of agreement while correcting the chance. let's denote $\bar{p}$ as the average individual classification accuracy, i.e.,

$$\bar{p} = \frac{1}{NT} \sum_{i=1}^{N} \sum_{k=1}^{T} \Psi_{ik} \tag{2.15}$$

   then, the interrater agreement could be formulated as:

$$\kappa = 1 - \frac{\sum_{i=1}^{N} t(\mathbf{x}_i)(T - t(\mathbf{x}_i))}{NT(T-1)\bar{p}(1-\bar{p})} \tag{2.16}$$

#### 2.4.3.3 Uncorrelated Errors

Standard diversity measures do not take into account that for classification purposes, identical correct decisions are preferred over identical erroneous decisions. It may be useful to analyze, in particular, the errors made by committee members. In this case, we will focus on the error types and whether this error occurs or not by the base models. If two classifiers incorrectly classify a sample into two separate categories, then this case is known as *uncorrelated errors/diversity of errors*, as the predicted class is not the same despite the fact that both make mistake. The most difficult samples are the cases where all classifiers agree to the same incorrect class. Diversity measures that capture the type of error are initiatives and powerful as they can solve the following challenges:

- There are a few correct recognition results in most recognition tasks, while it is hard for the combination function to predict the correct output from this whole set of incorrect predictions. This can be solved by knowing the class of error.
- Classifiers that agree on the correct outcome should be credited rather than eliminated when selecting classifiers, however, this is conflicting with the principle of diversity maximization.

Naturally, It is best if all classifiers make a correct prediction and it is better if we have fewer classifiers make a mistake. Even, for those mistakes, it is exceedingly helpful if the errors are different as often as possible, i.e. maximum diverse errors. Hence, the oracle-level (binary) classifier outputs should be interconnected with the predicted class category to identify the diversity of errors.

As stated in the introduction of this part, the difference in the mistakes made by the member classifiers really affects the performance. Given the following notations; $N^{00}_{same}$ indicates the number of samples when both classifiers are inaccurate and suggest the same decision. $N^{00}_{diff}$ stands for the number of samples when both classifiers are incorrect, but suggest different decisions. Next, let's present metrics [164] that could be used to measure the diversity of errors:

1. *Same-fault measure :* In an extension of the Double-Fault measure, the simultaneous fault could be restricted to measure when both classifiers are

inaccurate and suggest the same output. This can be measured for two classifiers $\Psi_j$ and $\Psi_k$ as:

$$SF_{j,k} = \frac{N^{00}_{same}}{N^{11} + N^{10} + N^{01} + N^{00}} \tag{2.17}$$

Then the averaged pairwise measures could be calculated similarly to Equation (2.9). The optimal classifier set is picked based on the minimum measure.

2. *Weighted count of errors and correct results:* It is designed to consider information about correct prediction. The number of samples that match specific cases can be weighted; based on classifier outputs, "both correct" is favorable and classifiers of this type are highly selected via increasing the weights. While "both incorrect and same" are highly penalized by assigning high negative weight, this can be defined for two classifiers $\Psi_j$ and $\Psi_k$ as:

$$WCEC_{j,k} = N^{11} + \frac{1}{2}(N^{10} + N^{01}) - N^{00}_{diff} - 5N^{00}_{same} \tag{2.18}$$

Moreover, based on the combiner's performance over the training set, the weights above could be optimized. The averaged pairwise measures are calculated similarly to Equation (2.9), and the optimal classifier set is picked based on the maximum measure.

3. *Exponential error count :* As more errors are encountered the classifier capability will be hindered. Here, this concept is emphasized by counting the number of errors and assigning a weight in an exponential fashion. Assume $\Psi^{k\times 0}_{same}$ denote the number of errors made by $k$ classifiers to the same class, then the measure can be defined as:

$$EEC_{(\Psi_1,\ldots,\Psi_T)} = \frac{\sum_{k=1}^{T}(\Psi^{k\times 0}_{same})^k}{N^{T\times 1} + 1} \tag{2.19}$$

This measure considers all classifiers set. In addition, the correct classification is considered by scaling the exponential sum with $N^{T\times 1}$ (the number of samples for which every member classifier was correct). The optimal classifier set is picked based on the minimum measure.

## 2.5 Soft Computing

Soft computing is one of the pioneer computing paradigms which resembles the human mind's remarkable capacity to think, understand, and solve difficult real-life problems [166, 167]. Soft computing exploits the tolerance for imprecision, partial truth, and uncertainty to achieve robustness, tractability, low solution cost, and very high-performance [168]. Modern machinery is more complex to be controlled and stabilized. The reasons for this difficulty is the lack of numerical models that describe exactly how they work, and the existence of many nonlinear and time-variant plants. Soft computing methods support intelligent control, nonlinear programming, optimization, and decision making support. Among those methods; fuzzy logic, genetic algorithms, artificial neural network. Soft computing has become popular with their wide applications to many research fields as speech recognition, communication, pattern recognition, signal processing, automatic control engineering. In their connectivity with the area of MCS, we present several soft computing methods that have been used to improve efficiency:

**Fuzzy Logic:** In domains and environments that are realistic, incomplete evidence inevitably emerges. During experiments, noise corruption or instrument errors may give rise to data when a certain value is measured, leading to incomplete data. In other scenarios, collecting the correct information can be excessively costly or unviable. In addition, using extra information from an expert, which is usually given by fuzzy logic and fuzzy sets, can be useful. Typically, data has a certain degree of vagueness. If the imprecision is significant, then the imprecise values must be handled in all the phases of learning and classification. To address the uncertainty of decision trees with minor disruptions, fuzzy sets and their underlying approximate reasoning capacities have been paired with decision trees in [169, 170]. The resulting trees exhibit increased tolerance to noise, and extend applicability to ambiguous or unclear circumstances. In [171], a fuzzy random forest was suggested to increase the diversity of the trees through randomness, with the versatility of handling incomplete data. The numeric attributes are discretized through a fuzzy partition, so each internal node in the fuzzy decision tree constructs a child node for each fuzzy set of the partition. Then the membership degree of the examples to different fuzzy sets is determined to optimize the node-split attribute.

In addition, a fuzzy combination method for one-class classifiers has been proposed in [172].

**Evolutionary Algorithms:** Evolutionary algorithms uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection to solve complex problems. For example, genetic algorithm (GA) [173] emulates the natural and human evolution, where common genes from parents can be transferred to their children. GA evolves an initial population of chromosomes; each chromosome represents one solution in the search space. The algorithm depends on applying crossover and mutation operators to explore and exploit different regions from the search space, which probably contains promising solutions. The process continues for a specific number of generations or till reaching threshold error or if the algorithm is trapped in a local minimum without the ability to find more accurate solutions. Ensemble systems are very complex architectures that need to be optimized. GA has been applied in [87] to control the ensemble size (classifier subset selection) via optimizing several criterias. While in [151] GA has been considered as a trainable combiner to tune the classifiers' weights for the fusion process. Furthermore, Kuncheva et al. [174] applied two versions of GA for selecting feature subsets to be used by base models. While tsymbal et al. [175] applied GA to optimize the diversity of the best-collected feature subsets.

**Artificial Neural Networks:** Neural networks were proved to be universal approximators with unlimited flexibility. In any number of dimensions, they could approximate any classification boundary. However, this capability comes at a price. There is a need to train large systems with a huge number of parameters. Then, an acceptable architecture with the tuned parameters will be trusted for all future classifications. In the face of several local minima, all global optimization methods yield "optimal" parameters ($w$) that differ significantly from one run of the algorithm to the next. This reveals a great deal of randomness arising from various initial weights ($w^0$) and various sequencing of training examples. This randomness appears to distinguish between network errors. The final weights correspond to various ways of identifying the training pattern. Hence, the collective decision created by several ANNs may, therefore, be much less fallible than any network. In [176], the ANNs in the form of base classifiers were generated to form bagging and boosting ensembles to predict the bankruptcy. While in [177] a convolutional

neural network is linked to a shallow neural network to classify arrhythmia in a Holter ECG signal.

**Swarm Intelligence:** SI algorithms are defined as *"nature-inspired algorithms that concern the collective, emerging behavior of multiple, interacting agents that follow some simple rules"* [178]. These algorithms mimic the social behavior of swarms/groups of creatures in nature. In [179], the authors discussed the benefits of SI over evolutionary algorithms. Most SI methods consider exploration and exploitation in their working mechanisms [180]. The popularity of those algorithms returns to; the simplicity of inspiration, flexibility, derivative-free mechanism, and local optimum avoidance [179]. Firefly algorithm has been applied in [181] to combine the ensemble pruning with a weighted classifier fusion module. In addition, the Ant colony algorithm has been incorporated to optimize the decision forest [182] to provide self-adaptability with the classification task. Recent SI algorithms will be discussed in Sections 4.3.5, 4.3.6, and 4.3.7.

*Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less.*

Marie Curie

CHAPTER

# 3

# State-of-the-art

The number of ensemble-based proposals has risen gradually. However, there is no single ensemble technique that could solve all problems, as each problem has its own characteristics [94]. Ensemble-based proposals are among the state-of-art techniques for the plurality of supervised learning tasks [18, 91]. As it was stated in Section 2.4.2, most ensemble algorithms differ in terms of combination function, kind of individual classifier, learning schema, and diversity promotion methodology. However, bagging [3] and boosting [110] ensembles are the most popular techniques with great attention from the machine learning researchers. The concept of ensemble-based learning has been published in many reviews and studies [14, 94, 183, 184]. In this chapter, popular, recent, effective ensemble-based classifier learning algorithms are reviewed. In Section 3.1, renowned bagging ensemble methods are revised. Next, a review of boosting ensembles is to be presented in Section 3.2. While the gradient boosting ensembles with their characteristics are presented in Section 3.3. The revision of those methods includes innovation, detailed description, and diversity promotion methodology. Following that, a summary of those algorithms in the form of a comparison table, and the available ensemble tools and software packages are shown in Section 3.4. Furthermore, the importance of metaheuristics for improving the efficiency and efficacy of MCS will be discussed in Section 3.5. Finally, Section 3.6 closes this chapter with the

identified remarks and gaps, upon which the research path of the next chapters was followed.

let's assume $T$ to represent the ensemble size, $k$ denote the current iteration. Therefore, $\Psi_k$ means the estimator/classifier trained in iteration $k$, $\epsilon_k$ denotes the error made by the estimator during the iteration $k$. While $w_i$ stands for the weight assigned to the training sample $\mathbf{x}_i$ by boosting methods.

## 3.1 Bagging-Like Ensembles

Firstly, let's focus on the original bagging algorithm and the bagging-like techniques, which train their base learners independently. Ensembles in this category use data transformations to promote diversity. The main purpose is to explain the algorithmic details, limitations, and improvements.

### 3.1.1 Bagging

Bagging [3] is one of the earliest techniques for creating an ensemble of classifiers. The pseudocode of bagging is shown in Algorithm 1. In bagging each classifier $\Psi_k$ is to be trained independently. A bootstrap dataset $D_k$ is derived from the initial training data via resampling with replacement. Approximately 1/3 of the samples are excluded from the training of each classifier. These samples are called ¨out-of-bag¨ (OOB); each classifier has a different set of OOB. The OOB samples comprise the independent test set for evaluating each $\Psi_k$. The final decision is made according to the majority voting if the outputs are labels, or by averaging if the outputs are measurement values. Bagging is particularly appealing when we have limited-size training data. Unstable base learners are preferred with bagging, as they are sensitive to diversity from the small perturbations in different training sets.

### 3.1.2 Forest-Like Ensembles

This section makes a review of the most used ensembles based on trees. Decision forest [183] seeks to enhance the predictivity of a single decision tree via training and integrating multiple trees. This part discusses how the decision forest can be

**Algorithm 1:** The pseudocode of Bagging.

**Input :** $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ - training set, $T$ - pool size, $B$ - percent of bootstrap, Train_Classifier() - method of classifier training.

**Output:** $(\Psi_1, \ldots, \Psi_T)$ `/* Ensemble of classifiers */`

**begin**

  **Train: for** $k \leftarrow 1$ **to** $T$ **do**

    $D_k$ =Perform bootstrap replica by drawing $B$ percent of $D$;

    $\Psi_k$ = *Train_Classifier* $(D_k)$;

    Add the classifier $\Psi_k$ to the ensemble pool;

**Test:** Evaluate all the classifiers $\{\Psi_1, \ldots, \Psi_T\}$ for the unknown sample **x**; Select the final class via the majority vote rule as:

$$\hat{\Psi}(\mathbf{x}) = \arg\max_{c \in \mathcal{M}} \sum_{k=1}^{T} [\Psi_k(\mathbf{x}) = c] \tag{3.1}$$

created. An analysis of 121 datasets, 179 classification algorithms originating from 17 learning families, concluded that decision forests appear to outperform other learning approaches [91].

#### 3.1.2.1 **Random Forest**

Random Forest (RF) [4], due to its simplicity and performance, is one of the most common ensembles. It is a variant of the bagging algorithm, with similar base classifiers of decision tree type. Each tree has certain training parameters that vary randomly. Such parameters can be the bootstrap replicas, as in bagging. In addition, those parameters could contain the maximum number of features $f$ to be considered by each tree node for splitting. The size of $f$ is a significant design parameter, it is recommended to be the square root of the number of attributes. As a result, certain attributes (including the best) might not be used for each split, but an attribute omitted from one split may be used for other splits in the same tree. The trees are grown without pruning to the maximum depth. The pseudocode of RF is shown in Algorithm 2.

**Algorithm 2:** The pseudocode of Random Forest.

**Input :** $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ - training set, $T$ - pool size, $B$ - percent of bootstrap, $f$ - maximum sub-features for splitting, *Fit_tree*() - method of classifier training.

**Output:** $(\Psi_1, \ldots, \Psi_T)$ `/* Ensemble of classifiers */`

**begin**

  **Train: for** $k \leftarrow 1$ **to** $T$ **do**

    $D_k$ =Perform bootstrap replica by drawing $B$ percent of $D$;

    $\Psi_k = \textit{Fit\_tree}(D_k, f)$;

    **function** *Fit_tree*$(D_k, f)\{$

      **for** !*Stop_Split()* **do**

        $a_1, \ldots, a_m = \textit{Pick\_Random\_Attributes}(f)$;

        **for** $a_j$ *in* $a_1, \ldots, a_m$ **do**

          $s_j = \textit{Pick\_Best\_Split}(D_k, a_j)$;

        $a, s = \textit{arg max}_j\ Score(D_k, a_j, s_j)$;

        *Split_Tree* $(D_k, a, s)$;

    Add the classifier $\Psi_k$ to the ensemble pool;

**Test:** Evaluate all the classifiers $\{\Psi_1, \ldots, \Psi_T\}$ for the unknown sample $\mathbf{x}$; Choose the class via the majority vote rule, as in Eq. (3.1);

#### 3.1.2.2 Extra-Trees

Extra-Trees (ET) [185] is a tree-based randomized ensemble. The splitting for each tree is randomly selected to promote the diversity. Similar to RF, at each node only a random subset of the features $f$ is considered for splitting. However, in contrast to RF, random thresholds for each feature are used rather than searching for the best possible thresholds. Then in terms of impurity calculation, the best cut is chosen from those randomly selected [94]. This random splitting is the only cause of diversity promotion, but it is adequate to reduce the model's variance. The second difference from RF, is that ET uses the original and the complete dataset to train each individual decision tree. Trees are extended to the fullest degree possible, and the final decision is given by the majority voting of base individuals. Extra-trees

are much faster to train than regular Random Forest. In addition, ET could employ bootstrapping for training. The pseudocode of ET is shown in Algorithm 3.

**Algorithm 3:** The pseudocode of Extra-Trees.

**Input :** $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ - training set, $T$ - pool size, $f$ - maximum sub-features for splitting, *Fit_Extra_tree*() - method of classifier training.

**Output:** $(\Psi_1, \ldots, \Psi_T)$ `/* Ensemble of classifiers */`

**begin**

**Train: for** $k \leftarrow 1$ **to** $T$ **do**

$\Psi_k =$ *Fit_Extra_tree* $(\hat{D}, f, pruning = False)$;

**function** *Fit_Extra_tree*$(\hat{D}, f)\{$

**for** !*Stop_Split()* **do**

$a_1, \ldots, a_m =$ *Pick_Random_Attributes*$(f)$;

**for** $a_j$ *in* $a_1, \ldots, a_m$ **do**

$s_j =$ *Pick_Random_Split*$(\hat{D}, a_j)$;

$a, s =$ *arg max*$_j$ $Score(\hat{D}, a_j, s_j)$;

*Split_Tree* $(\hat{D}, a, s)$;

Add the classifier $\Psi_k$ to the ensemble pool;

**Test:** Evaluate all the classifiers $\{\Psi_1, \ldots, \Psi_T\}$ for the unknown sample **x**; Choose the class via the majority vote rule, as in Eq. (3.1);

#### 3.1.2.3 Random Patches

Random Patches (RP) [186] are known as Random Subspace generalization. By training the base models with different data patches, diversity can be promoted. A random patch is captured from the training data in terms of samples ($p_s * N$) and features ($p_f * d$) to train each base model independently. Where $p_s$ and $p_f$ are the percentages of samples and features, respectively. The pseudocode of RP is shown in Algorithm 4. For the base learner, RP is not inherently limited to the decision tree, even though decision trees or extra-trees are preferred. Without replacement,

the samples of each training subset are selected. Finally, in contrast to RF, the subset of features is chosen globally in advance before the training of each model.

**Algorithm 4:** The pseudocode of Random Patches.

**Input :** $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ - training set, $T$ - pool size, $p_s$ - percentage of considered samples, $p_f$ - percentage of considered features, *Fit_Estimator*() - method of classifier training.

**Output:** $(\Psi_1, \ldots, \Psi_T)$ `/* Ensemble of classifiers */`

**begin**

  **Train: for** $k \leftarrow 1$ **to** $T$ **do**

    $D_k = \textit{Sampler}\ (D, p_s * N, p_f * d)$;

    $\Psi_k = \textit{Fit\_Estimator}\ (D_k)$;

    Add the classifier $\Psi_k$ to the ensemble pool;

**Test:** Evaluate all the classifiers $\{\Psi_1, \ldots, \Psi_T\}$ for the unknown sample $\mathbf{x}$; Choose the class via the majority vote rule, as in Eq. (3.1);

### 3.1.3 Rotation Feature Space Ensembles

This section reviews the ensembles based on feature space rotations. The concept of the rotation strategy is to promote individual precision and diversity within the ensemble simultaneously. A new extracted feature set is generated to train the base models. The decision tree is generally used as a base classifier because it is sensitive to the features that have been rotated.

#### 3.1.3.1 Rotation Forest

Rotation Forest (RotF) [86] is a bagging-like ensemble that trains independent base classifiers. The variability of the ensemble comes from; data sampling, and feature extraction techniques. The pseudocode of RotF is shown in Algorithm 5. To force more variance, the feature space $\mathcal{X}$ is split into $F_{max}$ disjoint parts. For each feature subset $F_j$, select a nonempty subset of classes randomly and then draw bootstrap samples with $75\%$ of the data count, to form $D_{k,j}$ dataset as summarized in Algorithm 5. Then, the feature extraction is applied to each data part, $D_{k,j}$, individually

via principal component analysis (PCA) [187]. When all the coefficients of the PCA are returned, they are rearranged in a rotation matrix $R_k^a$, to be compatible with the original features [86]. Due to its sensitivity to feature rotation, a decision tree is used as a base classifier. Finally, the class with the highest probability will be the final ensemble decision.

**Algorithm 5:** The pseudocode of Rotation Forest.

**Input :** $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$ - training set, $T$ - pool size, $B$ - percent of bootstrap, $F_{max}$ - number of disjoint feature subsets, *Fit_Tree*() - method of classifier training.

**Output:** $(\Psi_1, \ldots, \Psi_T)$ `/* Ensemble of classifiers */`

**begin**

  **Train: for** $k \leftarrow 1$ **to** $T$ **do**

    $F_1, \ldots, F_{max} =$ *Split_Features* $(\mathcal{X}, F_{max})$;

    **for** $j$ *in [1,max]* **do**

      $D_{k,j} =$ *Filter_Features*$(D, F_j)$;

      $D_{k,j} =$ *Randomly_Select_Classes*$(D_{k,j})$;

      $D_{k,j} =$ *Bootstrap_Sampler*$(D_{k,j}, B = 75\%)$;

      $a_{k,j}^{(1)}, a_{k,j}^{(2)}, \ldots, a_{k,j}^{(\mathcal{X}^{(d)})} =$ *PCA*$(D_{k,j})$;

    $R_k^a =$ *Construct_Rotation_Matrix*();

    $\Psi_k =$ *Fit_Tree* $(DR_k^a)$;

    Add the classifier $\Psi_k$ to the ensemble pool;

**Test:** Evaluate all the classifiers $\{\Psi_1, \ldots, \Psi_T\}$ for the unknown sample $\mathbf{x}$; Choose the class that receives the highest probability via Eq. (3.2);

$$\hat{\Psi}(\mathbf{x}) = \arg\max_{c \in \mathcal{M}} \sum_{k=1}^{T} \left[ \frac{\Psi_{k,c}(\mathbf{x} R_k^a)}{T} \right] \tag{3.2}$$

#### 3.1.3.2 Random Rotation Ensembles

Random Rotation (RR) [188] ensembles are designed to provide well-known ensembles with more diversity without limiting the precision of the individual classifier. Usually, a decision tree is used as a base classifier. The pseudocode of RR is

shown in Algorithm 6. The methodology depends on constructing rotation matrices $R$ that should be an orthogonal square matrix with $d \times d$ real values, given $d$ to be the original number of features we have. Matrix $A$ is to be drawn randomly to contain $d^2$ independent values. Then, the matrix $A$ is factorized by Householder QR decomposition, with the restriction to make the unit determinant ($|R| = 1$). Once we obtain $R$, it is multiplied by the original dataset to get a rotated dataset where the decision tree could be trained. The diversity is promoted by the random rotations of the features with their implication to improve the individual's accuracy.

**Algorithm 6:** The pseudocode of Random Rotation Ensembles.

**Input :** $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ - training set, $T$ - pool size, *Fit_Tree*() - method of classifier training.

**Output:** $(\Psi_1, \ldots, \Psi_T)$ `/* Ensemble of classifiers */`

**begin**

  **Train: for** $k \leftarrow 1$ **to** $T$ **do**

    $D =$ *Variable_Scaling* $(D)$;

    $A =$ *Random_Matrix*$(d \times d)$;

    $Q, R =$ *Householder_Decomposition*$(A)$;

    $R_k =$ Q * *Diag*$(Sign(Diag(R)))$;

    **if** $|R_k| < 0$ **then**

      $R_k[, 0] = -R_k[, 0]$;

    $\Psi_k =$ *Fit_Tree* $(DR_k)$;

    Add the classifier $\Psi_k$ to the ensemble pool;

**Test:** Evaluate all the classifiers $\{\Psi_1, \ldots, \Psi_T\}$ for the unknown sample $\mathbf{x}$; Choose the class via the majority vote rule as in Eq. (3.1);

## 3.2 Boosting Ensembles

In this part, we describe ensemble techniques that generate a dependent/iterative pool of classifiers. The training of each classifier in the chain is controlled by the ability of the previous item(s) to classify samples. They are called "boosting" because Schapire [189] proved that the performance of a weak learner can be

boosted to be a strong one. Those techniques received more development, with a widespread in the area of machine learning.

### 3.2.1 AdaBoost

AdaBoost [5] is an algorithm that is rarely matched in computational intelligence. The ensemble is formed by adding one classifier at a time. The classifier that joins the ensemble at step $k$ is trained on a dataset selectively sampled from the original dataset. In the beginning, all the samples are equally weighted meaning that all of them are important. By increasing the value of $k$, the likelihood of misclassified samples is increased, so that they can be correctly classified in the next iteration(s). Hence, the most informative training data is provided for each consecutive learning model. The AdaBoost variants are determined based on how the weights $w_i$ are calculated for each $\mathbf{x}_i$. The final decision for unknown sample $\mathbf{x}$ is determined by a weighted majority voting. In Algorithm 7, a multi-class variant called AdaBoost.M1 is introduced. From the algorithm, the classifier's prediction is weighted by $\alpha_k$ determined from its precision. In addition, the weights of the correctly classified samples are lowered and this reduction is always measured in reference to the classifier's error. The algorithm assumes the error of each classifier to be less than 0.5 ($\epsilon_k < 0.5$ ), otherwise the correctly classified samples will receive a higher weight than the misclassified ones. The upper limitation of $\epsilon_k$ to $0.5$ is possible, however in this case the instance weight keeps unchanged. AdaBoost.M2 [5] is an extension to adjust the weights by considering the probabilities provided by the base classifiers for all classes. Stagewise Additive Modeling (SAMME) [190] is another AdaBoost extension, with the aim to elevate the restriction $\epsilon_k < 0.5$ by summing the term $\log(M-1)$ to the calculation of $\alpha_k$, where $M$ is the number of classes. In this case, the performance of each classifier needs to be $> 1/M$ instead of $> 1/2$.

### 3.2.2 LogitBoost

LogitBoost (LB) [191] uses boosting schema to fit an additive logistic regression. This methodology can be seen as a greedy tool for improving the generalization of

**Algorithm 7:** The pseudocode of AdaBoost.M1.

**Input :** $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ - training set, $T$ - pool size, *Fit_Estimator*() - method of classifier training.

**Output:** $(\Psi_1, \ldots, \Psi_T)$ `/* Ensemble of classifiers */`

**begin**

Initialize instance weight ($w_i = 1/N$);

**Train: for** $k \leftarrow 1$ **to** $T$ **do**

$\Psi_k =$ *Fit_Estimator* $(D, w)$;

$\epsilon_k = \sum_{\mathbf{x}_i \in D | \Psi_k(\mathbf{x}_i) \neq y_i} w_i / \sum_{\mathbf{x}_i} w_i$;

$\alpha_k = \log\left(\frac{1-\epsilon_k}{\epsilon_k}\right)$;

**for** $\boldsymbol{x}_i$ *in* $D$ **do**

**if** $\Psi_k(\boldsymbol{x}_i) = y_i$ **then**

$w_i = w_i . \frac{\epsilon_k}{1-\epsilon_k}$;

Normalize $(w)$;

Add the classifier $\Psi_k$ to the ensemble pool;

**Test:** Evaluate all the classifiers $\{\Psi_1, \ldots, \Psi_T\}$ for the unknown sample $\mathbf{x}$; Choose the class via the weighted majority vote rule, as in Eq. (3.3);

$$\hat{\Psi}(\mathbf{x}) = \arg\max_{c \in \mathcal{M}} \sum_{k=1}^{T} \left[\Psi_k(\mathbf{x}) = c\right] \alpha_k \tag{3.3}$$

logistic regression. The class probability $P_c(\mathbf{x}_i)$ can be calculated as in Equation (3.4), where $\hat{\Psi}_c(\mathbf{x}_i)$ is the aggregated regression function of class $c$.

$$P_c(\mathbf{x}_i) = \frac{e^{\hat{\Psi}_c(\mathbf{x}_i)}}{\sum_{\mathbf{c=1}}^{M} e^{\hat{\Psi}_c(\mathbf{x}_i)}} \tag{3.4}$$

The pseudocode of LB is shown in Algorithm 8. Initially, all the class probabilities of the training samples are equal to $(1/M)$. Where $M$ is the number of classes. For each iteration $k$, and for each class, a regression model $\Psi_{k,c}$ is independently trained. The weight $w_{i,c}$ and the response $z_{i,c}$ of each sample $\mathbf{x}_i$ are updated in relation to the probabilities $P_c(\mathbf{x}_i)$ for each class $c$. Only the values $z_{i,c}$ are computed based on both: the $y_i$ and the probability of the class $P_c$. Then, a weighted

**Algorithm 8:** The pseudocode of LogitBoost.

**Input :** $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ - training set, *Fit_Weighted_Regression*() - method of classifier training, $T$ - pool size.

**Output:** $\{\hat{\Psi}_1(\mathbf{x}), \hat{\Psi}_c(\mathbf{x}) \ldots, \hat{\Psi}_M(\mathbf{x})\}$ `/* Aggregated regression functions of classes */`

**begin**

Initialize instance weight ($w_i = 1/N$);

Initialize class probabilities $P_c(\mathbf{x}_i) = 1/M$;

**Train: for** $k \leftarrow 1$ **to** $T$ **do**

**for** $c \leftarrow 1$ **to** $M$ **do**

**for** $\boldsymbol{x}_i$ *in* $D$ **do**

$w_{i,c} = P_c(\mathbf{x}_i)(1 - P_c(\mathbf{x}_i))$;

**if** $y_i = c$ **then**

$z_{i,c} = 1/P_c(\mathbf{x}_i)$;

**else**

$z_{i,c} = -1/(1 - P_c(\mathbf{x}_i))$

$\Psi_{k,c}(\mathbf{x}) =$ *Fit_Weighted_Regression* $(D, z_c, w_c)$;

$\Psi_{k,c}(\mathbf{x}) = \frac{M-1}{M}(\Psi_{k,c}(\mathbf{x}) - \frac{1}{M}\sum_{\mathbf{j=1}}^{M} \Psi_{j,c}(\mathbf{x}))$;

$\hat{\Psi}_c(\mathbf{x}) = \hat{\Psi}_{k-1,c}(\mathbf{x}) + \Psi_{k,c}(\mathbf{x})$;

Update probability $P_c(\mathbf{x}_i)$ via Eq. (3.4);

**Test:**

For the unknown **x**, get all class probabilities $\{\hat{\Psi}_1(\mathbf{x}), \hat{\Psi}_c(\mathbf{x}) \ldots, \hat{\Psi}_M(\mathbf{x})\}$; Return the class with the highest score as in Eq. (3.5);

$$\hat{\Psi}(\mathbf{x}) = \arg\max_{c \in \mathcal{M}} \hat{\Psi}_c(\mathbf{x}) \tag{3.5}$$

regression tree $\Psi_{k,c}$ is fitted with the training samples $(\mathbf{x}_i, z_{i,c})$ and their weights $w_{i,c}$. Once all the models $\Psi_{k,c}$ have been trained, they are included in the additive model $\hat{\Psi}_c$ of their corresponding class $c$. Unlike AdaBoost, for the full ensemble assembled so far and not only with the last trained model, the class probabilities are estimated to update the weights.

## 3.3 Gradient Boosting Ensembles

This part discusses the powerful machine learning techniques that have received great interest from researchers recently. Those techniques proved their efficiency with complex datasets. Their power comes from the iterative generation and the combination of base models via a greedy procedure. The gradient descent in the function space optimizes the appended classifier with the aim to minimize the given loss function.

### 3.3.1 Gradient Boosting Machine

Gradient Boosting Machine (GBM) [192] is one of the most renowned algorithms that has inspired the design of several gradient boosting-based ensembles. GBM is a stagewise strategy by which the newly added model is optimized to minimize the given loss function. Via the gradient descent minimization, the parameters $\mathbf{a}_k$ of a regression tree are tuned; splitting variables, split locations, and the terminal node. Then, the resultant decision tree is added to the pool with an optimized weight of $\alpha_k$. This weight measures the contribution of the new tree, allowing the algorithm to increase its precision steadily and gradually. The general additive model could be viewed as in Equation (3.6). Where $\Psi(\mathbf{x}; \mathbf{a}_k)$ is the weak learner that gives the best steepest-descent step direction, $-g_k = \{-g_k(\mathbf{x}_i)\}_1^N$ in the N-dimensional data space at $\hat{\Psi}_{k-1}(\mathbf{x})$. From Equation (3.7) this is calculated to minimize some specified loss function $\mathcal{L}(\mathcal{Y}_i, F(\mathbf{x}_i))$. Then the best parameters of a regression tree $\mathbf{a}_k$ can be estimated to be most highly correlated with $-g_k$ over the data distribution, as in Equation (3.8). After tuning the parameters of the tree, a line search is performed via Equation (3.9) to seek the best weight for the added model. The pseudocode of GBM is shown in Algorithm 9. GBM is adaptable to any regression or classification problem described with a derivable loss function. GBM estimates negative binomial log-likelihood loss criteria for binary tasks and multinomial loss criteria for multiclass tasks.

$$\hat{\Psi}_k(\mathbf{x}) = \hat{\Psi}_{k-1}(\mathbf{x}) + \alpha_k \Psi(\mathbf{x}; \mathbf{a}_k) \tag{3.6}$$

$$- g_k(\mathbf{x}_i) = - \left[ \frac{\partial \mathcal{L}(\mathcal{Y}_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)} \right]_{F(\mathbf{x})=\hat{\Psi}_{k-1}(\mathbf{x})} \tag{3.7}$$

$$\mathbf{a}_k = \arg\min_{\mathbf{a},\alpha} \sum_{i=1}^{N} \Big[ - g_k(\mathbf{x}_i) - \alpha \Psi(\mathbf{x}_i; a) \Big]^2 \tag{3.8}$$

$$\boldsymbol{\alpha}_k = \arg\min_{\alpha} \sum_{i=1}^{N} \mathcal{L}(\mathcal{Y}_i, \hat{\Psi}_{k-1}(\mathbf{x}_i) + \alpha \Psi(\mathbf{x}_i; \mathbf{a}_k)) \tag{3.9}$$

**Algorithm 9:** The pseudocode of Gradient Boosting Machine.

**Input :** $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ - training set, $T$ - pool size, $\mathcal{L}(\mathcal{Y}, F(\mathbf{x}))$ - loss function, $\alpha$ - weight/learning rate.

**Output:** $\{\hat{\Psi}_T\}$ `/* Additive ensemble */`

**begin**

  **Initialize:** $F_0(\mathbf{x}) = \arg\min_{\rho} \sum_{i=1}^{N} \mathcal{L}(\mathcal{Y}_i, \rho)$;

  **for** $k \leftarrow 1$ **to** $T$ **do**

    Calculate the steepest-descent $g_k$ for the past $\hat{\Psi}_{k-1}$ via Eq. (3.7);

    Tune parameters $\mathbf{a}_k$ by least squares minimization via Eq. (3.8);

    Perform line search for $\boldsymbol{\alpha}_k$ via Eq. (3.9);

    Add the classifier $\Psi_k$ to the ensemble pool;

**Test:** Evaluate all the classifiers $\{\Psi_1, \ldots, \Psi_T\}$ for the unknown sample $\mathbf{x}$;

The final decision is the aggregation via Eq. (3.6);

### 3.3.2 XGBoost: eXtreme Gradient Boosting

XGBoost [193] is one of the most popular gradient boosting-based ensembles. In a variety of machine learning competitions, it has been commonly accepted with a scoring mark at the top. The algorithmic design considers the computing and memory capacities; to obtain a scalable, high speed, and accurate tree boosting system. XGBoost extends its use to all real-world problems due to its capability to prevent over-fitting. To handle over-fitting, two additional techniques are used; the first technique is shrinkage [194], to reduce the effect of each tree and to leave

space for future trees to improve the model. The second technique is to use the traditional row and/or column sub-sampling(s). The following features have been incorporated to improve the traditional GBM [192]:

- Regularized Learning Objective: The difference between the estimation $\hat{\mathcal{Y}}_i$ and the target $\mathcal{Y}_i$ is determined by a differentiable convex loss function. In addition, a regularized term $\Omega$ is added to penalize the complexity of the model, and to smooth the final learned weights, Equation (3.10).

$$\mathcal{L} = \sum_{i=1} \mathcal{L}(\mathcal{Y}_i, \hat{\mathcal{Y}}_i) + \sum_{k} \Omega(\Psi_k), \text{where} \quad \Omega(\Psi_k) = \gamma L + \frac{1}{2}\lambda \sum_{j}^{L} w_j^2 \quad (3.10)$$

  Where $\Psi_k$ is the additive regression tree to be learned with number of leaves $L$, regularization parameter $\lambda$ to reduce the sensitivity to training data, and the parameter $\gamma$ which works as a threshold of the score function improvement. Each tree has a different structure that maps an instance to the corresponding leaf index $j$. With the inclusion of a new tree, XGBoost is greedily trained via minimizing the objective function, Equation (3.11).

$$\mathit{Obj}_k = \sum_{i=1} \mathcal{L}\Big(\mathcal{Y}_i, \hat{\Psi}_{k-1}(\mathbf{x}_i) + \Psi_k(\mathbf{x}_i)\Big) + \Omega(\Psi_k) \quad (3.11)$$

  The authors derived the formulation of the optimal score weight $w_j^*$ of the leaf $j$ for specific tree structure as in Equation (3.12).

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \quad (3.12)$$

  Where $g_i = \partial_{\hat{\Psi}_{k-1}(\mathbf{x}_i)} \mathcal{L}(\mathcal{Y}, \hat{\Psi}_{k-1}(\mathbf{x}_i))$ and $h_i = \partial^2_{\hat{\Psi}_{k-1}(\mathbf{x}_i)} \mathcal{L}(\mathcal{Y}, \hat{\Psi}_{k-1}(\mathbf{x}_i))$ are the first and the second order gradient values on the loss function. Furthermore, $I_j$ reflects the samples allocated to the $j$ leaf.

- Approximate split finding, parallel tree learning, and out-of-core computation are among other features that enhance the efficiency of XGBoost [193].

### 3.3.3 LightGBM: Light Gradient Boosting Machine

LightGBM [195] is a gradient boosting decision tree ensemble, aimed at speeding up the learning process via adapting mechanisms for data reduction. The classical GBM becomes inefficient when the feature dimension is high and the data size is big. A major reason is that for each feature and all possible splitting points, the information gain is to be estimated for all the data instances. Therefore, the computational complexity will be proportional to both; the number of features and the number of instances. Two new techniques were proposed to resolve this issue: *Gradient-based One-Side Sampling* (GOSS) and *Exclusive Feature Bundling* (EFB).

GOSS is a method of under-sampling driven by the training set's gradients. There is a minimum training error in the samples correlated with small gradients, and they are already well-trained. Therefore, those samples with small gradients can be discarded without affecting the accuracy. GOSS preserves $a\%$ of samples with the largest gradients, then samples $b\%$ from the remainder of the results randomly. After that, GOSS amplifies the sampled data with a small gradient by a constant $\frac{1-a}{b}$ when calculating the information gain. Doing that, the algorithm focuses more to train large gradient-samples while compensating for the influence of data distribution. In addition, the sampling strategy could improve the diversity of the decision tree, which could boost overall precision.

EFB is a strategy for handling high-dimensional feature spaces. The features are quite sparse, many features are exclusive, i.e., never take nonzero values simultaneously. Therefore, the most exclusive features are bundled while reducing the inside bundle conflict. Regarding that, the unnecessary computation for zero feature value can be avoided to improve efficiency. However, the problem of grouping features into the smallest number of exclusive bundles is NP-hard. The authors accommodated the graph coloring problem to match that challenge.

### 3.3.4 CatBoost: Gradient Boosting with Categorical features

CatBoost [196, 197] is the most recent GBM-based ensemble. CatBoost has the capability to support categorical features and to avoid prediction shift. Instead of handling the categorical features in a preprocessing stage, they are manipulated

during the tree splitting. For low-cardinality categorical features, CatBoost applies one-hot encoding. Categorical features are transformed into numerical in relation to the number of appearances of the categories.

CatBoost implements an effective approach that eliminates overfitting and encourages the whole dataset to be used. For that, a random permutation is performed for the dataset, then the average label value for each sample is computed based on the sample with the same category value placed before the given one. The transformation of features may contribute to the loss of knowledge in the interactions between categorical features. Therefore, CatBoost considers greedy combinations of features in the current state of the tree with the rest of the categorical features.

Classical boosting algorithms suffer from prediction shift and overfitting since gradients are estimated over the same samples used to build the model. CatBoost considers the unbiased estimation of the gradient to alleviate that challenge. The trick is to train a number of models equivalent to the number of samples. For each sample $\mathbf{x}_i$, the model $\Psi_i$ is to be trained without considering the gradient estimate for this instance. Therefore, the residual of the $i$th sample is measured using $\Psi_{i-i}$, which does not have that sample in its training. Moreover, the missing values are managed by CatBoost, they are treated as an individual category.

## 3.4 Comparison of Different Ensembles

This part is inspired by the great work that introduced in [94], for presenting the available ensemble packages. However, we will focus only on the R language and its support for the aforementioned ensemble algorithms. The reason for that, all the experiments and the results in this thesis are tested under the framework of R, the reader can refer to [94] for more details about other programming languages. Table 3.1 presents this revision for the bagging-like, boosting, and gradient boosting-based ensembles in R language.

The second part summarizes the main features of the aforementioned ensembles. The comparison between the 12 ensembles will be based on diversity promotion methodology in each algorithm, and the used base model type. The main properties of each algorithm are presented in Table 3.2.

**Table 3.1:** Ensemble software packages for Bagging-like, Boosting, and Gradient Boosting ensembles (in R language); SC: Sequential CPU computing, PCC: Parallel CPU computing, GP: GPU computing, and DC: Distributed computing.

| Ensemble Method | SC | PCC | GC | DC |
|---|---|---|---|---|
| **Bagging** [3] | | | | |
| ipred [a], 2019 | ✓ | ✗ | ✗ | ✗ |
| adabag [2], 2018 | ✓ | ✓ | ✗ | ✗ |
| **RF** [4] | | | | |
| randomForest [3], 2018 | ✓ | ✓ | ✗ | ✗ |
| XGBoost Library [4], 2019 | ✗ | ✗ | ✓ | ✗ |
| Apache Spark MLlib [5], 2019 | ✗ | ✗ | ✗ | ✓ |
| **ET** [185] | | | | |
| extraTrees [6], 2016 | ✓ | ✓ | ✗ | ✗ |
| **RP** [186] | Not supported in R | | | |
| **RotF** [86] | | | | |
| rotationForest [7], 2017 | ✓ | ✗ | ✗ | ✗ |
| Rweka Rotation Forest [8], 2019 | ✓ | ✓ | ✗ | ✗ |
| **RR** [188] | Not supported in R | | | |

| Ensemble Method | SC | PCC | GC | DC |
|---|---|---|---|---|
| **AdaBoost** [5] | | | | |
| fastAdaboost [9], 2016 | ✓ | ✗ | ✗ | ✗ |
| adabag [2], 2018 | ✓ | ✗ | ✗ | ✗ |
| **LB** [191] | | | | |
| caTools [10], 2019 | ✓ | ✗ | ✗ | ✗ |
| Rweka LogitBoost, [8], 2019 | ✓ | ✓ | ✗ | ✗ |
| **GBM** [192] | | | | |
| gbm [11], 2019 | ✓ | ✗ | ✗ | ✗ |
| gbm3 [12], 2017 | ✓ | ✓ | ✗ | ✗ |
| Apache Spark MLlib [5], 2019 | ✗ | ✗ | ✗ | ✓ |
| H2O3 [13], 2019 | ✗ | ✗ | ✗ | ✓ |
| **XGBoost** [193] | | | | |
| XGBoost library [4], 2019 | ✓ | ✓ | ✓ | ✗ |
| H2O4GPU [14], 2019 | ✗ | ✗ | ✓ | ✗ |
| H2O3 [13], 2019 | ✗ | ✗ | ✗ | ✓ |
| sparkxgb [15], 2019 | ✗ | ✗ | ✗ | ✓ |
| **LightGBM** [195] | | | | |
| LightGBM library [16], 2019 | ✓ | ✓ | ✓ | ✗ |
| **CatBoost** [196] | | | | |
| CatBoost library, [17], 2019 | ✓ | ✓ | ✓ | ✗ |

[1] ipred: https://cran.r-project.org/web/packages/ipred
[2] adabag:https://cran.r-project.org/web/packages/adabag
[3] randomForest: https://cran.r-project.org/web/packages/randomForest
[4] XGBoost:https://xgboost.readthedocs.io
[5] Apache Spark: https://spark.apache. org/docs/2.1.0/mllib-ensembles.html
[6] extraTrees: https://cran.r-project.org/web/packages/extraTrees
[7] rotationForest: https://cran.r-project.org/web/packages/rotationForest
[8] Rweka: https://cran.r-project.org/web/packages/RWeka
[9] fastAdaboost: https://cran.r-project.org/web/packages/fastAdaboost
[10] caTools: https://cran.r-project.org/web/packages/caTools
[11] gbm: https://cran.r-project.org/web/packages/gbm
[12] gbm3:https://github.com/gbm-developers/gbm3
[13] H2O3:http://docs.h2o.ai/h2o/latest-stable/h2o-docs/welcome.html
[14] H2O4GPU:https://github.com/h2oai/h2o4gpu
[15] sparkxgb:https://cran.r-project.org/web/packages/sparkxgb/
[16] LightGBM:https://lightgbm. readthedocs.io/en/latest/index.html
[17] CatBoost:https://catboost.ai/docs/

**Table 3.2:** Comparison of state-of-the-art ensemble techniques.

| Ensemble Method | Year | Diversity Promotion | Type of base model |
|---|---|---|---|
| Bagging [3] | 1996 | Bootstrap sampling | Any classifier |
| RF [4] | 2001 | (1) Bootstrap sampling<br>(2) Random selection of $f$ feature subsets | DT |
| ET [185] | 2006 | Random selection of the splitting cut-point | DT |
| RP [186] | 2012 | Vertical and horizontal data patches | Any classifier |
| RotF [86] | 2006 | (1) Resampling<br>(2) Feature extraction of disjoint feature subsets | DT |
| RR [188] | 2016 | Random rotation of feature space | DT |
| AdaBoost [5] | 1997 | Selective sampling via weighting | Any classifier |
| LB [191] | 2000 | Training a weighted regression estimator | Logistic regression (LR) |
| GBM [192] | 2001 | Gradient search to minimize the loss function | Regression tree |
| XGBoost [193] | 2016 | 1) Regularized term in the objective function<br>2) Shrinkage and sub-sampling | Regression tree |
| LightGBM [195] | 2017 | Sampling from small gradient-instances | DT |
| CatBoost [196] | 2018 | Unbiased estimation of the gradient step | Oblivious trees |

## 3.5 Metaheuristic Algorithms For MCS

This part discusses the importance of metaheuristic algorithms to design an effective classifier ensemble. Most of the work was directed either to optimize the classifiers combination function [151, 198–201] or to find subset of classifiers from a large pool of members [29, 202, 203]. In addition, the inclusion of MA in some proposals to handle the multiclass classification tasks [204]. In those proposals, the optimization-based ensembles can predict better than the classical ensemble approach: Bagging, AdaBoost, Random subspaces, and majority voting.

**Metaheuristic weighted voting scheme:** Since there could be different performances for each classifier on the given dataset, the classifiers to be combined are considered with distinct weights. In this respect, by weighted voting schemes, the robustness and precision of the classifier ensemble can be increased. The assigning of appropriate weight values to classifiers can be modeled as an optimization problem whose optimal solutions can be offered by the well-established MA algorithms. In [200], based on the reliability of classification predictions, the authors applied genetic algorithms to utilize variant weights of classifiers for specific output classes. A multiobjective optimized weighted voting scheme has been applied in [205] for diagnosing heart disease. An optimized-based weighted voting

schema has been applied in [206] to create an effective remote sensing ensemble model. While differential evolution has been applied to optimize the weighted voting schema in ensemble learning [207].

In [151], the authors used GA as a meta-learner to learn a weight distribution vector $[w_1, w_2, \ldots, w_k, \ldots w_T]$. The proposed method is a modified stacking that assigns a weight $w_k$ for each classifier $\Psi_k$. Each classifier provides a class distribution vector that represents a particular instance's probability of belonging to each class. The final decision will be the highest score from multiplying the class distribution matrix by the weight distribution vector. Figure 3.1 shows the proposed topology from [151].

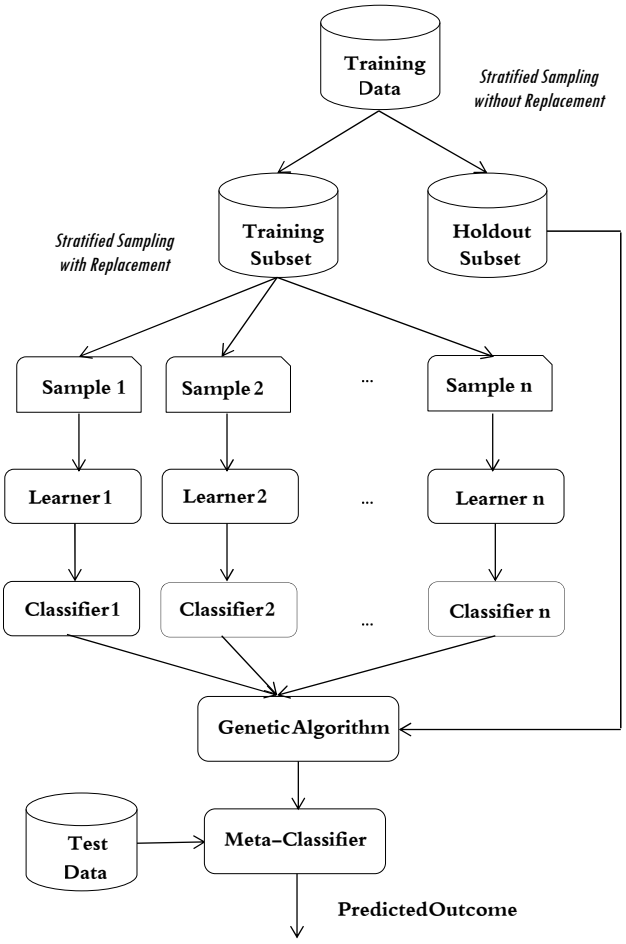


**Figure 3.1:** The stacking ensemble learning using GA, taken from [151].

In [198], Several weighted voting schemes via soft computing were tested, including those based on multi-objective differential evolution, multi-objective particle swarm, GA, and multi-objective simulated annealing. The novelty of their work was to apply a multi-objective differential evolution in ensemble classification for sentiment analysis. Specific weight is dedicated to each classifier based on the precision and recall values for each output class, as in Figure 3.2. From that figure, for the first output class, Classifier 3 has the maximum weighted voting power while for the second output class, Classifier 2 has the greatest weight.

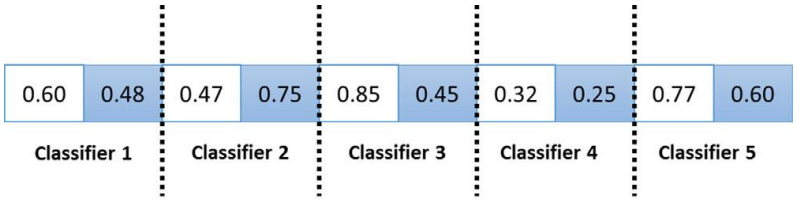


**Figure 3.2:** Weight vector for a problem with five classifiers and two classes, taken from [198].

**Metaheuristic for MCS pruning:** There is an extra intermediate step concerned with reducing the size of the ensemble before combining. [28, 34]. This phase is recognized as *ensemble selection, ensemble pruning, ensemble thinning*. By pruning down low-performance models, the effective ensemble can be acquired while retaining a high diversity among the remaining participants. Ensemble pruning can be formulated as an optimization problem to find a subset that optimizes a measure indicative. For that, metaheuristics are promising approaches to solve that challenge, with their capability to explore the search space. GASEN [29], uses a genetic algorithm to perform a stochastic search in the space of models. The ensemble is described as a string of bits, a specific model will be included or excluded based on its corresponding bit value [34]. The fitness function is the accuracy of the subensemble on a separate validation set. In [123], the authors formulated the ensemble pruning as a quadratic integer programming that searches for the minimum misclassification and maximum divergence of a fixed-size subset of $k$ classifiers. The authors used a semi-definite programming technique to solve the problem in polynomial time.

Direct hill-climbing explores the space of models through greedy search. Based on the current subensemble, the algorithm visits the neighborhood. The neighborhoods consist of those subsets that can be constructed by adding (removing) one classifier to (from) the current state. A directed hill-climbing that traverses the search space is represented as in Figure 3.3.

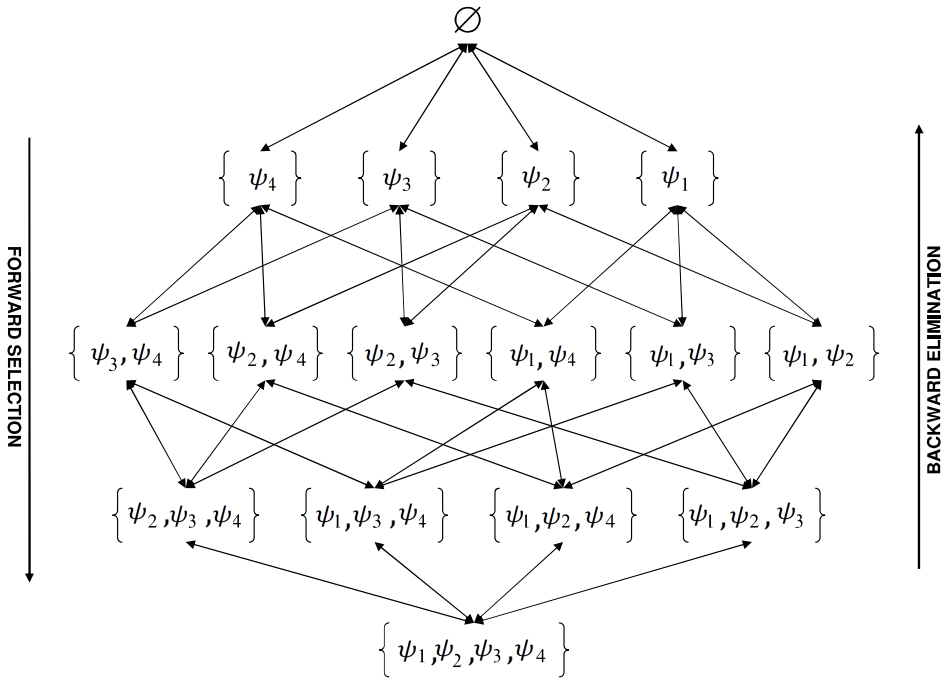


**Figure 3.3:** Ensemble pruning method of four models via directed hill climbing, taken from [34].

Depending on the direction of the search, we have forward selection [35, 208] and backward elimination [209, 210]. In both cases, $\frac{T(T+1)}{2}$ subsets had to be evaluated, leading to a time complexity of $\mathcal{O}(T^2 \cdot g(T,N))$. The term $g(T,N)$ concerns the complexity of the evaluation measure. In general, the main component for ensemble pruning is the evaluation measure upon which the subensemble could be identified. Evaluation measures can be divided into two major categories: *performance* based and *diversity* based. In addition, the idea of feature selection has been extended to the domain of ensemble reduction by transforming classifier predictions into artificial features to be reduced and selected by the harmony search algorithm [30]. Recently, the firefly algorithm has been modified to adapt to the

area of ensemble selection by focusing on discriminant base classifiers [126]. Furthermore, Onan et al. [26] proposed a hybrid ensemble pruning scheme based on clustering and multiobjective evolutionary algorithms. While heuristic metrics for MCS pruning will be discussed in Section 6.2.

## 3.6 Remarks

By comparing the literature on ensemble learning for classification tasks, the proposals in this thesis differ from other studies in several ways:

- As observable from our literature review on ensemble learning, most works have focused on using the available amount of data for training the base models [143, 151, 185, 188, 191, 192, 198, 204]. Regarding that, an additive computational complexity will be required to generate a large pool of classifiers, especially if we are dealing with complex models. *To partly solve this challenge*, intelligent data sampling in the form of IS techniques can be used to downsize the training data with their significant impact to generate MCS quickly.

- The performance of ensemble systems is restricted with generating accurate and diverse base models [12, 15, 16, 28, 98]. *To match this observation*, IS techniques could be applied to refine the accuracy of base models via focusing on pure and consistent data [211, 212]. While, the diversity could be achieved via using different base models [151, 213, 214] and via using a novel weight-based combination schema to weigh specific-class outputs [198, 200].

- SI algorithms [178] as a branch of MA proved their superiority over evolutionary algorithms. From our revision, there are recent SI proposals [179, 215, 216] with their unstudied effect for MCS. *To partly close that gap*, it was interesting to evaluate the performance of those SI algorithms for adjusting class-specific weights, to optimize model integration.

- The efficiency and efficacy of MCS had been proved theoretically and empirically via ensemble pruning [23, 217]. Again, the presented work for MCS

pruning [23, 24, 31, 32, 35–37] uses the whole data to train individual models. Furthermore, from [36], the accuracy of ensemble pruning is better optimized when it relies on both metrics of the classifier's accuracy and the ensemble diversity. However, each metric should be weighted differently (trade-off) to cope properly with the classification task at hand. *To fill that gap*, a guided search-based ensemble pruning will be proposed to solve the parameter tuning challenges. *In addition*, we downsize the data via IS and downsize multiple classifiers simultaneously without any negative effect on the general accuracy of MCS. *To prove our hypothesis*, the analysis of the results should be extended to include ensembles that are trained from nonreduced data.

- Related to our best knowledge, the presented work in this thesis is the first to combine between three intelligent computational paradigms: IS, MCS, and SI/ensemble pruning. *From the perspective of MCS*, the generation of the models and the predictive performance can be accelerated and improved, respectively. *From perspective of IS*, the limited accuracy of IS techniques can be significantly enhanced via MCS.

- Reordering-based MCS pruning metrics [35, 217, 218] are recognized as accurate and fast strategies to identify subset of classifiers properly. Since the analysis by Martínez-Muñoz et al. [23] in 2009, recent and efficient metrics have been proposed. Related to our best knowledge, the literature review is missing a deep analysis to compare all those superior metrics together. *In this thesis, we fill that gap* via complementing and extending the previous study which was conducted in [23].

*You cannot teach a man anything; you can only help him discover it in himself.*

Galileo

CHAPTER

# 4

# Training Set Selection and Swarm Intelligence for MCS

In data mining, a set of steps should be executed sequentially prior to the construction of any predictive model [19]. First, end-user requirements should be examined. Second, data preprocessing or data cleaning (removing inconsistent data, data transformation, selection of features, or selection of samples) should be applied. Finally, a suitable prediction model that considers feature characteristics should be utilized to maximize pattern extraction. Regarding that, a data reduction schema could be applied to select representative samples and to alleviate the computational complexity of training MCS. A data reduction algorithm that focuses on the selection of a subset of examples according to rules or heuristics is known with the name of IS [19]. The reduced data should maintain the integrity of the original data set; thus, the accuracy can be the same or even higher. There are two main types of IS from the literature: *Prototype selection* aims at identifying a reduced training set with higher classification accuracy for use by nearest neighbor classifiers (memory-based learning) [212]. *Training set selection* applies the same concept to obtain a subset as a training source for any data mining algorithm [219].

The taxonomy [212] categorizes most IS methods according to both the selection type and the search direction. Mainly by discussing the selection type:

- Condensation: Those kinds of techniques give high priority to retain decision boundary points with significant reduction capabilities gained from removing vast amounts of internal points, but the generalization accuracy over the test set can be negatively affected. Besides, the returned subset will be too small to train an effective ensemble.
- Edition: Those kinds of techniques seek to remove noisy border points, leaving a smoother decision boundary. The reduction capability of those techniques is acceptable with a nice improvement of generalization accuracy and we think that the returned subset will be valuable to train ensemble models with increased diversity, and increased predictive accuracy.
- Hybrid: Those kinds of techniques are hybrid between the above two selection strategies. They seek to find the smallest subset of examples, which maintain or increase the generalization accuracy. Again, the proposed ensemble cannot benefit from the too-small returned subset.

Under the above three categories, the IS techniques can be further grouped in sub-categories (search direction), which influence the time complexity and accuracy. Logically and related to brainstorm analysis, the methods which belong to the Edition category are preferred to return the free noise subset for training competitive ensembles.

Data reduction techniques are evaluated according to four criteria: consumed selection time, noise tolerance, reduction rate, and accuracy. The design of a high-speed IS algorithm with higher reduction capacity and with higher accuracy remains an open challenge. The selection time and the reduction capability are IS-dependent, while the accuracy from the reduced set can be elevated using MCS. From the perspective of data reduction, the loss in accuracy can be compensated by building MCS. In addition, it will be interesting to study how ensemble models could improve the learning from reduced data.

In the literature [143, 151, 185, 188, 191, 192, 198, 204], the MCS prediction model is often trained from the whole training data. Thus, an expected computational complexity will rise exponentially from both; the increasing number of training samples, and the number of individual classifiers. Other attempts to reduce the training data were inspired by random sampling, which could degrade base model accuracy. From the perspective of MCS, it will be great to reduce their

training complexity via learning from most representative samples. For that, intelligent data sampling, IS, could have a positive effect to generate promising MCS.

Finally, SI algorithms can be embedded to enhance the decision fusion of MCS. Recently, a set of new SI algorithms have been proposed: the moth-flame optimization algorithm (MFO) [215], the grey wolf optimizer (GWO) [179] and the whale optimization algorithm (WOA) [216]. For that, the recent SI algorithms can be applied to tune the weights of each classifier based on its accuracy to predict class samples. One of the objectives of this article is to highlight on the importance of training set selection and SI for enhancing the performance of MCS.

The remainder of this chapter is organized as follows: In Section 4.1, we present the motivations and the contributions. Section 4.2 discusses a class-weight combination function. The proposed framework and combination via SI algorithms are discussed in detail in Section 4.3. The experimental results and the discussion are presented in Sections 4.4 and 4.5, respectively. Finally, the conclusions of this study and future research are discussed in Section 4.6.

## 4.1 Motivations and Contributions

The major contribution of this proposal is a focus on the construction of a competitive ensemble learning from a reduced training set using the search capability of SI. The motivations and contributions that were considered while conducting this research are as follows:

1. The data quality is crucial in the learning paradigm of any classification algorithm. In this study, we use a general data reduction algorithm, namely, AllKNN [220] to train MCS without a loss of precision and with increased efficiency.
2. Since the accuracy of the reduced data is a property of the instance selection algorithm, we fill this gap by building MCS.
3. The proposed ensemble is heterogeneous as it conducts two additional steps to increase the diversity of the classifier set: bagging [3] and distance-based feature selection.
4. SI algorithms are included in the framework for the optimization of class-specific weights to enhance the integration process.

5. The evaluation metric, fitness function, of any meta-heuristic has a critical impact on the identification of a reliable solution. Here, we use the Mathews correlation coefficient (MCC) [221, 222] to evaluate the ensemble for each search agent.

## 4.2 Class-specific weight

Combination rules are used to obtain the final decision from combined classifiers; therefore, it is crucial to select them correctly. We will focus on weighted voting, which is based on the classifier combination rules of crisp labels. The discriminating power of MCS to categorize different class samples could be maximized by assigning a class-specific weight, as each classifier has different performance capabilities for the prediction of per class instances. The final decision is based on the weighted sum of several decisions, as in Equation (4.1).

$$\hat{\Psi}(\mathbf{x}) = \arg\max_{i \in \mathcal{M}} \sum_{k=1}^{T} \left[\Psi_k(\mathbf{x}) = i\right] w_{ik} \tag{4.1}$$

where $w_{ik}$ is a weight that is assigned to the $k$th classifier and the $i$th class. The weight settings play a pivotal role in controlling the accuracy of MCS [223].

The Belief value [141] acts as posterior probability, *weights for each class*, to measure the classification results from the confusion matrix of each classifier. Assume the confusion matrix of classifier $k$ is as expressed in Equation (4.2). The rows correspond to the original samples per class, while the columns correspond to the predicted class samples. Element $s_{ij}$ corresponds to the number of samples of class $i$ that are predicted as class $j$, and $s_{ii}$ denotes the number of samples from class $i$ that are correctly classified.

$$PT_k = \begin{pmatrix} s_{11} & s_{12} & \dots & s_{1M} \\ s_{21} & s_{22} & \dots & s_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ s_{M1} & s_{M2} & \dots & s_{MM} \end{pmatrix} \tag{4.2}$$

The posterior probability of classifier $k$ for unknown pattern $\mathbf{x}$ can be calculated via Eq. (4.3). In this chapter, this method will be referred to as *belief* and will be

used to form $M \times T$ dimensional matrix with $M$ classes and $T$ classifiers. The $k$th column in this matrix will be calculated from the diagonal elements of the $k$th confusion matrix using Equation (4.3).

$$P(\mathbf{x} \in i \Big| \Psi_k(\mathbf{x}) = j^{(k)}) = s_{ij}^{(k)} \Big/ \sum_{i=1}^{M} s_{ij}^{(k)} \tag{4.3}$$

## 4.3 Proposed Framework

The proposed framework is graphically presented in Figure 4.1. The training fold is reduced via AllKNN [220], which is an instance selection method, for the selection of representative samples. AllKNN virtually divides the training data into two parts: the selected instances and the neglected instances. After that, using stratified sampling with 65% from the selected instances and 35% from the neglected instances, a validation dataset is formed. Under laboratory experimentation, we tested various configurations of the above percentages, and the proposed percentages realized reasonable accuracy over a large proportion of the tested datasets. This validation data will be used to tune class-specific weights via SI algorithms in consideration of the MCC metric (Section 4.3.4). After optimizing the weights using SI algorithms, the ensemble is evaluated over the extracted unseen test set.

Two models will be trained for comparison: RFSM, which denotes random forest that is trained with AllKNN-selected data, and RFCOM, which denotes random forest that is trained on the original training data. The objective is to analyze the performance of the proposed MCS against those of RFSM, RFCOM, and the following combination methods of the built classifiers:

- Maj: Majority voting of the built classifiers.
- Belief: As discussed in Section 4.2.
- CW-NN: A neural network will be used to optimize class-specific weights.
- Stacking: A neural network will be used as a meta-classifier to tune the classifiers' weights.

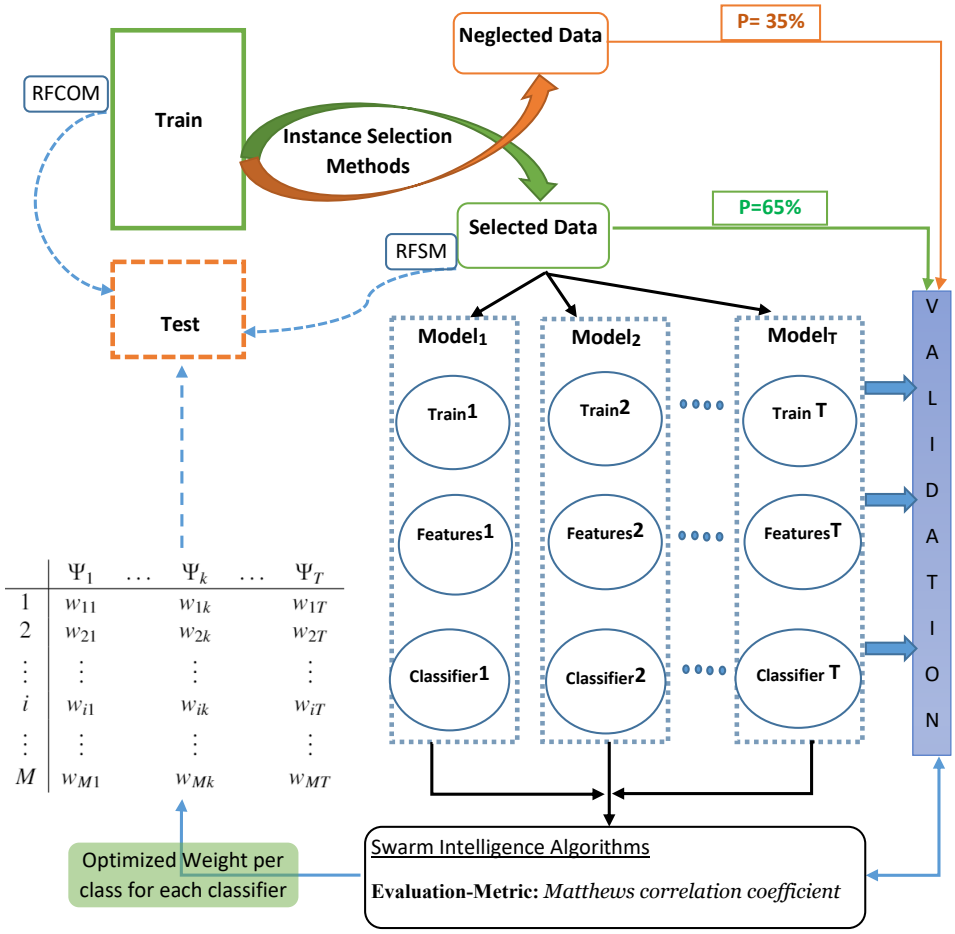


**Figure 4.1:** The proposed framework for building MCS from reduced training set.

### 4.3.1 Training Set Selection

AllKNN [220] removes noisy border points to produce a smoother decision boundary. The reduction capacity of this algorithm is not high, but acceptable prediction accuracy is realized over the test data. For $i = 1$ to $K$ neighbors, flag as incorrect any instance that is not classified correctly by its $i$ nearest neighbors. After completing the loop, all instances that have been flagged as bad are removed [224]. This method performs well if the classes are represented sufficiently well. Otherwise, the total disappearance of minor classes can be induced. This method depends on $K$ neighbors and the distance function. The default setup of $K = 3$ with the Euclidean distance has been used.

AllKNN is selected as the instance selection method because it reasonably balances the reduction rate, the accuracy of the resulting model, and the execution time; as demonstrated in [212].

### 4.3.2 Proposed MCS

MCS will be built from the selected data with the following characteristics:

- Samples diversity: Bagging is used to obtain various training samples for each individual classifier.
- Feature space diversity: Sixty percent of the features are selected randomly for each classifier. This procedure is repeated 20 times before selecting the most diversified set with the highest Hamming distance over the selected sub-features.
- Learning model diversity: Five types of classification models, namely, *DT*[1], *NB*[2], *JRip*[3], *Multinom*[4] and *KNN*[5] are used with a proportional representation of 20% by each model from the whole pool size. Those models are described in Section 2.3.

### 4.3.3 Proposed Candidate Solution

The power of SI relies upon stochastic operators [178, 225], information sharing, and the preservation of search space information throughout the iterations [216]. SI algorithms will be used to improve the fusion of MCS by optimizing the decision weight of each classifier based on its class prediction. Each candidate solution $(X)$ from the population of search agents simulates the same representation schema, as expressed in Equation (4.4). Thus, each solution represents practical weights to be used for multiple-decision fusion. All weights $w_{ik}$ are restricted to be in [0,1]

[1]Package C50 :https://cran.r-project.org/web/packages/C50/index.html
[2]Package e1071:https://cran.r-project.org/web/packages/e1071/index.html
[3]Package RWeka:https://cran.r-project.org/web/packages/RWeka/index.html
[4]Package nnet:http://cran.r-project.org/web/packages/nnet/index.html
[5]Package caret:https://cran.r-project.org/web/packages/caret/index.html

$\forall i \in \{1, 2, \ldots, M\}, \forall k \in \{1, 2, \ldots, T\}$

$$\begin{array}{c|ccccc} & \Psi_1 & \ldots & \Psi_k & \ldots & \Psi_T \\ \hline 1 & w_{11} & & w_{1k} & & w_{1T} \\ 2 & w_{21} & & w_{2k} & & w_{2T} \\ \vdots & \vdots & & \vdots & & \vdots \\ i & w_{i1} & & w_{ik} & & w_{iT} \\ \vdots & \vdots & & \vdots & & \vdots \\ M & w_{M1} & & w_{Mk} & & w_{MT} \end{array} \tag{4.4}$$

### 4.3.4 Proposed Objective Function

The accuracy of the proposed MCS is optimized according to the MCC metric [221]. MCC is a correlation coefficient between the target and the prediction. It is always in the interval [-1,1], where +1 corresponds to a completely correct prediction. MCC realizes a good satisfactory compromise among discrimination capability, consistency, and coherent behavior in binary and multiclass problems [222]. Each candidate solution ($X$) from the social flock represents a potential set of weight coefficients that should be tuned to maximize MCC(*pred($X$), target*); Equation (4.5). Where, *pred($X$)* is the ensemble prediction over the validation set using fusion weights of $X$, and is calculated via Equation (4.6). While *target* is the real class column from the validation set.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4.5}$$

$$pred(X) = \hat{\Psi}(\mathbf{x}) = \arg\max_{i \in M} \sum_{k=1}^{T} \left[\Psi_k(\mathbf{x}) = i\right] w_{ik} \tag{4.6}$$

### 4.3.5 Moth-Flame Optimization Algorithm

The MFO algorithm, which was proposed in [215], was inspired by the navigation method of moths during the night. Moths use transverse orientation for navigation according to the moonlight, but their movement is affected by artificial human lights, which cause deadly spiral paths. This algorithm mimics the death behavior

of moths that is caused by following artificial lights. Consider a feasible search space $S = \{X_1, X_2, ...., X_N\}$ that contains a set of positions for $N$ moths. Each moth position ($X_i$) represents the potential fusion weights for aggregating a set of classifiers. The promising areas that are identified during the search process are flagged or pinned as lights (flames); therefore, other moths can follow those lights to search for food. Hence, the initial fusion weights will be tuned by searching around promising weights iteratively.

Due to the intelligent mimic, it is possible to discover the global and local regions in the search space by simulating the spiral path of a moth around a flame via Equation (4.7). The new position of the moth will be on a hyper ellipse around the flame with a random number $\hat{t} \in [r, 1]$, where $r$ is linearly decreased from -1 to -2 throughout the iterations. While $D_i$ is the distance between the moth ($X_i$) and the flame ($F_j$), which is calculated via Equation (4.8) and $b$ is a constant for defining the shape of the logarithmic spiral, it takes +1 in the open-source code.

$$S(X_i, F_j) = D_i \cdot e^{b\hat{t}} \cdot cos\Big(2\pi \cdot \hat{t}\Big) + F_j \tag{4.7}$$

$$D_i = |F_j - X_i| \tag{4.8}$$

The flame matrix collects the best fusion weights for each moth during the search process. Consequently, it is useful for retaining the previous best solutions, and it will specify the lights where other moths can fly around to find more promising weights. The flame matrix is updated by considering both the current population and the past population of moths, which are denoted as $P_t$ and $P_{t-1}$, respectively. The best weights from the past and the current iterations are sorted in descending order to serve as the source for updating the current moth positions in the next iteration via Equation (4.7). To reduce the stochastic process in the final iterations, a focus on the exploitation milestone is realized by reducing the number of flames throughout the iterations via Equation (4.9).

$$N\_flames = round\left(N - t * \frac{N-1}{t_{max}}\right) \tag{4.9}$$

where $N$, $t$, and $t_{max}$ represent the maximum flame size, the current iteration number, and the maximum number of iterations, respectively. Via this methodology,

only the best flame, which corresponds to the best-obtained fusion weight, will remain at the final iteration to guide all other moths. The pseudocode of MFO is presented in Algorithm 10.

**Algorithm 10:** The pseudocode of MFO algorithm.

**Input :** Population $P = \{X_1, X_2, ...., X_N\}$, Parameters ($r, \hat{t}$, $N\_flames$), decisions of classifiers over validation set, real class column from validation (target).

**Output:** $X^*$ */* Optimized set of weight coefficients*

**begin**

*Calculate Fitness*: Mcc(pred($X_i$),target), $i \in \{1, 2, 3, ..., N\}$ via Eq. (4.5);

**for** $t \leftarrow 1$ **to** $(t_{max} - 1)$ **do**

Update number of flames via Eq. $(4.9)$;

**if** *iteration* $== 1$ **then**

Flames= sort(P);

**else if** *iteration* $> 1$ **then**

Flames=sort($P_{t-1}, P_t$);

Update r and $\hat{t}$;

Calculate $D$ via Eq. $(4.8)$ for each Moth;

Update Moth position via Eq. $(4.7)$;

Calculate fitness for each updated Moth;

Return $X^*$;

### 4.3.6 Grey Wolf Optimizer

GWO, which was proposed in [179], is inspired by the hierarchal leadership and hunting mechanism of grey wolves against prey. Grey wolves follow a strictly dominant social hierarchy. The most powerful wolf is $\alpha$ which is responsible for making decisions about hunting, sleeping place, and migration. The $\beta$ wolves are members in the second level that obey the alpha's decisions and give commands to the $\delta$ wolves. The $\delta$ wolves include scouts, sentinels, elders, hunters, and caretakers. The $\omega$ wolves are the lowest-ranking wolves in the group and play the role

of scapegoat when there is no food. Let $S \subset R^n$ be the feasible search space, where $S = \{X_1, X_2, ...., X_N\}$ contains the set of positions for $N$ grey wolves. Each solution ($X_i$) represents possible fusion weights for the aggregation of a set of classifiers. The best grey wolves ($\alpha, \beta, \delta$) provide suitable estimates of the location of the prey (suitable fusion weights) and guide the other $\omega$ wolves in tracking and chasing the prey. Before attacking, the $\omega$ wolves encircle and harass the prey until it stops moving according to Eq. (4.10).

$$\begin{aligned} X_1 &= X_\alpha(t) - A_1 \cdot D_\alpha \qquad D_\alpha = |C_1 \cdot X_\alpha - X(t)| \\ X_2 &= X_\beta(t) - A_2 \cdot D_\beta \qquad D_\beta = |C_2 \cdot X_\beta - X(t)| \\ X_3 &= X_\delta(t) - A_3 \cdot D_\delta \qquad D_\delta = |C_3 \cdot X_\delta - X(t)| \end{aligned} \tag{4.10}$$

where $X(t)$ is the fusion weights to be enhanced by each $\omega$ wolf. The controlling vectors $A$ and $C$ balance exploration and exploitation via Equation (4.11) with $r_1$ and $r_2$ as random vectors $\in$ [0,1]. $C$ returns a random value in the interval $[0, 2]$. The effect of gravity on the prey is increased when $C > 1$. The value of $A$ is defined by the parameter $a$, which is linearly decreased from 2 to 0 throughout the iterations. The range of $A$ is always in the interval $[-2, 2]$. Exploration is promoted in the first half of the iterations when $A > 1$ or $A < -1$, whereas exploitation is forced in the second half of the iterations when $-1 < A < 1$.

$$\begin{aligned} C &= 2 \cdot r_2 \\ A &= 2a \cdot r_1 - a \\ a &= 2 - t(\frac{2}{t_{max}}) \end{aligned} \tag{4.11}$$

Unsuitable fusion weights, $\omega$ wolves, will be tuned by chasing and searching around the proper fusion weights that are estimated by $\alpha$, $\beta$, and $\delta$ wolves using Equation (4.12).

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \tag{4.12}$$

Where $X_1$, $X_2$, and $X_3$ are calculated as in Equation (4.10). According to the discussion above, GWO depends on very few parameters ($a$, $A$, and $C$) that are updated prior to position updating. The pseudocode of GWO is presented in Algorithm 11.

**Algorithm 11:** The pseudocode of GWO algorithm.

**Input :** Population $P = \{X_1, X_2, ...., X_N\}$, Parameters $(a, A, C)$, decisions of classifiers over validation set, real class column from validation (target).

**Output:** $X_\alpha$ */* Optimized set of weight coefficients*

**begin**

*Calculate Fitness*: Mcc(pred($X_i$),target), $i \in \{1, 2, 3, ..., N\}$ via Eq. (4.5);

$X_\alpha \leftarrow$ The best search agent;

$X_\beta \leftarrow$ The second best search agent;

$X_\delta \leftarrow$ The third best search agent;

**for** $t \leftarrow 1$ **to** $(t_{max} - 1)$ **do**

Update position $\forall\omega$ via Eq. $(4.10, 4.12)$;

Update $a$, $A$, and $C$ via Eq. $(4.11)$;

Calculate fitness for each updated wolf;

Update $X_\alpha, X_\beta, X_\delta$;

Return $X_\alpha$;

### 4.3.7 Whale Optimization Algorithm

This algorithm was originally developed in [216] for mimicking the hunting behaviors of humpback whales. These whales are intelligent creatures that can sense and learn. The mathematical model simulates the feeding of humpback whales on krill and small fish herds by creating a bubble in a spiral shape around them. The optimization algorithm is similar to the GWO for encircling the prey, as described in Section 4.3.6. The power of this algorithm is derived from its focus on fewer stochastic operators and its balancing between exploration and exploitation. Suppose $S = \{X_1, X_2, ...., X_N\}$ is the feasible search space with a set of positions for $N$ whales. Each whale $(X_i)$ represents a prospective fusion weights for the aggregation of a set of classifiers. During the first half of the iterations, each whale $X_i$ is forced to discover search regions by selecting a whale randomly and moving away from its location according to Equation (4.13).

$$\begin{aligned} X_i(t+1) &= X_{rand} - A \cdot D \\ D &= |C \cdot X_{rand} - X_i(t)| \end{aligned} \tag{4.13}$$

where $A$ and $C$ are coefficient vectors having the same effect and values as calculated by Equation (4.11). Exploration is promoted in the first half of the iterations when $A > 1$ or $A < -1$, whereas exploitation is forced in the second half of the iterations when $-1 < A < 1$.

During the second half of the iterations, the other whales $X_i$ adapt their location by encircling $X^*$ (suitable fusion weights) via Equation (4.14) or by updating their spiral positions via Equation (4.15), based on a random parameter $p$.

$$\begin{aligned} X_i(t+1) &= X^*(t) - A \cdot D \\ D &= |C \cdot X^*(t) - X_i(t)| \end{aligned} \tag{4.14}$$

$$\begin{aligned} X_i(t+1) &= D' \cdot e^{b\ell} \cdot cos(2\pi\ell) + X^*(t) \\ D' &= |X^*(t) - X_i(t)| \end{aligned} \tag{4.15}$$

As a result, the local search and the exploitation process can be enhanced. $D'$ represents the distance between two whales, $b$ is a constant for defining the shape of the logarithmic spiral and $\ell$ is a random number in [-1,1]. The pseudocode of WOA is presented in Algorithm 12. *The implementations of the SI algorithms that have been used in this chapter are from R package*: metaheuristicOpt. [1]

## 4.4 Experimental Results

The experiments are dedicated to achieving objective 1, to build more diverse and highly accurate MCS only from a reduced portion of the available data. The two main questions to be answered are:

- $\boldsymbol{Q_1}$: What is the impact of reduced and consistent data on the performance of ensemble learning?
- $\boldsymbol{Q_2}$: Is it possible with the search capability of swarm intelligence to enhance the combination of classifiers?

[1] SI:https://cran.r-project.org/web/packages/metaheuristicOpt/index.html

**Algorithm 12:** The pseudocode of WOA algorithm.

**Input :** Population $P = \{X_1, X_2, ...., X_N\}$, Parameters ($a, A, C, \ell,\ p$), decisions of classifiers over validation set, real class column from validation (target).

**Output:** $X^*$ */* Potential set of weight coefficients*

**begin**

Initialize $a, A, C, \ell, p$ ;

*Calculate Fitness*: Mcc(pred($X_i$),target), $i \in \{1, 2, 3, ..., N\}$ via Eq. (4.5);

$X^* =$ The best search agent;

**for** $t \leftarrow 1$ **to** $(t_{max} - 1)$ **do**

  **if** $\mid A \mid \geqslant 1$ **then**

    Select random search agent $X_{rand}$;

    Update position of $X_i$ via Eq. $(4.13)$;

  **else if** $\mid A \mid < 1$ **then**

    **if** $p < 0.5$ **then**

      Update position of $X_i$ via Eq. $(4.14)$;

    **else if** $p \geqslant 0.5$ **then**

      Update position of $X_i$ via Eq. $(4.15)$;

  Calculate the fitness of each search agent;

  Update $X^*$ if there is better solution;

  Update Parameters ($a, A, C, \ell, p$);

Return $X^*$;

### 4.4.1 Setup of Experiments

During setup and validation, all the datasets are preprocessed by unifying the scales of the features via normalization. For each dataset, 10 repetitions with a 5-fold cross-validation procedure are considered. Therefore, a total of 50 runs are conducted per dataset. The accuracy metric is used to evaluate the proposed strategy. The parameters of SI *(without tuning)* are as follows: Population Size= 100, Iterations = 80 and for the *genetic algorithm,* Pm = 0.1 and Pc = 0.8.

A total of 25 datasets that were obtained from OpenML[1] and KEEL [2] are used in this study for experimentation. The characteristics of the data are presented in Table 4.1, where #S, #F, #C, and R represent the number of samples, the number of features, the number of classes, and the ratio between the smallest and the largest class for each dataset respectively. The number of classes varies from 2 to 15, while the maximum number of features is 180, and the ratio of minor class to the major ranges from 0.019 to 1.0.

**Table 4.1:** The characteristics of the selected datasets, *sorted by samples and classes*.

| DataSet | #S | #F | #C | R | DataSet | #S | #F | #C | R |
|---|---|---|---|---|---|---|---|---|---|
| Sonar | 208 | 60 | 2 | 0.874 | Newthyroid | 215 | 5 | 3 | 0.2 |
| Heart-statlog | 270 | 13 | 2 | 0.8 | Balance | 625 | 4 | 3 | 0.170 |
| Ionosphere | 351 | 33 | 2 | 0.56 | Dna | 3 186 | 180 | 3 | 0.463 |
| Saheart | 462 | 9 | 2 | 0.529 | Waveform | 4 999 | 21 | 3 | 0.972 |
| Wdbc | 569 | 30 | 2 | 0.594 | Vehicle | 846 | 18 | 4 | 0.913 |
| Wisconsin | 683 | 9 | 2 | 0.538 | Heart-long-beach | 200 | 13 | 5 | 0.196 |
| Australian | 690 | 14 | 2 | 0.802 | Thyroid-dis | 2 800 | 26 | 5 | 0.019 |
| German | 1 000 | 20 | 2 | 0.429 | Dermatology | 358 | 34 | 6 | 0.18 |
| Biodegradation | 1 055 | 41 | 2 | 0.509 | Mfeat-zernike | 2 000 | 47 | 10 | 1.0 |
| Diabetic | 1 151 | 19 | 2 | 0.884 | Mfeat-karh | 2 000 | 64 | 10 | 1.0 |
| Ringnorm | 7 400 | 20 | 2 | 0.981 | Mfeat-fourier | 2 000 | 76 | 10 | 1.0 |
| Twonorm | 7 400 | 20 | 2 | 0.998 | Movement-libras | 360 | 90 | 15 | 1.0 |
| Tae | 151 | 5 | 3 | 0.942 | | | | | |

### 4.4.2 Experiment 1

The pool size $T$ is crucial, and this size should be analyzed carefully to avoid model duplication and degradation of the overall performance. The proposed ensemble was tested with 8 sizes $\in \{5, 10, 15, 20, 25, 30, 40, 50\}$ over 25 datasets with 50 runs per dataset (10 replicas $\cdot$ 5 folds) to obtain 10,000 record (8·25·50) of results. Each row from Table 4.2 corresponds to the average accuracy of 1250 records, which is used to identify a suitable ensemble size for the presentation of the final results. From Table 4.2, the highest average accuracy is obtained with $T = 50$. GA is competitive with MFO over various ensemble sizes. The poorest values from the proposed SI algorithms are presented in *italics* and correspond

[1]Machine Learning Repository: https://www.openml.org
[2]KEEL Repository: http://www.keel.es/

**Table 4.2:** Ensemble size analysis for higher prediction accuracy over all 25 datasets. The best two values are shown in bold.

| | Random Forest | | Individual Models | | | | | Cobmination Methods | | | | Combination by GA and SI | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | RFCOM | RFSM | DT | NB | Mlnom | KNN | JRip | Maj | Belief | CW-NN | Stacking | GA | GWO | MFO | WOA |
| 5 | 78.26 | 77.41 | 72.10 | 70.87 | 74.95 | 72.82 | 71.07 | 78.01 | 78.43 | 77.73 | 77.51 | **78.99** | 78.96 | **78.97** | *78.89·* |
| 10 | **80.44** | 78.77 | 74.09 | 73.02 | 76.78 | 74.88 | 73.06 | 78.56 | 79.40 | 78.71 | 77.33 | **80.30** | 80.13 | 80.23 | *80.03·* |
| 15 | **81.32** | 79.39 | 75.18 | 74.29 | 77.70 | 75.77 | 73.83 | 79.48 | 79.88 | 78.99 | 77.15 | **80.91** | 80.72 | 80.88 | *80.53·* |
| 20 | **81.58** | 79.63 | 75.76 | 74.77 | 78.28 | 76.32 | 74.79 | 79.49 | 80.16 | 78.95 | 76.66 | **81.07** | 80.93 | 80.96 | *80.63·* |
| 25 | **82.05** | 79.79 | 76.19 | 75.16 | 78.67 | 76.74 | 75.14 | 79.86 | 80.32 | 78.94 | 76.28 | 81.16 | 81.15 | **81.17** | *80.77·* |
| 30 | **82.16** | 79.86 | 76.42 | 75.41 | 78.92 | 77.14 | 75.48 | 79.66 | 80.25 | 78.69 | 75.84 | 81.15 | 81.23 | **81.27** | *80.74·* |
| 40 | **82.51** | 79.98 | 76.86 | 76.00 | 79.17 | 77.47 | 75.98 | 79.83 | 80.34 | 78.56 | 75.40 | **81.22** | 81.18 | 81.21 | *80.79·* |
| 50 | **82.42** | 80.12 | 77.40 | 76.51 | 79.67 | 77.86 | 76.47 | 80.07 | 80.58 | 78.23 | 74.91 | 81.30 | 81.38 | **81.49** | *81.07·* |

to WOA; however, they are better than those of other individual classifiers and combination methods. DT represents the average accuracy of the best individual of type DT. The same holds for all best individuals of types NB, Mlnom, KNN, and JRip. Figure 4.2 presents the superior performance of both (GA, MFO) over various combination methods. The most interesting result in this graph is that the proposed fusion method at a smaller ensemble size with $T = 15$ (*dotted horizontal line*) outperforms other combination methods for large ensemble sizes.

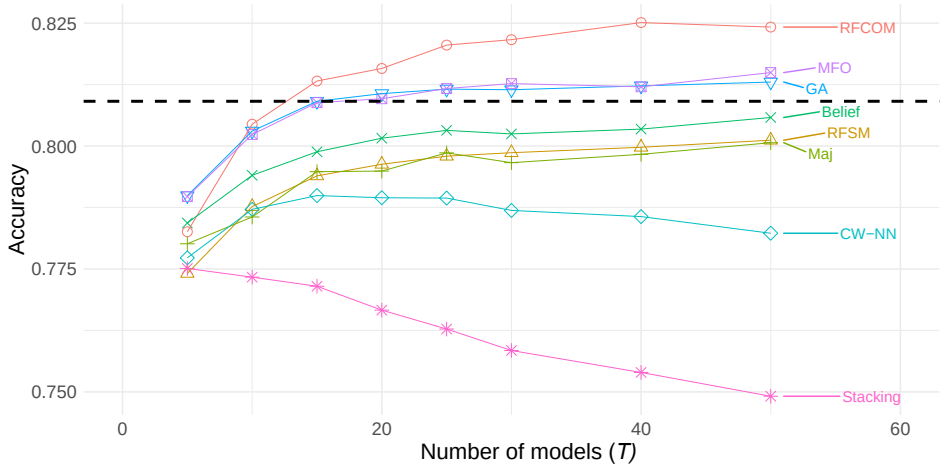


**Figure 4.2:** Correlation between ensemble size and prediction accuracy that is averaged over all the datasets.

### 4.4.3 Experiment 2

To present the final results, we selected an ensemble size of $T = 50$. Hence, we have 10 models for each classifier type $\in \{$DT, NB, Mlnom, KNN, JRip$\}$. Table 4.3

presents the average accuracy on each dataset that is realized by RFCOM, RFSM, the best individual type, and various combination methods (Maj, Belief, CW-NN, Stacking). The results will be analyzed in terms of the research questions.

**To answer $Q_1$**, we analyze the effect of data reduction. For that, the effect of AllKNN [220] is analyzed via comparing the results that were obtained by both RFCOM and RFSM, in the presence of a realized reduction rate that results in fast ensemble learning. From Table 4.3, RFSM realized only 4 improvements over RFCOM according to ($D_2$, $D_4$, $D_9$, $D_{17}$). The highest improvement in accuracy over RFCOM reaches 3.47% for $D_{17}$. For $D_8$, the reduction capacity (Red-Rate) is the highest, where the reduced training subset reaches 18.75% on the complete training set without a large deviation in precision from the RFSM side. Furthermore, for $D_5$, RFCOM outperforms RFSM with 1.85%, but it uses the complete training set, while RFSM uses only 45% as a training subset. Finally, the training set selection time (Tss-TM) is problem-dependent and differs according to the number of samples and the number of features. The slowest time is 220.59 seconds for $D_{21}$, while the fastest selection time is 2.4 seconds for $D_8$.

**To answer $Q_2$**, we analyze the effect of SI. The prediction accuracy of the proposed ensemble is compared with those of individual classifiers, various combination methods, and RFSM. The comparison includes the last 14 columns from Table 4.3 since all use only the reduced data. The best prediction accuracy for each dataset is highlighted in bold, while the value in italics represents the best from the proposed combination. The results demonstrate the performance of Mlnom as an individual classifier and Belief as a combination method. The average accuracy for each base classifier is based on the selection of the best classifier from the corresponding type; however, the degradation in the performance of those individuals is too large compared to the performances of the proposed strategies. For example, the accuracy of Mlnom reaches 71.17% according to $D_{16}$, while the proposed fusion realizes 97.23%. For $D_2$, Mlnom classifies 73.6% correctly, while 86.5% is the highest prediction accuracy that is realized by SI. The highest prediction by Mlnom compared with SI has been recorded as 6.7% for $D_8$. Thus, the deviation in the prediction accuracy level is highly significant. For the combination methods, the Belief function realizes 91.7% accuracy for $D_{16}$, while the highest prediction accuracy is 97.2% by SI. At least one SI algorithm outperforms RFSM for all 25

**Table 4.3:** Average accuracy for $T = 50$. The best two values are in bold, while the best from GA and SI is in *italic*.

| | | AllKNN | | Random Forest | | Individual Models | | | | | Combination Methods | | | | Combination by GA and SI | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # | Data | Red_Rate | Tss_TM | RFCOM | RFSM | DT | NB | Mlnom | KNN | JRip | Maj | Belief | CW-NN | Stacking | GA | GWO | MFO | WOA |
| $D_1$ | Australian | 0.26 | 9.13 | 86.97 | 86.13 | **87.29** | 86.01 | **87.22** | 86.67 | 86.45 | 85.97 | 86.16 | 83.48 | 83.58 | 86.23 | 86.04 | *86.33*· | 86.13 |
| $D_2$ | Balance | 0.28 | 6.45 | 84.18 | 85.73 | 71.76 | 73.62 | 73.60 | 72.21 | 72.13 | 82.06 | 82.29 | 83.76 | 86.21 | **86.31** | 86.23 | ***86.50***· | 86.10 |
| $D_3$ | Biodegradation | 0.25 | 21.99 | **86.25** | 85.26 | 84.67 | 73.18 | **86.81** | 84.61 | 84.07 | 85.64 | 86.16 | 83.86 | 83.24 | 85.93 | *86.08*· | 86.00 | 85.89 |
| $D_4$ | Dermatology | 0.07 | 8.21 | 97.07 | 97.13 | 95.95 | 94.45 | 95.81 | 95.90 | 90.42 | 97.46 | **97.52** | 92.96 | 87.35 | 97.46 | 97.46 | ***97.49***· | 97.13 |
| $D_5$ | Diabetic | 0.55 | 13.42 | 67.10 | 65.88 | 66.78 | 63.66 | **73.12** | 65.65 | 67.48 | 66.25 | 67.28 | 67.55 | 67.74 | 68.08 | 68.88 | 68.53 | ***68.93***· |
| $D_6$ | Dna | 0.36 | 199.93 | 94.80 | 93.50 | 88.86 | 82.17 | 87.74 | 73.55 | 87.47 | 94.10 | 94.84 | 92.96 | 93.94 | 95.04 | **95.06** | ***95.13***· | 94.78 |
| $D_7$ | German | 0.44 | 14.71 | **76.31** | 72.18 | 73.67 | 74.38 | **75.10** | 72.52 | 73.31 | 71.50 | 71.26 | 70.03 | 70.06 | 73.17 | *73.50*· | 73.47 | 72.68 |
| $D_8$ | Hert-Beach | 0.81 | 2.43 | 33.26 | 31.49 | **38.36** | 36.76 | **38.40** | 37.64 | 37.35 | 31.51 | 31.49 | 28.85 | 29.83 | *31.66*· | 31.14 | 31.62 | 30.58 |
| $D_9$ | Heart-Statlog | 0.34 | 6.53 | 81.33 | 81.37 | 83.04 | **84.11** | **85.00** | 83.00 | 83.15 | 82.96 | 83.04 | 77.15 | 79.07 | *82.85*· | 82.41 | 82.48 | 81.37 |
| $D_{10}$ | Ionosphere | 0.18 | 12.18 | **93.28** | 91.31 | 90.83 | 91.60 | 88.55 | 86.33 | 91.26 | 89.21 | 89.55 | 91.20 | **91.97** | 90.72 | 90.43 | 91.14 | *91.60*· |
| $D_{11}$ | Mf-fourier | 0.28 | 71.93 | **82.71** | 80.02 | 72.27 | 75.76 | 71.47 | 77.36 | 68.57 | 79.94 | 80.19 | 69.86 | 54.46 | 80.57 | ***80.96***· | 80.86 | 80.14 |
| $D_{12}$ | Mf-karh | 0.07 | 62.94 | 95.46 | 94.86 | 78.49 | 91.61 | 89.37 | 93.01 | 76.44 | **96.01** | **96.07** | 89.04 | 65.35 | *96.00*· | 95.87 | 95.88 | 95.44 |
| $D_{13}$ | Mf-zernike | 0.26 | 51.83 | 77.08 | 75.60 | 67.32 | 73.42 | 78.77 | 80.02 | 64.82 | 80.62 | 80.26 | 76.62 | 57.38 | **80.83** | ***81.02***· | 80.75 | 80.30 |
| $D_{14}$ | Mov-Libras | 0.32 | 13.19 | **81.14** | 64.23 | 57.25 | 57.32 | 67.68 | 63.89 | 45.39 | 66.21 | **69.08** | 59.42 | 29.56 | 67.74 | *68.87*· | 68.66 | 67.21 |
| $D_{15}$ | Newthyroid | 0.07 | 3.75 | 95.81 | 93.86 | 95.49 | **98.05** | **97.30** | 95.86 | 95.02 | 95.12 | 95.40 | 93.44 | 95.44 | 95.67 | 95.44 | *95.91*· | 94.98 |
| $D_{16}$ | Ringnorm | 0.32 | 125.09 | 94.81 | 93.03 | 85.81 | 93.61 | 71.17 | 68.19 | 87.06 | 86.01 | 91.73 | 96.86 | 96.66 | 97.06 | ***97.23***· | **97.18** | 96.93 |
| $D_{17}$ | Sa-Heart | 0.49 | 5.19 | 67.97 | 70.32 | 73.20 | 73.40 | **74.57** | 72.81 | **74.18** | 71.10 | 71.36 | 66.49 | 66.97 | 71.02 | 71.13 | *71.62*· | 71.02 |
| $D_{18}$ | Sonar | 0.22 | 6.43 | **81.98** | 78.36 | 78.90 | 74.72 | 79.67 | **83.13** | 79.52 | 78.18 | 78.62 | 75.82 | 77.92 | 78.09 | 78.58 | 78.72 | *78.76*· |
| $D_{19}$ | Tae | 0.70 | 2.81 | **62.96** | 49.61 | 54.89 | **55.05** | 54.03 | 54.13 | 53.81 | 48.08 | 47.96 | 51.32 | 50.38 | 49.94 | 49.29 | *50.34*· | 49.72 |
| $D_{20}$ | Thyroid-dis | 0.48 | 48.04 | **70.79** | 69.84 | 68.32 | 35.40 | **69.96** | 67.60 | 68.18 | 69.58 | 69.35 | 65.46 | 63.93 | *69.95*· | 69.53 | 69.53 | 69.63 |
| $D_{21}$ | Twonorm | 0.08 | 220.59 | 96.78 | 96.66 | 84.91 | 94.75 | 94.61 | 91.41 | 88.58 | **97.49** | **97.51** | 96.50 | 96.59 | *97.38*· | 97.31 | 97.32 | 97.35 |
| $D_{22}$ | Vehicle | 0.41 | 12.78 | **74.89** | 70.47 | 68.18 | 55.82 | 72.31 | 67.19 | 66.18 | 68.77 | 69.03 | 69.22 | 69.34 | 71.40 | ***73.16***· | 72.64 | 71.31 |
| $D_{23}$ | Waveform | 0.32 | 78.21 | 84.76 | 84.46 | 76.99 | 81.68 | 84.02 | 79.14 | 78.23 | 84.40 | 84.86 | 82.69 | 83.12 | ***85.78***· | 85.39 | **85.66** | 85.38 |
| $D_{24}$ | Wdbc | 0.07 | 11.70 | 95.71 | 95.08 | 95.80 | 95.36 | **98.07** | **97.37** | 95.94 | 96.19 | 96.19 | 95.68 | 96.66 | 96.40 | 96.29 | 96.38 | *96.50*· |
| $D_{25}$ | Wisconsin | 0.06 | 9.49 | 97.09 | 96.59 | 95.93 | 96.97 | **97.51** | 97.22 | 96.85 | **97.33** | 97.32 | 95.43 | 96.03 | *97.25*· | 97.12 | 97.20 | 96.93 |
| | **AR-Friedman** | | | 5.44 | 9.5 | 9.42 | 9.66 | 5.92 | 8.92 | 10.04 | 8.48 | 6.94 | 11.64 | 10.86 | **5.4** | 5.94 | **4.84** | 7 |

datasets, and the highest percentages of improvement are 7.23% and 4.5% over RFSM for $D_{14}$ and $D_{16}$, respectively.

It is interesting to connect this part with the previous section to prove how the proposed fusion mechanisms outperform learning from nonreduced data in terms of prediction accuracy. Comparing with RFCOM, the proposed fusion mechanism using meta-heuristics outperforms RFCOM on 14 datasets $\{D_2, D_4, D_5, D_6, D_9, D_{12}, D_{13}, D_{15}, D_{16}, D_{17}, D_{21}, D_{23}, D_{24}, D_{25}\}$ with maximum improvement percentages of 5.38% and 5.11% on $D_{17}$ *and* $D_{13}$, respectively. Figure 4.3 presents the ordered histograms of all combination methods (left to right) for comparisons against RFCOM.

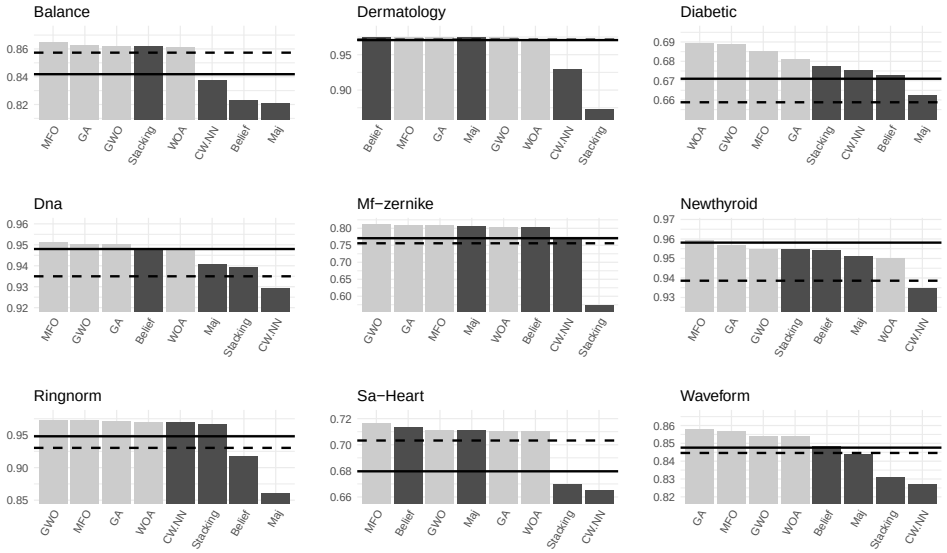


**Figure 4.3:** Comparing SI combination strategies against RFCOM (——) and RFSM (- - -). Dark gray columns denote combination methods, while light gray columns denote SI strategies.

### 4.4.4 Statistical Analysis of the Results

To compare the results, the null hypothesis of no improvement over the standard algorithms will be analyzed. Two nonparametric statistical tests, namely, the Friedman test [226] for multiple comparisons and the Wilcoxon signed ranks test [227] for pairwise comparisons, will be conducted. The average rank of the Friedman test

*(AR-Friedman)* is presented in the last row of Table 4.3, with the best ranks scored sequentially as MFO, GA, RFCOM, Mlnom, and GWO. The Friedman statistic that was distributed according to the chi-square distribution with 14 degrees of freedom is 84.603, and the computed p-value is smaller than 0.01%.

Next, the Wilcoxon test [227] aims at detecting significant differences between the two sample means. From Table 4.4, none of the standard combination methods (Maj, Belief, Stacking, and CW-NN) outperforms RFSM, while our proposed combination methods (GA, GWO, MFO, and WOA) all significantly outperform RFSM by 95%. Moreover, the proposed fusion methods are at the same level of competence as RFCOM. Further, Mlnom cannot outperform RFSM, Maj, or Belief which extends the superior performance of our proposed methods.

**Table 4.4:** Summary of the Wilcoxon test. ✓= the method in the row improves the method of the column. ○= the method in the column improves the method of the row. Upper diagonal of level significance $\alpha = 0.9$, lower diagonal level of significance $\alpha = 0.95$

| | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) | (13) | (14) | (15) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RFCOM (1) | - | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | |
| RFSM (2) | ○ | - | ✓ | ✓ | | | ✓ | | | ✓ | ✓ | ○ | ○ | ○ | ○ |
| DT (3) | ○ | | - | | ○ | | | | ○ | | | ○ | ○ | ○ | ○ |
| NB (4) | ○ | | | - | ○ | | | ○ | ○ | | | ○ | ○ | ○ | ○ |
| Mlnom (5) | | | ✓ | ✓ | - | ✓ | ✓ | | | ✓ | ✓ | | | | |
| KNN (6) | ○ | | | | ○ | - | | ○ | ○ | | | ○ | ○ | ○ | ○ |
| JRip (7) | ○ | ○ | | | ○ | | - | ○ | ○ | | | ○ | ○ | ○ | ○ |
| Maj (8) | ○ | | | ✓ | | | | - | ○ | ✓ | ✓ | ○ | ○ | ○ | ○ |
| Belief (9) | | | ✓ | ✓ | | | ✓ | ✓ | - | ✓ | ✓ | ○ | ○ | ○ | |
| CW-NN (10) | ○ | ○ | | | ○ | | | ○ | ○ | - | | ○ | ○ | ○ | ○ |
| Stacking (11) | ○ | ○ | | | ○ | | | ○ | ○ | | - | ○ | ○ | ○ | ○ |
| GA (12) | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | - | | ○ | ✓ |
| GWO (13) | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | | - | | ✓ |
| MFO (14) | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | - | ✓ |
| WOA (15) | | ✓ | ✓ | ✓ | | | ✓ | ✓ | | ✓ | ✓ | ○ | ○ | ○ | - |

After statistical analysis and according to the average rank of the Friedman test, we selected {MFO, GA, GWO, Mlnom, Belief, RFSM, RFCOM} and analyzed the distribution of their prediction accuracy levels. Figure 4.4 presents the range of the prediction accuracy around the median and how the proposed fusion strategies realize robust and stable prediction for $\{D_2, D_4, D_6, D_{13}, D_{16}, D_{23}\}$.

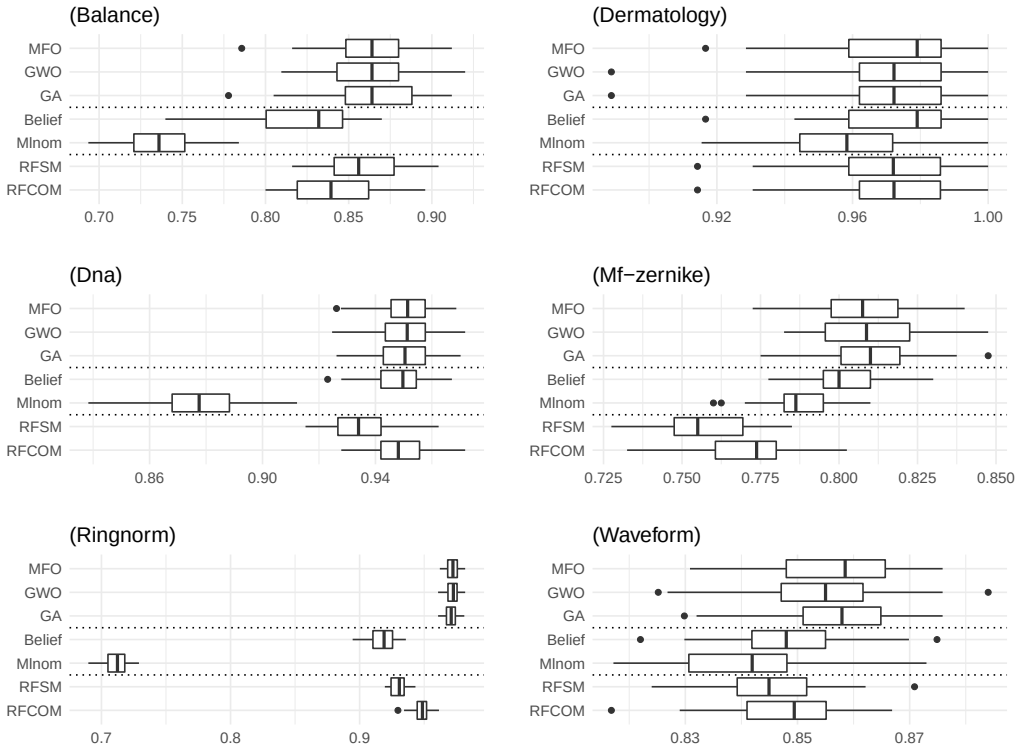


**Figure 4.4:** Distribution of the prediction accuracy.

### 4.4.5 Exploration of the Research Questions

In this part, we determine whether the research questions of the first objective are answered or not.

***($Q_1$)*** *The impact of reduced and consistent data on the performance of the ensemble learning* can be identified, but it depends mainly on the building schema of the ensemble.

- Random forest: RFSM outperformed RFCOM on only 4 datasets, as presented in the first part of Section 4.4.3.
- Proposed MCS in Section 4.3.2: The simple majority voting (Maj) of the proposed MCS outperformed RFCOM on 8 datasets ($D_4$, $D_9$, $D_{12}$, $D_{13}$, $D_{17}$, $D_{21}$, $D_{24}$, and $D_{25}$), while the class degree (Belief) outperformed RFCOM on 11 datasets ($D_4$, $D_5$, $D_6$, $D_9$, $D_{12}$, $D_{13}$, $D_{17}$, $D_{21}$, $D_{23}$, $D_{24}$, and $D_{25}$).
- The conclusion, IS proved its effectiveness to reduce the training data-size of

17 datasets by more than 25%. AllKNN, as IS technique, did not prove its capability to capture the integrity from the whole data. The proposed MCS could compensate the error of IS method.

***($Q_2$)*** *The search capability of swarm intelligence to enhance the combination of classifiers* has been evaluated and discussed in the last part of Section 4.4.3 with SI outperforming RFCOM on 14 datasets. The conclusion that SI algorithms are so effective to optimize the classifiers' combination function.

## 4.5 Discussion

The diversity of the proposed MCS is maximized via three steps (bagging, Hamming distance-based feature selection, and heterogeneous classifiers). Sixty percent of the features are selected randomly for each classifier according to the maximum Hamming distance over the ensemble. This percentage can be changed or tuned, or intelligent feature selection can be considered for the construction of the MCS.

The effect of scaling up the data to be handled by this framework has not been studied in this article. However, this is possible via the application of the stratification mechanism [212], via which a huge train can be stratified into several parts while preserving the class distribution in each part. Then, the reduced data will be formed by combining the outputs of the application of IS to each part individually.

The proposed method can be successfully used in predictive system learning, especially if the problem is characterized by high dimensionality due to both the number of attributes and the number of records.

The time complexity of the proposed framework can be divided into two parts: The first part is the *overhead time complexity*, which is proportional to the properties of the IS method. For that, AllKNN has been selected due to its reasonable selection time. The second part is the *ensemble time complexity*, where the MCS training time will be reduced due to the size of the selected data. In addition, the weight tuning time for the aggregation of classifier decisions by SI can be controlled by tuning the algorithmic parameters; the population size, and the number of iterations.

However, similar to popular approaches that are based on deep learning (DL), the proposed approach is characterized by a relatively high computational com-

plexity. Therefore, it is not suitable for *online* learning, e.g., in the case of non-stationary data classification streams, namely, when the *concept drift* phenomenon can occur. Instead, it can be a suitable alternative if the training time is not a critical parameter from the application perspective.

Nevertheless, the proposed framework can outperform DL methods according to the following: MCS has less tendency to overfit, especially in this proposed framework, because heterogeneous classifiers are trained from bootstrapped samples over various sub-features. DL is based on a complex model that fits a dataset well. Additionally, MCS can be more easily deployed in production systems than DL. Moreover, DL requires a large amount of data for success and its training is complicated as no precise method is available for obtaining the best set of hyper-parameters [228].

## 4.6 Conclusions

In this chapter, the application of SI and GA as complementary methods for enhancing information fusion in MCS was demonstrated. Calibration weights for combining multiple decisions have been optimized by considering the decision (class) of each expert (classifier) from previous results (validation set). Experimentation on the presented fusion framework has been conducted over 25 datasets. In searching for suitable weights, SI outperformed Stacking, Neural Network, and Belief. Moreover, fusion by SI outperformed simple majority voting and performed competitively with Random Forest. The proposed framework consists of two main connected paradigms: first, the use of instance selection methods for data cleaning, followed by the construction of heterogeneous multiple classifiers; second, the application of an improved weighted fusion of classifiers' predictions over the validation set using meta-heuristics.

It is recommended to conduct data reduction (instance selection and feature selection) prior to the construction of ensemble methods. Hence, the computational complexities of individual classifiers can be reduced. The reduced amounts of data can degrade the performance of the classification algorithms. Therefore, this chapter presents two strategies for increasing the prediction accuracy: First, on the data level, intelligent sample selection is used to reduce the number of data samples

and to select relevant data. Second, on the algorithm level, multi classifiers are designed and their decisions are fused by the search capabilities of meta-heuristic algorithms. To better evaluate the performances of the new fusion strategies, the statistically significant differences are calculated via rank-based transformations, and the superior performance of the moth-flame optimization algorithm (MFO) is demonstrated. After training on reduced data, SI outperforms Random Forest, which is trained on nonreduced data, in 14 out of 25 datasets. The distribution of accuracy levels has been analyzed to evaluate the consistency, robustness, and stability of the prediction.

In future research, the framework can be enhanced by applying promising reduction methods. Furthermore, static and dynamic classifier selection approaches can be applied to merge a subset of the classifiers instead of all classifiers. Finally, the hybrid swarm intelligence algorithms and multi-objective optimization will be used to enhance decision making.

*It is strange that only extraordinary men make the discoveries, which later appear so easy and simple.*

Georg C. Lichtenberg

CHAPTER

# 5

# A Guided Search for MCS pruning

In chapter 4, two intelligent paradigms have been discussed to enhance the performance of MCS. First, *on data level* via applying IS techniques to learn from representative samples. Second, *on the fusion level* via applying SI algorithms to combine a set of classifiers. However, the complexity of the investigated MCS is proportional to both the number of classes and the number of individual models. For example, to solve a 10-class problem using 200 classifiers, each candidate from the population of SI will be composed of 2000 elements (matrix of $10 \times 200$). This chapter discusses the alternative procedure to tackle this challenge. First, to keep on applying the IS techniques in the first part. Second. to perform a guided search for pruning the generated MCS.

Building ensemble models, in general, requires an answer to the following questions [141] (1) How many classifiers should we use? (2) What type of classifiers should be chosen? (3) Which feature subset should be presented to each classifier? (4) Which combination rules to be used?. To answer the above questions, the authors in [111, 162] argued that ensemble members should have both high accuracy and high diversity to gain more. Hence, manipulation of algorithms [229] and manipulation of the features [86] are answers to 2nd and 3rd questions,

to obtain different model structures that are trained from diverse feature spaces. Regarding the combination methods; many strategies, which vary between trained [229, 230] or untrained [15, 21], have been applied and also been analyzed in [143]. In this chapter, we concentrate on answering the 1st question via overproducing many learning models followed by ensemble pruning. The general objective is to improve the overall accuracy, speed up the classification process, and to save the computational and storage resources.

Despite the remarkable performance of the ensemble methods, a large-size ensemble can be considered as a drawback. In [37] and related to the investigated classification tasks, it has been noticed that the ensemble size can be reduced up to 60-80% without significant deterioration to its performance. Therefore, it is practically useless to keep all the ensemble members for the prediction process. In addition, when the models are distributed over a network, the reduction of models leads to the reduction of communication costs [34]. A possible solution to downsize/prune MCS can be broadly divided into the following five solutions [26, 27]:

- **Exhaustive search:** Requires an evaluation to all the possible $2^T-1$ nonempty subsets of classifiers. However, this is an NP-combinatorial search problem as the complexity grows exponentially with the ensemble size. Exhaustive search methods are only applicable when the number of classifiers is small [231], while they are unfeasible for typical large-size ensembles [23].
- **Optimization-based search:** Meta-heuristic search methods are the alternative to offer huge computational savings. All the search methods in this category optimize an evaluation metric. The metric can be diversity [232] in an attempt not to select similar classifiers and to obtain complementary subsets. Another popular metric is to reduce the ensemble error of a particular combination rule. Genetic Algorithms have been shown to find near-optimal solutions [132]. Other evolutionary-based search methods showed comparable performance with faster convergence rate such as: Tabu search [233] and population-based incremental learning [133]. In general, the computational costs of these techniques are still rather large.
- **Sequential search:** The following strategies can be applied to find the preferred ensemble subset;

- *Forward Search*: Starts with an empty set followed by sequential addition, one classifier at a time, and conditioned by enhancing specific metrics; otherwise, the search stops. There is no guarantee to find the optimal subensemble.
- *Backward Search*: The algorithm starts with the whole ensemble size, and one classifier is removed iteratively without affecting the evaluation metric. The complexity of this algorithm is the same as in forward search.

- **Clustering-based pruning:** Clustering techniques, non-supervised methods, are used to group similar classifiers together. The formed clusters are separately pruned to select a high diversity classifiers' subset. This methodology could suffer from cluster instability as mentioned in [234], whereas an alternative solution is in using hybrid clustering techniques, consensus clustering, to aggregate different clustering results [26, 235].
- **Ranking-based pruning** (Ordering-based pruning): Firstly, the classifiers are ranked based on an evaluation criteria, then the top set in the list is selected. The computational costs of these methods are less in comparison with the aforementioned solutions. In addition, these methods are more efficient to work with parallel ensembles where individual classifiers can be built independently.

In this chapter, a guided search-based pruning method is proposed to consider both the individual's accuracy and the ensemble diversity to improve the overall accuracy, in the light of reduced data in advance. The remainder of this chapter is organized as follows: In Section 5.1, the motivations and the contributions are presented. While in Section 5.2, the concept of ensemble pruning via reordering the classifiers is explained. The proposed framework is to be presented in detail in Section 5.3. The experimental results are presented in Sections 5.4. Finally, Section 5.5 will be dedicated for the conclusions and future work.

## 5.1 Motivations and Contributions

The main contributions of this chapter can be highlighted in the following points:

1. Forming small-size ensembles with less complex individual models.
2. Proposing a guided search-based ensemble pruning method.
3. Analyzing how the proposed method can be an alternative to large-size ensembles.
4. Getting out more accuracy from the reduced data and comparing it with state-of-art ensembles (Random Forest [4], SAMME [190], and XGBoost [193]) that are trained from nonreduced data.

## 5.2 Ordering-based Pruning

As mentioned before, those strategies are more promising to select an efficient subensemble in less computational time. In bagging, the base classifiers are generated independently based on different bootstrap samples from the training data. The prediction accuracy of the ensemble is positively correlated with the number of aggregated models. Notwithstanding, the accuracy of the ensemble levels off after some point. After this point the inclusion of further models becomes useless. As shown in [23, 35], the general accuracy (error) can be maximized (minimized) by changing the order in which the classifiers are aggregated. The authors proved that the first 20% from the modified ordered bagging ensemble was sufficient to speed up the classification decision, to save memory storage, and to get an improved composite prediction. The core component in the ordering strategies is the heuristic metric used to give the ordering process. That metric exploits the aggregation relationship between the classifiers based on maximizing (minimizing) specific measure as in greedy search [31, 35, 37] or rank the significance of each base classifier in one batch as in [24, 32, 36]. These ordering metrics require a selection or pruning set composed of labeled samples, $D_{pr}$, to validate and guide the ordering process. For that, the pruning set can be an independent part, not used for training, or can be sampled from the original training. After that, the predictions of the selected classifiers are aggregated by unweighted voting as:

$$\hat{\Psi}(\mathbf{x}_i) = \underset{y_i \in M}{\arg\max} \sum_{k=1}^{\hat{T}} [\Psi_k(\mathbf{x}_i) = y_i] \tag{5.1}$$

where [ ] denotes Iverson's bracket and $\hat{T}$ represents the subensemble size. As a promising strategies, we concentrate on common ordering-based pruning methods from the literature review with the discovered challenges.

### 5.2.1 Diversity Contribution of Individuals

Ensemble Pruning via Individual Contributions (EPIC) is introduced in [32]. The appropriate handling of the trade-off between diversity and the accuracy of the ensemble members is the key to gain efficient ensemble models [123, 162, 236]. Increasing the accuracy of individual models leads to producing similar classifiers, in their decisions, with less integration in-between [32] due to lack of diversity. Two research assumptions have been mentioned in [32] with deep analysis of the first one: (1) When two ensembles have individual models with the same accuracy, the more diverse ensemble should perform better. (2) When two ensembles are similarly diverse, the one which has more accurate individuals should perform better. These two assumptions represent the balance between ensemble diversity and the individual's accuracy, respectively. The classifier's rank shouldn't be assigned based on its accuracy alone, but further on how it is more diverse with the majority voting of the ensemble (*different from peer members*).

During the simple majority voting over a specific sample $\mathbf{x}_i$, the classifier's prediction can be categorized into one of four cases (a) Correct Prediction, but ensemble prediction is incorrect (b) Correct Prediction and ensemble prediction is also correct (c) Incorrect Prediction, but ensemble prediction is correct (d) Incorrect Prediction and ensemble prediction is incorrect. The classifiers belong to case (a) are more critical to change the ensemble decision by giving them higher priority to be selected, hence reducing the effect of negative voting. While classifiers in the category (c) receive a lower rank, even their presence in the ensemble is less harmful.

EPIC [32] measures the diversity contribution of each individual via Equation (5.2), where $\nu^{(i)}_{max}, \nu^{(i)}_{sec}$ are the number of votes for the top two classes over sample $\mathbf{x}_i$ respectively. While $\nu^{(i)}_{\Psi_k(\mathbf{x}_i)}$ denotes the number of classifiers that agree with the prediction $\Psi_k(\mathbf{x}_i)$ (including itself), and $\nu^{(i)}_{correct}$ denotes number of votes of the correct prediction over sample $\mathbf{x}_i$.

$$IC_k = \sum_{i=1}^{N} \left( \alpha_{ki} \left(2\nu_{max}^{(i)} - \nu_{\Psi_k(\mathbf{x}_i)}^{(i)}\right) + \beta_{ki}\, \nu_{sec}^{(i)} + \theta_{ki} \left(\nu_{correct}^{(i)} - \nu_{\Psi_k(\mathbf{x}_i)}^{(i)} - \nu_{max}^{(i)}\right) \right) \quad (5.2)$$

*Where:*

$$\alpha_{ki} = \begin{cases} 1 & \text{if } \Psi_k(\mathbf{x}_i) = y_i \ \wedge \Psi_k(\mathbf{x}_i) \text{ is in the minority voting;} \\ 0 & \text{otherwise.} \end{cases}$$

$$\beta_{ki} = \begin{cases} 1 & \text{if } \Psi_k(\mathbf{x}_i) = y_i \ \wedge \Psi_k(\mathbf{x}_i) \text{ is in the majority voting;} \\ 0 & \text{otherwise.} \end{cases}$$

$$\theta_{ki} = \begin{cases} 1 & \text{if } \Psi_k(\mathbf{x}_i) \neq y_i; \\ 0 & \text{otherwise.} \end{cases}$$

In [32], the authors concluded that classifiers that have more votes in the minority groups bring more diversity contributions to the ensemble and contain more useful knowledge for constructing subensembles. Those classifiers are assigned a higher positive degree of contribution for their correct prediction and less negative degree of contribution for their incorrect prediction.

### 5.2.2 Unsupervised Ensemble Margin

Unsupervised Margin based Ensemble Pruning (UMEP) has been proposed in [24] with the focus on classifier properties to classify the hard patterns correctly. The innovation of this metric is based on measuring the margin of $\mathbf{x}_i$. The larger the margin of $\mathbf{x}_i$ the more certain its classification is. As in boosting [5], the idea of this method is to focus on low margin instances. The absolute margin of $\mathbf{x}_i$ can be measured from ensemble decisions as:

$$\text{Margin}(\mathbf{x}_i) = (\nu_{max} - \nu_{sec}) \Big/ \sum_{i=1}^{M} (\nu_{c_i}) \quad (5.3)$$

This measure considers only the difference between the votes of the top two classes $(\nu_{max}, \nu_{sec})$ over the sample $\mathbf{x}_i$. For that, it is an unsupervised measure that does not require the true class label. Here the margin takes a value in the interval

$[0, 1]$. The set of samples, $\mathbf{x}_i \in D_{pr}$, that are classified correctly by each classifier will be considered for calculating it's margin-based information quantity as:

$$\Psi_k(D_{pr}) = \frac{-1}{N}\sum_{i=1}^{N} \log\left(\text{Margin}(\mathbf{x}_i)\right) \quad \Big| \Psi_k(\mathbf{x}_i) = y_i \tag{5.4}$$

Then, the classifiers are ranked based on descending the measured values from Equation (5.4). The more hard samples which are predicted correctly by the classifier, the more rank it receives to be included in the subensemble.

### 5.2.3 Margin & Diversity

Margin and Diversity-based Ensemble Pruning (MDEP) [36] considers two aspects to better reorder the set of classifiers: (1) focusing on examples with small absolute margin and (2) focusing on classifiers with large diversity contribution to the ensemble. The MDEP measures the rank of each classifier via Equation (5.5) $\forall \mathbf{x}_i \in D_{pr} \big| \Psi_k(\mathbf{x}_i) = y_i$.

$$\text{MDEP}(\Psi_k) = \sum_{\mathbf{x}_i \in D_{pr}} \Big[ \alpha f_m(\mathbf{x}_i) + (1-\alpha) f_d(\Psi_k, \mathbf{x}_i) \Big] \tag{5.5}$$

Where $\alpha \in [0, 1]$ represents the balance of importance between the margin of examples and the ensemble diversity. $f_m(\mathbf{x}_i)$ and $f_d(\Psi_k, \mathbf{x}_i)$ are the $\log$ functions of $\mathbf{x}_i$'s margin and $\Psi_k$'s diversity contribution on $\mathbf{x}_i$, are calculated via Equation (5.6) and Equation (5.7), respectively. Where $\bar{y_i} \neq y_i$ is the class that receives the maximum number of votes on $\mathbf{x}_i$. The challenge of the MDEP metric is the dependence on the predefined value of $\alpha$ that controls the trade-off between focusing on classifiers that correctly predict hard samples or focusing on classifiers that increase ensemble's diversity.

$$f_m(\mathbf{x}_i) = \log\left(\left|\frac{\nu_{y_i}^{(i)} - \nu_{\bar{y}_i}^{(i)}}{M}\right|\right) \tag{5.6}$$

$$f_d(\Psi_k, \mathbf{x}_i) = \log\left(\frac{\nu_{y_i}^{(i)}}{M}\right) \tag{5.7}$$

Determining the value of $\alpha$ is not a trivial task and should be analyzed for each dataset separately. It can be tuned by searching for the best value in the range [0,1] through the cross-validation process, which is a time-consuming procedure. Increasing the value of $\alpha$ directs the search process to select a subensemble that better classifies low margin samples. While reducing the value of $\alpha$ enforces the search process to select a subensemble with high diversity. We praise with the effect of $\alpha$ that has been discussed in [36], in an attempt to capture this conflict, but restricted with parameter tuning.

Our contribution related to this part is to propose a guided search pruning method that handles the bias in the subensemble search strategy. We prove how our proposed method is stable with different datasets without assuming any parameters in advance. Furthermore, this chapter discusses the importance of the instance selection method to form simpler ensembles. Moreover, in the literature review; the performance of the pruned ensemble often compared with the unpruned one. Here we extend this level of comparisons to include several baseline and efficient ensemble models as: Random Forest [4], SAMME [190], and XGBoost [193]. The next section discusses the proposed method to introduce a dual reduction to the formed ensemble.

## 5.3 Proposed Framework

The proposed framework is graphically presented in Figure 5.1. The training fold is reduced by AllKNN [220] as an instance selection method to select representative samples. The selected data by AllKNN will be the source to build a group of heterogeneous classifiers. A pruning set is formed using stratified sampling with 70% from the train. The pruning set will be the source to rank the individual classifiers during ensemble selection. Using this methodology, the following state-of-art ensembles will be trained for comparison purposes:

- RFSM: Random Forest [4] is learned from the selected data.
- UNP: The proposed heterogeneous classifiers, Section (4.3.2), without pruning learned from the selected data.
- RFCOM: Random Forest is learned from the whole training data.

- SAMME: (Stagewise Additive Modeling using a Multi-class Exponential loss function [190]), which is trained from whole training data and extends the AdaBoost algorithm to the multiclass case.
- XGBoost: eXtreme Gradient Boosting decision tree [193] which is trained from the whole training data.

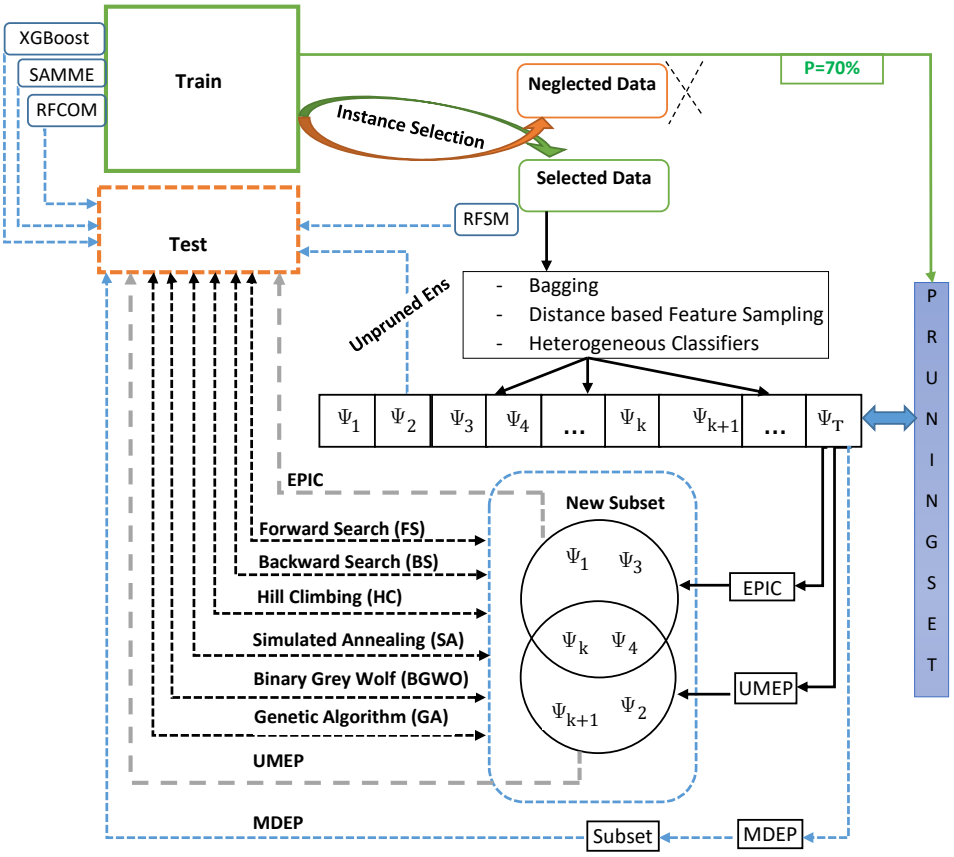


**Figure 5.1:** The proposed ensemble selection in the presence of selected samples.

The two phases for cleaning noisy border samples, Section 4.3.1, and building heterogeneous MCS, Section 4.3.2, will not be changed. While we propose a guided search to prune the proposed ensemble. Therefore, we benefit from IS techniques and ensemble selection, rather than training large-size ensembles from large-size training data.

**The Proposed Guided Search for Ensemble Pruning**: Having a large-size ensemble will limit the search capability of the well-known pruning metrics. As each pruning metric puts pressure to select a subset of classifiers with a specific

property. For example, EPIC [32] concentrates on the classifier's diversity with the peer members. While UMEP [24] focuses more to select classifiers that behave well with low-margin samples. Regarding that, and with the large pool size of classifiers, the performance of the current pruning methods could vanish. Experimentally, Fig. 5.2 shows the generalized accuracy of the identified subensemble via different pruning methods, UMEP and EPIC, from different ensemble sizes. From Fig. 5.2, we confirm that each metric is more important to a particular dataset. We conclude that paying no attention to other heuristic information affects the search process badly, and returns a subensemble with a limited accuracy.

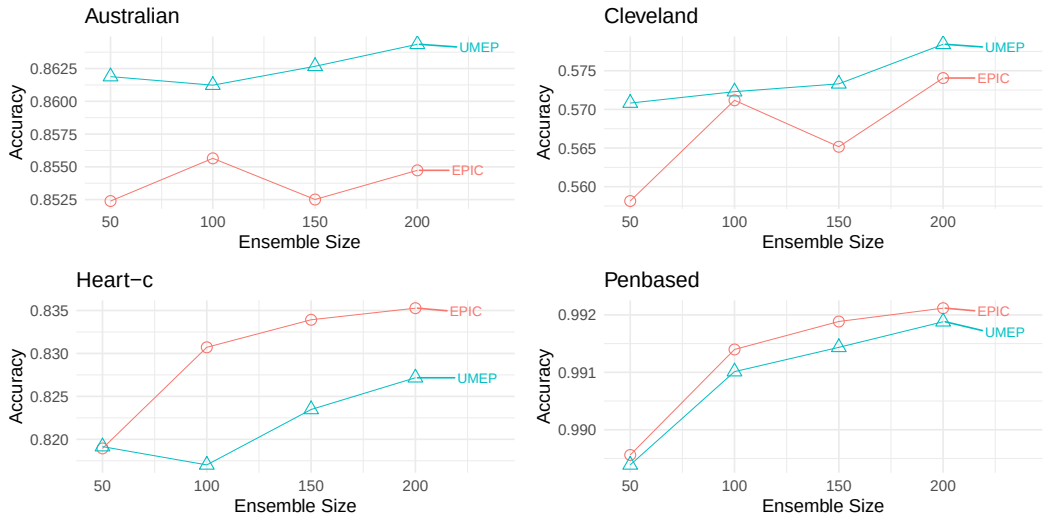


**Figure 5.2:** The conflict between both EPIC and UMEP, to be handled in this chapter.

Inspired by the above analysis, we can narrow the original search space by directing the search to promising areas. This is what EPIC and UMEP return, each metric recommends a subspace according to their heuristic measures. Preferring one metric over the other will blinds a subspace that could contain crucial information. The two ordering-based pruning techniques, EPIC [32] and UMEP [24], will be applied to determine the promising subspaces in advance. After that, the search process can be started. The details are as follows:

- Each metric will rank the set of classifiers differently, but at least there will be common classifiers in-between.
- After ranking, a predetermined percentage, $P$, will be used to select a subset

of classifiers.

- A narrow space carrying the properties of both metrics will be formed via merging their output subsets, *as in Fig. 5.1*.
- The new subspace will be searched via Forward Search (FS) [1], Backward Search (BS)[1], Hill Climbing (HC) [237], Simulated Annealing (SA) [237], Binary Grey Wolf Optimizer (BGWO) [238], and Binary Genetic Algorithm (GA)[2].

Via guided search, we try to alleviate the randomness of the search process that could increase with large-size ensembles. To the best of our knowledge, this work is the first to handle the conflict between EPIC and UMEP. Moreover, it can be categorized as ensembling the pruning metrics themselves to get better performance than each of them.

## 5.4 Experimental Results

The experiments are dedicated to achieving objective 2, to increase the efficiency of MCS, and to go beyond what can be achieved from ensemble pruning methods. The two main questions to be answered are:

- $\boldsymbol{Q_3}$. What is the effect of combining multiple pruning metrics together?
- $\boldsymbol{Q_4}$. What is the effect of downsizing data and downsizing the number of classifiers simultaneously?

### 5.4.1 Setup of Experiments

During setup and validation, all datasets are preprocessed by unifying the scale of the features via normalization. For each dataset, 20 repetitions with 5 fold cross-validation procedure are considered. Thus, a total of 100 runs per dataset. The accuracy metric is calculated by the majority voting of all ensemble members. In addition, MDEP depends on an internal parameter $\alpha$; three values for MDEP with different $\alpha \in \{0.1, 0.5, 0.9\}$ are considered, and the best-optimized alpha according to the in train-validation is used to report the test for each dataset separately.

[1]Package Fselector:https://cran.r-project.org/web/packages/FSelector
[2]Package genalg:https://cran.r-project.org/web/packages/genalg

A total number of 25 datasets captured from OpenML[1] and KEEL [2] are used in this work in order to provide experimentation. The characteristics of data can be found in Table 5.1, where #S, #F, #C, and R represent the number of samples, the number of features, the number of classes, and the ratio between the smallest and the largest class for each dataset respectively.

**Table 5.1:** Characteristics of the selected datasets for experimentation, *sorted by samples and classes*.

| DataSet | #S | #F | #C | R | DataSet | #S | #F | #C | R |
|---|---|---|---|---|---|---|---|---|---|
| Heart-statlog | 270 | 13 | 2 | 0.8 | Waveform | 4 999 | 21 | 3 | 0.972 |
| Heart-c | 303 | 13 | 2 | 0.836 | Cleveland | 297 | 13 | 5 | 0.081 |
| Ionosphere | 351 | 33 | 2 | 0.56 | Dermatology | 358 | 34 | 6 | 0.18 |
| Sa-heart | 462 | 9 | 2 | 0.529 | Satimage | 6 435 | 36 | 6 | 0.408 |
| Wdbc | 569 | 30 | 2 | 0.594 | Segment | 2 310 | 18 | 7 | 1.0 |
| Breast-w | 699 | 9 | 2 | 0.526 | Mfeat-fourier | 2 000 | 76 | 10 | 1.0 |
| Australian | 690 | 14 | 2 | 0.802 | Mfeat-karh | 2 000 | 64 | 10 | 1.0 |
| Blood-transfusion | 748 | 4 | 2 | 0.312 | Mfeat-zernike | 2 000 | 47 | 10 | 1.0 |
| Mammographic | 830 | 5 | 2 | 0.944 | Led24 | 3 200 | 24 | 10 | 0.878 |
| Diabetic Retinopathy | 1 151 | 19 | 2 | 0.884 | Optdigits | 5 620 | 62 | 10 | 0.969 |
| Abalone | 4 177 | 8 | 2 | 0.464 | Penbased | 10 992 | 16 | 10 | 0.922 |
| Ringnorm | 7 400 | 20 | 2 | 0.981 | Texture | 5 500 | 40 | 11 | 1.0 |
| Twonorm | 7 400 | 20 | 2 | 0.998 | | | | | |

### 5.4.2 Analysis of Guided Search

In this section, we concentrate on the performance of the proposed ensemble selection strategy. Table 5.2 shows the accuracy of subensemble related to percentage $P$ from the ensemble size $T$. Each row in Table 5.2 represents the average of 5,000 records of experiments (dataset=25 $\cdot$ runs=100 $\cdot$ $P$=2). From this table, the highest accuracy can be obtained by XGBoost, and BS which merges EPIC and UMEP subspaces to exploit efficient classifiers. Moreover, the general accuracy is elevated as more classifiers are selected ($P$=20%). It is interesting to observe that for $T = 200$, SAMME, RFCOM, RFSM, and UNP realize an accuracy lower than 85.99% that is reported by BS when it uses 12 classifiers. Regarding that, the proposed guided search from a low-size ensemble, $T = 50$, cancels the necessity to

[1]Machine Learning Repository: https://www.openml.org
[2]KEEL Repository: http://www.keel.es/

initialize large-size ensembles. This could save huge training costs without losing the accuracy.

The average subensemble size related to the predetermined $P$ value is shown in Table 5.3. Where FS guarantees to return a small-size subensemble, but unfortunately not accurate as shown in Table 5.2. Furthermore, we can notice the performance of HC, SA, and BGWO according to both the accuracy and the subensemble size. The statistical analysis of the proposed pruning strategies shows that BS significantly outperforms the other five search methods (FS, HC, SA, BGWO, and GA) to identify a high accurate subensemble. Therefore, it is interesting to compare BS with EPIC, UMEP, and MDEP.

**Table 5.2:** The average accuracy of subensemble related to selection percentage ($P$) of EPIC and UMEP, results over all datasets.

| $T$ | XGBoost | SAMME | RFCOM | RFSM | UNP | EPIC | UMEP | MDEP | FS | BS | HC | SA | BGWO | GA | $P$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 50 | **86.60** | 84.46 | 85.67 | 85.10 | 85.00 | 84.41 | 85.32 | 84.49 | 84.17 | **85.55** | 85.34 | 85.29 | 85.40 | 83.38 | 10% |
| | | | | | | 85.34 | 85.78 | 85.34 | 84.30 | **85.99** | 85.74 | 85.72 | 85.78 | 84.88 | 20% |
| 100 | **86.60** | 84.88 | 85.86 | 85.23 | 85.15 | 85.14 | 85.75 | 85.40 | 84.50 | **86.11** | 85.85 | 85.87 | 85.85 | 85.17 | 10% |
| | | | | | | 85.94 | 86.12 | 85.97 | 84.61 | **86.33** | 86.14 | 86.14 | 86.15 | 85.65 | 20% |
| 150 | **86.53** | 84.99 | 85.95 | 85.28 | 85.23 | 85.46 | 86.13 | 85.81 | 84.72 | **86.28** | 86.11 | 86.12 | 86.09 | 85.47 | 10% |
| | | | | | | 86.06 | 86.34 | 86.16 | 84.76 | **86.47** | 86.33 | 86.31 | 86.31 | 85.83 | 20% |
| 200 | **86.57** | 85.06 | 85.96 | 85.27 | 85.26 | 85.57 | 86.29 | 85.93 | 84.68 | **86.41** | 86.28 | 86.24 | 86.18 | 85.71 | 10% |
| | | | | | | 86.18 | 86.46 | 86.30 | 84.75 | **86.56** | 86.46 | 86.42 | 86.33 | 85.95 | 20% |

**Table 5.3:** The average size of subensemble related to selection percentage ($P$) of EPIC and UMEP, results over all datasets.

| XGBoost | SAMME | RFCOM | RFSM | UNP | EPIC | UMEP | MDEP | FS | BS | HC | SA | BGWO | GA | $P$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 50 | | | | | 5 | | | **1.83** | 6.62 | 5.06 | 5.03 | 5.22 | 2.74 | 10% |
| | | | | | 10 | | | **2.06** | 12.37 | 8.00 | 7.89 | 7.89 | 4.77 | 20% |
| 100 | | | | | 10 | | | **2.13** | 13.69 | 8.55 | 8.48 | 8.41 | 5.03 | 10% |
| | | | | | 20 | | | **2.31** | 25.90 | 14.30 | 14.42 | 13.19 | 6.74 | 20% |
| 150 | | | | | 15 | | | **2.32** | 20.99 | 12.01 | 12.06 | 11.35 | 6.13 | 10% |
| | | | | | 30 | | | **2.50** | 39.69 | 20.94 | 20.84 | 18.54 | 7.95 | 20% |
| 200 | | | | | 20 | | | **2.41** | 28.54 | 15.56 | 15.59 | 12.49 | 6.86 | 10% |
| | | | | | 40 | | | **2.55** | 53.55 | 27.54 | 27.65 | 19.37 | 8.98 | 20% |

**To answer $Q_3$**, Table 5.4 illustrates the average accuracy and standard deviation via BS, EPIC, UMEP, and MDEP for 100 runs per dataset. As shown, BS obtains the best average rankings by Friedman Test [239], *AR-Friedman*, to select an effective subensemble. EPIC-10%, UMEP-10%, and MDEP-10% return a subensemble with 20 classifiers, while BS-10 returns a large-size subensemble, as demonstrated

**Table 5.4:** Classification accuracy of BS, EPIC, UMEP, and MDEP for selecting subensemble from $T = 200$ model, the best value is in bold.

| # | Dataset | BS-10 | EPIC-10% | UMEP-10% | MDEP-10% | BS-20 | EPIC-20% | UMEP-20% | MDEP-20% |
|---|---|---|---|---|---|---|---|---|---|
| $D_1$ | Abalone | 69.88 ± 1.21 | 66.73 ± 2.39 | 69.83 ± 1.04 | 69.78 ± 1.12 | **70.04 ± 1.14** | 68.11 ± 1.80 | 69.85 ± 1.00 | 69.96 ± 1.12 |
| $D_2$ | Australian | 86.58 ± 2.47 | 84.60 ± 3.17 | **86.59 ± 2.65** | 86.37 ± 2.76 | 86.50 ± 2.60 | 85.47 ± 2.62 | 86.43 ± 2.65 | 86.25 ± 2.80 |
| $D_3$ | Blood-transfusion | 77.19 ± 2.30 | 73.88 ± 4.73 | 77.00 ± 2.40 | 76.51 ± 2.61 | **77.64 ± 2.22** | 76.22 ± 3.65 | 77.17 ± 2.08 | 77.22 ± 2.08 |
| $D_4$ | breast-w | 96.99 ± 1.20 | 96.63 ± 1.39 | 96.99 ± 1.28 | 97.00 ± 1.40 | 97.12 ± 1.23 | 96.87 ± 1.30 | **97.13 ± 1.25** | **97.13 ± 1.31** |
| $D_5$ | Cleveland | 57.25 ± 4.27 | 54.98 ± 5.84 | 57.52 ± 3.41 | 57.06 ± 3.43 | **57.96 ± 3.66** | 57.41 ± 4.27 | 57.84 ± 3.12 | 57.56 ± 3.10 |
| $D_6$ | Dermatology | **97.63 ± 1.69** | 97.40 ± 1.79 | 97.50 ± 1.80 | 97.49 ± 1.74 | 97.53 ± 1.72 | 97.32 ± 1.89 | 97.56 ± 1.62 | **97.63 ± 1.65** |
| $D_7$ | Diabetic | 70.77 ± 2.99 | 69.71 ± 2.59 | **70.89 ± 3.28** | 70.71 ± 2.96 | 69.81 ± 3.10 | 69.63 ± 2.75 | 70.26 ± 3.20 | 70.49 ± 3.00 |
| $D_8$ | Heart-c | 82.83 ± 4.55 | 82.75 ± 4.38 | 82.25 ± 4.76 | 82.06 ± 4.68 | **83.61 ± 4.26** | 83.53 ± 4.25 | 82.72 ± 4.58 | 83.12 ± 4.33 |
| $D_9$ | Heart-statlog | 83.02 ± 4.29 | 81.13 ± 4.14 | 82.94 ± 4.21 | 83.24 ± 4.29 | 83.46 ± 4.27 | 82.57 ± 4.45 | 83.11 ± 4.07 | **83.70 ± 4.26** |
| $D_{10}$ | Ionosphere | **92.54 ± 2.85** | 92.07 ± 2.83 | 91.67 ± 3.09 | 92.14 ± 2.59 | 92.34 ± 2.75 | 92.41 ± 2.61 | 91.93 ± 2.89 | 92.49 ± 2.74 |
| $D_{11}$ | Led24 | 71.62 ± 1.54 | 71.16 ± 1.74 | 71.58 ± 1.51 | 65.89 ± 4.76 | **71.89 ± 1.45** | 71.75 ± 1.46 | 71.78 ± 1.45 | 68.66 ± 3.79 |
| $D_{12}$ | Mammographic | 83.76 ± 2.51 | 83.05 ± 2.55 | 83.72 ± 2.54 | 83.56 ± 2.47 | 83.89 ± 2.36 | 83.37 ± 2.64 | 84.00 ± 2.32 | **84.05 ± 2.27** |
| $D_{13}$ | Mfeat-fourier | **81.41 ± 1.55** | 80.34 ± 1.76 | 80.97 ± 1.62 | 80.41 ± 1.67 | **81.41 ± 1.62** | 80.89 ± 1.65 | 81.34 ± 1.56 | 81.12 ± 1.63 |
| $D_{14}$ | Mfeat-karh | 96.08 ± 0.88 | 95.67 ± 0.99 | 96.05 ± 0.86 | 95.99 ± 1.13 | **96.31 ± 0.89** | 96.19 ± 0.86 | 96.27 ± 0.85 | 96.28 ± 0.96 |
| $D_{15}$ | Mfeat-zernike | 82.05 ± 1.47 | 81.94 ± 1.58 | 82.08 ± 1.49 | 81.49 ± 1.58 | 82.38 ± 1.39 | 82.19 ± 1.52 | **82.48 ± 1.44** | 82.20 ± 1.41 |
| $D_{16}$ | Optdigits | 98.53 ± 0.32 | 98.50 ± 0.34 | 98.54 ± 0.34 | 98.53 ± 0.34 | **98.61 ± 0.32** | **98.61 ± 0.31** | 98.60 ± 0.31 | 98.59 ± 0.30 |
| $D_{17}$ | Penbased | 99.15 ± 0.19 | 99.15 ± 0.18 | 99.10 ± 0.19 | 99.03 ± 0.71 | 99.20 ± 0.16 | **99.21 ± 0.16** | 99.19 ± 0.17 | 99.10 ± 0.69 |
| $D_{18}$ | Ringnorm | 96.36 ± 0.54 | 95.94 ± 0.55 | 95.92 ± 0.56 | 96.00 ± 0.51 | **96.50 ± 0.48** | 96.33 ± 0.51 | 96.33 ± 0.51 | 96.37 ± 0.49 |
| $D_{19}$ | Sa-heart | 70.49 ± 4.17 | 69.44 ± 4.33 | 70.64 ± 3.90 | 70.47 ± 3.53 | 70.98 ± 4.06 | 70.50 ± 4.30 | 70.98 ± 3.86 | **71.25 ± 3.65** |
| $D_{20}$ | Satimage | 90.12 ± 0.78 | 90.04 ± 0.80 | 90.00 ± 0.81 | 89.70 ± 1.46 | 90.16 ± 0.72 | **90.17 ± 0.75** | 90.05 ± 0.77 | 89.81 ± 1.39 |
| $D_{21}$ | Segment | 96.56 ± 0.74 | 96.34 ± 0.74 | 96.41 ± 0.69 | 96.20 ± 1.02 | **96.61 ± 0.77** | **96.61 ± 0.77** | 96.54 ± 0.74 | 96.51 ± 0.87 |
| $D_{22}$ | Texture | 99.62 ± 0.20 | 99.61 ± 0.20 | 99.60 ± 0.19 | 99.62 ± 0.18 | **99.63 ± 0.18** | **99.63 ± 0.19** | **99.63 ± 0.19** | **99.63 ± 0.16** |
| $D_{23}$ | Twonorm | 97.31 ± 0.40 | 97.04 ± 0.42 | 97.14 ± 0.41 | 97.09 ± 0.45 | **97.55 ± 0.35** | 97.38 ± 0.39 | 97.46 ± 0.38 | 97.47 ± 0.42 |
| $D_{24}$ | Waveform | 85.04 ± 0.94 | 83.82 ± 0.98 | 84.95 ± 1.0 | 84.82 ± 1.13 | **85.42 ± 0.93** | 84.59 ± 1.00 | 85.41 ± 0.95 | 85.34 ± 1.07 |
| $D_{25}$ | Wdbc | 97.42 ± 1.66 | 97.34 ± 1.68 | 97.35 ± 1.68 | 97.16 ± 1.58 | 97.49 ± 1.56 | **97.52 ± 1.46** | 97.48 ± 1.63 | 97.34 ± 1.58 |
| **AR-Friedman** | | **3.92** | 7.16 | 5.32 | 6.36 | **2.06** | 4.56 | 3.32 | 3.3 |

in Table. 5.3. While the size and the accuracy of the subensemble are important, Table 5.5 shows the pairwise statistical analysis [227] for calibration between them. From Table 5.5, BS-10 is comparable with EPIC-20%, UMEP-20%, and MDEP-20% in terms of accuracy, however, it significantly outperforms them regarding the subensemble size. Furthermore, the obtained subensemble's accuracy by BS-20 significantly outperforms all the pruning methods, this is marked by (▲). The user has more flexibility to prefer a high accurate subensemble (▲), the lower part of the table, or to prefer a small-size subensemble (•) as in the upper part.

**Table 5.5:** Summary of the Wilcoxon test. Shape denotes the measure used, accuracy (▲ △) and size (•○). The filled shape (▲ •) represents if the method in the row outperforms the one in the column or vice versa for (△ ○). Upper diagonal of level significance $\alpha = 0.9$, lower diagonal level of significance $\alpha = 0.95$.

| | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) |
|---|---|---|---|---|---|---|---|---|
| BS-10 (1) | - - | ▲ ○ | ▲ ○ | ▲ ○ | △ • | • | • | • |
| EPIC-10% (2) | △ • | - - | △ | △ | △ • | △ • | △ • | △ • |
| UMEP-10% (3) | △ • | ▲ | - - | ▲ | △ • | • | △ • | △ • |
| MDEP-10% (4) | △ • | | △ | - - | △ • | • | △ • | △ • |
| BS-20 (5) | ▲ ○ | ▲ ○ | ▲ ○ | ▲ ○ | - - | ▲ ○ | ▲ ○ | ▲ ○ |
| EPIC-20% (6) | ○ | ▲ ○ | ○ | ○ | △ • | - - | △ | △ |
| UMEP-20% (7) | ○ | ▲ ○ | ▲ ○ | ▲ ○ | △ • | ▲ | - - | |
| MDEP-20% (8) | ○ | ▲ ○ | ▲ ○ | ▲○ | • | ▲ | | - - |

### 5.4.3 Classification Performance of the Proposed Method

**To answer $Q_4$**, we focus on the effectiveness of the proposed method to learn from representative samples and to reduce the bias of the pruning method via integrating both EPIC and UMEP. Both $T$, $P$ will be 200 and 20% respectively to realize higher accuracy as demonstrated in Table 5.2. The results are shown in Table 5.6 to report the performance of fourteen ensemble/subensemble models:

- RFCOM, SAMME, and XGBoost, *large-size ensembles which are trained from non-reduced datasets*. They are called Non-Reduced Data Non-Selected Ensembles (NRD-NSE).
- RFSM and UNP, *large-size ensembles which are trained from reduced datasets*. They are called Reduced Data Non-Selected Ensembles (RD-NSE).

**Table 5.6:** Average accuracy and standard deviation of NRD-NSE, RD-NSE, and RD-SE from ensemble size $T = 200$. The best value is in bold, while the second best is underlined.

| | | AllKNN | NRD-NSE | | | RD-NSE | | RD-SE | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # | Dataset | Red_Rate | XGBoost | SAMME | RFCOM | RFSM | UNP | EPIC | UMEP | MDEP | FS | BS | HC | SA | BGWO | GA |
| $D_1$ | Abalone | 0.47 | 69.05± 1.29 | 69.67 ± 1.21 | 69.11 ± 1.27 | 69.99 ± 1.07 | 69.52 ± 0.64 | 68.11 ± 1.80 | 69.85 ± 1.00 | 69.96 ± 1.12 | 68.60 ± 1.46 | **70.04 ± 1.14** | 69.90 ± 1.12 | 70.00 ± 1.19 | 69.81 ± 1.10 | 69.58 ± 1.12 |
| $D_2$ | Australian | 0.26 | **87.22 ± 2.60** | 86.50 ± 2.36 | 87.12 ± 2.56 | 86.33 ± 2.58 | 86.05 ± 2.46 | 85.47 ± 2.62 | 86.43 ± 2.65 | 86.25 ± 2.80 | 86.17 ± 2.53 | 86.50 ± 2.60 | 86.64 ± 2.42 | 86.60 ± 2.45 | 86.44 ± 2.51 | 86.10 ± 2.68 |
| $D_3$ | Blood-transfusion | 0.49 | 77.55 ± 2.25 | 74.50 ± 2.60 | 75.51 ± 2.43 | 77.22 ± 2.60 | 76.90 ± 1.45 | 76.22 ± 3.65 | 77.17 ± 2.08 | 77.22 ± 2.08 | 77.23 ± 2.35 | 77.64 ± 2.22 | **77.79 ± 2.16** | 77.75 ± 2.19 | 77.51 ± 1.97 | 77.31 ± 2.36 |
| $D_4$ | Breast-w | 0.06 | 96.55 ± 1.38 | 96.88 ± 1.19 | 97.09 ± 1.22 | 96.61 ± 1.39 | **97.42 ± 1.17** | 96.87 ± 1.30 | 97.13 ± 1.25 | 97.13 ± 1.31 | 96.11 ± 1.55 | 97.12 ± 1.23 | 97.07 ± 1.28 | 97.09 ± 1.24 | 97.02 ± 1.17 | 96.78 ± 1.35 |
| $D_5$ | Cleveland | 0.55 | 56.15 ± 4.30 | 53.02 ± 5.11 | 57.14 ± 3.75 | 56.49 ± 2.79 | 56.50 ± 2.88 | 57.41 ± 4.27 | 57.84 ± 3.12 | 57.56 ± 3.10 | 56.39 ± 3.82 | **57.96 ± 3.66** | 57.74 ± 3.60 | 57.51 ± 3.72 | 57.86 ± 3.30 | 57.20 ± 3.85 |
| $D_6$ | Dermatology | 0.07 | **97.96 ± 1.57** | 96.69 ± 1.77 | 97.45 ± 1.55 | 97.44 ± 1.46 | 97.46 ± 1.65 | 97.32 ± 1.89 | 97.56 ± 1.62 | 97.63 ± 1.65 | 96.00 ± 2.22 | 97.53 ± 1.72 | 97.53 ± 1.61 | 97.61 ± 1.76 | 97.26 ± 1.80 | 96.64 ± 1.95 |
| $D_7$ | Diabetic | 0.55 | 68.99 ± 2.55 | 69.14 ± 2.76 | 67.86 ± 2.82 | 65.83 ± 2.82 | 66.38 ± 2.94 | 69.63 ± 2.75 | 70.26 ± 3.20 | 70.49 ± 3.00 | **70.90 ± 3.15** | 69.81 ± 3.10 | 69.97 ± 3.13 | 69.80 ± 3.02 | 70.08 ± 3.07 | 70.19 ± 3.34 |
| $D_8$ | Heart-c | 0.32 | **83.85 ± 4.96** | 79.88 ± 4.75 | 83.00 ± 4.60 | 82.18 ± 4.35 | 83.22 ± 4.49 | 83.53 ± 4.25 | 82.72 ± 4.58 | 83.12 ± 4.33 | 79.91 ± 4.99 | 83.61 ± 4.26 | 83.59 ± 4.37 | 83.22 ± 4.64 | 83.39 ± 4.24 | 82.21 ± 4.62 |
| $D_9$ | Heart-statlog | 0.34 | **83.93 ± 4.40** | 79.28 ± 5.02 | 82.78 ± 4.34 | 81.30 ± 4.99 | 83.07 ± 4.71 | 82.57 ± 4.45 | 83.11 ± 4.07 | 83.70 ± 4.26 | 80.35 ± 5.14 | 83.46 ± 4.27 | 83.31 ± 4.32 | 83.33 ± 4.24 | 82.78 ± 4.42 | 82.67 ± 3.92 |
| $D_{10}$ | Ionosphere | 0.19 | 92.95 ± 3.02 | **93.70 ± 2.26** | 93.26 ± 2.53 | 91.08 ± 2.75 | 89.67 ± 2.98 | 92.41 ± 2.61 | 91.93 ± 2.89 | 92.49 ± 2.74 | 90.69 ± 3.53 | 92.34 ± 2.75 | 92.44 ± 2.94 | 92.25 ± 2.75 | 92.34 ± 2.88 | 92.21 ± 2.86 |
| $D_{11}$ | Led24 | 0.68 | **72.56 ± 1.43** | 66.40 ± 1.41 | 72.21 ± 1.40 | 71.55 ± 1.63 | 70.02 ± 1.76 | 71.75 ± 1.46 | 71.78 ± 1.45 | 68.66 ± 3.79 | 70.38 ± 2.23 | 71.89 ± 1.45 | 71.75 ± 1.58 | 71.73 ± 1.51 | 71.72 ± 1.55 | 71.29 ± 1.58 |
| $D_{12}$ | Mammographic | 0.35 | 82.56 ± 2.66 | 79.81 ± 2.36 | 81.76 ± 2.49 | 82.20 ± 2.43 | 82.06 ± 2.51 | 83.37 ± 2.64 | 84.00 ± 2.32 | **84.05 ± 2.27** | 82.96 ± 2.77 | 83.89 ± 2.36 | 83.67 ± 2.45 | 83.73 ± 2.44 | 83.57 ± 2.45 | 83.39 ± 2.39 |
| $D_{13}$ | Mfeat-fourier | 0.28 | **83.47 ± 1.86** | 82.33 ± 1.64 | 83.07 ± 1.53 | 80.45 ± 1.52 | 80.14 ± 1.50 | 80.89 ± 1.65 | 81.34 ± 1.56 | 81.12 ± 1.63 | 76.67 ± 2.46 | 81.41 ± 1.62 | 80.91 ± 1.63 | 80.96 ± 1.77 | 80.68 ± 1.57 | 79.89 ± 1.88 |
| $D_{14}$ | Mfeat-karh | 0.07 | 95.28 ± 0.99 | 96.09 ± 0.95 | 96.25 ± 0.87 | 95.46 ± 0.94 | 96.13 ± 0.88 | 96.19 ± 0.86 | 96.27 ± 0.85 | 96.28 ± 0.96 | 92.14 ± 2.14 | **96.31 ± 0.89** | 96.06 ± 0.87 | 96.06 ± 0.86 | 95.77 ± 1.00 | 95.23 ± 0.98 |
| $D_{15}$ | Mfeat-zernike | 0.26 | 81.73 ± 1.52 | 80.12 ± 1.65 | 77.87 ± 1.49 | 76.20 ± 1.45 | 80.72 ± 1.24 | 82.19 ± 1.52 | **82.48 ± 1.44** | 82.20 ± 1.41 | 78.22 ± 2.28 | 82.38 ± 1.39 | 82.20 ± 1.37 | 82.14 ± 1.35 | 81.92 ± 1.40 | 81.58 ± 1.57 |
| $D_{16}$ | Optdigits | 0.02 | 98.13 ± 0.39 | 98.14 ± 0.36 | 98.32 ± 0.35 | 98.00 ± 0.38 | 98.09 ± 0.38 | **98.61 ± 0.31** | 98.60 ± 0.31 | 98.59 ± 0.30 | 97.14 ± 0.66 | **98.61 ± 0.32** | 98.51 ± 0.34 | 98.51 ± 0.34 | 98.42 ± 0.41 | 98.10 ± 0.49 |
| $D_{17}$ | Penbased | 0.01 | 99.13 ± 0.20 | 98.73 ± 0.26 | 99.16 ± 0.19 | 99.01 ± 0.20 | 98.22 ± 0.27 | **99.21 ± 0.16** | 99.19 ± 0.17 | 99.10 ± 0.69 | 97.57 ± 0.62 | 99.20 ± 0.16 | 99.14 ± 0.18 | 99.15 ± 0.16 | 99.12 ± 0.19 | 98.93 ± 0.25 |
| $D_{18}$ | Ringnorm | 0.32 | 96.67 ± 0.41 | **97.24 ± 0.36** | 95.04 ± 0.56 | 93.09 ± 0.78 | 86.69 ± 1.15 | 96.33 ± 0.51 | 96.33 ± 0.51 | 96.37 ± 0.49 | 92.89 ± 0.63 | 96.50 ± 0.48 | 96.43 ± 0.48 | 96.32 ± 0.49 | 96.43 ± 0.49 | 95.89 ± 0.80 |
| $D_{19}$ | Sa-heart | 0.49 | **72.82 ± 3.75** | 64.57 ± 4.28 | 68.83 ± 4.14 | 70.57 ± 4.03 | 71.41 ± 3.77 | 70.50 ± 4.30 | 70.98 ± 3.86 | 71.25 ± 3.65 | 69.29 ± 4.29 | 70.98 ± 4.06 | 70.67 ± 3.91 | 70.72 ± 4.05 | 70.38 ± 4.13 | 70.07 ± 4.24 |
| $D_{20}$ | Satimage | 0.14 | **92.08 ± 0.67** | 88.72 ± 0.78 | 91.71 ± 0.71 | 89.84 ± 0.71 | 88.70 ± 0.78 | 90.17 ± 0.75 | 90.05 ± 0.77 | 89.81 ± 1.39 | 89.76 ± 0.80 | 90.16 ± 0.72 | 90.10 ± 0.74 | 90.12 ± 0.74 | 89.96 ± 0.77 | 89.67 ± 0.72 |
| $D_{21}$ | Segment | 0.06 | 98.08 ± 0.67 | **98.43 ± 0.60** | 97.89 ± 0.69 | 96.53 ± 0.83 | 95.94 ± 0.90 | 96.61 ± 0.77 | 96.54 ± 0.74 | 96.51 ± 0.87 | 94.92 ± 1.38 | 96.61 ± 0.77 | 96.57 ± 0.80 | 96.59 ± 0.84 | 96.69 ± 0.87 | 96.44 ± 0.90 |
| $D_{22}$ | Texture | 0.02 | 98.37 ± 0.38 | 98.56 ± 0.40 | 97.77 ± 0.47 | 97.14 ± 0.50 | 98.40 ± 0.35 | **99.63 ± 0.19** | **99.63 ± 0.19** | **99.63 ± 0.16** | 99.27 ± 0.39 | **99.63 ± 0.18** | 99.59 ± 0.21 | 99.60 ± 0.21 | 99.60 ± 0.21 | 99.52 ± 0.25 |
| $D_{23}$ | Twonorm | 0.08 | 97.29 ± 0.38 | 97.31 ± 0.38 | 97.23 ± 0.41 | 97.12 ± 0.39 | **97.69 ± 0.37** | 97.38 ± 0.39 | 97.46 ± 0.38 | 97.47 ± 0.42 | 94.85 ± 1.41 | 97.55 ± 0.35 | 97.38 ± 0.40 | 97.35 ± 0.41 | 97.25 ± 0.43 | 96.86 ± 0.48 |
| $D_{24}$ | Waveform | 0.32 | 85.24 ± 1.05 | 83.43 ± 1.05 | 85.39 ± 0.99 | 84.78 ± 1.06 | 84.87 ± 1.16 | 84.59 ± 1.00 | 85.41 ± 0.95 | 85.34 ± 1.07 | 83.80 ± 1.12 | **85.42 ± 0.93** | 85.16 ± 0.89 | 85.10 ± 1.04 | 84.90 ± 1.03 | 84.22 ± 1.19 |
| $D_{25}$ | Wdbc | 0.07 | 96.74 ± 1.66 | 97.29 ± 1.60 | 96.23 ± 1.52 | 95.28 ± 1.75 | 96.17 ± 1.85 | **97.52 ± 1.46** | 97.48 ± 1.63 | 97.34 ± 1.58 | 96.46 ± 2.00 | 97.49 ± 1.56 | 97.35 ± 1.51 | 97.34 ± 1.49 | 97.48 ± 1.49 | 96.80 ± 1.72 |
| | **AR-Friedman** | | 6.04 | 9.62 | 7.76 | 10.82 | 9.86 | 7.08 | 4.98 | 5.3 | 11.84 | **3.22** | 5.42 | 5.84 | 7.02 | 10.2 |

- EPIC, UMEP, MDEP, FS, BS, HC, SA, BGWO, and GA, *small-size ensembles which are trained from reduced datasets*. They are called Reduced Data Selected Ensembles (RD-SE).

The percentage of instances removed from the training set is represented by the reduction rate, Red_Rate, of AllKNN [220]. The reduction rate shows the possibility of forming simpler ensembles. Especially here, where the instance selection method is applied once, regardless of how many classifiers could be generated. The reduction reaches around 35% for $\{D_8, D_9, D_{12}, D_{24}\}$. While it reaches around 50% for $\{D_1, D_3, D_5, D_7, D_{19}\}$. BS outperforms both RFCOM and SAMME in 18 datasets. The marked performance of BS is realized while depending on 54 classifiers, on average, instead of 200 classifiers by RFCOM and SAMME. Furthermore, BS is comparable with XGBoost where BS wins in 14 out of 25 datasets.

The average rankings of Friedman Test [239], *AR-Friedman*, are presented in the last row of Table 5.6. The best ranks are scored sequentially by BS, UMEP, MDEP, and HC respectively. Those ranks prove the superiority of RD-SE to identify a high accurate subensemble. The results of the 25 datasets in Table 5.6 are illustrated in Fig. 5.3. Notably, there are more points under the diagonal, furthermore, the distance of these points from the diagonal is larger. Moreover, the figure shows a limited accuracy when we only couple the instance selection method with the unpruned proposed ensemble (UNP). Ensemble pruning via guided search, BS, proves its superiority over UNP in 22 out of 25 datasets.

It is interesting to analyze the performance of ensemble pruning via guided search for different ensemble sizes. This is important to check if its performance is consistent and not obtained due to randomness. Random Forest which is trained from whole training data, RFCOM, will be the baseline to compare against. Figure 5.4 shows the average accuracy of the pruned ensemble by (BS, UMEP, EPIC) and the unpruned one by RFCOM. Each point, in the figure, represents an average of 100 runs, and a percentage of $P = 20\%$ is determined in advance to control the subensemble size. From Fig. 5.4, the behavior of BS dominates RFCOM according to both the accuracy and the ensemble size. Furthermore, the performance of BS over RFCOM can be reached, even, by pruning a low-size ensemble, the vertical dashed-line, regardless of what the size of RFCOM is. This shows how the proposed method can be an alternative to large-size ensembles, and a lot of

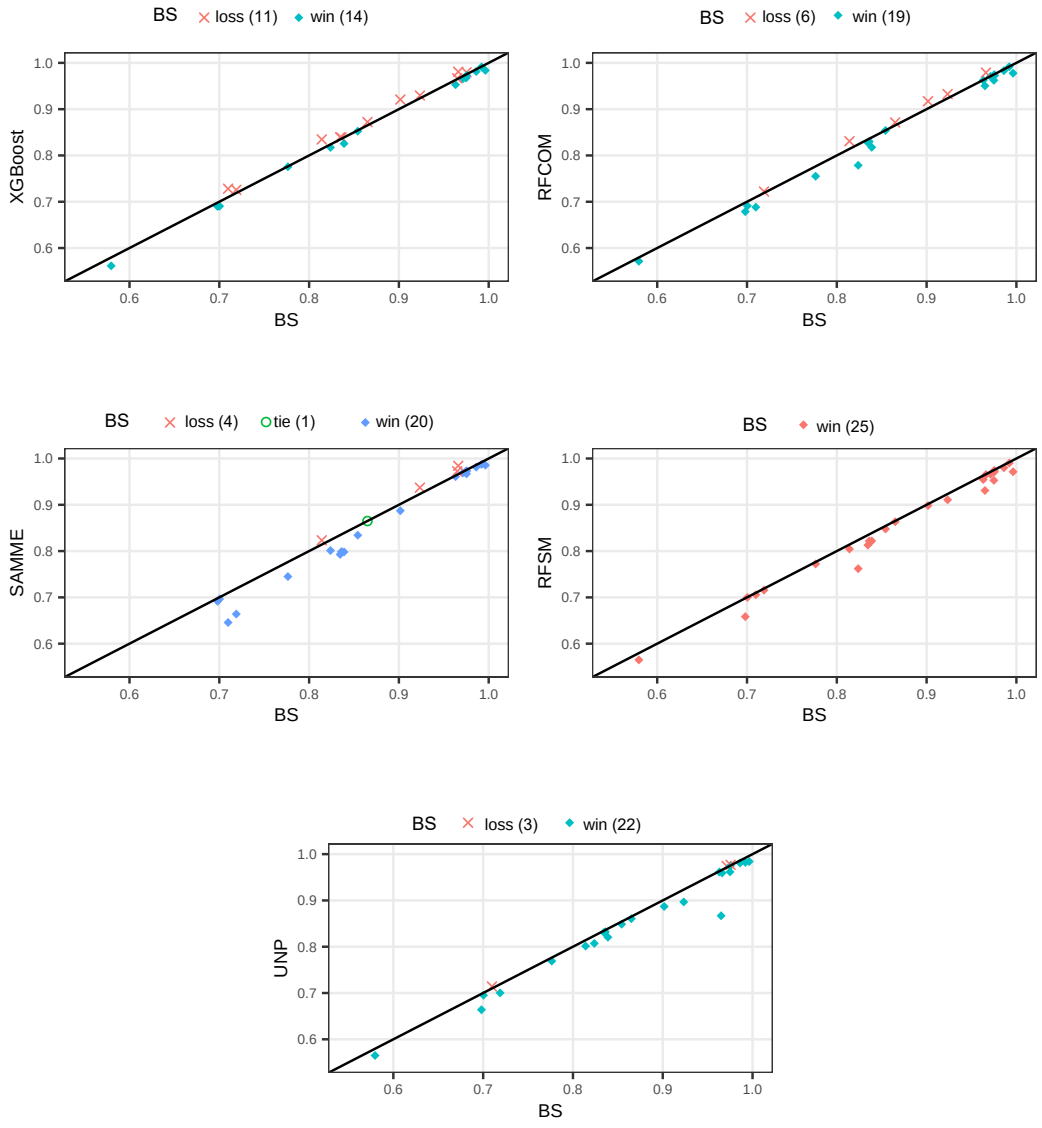


**Figure 5.3:** The performance of the proposed method against the defined ensembles NRD-NSE and RD-NSE.

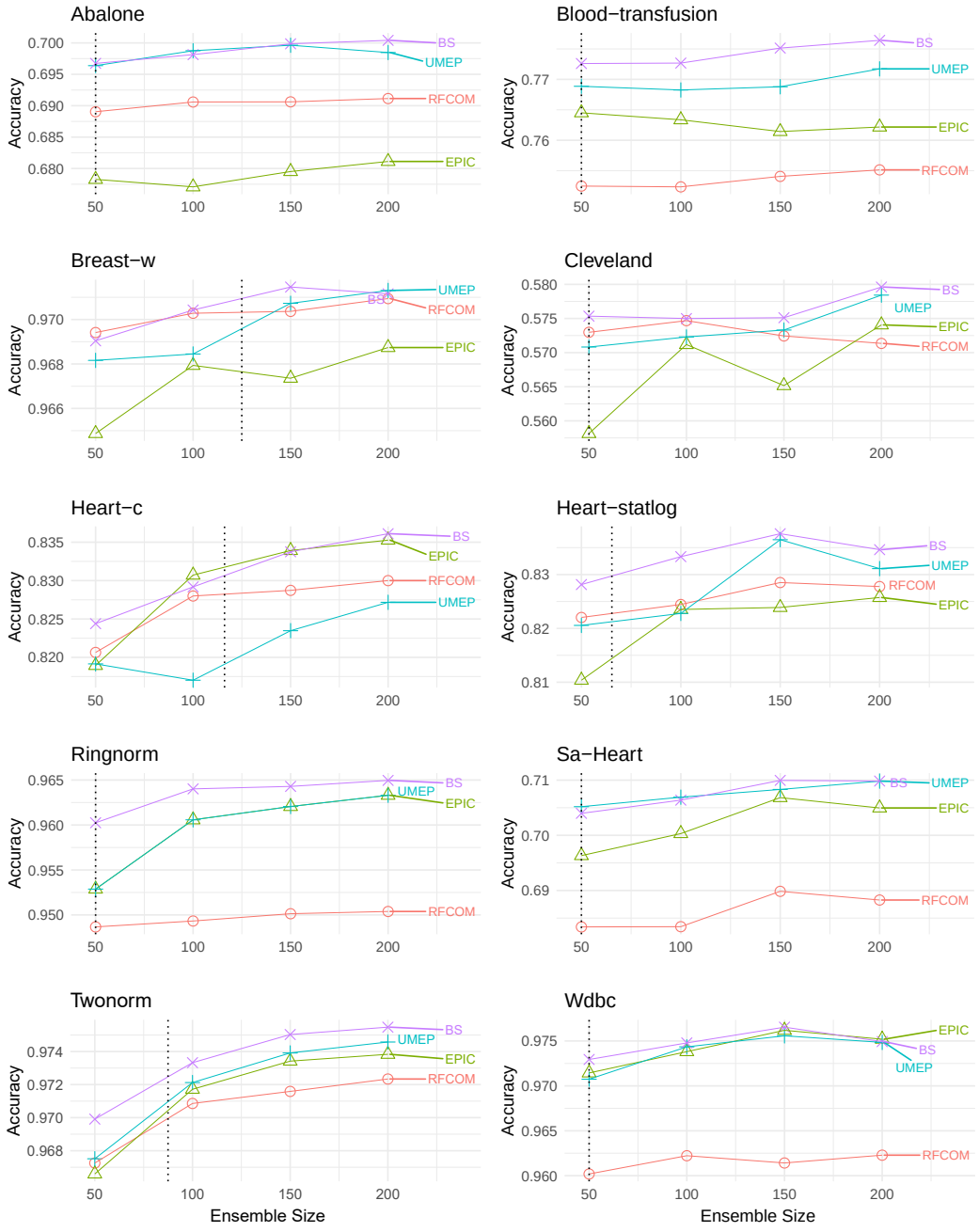


**Figure 5.4:** The performance of ensemble pruning via guided search, a comparison with RFCOM, EPIC, and UMEP.

computational resources could be saved by not overproducing models. However, this property could be different based on the classification task. We observe from Fig. 5.4, when EPIC and UMEP are ensembled together, via guided search, the subensemble realized a higher accuracy that was unreachable before. The statistical analysis will be discussed in Section 5.4.4 to determine whether there is a significant improvement or not.

### 5.4.4 Statistical Analysis

To show if there is a significant improvement or not, Wilcoxon Signed Ranks Test [227] for pairwise comparison has been used. Table 5.7 shows Wilcoxon Test [227] for the averages of the subensemble accuracy and subensemble size. The analysis from this table is as follows:

- From the upper two parts (NRD-NSE, RD-NSE) which is a matrix $5 \times 5$: RFCOM significantly outperforms SAMME and RFSM by 90% and 95%, respectively. In addition, XGBoost is the best ensemble to learn from complete training data.
- The proposed ensemble without pruning, UNP, is competitive with SAMME and RFCOM.
- All RD-SE are significantly better than UNP in terms of accuracy and ensemble size, except FS and GA are only significantly better in terms of size. This proves the limitation of the proposed ensemble without pruning.
- HC and SA are comparable with UMEP, MDEP, and RFCOM in terms of accuracy, while HC and SA are significantly better in terms of subensemble size.
- MDEP is comparable with RFCOM and UMEP in terms of accuracy. While it is better than RFCOM in terms of the ensemble size.
- All RD-SE except FS and GA are comparable with XGBoost in terms of accuracy, while all RD-SE significantly outperform XGBoost in terms of ensemble size.
- BS is the only selection strategy under the proposed method that scores many (▲). While it significantly outperforms the recent MDEP by 90%.

- In general, the lower part of Table 5.7, RD-SE, includes subensembles with many (▲) and many (•) where we have more flexibility to choose, according to the rigor of prediction and the computational resources.

**Table 5.7:** Summary of the Wilcoxon test. Shape denotes the measure used, accuracy (▲ △) and size (•○). The filled shape (▲ •) represents if the method in the row outperforms the one in the column or vice versa for (△ ○). Upper diagonal of level significance $\alpha = 0.9$, lower diagonal level of significance $\alpha = 0.95$.

| | | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) | (13) | (14) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NRD-NSE | XGBoost(1) | - - | ▲ | ▲ | ▲ | ▲ | ○ | ○ | ○ | ▲○ | ○ | ○ | ○ | ○ | ▲○ |
| | SAMME(2) | △ | - - | △ | | | △○ | △○ | △○ | ○ | △○ | △○ | △○ | △○ | ○ |
| | RFCOM (3) | △ | | - - | ▲ | | ○ | △○ | ○ | ▲○ | △○ | ○ | ○ | ○ | ○ |
| RD-NSE | RFSM (4) | △ | | △ | - - | | △○ | △○ | △○ | ▲○ | △○ | △○ | △○ | △○ | ○ |
| | UNP (5) | △ | | | | - - | △○ | △○ | △○ | ○ | △○ | △○ | △○ | △○ | ○ |
| RD-SE | EPIC (6) | • | ▲• | • | ▲• | ▲• | - - | △ | △ | ▲○ | △• | △○ | ○ | ○ | ▲○ |
| | UMEP (7) | • | ▲• | • | ▲• | ▲• | ▲ | - - | | ▲○ | △• | ○ | ○ | ▲○ | ▲○ |
| | MDEP(8) | • | ▲• | • | ▲• | ▲• | ▲ | | - - | ▲○ | △• | ○ | ○ | ○ | ▲○ |
| | FS (9) | △• | • | △• | △• | • | △• | △• | △• | - - | △• | △• | △• | △• | △• |
| | BS (10) | • | ▲• | ▲• | ▲• | ▲• | ▲○ | ▲○ | ○ | ▲○ | - - | ▲○ | ▲○ | ▲○ | ▲○ |
| | HC (11) | • | ▲• | • | ▲• | ▲• | ▲• | • | • | ▲○ | △• | - - | | ▲○ | ▲○ |
| | SA (12) | • | ▲• | • | ▲• | ▲• | • | • | • | ▲○ | △• | | - - | ▲○ | ▲○ |
| | BGWO (13) | • | ▲• | • | ▲• | ▲• | • | △• | • | ▲○ | △• | △• | • | - - | ▲○ |
| | GA (14) | △• | • | • | • | • | △• | △• | △• | ▲○ | △• | △• | △• | △• | - - |

### 5.4.5 Validation of the Methodology

We have shown the superiority of the proposed ensemble in the previous sections. However, it is necessary to validate its components (instance selection, proposed ensemble, and guided search-based pruning). Thus, we performed additional experiments to validate our method.

*The effect of instance selection to the constructed ensemble*, we compared the obtained results without instance selection (wIS) and with instance selection (proposed). Table 5.8 shows the complete results of the 25 datasets. The worst average rank via Friedman Test [239] is scored by UNP. Hence, coupling instance selection with the unpruned proposed ensemble has a limited accuracy in comparison with wIS-UNP. This means that IS contributed to form simpler, less space complexity, ensembles rather than more accurate ensembles. Table 5.9 shows the Wilcoxon Test [227] for the results from Table 5.8. The results confirm that the proposed ensemble without pruning, UNP, is not effective. However, it is comparable with ensembles that are trained from whole training data, RFCOM and SAMME.

**Table 5.8:** Average accuracy and standard deviation of the proposed method with and without instance selection, ensemble size $T = 200$, $P$=20%. The best value is in bold, while the second best is underlined.

| | | Without Instance Selection | | | | | Instance Selection (Proposed) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # | Dataset | wIS-UNP | wIS-EPIC | wIS-UMEP | wIS-MDEP | wIS-BS | UNP | EPIC | UMEP | MDEP | BS |
| $D_1$ | Abalone | **70.41±1.02** | 69.31 ± 1.00 | 69.56 ± 1.14 | 69.48 ± 1.08 | 69.47 ± 1.08 | 69.52 ± 0.64 | 68.11 ± 1.80 | 69.85 ± 1.00 | 69.96 ± 1.12 | 70.04 ± 1.14 |
| $D_2$ | Australian | 87.00± 3.10 | **87.18 ± 2.92** | 86.89 ± 2.93 | 87.13 ± 2.91 | 87.14 ± 2.88 | 86.05 ± 2.46 | 85.47 ± 2.62 | 86.43 ± 2.65 | 86.25 ± 2.80 | 86.50 ± 2.60 |
| $D_3$ | Blood-transfusion | 77.24± 1.38 | 77.20 ± 2.16 | 76.92 ± 2.14 | 77.19 ± 2.09 | 77.03 ± 2.10 | 76.90 ± 1.45 | 76.22 ± 3.65 | 77.17 ± 2.08 | 77.22 ± 2.08 | **77.64 ± 2.22** |
| $D_4$ | Breast-w | 97.25± 1.43 | 96.96 ± 1.56 | 97.17 ± 1.41 | 97.19 ± 1.48 | 97.20 ± 1.47 | **97.42 ± 1.17** | 96.87 ± 1.30 | 97.13 ± 1.25 | 97.13 ± 1.31 | 97.12 ± 1.23 |
| $D_5$ | Cleveland | 57.47± 3.04 | 57.58 ± 3.46 | 57.43 ± 3.08 | 57.60 ± 3.02 | 57.40 ± 3.57 | 56.50 ± 2.88 | 57.41 ± 4.27 | 57.84 ± 3.12 | 57.56 ± 3.10 | **57.96 ± 3.66** |
| $D_6$ | Dermatology | 97.75± 1.45 | 97.86 ± 1.58 | **97.93 ± 1.47** | **97.93 ± 1.49** | 97.92 ± 1.47 | 97.46 ± 1.65 | 97.32 ± 1.89 | 97.56 ± 1.62 | 97.63 ± 1.65 | 97.53 ± 1.72 |
| $D_7$ | Diabetic | **71.08± 2.79** | 68.64 ± 2.46 | 68.68 ± 2.52 | 68.71 ± 2.33 | 68.59 ± 2.44 | 66.38 ± 2.94 | 69.63 ± 2.75 | 70.26 ± 3.20 | 70.49 ± 3.00 | 69.81 ± 3.10 |
| $D_8$ | Heart-c | **84.02 ± 4.38** | 82.65 ± 4.28 | 82.64 ± 4.19 | 82.60 ± 4.17 | 82.97 ± 4.05 | 83.22 ± 4.49 | 83.53 ± 4.25 | 82.72 ± 4.58 | 83.12 ± 4.33 | 83.61 ± 4.26 |
| $D_9$ | Heart-statlog | **83.98± 4.09** | 82.69 ± 4.13 | 82.50 ± 4.45 | 82.83 ± 4.28 | 82.72 ± 4.26 | 83.07 ± 4.71 | 82.57 ± 4.45 | 83.11 ± 4.07 | 83.70 ± 4.26 | 83.46 ± 4.27 |
| $D_{10}$ | Ionosphere | 93.32± 2.98 | 93.85 ± 2.77 | **93.96 ± 2.67** | 93.93 ± 2.79 | 93.93 ± 2.76 | 89.67 ± 2.98 | 92.41 ± 2.61 | 91.93 ± 2.89 | 92.49 ± 2.74 | 92.34 ± 2.75 |
| $D_{11}$ | Led24 | 71.99 ± 1.61 | 70.37 ± 1.78 | **72.04 ± 1.47** | 69.84 ± 2.67 | 67.84 ± 2.82 | 70.02 ± 1.76 | 71.75 ± 1.46 | 71.78 ± 1.45 | 68.66 ± 3.79 | 71.89 ± 1.45 |
| $D_{12}$ | Mammographic | 82.85 ± 2.67 | 83.32 ± 2.63 | 83.57 ± 2.13 | 83.66 ± 2.21 | 83.84 ± 2.34 | 82.06 ± 2.51 | 83.37 ± 2.64 | 84.00 ± 2.32 | **84.05 ± 2.27** | 83.89 ± 2.36 |
| $D_{13}$ | Mfeat-fourier | 82.81 ± 1.44 | 83.24 ± 1.32 | 83.32 ± 1.40 | **83.48 ± 1.42** | 83.38 ± 1.48 | 80.14 ± 1.50 | 80.89 ± 1.65 | 81.34 ± 1.56 | 81.12 ± 1.63 | 81.41 ± 1.62 |
| $D_{14}$ | Mfeat-karh | **96.80± 0.86** | 95.82 ± 0.83 | 95.84 ± 0.86 | 95.98 ± 0.89 | 95.96 ± 0.86 | 96.13 ± 0.88 | 96.19 ± 0.86 | 96.27 ± 0.85 | 96.28 ± 0.96 | 96.31 ± 0.89 |
| $D_{15}$ | Mfeat-zernike | 80.06± 1.22 | 78.15 ± 1.48 | 78.83 ± 1.35 | 79.37 ± 1.39 | 78.89 ± 1.37 | 80.72 ± 1.24 | 82.19 ± 1.52 | **82.48 ± 1.44** | 82.20 ± 1.41 | 82.38 ± 1.39 |
| $D_{16}$ | Optdigits | 98.29± 0.35 | 98.79 ± 0.33 | 98.80 ± 0.32 | 98.80 ± 0.32 | **98.81 ± 0.32** | 98.09 ± 0.38 | 98.61 ± 0.31 | 98.60 ± 0.31 | 98.59 ± 0.30 | 98.61 ± 0.32 |
| $D_{17}$ | Penbased | 98.31± 0.27 | **99.32 ± 0.16** | 99.30 ± 0.17 | 99.29 ± 0.17 | 99.31 ± 0.16 | 98.22 ± 0.27 | 99.21 ± 0.16 | 99.19 ± 0.17 | 99.10 ± 0.69 | 99.20 ± 0.16 |
| $D_{18}$ | Ringnorm | **96.66± 0.46** | 96.62 ± 0.47 | 96.64 ± 0.49 | 96.64 ± 0.49 | 96.65 ± 0.47 | 86.69 ± 1.15 | 96.33 ± 0.51 | 96.33 ± 0.51 | 96.37 ± 0.49 | 96.50 ± 0.48 |
| $D_{19}$ | Sa-heart | 71.23± 3.38 | 69.06 ± 3.56 | 69.13 ± 3.56 | 69.44 ± 3.39 | 69.35 ± 3.19 | **71.41 ± 3.77** | 70.50 ± 4.30 | 70.98 ± 3.86 | 71.25 ± 3.65 | 70.98 ± 4.06 |
| $D_{20}$ | Satimage | 89.45± 0.74 | 91.33 ± 0.73 | 91.33 ± 0.73 | 91.34 ± 0.72 | **91.39 ± 0.75** | 88.70 ± 0.78 | 90.17 ± 0.75 | 90.05 ± 0.77 | 89.81 ± 1.39 | 90.16 ± 0.72 |
| $D_{21}$ | Segment | 96.87± 0.89 | **98.05 ± 0.68** | 98.00 ± 0.68 | 98.01 ± 0.69 | 98.03 ± 0.65 | 95.94 ± 0.90 | 96.61 ± 0.77 | 96.54 ± 0.74 | 96.51 ± 0.87 | 96.61 ± 0.77 |
| $D_{22}$ | Texture | 98.80± 0.33 | **99.71 ± 0.17** | **99.71 ± 0.17** | **99.71 ± 0.17** | **99.71 ± 0.17** | 98.40 ± 0.35 | 99.63 ± 0.19 | 99.63 ± 0.19 | 99.63 ± 0.16 | 99.63 ± 0.18 |
| $D_{23}$ | Twonorm | **97.69± 0.36** | 96.61 ± 0.45 | 96.62 ± 0.44 | 96.62 ± 0.44 | 96.68 ± 0.41 | **97.69 ± 0.37** | 97.38 ± 0.39 | 97.46 ± 0.38 | 97.47 ± 0.42 | 97.55 ± 0.35 |
| $D_{24}$ | Waveform | **85.66 ± 1.04** | 84.42 ± 1.11 | 84.39 ± 1.15 | 84.40 ± 1.17 | 84.46 ± 1.11 | 84.87 ± 1.16 | 84.59 ± 1.00 | 85.41 ± 0.95 | 85.34 ± 1.07 | 85.42 ± 0.93 |
| $D_{25}$ | Wdbc | 96.81± 1.44 | 97.57 ± 1.21 | 97.46 ± 1.29 | 97.56 ± 1.22 | **97.66 ± 1.16** | 96.17 ± 1.85 | 97.52 ± 1.46 | 97.48 ± 1.63 | 97.34 ± 1.58 | 97.49 ± 1.56 |
| **AR-Friedman** | | **4.3** | 5.76 | 5.6 | 4.84 | 4.88 | 7.54 | 6.84 | 5.44 | 5.4 | 4.4 |

*The validation of the proposed ensemble*, we show how the proposed ensemble, Section 4.3.2, is effective. Table 5.9 confirms that wIS-UNP which uses the same amount of training data is significantly better than SAMME and RFCOM at a 95% confidence level. Furthermore, there is no ensemble, among the thirteen models, that is significantly higher than wIS-UNP.

*The validation of guided search-based pruning*, we prove how guided search is more effective whether an instance selection is applied or not. From Table 5.9, when instance selection is performed, BS shows a significant difference at a 95% confidence level better than EPIC and UMEP. In addition, MDEP is significantly worse than BS at a 90% confidence level. Moreover, BS is able to compensate for the error of UNP, confirmed by the significant difference between BS and RFCOM. Without applying instance selection, wIS-BS is still outperforming wIS-EPIC and wIS-UMEP at a 95% confidence level. This shows the limited performance of the existing pruning methods in comparison with the proposed guided search-based pruning.

As a summary, with the experiments of this section. We have separately validated the different parts of our method, showing their usefulness. We proved that simpler and better ensembles could be properly designed when all the parts are used together.

**Table 5.9:** Summary of the Wilcoxon test. ▲ = the method in the row improves the method of the column. △ = the method in the column improves the method of the row. Upper diagonal of level significance $\alpha = 0.9$, Lower diagonal level of significance $\alpha = 0.95$

| | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) | (13) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| XGBoost (1) | - | ▲ | ▲ | | | | | | ▲ | | | | |
| SAMME (2) | △ | - | △ | △ | △ | △ | △ | △ | | △ | △ | △ | △ |
| RFCOM (3) | △ | | - | △ | △ | △ | △ | △ | | | △ | | △ |
| wIS-UNP (4) | | ▲ | ▲ | - | | | | | ▲ | ▲ | | | |
| wIS-EPIC (5) | | ▲ | | | - | | △ | △ | ▲ | | | | |
| wIS-UMEP (6) | | ▲ | ▲ | | | - | △ | △ | ▲ | | | | |
| wIS-MDEP (7) | | ▲ | ▲ | | ▲ | ▲ | - | | ▲ | | | | |
| wIS-BS (8) | | ▲ | ▲ | | ▲ | ▲ | | - | ▲ | | | | |
| UNP (9) | △ | | | △ | △ | △ | △ | | - | △ | △ | △ | △ |
| EPIC (10) | | ▲ | | | | | | | ▲ | - | △ | △ | △ |
| UMEP (11) | | ▲ | | | | | | | ▲ | ▲ | - | | △ |
| MDEP (12) | | ▲ | | | | | | | ▲ | ▲ | | - | △ |
| BS (13) | | ▲ | ▲ | | | | | | ▲ | ▲ | ▲ | | - |

### 5.4.6 Advantages of the proposed method

Among the advantages of the proposed method are: (1) The instance selection method is applied only once from the whole training data. (2) The individual models are independent, and the generation step could be accelerated by a parallel implementation. (3) The reduced data helps to generate less complex models with a fast prediction time. (4) The guided search preserves promising search areas to identify a subensemble properly. (5) The proposed methodology proved its capability to maximize learning from the reduced data. Regarding that, it can be used to improve the precision of IS method. (6) The ease of implementation with wide flexibility to apply different IS methods and different classifier types.

### 5.4.7 Time Complexity

The classification speed as determined in [23] mainly depends on (1) the ensemble size and (2) the complexity of individual classifiers. While Table 5.10 summarize the time complexity of the applied ensemble pruning methods. For GA & BGWO, ($\hat{P}$) represents population size, and ($E$) represents the number of epochs. While $\hat{T}$ denotes the size of the preselected classifiers, *New subset in Fig. 5.1*, that is formed by both EPIC & UMEP.

**Table 5.10:** Time complexity of ensemble selection

| Ensemble selection | Time complexity |
|---|---|
| MDEP | $\mathcal{O}(T \cdot \log T)$ |
| UMEP | $\mathcal{O}(T \cdot N) \rightarrow$ ① |
| EPIC | $\mathcal{O}(T \cdot \log T) \rightarrow$ ② |
| FS & BS | ①+②+ $\mathcal{O}(\hat{T}^3)$ |
| HC - SA | ①+②+ $\mathcal{O}(\hat{T} \cdot N \cdot M)$ |
| GA - BGWO | ①+②+ $\mathcal{O}(E \cdot \hat{P} \cdot \hat{T} \cdot N \cdot M)$ |

### 5.4.8 Exploration of the Research Questions

In this part, we determine whether the research questions of the second objective are answered or not.

***($Q_3$)*** *The effect of combining multiple pruning metrics* is so promising to go beyond, in terms of accuracy, what can be achieved from each metric alone. This has been confirmed by the outperformance of BS over both EPIC and UMEP in many datasets. In addition, the proposed method significantly outperforms MDEP by 90% without any dependence on assuming parameters in advance. Furthermore, new promising solutions can be reached in terms of both subensemble size and subensemble accuracy, i.e. (HC, SA, and BGWO).

***($Q_4$)*** *The effect of downsizing data and downsizing the number of classifiers simultaneously* has been confirmed experimentally and statistically. The RD-NSE represented by RFSM and UNP have a limited accuracy while depending on the whole ensemble members. In contrast, RD-SE represented by the lower part of Table 5.7 has many (▲) and many (•) where we have more flexibility to choose among them, according to the rigor of prediction and the ensemble size. Furthermore, the best investigated RD-SE represented by BS proved its significant superiority over state-of-art ensembles that are trained from non reduced data.

## 5.5 Conclusions

In this chapter, we have proposed a new method to alleviate the drawbacks while building and pruning an ensemble of classifiers. The objective was to generate less complex and small-size ensembles. Thus, a dual reduction on the data level and ensemble level has been considered. An instance selection method is applied to downsize the training set. After that, a bagging-like ensemble is proposed to train a set of classifiers from the reduced data. The advantage of this step is to form ensemble models quickly, especially when we generate a large number of models. In addition, the complexity of ensemble members will be reduced.

The ensemble accuracy and the ensemble size are so important and both of them can be improved through ensemble selection strategies. However, the pruning method could be biased by the selection criteria. Therefore, we have proposed a guided search that captures the properties of ensemble diversity and the individual's accuracy simultaneously. The proposed pruning strategy is capable to compensate for the limited accuracy of the unpruned ensemble. Furthermore, we showed,

graphically, how the proposed method could be an alternative to large-size ensembles. The statistically significant difference is conducted by rank-based transformations to show the superiority of the proposed methodology. The second set of experiments were conducted to validate the connected components. Notably, the superiority of the system was endowed via guided search. Moreover, the proposed method was able to maximize the learning from the reduced data.

Regarding future research, the framework can be enhanced by applying different training set selection methods. In addition, multi-objective optimization will be applied to capture the conflict between the ordering-based ensemble pruning methods that have been discussed. Furthermore, a weighted sum voting will be applied to fuse the pruned classifiers.

*As long as I have a want, I have a reason for living. Satisfaction is death.*

George Bernard Shaw

CHAPTER

# 6

# An Analysis of Heuristic Metrics for MCS Pruning

Including more classifiers in the classification task may provide more discriminating power with an equal or weighted contribution of each classifier to the final decision [31]. However, the increasing size of the ensemble hardly copes with the increasing demand to speed up the decision and to save computational resources. In bagging [3], an independent set of classifiers are generated in random order, then the final decision is adopted by a simple majority voting-based aggregation. While, it has been demonstrated that reordering the generated pool and selecting the first subset of classifiers impacts the ensemble size and the composite accuracy positively [23, 24, 31, 32, 35, 36]. The first subset of classifiers from the ordered list is expected to perform better than aggregating the whole list.

It has been proved that the generalization performance of a subensemble reports superior results over the traditional combination approaches, such as majority voting of the whole ensemble [23, 29]. Moreover, pruning down the redundant models reduces the memory burden [30]. Regarding that, MCS pruning is a proven mechanism to enhance the efficiency and elevate the efficacy of classification ensemble systems [23, 24, 31, 32]. From the review of the different pruning strategies in chapter 5, ordered-based pruning is a fast strategy with proven accuracy. Those

strategies have the following merits: (1) return subsets that are close to optimal solution (*Efficacy*) [23]. (2) easy adaptation to any given storage and computational restrictions [31, 36]. (3) the time complexity of those strategies is low, in comparison with exhaustive or optimization-based search methods (*Efficiency*) [23].

The classifiers can be reordered via a greedy search, where the set of classifiers that are expected to perform better are aggregated first. Sequentially a new subset $S_u$ is constructed from $S_{u-1}$ by incorporating a single classifier from $L_{u-1}$; $S_u = S_{u-1} \cup \Psi_k \Big| \Psi_k \in L_{u-1}$, where $T = S_u \cup L_u$ and $u = \{1, 2, 3, \ldots, T\}$. Such that the single classifier selection from $L_{u-1}$ is guided upon *a heuristic metric* to optimize the augmented ensemble $S_u$. The number of iterations or subensemble size, $\hat{T}$, can be controlled in advance to meet the computational restrictions. Furthermore, some metrics were proposed to rank all the classifiers in one batch without the sequential search. While, the property of the base classifier, an individual's accuracy, is not effective to determine this rank [35]. The generated ensemble needs to consider the hybrid between an individual's performance and an individual contribution to the ensemble diversity [32, 36]. Where it has been confirmed that the weakness of individuals can be compensated by the consensus of correct peers over different samples.

Since the practical analysis of the power of greedy search methods in [23], many research efforts have been directed to propose new heuristic measures to guide the selection of subensemble [24, 31, 32, 36]. Till now and related to our best knowledge, no work has considered the analysis of all those promising metrics together. This chapter fills that gap by comparing all these new techniques with the best performing techniques found in [23], and against other popular baseline metrics [35, 37].

In this chapter, we shed light on the importance of ordering-based ensemble pruning metrics. Reviewing and analyzing popular and recent metrics that work for bagging-based ensembles. We present a sophisticated analysis of how the pruning metrics can be affected by; the initial ensemble size ($T$), the required subensemble size ($\hat{T}$), the individual classifier type, and the binary or multiclass classification task. In addition, this study can be considered as a secure methodology to elevate the performance of bagging-like ensembles. This analysis is promising due to the

precision, prediction consistency, time-complexity, and space complexity of the investigated metrics to meet different computational restrictions.

Finally, this chapter is organized as follows: In Section 6.1, we present the main motivations to conduct this research and our contributions. The heuristic metrics in detail are to be introduced in Section 6.2. The experimental results and the statistical analysis are presented in Section 6.3. Finally, the conclusions are presented in Section 6.4.

## 6.1 Motivations and Contributions

The contribution of this proposal can be highlighted in the following points:

1. Focusing on static ensemble selection as an active research topic in multiple classifier systems.
2. Analyzing the effectiveness of different heuristic metrics to reorder the randomly bagging ensembles.
3. Separate analysis of those metrics over binary and multiclass classification tasks.
4. As far as we know, we are the first to group recent and efficient heuristic metrics for reordering bagging ensembles since they were analyzed by Martínez-Muñoz et al. [23] in 2009.

## 6.2 Heuristic Metrics

The investigated metrics are based on modifying the order of the classifiers in the bagging algorithm with the selection of the first set in the queue. Those techniques comprise dissimilar heuristic measures as: ensemble diversity [32], ensemble margin [24, 35], margin hybrid diversity [36], discriminating classifiers [31], ensemble error [37], Complementariness of misclassification [35], and relative accuracy with minimum redundancy [31]. Table 6.1 shows the heuristic metrics to be analyzed in this chapter over 30 datasets, divided into two parts as 15 binary datasets and 15 multiclass datasets.

**Table 6.1:** Heuristic metrics to guide the ordered bagging ensembles.

| Name | Section | Heuristic Measure | Year | Ref. |
|---|---|---|---|---|
| RE | 6.2.1 | Reduced Ensemble Error | 1997 | [37] |
| CC | 6.2.2 | Complementary of Misclassification | 2004 | [35] |
| MDSQ | 6.2.3 | Supervised Ensemble Margin | 2009 | [23] |
| EPIC | 5.2.1 | Diversity Contribution of Individuals | 2010 | [32] |
| UMEP | 5.2.2 | Unsupervised Ensemble Margin | 2013 | [24] |
| MDEP | 5.2.3 | Margin & Diversity | 2018 | [36] |
| MRMR | 6.2.4 | Max. Relevance & Min. Redundancy | 2018 | [31] |
| DISC | 6.2.5 | Discriminant Classifiers | 2018 | [31] |

### 6.2.1 Reduce-Error Pruning

Reduce error pruning (RE) was firstly proposed in [37]. The classifier with the highest (lowest) accuracy (error), as estimated on the pruning set $D_{pr}$, is stored in $S_1$ as the initial subset to be extended. The sequential addition of more classifiers, one at a time, is performed to get as much (less) accuracy (error) as possible. This heuristic incorporates into the subensemble the classifier $s_u$ as:

$$s_u = \arg\max_{k} \sum_{(\mathbf{x}_i, y_i) \in D_{pr}} \left[ \hat{\Psi}_{S_{u-1} \cup \Psi_k}(\mathbf{x}_i) = y_i \right] \tag{6.1}$$

where the index $k \in L_{u-1}$ and $S_u = S_{u-1} \cup \{s_u\}$. That metric has been applied in many articles as a baseline for the comparison purpose [24, 31] with superior performance over the unpruned ensemble [23].

### 6.2.2 Complementariness measure

Complementariness measure (CC) was proposed in [35] and it considers the complementariness between the incorporated models. The first subset, $S_1$, is initialized by selecting the classifier with the highest accuracy on $D_{pr}$. Then, the classifier to be nominated is the one with the highest prediction accuracy over the set of instances that are misclassified by $S_{u-1}$:

$$s_u = \arg\max_{k} \sum_{(\mathbf{x}_i, y_i) \in D_{pr}} \left[ \hat{\Psi}_{S_{u-1}}(\mathbf{x}_i) \neq y_i \wedge \Psi_k(\mathbf{x}_i) = y_i \right] \tag{6.2}$$

where $k \in L_{u-1}$ and $S_u = S_{u-1} \cup \{s_u\}$. With this heuristic, the ensemble decision is expected to be shifted towards the correct classification. However, this metric

concentrates only on the misclassified samples with no restriction to preserve the previous correct decisions.

### 6.2.3 Supervised Ensemble Margin

Margin distance minimization (MDSQ) is introduced in [23, 35], where the decision space of the individual members over the selection set, $D_{pr}$, is transformed into signature vectors. The signature vector of $\Psi_k$, $r^{(k)}$, is defined by an N-dimensional vector whose $i$th component is calculated as:

$$r_i^{(k)} = 2\left[\Psi_k(\mathbf{x}_i) = y_i\right] - 1 \quad (\mathbf{x}_i, y_i) \in D_{pr} \tag{6.3}$$

The quantity $r_i^{(k)}$ will be 1, if the $k$th classifier correctly classifies the $i$th example in $D_{pr}$, otherwise it will be -1. The ensemble signature vector, $\langle R \rangle$, is defined as the average sum of all $r^{(k)}$ as:

$$\langle R \rangle = T^{-1} \sum_{k=1}^{T} r^{(k)} \tag{6.4}$$

The subensemble whose average signature vector $\langle R \rangle$ is in the first quadrant, that is all the components are positive, correctly classifies all the examples in $D_{pr}$. The objective is to select a subensemble whose $\langle R \rangle$ is as close as possible to a reference vector, $O$, placed somewhere in the first quadrant. Hence, the reference vector is mathematically represented as:

$$O_i = q \quad , i = \{1, 2, \ldots, N\} \quad and \quad 0 < q < 1 \tag{6.5}$$

The promoted classifiers are the ones with the minimum distance between their $\langle R \rangle$ and $O$ and can be selected sequentially by minimizing:

$$s_u = \arg\min_k \; d\left(O, T^{-1}\left(r^{(k)} + \sum_{t=1}^{u-1} r^{(t)}\right)\right) \tag{6.6}$$

where $k \in L_{u-1}$ and $d(\mathrm{O},\langle R \rangle)$ is the usual Euclidean distance. The constant $q$ should be sufficiently small, $0.075$, to progressively focus on hard samples to be classified. Therefore, a subensemble with a large number of small positive values in $\langle R \rangle$ is preferred. By contrast, if the value of $q$ is close to 1 the effectiveness of the

method will be diminished as the selection will be guided upon the easy samples. In this chapter, the modified version of this metric is applied with a moving reference point $q^{(u)} = 2\sqrt{2u}/T$ as it was discussed in [23].

### 6.2.4 Maximum Relevance & Minimum Redundancy

Maximum Relevance & Minimum Redundancy pruning (MRMR) was recently proposed in [31]. It is inspired by the popular algorithm mRMR [240, 241] for reducing redundancy in feature selection problem. The metric involves two relationships; one is between the candidate class and the component class, and the other is between the candidate class and the target class. The candidate class represents the class label output of the $k$th classifier to be included, while the component class represents the class label output of the composite ensemble. The classifier with the highest accuracy, estimated on the pruning set $D_{pr}$, is stored in $S_1$ as the initial subset to be extended. The next $k$th classifier to be incorporated, $s_u$, is selected according to:

$$s_u = \arg\max_k \left[ I(\Psi_k; Y) - \frac{1}{u-1} \sum_{\Psi_i \in S_{u-1}} I(\Psi_k; \Psi_i) \right] \tag{6.7}$$

where $I(m, n)$ is the mutual information of variable $m$ and $n$; Y is the target class; $k \in L_{u-1}$ and $S_u = S_{u-1} \cup \{s_u\}$. The classifier to be selected is the one with the maximum relevance with the target class, $I(\Psi_k; Y)$, and simultaneously with minimum redundancy with $S_{u-1}$, $\frac{1}{u-1} \sum_{\Psi_i \in S_{u-1}} I(\Psi_k; \Psi_i)$.

### 6.2.5 Discriminant Classifiers

Discriminant classifiers pruning (DISC) has also been proposed in [31]. The good classifier to be incorporated is the one to compensate the current subensemble, $S_{u-1}$, taking into account the following two assumptions.

- *Assumption (1)*: Regarding the samples correctly classified by $S_{u-1}$, a good candidate is expected to do the same decisions on as many of such samples as possible.

- *Assumption (2)*: In relation to the samples misclassified by $S_{u-1}$, a good candidate is expected to classify correctly as many of those instances as possible.

The first assumption relates to the candidate classifier and the composite ensemble, while the second assumption represents how the candidate classifier relates to the target. This metric concentrates on finding the most discriminant classifier, which is relative to both $S_{u-1}$ and $Y$. The instances are divided into two parts; $\{mis\}$ represents the misclassified set by $S_{u-1}$, while $\{cor\}$ represents the set which is correctly classified by $S_{u-1}$. The incorporated classifier is selected as:

$$s_u = \arg\max_k \left[ I({\Psi_k}^{mis}; Y^{mis}) + \frac{1}{u-1} \sum_{\Psi_i \in S_{u-1}} I({\Psi_k}^{cor}; \Psi_i^{cor}) \right] \tag{6.8}$$

where $k \in L_{u-1}$ and $S_u = S_{u-1} \cup \{s_u\}$. The first term $I({\Psi_k}^{mis}; Y^{mis})$ is the mutual information that $\Psi_k$ can gain from the true labels $Y$ according to the mislabeled instances by $S_{u-1}$. Whereas the second term $\frac{1}{u-1} \sum_{\Psi_i \in S_{u-1}} I({\Psi_k}^{cor}; \Psi_i^{cor})$ is the average mutual information that $\Psi_k$ can gain from all $\Psi_i$ members of $S_{u-1}$ related to the correct classified samples.

## 6.3 Experimental Results

The experiments are dedicated to achieve objective 3, to group and analyze fast and accurate heuristic metrics for MCS pruning. The five main questions to be answered are:

- $\boldsymbol{Q_5}$. How the initial classifier pool size and the required subensemble size affect the performance of heuristic pruning metrics?
- $\boldsymbol{Q_6}$. How the heuristic pruning metrics are affected by the individual classifier type?
- $\boldsymbol{Q_7}$. How the efficacy of the pruning metrics could be affected by binary and multi-class datasets?
- $\boldsymbol{Q_8}$. How the pruning metrics are effective to reduce the performance variance?
- $\boldsymbol{Q_9}$. How the efficiency of the heuristic pruning metrics differs, in terms of time and space complexities?

### 6.3.1 Set Up

The design of experiments has considered the recommendations from [23] according to the following two issues: (A) The influence of training conditions (B) The influence of the initial pool size. *For training conditions*; the whole training data has been used, for both, to train the bagged ensemble and to prune it. *For the initial pool size*; the initial pool should contain a sufficient number of classifiers. However, the gained accuracy is not worthing the added complexity resulting from expanding the pool. In this chapter, two ensemble systems have been constructed as a part of the analysis.

1. Heterogeneous (*Different Classifier types-DC*)
    - *Samples*: Bootstrap samples, with replacement, are generated from the training data.
    - *Features*: Sixty percent of features are selected randomly for each classifier.
    - *Classifiers*: Five different classifier models, with their default setup parameters, have been used ( *DT*[1], *NB*[2], *JRip*[3], *Multinom*[4] and *KNN*[5]) with 20% as a proportional representation by each model from the whole pool size.
2. Homogeneous (*Similar Classifiers-SC*)
    - Similar to the previous, while the difference is that all individual members are of type *DT*[1].

Finally, all the datasets are preprocessed by unifying the scales of the features via normalization. For each dataset, 10 repetitions of 10 fold cross-validation procedure have been tested to get 100 runs per dataset. In addition, MDEP depends on an internal parameter $\alpha$; three values for MDEP with different $\alpha \in \{0.1, 0.5, 0.9\}$ are considered, and the best-optimized alpha according to the in train-validation is used to report the test for each dataset separately. The results for Random Forest (RF) [4], Adaboost (AdaB) [5], and the single best model (SBM) from the pool,

[1]Package C50 :https://cran.r-project.org/web/packages/C50
[2]Package e1071:https://cran.r-project.org/web/packages/e1071
[3]Package RWeka:https://cran.r-project.org/web/packages/RWeka
[4]Package nnet:http://cran.r-project.org/web/packages/nnet
[5]Package caret:https://cran.r-project.org/web/packages/caret

*according to the measured accuracy of the models on the pruning set*, are included as references in the comparison.

The default set-up for the individual classifiers types, RF, and AdaB are as follows: DT uses C5.0 decision trees of Quinlan [242] in its pruned version. NB uses Naïve Bayes with Laplace smoothing to solve the problem of zero probability. KNN applies $k$-nearest neighbor classification with $k = 3$ as the number of neighbors. RF[1] implements Breiman's random forest algorithm with ntrees=$T$, number of variables that are randomly sampled at each split=$sqrt(ncol(\mathbf{x}))$. AdaB[2] fits the AdaBoost.M1 using classification trees as base classifiers with iterations=$T$.

A total of 30 datasets that were obtained from OpenML[3] and KEEL [4] are used in this study for experimentation. The characteristics of the data are presented in Table 6.2, where #S, #F, #C, and R represent the number of samples, the number of features, the number of classes, and the ratio between the smallest, and the largest class for each dataset respectively. The number of classes varies from 2 to 10, while the maximum number of features is 100.

**Table 6.2:** Characteristics of the selected datasets for experimentation, *sorted by samples and classes*.

| DataSet | #S | #F | #C | R | DataSet | #S | #F | #C | R |
|---|---|---|---|---|---|---|---|---|---|
| Breast-cancer | 286 | 9 | 2 | 0.42 | Wine | 178 | 13 | 3 | 0.676 |
| SPECTF | 349 | 44 | 2 | 0.37 | Newthyroid | 215 | 5 | 3 | 0.2 |
| Ionosphere | 351 | 33 | 2 | 0.56 | Cmc | 1 473 | 9 | 3 | 0.529 |
| Wdbc | 569 | 30 | 2 | 0.594 | Lymphography | 148 | 18 | 4 | 0.025 |
| Indian Liver Patient (ILP) | 583 | 10 | 2 | 0.401 | Vehicle | 846 | 18 | 4 | 0.913 |
| Australian | 690 | 14 | 2 | 0.802 | Wall-Following-Robot (WFR) | 5 456 | 24 | 4 | 0.149 |
| Wisconsin | 699 | 9 | 2 | 0.526 | Cleveland | 297 | 13 | 5 | 0.081 |
| Blood-transfusion | 748 | 4 | 2 | 0.312 | Dermatology | 358 | 34 | 6 | 0.18 |
| Mammographic | 830 | 5 | 2 | 0.944 | Flare | 1 066 | 11 | 6 | 0.13 |
| Tic-tac-toe | 958 | 9 | 2 | 0.530 | Wine quality-red | 1 599 | 11 | 6 | 0.015 |
| German | 1 000 | 20 | 2 | 0.429 | Satimage | 6 435 | 36 | 6 | 0.408 |
| Hill-valley | 1 212 | 100 | 2 | 1.0 | Segment | 2 310 | 18 | 7 | 1 |
| Kr-vs-kp | 3 196 | 36 | 2 | 0.915 | Led7digit | 500 | 7 | 10 | 0.649 |
| spambase | 4 601 | 57 | 2 | 0.650 | Mfeat-Karh | 2 000 | 64 | 10 | 1 |
| Ringnorm | 7 400 | 20 | 2 | 0.981 | Mfeat-Fourier | 2 000 | 76 | 10 | 1 |

[1] randomForest:https://cran.r-project.org/web/packages/randomForest

[2] Adaboost:https://cran.r-project.org/web/packages/adabag

[3] Machine Learning Repository: https://www.openml.org

[4] KEEL Repository: http://www.keel.es

### 6.3.2 Influence of Pool Size and Selection Size

**To answer $Q_5$**, we analyze how ordered aggregation is affected by both the initial pool size ($T$) and the selection percentage ($P$) simultaneously. A heterogeneous ensemble, see Section 6.3.1, composed of 201 models is built. The classifiers will be ordered considering only the first 25, 51, 75, 101, 151, and 201 models from ensemble size. This means that the classifiers are nested such that all classifiers in an initial pool are also included in larger pools. The average accuracy over 100 runs is computed for each ensemble size $T$.

Table 6.3 shows the analysis of 1200 records ($T = 6 \times$ runs=100 $\times$ $P = 2$) for the *SPECTF* dataset. The last two columns represent the size of the ordered subensemble according to an initial pool of $T$. For each $T$ and $P$, the best value is highlighted in bold. The prediction accuracy of the bagging increases monotonically as a function of the initial pool size, while this saturation level sometimes decreases according to the classification task. All the investigated metrics prove their superiority over bagging, *regarding SPECTF dataset*. The poor results reported for BSM, confirm that ensemble outperforms all single models from which it is composed. Regarding the selection percentage of $P = 30\%$, the accuracy of DISC, EPIC, and UMEP keep on increasing as more classifiers are included in the initial pool. *In general*, the accuracy of the ordered classifiers with a percentage of 30% is better than 20%. In addition, the poorest accuracy by any ordering metric is better than the highest accuracy of bagging. The highest accuracy of 91.49 is reported by DISC using a subensemble composed of 61 classifiers, $\hat{T}$, instead of the 201 classifiers used by bagging.

Figure 6.1 shows the complementary perspective about the results. The comparison of DISC-30% with DISC-20% and the comparison of UMEP-30% with UMEP-20% confirm that the ordered subset with a larger number of classifiers returns higher accuracy. Furthermore, the lowest accuracy by UMEP-20%, *horizontal-dashed line*, with only 5 classifiers outperforms the accuracy of bagging for any size. The prediction accuracy of DISC-30% and UMEP-30% keeps on increasing even after bagging has stabilized. In addition, the figure shows that BSM is the worst ensemble selection strategy. We confirm that the main objective of heuristic metrics is to return a subensemble with a reduced number of classifiers while

**Table 6.3:** The average accuracy of the subensemble related to a selection percentage ($P$) from an inital pool size ($T$); *SPECTF dataset*.

| $T$ | Bagging | BSM | RE | CC | MDSQ | MRMR | DISC | EPIC | UMEP | MDEP | $P$ | $\hat{T}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 25 | **84.73** | 81.87 | 86.73 | 86.21 | 87.23 | 86.43 | **88.43** | 86.49 | 86.65 | 86.77 | 20% | 5 |
| | | | 87.66 | 87.11 | 88.18 | 87.19 | **88.69** | 87.77 | 87.74 | 87.71 | 30% | 7 |
| 51 | **85.27** | 82.96 | 89.22 | 88.41 | 89.91 | 88.35 | **90.41** | 89.65 | 89.54 | 89.72 | 20% | 11 |
| | | | 88.97 | 88.54 | 89.82 | 88.62 | **90.49** | 89.65 | 89.76 | 89.80 | 30% | 15 |
| 75 | **85.48** | 82.92 | 88.89 | 88.80 | 90.24 | 88.12 | **91.10** | 90.28 | 90.83 | 90.66 | 20% | 15 |
| | | | 88.80 | 88.94 | 89.69 | 88.20 | **90.89** | 90.38 | 90.32 | 90.21 | 30% | 23 |
| 101 | **85.85** | 83.09 | 89.08 | 89.06 | 89.90 | 87.77 | **91.21** | 90.56 | 90.22 | 90.44 | 20% | 21 |
| | | | 88.93 | 88.80 | 88.83 | 88.14 | **91.01** | 90.84 | 90.41 | 90.78 | 30% | 31 |
| 151 | **85.82** | 82.78 | 89.18 | 88.89 | 89.29 | 88.54 | **91.09** | 90.58 | 90.59 | 90.44 | 20% | 31 |
| | | | 88.82 | 88.97 | 88.03 | 88.91 | **91.24** | 90.64 | 90.78 | 90.70 | 30% | 45 |
| 201 | **85.84** | 83.68 | 89.12 | 89.19 | 88.48 | 88.82 | 91.07 | 91.07 | **91.15** | 91.10 | 20% | 41 |
| | | | 89.22 | 88.88 | 87.36 | 88.70 | **91.49** | 91.00 | 91.04 | 91.09 | 30% | 61 |

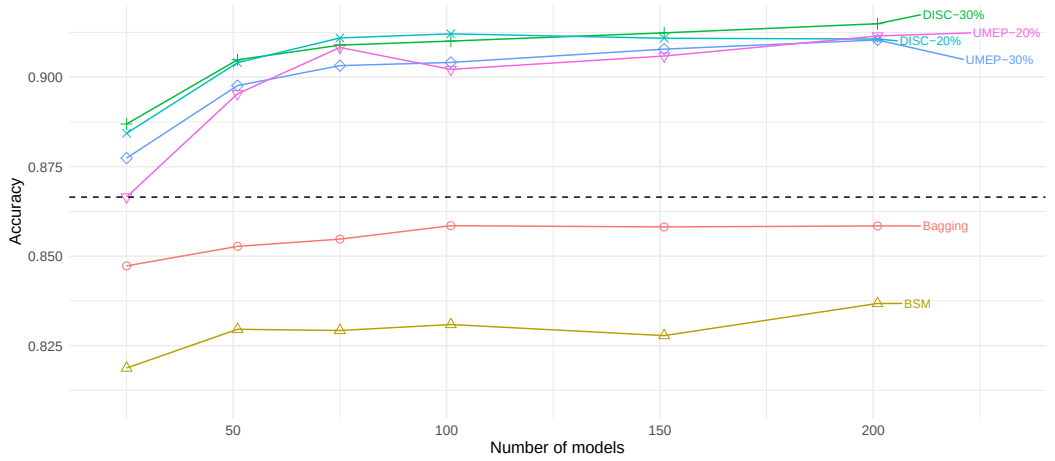


**Figure 6.1:** Influence of pool size and selection size on the general accuracy; *SPECTF dataset*.

keeping the accuracy unaffected. Going beyond the accuracy of the bagging is conditioned by the effectiveness to reorder the ensemble as we will discuss in Sections 6.3.4 and 6.3.5.

### 6.3.3 Influence of Heterogeneous Classifiers

**To answer $Q_6$**, we analyze how the general accuracy and the performance of the metrics are affected by the type of combined individual models. For that, the initial pool size is fixed with $T = 101$ as a balance between accuracy and ensemble's

complexity. Then the two types of ensembles, *homogeneous and heterogeneous*, described in Section 6.3.1, are constructed for comparison purposes. Table 6.4 presents the average accuracy and the standard deviation for the ensembles using different and similar individual classifiers over six representative datasets.

**Table 6.4:** Average accuracy and standard deviation for Different Classifiers (DC) and Similar Classifiers (SC); $T$= 101 and $P$=30%.

| | DC | | | | SC | | | |
|---|---|---|---|---|---|---|---|---|
| Dataset | Bagging | RE | DISC | UMEP | Bagging | RE | DISC | UMEP |
| Ensemble size | $T = 101$ | $\hat{T} = 31$ | | | $T = 101$ | $\hat{T} = 31$ | | |
| Australian | 86.70 ± 3.39 | **86.97 ± 3.52** | **86.97 ± 3.48** | **87.36 ± 3.35** | 86.85 ± 3.43 | 86.66 ± 3.85 | 86.34 ± 3.81 | **86.95 ± 3.67** |
| Blood | 77.37 ± 1.77 | 77.35 ± 2.68 | **77.53 ± 2.74** | 77.03 ± 2.73 | 76.27 ± 0.73 | **77.53 ± 2.77** | **77.92 ± 2.47** | **77.14 ± 2.80** |
| Breast-cancer | 73.41 ± 5.73 | **73.94 ± 6.84** | **73.45 ± 6.03** | 72.97 ± 6.90 | 73.48 ± 5.08 | 73.02 ± 5.63 | 73.10 ± 5.58 | 71.41 ± 6.39 |
| Cmc | 53.30 ± 3.74 | 53.29 ± 3.83 | **53.80 ± 3.72** | **53.82 ± 3.69** | 53.33 ± 3.84 | **53.38 ± 3.41** | **53.57 ± 3.58** | **53.65 ± 3.61** |
| Mammographic | 83.04 ± 4.02 | 82.72 ± 4.00 | 82.59 ± 4.35 | **83.87 ± 4.02** | 83.70 ± 3.41 | 82.99 ± 3.62 | 83.19 ± 3.56 | 83.64 ± 3.53 |
| Wdbc | 96.63 ± 2.42 | **97.10 ± 2.32** | **97.44 ± 2.03** | **97.54 ± 1.95** | 96.58 ± 2.23 | 96.48 ± 2.40 | **96.76 ± 2.32** | **96.67 ± 2.27** |
| **AR-Friedman** | 5.33 | 4.42 | 3.33 | **3** | 5 | 5.75 | **4.33** | 4.83 |

The results prove that the general accuracy of bagging can be outperformed in both cases by using pruned ensemble, *highlighted bold values in each part*. To differentiate among the methods, the average rank of Friedman test [226] *(AR-Friedman)* is calculated and is shown in the last row of Table 6.4. The diversity in decision space, which is achieved by different classifiers, guarantees effective performance for the ordering methods. The best ranks, upon the selected datasets, are reported for DC-UMEP, DC-DISC, and DC-RE in comparison with their versions under similar classifiers. We conclude that similar classifiers produce similar decisions approximately, and the ability to differentiate among them by ordering metrics decreases. For the rest of our experiments, an ensemble using different classifier types with fixed pool size, $T = 101$, will be formed for the evaluation purpose.

### 6.3.4 Analysis Over Binary Datasets

**To answer first part of $Q_7$**, for solving binary datasets. Table 6.5 shows the average accuracy and standard deviation for the different datasets. Adaboost achieves the highest accuracy in six datasets $\{D_6, D_7, D_8, D_{10}, D_{11}, D_{13}\}$ while it uses 101 classifiers. RE and MDEP both of them achieve the highest accuracy for $\{D_3\}$ and $\{D_1, D_9, D_{14}\}$ respectively using 31 classifiers. The highest improvement percentage of 14.52% has been recorded by AdaB in comparison with MDEP for $D_{13}$.

**Table 6.5:** Average accuracy and standard deviation over binary datasets for ensemble size $T = 101$ and $P$=30%. The values that outperform bagging are highlighted in bold.

| | | $T = 101$ | $\hat{T} = 1$ | $\hat{T} = 31$ | | | | | | | | $T = 101$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # | file | Bagging | BSM | RE | CC | MDSQ | MRMR | DISC | EPIC | UMEP | MDEP | AdaB | RF |
| $D_1$ | Australian | 86.70 ± 3.39 | 84.03 ± 4.13 | **86.97 ± 3.52** | **86.87 ± 3.39** | 86.47 ± 3.78 | **86.75 ± 3.44** | **86.97 ± 3.48** | **87.29 ± 3.38** | **87.36 ± 3.35** | **87.38 ± 3.32** | **86.94 ± 3.70** | **86.97 ± 3.57** |
| $D_2$ | Blood-transfusion | 77.37 ± 1.77 | 73.69 ± 4.33 | 77.35 ± 2.68 | **77.70 ± 2.85** | **77.65 ± 2.79** | **77.70 ± 2.80** | **77.53 ± 2.74** | 77.34 ± 2.96 | 77.03 ± 2.73 | 77.36 ± 2.95 | 76.09 ± 3.85 | 75.31 ± 4.10 |
| $D_3$ | Breast-cancer | 73.41 ± 5.73 | 68.64 ± 8.16 | **73.94 ± 6.84** | **73.45 ± 5.98** | 73.03 ± 4.99 | **73.77 ± 6.10** | **73.45 ± 6.03** | 73.28 ± 6.63 | 72.97 ± 6.90 | 73.32 ± 6.56 | 68.75 ± 7.62 | 71.50 ± 8.06 |
| $D_4$ | German | 74.98 ± 2.90 | 69.26 ± 4.43 | **75.65 ± 3.24** | **75.48 ± 3.10** | 74.67 ± 2.87 | **75.19 ± 3.22** | **75.77 ± 3.06** | **75.52 ± 3.24** | **75.77 ± 3.19** | **75.77 ± 3.22** | **75.39 ± 3.56** | **76.09 ± 3.30** |
| $D_5$ | Hill-valley | 67.81 ± 5.09 | **86.00 ± 4.66** | **82.80 ± 6.54** | **80.10 ± 7.26** | **78.47 ± 5.23** | **80.04 ± 7.29** | **79.23 ± 5.32** | **74.82 ± 9.06** | **81.71 ± 4.12** | **81.79 ± 4.17** | 59.06 ± 3.78 | 57.58 ± 3.74 |
| $D_6$ | ILP | 70.33 ± 5.55 | 67.22 ± 5.13 | 69.99 ± 5.20 | 68.96 ± 5.75 | 70.28 ± 5.10 | 68.84 ± 5.76 | 70.30 ± 5.13 | **71.44** ± 4.36 | **71.29 ± 5.03** | **71.43 ± 4.59** | **71.59 ± 5.06** | **70.52 ± 5.30** |
| $D_7$ | Ionosphere | 93.37 ± 3.71 | 89.25 ± 5.54 | **93.48 ± 3.81** | **93.54 ± 3.79** | **93.90 ± 3.11** | **93.60 ± 3.89** | **93.80 ± 3.75** | **93.60 ± 3.90** | **93.80 ± 3.89** | **93.85 ± 3.88** | **94.08 ± 3.30** | 93.25 ± 4.05 |
| $D_8$ | Kr-vs-kp | 96.25 ± 1.20 | **96.46 ± 1.46** | **98.42 ± 0.83** | **98.25 ± 0.76** | **97.92 ± 0.88** | **98.20 ± 0.77** | **98.37 ± 0.68** | **96.76 ± 1.23** | **96.73 ± 1.23** | **96.76 ± 1.22** | **99.62 ± 0.33** | **98.67 ± 0.68** |
| $D_9$ | Mammographic | 83.04 ± 4.02 | 82.46 ± 4.27 | 82.72 ± 4.00 | 82.21 ± 4.36 | 82.35 ± 3.90 | 82.37 ± 4.38 | 82.59 ± 4.35 | **83.94 ± 3.77** | **83.87 ± 4.02** | **84.00 ± 3.97** | 80.73 ± 4.18 | 81.78 ± 4.09 |
| $D_{10}$ | Ringnorm | 96.66 ± 0.60 | 93.72 ± 2.11 | **96.77 ± 0.64** | **96.77 ± 0.59** | 95.07 ± 0.69 | **96.79 ± 0.60** | **96.85 ± 0.61** | **96.80 ± 0.66** | **96.79 ± 0.68** | **96.79 ± 0.68** | **97.33 ± 0.55** | 94.98 ± 0.80 |
| $D_{11}$ | Spambase | 94.98 ± 1.07 | 91.52 ± 1.44 | **95.04 ± 1.00** | 94.75 ± 1.04 | 94.72 ± 1.04 | 94.76 ± 1.01 | **94.99 ± 1.15** | 94.93 ± 1.06 | 94.85 ± 1.11 | 94.91 ± 1.08 | **95.72 ± 0.87** | **95.31 ± 0.98** |
| $D_{12}$ | SPECTF | 85.96 ± 5.32 | 82.18 ± 5.98 | **89.17 ± 4.78** | **89.20 ± 4.85** | **88.48 ± 5.50** | **88.64 ± 4.85** | **91.35 ± 4.71** | **90.66 ± 4.75** | **90.84 ± 4.74** | **90.92 ± 4.69** | **91.24 ± 3.96** | **92.39 ± 4.30** |
| $D_{13}$ | Tic-tac-toe | 76.57 ± 2.85 | **77.49 ± 4.41** | **86.10 ± 3.14** | **86.18 ± 3.24** | **86.65 ± 3.24** | **86.23 ± 3.17** | **86.53 ± 3.18** | **87.08 ± 2.98** | **85.66 ± 3.22** | **87.06 ± 3.34** | **99.47 ± 0.85** | **98.72 ± 1.18** |
| $D_{14}$ | Wdbc | 96.63 ± 2.42 | 95.71 ± 2.55 | **97.10 ± 2.32** | **97.10 ± 2.24** | **96.89 ± 2.10** | **97.17 ± 2.31** | **97.44 ± 2.03** | **97.42 ± 2.03** | **97.54 ± 1.95** | **97.54 ± 1.94** | **96.82 ± 2.13** | 96.14 ± 2.37 |
| $D_{15}$ | Wisconsin | 97.31 ± 2.01 | 95.20 ± 2.33 | 97.22 ± 1.95 | 97.22 ± 2.11 | 97.21 ± 2.15 | 97.19 ± 2.18 | 97.18 ± 2.15 | 97.27 ± 2.04 | 97.12 ± 1.95 | 97.22 ± 2.14 | 96.83 ± 2.13 | 97.12 ± 1.84 |
| **AR-Friedman** | | 8 | 10.8 | **5.6** | **6.73** | **7.8** | **6.73** | **4.6** | **5.07** | **5.83** | **4** | **5.87** | **6.97** |

While MDEP recorded the highest improvement over AdaB for $D_5$ by 38.49%. For almost all the datasets, the reordering metrics guarantee higher accuracy and go beyond what can be achieved by bagging. MDSQ is the only metric with a tendency to form the worst subensemble related to the investigated tasks. For the *Hill-valley* dataset, the poor performance of bagging is caused by the uneven response of individual members. The performance analysis of RE to select a subensemble over the 100 executions, depends on grouping specific individual types. Where DT, Multinom, NB, JRip, and KNN are represented in-order according to the following percentages 33.97%, 28.97%, 20.48%, 10.45%, 6.13% respectively.

The average rank of Friedman test [226], *AR-Friedman*, is presented in the last row of Table 6.5 with the best ranks being (in order) MDEP, DISC, EPIC, RE, UMEP, AdaB, CC, and MRMR. Next, the Wilcoxon test [243, 244] for pairwise comparison has been performed to detect significant differences between the two sample means. From Table 6.6, BSM is the worst ensemble selection strategy. All heuristic metrics, except MDSQ, significantly outperform bagging by 95% or 90%. Furthermore, we notice that MDSQ, AdaB, and RF are at the same level of accuracy as bagging. While the heterogeneous classifiers selected by RE and CC significantly outperform bagging. Finally, MDEP is the only metric that significantly outperforms both EPIC and UMEP by 95% and is the best metric regarding the investigated tasks.

**Table 6.6:** Summary of the Wilcoxon test (for binary datasets). •= the method in the row improves the method of the column. ○= the method in the column improves the method of the row. Upper diagonal of level significance $\alpha = 0.9$, lower diagonal level of significance $\alpha = 0.95$.

| | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bagging (1) | - | • | ○ | ○ | | ○ | ○ | ○ | ○ | ○ | | |
| BSM (2) | ○ | - | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| RE (3) | • | • | - | • | • | • | | | | | | |
| CC (4) | | • | | - | | | ○ | | | ○ | | |
| MDSQ (5) | | • | ○ | | - | | ○ | | ○ | ○ | | |
| MRMR (6) | | • | ○ | | | - | ○ | | | ○ | | |
| DISC (7) | • | • | | • | • | | - | | | | | |
| EPIC (8) | • | • | | | | | | - | | ○ | | |
| UMEP (9) | • | • | | | | | | | - | ○ | | |
| MDEP (10) | • | • | | • | • | • | | • | • | - | | |
| AdaB (11) | | • | | | | | | | | | - | |
| RF (12) | | • | | | | | | | | | | - |

### 6.3.5 Analysis Over Multiclass Datasets

**To answer second part of $Q_7$**, to analyze the stability of the different pruning methods in multiclass classification tasks. Table 6.7 shows the average accuracy and the standard deviation over multi-class datasets. Adaboost and RF achieve the highest accuracy in datasets $\{D_{17}, D_{26}, D_{27}, D_{28}\}$ and $\{D_{16}, D_{25}, D_{29}, D_{30}\}$, respectively using the complete set of 101 classifiers. The highest improvement percentage of Adaboost over DISC has been recorded by 3.94% for $D_{27}$. While DISC recorded the highest improvement over Adaboost by 12.77% for $D_{30}$.

For datasets with a large number of samples and classes like $\{D_{22}, D_{23}\}$, as expected, we notice that DISC, MDEP are the best. Our explanation, for complex decision spaces, it will be preferred to select classifiers that are more discriminant or that can classify difficult samples. For that, DISC is more promising to acquire complementary information inside the subensemble by reducing the internal conflict. While MDEP is preferred to balance between the individual's accuracy and ensemble diversity. In addition, DISC is the best for $D_{25}$ as the largest size multiclass dataset.

For datasets with a small number of classes and a small number of instances like $\{D_{16}, D_{21}, D_{24}\}$, the decision space becomes easier as low conflict will exist. For that, except MDSQ for $D_{16}$, RE proved its superiority to select effective subensemble based on its general accuracy to outperform margin-based metrics {MDSQ, UMEP, MDEP}. For datasets with a small number of classes and a larger number of instances like $\{D_{17}, D_{28}\}$; DISC, EPIC, and UMEP outperform RE.

For the statistical ranking, *AR-Friedman* is presented in the last row of Table 6.7 with the best ranks scored sequentially by DISC, EPIC, RF, UMEP, CC, MRMR & MDEP, and RE. While Table 6.8 shows the pairwise comparison between the 3 unpruned ensembles and the 9 pruned ones. Table 6.8 confirms that BSM is the worst selection/pruning strategy. All the investigated metrics, except MRMR, significantly outperform bagging by 95% according to both the accuracy and the ensemble size. Adaboost and RF are at the same level of accuracy with all pruning metrics, however, their performance is achieved using $T$=101 classifiers. MDEP waived his rank in favor of DISC and EPIC. While DISC is the best metric for selecting subensemble according to the investigated datasets. It has been confirmed

**Table 6.7:** Average accuracy and standard deviation over multiclass datasets for ensemble size $T = 101$ and $P$=30%. The values that outperform bagging are highlighted in bold.

| | | $T = 101$ | $\hat{T} = 1$ | $\hat{T} = 31$ | | | | | | | | $T = 101$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # | file | Bagging | BSM | RE | CC | MDSQ | MRMR | DISC | EPIC | UMEP | MDEP | AdaB | RF |
| $D_{16}$ | Cleveland | 57.07 ± 4.41 | 51.40 ± 7.76 | **57.30 ± 5.32** | 57.02 ± 4.75 | **57.35 ± 4.91** | **57.11 ± 4.66** | **57.38 ± 4.86** | **57.43 ± 5.08** | 56.89 ± 4.81 | **57.21 ± 5.05** | **57.52 ± 5.43** | **57.72 ± 5.29** |
| $D_{17}$ | Cmc | 53.30 ± 3.74 | 49.58 ± 3.81 | 53.29 ± 3.83 | 53.20 ± 3.66 | **53.36 ± 3.81** | **53.49 ± 3.97** | **53.80 ± 3.72** | **53.43 ± 3.69** | **53.82 ± 3.69** | 53.14 ± 4.27 | **56.30 ± 3.44** | **53.98 ± 3.42** |
| $D_{18}$ | Dermatology | 97.66 ± 2.57 | 93.75 ± 4.85 | 97.49 ± 2.62 | 97.52 ± 2.63 | 97.59 ± 2.44 | 97.57 ± 2.62 | **97.88 ± 2.18** | **97.99 ± 2.14** | **97.96 ± 2.24** | **97.99 ± 2.14** | 96.48 ± 2.64 | 97.65 ± 2.28 |
| $D_{19}$ | Flare | 73.23 ± 3.41 | **73.59 ± 3.27** | **74.65 ± 2.77** | **75.22 ± 3.03** | **75.31 ± 2.72** | 69.89 ± 4.29 | **75.60 ± 2.81** | **75.65 ± 2.82** | **75.57 ± 2.77** | **74.78 ± 3.63** | **75.26 ± 2.99** | **75.14 ± 2.45** |
| $D_{20}$ | Led7digit | 69.51 ± 5.28 | 60.31 ± 5.72 | **72.12 ± 5.34** | **70.89 ± 5.69** | **73.17 ± 5.77** | **69.65 ± 6.07** | **71.39 ± 6.11** | **72.11 ± 5.88** | **71.22 ± 5.69** | **70.42 ± 6.52** | **72.66 ± 5.94** | **71.86 ± 5.64** |
| $D_{21}$ | Lymphography | 85.45 ± 8.78 | 77.45 ± 10.29 | **86.34 ± 8.17** | **86.42 ± 8.75** | **86.23 ± 9.58** | **86.57 ± 9.17** | 85.35 ± 9.39 | 84.94 ± 9.27 | 84.32 ± 9.59 | 85.30 ± 9.72 | 85.11 ± 9.01 | 84.55 ± 9.51 |
| $D_{22}$ | Mfeat-Fourier | 83.10 ± 2.20 | 71.81 ± 2.96 | **83.36 ± 2.51** | **83.38 ± 2.34** | **83.19 ± 2.43** | **83.58 ± 2.27** | **83.56 ± 2.46** | **83.20 ± 2.44** | **83.38 ± 2.40** | **83.58 ± 2.36** | 81.86 ± 2.56 | 83.10 ± 2.17 |
| $D_{23}$ | Mfeat-karh | 96.87 ± 1.08 | 90.75 ± 3.61 | 96.83 ± 1.13 | 96.78 ± 1.16 | 96.50 ± 1.41 | 96.75 ± 1.23 | **97.13 ± 1.09** | 96.74 ± 1.18 | 96.86 ± 1.17 | **97.01 ± 1.11** | 95.40 ± 1.47 | 96.04 ± 1.23 |
| $D_{24}$ | Newthyroid | 96.55 ± 3.68 | 95.43 ± 4.73 | **96.88 ± 3.51** | **96.84 ± 3.50** | 96.32 ± 4.11 | **96.65 ± 3.45** | **96.60 ± 3.69** | **96.69 ± 3.78** | 95.91 ± 3.98 | 96.46 ± 3.92 | 95.24 ± 4.07 | 96.08 ± 4.04 |
| $D_{25}$ | Satimage | 89.66 ± 1.14 | 85.09 ± 1.39 | **91.48 ± 1.02** | **91.63 ± 1.05** | **91.09 ± 1.04** | **91.61 ± 1.02** | **91.67 ± 1.04** | **91.52 ± 1.02** | **91.54 ± 0.98** | **91.50 ± 0.98** | 88.25 ± 1.24 | **91.82 ± 0.95** |
| $D_{26}$ | Segment | 97.05 ± 1.07 | 95.88 ± 1.58 | **97.94 ± 0.95** | **98.01 ± 0.90** | **97.49 ± 1.06** | **97.97 ± 0.90** | **98.09 ± 0.98** | **98.13 ± 0.95** | **98.12 ± 0.91** | **98.13 ± 0.92** | **98.55 ± 0.83** | **97.93 ± 1.01** |
| $D_{27}$ | Vehicle | 74.99 ± 4.07 | 69.23 ± 4.85 | **75.56 ± 4.11** | **75.42 ± 3.92** | **75.14 ± 3.90** | **75.48 ± 4.10** | **75.32 ± 3.92** | **75.13 ± 4.03** | **75.14 ± 3.80** | **75.13 ± 3.86** | **78.29 ± 4.10** | **75.13 ± 4.29** |
| $D_{28}$ | WFR | 96.62 ± 0.85 | **99.03 ± 0.53** | **99.04 ± 0.41** | **99.06 ± 0.38** | **98.95 ± 0.44** | **99.04 ± 0.43** | **99.40 ± 0.32** | **99.40 ± 0.32** | **99.40 ± 0.31** | **99.38 ± 0.38** | **99.88 ± 0.14** | **99.43 ± 0.32** |
| $D_{29}$ | Wine | 98.11 ± 3.07 | 94.45 ± 5.44 | 97.94 ± 3.41 | 97.94 ± 3.41 | 97.81 ± 3.33 | 97.77 ± 3.53 | **98.16 ± 3.17** | **98.16 ± 3.07** | **98.16 ± 3.07** | **98.16 ± 3.07** | 96.20 ± 4.54 | **98.22 ± 3.24** |
| $D_{30}$ | Winequality-red | 63.40 ± 2.97 | 56.71 ± 4.30 | **68.38 ± 2.99** | **68.57 ± 3.01** | **68.30 ± 3.31** | **68.76 ± 3.08** | **68.55 ± 3.08** | **68.44 ± 2.92** | **68.06 ± 2.98** | **67.72 ± 3.13** | 60.79 ± 3.30 | **70.53 ± 3.44** |
| **AR-Friedman** | | 8.5 | 11.67 | **6.13** | **5.87** | **7.03** | **6.07** | **3.9** | **4.83** | **5.83** | **6.07** | **6.6** | **5.5** |

that ordered bagging based on complementary decisions works well for multiclass datasets, *confirmed by CC performance*. Next, it is interesting to combine all the binary and multiclass datasets to analyze ordering based pruning metrics in general.

**Table 6.8:** Summary of the Wilcoxon test (for Multiclass datasets). •= the method in the row improves the method of the column. ○= the method in the column improves the method of the row. Upper diagonal of level significance $\alpha = 0.9$, lower diagonal level of significance $\alpha = 0.95$.

| | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bagging (1) | - | • | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | | ○ |
| BSM (2) | ○ | - | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| RE (3) | • | • | - | | | | | | | | | |
| CC (4) | • | • | | - | | | ○ | | | | | |
| MDSQ (5) | • | • | | | - | | ○ | | | | | |
| MRMR (6) | | • | | | | - | ○ | | | | | |
| DISC (7) | • | • | | | | | - | | • | • | | |
| EPIC (8) | • | • | | | | | | - | | | | |
| UMEP (9) | • | • | | | | | ○ | | - | | | |
| MDEP (10) | • | • | | | | | ○ | | | - | | |
| AdaB (11) | | • | | | | | | | | | - | |
| RF (12) | • | • | | | | | | | | | | - |

### 6.3.6 General Analysis Over All Datasets

A nonparametric statistical test is conducted over the thirty datasets, $D_1$ to $D_{30}$, to check if there is a significant difference between the performance of the ordered bagging methods or not. Using the methodology proposed by Demšar [245], Figure 6.2 shows the Nemenyi post hoc test for $\alpha = 0.05$. Methods that are connected are not significantly different based on the absolute difference in the average rankings. The Critical Difference is shown (CD=3.06 for 12 methods, 30 datasets, $\alpha = 0.05$). The analysis shows that DISC, EPIC, and MDEP significantly outperform bagging by 95%. While, the inferior performance is recorded by BSM, Bagging, and MDSQ.

### 6.3.7 Prediction Consistency

**To answer $Q_8$**, after statistical analysis and according to the Nemenyi test, we selected {DISC, EPIC, MDEP, UMEP, AdaB, RF, Bagging, BSM}, and analyzed

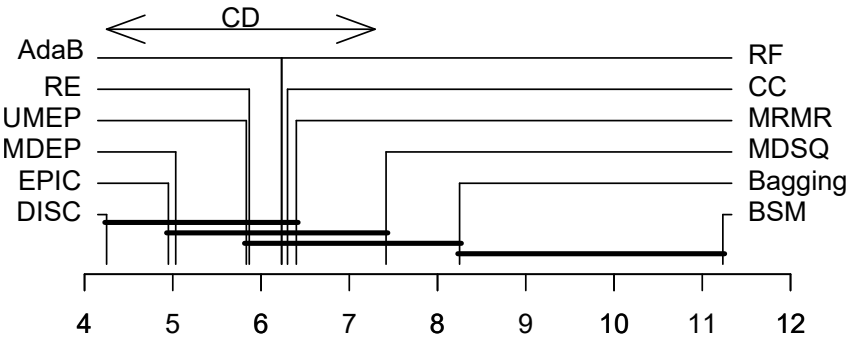


**Figure 6.2:** Comparison of the different metrics over the 30 datasets using the Nemenyi test. Methods not significantly different ($\alpha$=0.05) are connected together.

the distribution of their prediction accuracy over the 100 executions. Figure 6.3 presents the range of the prediction accuracy around the median and how the heuristic metrics realize robust and stable predictions, with less number of internal classifiers, for a set of representative datasets $\{D_1, D_3, D_4, D_6, D_9, D_{19}, D_{25}, D_{28}\}$. The conclusion is that the ordering metrics are more effective to select a promising subensemble and their behavior reduces the performance variance.

### 6.3.8 Efficiency Analysis

As demonstrated in [23], the efficiency of reordering metrics can be evaluated according to the following three aspects: the computational cost to extract the pruned subensemble, the required memory space to store the pruned ensembles, and the classification speed. While, some steps can be performed in parallel: the generation of the initial pool of classifiers, and the retrieving of classification decisions from the selected classifiers.

**To answer $Q_9$**, space and time complexities of the different heuristic metrics are summarized in Table 6.9 in terms of $T$, $N$, $M$. The memory requirements are estimated assuming that the decisions of the classifiers are stored in a matrix of size $N \times T$. For large datasets, it might be difficult to store this matrix in memory. In such a case, the whole matrix can be stored to a secondary memory device like the hard disk [23]. This would reduce the memory requirements to $\mathcal{O}(N)$, but the required disk access will slow down the classification process.

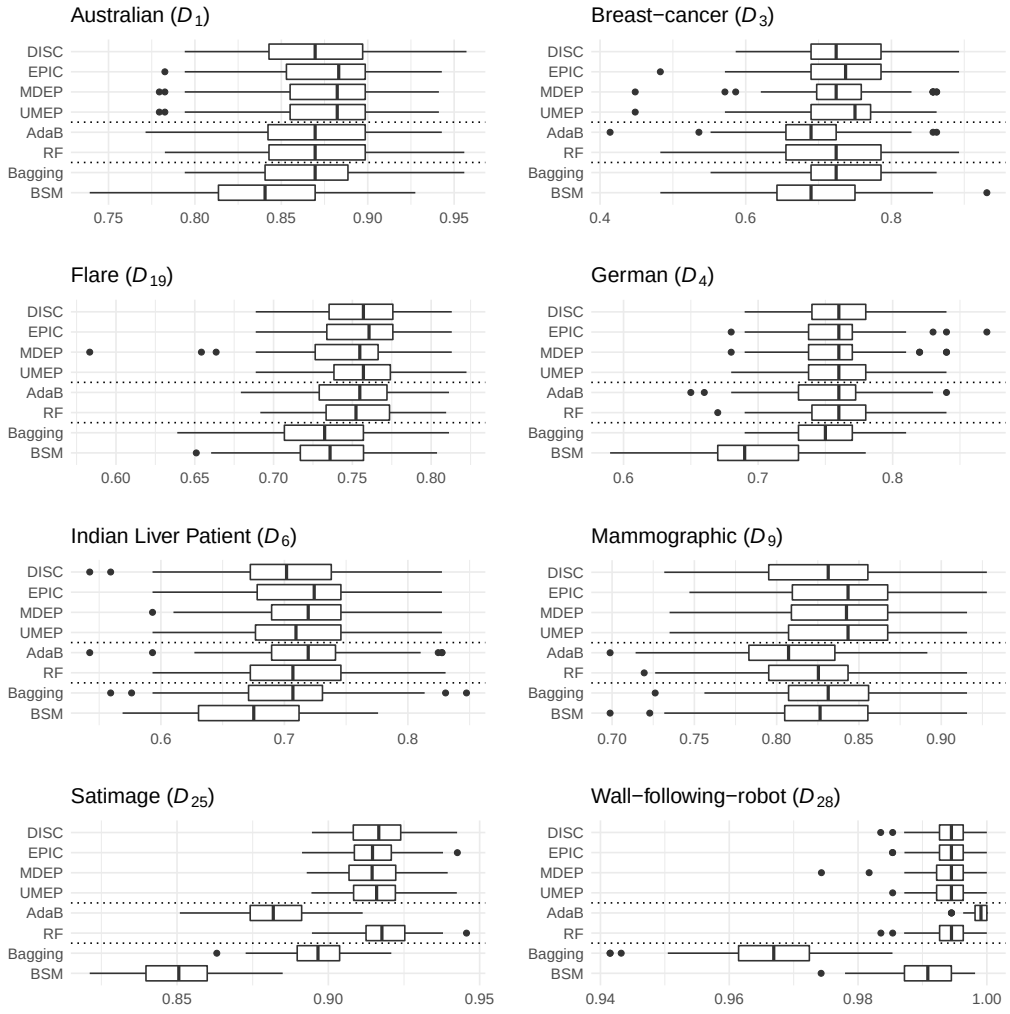


**Figure 6.3:** Distribution of the prediction accuracy.

To empirically investigate how the heuristic measures depend on $T$, a series of experiments over the *SPECTF* dataset are performed. Table 6.10 presents the execution time of several heterogeneous classifiers with initial bagging of 25, 51, 75, 101, 151, and 201. The values of the remaining parameters are $M = 2$, $N_{train} = 314$, $D_{pr} = N_{train}$, $P = 20\%$. The times are averaged over 100 executions in Intel(R) Core(TM) i5-7200 CPU @ 2.50GHZ with 8 GB of RAM.

The results show that UMEP is the fastest method because the rank is calculated based on the classifier's correct classified samples from the pruning set $D_{pr}$.

**Table 6.9:** Space and time complexities of different metrics.

| Metric | Space complexity | Time complexity |
|---|---|---|
| RE | $\mathcal{O}(N \cdot T + M)$ | $\mathcal{O}(T^2 \cdot N \cdot M)$ |
| CC | $\mathcal{O}(N \cdot T + M)$ | $\mathcal{O}(T^2 \cdot N)$ |
| MDSQ | $\mathcal{O}(N \cdot T)$ | $\mathcal{O}(T^2 \cdot N)$ |
| MRMR | $\mathcal{O}(N \cdot T + M^2)$ | $\mathcal{O}(T^2 \cdot N \cdot M)$ |
| DISC | $\mathcal{O}(N \cdot T + M^2)$ | $\mathcal{O}(T^2 \cdot N \cdot M)$ |
| EPIC | $\mathcal{O}(N \cdot T + M)$ | $\mathcal{O}(T \cdot N + T \cdot \log(T))$ |
| UMEP | $\mathcal{O}(N \cdot T)$ | $\mathcal{O}(T \cdot N + T \cdot \log(T))$ |
| MDEP | $\mathcal{O}(N \cdot T + M)$ | $\mathcal{O}(T \cdot N + T \cdot \log(T))$ |

**Table 6.10:** Average execution time in seconds for the *SPECTF* dataset.

| Metric \ $T$ | 25 | 51 | 75 | 101 | 151 | 201 |
|---|---|---|---|---|---|---|
| RE | 0.09 | 0.26 | 0.78 | 1.48 | 3.95 | 5.91 |
| CC | 0.06 | 0.26 | 0.68 | 1.20 | 2.85 | 4.87 |
| MDSQ | 0.05 | 0.19 | 0.32 | 0.48 | 1.13 | 2.96 |
| MRMR | 0.10 | 0.91 | 1.10 | 2.83 | 4.09 | 7.66 |
| DISC | 0.10 | 0.58 | 1.43 | 2.17 | 3.64 | 7.01 |
| EPIC | 0.01 | 0.03 | 0.04 | 0.05 | 0.09 | 0.11 |
| UMEP | 0.01 | 0.02 | 0.03 | 0.04 | 0.07 | 0.10 |
| MDEP | 0.02 | 0.04 | 0.05 | 0.07 | 0.12 | 0.15 |

Additionally, the computation time of EPIC and MDEP is also rather fast with an increasing complexity approximately log-linear with respect to $T$. While, the execution time of RE, CC, MDSQ, MRMR, and DISC is quadratic in $T$. Both MRMR and DISC are much slower, as the selection of one more classifier is conditioned by more internal calculations.

Besides the efficacy of the ensemble pruning metrics, other main benefits concern the efficiency as; storage requirements and the classification speed. The booked memory space for complete bagging will be released to store the pruned subensemble. While the classification speed depends on both (1) the size of the pruned ensemble and (2) the complexity of the base classifiers.

### 6.3.9 Exploration of the Research Questions

In this part, we determine whether the research questions of the third objective are answered or not.

***($Q_5$)*** *The initial classifier pool size and the required subensemble size* affect the performance of the pruning metrics. In general, as the pool size $T$ increases the possibility of finding better subensembles will increase. However, this adds more computational complexity to the pruning process. In addition, it is so difficult to identify a fixed subensemble size $P$ for each pool size $T$, upon which a higher predictive performance can be guaranteed.

***($Q_6$)*** *The pruning metrics can be affected by the individual classifier type*. Different classifier types produce different decisions and affect the majority voting from the pool. Experimentally, the predictive performance of the identified subensemble, via the pruning metrics, is not static and can be affected based on what we use as similar or different classifier types.

***($Q_7$)*** *The efficacy of the pruning metrics could be affected by binary and multiclass datasets*. Experimentally, the rank of the pruning metrics differs upon the classification task. For the investigated binary datasets, the best ranks were scored by MDEP, DISC, EPIC, RE, and UMEP. In contrast, for the investigated multiclass datasets, the best ranks were scored by DISC, EPIC, UMEP, CC, and MRMR. Our explanation for that, the complexity of an individual's decision will increase with the number of classes, and different decisions will be encountered. Regarding that, the capability of each pruning metric could be different.

***($Q_8$)*** *The pruning metrics are effective to reduce the performance variance*. Experimentally, the pruning metrics are capable of selecting non-random subensembles. This was confirmed via the distribution of the predictive accuracy of the pruned ensembles around the median value, to be better than original ensembles.

***($Q_9$)*** *The efficiency of the pruning metrics differs in terms of time and space complexities*. For space complexity, MDSQ and UMEP require less memory space to operate. For time complexity, three metrics have the lowest computational cost which is linear with $T$ (UMEP, EPIC, and MDEP). In contrast, the other five metrics (RE, CC, MDSQ, MRMR, DISC) perform at a larger computational cost which is quadratic in $T$.

## 6.4 Conclusions

Multiple classifier systems are superior to any random single classifier. However, three main defects are reported for those systems; (1) A large pool of classifiers should be built, (2) A Large memory space should be available to store those models, and (3) A large classification time will be consumed to combine multi-decisions. To alleviate these drawbacks, this chapter discussed the concept and the benefits of thinning, pruning, selecting a subset of classifiers. Effective, fast, and implementable heuristic metrics are analyzed to reorder the classifier's position in the generated random bagging. The main conclusions from this study are highlighted as follows:

- The accuracy from ordering metrics is affected by both, the original number of classifiers and the required percentage for the selected subensemble.
- The heuristic metrics with small size bagged ensembles can easily outperform the large size ensembles.
- The performance of heuristic methods can keep on improving regardless of the degradation in the performance of the bagged ensembles.
- The inclusion of different classifiers is more promising to create diversity and complementary in the subensemble than similar classifiers.
- For binary datasets, most of the investigated metrics are significantly outperforming bagging by 95%, while MDEP is the best among of them.
- The best selected single model from a group of classifiers is the worst strategy for selecting subensemble.
- For multiclass datasets, most of the investigated metrics are significantly outperforming bagging by 95%, while DISC and EPIC are the best.
- For the analysis over the 30 datasets, the three metrics DISC, EPIC, and MDEP are still significantly outperforming bagging by 95%.
- The behavior of heuristic metrics against randomness has been analyzed by displaying the prediction accuracies of subensembles around the median.
- Regarding efficiency, UMEP is the fastest heuristic metric to select promising subensemble. While MRMR and DISC are too much slower than other investigated methods due to their internal calculations.

- The general analysis over the 30 datasets proves that ordering metrics are comparable with Random Forest and Adaboost according to accuracy, but they are better regarding the ensemble size.

In summary, ordering based pruning metrics are more effective and efficient to outperform bagging ensembles. The subensemble that minimizes/maximizes a predefined generalization performance criterion is located easily. Analysis of removing the similarity, as in MRMR, makes the usual aggregation of majority voting no longer favorable since the majority has now been significantly reduced. Among the heuristic metrics, UMEP, EPIC, and MDEP have the lowest computational cost which is linear with the size of the initial pool of classifiers. The other investigated metrics are good, but with larger computational cost (quadratic in $T$).

Regarding future research, the concept of uncorrelated error is so important to alleviate the consolidated error. Regarding that, the subensemble could be boosted if the pruning metrics incorporate the uncorrelated error

*Everything is theoretically impossible, until it is done.*

Robert A. Heinlein

CHAPTER

# 7

# Conclusions and Future Work

In this chapter, we summarize the main contributions, limitations, and future lines of research resulted from this thesis. First, a consolidated view of results and contributions is presented in Section 7.1. Afterward, the limitations of this work are identified in Section 7.2. Finally, the future research lines from this thesis are exposed in Section 7.3.

## 7.1 Summary of Contributions

Machine learning is an artificial intelligence technology that gives systems the ability to learn and develop from experience automatically without being programmed specifically. The primary aim is to help computers learn automatically without human intervention or assistance. This field still receiving great interest due to the growth of data, computing power, and statistical algorithms. While data mining is a closely related topic, aims to discover useful, valid, understandable, and unexpected patterns from data. However, each learning algorithm discovers the pattern from a different perspective. Regarding that, ensemble data mining has been proved as an optimal strategy to aggregate and combine several learning algorithms

together. For the classification task, the aggregated models are well-known with the name of multiple classifier systems (MCS). The main objective of the thesis is to enhance the generation and the integration of MCS via soft computing techniques.

In the context discussed above, MCS are hybrid intelligent systems with the potentiality to cope with ambiguity, uncertainty, and complex problems. While soft computing methods support intelligent control, nonlinear programming, optimization, and decision making. Soft computing methods exploit the tolerance for imprecision, partial truth, and uncertainty to achieve tractability, robustness, low cost solution. With the human mind as a role model, soft computing enables solutions for problems that may be either unsolvable or just too time-consuming. This could be particularly helpful to optimize the design and the integration of the complex MCS.

The research developed in this dissertation has contributed to the enhancement of MCS. The experiments have been conducted on popular benchmark datasets. In chapter 2, we presented a nice taxonomy for MCS. Afterward, we have discussed the importance of diversity and how we can measure it through pairwise and non-pairwise metrics. Furthermore, the concept of error diversity is presented to alleviate the consolidated error. Finally, the importance of soft computing is highlighted within the area of MCS. In chapter 3, we have reviewed the state-of-the-art classifier ensembles, this included; bagging-like ensembles, boosting, and gradient boosting ensembles. The presented algorithms were compared together in terms of how to promote diversity, and the type of the base model. Afterward, due to the complexity of MCS, metaheuristic algorithms (MA) were specifically applied to better integrate or prune the classifiers set. Finally, from the conducted revision, we identified the gaps and the research questions that we answered in this thesis. Among the chief contributions of this thesis:

Chapter 4, is dedicated to the first objective: *"To build more diverse and highly accurate MCS, only from a reduced portion of the available data"*. The complexity of the classifiers ensemble increases positively with the size of training data. To reduce this complexity, IS techniques could be used as a preliminary step to reduce the training data-size. First, the border noisy samples are removed via intelligent data sampling. Hence, pure, reduced, and informative data samples can be obtained to train individual classifiers quickly and correctly. Second, the proposed

MCS, Section 4.3.2, considers two strategies to promote diversity; data manipulation and algorithm manipulation techniques. For data manipulation, the bagging and the random feature selection are applied to the reduced dataset from phase one. While, for algorithmic manipulation, the diversity is promoted via generating heterogeneous classifiers. Finally, SI algorithms were incorporated to enhance MCS predictivity. Where, a weighted voting schema is optimized via MFO, GWO, and WOA algorithms to enhance the fusion of multiple decisions. The novelty of this contribution is the intersection between three computational intelligence paradigms: instance selection, ensemble learning, and swarm intelligence. The effect of IS and SI on the generalization performance of MCS has been analyzed and discussed in detail in Sections 4.4.3, and 4.4.5, respectively. With the conclusion, IS proved its effectiveness to reduce the training data-size of 17 datasets by more than 25%. AllKNN, as IS technique, did not prove its capability to capture the integrity from the whole data, where RFSM outperform RFCOM in only 4 out of 25 datasets. The mistake of IS to capture informative samples has been compensated via our proposal. The proposed MCS with a simple combination function, majority voting, outperforms RFCOM in 8 out of 25 datasets. While, a great improvement has been achieved via SI, weighted voting strategy, to outperform RFCOM in 14 out of 25 datasets. Concluding, the predetermined first objective is partly achieved due to the limited performance of the IS method.

Chapter 5, is dedicated to the second objective: *"Increasing the efficiency of MCS and going beyond what can be achieved from ensemble pruning methods"*. A framework has been proposed to benefit from the power of instance selection and ensemble selection simultaneously. In relation to that, IS methods can be applied as a kind of data preprocessing to clean and eliminate inconsistent data. While ensemble pruning is the strategy by which a small-size ensemble can be selected without affecting the general performance of the original ensemble. Hence, the computational resources can be saved, and the testing time can be accelerated by depending on some classifiers instead of all. Ensemble pruning is the second component, that was integrated into the framework, Section 5.3, to elevate its performance. A guided search-based MCS pruning has been proposed to consider both the classifier's accuracy and ensemble diversity. First, two ordering-based pruning metrics have been applied, where each of them returns a set of classifiers. The suggested

classifier set from each metric is merged together to form a new subensemble to be further searched via metaheuristic methods. The proposal successfully solved the parameter tuning challenges, the alpha parameter in Equation 5.5, which is part of a novel and recent pruning metric, MDEP [36]. The limited accuracy of MCS pruning metrics has been maximized via the proposed guided search-based pruning. This has been proved and clarified as in Figure 5.4, and according to the statistical analysis, Section 5.4.4. In this work, small-size ensembles with training on fewer samples could outperform significantly the large-size ensembles which use the whole available training data. Concluding, the predetermined second objective is totally achieved.

Chapter 6, is dedicated to the third objective: *"Grouping and analyzing fast and accurate heuristic metrics for MCS pruning"*. Ensemble pruning strategies are one of the hot topics to gain efficient and effective ensembles. The efficiency could be reached via forming small-size ensembles with their impact; to consume short memory space, reduce the communication cost of the distributed models, and to accelerate the prediction time. While the efficacy could be achieved via building trustable models with a high level of prediction accuracy. We applied a fast and accurate strategy to work with bagging ensembles via ranking the ensemble members. This pruning strategy is known as "ordering-based pruning" to identify the best subset of classifiers for early aggregation. The identification of this subset is mainly controlled through a heuristic measurement to optimize the augmented subensemble. In this chapter, greedy search methods and group-based ranking have been considered to reorder the pool of classifiers. Since the great analysis by Martínez-Muñoz et al. [23] in 2009, several heuristic metrics belonging to this category were proposed [24, 31, 32, 36] with no existence comparison between them. Regarding that, our proposal was conducted to solve that gap. The conclusions from this analysis proved that the efficacy of those metrics is affected by the original ensemble size, the required subensemble size, the kind of individual classifiers, and the number of classes. Furthermore, the investigated metrics realize robust and stable predictions, this is analyzed via the range of their prediction accuracy around the median as shown in Figure 6.3. Finally, the computational cost of some metrics is linear with the initial pool size $T$, while other metrics have a

larger computational cost, which is quadratic in $T$. Concluding, the predetermined third objective is totally achieved.

## 7.2 Limitations

In relation to the contributions summarised above, this thesis also presents limitations that need to be mentioned. In this section, we highlight some aspects that are a barrier to better improvement.

- In Chapter 4, the proposed MCS uses reduced data for training. However, we cannot guarantee whether or not the reduced data via IS could keep the integrity from the original data. In some cases, the reduced data miss informative samples which leads to bad training. This has been analyzed and clarified via the performance on $D_{14}$, $D_{19}$ from Table 4.3.

- In Chapter 4, the proposed approach is characterized by a relatively high computational complexity. Therefore, it is not suitable for *online* learning, e.g., in the case of nonstationary data classification streams, namely, when the *concept drift* phenomenon can occur. Instead, it can be a suitable alternative if the training time is not a critical parameter from the application perspective.

- In Chapter 5, the guided search-based pruning identifies a subensemble with higher predictive performance, but the other pruning metrics (EPIC, UMEP, MDEP) identify a thinner/small-size subensemble. This is clarified by the statistical analysis in Table 5.7.

- The presented work in this thesis has not been scaled up to prove its suitability for big datasets. In addition, the investigated work did not consider problems with certain properties, like imbalanced class labels.

## 7.3 Future works

As the limitations of our study indicate, there are numerous ways to beneficially extend this line of research in the future. They range from improving the training

phase of MCS to pruning and combining MCS efficiently.

1. The proposed MCS is heterogeneous, containing different models, with the aim to promote diversity. While the identification of classifiers to be included in classifier ensembles remains a key issue for predictive performance. In [198], the classifiers types to be consolidated are selected manually via the majority voting error and forward search. Experimentally, the set of classifiers solving a task could not be effective for another task [92]. The inclusion of weak classifiers could degrade the general performance unless if they create complement decisions for particular samples, this is too complex and need to be further discovered and analyzed via tailored metrics to determine the individual classifier type.

2. In Chapters 4 and 5, AllKNN [220], as a training set selection, has been applied due to its reasonable selection time, reduction rate, and limited-accuracy over test [212]. An interesting research line would be to evaluate the performance of other training set selection algorithms. Particularly, how each reduction method could add to the performance of MCS. In addition, the concept of instance selection had been widely used with the support vector machine classifier [62, 246] to alleviate its computational difficulties when dealing with huge amounts of data. In connection with that, a Pareto-based SVM ensemble has been proposed in [247] to optimize the size of the training set and the classification performance attained by the selection of the instances. Thus, the consideration of those modeling approaches could potentially lead to new promising versions for both homogeneous and heterogeneous MCS.

3. In Chapter 5, We have identified that both the classifier's accuracy and the ensemble diversity are crucial for MCS pruning. Our proposed schema is so simple and innovative. The computational cost of our solution is light, due to the usage of ordering metrics, in comparison to evolutionary algorithms. While multi-objective optimization could be further applied to tune the calibration between the subensemble size and the subensemble accuracy. Examples of recent works in this area [248].

4. The concept of uncorrelated error, Section 2.4.3.3, is so important to alleviate the consolidated error. Regarding that, the weights of Equation (2.18) can be optimized by evolutionary algorithms. Furthermore, the subensemble could be better located if the concept of uncorrelated error can be incorporated in the investigated metrics of Chapter 6.

5. In Chapter 4, the SI-based weighted voting schema considers the abstract predictions from the individual classifiers. In contrast, the class probability distribution carries information that should not be ignored [151]. For example, if we take the votes according to the class labels, then the incorrect outputs of the mistaken members will be amplified into wrong votes. A future research line could be interesting to apply SI for the class probability distribution. Furthermore, transforming the outputs of ensemble members into class labels before considering their soft votes, leads to destroying the real distribution and losing useful information [249]. Besides, the usage of soft voting could be more efficient for ensemble pruning via decreasing the probability that a tie occurs, i.e in majority voting.

6. In [22], the authors recommended using the classification confidence and the weights of the base classifiers to define the ensemble margin. They used a homogeneous ensemble of SVM, and the classification confidence for each classifier is calculated in terms of the distance between sample $\mathbf{x}_i$ and the hyperplane. However, it is not clear how to measure the classification confidence for different classifier types, an interesting research line could be to extend their proposal to adapt to a heterogeneous ensemble. In addition, in the same article, the weighted voting of the pruned ensemble proved superior results, in terms of accuracy, over the majority voting of the pruned ensemble. Regarding that, the investigated metrics of Chapters 5 and 6 could be further enhanced.

By taking these issues into account, the predictive performance of MCS scheme could be further enhanced.

# Bibliography


[1] L. Torgo, *Data mining with R: learning with case studies*. CRC press, 2016.

[2] D. J. Hand and N. M. Adams, "Data mining," *Wiley StatsRef: Statistics Reference Online*, pp. 1–7, 2014.

[3] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[4] ——, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[5] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[6] R. Polikar, "Ensemble learning," in *Ensemble machine learning*. Springer, 2012, pp. 1–34.

[7] T. G. Dietterich *et al.*, "Ensemble learning," *The handbook of brain theory and neural networks*, vol. 2, pp. 110–125, 2002.

[8] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999.

[9] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[10] N. C. Oza and S. Russell, *Online ensemble learning*. University of California, Berkeley, 2001.

[11] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[12] C. Zhang and Y. Ma, *Ensemble machine learning: methods and applications*. Springer, 2012.

[13] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and systems magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[14] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010.

[15] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014.

[16] M. Woźniak, M. Graña, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Information Fusion*, vol. 16, pp. 3–17, 2014.

[17] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/-variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[18] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[19] S. García, J. Luengo, and F. Herrera, *Data preprocessing in data mining*. Springer, 2015.

[20] R. Mousavi and M. Eftekhari, "A new ensemble learning methodology based on hybridization of classifier ensemble selection approaches," *Applied Soft Computing*, vol. 37, pp. 652–666, 2015.

[21] B. Krawczyk and M. Woźniak, "Untrained weighted classifier combination with embedded ensemble pruning," *Neurocomputing*, vol. 196, pp. 14–22, 2016.

[22] L. Li, Q. Hu, X. Wu, and D. Yu, "Exploration of classification confidence in ensemble learning," *Pattern recognition*, vol. 47, no. 9, pp. 3120–3131, 2014.

[23] G. Martínez-Muñoz, D. Hernández-Lobato, and A. Suárez, "An analysis of ensemble pruning techniques based on ordered aggregation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 245–259, 2008.

[24] L. Guo and S. Boukir, "Margin-based ordered aggregation for ensemble pruning," *Pattern Recognition Letters*, vol. 34, no. 6, pp. 603–609, 2013.

[25] H. Zhou, X. Zhao, and X. Wang, "An effective ensemble pruning algorithm based on frequent patterns," *Knowledge-Based Systems*, vol. 56, pp. 79–85, 2014.

[26] A. Onan, S. Korukoğlu, and H. Bulut, "A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification," *Information Processing & Management*, vol. 53, no. 4, pp. 814–833, 2017.

[27] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. D. Sousa, "Ensemble approaches for regression: A survey," *Acm computing surveys (csur)*, vol. 45, no. 1, pp. 1–40, 2012.

[28] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "Dynamic classifier selection: Recent advances and perspectives," *Information Fusion*, vol. 41, pp. 195–216, 2018.

[29] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[30] R. Diao, F. Chao, T. Peng, N. Snooke, and Q. Shen, "Feature selection inspired classifier ensemble reduction," *IEEE transactions on cybernetics*, vol. 44, no. 8, pp. 1259–1268, 2013.

[31] J. Cao, W. Li, C. Ma, and Z. Tao, "Optimizing multi-sensor deployment via ensemble pruning for wearable activity recognition," *Information Fusion*, vol. 41, pp. 68–79, 2018.

[32] Z. Lu, X. Wu, X. Zhu, and J. Bongard, "Ensemble pruning via individual contribution ordering," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 871–880.

[33] C. Tamon and J. Xiang, "On the boosting pruning problem," in *European conference on machine learning*. Springer, 2000, pp. 404–412.

[34] G. Tsoumakas, I. Partalas, and I. Vlahavas, "An ensemble pruning primer," in *Applications of supervised and unsupervised ensemble methods*. Springer, 2009, pp. 1–13.

[35] G. Martınez-Munoz and A. Suárez, "Aggregation ordering in bagging," in *Proc. of the IASTED International Conference on Artificial Intelligence and Applications*. Citeseer, 2004, pp. 258–263.

[36] H. Guo, H. Liu, R. Li, C. Wu, Y. Guo, and M. Xu, "Margin & diversity based ordering ensemble pruning," *Neurocomputing*, vol. 275, pp. 237–246, 2018.

[37] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *ICML*, vol. 97. Citeseer, 1997, pp. 211–218.

[38] G. D. Cavalcanti, L. S. Oliveira, T. J. Moura, and G. V. Carvalho, "Combining diversity measures for ensemble pruning," *Pattern Recognition Letters*, vol. 74, pp. 38–45, 2016.

[39] B. Lantz, *Machine learning with R*. Packt publishing ltd, 2013.

[40] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[41] R. Nisbet, J. Elder, and G. Miner, *Handbook of statistical analysis and data mining applications*. Academic Press, 2009.

[42] J. A. Adebayo *et al.*, "Fairml: Toolbox for diagnosing bias in predictive modeling," Ph.D. dissertation, Massachusetts Institute of Technology, 2016.

[43] M. T. Ribeiro, S. Singh, and C. Guestrin, "Model-agnostic interpretability of machine learning," *arXiv preprint arXiv:1606.05386*, 2016.

[44] R. Konig, U. Johansson, and L. Niklasson, "G-rex: A versatile framework for evolutionary data mining," in *2008 IEEE International Conference on Data Mining Workshops*. IEEE, 2008, pp. 971–974.

[45] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in neural information processing systems*, 2017, pp. 4765–4774.

[46] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *arXiv preprint arXiv:1708.08296*, 2017.

[47] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, pp. 1–15, 2018.

[48] P. Cortez and M. J. Embrechts, "Opening black box data mining models using sensitivity analysis," in *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE, 2011, pp. 341–348.

[49] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation," *Journal of Computational and Graphical Statistics*, vol. 24, no. 1, pp. 44–65, 2015.

[50] E. Rahm and H. H. Do, "Data cleaning: Problems and current approaches," *IEEE Data Eng. Bull.*, vol. 23, no. 4, pp. 3–13, 2000.

[51] M. Lenzerini, "Data integration: A theoretical perspective," in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2002, pp. 233–246.

[52] A. Doan, A. Halevy, and Z. Ives, *Principles of data integration*. Elsevier, 2012.

[53] B. Pochon and P. Scotton, "Method and apparatus for data normalization," Apr. 8 2008, uS Patent 7,356,599.

[54] D. Pyle, *Data preparation for data mining*. morgan kaufmann, 1999.

[55] S. v. Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," *Journal of statistical software*, pp. 1–68, 2010.

[56] H. Wang and S. Wang, "Mining incomplete survey data through classification," *Knowledge and information systems*, vol. 24, no. 2, pp. 221–233, 2010.

[57] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.

[58] M. Dash and H. Liu, "Feature selection for classification," *Intelligent data analysis*, vol. 1, no. 3, pp. 131–156, 1997.

[59] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[60] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. Kittler, "A review of instance selection methods," *Artificial Intelligence Review*, vol. 34, no. 2, pp. 133–143, 2010.

[61] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," *Knowledge and information systems*, vol. 35, no. 2, pp. 249–283, 2013.

[62] C. Liu, W. Wang, M. Wang, F. Lv, and M. Konan, "An efficient instance selection algorithm to reconstruct training set for support vector machine," *Knowledge-Based Systems*, vol. 116, pp. 58–73, 2017.

[63] Y. Ding, L. Zhu, X. Zhang, and H. Ding, "A full-discretization method for prediction of milling stability," *International Journal of Machine Tools and Manufacture*, vol. 50, no. 5, pp. 502–509, 2010.

[64] S. Chakrabarti, M. Ester, U. Fayyad, J. Gehrke, J. Han, S. Morishita, G. Piatetsky-Shapiro, and W. Wang, "Data mining curriculum: A proposal (version 1.0)," *Intensive Working Group of ACM SIGKDD Curriculum Committee*, vol. 140, pp. 1–10, 2006.

[65] C. Clifton, "Encyclopædia britannica: definition of data mining," *Retrieved on*, vol. 9, no. 12, p. 2010, 2010.

[66] D. H. Fisher, M. J. Pazzani, and P. Langley, *Concept formation: Knowledge and experience in unsupervised learning*. Morgan Kaufmann, 2014.

[67] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[68] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1, no. 10.

[69] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[70] M. M. Saritas and A. Yasar, "Performance analysis of ann and naive bayes classification algorithm for data classification," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 7, no. 2, pp. 88–91, 2019.

[71] N. Lopes and B. Ribeiro, "On the impact of distance metrics in instance-based learning algorithms," in *Iberian Conference on Pattern Recognition and Image Analysis*. Springer, 2015, pp. 48–56.

[72] J. Nalepa and M. Kawulok, "Selecting training sets for support vector machines: a review," *Artificial Intelligence Review*, vol. 52, no. 2, pp. 857–900, 2019.

[73] J. Fürnkranz, D. Gamberger, and N. Lavrač, *Foundations of rule learning*. Springer Science & Business Media, 2012.

[74] F. Galbusera, G. Casaroli, and T. Bassani, "Artificial intelligence and machine learning in spine research," *JOR spine*, vol. 2, no. 1, p. e1044, 2019.

[75] Z.-H. Zhou and W. Tang, "Clusterer ensemble," *Knowledge-Based Systems*, vol. 19, no. 1, pp. 77–83, 2006.

[76] S. Vega-Pons and J. Ruiz-Shulcloper, "A survey of clustering ensemble algorithms," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 03, pp. 337–372, 2011.

[77] A. Cornuéjols, C. Wemmert, P. Gançarski, and Y. Bennani, "Collaborative clustering: Why, when, what and how," *Information Fusion*, vol. 39, pp. 81–95, 2018.

[78] Z. Zhao, K. G. Mehrotra, and C. K. Mohan, "Ensemble algorithms for unsupervised anomaly detection," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2015, pp. 514–525.

[79] Y. Ren, P. Suganthan, and N. Srikanth, "Ensemble methods for wind and solar power forecasting—a state-of-the-art review," *Renewable and Sustainable Energy Reviews*, vol. 50, pp. 82–91, 2015.

[80] J. A. Benediktsson and P. H. Swain, "Consensus theoretic classification methods," *IEEE transactions on Systems, Man, and Cybernetics*, vol. 22, no. 4, pp. 688–704, 1992.

[81] A. I. Naimi and L. B. Balzer, "Stacked generalization: an introduction to super learning," *European journal of epidemiology*, vol. 33, no. 5, pp. 459–464, 2018.

[82] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *The 2011 international joint conference on neural networks*. IEEE, 2011, pp. 1918–1921.

[83] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[84] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[85] Z. Yu, L. Li, J. Liu, and G. Han, "Hybrid adaptive classifier ensemble," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 177–190, 2014.

[86] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 10, pp. 1619–1630, 2006.

[87] D. Ruta and B. Gabrys, "Classifier selection for majority voting," *Information fusion*, vol. 6, no. 1, pp. 63–81, 2005.

[88] C. Catal and M. Nangir, "A sentiment classification model based on multiple classifiers," *Applied Soft Computing*, vol. 50, pp. 135–141, 2017.

[89] Z. Liu, Q. Pan, J. Dezert, J.-W. Han, and Y. He, "Classifier fusion with contextual reliability evaluation," *IEEE transactions on cybernetics*, vol. 48, no. 5, pp. 1605–1618, 2017.

[90] M. A. Tahir, J. Kittler, and A. Bouridane, "Multilabel classification using heterogeneous ensemble of multi-label classifiers," *Pattern Recognition Letters*, vol. 33, no. 5, pp. 513–523, 2012.

[91] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *The journal of machine learning research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[92] D. H. Wolpert, "The supervised learning no-free-lunch theorems," in *Soft computing and industry*. Springer, 2002, pp. 25–42.

[93] P. Ksieniewicz, B. Krawczyk, and M. Woźniak, "Ensemble of extreme learning machines with trained classifier combination and statistical features for hyperspectral data," *Neurocomputing*, vol. 271, pp. 28–37, 2018.

[94] S. González, S. García, J. Del Ser, L. Rokach, and F. Herrera, "A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities," *Information Fusion*, vol. 64, pp. 205–237, 2020.

[95] A. S. Britto Jr, R. Sabourin, and L. E. Oliveira, "Dynamic selection of classifiers—a comprehensive review," *Pattern recognition*, vol. 47, no. 11, pp. 3665–3680, 2014.

[96] M. Re and G. Valentini, "1 ensemble methods: a review 3," 2012.

[97] M. Sewell, "Ensemble learning," *RN*, vol. 11, no. 02, 2008.

[98] L. Rokach, "Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography," *Computational statistics & data analysis*, vol. 53, no. 12, pp. 4046–4072, 2009.

[99] X. Hu, "Using rough sets theory and database operations to construct a good ensemble of classifiers for data mining applications," in *Proceedings 2001 IEEE International Conference on Data Mining*. IEEE, 2001, pp. 233–240.

[100] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[101] D. W. Opitz and J. W. Shavlik, "Generating accurate and diverse members of a neural-network ensemble," in *Advances in neural information processing systems*, 1996, pp. 535–541.

[102] K. Woods, W. P. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE transactions on pattern analysis and machine intelligence*, vol. 19, no. 4, pp. 405–410, 1997.

[103] O. Maimon and L. Rokach, "Decomposition methodology for knowledge discovery and data mining," in *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 981–1003.

[104] L. Rokach, O. Maimon, and O. Arad, "Improving supervised learning by sample decomposition," *International Journal of Computational Intelligence and Applications*, vol. 5, no. 01, pp. 37–53, 2005.

[105] S. J. Nowlan and G. E. Hinton, "Evaluation of adaptive mixtures of competing experts," in *Advances in neural information processing systems*, 1991, pp. 774–780.

[106] S. Cohen, L. Rokach, and O. Maimon, "Decision-tree instance-space decomposition with grouped gain-ratio," *Information Sciences*, vol. 177, no. 17, pp. 3592–3612, 2007.

[107] K. Tumer and N. C. Oza, "Input decimated ensembles," *Pattern Analysis & Applications*, vol. 6, no. 1, pp. 65–77, 2003.

[108] R. Bryll, R. Gutierrez-Osuna, and F. Quek, "Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets," *Pattern recognition*, vol. 36, no. 6, pp. 1291–1302, 2003.

[109] M. Skurichina and R. P. Duin, "Bagging for linear classifiers," *Pattern Recognition*, vol. 31, no. 7, pp. 909–930, 1998.

[110] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[111] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, vol. 40, no. 2, pp. 139–157, 2000.

[112] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 161–168.

[113] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for adaboost," *Machine learning*, vol. 42, no. 3, pp. 287–320, 2001.

[114] R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka, "Efficient classification for multiclass problems using modular neural networks," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 117–124, 1995.

[115] B.-L. Lu and M. Ito, "Task decomposition and module combination based on class relations: a modular neural network for pattern classification," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1244–1256, 1999.

[116] D. M. Sivalingam, N. PanDian, and J. Ben-Arie, "Minimal classification method with error-correcting codes for multiclass recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 05, pp. 663–680, 2005.

[117] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of artificial intelligence research*, vol. 2, pp. 263–286, 1994.

[118] L. Breiman, "Randomizing outputs to increase prediction accuracy," *Machine Learning*, vol. 40, no. 3, pp. 229–242, 2000.

[119] G. Martínez-Muñoz and A. Suárez, "Switching class labels to generate classification ensembles," *Pattern Recognition*, vol. 38, no. 10, pp. 1483–1494, 2005.

[120] Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm," in *icml*, vol. 96. Citeseer, 1996, pp. 148–156.

[121] D. Tao, X. Tang, X. Li, and X. Wu, "Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 7, pp. 1088–1099, 2006.

[122] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "A comparison of decision tree ensemble creation techniques," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 1, pp. 173–180, 2006.

[123] Y. Zhang, S. Burer, and W. N. Street, "Ensemble pruning via semi-definite programming," *Journal of machine learning research*, vol. 7, no. Jul, pp. 1315–1338, 2006.

[124] H. Ykhlef and D. Bouchaffra, "An efficient ensemble pruning approach based on simple coalitional games," *Information Fusion*, vol. 34, pp. 28–42, 2017.

[125] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "Ensemble diversity measures and their application to thinning," *Information Fusion*, vol. 6, no. 1, pp. 49–62, 2005.

[126] L. Zhang, W. Srisukkham, S. C. Neoh, C. P. Lim, and D. Pandit, "Classifier ensemble reduction using a modified firefly algorithm: An empirical evaluation," *Expert Systems with Applications*, vol. 93, pp. 395–422, 2018.

[127] M. Aksela, "Comparison of classifier selection methods for improving committee performance," in *International Workshop on Multiple Classifier Systems*. Springer, 2003, pp. 84–93.

[128] G. Brown and L. I. Kuncheva, ""good" and "bad" diversity in majority vote ensembles," in *International workshop on multiple classifier systems*. Springer, 2010, pp. 124–133.

[129] E. M. Dos Santos, R. Sabourin, and P. Maupin, "Overfitting cautious selection of classifier ensembles with genetic algorithms," *Information Fusion*, vol. 10, no. 2, pp. 150–162, 2009.

[130] F. Yang, W.-h. Lu, L.-k. Luo, and T. Li, "Margin optimization based pruning for random forest," *Neurocomputing*, vol. 94, pp. 54–63, 2012.

[131] D. Hernández-Lobato, G. Martínez-Muñoz, and A. Suárez, "Statistical instance-based pruning in ensembles of independent classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 364–369, 2008.

[132] M. N. Adnan and M. Z. Islam, "Optimizing the number of trees in a decision forest to discover a subforest with high ensemble accuracy using a genetic algorithm," *Knowledge-Based Systems*, vol. 110, pp. 86–97, 2016.

[133] D. Ruta and B. Gabrys, "Application of the evolutionary algorithms for classifier selection in multiple classifier systems with majority voting," in *International Workshop on Multiple Classifier Systems*. Springer, 2001, pp. 399–408.

[134] P. Zyblewski and M. Woźniak, "Novel clustering-based pruning algorithms," *Pattern Analysis and Applications*, pp. 1–10, 2020.

[135] I. Partalas, G. Tsoumakas, and I. Vlahavas, "An ensemble uncertainty aware measure for directed hill climbing ensemble pruning," *Machine Learning*, vol. 81, no. 3, pp. 257–282, 2010.

[136] S. Mao, L. Jiao, L. Xiong, and S. Gou, "Greedy optimization classifiers ensemble based on diversity," *Pattern Recognition*, vol. 44, no. 6, pp. 1245–1261, 2011.

[137] A. H. Ko, R. Sabourin, and A. S. Britto, Jr., "From dynamic classifier selection to dynamic ensemble selection," *Pattern Recognition*, vol. 41, no. 5, pp. 1718 – 1731, 2008.

[138] R. M. Cruz, R. Sabourin, G. D. Cavalcanti, and T. Ing Ren, "Meta-des: A dynamic ensemble selection framework using meta-learning," *Pattern Recognition*, vol. 48, no. 5, pp. 1925 – 1935, 2015.

[139] L. I. Kuncheva, "Clustering-and-selection model for classifier combination," in *KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No. 00TH8516)*, vol. 1. IEEE, 2000, pp. 185–188.

[140] S. Soares, C. H. Antunes, and R. Araújo, "Comparison of a genetic algorithm and simulated annealing for automatic neural network ensemble development," *Neurocomputing*, vol. 121, pp. 498–511, 2013.

[141] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 3, pp. 418–435, 1992.

[142] J. Parker, "Rank and response combination from confusion matrix data," *Information Fusion*, vol. 2, no. 2, pp. 113 – 120, 2001.

[143] L. I. Kuncheva and J. J. Rodríguez, "A weighted voting framework for classifiers ensembles," *Knowledge and Information Systems*, vol. 38, no. 2, pp. 259–275, 2014.

[144] G. Niu, T. Han, B.-S. Yang, and A. C. C. Tan, "Multi-agent decision fusion for motor fault diagnosis," *Mechanical Systems and Signal Processing*, vol. 21, no. 3, pp. 1285–1299, 2007.

[145] L. I. Kuncheva, J. C. Bezdek, and R. P. Duin, "Decision templates for multiple classifier fusion: an experimental comparison," *Pattern Recognition*, 2001.

[146] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.

[147] Y. S. Huang and C. Y. Suen, "A Method of Combining Multiple Experts for the Recognition of Unconstrained Handwritten Numerals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[148] T. kam Ho, J. J. Hull, and S. N. Srihari, "Decision Combination in Multiple Classifier Systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1994.

[149] B. Krawczyk, M. Woźniak, and G. Schaefer, "Cost-sensitive decision tree ensembles for effective imbalanced classification," *Applied Soft Computing*, vol. 14, pp. 554–562, 2014.

[150] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[151] R. Sikora *et al.*, "A modified stacking ensemble machine learning algorithm using genetic algorithms," in *Handbook of Research on Organizational Transformations through Big Data Analytics*, 2015, pp. 43–53.

[152] Z. Cordeiro and G. L. Pappa, "A pso algorithm for improving multi-view classification," in *IEEE Congress of Evolutionary Computation (CEC)*, 2011, pp. 925–932.

[153] A. Kausar, M. Ishtiaq, M. A. Jaffar, and A. M. Mirza, "Optimization of ensemble based decision using pso," in *Proceedings of the World Congress on Engineering*, 2010, pp. 671–676.

[154] D. H. Wolpert, "Stacked generalization," *Neural Networks*, 1992.

[155] A. K. Seewald and J. Fürnkranz, "An evaluation of grading classifiers," in *International symposium on intelligent data analysis*. Springer, 2001, pp. 115–124.

[156] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Information Fusion*, 2008.

[157] T. K. Ho, "Multiple classifier combination: Lessons and next steps," in *Hybrid methods in pattern recognition*. World Scientific, 2002, pp. 171–198.

[158] G. Rogova, "Combining the results of several neural network classifiers," *Neural networks*, vol. 7, no. 5, pp. 777–781, 1994.

[159] D. Ruta and B. Gabrys, "Analysis of the Correlation Between Majority Voting Error and the Diversity Measures in Multiple Classifier Systems," *Soft Computing and Intelligent Systems for Industry*, 2001.

[160] G. Giacinto and F. Roli, "Design of effective neural network ensembles for image classification purposes," *Image and Vision Computing*, vol. 19, no. 9-10, pp. 699–707, 2001.

[161] J. Kim, K. Seo, and K. Chung, "A systematic approach to classifier selection on combining multiple classifiers for handwritten digit recognition," in *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, vol. 2. IEEE, 1997, pp. 459–462.

[162] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine learning*, vol. 51, no. 2, pp. 181–207, 2003.

[163] N. Li, Y. Yu, and Z.-H. Zhou, "Diversity regularized ensemble pruning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 330–345.

[164] M. Aksela and J. Laaksonen, "Using diversity of errors for selecting members of a committee classifier," *Pattern Recognition*, vol. 39, no. 4, pp. 608–623, 2006.

[165] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[166] S. K. Das, A. Kumar, B. Das, and A. Burnwal, "On soft computing techniques in various areas," *Computer Science & Information Technology (CS & IT)*, vol. 3, pp. 59–68, 2013.

[167] A. Konar, *Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain*. CRC press, 2018.

[168] Q. Zhu and A. T. Azar, *Complex system modelling and control through intelligent soft computations*. Springer, 2015, vol. 319.

[169] Y. Yuan and M. J. Shaw, "Induction of fuzzy decision trees," *Fuzzy Sets and systems*, vol. 69, no. 2, pp. 125–139, 1995.

[170] C. Olaru and L. Wehenkel, "A complete fuzzy decision tree technique," *Fuzzy sets and systems*, vol. 138, no. 2, pp. 221–254, 2003.

[171] P. Bonissone, J. M. Cadenas, M. C. Garrido, and R. A. Díaz-Valladares, "A fuzzy random forest," *International Journal of Approximate Reasoning*, vol. 51, no. 7, pp. 729–747, 2010.

[172] T. Wilk and M. Wozniak, "Soft computing methods applied to combination of one-class classifiers," *Neurocomputing*, vol. 75, no. 1, pp. 185–193, 2012.

[173] J. H. Holland, "Adaptation in Natural and Artificial Systems," *Ann Arbor MI University of Michigan Press*, vol. Ann Arbor, p. 183, 1992.

[174] L. I. Kuncheva and L. C. Jain, "Designing classifier fusion systems by genetic algorithms," *IEEE Transactions on Evolutionary computation*, vol. 4, no. 4, pp. 327–336, 2000.

[175] A. Tsymbal, M. Pechenizkiy, and P. Cunningham, "Diversity in search strategies for ensemble feature selection," *Information fusion*, vol. 6, no. 1, pp. 83–98, 2005.

[176] M.-J. Kim and D.-K. Kang, "Ensemble with neural networks for bankruptcy prediction," *Expert systems with applications*, vol. 37, no. 4, pp. 3373–3379, 2010.

[177] F. Plesinger, P. Nejedly, I. Viscor, J. Halamek, and P. Jurak, "Parallel use of a convolutional neural network and bagged tree ensemble for the classification of holter ecg," *Physiological measurement*, vol. 39, no. 9, p. 094002, 2018.

[178] A. E. Hassanien and E. Emary, *Swarm intelligence: principles, advances, and applications*. CRC Press, 2018.

[179] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[180] S. Mirjalili and S. Z. M. Hashim, "A new hybrid psogsa algorithm for function optimization," in *IEEE international conference on computer and information application*, 2010, pp. 374–377.

[181] B. Krawczyk, "One-class classifier ensemble pruning and weighting with firefly algorithm," *Neurocomputing*, vol. 150, pp. 490–500, 2015.

[182] J. Kozak and U. Boryczka, "Multiple boosting in the ant colony decision forest meta-classifier," *Knowledge-Based Systems*, vol. 75, pp. 141–151, 2015.

[183] L. Rokach, "Decision forest: Twenty years of research," *Information Fusion*, vol. 27, pp. 111–125, 2016.

[184] Y. Ren, L. Zhang, and P. N. Suganthan, "Ensemble classification and regression-recent developments, applications and future directions," *IEEE Computational intelligence magazine*, vol. 11, no. 1, pp. 41–53, 2016.

[185] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[186] G. Louppe and P. Geurts, "Ensembles on random patches," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 346–361.

[187] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[188] R. Blaser and P. Fryzlewicz, "Random rotation ensembles," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 126–151, 2016.

[189] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[190] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[191] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[192] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[193] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[194] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[195] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in neural information processing systems*, 2017, pp. 3146–3154.

[196] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: gradient boosting with categorical features support," *arXiv preprint arXiv:1810.11363*, 2018.

[197] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," in *Advances in neural information processing systems*, 2018, pp. 6638–6648.

[198] A. Onan, S. Korukoğlu, and H. Bulut, "A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification," *Expert Systems with Applications*, vol. 62, pp. 1–16, 2016.

[199] Y. Chen and M. L. Wong, "An ant colony optimization approach for stacking ensemble," in *2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC)*. IEEE, 2010, pp. 146–151.

[200] A. Ekbal and S. Saha, "Weighted vote-based classifier ensemble for named entity recognition: A genetic algorithm-based approach," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 10, no. 2, pp. 1–37, 2011.

[201] ——, "A multiobjective simulated annealing approach for classifier ensemble: Named entity recognition in indian languages as case studies," *Expert Systems with Applications*, vol. 38, no. 12, pp. 14 760–14 772, 2011.

[202] A. Peimankar, S. J. Weddell, T. Jalal, and A. C. Lapthorn, "Ensemble classifier selection using multi-objective pso for fault diagnosis of power transformers," in *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, pp. 3622–3629.

[203] E. M. Dos Santos, "Evolutionary algorithms applied to classifier ensemble selection," *XLIV SBPO/XVI CLAIO*, pp. 419–430, 2012.

[204] A. A. Aburomman and M. B. I. Reaz, "A novel svm-knn-pso ensemble method for intrusion detection system," *Applied Soft Computing*, vol. 38, pp. 360–372, 2016.

[205] S. Bashir, U. Qamar, and F. H. Khan, "Bagmoov: A novel ensemble for heart disease prediction bootstrap aggregation with multi-objective optimized voting," *Australasian physical & engineering sciences in medicine*, vol. 38, no. 2, pp. 305–323, 2015.

[206] J. García-Gutiérrez, D. Mateos-García, M. Garcia, and J. C. Riquelme-Santos, "An evolutionary-weighted majority voting and support vector machines applied to contextual classification of lidar and imagery data fusion," *Neurocomputing*, vol. 163, pp. 17–24, 2015.

[207] Y. Zhang, H. Zhang, J. Cai, and B. Yang, "A weighted voting classifier based on differential evolution," in *Abstract and Applied Analysis*, vol. 2014. Hindawi, 2014.

[208] W. Fan, F. Chu, H. Wang, and P. S. Yu, "Pruning and dynamic scheduling of cost-sensitive ensembles," in *AAAI/IAAI*, 2002, pp. 146–151.

[209] I. Partalas, G. Tsoumakas, and I. P. Vlahavas, "Focused ensemble selection: A diversity-based method for greedy ensemble selection." in *ECAI*, 2008, pp. 117–121.

[210] Y. Yang, K. Korb, K. M. Ting, and G. I. Webb, "Ensemble selection for superparent-one-dependence estimators," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2005, pp. 102–112.

[211] H. Liu and H. Motoda, *Instance selection and construction for data mining*. Springer Science & Business Media, 2013, vol. 608.

[212] S. Garcia, J. Derrac, J. R. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 3, pp. 417–435, 2011.

[213] G. Wang, J. Sun, J. Ma, K. Xu, and J. Gu, "Sentiment classification: The contribution of ensemble learning," *Decision support systems*, vol. 57, pp. 77–93, 2014.

[214] S. Bashir, U. Qamar, and F. H. Khan, "Heterogeneous classifiers fusion for dynamic breast cancer diagnosis using weighted vote based ensemble," *Quality & Quantity*, vol. 49, no. 5, pp. 2061–2076, 2015.

[215] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.

[216] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in engineering software*, vol. 95, pp. 51–67, 2016.

[217] G. Martínez-Muñoz and A. Suárez, "Pruning in ordered bagging ensembles," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 609–616.

[218] ——, "Using boosting to prune bagging ensembles," *Pattern Recognition Letters*, vol. 28, no. 1, pp. 156–165, 2007.

[219] J. R. Cano, F. Herrera, and M. Lozano, "Evolutionary stratified training set selection for extracting classification rules with trade off precision-interpretability," *Data & Knowledge Engineering*, vol. 60, no. 1, pp. 90–108, 2007.

[220] I. Tomek, "An experiment with the edited nearest-neighbor rule," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 6, no. 6, pp. 448–452, 1976.

[221] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA)-Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[222] G. Jurman, S. Riccadonna, and C. Furlanello, "A comparison of mcc and cen error measures in multi-class prediction," *PloS one*, vol. 7, no. 8, 2012.

[223] M. Wozniak, "Experiments on linear combiners," in *Information technologies in biomedicine*. Springer, 2008, pp. 445–452.

[224] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework." *Journal of Multiple-Valued Logic & Soft Computing*, vol. 17, 2011.

[225] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Computing*, vol. 8, no. 2, pp. 239–287, 2009.

[226] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the american statistical association*, vol. 32, no. 200, pp. 675–701, 1937.

[227] F. Wilcoxon, "Individual comparisons of grouped data by ranking methods," *Journal of economic entomology*, vol. 39, no. 2, pp. 269–270, 1946.

[228] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, pp. 60–88, 2017.

[229] M. P. Sesmero, A. I. Ledezma, and A. Sanchis, "Generating ensembles of heterogeneous classifiers using stacked generalization," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 1, pp. 21–34, 2015.

[230] M. Wozniak and M. Zmyslony, "Combining classifiers using trained fuser—analytical and experimental results," *Neural Network World*, vol. 20, no. 7, p. 925, 2010.

[231] A. J. Sharkey, N. E. Sharkey, U. Gerecke, and G. O. Chandroth, "The "test and select" approach to ensemble combination," in *International Workshop on Multiple Classifier Systems*. Springer, 2000, pp. 30–44.

[232] C. A. Shipp and L. I. Kuncheva, "Relationships between combination methods and measures of diversity in combining classifiers," *Information fusion*, vol. 3, no. 2, pp. 135–148, 2002.

[233] F. Roli and G. Giacinto, "Design of multiple classifier systems," in *Hybrid methods in pattern recognition*. World Scientific, 2002, pp. 199–226.

[234] C. Lin, W. Chen, C. Qiu, Y. Wu, S. Krishnan, and Q. Zou, "Libd3c: ensemble classifiers with a clustering and dynamic selection strategy," *Neurocomputing*, vol. 123, pp. 424–435, 2014.

[235] J. Ghosh and A. Acharya, "Cluster ensembles," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 4, pp. 305–315, 2011.

[236] S. Barak, A. Arjmand, and S. Ortobelli, "Fusion of multiple diverse predictors in stock market," *Information Fusion*, vol. 36, pp. 90–102, 2017.

[237] P. Cortez, *Modern optimization with R*. Springer, 2014.

[238] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary grey wolf optimization approaches for feature selection," *Neurocomputing*, vol. 172, pp. 371–381, 2016.

[239] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[240] C. O. Sakar, O. Kursun, and F. Gurgen, "A feature selection method based on kernel canonical correlation analysis and the minimum redundancy–maximum relevance filter method," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3432–3437, 2012.

[241] A. Unler, A. Murat, and R. B. Chinnam, "mr2pso: A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification," *Information Sciences*, vol. 181, no. 20, pp. 4625–4641, 2011.

[242] J. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[243] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in statistics*. Springer, 1992, pp. 196–202.

[244] S. Garcia and F. Herrera, "An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons," *Journal of machine learning research*, vol. 9, no. Dec, pp. 2677–2694, 2008.

[245] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[246] J. Chen, C. Zhang, X. Xue, and C.-L. Liu, "Fast instance selection for speeding up support vector machines," *Knowledge-Based Systems*, vol. 45, pp. 1–7, 2013.

[247] A. Rosales-Pérez, S. García, J. A. Gonzalez, C. A. C. Coello, and F. Herrera, "An evolutionary multiobjective model and instance selection for support vector machines with pareto-based ensembles," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 6, pp. 863–877, 2017.

[248] S. Fletcher, B. Verma, and M. Zhang, "A non-specialized ensemble classifier using multi-objective optimization," *Neurocomputing*, 2020.

[249] Q. Dai, T. Zhang, and N. Liu, "A new reverse reduce-error ensemble pruning algorithm," *Applied Soft Computing*, vol. 28, pp. 237–249, 2015.