Pós-Graduação em Ciência da Computação

Felipe Nunes Walmsley

An Investigation into the Effects of Label Noise on Dynamic Selection Algorithms

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

Recife
2020

Felipe Nunes Walmsley

An Investigation into the Effects of Label Noise on Dynamic Selection Algorithms

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de Concentração: inteligência computacional

Orientador: George Darmiton da Cunha Cavalcanti
Coorientador: Robert Sabourin

Recife
2020

Catalogação na fonte
Bibliotecária Arabelly Ascoli CRB4-2068

W216i Walmsley, Felipe Nunes
An investigation into the effects of label noise on dynamic selection algorithms / Felipe Nunes Walmsley. – 2020.
117 f.: il. fig., tab.

Orientador: George Darmiton da Cunha Cavalcanti
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn. Ciência da Computação. Recife, 2020.
Inclui referências e apêndices.

1. Métodos de Ensemble. 2. Sistemas de múltiplos classificadores. 3. Seleção dinâmica. 4. Ruído de classe. I. Cavalcanti, George Darmiton da Cunha. (orientador). II. Título.

006.31 CDD (22. ed.) UFPE-CCEN 2020-79

Felipe Nunes Walmsley

"An Investigation into the Effects of Label Noise on Dynamic Selection Algorithms"

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 22/01/2020.

Orientador: George Darmiton da Cunha Cavalcanti

To my mother, whose example I am merely following, and from whom I still have much to learn.

ACKNOWLEDGEMENTS

I'd like to thank my advisors, Prof. George Cavalcanti and Prof. Robert Sabourin, not only for their guidance in academic matters, but also for their patience and understanding when dealing with the more human matters involved in the creation of this work.

I'd also like to thank Dr. Rafael Cruz for his assistance in executing the experiments related to the Fire-DES++ algorithm, and also for just generally helping with issues related to our experimental methodology.

My constant and unending gratitude goes to my family, and particularly, to my mother, for the support and encouragement I have been given during these past twenty years or so of formal education, of which this document is merely the culmination.

I'd like to thank my coworkers and bosses at the Projeto Samsung/CIn-UFPE, for their support and encouragement during the final stretch of this endeavor.

Finally, I'd like to acknowledge the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for the financial support received.

"A good traveler has no fixed plans and is not intent upon arriving. A good artist lets his intuition lead him wherever it wants. A good scientist has freed himself of concepts and keeps his mind open to what is." (LAOZI, 2006)

ABSTRACT

In the literature on classification problems, it is widely discussed how the presence of label noise can bring about severe degradation in performance. Several works have applied Prototype Selection techniques, Ensemble Methods, or both, in an attempt to alleviate this issue. Nevertheless, these methods are not always able to sufficiently counteract the effects of noise. In this work, we investigate the effects of noise on a particular class of Ensemble Methods, that of Dynamic Selection algorithms, and we are especially interested in the behavior of the Fire-DES++ algorithm, a state of the art algorithm which applies the ENN to algorithm to deal with the effects of noise and imbalance. We propose a method which employs multiple Dynamic Selection sets, based on the Bagging-IH algorithm, which we dub Multiple-Set Dynamic Selection (MSDS), in an attempt to supplant the ENN algorithm on the filtering step. We find that almost all methods based on Dynamic Selection are severely affected by the presence of label noise, with the exception of the KNORAU algorithm. We also find that our proposed method can alleviate the issues caused by noise in some specific scenarios.

Keywords: Ensemble Methods. Multiple Classifier Systems. Dynamic Selection. Label Noise. Bagging.

RESUMO

Na literatura de problemas de classificação, é amplamente discutido como a presença de ruído nos rótulos de classe pode acarretar grave degradação na performance. Vários trabalhos aplicam técnicas de Seleção de Protótipos, Métodos de *Ensemble*, ou ambos, em uma tentativa de aliviar esse problema. Não obstante, esses métodos nem sempre são capazes de contrabalançar os efeitos do ruído. Neste trabalho, nós investigamos o efeito do ruído em uma classe em particular de Métodos de *Ensemble*, a classe dos métodos de Seleção Dinâmica, e estamos particularmente interessados no comportamento do algoritmo Fire-DES++, um algoritmo estado da arte que aplica o método *Edited Nearest Neighbors* (ENN) para lidar com os efeitos de ruído e desbalanceamento. Nós propomos um método que emprega múltiplos conjuntos de Seleção Dinâmica, baseado no algoritmo *Bagging-IH*, que nós nomeamos *Multiple-Set Dynamic Selection* (MSDS), em uma tentativa de suplantar o algoritmo ENN no passo de filtragem. Nós observamos que quase todos os métodos baseados em Seleção Dinâmica são fortemente afetados pela presença de ruído, exceto o algoritmo KNORAU. Nós também observamos que, em alguns cenários específicos, o nosso método proposto pode amenizar os problemas causados pelo ruído.

Palavras-chaves: Métodos de Ensemble. Sistemas de Múltiplos Classificadores. Seleção Dinâmica. Ruído de Classe. Bagging.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AUC	Area Under the (Receiver Operating Characteristic) Curve
DCS	Dynamic Classifier Selection
DES	Dynamic Ensemble Selection
DS	Dynamic Selection
DSEL	Dynamic Selection data set
ENN	Edited Nearest Neighbors
GGA	Generational Genetic Algorithm
KDN	K-Disagreeing Neighbors
KNORAE	K-Nearest Oracles Eliminate
KNORAU	K-Nearest Oracles Union
LCA	Local Class Accuracy
MCB	Multiple Classifier Behavior
MCS	Multiple Classifier Systems
MSDS	Multiple-Set Dynamic Selection
OLA	Overall Local Accuracy
PS	Prototype Selection
PSO	Particle Swarm Optimization
RNG	Relative Neighborhood Graph
RMHC	Random Mutation Hill Climbing
RoC	Region of Competence
RRC	Randomized Reference Classifier
SSMA	Steady-State Memetic Algorithm

LIST OF SYMBOLS

$\emptyset$	The empty set
$\cup$	Set union
$\lvert\cdot\rvert$	Set cardinality
$\leftarrow$	Variable Assignment
$\setminus$	Set difference
$\wedge$	Logical and
$\in$	Set membership

CONTENTS

1 INTRODUCTION

In Machine Learning, Ensemble Methods (ZHOU, 2012), (KUNCHEVA, 2014) are a class of methods which combine multiple, independently trained predictors into one. More specifically, we are interested in the subclass of methods based on Dynamic Selection (BRITTO; SABOURIN; OLIVEIRA, 2014), (CRUZ; SABOURIN; CAVALCANTI, 2018), which are methods that attempt to select the best set of classifiers to predict the label of each test instance. These methods have been shown to outperform not only techniques that use a single classifier , but also those which combine all classifiers in an ensemble (BRITTO; SABOURIN; OLIVEIRA, 2014), (CRUZ; SABOURIN; CAVALCANTI, 2018).

Despite the successes achieved by Ensemble Methods, they are still susceptible to problems when faced with noisy scenarios, as are most Machine Learning algorithms dealing with classification problems (FRÉNAY; VERLEYSEN, 2014). Noise, and specifically label noise, as understood in this work, is any process that changes the true value of a instance's class label from its true value. The presence of noise may lead classifiers to overfit on the training data in an attempt to adjust itself to the noise, which may lead to severe degradation in performance.

In (FRÉNAY; VERLEYSEN, 2014), the authors list four approaches to dealing with noisy data:

1. Adopting probabilistic models, which inherently account for possible variations within data when making predictions.

2. Modifying the loss function during training, in order to make the classifiers less likely to overfit on noisy instances.

3. Using Prototype Selection (GARCÍA et al., 2012) techniques to clean up the data set.

4. Using Ensemble Methods.

Prototype Selection techniques (GARCÍA et al., 2012) can be considered an important set of tools for dealing with noisy scenarios. Methods such as the Edited Nearest Neighbors (ENN) rule (WILSON, 1972) attempt to remove instances which appear to be inconsistent with the rest of the data, and therefore can be used in order to attempt a "clean-up" of noisy data.

However, of particular interest to us, are the results in the classification literature which describe Ensemble Methods as being particularly resilient to these noisy scenarios (FRÉNAY; VERLEYSEN, 2014), (MELVILLE et al., 2004), making them an interesting tool for dealing with noise.

We can see then how Ensemble Methods can be used not only to achieve good performance on classification problems, but also to ensure a degree of resistance to noise in these types of problems. Nevertheless, Dynamic Selection techniques, which have been on the vanguard of Ensemble Methods, can be sensitive to the presence of noise. For these techniques, the quality of the Dynamic Selection data set (DSEL), which is used during the process of selecting the classifiers, is of paramount importance for achieving good performance (CRUZ; CAVALCANTI; TSANG, 2011), (CRUZ; SABOURIN; CAVALCANTI, 2017a), (CRUZ; SABOURIN; CAVALCANTI, 2018).

Considering the aforementioned issue, in (CRUZ et al., 2018), Cruz et al. propose using the ENN rule, in combination with the KNN-Equality algorithm to increase the performance of the Dynamic Frienemy Pruning technique (OLIVEIRA; CAVALCANTI; SABOURIN, 2017), a proposal which they name the Fire-DES++ framework. The ENN rule is used as a means to perform a cleaning of the data set, in order to avoid the case in which classifiers are not pruned from the ensemble due to the presence of noisy instances or outliers. However, in (FRÉNAY; VERLEYSEN, 2014), the authors argue that noise removal techniques can cause excessive removal of instances, removing instances that are not noise or distorting the border of classes, since the instances near the border of classes tend to be inherently hard and prone to being removed.

Taking these matters into account, in this work, we attempt to answer the following questions:

1. Just how deleterious is the presence of noise to Dynamic Selection algorithms?
2. How do the effects of noise on Dynamic Selection algorithms manifest themselves? Are all algorithms equally affected?
3. Is there an alternative filtering approach to that of the Fire-DES++ algorithm, which does not incur the risk of excessive removal?

In order to answer the first two questions, we perform experiments on 64 public data sets, with varying noise levels and a noise-free scenario as a baseline. We observe how different Dynamic Selection algorithms perform under these conditions, and compare their behavior both against each other and on different noise levels.

Considering the third question, we propose replacing the ENN filtering step in (CRUZ et al., 2018) with a similar bootstrapping procedure as that proposed by Walmsley et al. in (WALMSLEY et al., 2018). Instead of a single DSEL set, filtered by ENN, we propose combining the output of several classifier ensembles, each using a bootstrapped Dynamic Selection set based on the original DSEL, using Instance Hardness as a means to determine the selection probability of each instance in the DSEL. As in (WALMSLEY et al., 2018), we chose this probabilistic filtering approach for its ability to filter out the noisy data, while still allowing for the retention of hard instances in the class borders, with non-zero probability. The results for our proposal are compared not only against the method of Cruz et al., but also against the traditional Overall Local Accuracy, Local Class Accuracy (WOODS; KEGELMEYER; BOWYER, 1997), K-Nearest Oracles-Eliminate and K-Nearest Oracles-Union (KO; SABOURIN; BRITTO, 2008) algorithms.

Our results show that the performance of most Dynamic Selections algorithms is heavily degraded by the presence of noise, to the point of losing to a simple combination of all classifiers in the ensemble. We present an analysis pointing to likely failure modes of these algorithms, and also discuss how the K-Nearest Oracles-Union manages to distinguish itself from the other methods, and maintain some degree of resistance to noise.

Furthermore, we find that our proposed method can, under certain circumstances, help Dynamic Selections algorithm maintain their superiority, even under noise scenarios.

1.1 STRUCTURE OF THE DOCUMENT

This work is organized as follows. In Chapter 2, we present the reader with the fundamental concepts of Ensemble Methods, Dynamic Selection and Prototype Selection, as well as discuss the interplay between these concepts. We also discuss the works which served as the basis for this investigation. In Chapter 3, we detail our proposed method, presenting a step-by-step breakdown of its working. In Chapter 4, we discuss our experimental methodology and the rationale behind each of our choices in developing the experiment. We then present our results and discuss our findings. Finally, in Chapter 5 we present a summary of our findings and suggest possible future lines of investigation.

2 BACKGROUND AND RELATED WORKS

In this chapter, we present fundamental concepts related to Ensemble Methods, Dynamic Selection, Noise, Prototype Selection, the Bagging-IH algorithm and the Fire-DES++ algorithm. These concepts are presented in order from the more basic to the more specific, allowing the reader to build an understanding of the area and where this work fits into it.

2.1 ENSEMBLE METHODS

In Machine Learning, Ensemble Learning techniques (KUNCHEVA, 2014), (ZHOU, 2012) are a family of techniques that combine multiple predictors, trained independently and on the same data set, into a single predictor . These techniques combine the outputs of the multiple classifiers into a single output, used as the output of the system. The field is also referred to as Ensemble Methods and Multiple Classifier Systems (in the case of classification problems), and these terms are used interchangeably in this work.

Although it is common in Machine Learning to utilize a single model trained on the whole of the training data set, the central assumption of Ensemble Methods is that by applying ensemble techniques one can obtain a pool (a set of models) with complementary competences, or equivalently, a *diverse* ensemble. By complementary competences, we mean that we expect to models to not all fail in the same manner, or, equivalently, to fail differently in different instances.

There are multiple approaches to Ensemble Learning, and while some of them are focused on regression problems, this work only concerns itself with classification problems. Thus, it is accurate to say that the present work is situated in the field of Multiple Classifier Systems (hereafter, MCS).

In (BRITTO; SABOURIN; OLIVEIRA, 2014) the authors propose a taxonomy of MCS, and in doing so, delineate three main phases of the use of MCS. The phases are:

1. **Pool Generation:** The process of creating the pool of classifiers which will be used for classification. The authors distinguish between homogeneous and heterogeneous systems, depending on whether all classifiers in the pool are of the same type or not.

2. **Selection:** The selection of which classifiers will be used in generating the output of the system. These techniques used in these steps can be broadly subdivided into Static Selection and Dynamic Selection schemes. In Dynamic Selection schemes, the system selects a subset of the pool of classifiers when classifying each test instance. On the other hand, in Static Selection schemes, once the choice of classifiers is made, it remains unchanged. Furthermore, it is possible to say that some methods do not perform classifier selection, since all classifiers are included in the final pool.

3. **Integration:** The combining of the output of the individual classifiers in order to generate the final prediction of the system. This process can be performed based on the class label predicted by each classifier, or on the estimated class probabilities, when available.

We shall now discuss each of the phases above in more detail.

2.1.1 Pool Generation

The pool generation process is the first step in any Multiple Classifier System. It is in this phase that we must take care to generate diverse classifiers. With this in mind, we must now make a small aside to discuss the topic of diversity in ensembles.

In (BRITTO; SABOURIN; OLIVEIRA, 2014), the authors note that diversity is paramount to ensembles, and also that we can naturally expect diversity from heterogeneous ensemble, but can nevertheless achieve diversity in homogeneous ensembles. This can be done by either varying the initialization of the model (such as in the case of the weights of a Neural Network), by varying the training examples to which the model has access (such as in the case of the Bagging algorithm (BREIMAN, 1996)) or by varying the features of the data that the model can see (such as in the Random Subspace algorithm (HO, 1998)).

For a comparative study between different measures of diversity, their correlation with ensemble accuracy, and the correlations between measures, we recommend the work in (SHIPP; KUNCHEVA, 2002), which also highlights the difficulty in precisely defining and quantifying diversity.

Coming back to the question of how classifier pools can be generated, we can turn to the work in (WOŹNIAK; GRAÑA; CORCHADO, 2014), which identifies two main topologies for Multiple Classifier Systems: parallel and serial. In parallel topologies, all classifiers will receive the same data and produce outputs, while in serial topologies not all classifiers will be called upon to produce outputs, the assumption being that classifiers further down the line are only used when the more "basic" classifiers fail to produce an output with enough confidence.

We have already mentioned two examples of parallel topologies, the Bagging and the Random Subspace algorithms. Since ideas from Bagging will be present in several moments during our discussion, and given that they inform decisions on the design of our proposed method, we would like to now describe the Bagging algorithm in more detail. Algorithm 1 describes the functioning of the Bagging algorithm.

Note that this algorithm describes the classifier pool generation step. At test time, the outputs of the pool can be combined using a suitable integration scheme, though this is usually done through a simple majority voting. Integration schemes are presented further ahead in this document.

Now, in order to finalize our discussion on parallel topologies, we would like to highlight the Random Forest algorithm (BREIMAN, 2001), which combines elements of both the Bagging and Random Subspace algorithms. The authors generate a pool of decision trees using the

Algorithm 1: The Bagging algorithm

```
   Input  : The training data set $T$
            The number of bootstrapped sets $m$
            The bootstrapped set size $n_b$
            A base classifier $C$
   Output: A pool of classifiers $P$
 1 begin
 2     $P \leftarrow \emptyset$
 3     for $i$ from 1 to $m$ do
 4         $B_i \leftarrow \emptyset$ ;            /* $B_i$ are the bootstrapped training sets */
 5         for $j$ from 1 to $n_b$ do
 6             Add an instance $x_j \in T$ to $B_i$, sampled with replacement.
 7         end
 8         Train a classifier $C_i$ using $B_i$
 9         $P \leftarrow P \cup \{C_i\}$
10     end
11 end
```

Bagging algorithm, but, at each tree node split, use only a random subset of the features to perform the split decision. The random forest algorithm has been applied successfully to many different domains.

Lastly, considering the serial topologies, we would like to discuss the AdaBoost algorithm (FREUND; SCHAPIRE, 1997), (SCHAPIRE, 1999), which enjoys wide adoption in the literature. The central idea behind the AdaBoost algorithm lies on training over several steps a set of classifiers, in which each classifier is specialized in classifying the examples misclassified during the previous step. This can be achieved by setting different weights to the term corresponding to each training example in the loss function, according to the error associated to that example.

2.1.2 Selection

In this stage, the classifiers which will be used to produce the final output of the system are selected. In Static Selection, these classifiers are selected only once, and then used for classifying all instances in the test set. In (CRUZ; SABOURIN; CAVALCANTI, 2018), the authors point out that the most common measures for selecting classifiers are diversity and accuracy, while also noting that techniques such as greedy search and evolutionary algorithms have also been used to this end.

Opposite to Static Selection, we have Dynamic Selection (DS) approaches, which are the focus of this work. The core idea behind DS is that, in a pool of diverse classifiers, different classifiers will be specialized in different types of instances, which lie in different regions in the data set. We will dedicate a more complete exposition to DS approaches, and shall come to it shortly.

2.1.3 Integration

The final step in Multiple Classifier Systems is the integration of the outputs of the classifiers to form the final answer of the system. In (CRUZ; SABOURIN; CAVALCANTI, 2018), the authors delineate three different strategies for integration of the outputs:

1. **Non-trainable approaches:** Which always combine the classifiers according to a fixed rule.
2. **Trainable approaches:** Which learn to combine the classifiers in an optimal manner.
3. **Dynamic weighting rules:** Which attribute instance-specific weights to each classifiers.

Non-trainable approaches are frequently used in the literature. Common examples are the Majority Vote Rule, the Sum Rule, the Product Rule, the Max Rule and the Min Rule. For a discussion of these rules and their derivation under a Bayesian approach, see (KITTLER, 1998). We emphasize here the Majority Vote rule, which is widely used and also used in the present work. The Majority Vote rule simply attributes to a instance the most commonly predicted class amongst the selected classifiers.

The authors in (CRUZ; SABOURIN; CAVALCANTI, 2018) note that trainable approaches have been found to often outperform non-trainable approaches, in spite of the latter's wide presence in the literature. For example, in (Cruz; Cavalcanti; Ren, 2010), the authors apply a Multi-layer Perceptron in the Integration step, and find that it achieves superior performance to that of all non-trainable rules.

Finally, the authors of the aforementioned survey also note that dynamic weighting rules are, after a manner, similar to Dynamic Selection approaches, though the former consider the output of all (previously selected) classifiers in producing the output of the system.

2.2 DYNAMIC SELECTION

We elaborate now on the topic of Dynamic Selection methods. We can say that the objective of Dynamic Selection algorithms is: Given a test pattern x_q, a pool of classifiers P, and a Dynamic Selection set $DSEL$ with which to evaluate the performance of the classifiers in the pool, to select the subset of classifiers $P^{'} \subseteq P$ we believe to be the most fit to classify x_q.

In (BRITTO; SABOURIN; OLIVEIRA, 2014), the authors perform a review of the literature and compare the performance of Dynamic Selection against the single best classifier in the pool, a Static Selection approach and a combination of all classifiers in the pool, and find Dynamic Selection to be superior to the alternatives.

We should note here that Dynamic Selection methods can be further divided into two types: Dynamic Classifier Selection algorithms, and Dynamic Ensemble Selection algorithms. The former selects only the single most competent classifier to classify an instance, while the latter forms an ensemble which may contain multiple classifiers.

In (CRUZ; SABOURIN; CAVALCANTI, 2018), the authors identify three main phases to the selection process, namely, definition of the Region of Competence, determination of selection criteria, and determination of selection mechanism. We will treat each step separately here.

2.2.1 Determination of the Region of Competence

For each test pattern x_q, we must determine which portions of the feature space will be used to evaluate the classifiers in P. More precisely, we must choose which instances in $DSEL$ will be considered when evaluating which classifiers will be selected. This set of instances is commonly referred to as the Region of Competence (RoC).

In (CRUZ; SABOURIN; CAVALCANTI, 2018), the authors divide the methods for defining the RoC into four types. The first type is the group of methods that use clustering techniques to partition the DSEL. During the classification of a test instance x_q, the instance is first attributed to a cluster, and the elements belonging to that cluster define the RoC. A simple implementation of such a method is given in (KUNCHEVA, 2000), in which the data set is partitioned into clusters, and the most competent classifier is selected for each cluster. Then, at test time, each instance x_q is attributed to its closest cluster centroid, and classified by the classifier associated to that cluster. The major drawback of these clustering approaches is that they rely on the quality of the produced clusters, and the clusters may either fail to capture the structure of the local area, or end up grouping multiple regions with different characteristics, into a single cluster.

The second type of algorithm consists of those methods which use the k-Nearest Neighbors algorithm to define the RoC. For each pattern x_q, its k-Nearest Neighbors on the $DSEL$ set are determined, and only those patterns are used in the selection of classifiers. This is a very common approach, used for example in the Overall Local Accuracy (OLA) and Local Class Accuracy (LCA) algorithms (WOODS; KEGELMEYER; BOWYER, 1997), the K-Nearest Oracles-Union (KNORAU) and K-Nearest Oracles-Eliminate (KNORAE) algorithms (KO; SABOURIN; BRITTO, 2008), the Multiple Classifier Behavior (MCB) algorithm (HUANG; SUEN, 1995), Apriori and Aposteriori (GIACINTO; ROLI, 1999) algorithms.

The third type of algorithm are those that use a potential function model to define the RoC. In those algorithms, all instances of $DSEL$ are considered to belong to the RoC, but their influence on the estimation of competence is weighted by their distance to x_q, according to a potential function, such that the farther the instances, the lower their influences. Commonly adopted potential functions are those based on a Gaussian distribution, in which the weight of an instances $x_i \in DSEL$ is of the form $w_i \propto -exp(d(x_i, x_q))$.

An important exponent of the class of algorithms which use potential functions is the Randomized Reference Classifier (RRC) algorithm (Woloszynski; Kurzynski, 2010), (WOLOSZYNSKI; KURZYNSKI, 2011), which has achieved some of the best results reported in the literature. The RRC algorithm is based on the homonymous concept of the Randomized Reference Classifier. Consider a classification problem with M classes. Given an instance of the $DSEL$ set, x_{NN}, and a classifier C_j which produces an M-dimensional vector of predicted class probabilities. The corresponding RRC is a classifier modeled by M random variables Δ_i, such that the expected value of Δ_i is equal to the class probability for class i predicted by C_j. The probability that the RRC would have correctly classified x_{NN} is calculated and denoted by $P(C_j, x_{NN})$. This probability is computed for all instances belonging to $DSEL$, and the competence of a classifier is computed as the sum of the terms $P(C_j, x_k)$, $1 \leq k \leq |DSEL|$, with each term being scaled by a factor of $-exp(d(x_k, x_q))$

The final type of algorithm for determining the RoC are those that change the view of the data set to the decision space. These algorithms, instead of working on the space of the features of the instances, actually considers the space of their output profiles. The output profile of an instance is simply the vector of the class labels attributed by each classifier in the pool to that instance. These algorithms then use the output profile of the instances in $DSEL$ and of x_q to find the instances in $DSEL$ most similar to x_q. A well-known example of algorithms which work in the decision space is the MCB algorithm (HUANG; SUEN, 1995), which computes the local accuracy of each classifier considering the k-Nearest Neighbors of the test pattern x_q in the decision space.

2.2.2 The Oracle Model

Before we proceed with our discussion, we must first introduce an important theoretical tool in the MCS toolbox, the Oracle model. Introduced in (KUNCHEVA, 2002), this model defines an ideal selection scheme as follows: if any classifier in the pool P correctly classifies the test pattern x_q, then the ensemble is said to correctly classify x_q.

Clearly, this scheme can not be realized without access to the actual class label of x_q, and therefore only actually exists as an idealized model. Nevertheless, it can be used as a means to establish a theoretical upper bound on the performance of selection schemes on a given data set. One should exercise caution, however, when using to Oracle model, since it can be a far too optimistic upper bound.

Souza et al. investigate this exact issue in (SOUZA et al., 2017). The authors note that a gap between the Oracle performance and actual observed performance of DCS techniques has been reported (DIDACI et al., 2005), (CRUZ et al., 2015), and set out to further investigate the issue. A method for generating classifier pools which ensures 100% Oracle accuracy on the training set is proposed, named the Self-Generating Hyperplanes model, that works by creating linear classifiers until each pair of examples in the set is correctly separated by at least one classifier. Nevertheless, the accuracy of the tested DS algorithms falls way short of the theoretical 100%. The authors argue that this indicates that the Oracle, with its ability to chose from any classifier in the pool, is a poor guide for generating classifiers to be used in a DS scheme, since these schemes often rely only on local information, and therefore can only evaluate the classifiers according to this view of the data.

These results not withstanding, the Oracle model still informs many algorithms in the literature, and is therefore an important concept to keep in mind. We shall shortly come to the discussion of these algorithms.

2.2.3 Determination of Selection Criteria

Returning from our brief detour, we now tackle the issue of the second step in the Dynamic Selection process - the determination of the selection criteria. These are the criteria used to estimate the competence of each classifier, i.e. to estimate how likely it is to be a good fit for classifying x_q. In (CRUZ; SABOURIN; CAVALCANTI, 2018), the authors divide the selection criteria into two groups: group-based measures and individual-based measures.

Group-based measures consider the overall composition of the final ensemble when selecting the classifiers which will compose it. While the methods vary, their central idea is to ensure that each classifier will contribute to the overall performance and robustness of the pool. These methods can be further subdivided into methods that consider Diversity, Data Handling and Ambiguity.

On the other hand, individual-based measures, as the name implies, evaluate each classifier separately during the selection of the most competent classifiers. The authors in (CRUZ; SABOURIN; CAVALCANTI, 2018) further subdivide them into the following categories: Ranking, Accuracy, Probabilistic, Behavior, Oracle, Data complexity and Meta-learning.

In this work, we focus on algorithms based on accuracy and the Oracle model. Methods based on accuracy consider the accuracy of the classifiers in the local region. From this class, we chose to use the OLA and LCA algorithms. From the class of selection algorithms which use the concept of Oracle, we chose the KNORAE and KNORAU algorithms. We will now take advantage of this opportunity to discuss these algorithms.

The OLA algorithm works by simply selecting the most accurate classifier in the local region. Algorithm 2 details the process of selecting the most competent classifier.

Algorithm 2: The Overall Local Accuracy Algorithm

```
Input  : The query pattern $x_q$
         The classifier pool $P$
         The $DSEL$ set
         The value of $k$ for the $k$-nearest neighbors classifier
Output: The most accurate classifier $c^*$
1 begin
2     $\theta \leftarrow kNN(x_q)$ ;      /* The k-Nearest Neighbors compose the region of
      competence. */
3     foreach $c_j \in P$ do
4         $\delta_{j,q} \leftarrow \frac{1}{k} |\{\theta_k : label(\theta_k) = c_j(\theta_k)\}|$
5     end
6     $c^* \leftarrow c_{MAX}, MAX = \underset{j}{\operatorname{argmax}} \, \delta_{j,q}$
7 end
```

Line 4 of Algorithm 2 indicates how the competence of each classifier is estimated: it is based on the fraction of correctly classified instances in the Region of Competence. Line 6 defines how the most competent classifier is selected - it is simply the classifier with the highest competence as estimated in Line 4.

The LCA algorithm operates in a similar manner, to the OLA algorithm. However, it only takes into account those instances in the Region of Competence to which classifier c_j attributed the same label as it attributed to x_q. Algorithm 3 details the working of the LCA algorithm.

Algorithm 3: The Local Class Accuracy algorithm

```
  Input  : The query pattern $x_q$
           The classifier pool $P$
           The $DSEL$ set
           The value of $k$ for the $k$-nearest neighbors classifier
  Output: The most accurate classifier $c^*$
1 begin
2     $\theta \leftarrow kNN(x_q)$ ;      /* The k-Nearest Neighbors compose the region of
       competence. */
3     foreach $c_j \in P$ do
4         $c_q \leftarrow c_j(x_q)$
5         $\delta_{j,q} \leftarrow \frac{1}{k} |\{\theta_k : label(\theta_k) = c_j(\theta_k) \wedge c_j(\theta_k) = c_q\}|$
6     end
7     $c^* \leftarrow c_{MAX}, MAX = \underset{j}{\operatorname{argmax}}\, \delta_{j,q}$
8 end
```

Considering now the algorithms which use the concept of Oracle, we consider the KNORAU algorithm. This algorithm selects a pool of competent classifiers, instead of a single classifier. Any classifier that correctly classifies at least one instance in the Region of Competence is considered competent, and added to the final pool.

It is important to note here that, during the integration step, each classifier will be assigned a number of votes proportional to the number of instances it correctly classified, that is, proportional to $\delta_{j,q}$.

The KNORAE algorithm on the other hand will only consider competent those classifiers which correctly classify *all* instances in the RoC. It starts with a neighborhood size of k, and if no classifier can correctly predict the label of all k instances in the RoC, the algorithm decrements k by 1 and attempts again to find competent classifiers.

Note that Line 3 of Algorithm 5 means that the algorithm will continue to run unless the size of the RoC reaches zero, or at least one competent classifier has been found (meaning that P is non-empty, see Lines 12-16).

Finally, we would like to briefly discuss some algorithms in the Meta-learning family, due to their excellent results, and due to the fact that they present a different paradigm. The methods we want to discuss are those based on the META-DES algorithm (CRUZ et al., 2015). The META-DES technique treats the problem of estimating the competence of classifiers as

Algorithm 4: The KNORAU algorithm

```
Input  : The query pattern $x_q$
         The classifier pool $P$
         The $DSEL$ set
         The value of $k$ for the $k$-nearest neighbors classifier
Output: The selected pool $P^{'}$
begin
    $\theta \leftarrow kNN(x_q)$ ;      /* The k-Nearest Neighbors compose the region of
     competence. */
    $P^{'} \leftarrow \emptyset$
    foreach $c_j \in P$ do
        $\delta_{j,q} \leftarrow \frac{1}{k} |\{\theta_k : label(\theta_k) = c_j(\theta_k)\}|$
        if $\delta_{j,q} > 0$ then
            $P^{'} \leftarrow P^{'} \cup \{c_j\}$
        end
    end
end
```

Algorithm 5: The KNORAE algorithm

```
Input  : The query pattern $x_q$
         The classifier pool $P$
         The $DSEL$ set
         The value of $k$ for the $k$-nearest neighbors classifier
Output: The selected pool $P^{'}$
begin
    $S \leftarrow false$
    while $k > 0$ and $!S$ do
        $\theta \leftarrow kNN(x_q)$ ;   /* The k-Nearest Neighbors compose the region of
         competence. */
        $P^{'} \leftarrow \emptyset$
        foreach $c_j \in P$ do
            $\delta_{j,q} \leftarrow \frac{1}{k} |\{\theta_k : label(\theta_k) = c_j(\theta_k)\}|$
            if $\delta_{j,q} = 1$ then
                $P^{'} \leftarrow P^{'} \cup \{c_j\}$
            end
        end
        if $P^{'} = \emptyset$ then
            $k \leftarrow k - 1$
        else
            $S \leftarrow true$
        end
    end
end
```

a meta-learning problem, in which a meta-classifier is trained to estimate the competence of classifier $c_j \in P$ in relation to the problem of predicting the class label of x_q.

In (CRUZ; SABOURIN; CAVALCANTI, 2015), the authors perform an evaluation concerning the choice of model for the meta-classifier, and find that the Naive Bayes algorithm offers the best results, and thus it is adopted as the meta-classifier. Furthermore, the authors investigate whether using Dynamic Selection, Dynamic Weighting or a combination (a hybrid) of both is better for accuracy, and conclude that the hybrid approach is the best approach.

Finally, in (CRUZ; SABOURIN; CAVALCANTI, 2017b), the authors evolve the META-DES framework further, by performing meta-feature selection using Particle Swarm Optimization (PSO) techniques. Each solution in the PSO optimization process consists of a set of meta-features, which are used to train a meta-classifier. The optimization objective (i.e. the value to be minimized) of the PSO algorithm is based on the distance of the competence estimated by the meta-classifier and the competence estimated by the Oracle for each instance. The competence estimated by the Oracle for a classifier c_j is simply 1 if the classifier c_j correctly classifies the instance, and 0 otherwise. The authors evaluate this new scheme and find that it significantly outperforms all other 10 state-of-the-art algorithms it was compared against.

2.2.4 Determination of Selection Mechanism

The final step defined in (CRUZ; SABOURIN; CAVALCANTI, 2018) is choosing whether only the most competent classifier will be used in classifying x_q (Dynamic Classifier Selection) or all classifiers that achieve a certain level of competence will be used (Dynamic Ensemble Selection). The authors point out that choosing multiple classifiers has the effects of avoiding the risk involved in trusting one single classifier, and also of doing away with the need to break ties in the event that multiple classifiers achieve the same level of competence.

2.3 NOISE AND CLASSIFICATION PROBLEMS

Having exposed the reader to the central concepts of Ensemble Methods and Dynamic Selection, we now approach the issue of the effect of the presence of noise on classification problems, since this work is focused on the effects of noise on Dynamic Selection algorithms. Nevertheless, we must first define what we mean by noise. Here, we adopt the definition used in (WALMSLEY et al., 2018), in which noise is considered as any process that changes the true value of a variable from its original value.

Noise could be introduced into data for several reasons, such as:

- Measurement errors (systematic or otherwise).
- On a physical channel, effects such as thermal noise or electromagnetic interference.
- Human mislabeling of data.

In the context of classification problems, we differentiate between two types of noise: **feature noise**, and **label noise**. The former acts on the features that describe an instance in a data set, while the latter acts on its class label. The present work deals with the issues caused by the presence of label noise.

In (FRÉNAY; VERLEYSEN, 2014), the authors discuss at length the negative effects of noise on classification problems, and we draw heavily from their framework and findings. In their work, the authors classify the possible variations of label noise into three broad categories:

- **Noisy completely at random:** The introduced noise is not correlated to either the label or the features of the instance.
- **Noisy at random:** The introduced noise depends on the label of the instance.
- **Noisy not at random:** The introduced noise depends on both the features and the label of the instance.

The authors of (FRÉNAY; VERLEYSEN, 2014) also enumerate different approaches to dealing with noise. The first of these approaches is by making the models themselves robust to noise, by means of adopting a loss function during training that is less sensitive to the presence of noise. Furthermore, the authors echo results in the literature which describe how Ensemble Methods can, under certain circumstances, lead to greater robustness to noise. We will soon discuss these results with more care.

The second approach is by applying filtering to the data in an attempt to clean the data. To this end, one may employ Prototype Selection (PS) techniques, which attempt to select only the most relevant examples in a data set. We shall discuss PS techniques in more detail in the next section, and in particular their application to data cleaning.

Moreover, the authors also consider the case of algorithms that achieve robustness to noise by taking a probabilistic approach. Finally, the authors discuss modifications to classic algorithms such as the SVM, Decision Trees, and Neural Networks in order to make them more robust to noise.

We are mainly interested in the data filtering and ensemble approaches, as these form the bases of the approaches which motivate our work, and will now discuss them. As a comprehensive treatment of noise in the general case is beyond the scope of this work, we refer the reader to aforementioned work of Frénay and Verleysen for further details.

2.3.1 Ensemble Learning and Noise

As discussed previously, the authors in (FRÉNAY; VERLEYSEN, 2014) discuss the use of Ensemble Methods as a possible approach to dealing with noise. In fact, a number of authors have discussed how Ensemble Learning can be helpful in dealing with noise. In (DIETTERICH, 2000), the authors conduct experiments to assess how the C4.5 algorithm (QUINLAN, 1993) performs under conditions of noise. While the AdaBoost algorithm seems to be the best alternative for constructing ensembles on noise-free scenarios, as the noise level rises, the Bagging algorithm proves to be much more robust.

These results are also echoed by (MELVILLE et al., 2004), who perform experiments involving missing features, label noise, and feature noise. Again employing the C4.5 algorithm, the authors find that Bagging is more robust to noise than both AdaBoost and the base classifier used to generate the pool. Furthermore, this work too reports that the performance of AdaBoost degrades as the noise level rises, to the point of being inferior to that of the base classifier. Finally, the authors in (KHOSHGOFTAAR; HULSE; NAPOLITANO, 2011) compare four different learners, four variations of Bagging and four variations of Boosting and also find that Bagging is the most suitable alternative, this time not only on noisy scenarios but also scenarios involving imbalanced data.

These two works illustrate an important point: while the use of Ensemble Algorithms can help in alleviating issues involving noise, the choice of algorithm is crucial. In fact, Long and Servedio show in (LONG; SERVEDIO, 2009) that a large class of Boosting algorithms based on convex potential functions are highly susceptible to the effects of random classification noise.

The results concerning Boosting not withstanding, the Bagging algorithm has found success in applications related to classification under noisy scenarios. In (ABELLÁN; MASEGOSA, 2012), the authors employ credal decision trees (ABELLÁN; MORAL, 2003) to construct ensembles using Bagging. Credal decision trees are based on the concept of imprecise information gain, which creates estimated intervals for the prior probabilities of classes, and for the conditional probabilities of the classes given the features, based on the observed frequency of class labels. This estimates are used inside the class probability terms on the equation for evaluating Information Gain, which is the criterion used to determine when there should be a node split in the tree. The authors conduct experiments comparing the credal tree ensemble with an ensemble based on C4.5, and show that on noise levels of 20% and 30%, their proposed ensemble of credal trees is superior.

While the Bagging-IH algorithm of Walmsley et al. (WALMSLEY et al., 2018) also falls into the category of methods based on Bagging and designed to deal with noise, we will discuss it in the section after the next. This is due to the fact that we would like to first introduce other concepts related to Prototype Selection, and also because it necessitates a more complete treatment, since we make heavy use of it in this work.

2.4 PROTOTYPE SELECTION

As explained previously, one possible option for dealing with the issue of noise is applying Prototype Selection. In (GARCÍA et al., 2012), the authors define the problem of Prototype Selection as that of finding a subset of the original instances in a data set which can be used for classification. However, the criteria which are used to select the instances which will compose the reduced set may vary. The objective of the selection can be just reducing the storage needs, trying to improve classification accuracy, or both. With these possible objectives in minds, García et al. divide PS techniques into three broad categories:

- **Condensation techniques:** which seek to remove instances from the "interior" of class, i.e. instances that are far from the borders between classes, while preserving those instances that make-up the border. The motivation for this approach is the idea that one does not need the innermost instances to define the classification surfaces, and therefore these instances can be removed, thus saving space, without harming classification accuracy.
- **Edition techniques:** which try to remove borderline instances, or instances that seem to be noisy. These techniques mostly aim to improve classification accuracy.
- **Hybrid techniques:** which try to reduce the set of instances as much as possible, without sacrificing accuracy.

We are mostly interested in Edition and Hybrid techniques, for their use in removing noise from data and reducing the computational requirements of the classification procedure. In particular, we will repeatedly discuss the use of the Edited Nearest Neighbors (ENN) algorithm (WILSON, 1972), and draw comparisons between it and other solutions for noise removal. For this reason, we now use this opportunity to explain the functioning of the ENN algorithm.

Algorithm 6: The ENN algorithm

```
Input  : The input data set $T$
         The value of $k$ for the $k$-nearest neighbors classifier
Output: The filtered data set $T'$
1 begin
2     $T' \leftarrow T$
3     foreach $x \in T$ do
4         $pred(x) = kNN(x)$ ;        /* The class of $x$ according to the kNN
           classifier */
5         if $pred(x) \neq label(x)$ then
6             $T' \leftarrow T \setminus \{x\}$
7         end
8     end
9 end
```

As Line 3 and 4 of Algorithm 6 indicate, we iterate over all instances in the set computing the class predicted by a kNN classifier. If the predicted class does not match the actual class, the instance is removed from the final set, as show in Lines 5-7.

In their experimental comparisons, the authors of (FRÉNAY; VERLEYSEN, 2014) highlight the Random Mutation Hill Climbing (RMHC) (SKALAK, 1994), Steady-State Memetic Algorithm (SSMA) (GARCÍA; CANO; HERRERA, 2008) and Relative Neighborhood Graph (RNG) (SÁNCHEZ; PLA; FERRI, 1997) algorithms as some of the best performing methods. Further discussions about PS methods can be found both in the aforementioned article, as well as in (WILSON; MARTINEZ, 2000), which considers a different, but not completely separate, set of PS techniques.

2.4.1 Prototype Selection and Ensemble Methods

Turning our attention to the use of Prototype Selection to increase the performance of Dynamic Selection methods, we can find multiple works using PS techniques to improve the quality of the $DSEL$ set. In (CRUZ; CAVALCANTI; TSANG, 2011), the authors utilize the ENN algorithm to filter the $DSEL$ data set before employing the KNORAE algorithm for ensemble selection. The authors also employ an adaptive KNN algorithm (WANG; NESKOVIC; COOPER, 2007) to define the Region of Competence for Dynamic Selection. The adaptive KNN algorithm works by weighting the distance of a instance x_i to a query instance x_q according to how close x_i is to instances of a different class. Thus, instances closer to other classes are considered more distant to x_q, and less favored than those which are far from another class.

The work in (CRUZ; SABOURIN; CAVALCANTI, 2018) expanded on the work in (CRUZ; CAVALCANTI; TSANG, 2011) by considering 10 Dynamic Selection techniques and 30 data sets, and found that 7 of the 10 evaluated Dynamic Selection algorithms had their classification accuracy significantly increased by using both the ENN algorithm and the adaptive KNN algorithm. Furthermore, the authors found that while using only the adaptive KNN or only the ENN algorithm lead to statistically equivalent results, using both in combination is significantly superior to using either technique alone.

In (CRUZ; SABOURIN; CAVALCANTI, 2017a), the authors consider the same 30 data sets as in the previously mentioned work, and 6 Dynamic Selection techniques, but this time consider 6 PS techniques - 3 edition and 3 hybrid techniques. Representing the edition techniques, RHMC, RNG and the ENN algorithm were chosen. Meanwhile, the three hybrid techniques chosen were the SSMA algorithm, CHC Evolutionary Algorithm (CHC) (ESHELMAN, 1991) and, Generational Genetic Algorithm (GGA) (KUNCHEVA, 1995) (KUNCHEVA; JAIN, 1999). RHMC, SSMA, CHC and GGA are all Evolutionary Algorithms (DEB, 2001). The authors found that all edition techniques were significantly better than both the baseline (no filtering applied to the DSEL) and than the hybrid techniques. Moreover, of the hybrid techniques only SSMA obtained a significantly superior number of wins over the baseline, and even then, only achieved an equivalent rank to the baseline.

A small aside is necessary here to explain that the work in (CRUZ; SABOURIN; CAVALCANTI, 2018) predates the work in (CRUZ; SABOURIN; CAVALCANTI, 2017a) both in inception and date of acceptance for publication, but the latter appeared in print form before the former, which explains the apparent discrepancy in publication dates.

Finally, it is worth noting that the Fire-DES++ algorithm also applies the ENN algorithm to filter the $DSEL$ data set. This algorithm will be discussed in detail in the next section.

2.5 THE BAGGING-IH ALGORITHM

We have already discussed how the use of Bagging and its variants can alleviate the issues faced when performing classification tasks under noisy scenarios, and also how Prototype Selection techniques can be used to remove noise from data sets. Nevertheless, PS techniques can incur the risk of removing too many instances (FRÉNAY; VERLEYSEN, 2014). In this context, Walmsley et al. propose in (WALMSLEY et al., 2018) an Ensemble Method based on Bagging also focused on dealing with noisy data, the Bagging-IH algorithm. This algorithm utilizes the concept of Instance Hardness to modify the bootstrapping process of the Bagging algorithm, by attributing non-uniform selection probabilities to instances, in an attempt to filter out noisy instances and outliers, while avoiding the excessive removal of examples which lie near complex class borders.

2.5.1 Instance Hardness

We adopt here the rather pragmatic definition of (SMITH; MARTINEZ; GIRAUD-CARRIER, 2014) of Instance Hardness: Instance Hardness is simply a measure (or a proxy for) the likelihood that a given instance will be misclassified, regardless of the classification algorithm.

Unlike in (BRIGHTON; MELLISH, 2002) and (MANSILLA; HO, 2004), which take more global approaches (over the whole data set) to estimate the hardness of the instances in the data set, the authors in (SMITH; MARTINEZ; GIRAUD-CARRIER, 2014) focus on each individual instance to estimate its hardness. Of particular interest to us is the k-Disagreeing Neighbors (kDN) measure defined by them according to Equation 2.1:

$$kDN(x) = \frac{|x' \mid x' \in kNN(x) \wedge label(x') \neq label(x)|}{k}, \tag{2.1}$$

where $kNN(x)$ are the k nearest neighbors of an instance x, k is the number of instances in the neighborhood considered and $label(x)$ is the class label of instance x. The kDN measure can be simply interpreted as the fraction of the nearest neighbors of an instance that do not share its class label. Since this value will always take on a value between 0 and 1, it is a natural fit for an estimate of the probability that an instance will be misclassified.

The rationale of Walmsley et al. for adopting this measure is that instances with a high kDN value are likely to be either noisy instances or outliers, and can most likely be safely discarded. Nevertheless, the authors caution that instance on complex borders between classes are likely to have high kDN values. Therefore, a probabilistic approach is adopted, in which each instance is attributed a selection likelihood during the bootstrapping procedure which is greater the lower its hardness, but all instances are given a minimum selection likelihood greater than 0, to avoid discarding hard instances in the border of classes.

The authors of Bagging-IH find that it offers performance gains over the original Bagging algorithm and over the Random Subspace algorithm under noisy scenarios. The authors also observed that the performance of Bagging-IH degrades more slowly than that of Bagging as the level of noise increases. Finally, Bagging-IH was shown as an effective method of removing noisy instances, as the bootstraps generated by this method consistently had lower percentages of noisy instances than that expected of the original Bagging algorithm.

2.6 DYNAMIC FRIENEMY PRUNING AND FIRE-DES++

We would like to now discuss the Fire-DES (OLIVEIRA; CAVALCANTI; SABOURIN, 2017) and Fire-DES++ (CRUZ et al., 2018) algorithms. Both algorithms are based around the concept of Dynamic Frienemy Pruning, and their ideas are central to the investigations developed in the present work.

2.6.1 Dynamic Frienemy Pruning (DFP)

In (OLIVEIRA; CAVALCANTI; SABOURIN, 2017), the authors consider what happens when, in the Dynamic Selection process, the Region of Competence used during the classification of a test instance x_q consists of an *indecision region*, which for the purposes of Dynamic Selection is defined as a region which contains samples from more than one class. The authors present examples of how the OLA, LCA, KNORAE and KNORAU algorithm can select incompetent classifiers when the selection is performed considering an indecision region as the RoC.

In order to remedy this issue, the authors propose the use of Dynamic Frienemy Pruning, which is based on the concept of frienemy. The authors define a pair of instances (x_a, x_b), $x_a, x_b \in RoC(x_q)$ as being frienemies if $label(x_a) \neq label(x_b)$. If the RoC of x_q is an indecision region, which can be detected by the presence of frienemies, then the Dynamic Frienemy Pruning step is applied. This step removes from the classifier pool all classifiers that are unable to correctly classify at least one pair of frienemies in the RoC.

The authors found in (OLIVEIRA; CAVALCANTI; SABOURIN, 2017) that applying the Frienemy Pruning step significantly improved performance for several DS algorithms.

2.6.2 Fire-DES++

The work in (CRUZ et al., 2018) improves upon the Fire-DES method, by integrating measures to improve the quality of the Region of Competence, and therefore to facilitate the selection of competent classifiers.

The first adaptation consists of applying filtering to the $DSEL$ data set, in order to remove noisy instances. This is done to ensure that no classifiers will be pruned due to noise, or that a classifier will be considered more or less competent that it actually is, due to the label they attributed to a noisy instance. The authors investigate the use of both the ENN and the RNG algorithms in the filtering step.

The second adaptation consists of applying the K-Nearest Neighbors Equality algorithm (SIERRA et al., 2011) in the definition of the Region of Competence, which selects an equal number of instances from each class in the data set to form a neighborhood. This ensures that instances from all classes present in the data set will be represented in the RoC.

The authors performed experiments in 64 public data sets, and found that the proposed changes resulted in performance gains for 7 out of the 8 DS algorithms tested, the exception being the LCA algorithm. Furthermore, the authors found that the use of the ENN algorithm during the filtering phase resulted in better performance than the use of the RNG algorithm.

3 PROPOSED METHOD

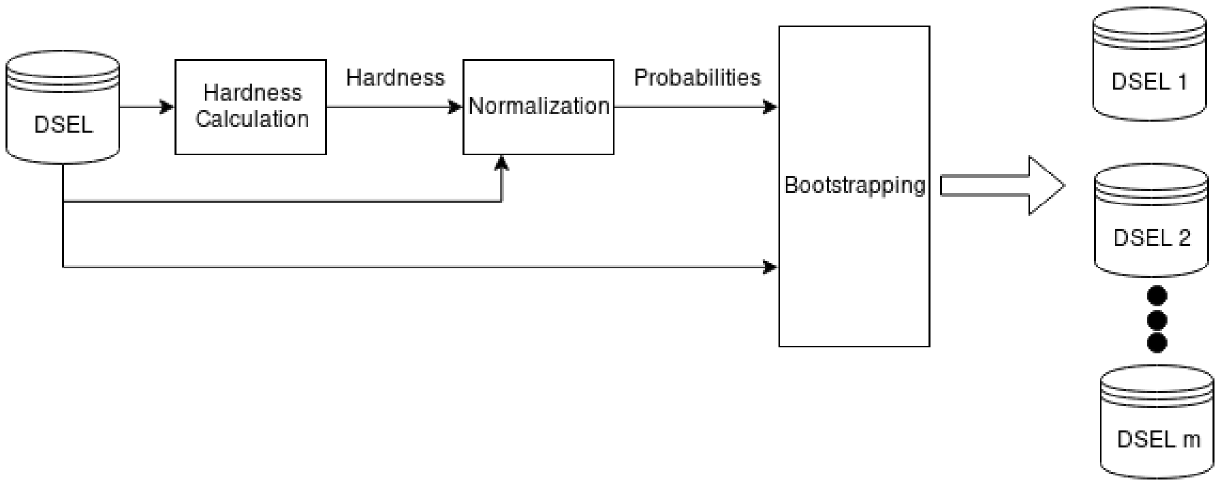
As discussed previously, the issues caused by the presence of noise can be alleviated, but not fully resolved by the use of ensemble methods. We have also discussed how the authors in (CRUZ et al., 2018) proposed to use Prototype Selection to deal with the issue of noise, and achieve good results by using the ENN algorithm. Nevertheless, we are still interested in performing a more detailed investigation into the performance of DS algorithms under noisy conditions. We are further motivated to investigate possible alternative methods of filtering noise from the $DSEL$ data set by the following three factors:

- The discussion in (FRÉNAY; VERLEYSEN, 2014) concerning the possible excessive removal of examples by PS techniques.
- The fact that the authors in (CRUZ et al., 2018) also demonstrate concern over this issue, in particular over the problem of preserving class borders.
- The results in (WALMSLEY et al., 2018), which indicate that noise filtering can be performed by using Instance Hardness information.

In this vein, we have proposed an algorithm similar in functioning to Bagging-IH, but applied to the Dynamic Selection process. We utilize the bootstrapping procedure from Bagging-IH to generate m bootstrapped DSEL sets. The selection probabilities are calculated according to the hardness of each example, in the same manner as in Bagging-IH. Then, we use each of the m bootstrapped sets as a dynamic selection set to predict the class of a test pattern x_q, and combine the outputs using some combination rule, yielding a final prediction for each example. We name this method Multiple-Set Dynamic Selection (MSDS) method.

Figure 1 illustrates the bootstrapping procedure used in our algorithm.

Figure 1 – A block diagram depiction of the bootstrapping procedure.



Source: The author (2019)

The procedure is the same as that described in (WALMSLEY et al., 2018), but we apply it to the DSEL data set. A more precise description of our method is given by Algorithm 7.

Algorithm 7: The algorithm for generating the new validation data sets

```
   Input  : The validation data set $DSEL$
            The number of bootstrapped sets $m$
            The bootstrapped set size $n_b$
            The value of k for the kDN measure
   Output: The new validation data sets $DSEL_i$
1  begin
2      foreach $x \in DSEL$ do
3          $f(x) = kDN(x)$
4      end
5      foreach $x \in DSEL$ do
6          $p(x) = normalize(x, f(x), DSEL)$ ;   /* The selection probability of
             the instance */
7      end
8      for $i$ from 1 to $m$ do
9          Initialize the validation data set $DSEL_i$ as the empty set
10         for $j$ from 1 to $n_b$ do
11             Add an instance $x_j \in T$ to $DSEL_i$, sampled with replacement
                 according to $p(x_i)$
12         end
13     end
14 end
```

Lines 2-4 of Algorithm 7 describe the process of calculating the Instance Hardness of each element in the DSEL set, according to Equation 2.1, a step that corresponds to the “hardness calculation" module in Figure 1. With the hardness of each instance calculated, lines 5-7 describe the process of calculating the selection probability of each instance, and also correspond to the "normalization" module describe in Figure 1. The first step of calculating the selection probability is described Equation 3.1. As explained in (WALMSLEY et al., 2018), the first term attributes an uniform selection likelihood to all instances in the data set, while the second term attributes higher selection likelihood to those instances that have a lower Instance Hardness, and that we therefore consider less likely to be noise.

$$f(x) = \frac{1}{n} + (1 - kDN(x)) \tag{3.1}$$

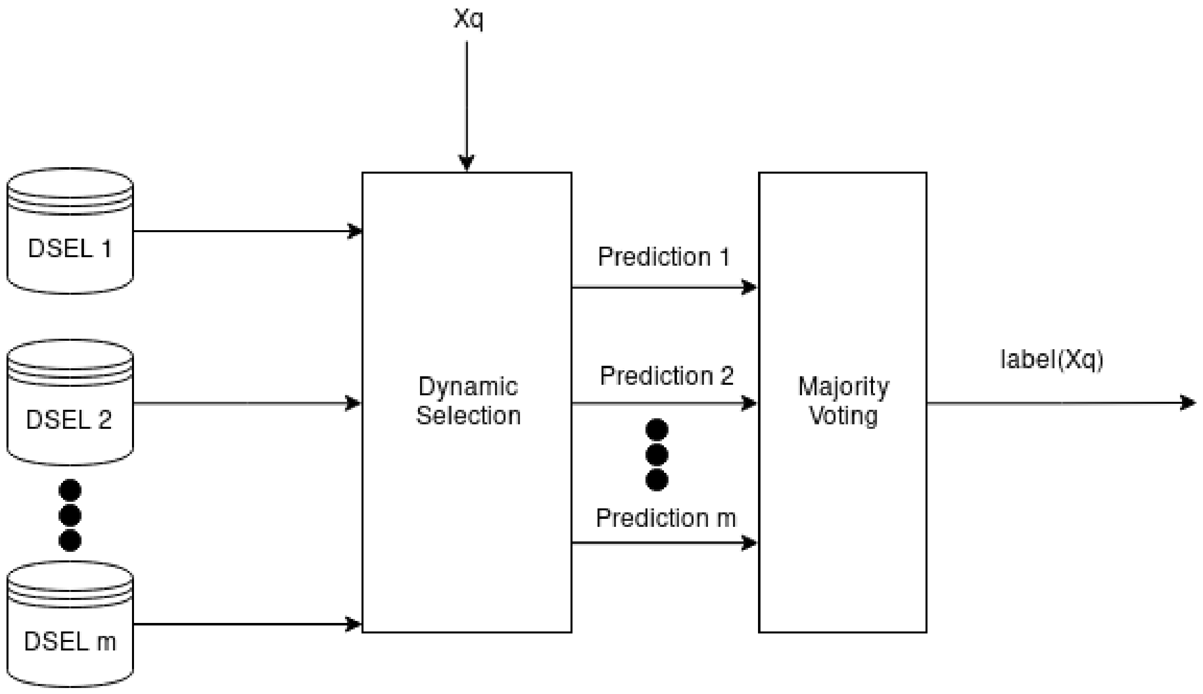
Note that the value computed by the function f does not correspond to a proper probability distribution. In order to actually be able to perform the bootstrapping procedure, we need to normalize the likelihoods into a probability distribution. This normalization is computed according to Equation 3.2

$$p(x_i) = \frac{f(x_i)}{\sum\limits_{i=1}^{n} f(x_i)} \tag{3.2}$$

Lines 8-13 describe how we generate the bootstrapped sets. For each of the m sets, we are going to generate, we select with replacement n_b instances from $DSEL$. Notice that unlike in the traditional Bagging algorithm, the selection probabilities are not uniform, but given by the distribution computed in the previous steps. This step corresponds to the "bootstrapping" module shown in Figure 1.

We now turn our attention to the procedure we follow in order to perform the classification of a test instance. Figure 2 illustrates the procedure, while Algorithm 8 outlines it in detail.

Figure 2 – A block diagram depiction of the test stage of the proposed system.



Source: The author (2019)

Algorithm 8: Prediction on a test instance

```
Input  : The trained pool $P$
         The number of validation data sets $m$
         The $m$ validation data sets $DSEL_i$
         The Dynamic Selection algorithm $A$
         A test instance $x_q$
         A voting scheme $V$ for combining the predictions
Output: The predicted class label of $x_q$, $y_{pred}(x_q)$
1  begin
2      Initialize the set of predictions $Y$ to the empty set
3      for i from 1 to $m$ do
4          Initialize the set of predictions $Y_i$ to the empty set
5          Use $A$ to select the subset of most competent classifiers $P'$ from $P$, using
             $DSEL_i$ as the Dynamic Selection Set
6          for j from 1 to $|P'|$ do
7              Calculate the class label $y_j$ of $x_q$, as predicted by $P'_j$
8              Add $y_j$ to $Y_i$
9          end
10         Aggregate the predictions in $Y_i$ according to the rules of $A$, returning a
             single prediction $y_i$
11         Add $y_i$ to $Y$
12     end
13     Calculate $y_{pred}(x_q)$ as the result of $V(Y)$
14 end
```

As indicated by line 2 of Algorithm 7, we will be working with a set of predictions. Line 4 tells us to create a set which will store the predictions of each classifier selected using $DSEL_i$. Line 5 tells us to use the DS algorithm A to select the most competent classifiers, and corresponds to the "dynamic selection" block in Figure 2. This means that we will select a pool P using $DSEL_i$ to define the Region of Competence, and to estimate the competence of each classifier in P. Lines 6 to 9 correspond to the computation of the predictions of each selected classifier. In Line 10 we aggregate the predictions of each selected classifier to produce a single label y_i. Finally, Line 13 tells us to aggregate all the labels predicted using each $DSEL_i$ to produce the final output of the system, and represents the "majority voting" block depicted in Figure 2.

Note that although we describe the process of predicting class labels, Algorithm 8 can also work with predicted class probabilities, as long as each classifier is capable of outputting such probabilities, and a suitable voting scheme is provided.

It is important to note that MSDS does not depend on the algorithm used for pool generation, nor on the DS algorithm, since it acts as a "wrapper" around the DS procedure. Nevertheless, in this work, we chose to investigate the use of MSDS both on its own and in conjunction with the Fire-DES++ algorithm, as a further evolution of this algorithm.

4 EXPERIMENTS AND ANALYSES

In this chapter, we describe the experiments which have been performed, and the relevant parameters pertaining to them. We then present the results of our experiments, and perform analysis on them. We present these analysis in several pieces, each one of them aiming to answer a particular question or to test a particular hypothesis.

4.1 METHODOLOGY

Here we enumerate all algorithms used and all comparisons performed, and detail the relevant parameters for the execution of the experiments. We also explain the rationale behind each choice, both in terms of how the literature supports these choices, and also how these choices are guided by our hypotheses and the needs of our investigation. Note that since our work stems from our desire to investigate possible shortcomings of the algorithm proposed in (CRUZ et al., 2018), we attempt to follow their methodology as closely as possible, in order to perform a precise and fair comparison.

4.1.1 Experimental Parameters

In the experiments described here we evaluated the OLA, LCA, KNORAU and KNORAE algorithms. These algorithms were chosen due to being classic methods, with a wide presence in the literature. Their behavior is well understood and characterized. Furthermore, since their working is somewhat simpler than more recent algorithms, such as META-DES, we can more easily extract insight from the results, and also better illustrate any interesting behavior. This allows us to perform a deep analysis of the behavior of these algorithms under noisy scenarios.

All algorithms are applied to the same classifier pool, which was generated using the Bagging algorithm, with a pool size of 100, and using the Perceptron algorithm as the base classifier. Each generated bootstrap had the same size as the original training set. This follows the procedure of (CRUZ et al., 2018). The implementation of the Bagging algorithm used was that provided by the scikit-learn library (PEDREGOSA et al., 2011), and the implementations of the DS algorithms used were those provided by the DESlib library (CRUZ et al., 2018). For all the DS algorithms, the size of the Region of Competence was set to $k = 7$, as used by (CRUZ; SABOURIN; CAVALCANTI, 2017a).

As in (WALMSLEY et al., 2018), we investigate the performance of all methods under all noise levels in the set $\{0, 0.1, 0.2, 0.3, 0.4\}$. A noise level of pr_{change} corresponds to a probability of pr_{change} of the label of a given instance being changed to another label in the set of labels present in the data. The new label is chosen completely at random, with the restriction that it cannot be equal to the original label.

We conduct experiments considering a noise level of 0 in order to obtain a baseline under a noise-free scenario. The results under all other scenarios are always be compared to the noise-free case. Furthermore, when noise was added, it was added directly to the training set, before the validation data set was created. The training data set was used as the validation data set for all algorithms, following the procedure used in (CRUZ et al., 2018).

The following scenarios for Dynamic Selection are considered:

1. Each DS algorithm used alone.
2. Each DS algorithm used in conjunction with Fire-DES++.
3. Each DS algorithm used in conjunction with MSDS.
4. Each DS algorithm used in conjunction with both Fire-DES++ and MSDS.

Since MSDS is based on Bagging-IH, which itself is based on Bagging, we also investigate as a baseline utilizing the original bootstrapping procedure of Bagging, with uniform probabilities, as a means to generate multiple validation sets. The prediction at test time is performed in the manner described in Algorithm 8. We dub this strategy BagDS. This means we add the following two scenarios to our experiments:

1. Each DS algorithm used in conjunction with BagDS.
2. Each DS algorithm used in conjunction with both Fire-DES++ and BagDS.

The first comparison we perform is between each DS-based method listed above and the result of combining the predictions of the base pool using Majority Voting. This was mainly devised as a sanity check to ensure our algorithms were working correctly, and to confirm we could replicate the results present in the literature for the noise-free scenario. Nevertheless, as the results will show, this analysis ended up revealing interesting behaviors when in the presence of noise.

We also compare the results of each DS-based method with the base DS algorithm used alone. This is to elucidate when modifications to the base DS are advantageous.

Finally, we compare the results of MSDS against all other DS-based methods. This is done both for MSDS alone, as well as used in conjunction with Fire-DES++. These two scenarios are also compared between them. Note that when MSDS (or BagDS) is used with Fire-DES++, the ENN algorithm is not employed to clean the data set, since this a role intended for MSDS in this case. We also compare MSDS with its "naive" counterpart in BagDS, in order to ascertain whether the use of Instance Hardness in the bootstrapping process is actually useful for the generation of multiple Dynamic Selection sets.

For the number of DSEL sets generated by both BagDS and MSDS, we consider values of m in the set $\{10, 50\}$, in order to investigate the effect of the number of DSEL sets on performance. Once again, each generated set had the same size as the the original $DSEL$ set.

While it may seem unintuitive to compare an algorithm using multiple DSEL against other algorithms which use a single DSEL set, we remind the reader that the MSDS algorithm is inspired by Bagging-IH, which was originally designed as a classifier pool generation algorithm, explicitly reliant on bootstrapping. Therefore, it doesn't make sense to consider a single set generated by MSDS in isolation, particularly considering the probabilistic nature of the the bootstrapping procedure, which naturally requires the generation of multiple sets.

As in (CRUZ et al., 2018), we use the Area Under the Receiver Operating Characteristic Curve (AUC) as our measure of performance, due to the fact that it better reflects performance under a scenario of imbalanced classes. For each classifier, the AUC for that classifier on a data set is computed. This value is then used to rank the classifiers from best to worst, with the best classifier receiving a rank of 1, the second best a rank of 2 and so on. When there is a tie between one or more classifiers, each classifier receives a rank equal to the average rank of all tied classifiers. We then report the average rank of each classifier over the data sets. As in (WALMSLEY et al., 2018), we use the Wilcoxon Signed-Rank test (WILCOXON, 1945) to compare pairs of techniques by their ranks, with each data set being considered a separate trial, for a total of 64 trials.

4.1.2 Data sets

We utilize the same 64 data sets from the KEEL (ALCALÁ-FDEZ et al., 2011) data set repository's collection of "imbalanced data sets" as those used in (CRUZ et al., 2018), in order to provide for a fair comparison. These data sets all represent binary classifications problems, which simplifies the analysis of noisy scenarios. As in the aforementioned work, we utilize the partition into folds provided by the KEEL team, in order to aid in comparing the results with other works. The data provided by the KEEL repository is already divided into five folds, with training and testing data being provided separately for each fold. We do not alter this division in any manner.

4.2 RESULTS AND ANALYSIS

4.2.1 The Effect of Noise on Dynamic Selection

The first question we aim to answer is: how deleterious is the effect of noise to algorithms based on Dynamic Selection? We have already argued that the literature seems to point us to the conclusion that any degradation of the Dynamic Selection set is harmful to the performance of DS algorithms, and also that noise is prejudicial in general to performance in classification problems. We now intend to ascertain just how harmful it is, whether all algorithms are equally affected, and also if DS algorithms are particularly susceptible to the effects of noise.

Firstly, let us observe how the methods behave under the noise-free scenario, in order to establish a baseline, and check our results against what has been reported in the literature.

Table 1 – Average Ranks for each method, for noise level 0.

Method	Avg. Rank
KNORAE - Fire	7.007812
KNORAU - Fire	8.328125
KNORAU - Fire + BagDS	9.789062
KNORAE - Fire + MSDS	10.531250
KNORAU - Fire + MSDS	10.632812
KNORAE - BagDS	10.679688
OLA - Fire	10.734375
KNORAE - Fire + BagDS	11.140625
KNORAU - BagDS	11.218750
KNORAE - DS	11.218750
KNORAE - MSDS	11.414062
KNORAU - DS	11.476562
OLA - DS	11.976562
OLA - BagDS	12.304688
KNORAU - MSDS	12.398438
OLA - Fire + BagDS	13.140625
Static	13.281250
OLA - Fire + MSDS	14.148438
OLA - MSDS	15.250000
LCA - BagDS	17.007812
LCA - Fire + BagDS	17.968750
LCA - DS	18.164062
LCA - Fire + MSDS	18.226562
LCA - MSDS	18.375000
LCA - Fire	18.585938

Source: The author (2019)

+

Table 2 – p-values for the Wilcoxon signed-rank test for noise level 0.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	0.0211	0.0244	0.7509	0.0141	0.0856	0.2767
LCA	0.9997	0.9985	1.0000	0.9941	0.9892	0.9964
KNORAE	0.0177	0.0055	0.1731	0.0000	0.0068	0.0030
KNORAU	0.0036	0.0005	0.0950	0.0000	0.0000	0.0002

Source: The author (2019)

Before discussing our results, we should explain how to interpret the tables. Looking at the first row and first column of Table 2, the result should be interpreted in this manner: The two contexts being compared are the use of Dynamic Selection vs. the use of Static Selection, and the Dynamic Selection algorithm being evaluated is the OLA algorithm. The p-value is 0.0211, which indicates that at a significance level of $\alpha = 0.05$, OLA is significantly better than Static Selection. We should also note that for space saving reasons we refer to the FireDES++ algorithm as either "FireDS" (when used alone) or simply "Fire" (when used with BagDS or MSDS).

Returning to our discussion, the first thing we observe is that methods using Fire-DES++ achieve the two highest rankings in this scenario. Beyond that, all DS methods, except for LCA, are significantly better than Static Selection, considering $\alpha = 0.05$. These results are consistent with what other researchers have reported, mainly the authors of (CRUZ et al., 2018).

It is also interesting to note that, under the noise-free scenario, the use of MSDS seems ill-advised. We observe that the combinations of KNORAE and MSDS, OLA and MSDS, and OLA/Fire-DES++/MSDS are unable to outperform even Static Selection. This is somewhat similar to the results of (WALMSLEY et al., 2018), though one should take care when drawing parallels, given that the aforementioned work deals with Static Selection of classifiers. However, it is possible to argue that a method designed to remove noise should indeed underperfom in a noise-free scenario, and even incur the risk of removing non-noisy instances, which could degrade performance.

Having performed this analysis on the noise-free scenario, we remind the reader that we are mostly interested in the results under noisy scenarios. The results considering noise are as follow:

Table 3 – Average Ranks for each method, for noise level 0.1.

Method	Avg. Rank
KNORAU - Fire + MSDS	6.320312

KNORAE - Fire + MSDS	7.968750
KNORAE - MSDS	8.554688
KNORAU - BagDS	8.851562
KNORAU - DS	9.070312
OLA - MSDS	9.734375
OLA - Fire + MSDS	9.812500
KNORAU - Fire	9.960938
KNORAU - MSDS	10.257812
OLA - BagDS	10.460938
KNORAU - Fire + BagDS	10.468750
OLA - DS	11.828125
Static	12.531250
KNORAE - Fire	14.171875
KNORAE - BagDS	14.179688
KNORAE - Fire + BagDS	15.289062
OLA - Fire + BagDS	15.734375
LCA - MSDS	15.867188
KNORAE - DS	16.164062
LCA - Fire + MSDS	17.148438
OLA - Fire	17.195312
LCA - Fire	17.765625
LCA - Fire + BagDS	18.023438
LCA - BagDS	18.054688
LCA - DS	19.585938

Source: The author (2019)

Table 4 – p-values for the Wilcoxon signed-rank test for noise level 0.1.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	0.2111	0.0743	0.0111	0.9859	0.9116	0.0453
LCA	1.0000	1.0000	0.9994	0.9992	0.9998	1.0000
KNORAE	0.9329	0.7308	0.0006	0.6944	0.9091	0.0000
KNORAU	0.0000	0.0000	0.0018	0.0254	0.0507	0.0000

Source: The author (2019)

At a noise level of 10%, we already observe that the addition of just a bit of noise seems to heavily affect the performance of DS methods, to the point that only KNORAU remains statistically superior to Static Selection. Nevertheless, the KNORAU algorithm occupies 3 of the 5 first positions in the rankings, further indicating that the KNORAU algorithm seems to be highly resilient to noise.

We are also able to observe that the Fire-DES++ seems to be heavily affected by the addition of noise. We posit that one of the main factors resulting in this performance is that this type of synthetic noise heavily affects the ENN algorithm, making it ill-suited to dealing with these types of noisy scenarios, as discussed by (WALMSLEY et al., 2018).

We should also point out that the proposed method of combining Fire-DES++ and MSDS, in conjunction with KNORAU, achieves the highest rank. We will soon compare the results of the proposed method with Dynamic Selection alone and Fire-DES++ usend with Dynamic Selection, but for now we can already point to the ranks as an indicator of the performance of our approach.

Table 5 – Average Ranks for each method, for noise level 0.2.

Method	Avg. Rank
KNORAU - Fire + MSDS	6.140625
OLA - MSDS	7.359375
KNORAU - BagDS	7.585938
KNORAU - DS	7.656250
KNORAU - MSDS	8.203125
KNORAE - MSDS	8.445312
KNORAE - Fire + MSDS	8.945312
OLA - BagDS	9.523438

OLA - Fire + MSDS	10.125000
Static	10.351562
KNORAU - Fire + BagDS	10.718750
KNORAU - Fire	11.500000
OLA - DS	11.843750
LCA - Fire + MSDS	14.210938
LCA - MSDS	14.234375
KNORAE - BagDS	15.132812
KNORAE - Fire + BagDS	15.992188
OLA - Fire + BagDS	17.093750
KNORAE - Fire	17.484375
LCA - Fire + BagDS	17.703125
KNORAE - DS	17.781250
LCA - Fire	18.390625
LCA - BagDS	18.562500
OLA - Fire	19.328125
LCA - DS	20.687500

Source: The author (2019)

Table 6 – p-values for the Wilcoxon signed-rank test for noise level 0.2.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	0.9572	0.5821	0.0514	1.0000	1.0000	0.8167
LCA	1.0000	1.0000	1.0000	1.0000	1.0000	0.9997
KNORAE	1.0000	0.9996	0.2561	1.0000	0.9999	0.5453
KNORAU	0.0000	0.0000	0.0016	0.9507	0.7352	0.0004

Source: The author (2019)

The results at the noise level of 20% are similar to the one observed previously. The trio KNORAU/Fire-DES++/MSDS achieves the highest rank, and only KNORAU maintains an advantage over Static Selection. We will, at the end of this section, elaborate on what we believe differentiates KNORAU from the other DS algorithms, and what explains this performance under noisy scenarios.

At a noise level of 30%, the use of Fire-DES++ is no longer advantageous, as evidenced by Tables 7 and 8. The phenomenon where Fire-DES++ is no longer the best performer can be attributed to the fact that there is a sizable presence of noisy instances in the data set, distributed approximately evenly, or at least as evenly as the original distribution itself. The fact that noisy instances are present everywhere in the data set means that many classifiers will be pruned by the Frienemy pruning process in Fire-DES++, for failing to correct separate instances of different classes. Nevertheless, in any give pair of instances going through the pruning phase, there is a chance that both of them originally belonged to the same class, but one of them had its label flipped by the addition of noise. This will result in an incorrect pruning of classifiers that were actually competent in that region.

Furthermore, since Fire-DES++ relies on the KNN-Equality algorithm, which selects the same number of instances for each class, it could end up selecting noisy instances to make up the region of competence. Instances which have had their label flipped, and which otherwise would not have been selected, could be mistakenly identified as being one of the nearest neighbors. As mentioned previously, this could heavily skew the measurement of classifier competence.

Table 7 – Average Ranks for each method, for noise level 0.3.

Method	Avg. Rank
KNORAU - MSDS	5.632812
KNORAU - DS	5.914062
KNORAU - BagDS	5.945312
KNORAU - Fire + MSDS	6.328125
OLA - MSDS	7.445312
Static	8.398438
KNORAE - MSDS	8.859375
KNORAE - Fire + MSDS	10.218750
OLA - BagDS	10.710938
KNORAU - Fire + BagDS	11.414062
OLA - Fire + MSDS	11.500000
OLA - DS	12.640625

LCA - Fire + MSDS	12.898438
LCA - MSDS	14.132812
KNORAU - Fire	14.867188
KNORAE - BagDS	15.007812
KNORAE - Fire + BagDS	16.382812
KNORAE - DS	16.976562
LCA - Fire + BagDS	17.242188
LCA - BagDS	17.578125
OLA - Fire + BagDS	17.914062
OLA - Fire	18.585938
KNORAE - Fire	19.296875
LCA - DS	19.375000
LCA - Fire	19.734375

Source: The author (2019)

Table 8 – p-values for the Wilcoxon signed-rank test for noise level 0.3.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	1.0000	0.9999	0.8869	1.0000	1.0000	0.9999
LCA	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
KNORAE	1.0000	1.0000	0.9818	1.0000	1.0000	0.9991
KNORAU	0.0006	0.0005	0.0007	1.0000	0.9998	0.5080

Source: The author (2019)

Table 9 – Average Ranks for each method, for noise level 0.4.

Method	Avg. Rank
KNORAU - MSDS	5.289062
KNORAU - BagDS	5.390625
KNORAU - DS	5.781250
Static	6.351562

KNORAU - Fire + MSDS	7.703125
OLA - MSDS	8.882812
KNORAE - MSDS	9.281250
OLA - BagDS	10.343750
KNORAU - Fire + BagDS	10.593750
KNORAE - Fire + MSDS	11.468750
LCA - Fire + MSDS	12.132812
OLA - DS	12.937500
OLA - Fire + MSDS	13.515625
KNORAU - Fire	15.046875
LCA - MSDS	15.187500
KNORAE - BagDS	15.257812
KNORAE - Fire + BagDS	15.953125
OLA - Fire + BagDS	17.007812
LCA - Fire + BagDS	17.179688
KNORAE - DS	17.460938
OLA - Fire	17.710938
LCA - BagDS	17.945312
KNORAE - Fire	18.421875
LCA - DS	19.070312
LCA - Fire	19.085938

Source: The author (2019)

Table 10 – p-values for the Wilcoxon signed-rank test for noise level 0.4.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	1.0000	1.0000	0.9996	1.0000	1.0000	1.0000
LCA	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
KNORAE	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
KNORAU	0.4023	0.1230	0.1531	1.0000	1.0000	0.9947

Source: The author (2019)

Any analysis at a 40% noise level has to be prefaced by the warning that this is an extreme scenario. In fact, it is basically as extreme of a noisy scenario as possible, while still maintaining enough information that it makes sense to attempt binary classification. We say this due to the fact that at 50% noise level, for binary classification (and supposing balanced classes), any instance with a given class label has a 50% chance of actually belonging to the other class. From an Information Theory standpoint, this means that there is no information to be gained about the real label of an instance by observing its possibly corrupted label, and that the entropy of the random variable which describes the class of the instance remains maximal.

That being said, we continue to observe the trend of Static Selection outperforming every variation of the DS methods. In fact, at this noise level, not even the KNORAU algorithm outperforms Static Selection consistently, even though it still offers the highest average rank. This last fact, however, suggest that when Static Selection fails, it does so quite dramatically, while KNORAU has more consistent performance.

4.2.2 Conclusions on the issue of noise vs. DS

The first conclusion we can extract from our results is that noisy scenarios seem to heavily affect the performance of methods based on Dynamic Selection of classifiers. As we have pointed out, this can be explained by the fact that the competence estimation step of Dynamic Selection will be heavily affected by the presence of noise in the region of competence.

This effect can be better visualized by considering the following example. Suppose we are performing Dynamic Selection using the KNORAE algorithm, and denote the query pattern as x_q. Now denote its nearest neighbor in the DSEL set as x_{NN}. Furthermore, imagine that x_q and x_{NN} have the same class label c_i. Now, imagine the class label of x_{NN} is flipped by the addition of noise. Any classifier that would correctly classify it as belonging to c_i would now be considered incompetent. Furthermore, these classifiers would be considered incompetent for any value of the neighborhood size k, since KNORAE only considers competent those classifiers which correctly classify all samples in the region of competence.

This is of course an extreme example, that depends on the specific combination of neighborhood structure, DS algorithm and the specific instances which had noise added to them. Nevertheless, it still illustrates how small changes to the region of competence can have catastrophic effects in the estimation of competence. It also highlights a paradox: while Dynamic Selection's use of local information is its greatest strength, it also makes it susceptible to failure when a particular local region has degraded or incorrect information somehow, particularly considering the fact that we often adopt relatively small sizes for the Region of Competence, such as the value of $k = 7$ used in this work. Static Selection, on the other hand, relies on information that is obtained from a more global perspective. While this global approach precludes the proper identification of classifiers that are experts on a given local region, it also means that we can combine the outputs of the entire pool, instead of risking using only the information present in an improperly pruned pool.

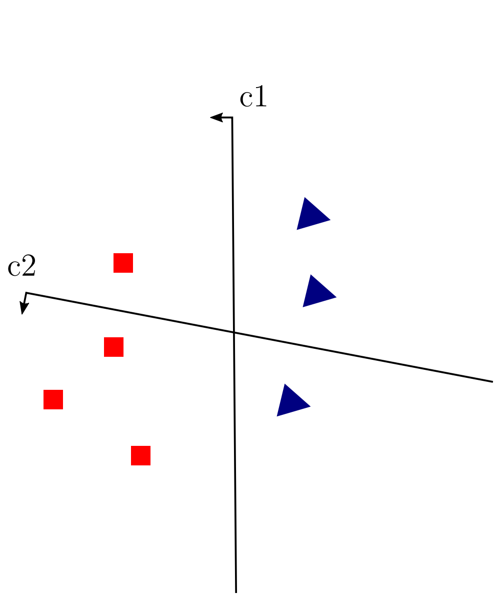
We should also highlight that, in the taxonomy of (FRÉNAY; VERLEYSEN, 2014), the noise model we have adopted would fall under the "noisy completely at random" classification. This model of noise may not be an accurate representation of the type of noise we would expect to encounter in a physical channel, or the type of noise introduced by a human manually labeling examples in a data set. Nevertheless, this model is still useful to our research, and the reasons are twofold.

The first reason is that, for binary classification problems, any kind of label noise will mean that the class label was flipped (0 to 1, or 1 to 0). This means that the noise is actually perfectly determined by the class label, unlike in the scenario of multiple class labels. This still leaves the question of the relationship between the label and the features, but nevertheless, the model becomes more realistic in this case.

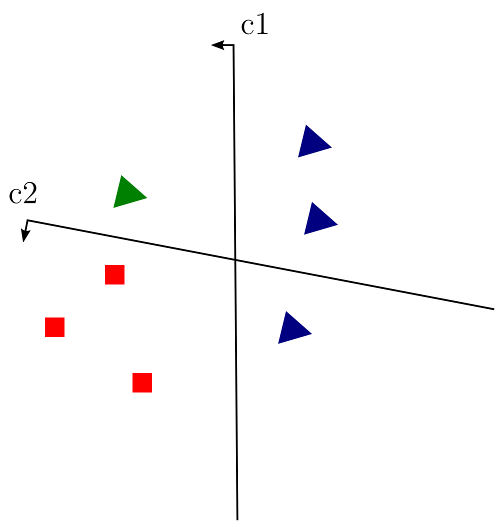
The second reason is that this model actually allows us to observe pathological cases for several algorithms. One of these cases is the aforementioned case involving KNORAE, but we can also consider cases involving for example Dynamic Frienemy Pruning. A very homogeneous region containing only examples of a given class could suddenly contain an instance of the opposite class, due to a label being flipped. This would lead to several classifiers that (correctly) classify all the examples in that region as belonging to the same class being pruned. This would lead to a severe removal of competent classifiers from the pool. Furthermore, a classifier that is not competent in that region could escape pruning simply by classifying correctly one spurious pair. In this manner, our selection of noise model actually allowed us to tease out the different weakness of the algorithms considered.

In spite of the fact that we have talked about a specific failure mode for the KNORAE algorithm, it is not hard to construct equivalent pathological cases for the other algorithms. In fact, for the OLA algorithm, we can envision the case described by Figure 3.

Figure 3 allows us to observe what happens when the label of a single instance is flipped. Originally, c_1 correctly classifies 7 out of 7 instances, while c_2 only correctly classifies 5. But when a single instance has its label flipped, both c_1 and c_2 correctly classify 6 instances. A



(a) Noise-free scenario.



(b) Noisy scenario. The triangle instance to the left of c_1 had its label flipped.

Figure 3 – A simple example of the effect of noise. Two classes are present, the triangle and square classes. The arrow on the tip of the classifiers indicates that all instances in that direction will be attributed to the square class, and instances in the opposition direction will receive the triangle label.

random tie-breaking strategy could result in c_2 being selected as the most competent classifier, when it clearly is not so.

And, considering now the LCA algorithm, it is even easier to devise a scenario where it would fail due to the addition of noise. Suppose a test instance x_q is classified by classifier C_j as belonging to class c_i. The LCA algorithm considers only the instances which C_j attributed to c_i when calculating competence. Let us label the set of those instances as $S_{j,i}$. If any of these instances in $S_{j,i}$ had its class label flipped by the addition noise, LCA would deem that C_j has misclassified this instance, and therefore its competence would be less than it would otherwise have been. Given that $S_{j,i}$ can be much smaller than the number of instances in the Region of Competence, the effect would be even more pronounced than that observed with other algorithms that consider the entire Region of Competence when calculating the competence of classifiers.

Having talked about these pathological cases, we can now discuss how the chosen model of noise allowed us to select a likely winner amongst DS algorithms in this scenario: The KNORAU algorithm. The KNORAU algorithm seems to have robust performance, regardless of noise level (at least considering reasonable noise levels). As we have seen so far, pruning of classifiers due to an incorrect estimation of competence is a grave issue in DS under noisy scenarios. However, the KNORAU algorithm retains any classifier that correctly classifies at least one sample in the region of competence. This means that a classifier that was considered competent in the noise-free scenario is likely to still be considered competent under the noisy scenario, unless the addition of noise means that the classifier no longer correctly classifies any

Table 11 – p-values for the combination of Fire-DES++ and MSDS for all noise levels. The base DS algorithm used was the KNORAU algorithm.

Noise level	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
0	0.0002	0.0427	0.0215	0.0006	1.0000	0.9580
0.1	0.0000	0.0000	0.0001	0.0000	0.0580	0.0036
0.2	0.0004	0.1279	0.1522	0.0973	0.0000	0.0000
0.3	0.5080	0.9951	0.9971	0.9985	0.0000	0.0000
0.4	0.9947	1.0000	1.0000	1.0000	0.0000	0.0000

Source: The author (2019)

samples, which should be a very extreme case.

Turning our attention to the observed behavior of the Fire-DES++ algorithm, and taking into account the fact that the authors in (WALMSLEY et al., 2018) criticize the use of ENN in these types of noisy scenarios, and also accounting for the proposed pathological cases for Dynamic Frienemy Pruning, we can conclude that the Fire-DES++ algorithm is not suitable for these types of scenarios (noisy completely at random). Nevertheless, we were able to observe that by combining KNORAU, Fire-DES++ and MSDS we usually obtained the best results. Therefore, we should not discard Fire-DES++ entirely, but rather recognize how it should be adapted to different scenarios. This allows us to extract the best performance out of an already strong algorithm.

To conclude, we can say that it is hard to prescribe any one single approach for all types of noise and for all noise levels. The only constant seems to be the performance of the KNORAU algorithm under noisy scenarios. Nevertheless, we could recommend with some certainty the use of Fire-DES++ in combination with MSDS to increase the performance of the KNORAU algorithm further.

4.2.3 The performance of MSDS

Having established how the presence of noise affects algorithms based on Dynamic Selection in general, we now turn our attention to the proposed method. In particular, we seek to ascertain in which scenarios is the use of MSDS (or BagDS) advantageous, and also which algorithms are suitable to be combined with our proposed approach. All of our analyses concerning this investigation is focused on the KNORAU algorithm, given that we have already established that it is the most noise-resistant DS algorithm. Table 11 presents these results.

What we can observe is that the best scenario for the combination of Fire-DES++ with

Table 12 – p-values for MSDS alone for all noise levels. The base DS algorithm used was the KNORAU algorithm.

Noise level	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
0	0.0950	0.9637	0.9925	1.0000	0.9990	0.9994
0.1	0.0018	0.9824	0.9726	0.7545	0.5847	1.0000
0.2	0.0016	0.9091	0.8144	0.0040	0.0146	0.9027
0.3	0.0007	0.0288	0.0813	0.0000	0.0000	0.0015
0.4	0.1531	0.0528	0.0961	0.0000	0.0000	0.0000

Source: The author (2019)

KNORAU is the 0.1 noise level. At the 0.2 noise level, we do not manage to significantly outperform KNORAU by itself, in spite of the fact that the combination KNORAU/Fire-DES++/MSDS obtained the highest rank at this noise level.

We should however consider the fact that the Fire-DES++ itself is highly affected by the presence of noise, as we have already discussed. While we can "clean" the data set to some extent, our method is not sufficient to bypass this shortcoming. However, we should try as much as possible to separate the effect of MSDS on performance, from the effect of Fire-DES++. In that vein, we present the results of MSDS by itself, as shown in Table 12.

Here, what we can observe is that MSDS performs the best at a noise level of 0.3. At this point that the performance of Fire-DES++ has degraded beyond any reasonable level, which makes for an easy win. Nevertheless, we also surpass the performance of KNORAU alone. This last fact seems to suggest that MSDS is useful by itself under this scenario, and considering that it works well with Fire-DES++ at lower noise levels, we posit that if we could make MSDS work in better harmony with Fire-DES++, we could possibly achieve a combination that is a winner in a broader range of scenarios.

Considering now the effect of the use of Instance Hardness, we should point out that MSDS seems to be superior to BagDS, at least when used together with Fire-DES++. This is in line with our expectations, and the results of (WALMSLEY et al., 2018), since the use of Instance Hardness in conjunction with Bagging allows for better preserving the structure of the data set, while still managing to remove noise from it.

Finally, we should consider the fact that we based MSDS on Bagging-IH as proposed by (WALMSLEY et al., 2018), a method focused on Static Selection of classifiers, and on classifier pool generation. This means that it is strongest in producing "cleaned-up" versions of the data to be used by algorithms which consider this data in a global manner. In the context of Dynamic Selection, the information will be extracted from each generated bootstrap in a much more local manner, which means that unless we can clean-up local regions very competently, there will be no such things as “strength in numbers”. It is likely necessary to consider further modifications and extensions to Bagging-IH in order to adapt it to Dynamic Selection, and to considering information on a local level. On this note, we can overall consider MSDS a reasonable, if not stellar, approach, with perhaps room for improvement and evolution.

4.2.4 The effect of the number of bags on MSDS

While we have so far discussed the results we obtained by using $m = 50$ for the experiments involving MSDS, we also considered important to discriminate the effect of the number of bags in the performance of MSDS and its combinations. While the full tables for wins, ties and losses are present in our appendices, we can summarize our results by stating that the number of bags mostly does not affect the performance of MSDS.

We do indeed observe some cases in which the result of the Wilcoxon test is statistically significant for one value of m and not for the other, but these mostly concern either the LCA algorithm, which had overall bad performance, or combinations which are not directly comparable, and therefore not really interesting to us, such as KNORAE/MSDS/Fire-DES++ vs KNORAE/BagDS.

A few notable exceptions happen at the noise level of 0.3, where the KNORAE algorithm is defeated by Static Selection at $\alpha = 0.01$ when $m = 10$, but not when $m = 50$, and also KNORAE + MSDS defeats Static Selection at $\alpha = 0.05$ when $m = 50$, but not when $m = 10$. However, KNORAE is not a good algorithm under noisy scenarios, so these changes don’t actually affect the main points of our previous analyses.

5 CONCLUSIONS AND FUTURE WORK

In this work, we set out to investigate the effects of label noise on the performance of Dynamic Selection algorithms, and in particular that of the Fire-DES++ algorithm. We discussed the fundamental ideas of Multiple Classifier Systems in general, and more specifically those of Dynamic Selection. We discussed how issues such as imbalance and noise can affect the performance of classification algorithms, with a particular focus on Ensemble Systems and Dynamic Selection, and what has so far been proposed to deal with this issue.

Building on the Bagging-IH algorithm, we proposed a new algorithm, which we named Multiple-Set Dynamic Selection, which attempts to alleviate the issues caused by the presence of noise in the Dynamic Selection Data Set. Instead of trying to remove noise by performing a hard filtering of the data only once, the algorithm relies on generating multiple Dynamic Selection sets, each filtered in a probabilistic manner, and incorporating Instance Hardness information in the process.

We performed experiments on 64 publicly available data sets. Our results confirm the results present in the literature for the noise-free case, showing not only that DS is a solid choice, but also that Fire-DES++ obtains the highest AUC values. But the results also revealed that the performance of DS algorithms quickly degrades as we add noise to the data, to the point that Static Selection outperforms DS in noisy scenarios. We also managed to identify one exception to this trend, namely the KNORAU algorithm, which seems to derive its robustness for its ability to not incorrectly discard competent classifiers.

Our analysis into the issue of the effect of noise on Dynamic Selection algorithms uncovered a multitude of issues that these algorithms faced under noisy conditions. We have managed to identify failure modes not only of the classic algorithms, but also of the methods that build upon them, such as Fire-DES++. These pathological cases are interesting not only because they allow us to better understand the behavior of classic algorithms, but also because they provide foundations for constructing similar cases for other algorithms. We now have the opportunity not only to analyze how more recent algorithms behave under noisy circumstances, we can also keep theses failure modes in mind when designing new Dynamic Selection algorithms, allowing us to build algorithms that are resilient to these scenarios.

Finally, we were able to identify some scenarios in which the proposed MSDS algorithm offered gains to Dynamic Selection algorithms under noisy scenarios, but it was not possible to tease out a clear trend as to when MSDS would be most useful. We do, however, believe, that we have established that utilizing Instance Hardness and probabilistic filtering is an interesting approach, worthy of further investigation.

5.1 FUTURE WORK

It would be interesting to investigate the behavior of more recent algorithms. We are particularly interested in the META-DES algorithm (CRUZ et al., 2015) and its variants (CRUZ; SABOURIN; CAVALCANTI, 2015), (CRUZ; SABOURIN; CAVALCANTI, 2017b), for its use of meta-learning to estimate the competence of classifiers, and also in the Online Local Pool Generation method of Souza et al. (SOUZA et al., 2018), for its objective of producing locally accurate pools. We would have liked to have also included these algorithms in our analysis, but did not do so due to two main reasons: the first is that their complexity precludes us from performing the failure analysis that we have performed as easily as we would have for the classic algorithms. They require more subtle observations, which are significantly more difficult. The second is simply a matter of combinatorial explosion on the number of comparisons. As the number of algorithms grows, so does the number of comparisons. This creates a problem in extracting insight from the data, and requires much more effort to neatly group and categorize the behavior of the algorithms as we have done. Nevertheless, armed with the findings of this work, we can now attack the problem of understanding how these algorithms function under noisy scenarios much more precisely and productively.

Another inviting line of inquiry is the utilization of Bagging-IH itself to generate the pools used in Dynamic Selection. In this work, we have repeatedly touched on the fact that the degradation of local neighborhoods is a grave issue for Dynamic Selection. Therefore, it is natural to question whether using Bagging-IH as the pool generation algorithm would have affected the results. Nevertheless, one should apply caution when considering this issue. Unlike the original Bagging algorithm, the bootstraps in Bagging-IH will be skewed towards a particular type of instance, namely those considered not noisy. This may result in the loss of information about local neighborhoods that exhibit complex distributions and significant class overlap. The degree to which this may happen is still an open matter, and an important investigation in and of itself. Nevertheless, the issue of the preservation of local neighborhoods is not really a concern in the design of the original Bagging-IH algorithm, since it adapts a more global approach, and adopts a trade-off which risks sacrificing local neighborhoods for an overall cleaner data set. However, DS algorithms rely heavily on local information, so a careful experimental design is necessary to separate the effects of noise during training, and the effects of noise during classifier selection, when using Bagging-IH and MSDS. This is yet another example of how the presence of noise shifts the priorities, and challenges the assumptions of ensemble methods.

REFERENCES

ABELLÁN, J.; MASEGOSA, A. R. Bagging schemes on the presence of class noise in classification. *Expert Systems with Applications*, v. 39, n. 8, p. 6827–6837, 2012. ISSN 09574174.

ABELLÁN, J.; MORAL, S. Building classification trees using the total uncertainty criterion. *International Journal of Intelligent Systems*, v. 18, n. 12, p. 1215–1225, 2003.

ALCALÁ-FDEZ, J.; FERNÁNDEZ, A.; LUENGO, J.; DERRAC, J.; GARCÍA, S.; SÁNCHEZ, L.; HERRERA, F. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, Citeseer, v. 17, 2011.

BREIMAN, L. Bagging predictors. *Machine Learning*, Springer, v. 24, n. 2, p. 123–140, 1996.

BREIMAN, L. Random Forests. *Machine Learning*, v. 45, n. 1, p. 5–32, 2001. ISSN 1573-0565.

BRIGHTON, H.; MELLISH, C. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, v. 6, n. 2, p. 153–172, 2002.

BRITTO, A. S.; SABOURIN, R.; OLIVEIRA, L. E. S. Dynamic selection of classifiers - A comprehensive review. *Pattern Recognition*, v. 47, n. 11, p. 3665–3680, 2014.

CRUZ, R. M.; CAVALCANTI, G. D.; TSANG, I. R. A method for dynamic ensemble selection based on a filter and an adaptive distance to improve the quality of the regions of competence. In: *Proceedings of the International Joint Conference on Neural Networks*. [S.l.: s.n.], 2011. p. 1126–1133. ISBN 9781457710865. ISSN 2161-4393.

CRUZ, R. M.; OLIVEIRA, D. V.; CAVALCANTI, G. D.; SABOURIN, R. FIRE-DES++: Enhanced Online Pruning of Base Classifiers for Dynamic Ensemble Selection. *Pattern Recognition*, Elsevier Ltd, 2018. ISSN 00313203.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. META-DES.H: A Dynamic Ensemble Selection technique using meta-learning and a dynamic weighting approach. *Proceedings of the International Joint Conference on Neural Networks*, v. 2015-September, 2015.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. Analyzing different prototype selection techniques for dynamic classifier and ensemble selection. *Proceedings of the International Joint Conference on Neural Networks*, v. 2017-May, p. 3959–3966, 2017. ISSN 09410643.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. META-DES.Oracle: Meta-learning and feature selection for dynamic ensemble selection. *Information Fusion*, Elsevier B.V., v. 38, p. 84–103, 2017. ISSN 15662535.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, Elsevier B.V., v. 41, p. 195–216, 2018. ISSN 15662535.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D.; Ing Ren, T. META-DES: A dynamic ensemble selection framework using meta-learning. *Pattern Recognition*, Elsevier, v. 48, n. 5, p. 1925–1935, 2015. ISSN 00313203.

Cruz, R. M. O.; Cavalcanti, G. D. C.; Ren, T. I. An ensemble classifier for offline cursive character recognition using multiple feature extraction techniques. In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2010. p. 1–8. ISSN 1098-7576.

CRUZ, R. M. O.; HAFEMANN, L. G.; SABOURIN, R.; CAVALCANTI, G. D. C. DESlib: A Dynamic ensemble selection library in Python. *arXiv preprint arXiv:1802.04967*, 2018.

CRUZ, R. M. O.; SABOURIN, R.; CAVALCANTI, G. D. C. Prototype selection for dynamic classifier and ensemble selection. *Neural Computing and Applications*, v. 29, n. 2, p. 447–457, Jan 2018. ISSN 1433-3058.

DEB, K. *Multi-objective optimization using evolutionary algorithms*. [S.l.]: John Wiley & Sons, 2001.

DIDACI, L.; GIACINTO, G.; ROLI, F.; MARCIALIS, G. L. A study on the performances of dynamic classifier selection based on local accuracy estimation. *Pattern Recognition*, v. 38, n. 11, p. 2188 – 2191, 2005. ISSN 0031-3203.

DIETTERICH, T. G. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and ranomization. *Machine Learning*, v. 40, n. 2, p. 139–157, 2000.

ESHELMAN, L. J. The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In: RAWLINS, G. J. (Ed.). [S.l.]: Elsevier, 1991, (Foundations of Genetic Algorithms, v. 1). p. 265 – 283.

FRÉNAY, B.; VERLEYSEN, M. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, v. 25, n. 5, p. 845–869, 2014. ISSN 21622388.

FREUND, Y.; SCHAPIRE, R. E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, v. 55, n. 1, p. 119–139, 1997. ISSN 0022-0000.

GARCÍA, S.; CANO, J. R.; HERRERA, F. A memetic algorithm for evolutionary prototype selection: A scaling up approach. *Pattern Recognition*, v. 41, n. 8, p. 2693–2709, 2008. ISSN 00313203.

GARCÍA, S.; DERRAC, J.; CANO, J. R.; HERRERA, F. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 34, n. 3, p. 417–435, 2012. ISSN 01628828.

GIACINTO, G.; ROLI, F. Methods for dynamic classifier selection. In: *Proceedings 10th International Conference on Image Analysis and Processing*. [S.l.: s.n.], 1999. p. 659–664.

HO, T. K. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, n. 8, p. 832–844, 1998. ISSN 01628828.

HUANG, Y.; SUEN, C. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 17, n. 1, p. 90–94, Jan 1995. ISSN 1939-3539.

KHOSHGOFTAAR, T. M.; HULSE, J. V.; NAPOLITANO, A. Comparing boosting and bagging techniques with noisy and imbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, IEEE, v. 41, n. 3, p. 552–568, 2011.

KITTLER, J. Combining classifiers: A theoretical framework. *Pattern Analysis and Applications*, v. 1, n. 1, p. 18–27, 1998. ISSN 14337541.

KO, A. H.; SABOURIN, R.; BRITTO, A. S. From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition*, v. 41, n. 5, p. 1735–1748, 2008. ISSN 00313203.

KUNCHEVA, L. 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)*. [S.l.: s.n.], 2000. v. 1, p. 185–188 vol.1. ISSN null.

KUNCHEVA, L. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 2, p. 281–286, Feb 2002. ISSN 1939-3539.

KUNCHEVA, L. I. Editing for the k-nearest neighbors rule by a genetic algorithm. *Pattern Recognition Letters*, v. 16, n. 8, p. 809–814, 1995. ISSN 0167-8655.

KUNCHEVA, L. I. *Combining Pattern Classifiers: Methods and Algorithms*. 2nd. ed. [S.l.]: Wiley Publishing, 2014. ISBN 1118315235, 9781118315231.

KUNCHEVA, L. I.; JAIN, L. C. Nearest neighbor classifier: Simultaneous editing and feature selection. *Pattern recognition letters*, Elsevier, v. 20, n. 11-13, p. 1149–1156, 1999.

LAOZI. *The Tao The Ching*. Tradução de Stephen Mitchell. [S.l.]: Harper Perennial Modern Classics, 2006. Paginação irregular.

LONG, P. M.; SERVEDIO, R. A. Random classification noise defeats all convex potential boosters. *Machine Learning*, v. 78, n. 3, p. 287–304, 2009. ISSN 15730565.

MANSILLA, E. B.; HO, T. K. On classifier domains of competence. *Proceedings - International Conference on Pattern Recognition*, v. 1, p. 136–139, 2004.

MELVILLE, P.; SHAH, N.; MIHALKOVA, L.; MOONEY, R. J. Experiments on ensembles with missing and noisy data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 3077, p. 293–302, 2004. ISSN 03029743.

OLIVEIRA, D. V.; CAVALCANTI, G. D.; SABOURIN, R. Online pruning of base classifiers for Dynamic Ensemble Selection. *Pattern Recognition*, Elsevier Ltd, v. 72, p. 44–58, dec 2017. ISSN 00313203.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

QUINLAN, J. R. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN 1-55860-238-0.

SÁNCHEZ, J.; PLA, F.; FERRI, F. Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognition Letters*, v. 18, n. 6, p. 507 – 513, 1997. ISSN 0167-8655.

SCHAPIRE, R. E. A brief introduction to boosting. In: *IJCAI International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 1999. v. 2, p. 1401–1406.

SHIPP, C. A.; KUNCHEVA, L. I. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion*, v. 3, n. 2, p. 135–148, 2002. ISSN 15662535.

SIERRA, B.; LAZKANO, E.; IRIGOIEN, I.; JAUREGI, E.; MENDIALDUA, I. K nearest neighbor equality: Giving equal chance to all existing classes. *Inf. Sci.*, Elsevier Science Inc., New York, NY, USA, v. 181, n. 23, p. 5158–5168, dez. 2011. ISSN 0020-0255.

SKALAK, D. B. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In: *Machine Learning: Proceedings of the Eleventh International Conference*. [S.l.]: Morgan Kaufmann, 1994. p. 293–301.

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

SOUZA, M. A.; CAVALCANTI, G. D.; CRUZ, R. M.; SABOURIN, R. On the characterization of the Oracle for dynamic classifier selection. *Proceedings of the International Joint Conference on Neural Networks*, v. 2017-May, p. 332–339, 2017.

SOUZA, M. A.; CAVALCANTI, G. D.; CRUZ, R. M.; SABOURIN, R. Online Local Pool Generation for Dynamic Classifier Selection. *Pattern Recognition*, Elsevier Ltd, 2018. ISSN 00313203.

WALMSLEY, F. N.; CAVALCANTI, G. D.; OLIVEIRA, D. V.; CRUZ, R. M.; SABOURIN, R. An Ensemble Generation Method Based on Instance Hardness. In: *Proceedings of the International Joint Conference on Neural Networks*. [S.l.: s.n.], 2018. v. 2018-July. ISBN 9781509060146.

WANG, J.; NESKOVIC, P.; COOPER, L. N. Improving nearest neighbor rule with a simple adaptive distance measure. *Pattern Recognition Letters*, v. 28, n. 2, p. 207–213, 2007. ISSN 0167-8655.

WILCOXON, F. Individual comparisons by ranking methods. *Biometrics Bulletin*, [International Biometric Society, Wiley], v. 1, n. 6, p. 80–83, 1945. ISSN 00994987.

WILSON, D. L. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2, n. 3, p. 408–421, July 1972. ISSN 2168-2909.

WILSON, D. R.; MARTINEZ, T. R. Reduction Techniques for Instance-Based Learning Algorithms. *Machine Learning*, v. 38, n. 3, p. 257–286, mar 2000. ISSN 1573-0565.

Woloszynski, T.; Kurzynski, M. A measure of competence based on randomized reference classifier for dynamic ensemble selection. In: *2010 20th International Conference on Pattern Recognition*. [S.l.: s.n.], 2010. p. 4194–4197. ISSN 1051-4651.

WOLOSZYNSKI, T.; KURZYNSKI, M. A probabilistic model of classifier competence for dynamic ensemble selection. *Pattern Recognition*, v. 44, n. 10, p. 2656 – 2668, 2011. ISSN 0031-3203. Semi-Supervised Learning for Visual Content Analysis and Understanding.

WOODS, K.; KEGELMEYER, W. P.; BOWYER, K. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 19, n. 4, p. 405–410, 1997. ISSN 01628828.

WOŹNIAK, M.; GRAÑA, M.; CORCHADO, E. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, v. 16, p. 3–17, 2014. ISSN 1566-2535.

ZHOU, Z.-H. *Ensemble Methods: Foundations and Algorithms*. [S.l.]: CRC Press, 2012. ISSN 1098-6596. ISBN 978-1-4398-3003-1.

APPENDIX A – TABLES - 10 BAGS

A.1 NOISE LEVEL: 0

A.1.1 Average Ranks

Method	Avg. Rank
KNORAE - Fire	7.070312
KNORAU - Fire	8.343750
KNORAU - Fire + BagDS	10.171875
KNORAE - Fire + MSDS	10.453125
KNORAE - BagDS	10.546875
OLA - Fire	10.710938
KNORAU - Fire + MSDS	10.875000
KNORAU - BagDS	11.109375
KNORAE - DS	11.218750
KNORAE - MSDS	11.476562
KNORAE - Fire + BagDS	11.546875
KNORAU - DS	11.578125
OLA - BagDS	11.632812
OLA - DS	11.734375
KNORAU - MSDS	12.265625
Static	13.265625
OLA - Fire + BagDS	13.445312
OLA - Fire + MSDS	14.179688
OLA - MSDS	14.851562
LCA - BagDS	16.835938
LCA - Fire + BagDS	18.070312
LCA - DS	18.078125
LCA - MSDS	18.296875
LCA - Fire	18.578125
LCA - Fire + MSDS	18.664062

Table 13 – Average Ranks for each method, for noise level 0.

A.1.2 Win/Tie/Loss comparisons

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	32 / 4 / 28	33 / 5 / 26	26 / 4 / 34	34 / 3 / 27	26 / 3 / 35	25 / 3 / 36
LCA	18 / 4 / 42	21 / 4 / 39	17 / 5 / 42	21 / 4 / 39	20 / 5 / 39	19 / 3 / 42
KNORAE	32 / 9 / 23	32 / 10 / 22	27 / 16 / 21	40 / 8 / 16	32 / 9 / 23	32 / 14 / 18
KNORAU	26 / 29 / 9	27 / 30 / 7	22 / 30 / 12	34 / 17 / 13	29 / 22 / 13	25 / 29 / 10

Table 14 – Wins, ties and losses for noise level 0.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	0.0211	0.0212	0.5147	0.0141	0.1290	0.2889
LCA	0.9997	0.9988	0.9999	0.9941	0.9932	0.9987
KNORAE	0.0177	0.0032	0.1477	0.0000	0.0097	0.0046
KNORAU	0.0036	0.0003	0.0844	0.0000	0.0001	0.0002

Table 15 – p-values for the Wilcoxon signed-rank test for noise level 0.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	28 / 4 / 32	25 / 14 / 25	17 / 10 / 37	30 / 3 / 31	21 / 6 / 37	22 / 7 / 35
LCA	42 / 4 / 18	36 / 18 / 10	25 / 8 / 31	31 / 6 / 27	34 / 6 / 24	35 / 4 / 25
KNORAE	23 / 9 / 32	23 / 26 / 15	28 / 11 / 25	40 / 7 / 17	22 / 8 / 34	30 / 10 / 24
KNORAU	9 / 29 / 26	15 / 40 / 9	10 / 34 / 20	28 / 16 / 20	23 / 20 / 21	21 / 25 / 18

Table 16 – Wins, ties and losses for noise level 0.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	0.9789	0.8115	0.9998	0.0389	0.8123	0.9871
LCA	0.0003	0.0026	0.8733	0.2454	0.0450	0.2007
KNORAE	0.9823	0.0333	0.7898	0.0000	0.2868	0.5052
KNORAU	0.9964	0.2375	0.9891	0.0002	0.0076	0.0338

Table 17 – p-values for the Wilcoxon signed-rank test for noise level 0.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	26 / 4 / 34	17 / 10 / 37	16 / 10 / 38	20 / 5 / 39	23 / 7 / 34	23 / 13 / 28
LCA	17 / 5 / 42	25 / 8 / 31	20 / 10 / 34	31 / 3 / 30	27 / 4 / 33	31 / 2 / 31
KNORAE	27 / 16 / 21	28 / 11 / 25	26 / 13 / 25	18 / 6 / 40	31 / 6 / 27	22 / 17 / 25
KNORAU	22 / 30 / 12	10 / 34 / 20	6 / 35 / 23	15 / 17 / 32	18 / 23 / 23	15 / 26 / 23

Table 18 – Wins, ties and losses for noise level 0.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	0.5147	0.9998	0.9999	1.0000	0.9947	0.9689
LCA	0.9999	0.8733	0.9955	0.7946	0.9398	0.8394
KNORAE	0.1477	0.7898	0.9236	1.0000	0.7840	0.9506
KNORAU	0.0844	0.9891	0.9995	1.0000	0.9979	0.9988

Table 19 – p-values for the Wilcoxon signed-rank test for noise level 0.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	25 / 3 / 36	22 / 7 / 35	23 / 8 / 33	28 / 13 / 23	16 / 4 / 44	21 / 10 / 33
LCA	19 / 3 / 42	35 / 4 / 25	25 / 5 / 34	31 / 2 / 31	27 / 9 / 28	22 / 14 / 28
KNORAE	32 / 14 / 18	30 / 10 / 24	25 / 13 / 26	25 / 17 / 22	15 / 6 / 43	29 / 11 / 24
KNORAU	25 / 29 / 10	21 / 25 / 18	21 / 25 / 18	23 / 26 / 15	13 / 20 / 31	16 / 25 / 23

Table 20 – Wins, ties and losses for noise level 0.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	0.2889	0.9871	0.9649	0.0311	1.0000	0.9688
LCA	0.9987	0.2007	0.6358	0.1606	0.5100	0.8727
KNORAE	0.0046	0.5052	0.7396	0.0494	1.0000	0.6013
KNORAU	0.0002	0.0338	0.0457	0.0012	1.0000	0.9652

Table 21 – p-values for the Wilcoxon signed-rank test for noise level 0.

A.2 NOISE LEVEL: 0.1

A.2.1 Average Ranks

Method	Avg. Rank
KNORAU - Fire + MSDS	6.765625
KNORAE - Fire + MSDS	7.281250
KNORAE - MSDS	8.617188
KNORAU - BagDS	8.875000
KNORAU - DS	9.046875
OLA - MSDS	9.421875
KNORAU - Fire	9.929688
OLA - Fire + MSDS	10.078125
KNORAU - Fire + BagDS	10.156250
OLA - BagDS	10.250000
KNORAU - MSDS	10.523438
OLA - DS	11.546875
Static	12.593750
KNORAE - BagDS	13.773438
KNORAE - Fire	14.000000
KNORAE - Fire + BagDS	15.914062
LCA - MSDS	15.953125
KNORAE - DS	15.992188
OLA - Fire + BagDS	16.109375

OLA - Fire	17.101562
LCA - Fire	17.328125
LCA - Fire + MSDS	17.500000
LCA - BagDS	18.140625
LCA - Fire + BagDS	18.468750
LCA - DS	19.632812

Table 22 – Average Ranks for each method, for noise level 0.1.

A.2.2 Win/Tie/Loss comparisons

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	36 / 0 / 28	33 / 0 / 31	38 / 0 / 26	22 / 0 / 42	24 / 0 / 40	38 / 0 / 26
LCA	12 / 0 / 52	12 / 0 / 52	19 / 1 / 44	18 / 0 / 46	17 / 0 / 47	18 / 0 / 46
KNORAE	22 / 0 / 42	26 / 0 / 38	42 / 2 / 20	28 / 0 / 36	22 / 1 / 41	44 / 2 / 18
KNORAU	48 / 5 / 11	49 / 5 / 10	42 / 5 / 17	33 / 0 / 31	36 / 1 / 27	52 / 0 / 12

Table 23 – Wins, ties and losses for noise level 0.1.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	0.2111	0.0466	0.0154	0.9859	0.9209	0.0466
LCA	1.0000	1.0000	0.9994	0.9992	1.0000	1.0000
KNORAE	0.9329	0.7219	0.0006	0.6944	0.9415	0.0000
KNORAU	0.0000	0.0000	0.0039	0.0254	0.0585	0.0000

Table 24 – p-values for the Wilcoxon signed-rank test for noise level 0.1.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	28 / 0 / 36	44 / 0 / 20	45 / 0 / 19	18 / 0 / 46	15 / 0 / 49	40 / 0 / 24
LCA	52 / 0 / 12	49 / 0 / 15	51 / 0 / 13	38 / 0 / 26	38 / 0 / 26	40 / 0 / 24
KNORAE	42 / 0 / 22	52 / 0 / 12	50 / 0 / 14	39 / 0 / 25	30 / 1 / 33	55 / 0 / 9
KNORAU	11 / 5 / 48	20 / 25 / 19	21 / 16 / 27	29 / 0 / 35	28 / 1 / 35	39 / 1 / 24

Table 25 – Wins, ties and losses for noise level 0.1.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	0.7889	0.0031	0.0116	1.0000	1.0000	0.0155
LCA	0.0000	0.0000	0.0000	0.0063	0.0173	0.0156
KNORAE	0.0671	0.0000	0.0000	0.0715	0.3533	0.0000
KNORAU	1.0000	0.8141	0.9953	0.3868	0.7080	0.0001

Table 26 – p-values for the Wilcoxon signed-rank test for noise level 0.1.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	38 / 0 / 26	45 / 0 / 19	38 / 0 / 26	48 / 0 / 16	46 / 0 / 18	38 / 0 / 26
LCA	19 / 1 / 44	51 / 0 / 13	48 / 2 / 14	35 / 0 / 29	41 / 0 / 23	41 / 2 / 21
KNORAE	42 / 2 / 20	50 / 0 / 14	46 / 0 / 18	47 / 0 / 17	50 / 0 / 14	30 / 1 / 33
KNORAU	42 / 5 / 17	21 / 16 / 27	22 / 14 / 28	34 / 0 / 30	34 / 0 / 30	22 / 1 / 41

Table 27 – Wins, ties and losses for noise level 0.1.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	0.0154	0.0116	0.2035	0.0000	0.0003	0.1469
LCA	0.9994	0.0000	0.0000	0.2189	0.0550	0.0130
KNORAE	0.0006	0.0000	0.0001	0.0025	0.0000	0.9439
KNORAU	0.0039	0.9953	0.9966	0.8185	0.6608	1.0000

Table 28 – p-values for the Wilcoxon signed-rank test for noise level 0.1.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	38 / 0 / 26	40 / 0 / 24	34 / 0 / 30	26 / 0 / 38	51 / 0 / 13	53 / 0 / 11
LCA	18 / 0 / 46	40 / 0 / 24	37 / 2 / 25	21 / 2 / 41	37 / 0 / 27	42 / 0 / 22
KNORAE	44 / 2 / 18	55 / 0 / 9	53 / 0 / 11	33 / 1 / 30	50 / 0 / 14	53 / 0 / 11
KNORAU	52 / 0 / 12	39 / 1 / 24	38 / 1 / 25	41 / 1 / 22	41 / 1 / 22	46 / 0 / 18

Table 29 – Wins, ties and losses for noise level 0.1.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	0.0466	0.0155	0.3792	0.8531	0.0000	0.0000
LCA	1.0000	0.0156	0.1876	0.9870	0.4310	0.1143
KNORAE	0.0000	0.0000	0.0000	0.0561	0.0000	0.0000
KNORAU	0.0000	0.0001	0.0004	0.0000	0.0678	0.0023

Table 30 – p-values for the Wilcoxon signed-rank test for noise level 0.1.

A.3 NOISE LEVEL: 0.2

A.3.1 Average Ranks

Method	Avg. Rank
KNORAU - Fire + MSDS	6.179688
KNORAU - BagDS	7.218750
KNORAU - DS	7.710938
KNORAU - MSDS	7.812500
OLA - MSDS	7.953125
KNORAE - MSDS	9.039062
KNORAE - Fire + MSDS	9.140625
OLA - Fire + MSDS	9.820312
OLA - BagDS	9.843750
Static	10.226562
KNORAU - Fire + BagDS	10.585938
KNORAU - Fire	11.562500
OLA - DS	11.765625
LCA - Fire + MSDS	14.265625
LCA - MSDS	14.453125
KNORAE - BagDS	14.960938
OLA - Fire + BagDS	15.796875
KNORAE - Fire + BagDS	16.679688
LCA - Fire + BagDS	17.460938

KNORAE - DS	17.640625
KNORAE - Fire	17.750000
LCA - Fire	18.437500
LCA - BagDS	18.789062
OLA - Fire	19.296875
LCA - DS	20.609375

Table 31 – Average Ranks for each method, for noise level 0.2.

A.3.2 Win/Tie/Loss comparisons

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	23 / 0 / 41	24 / 1 / 39	31 / 0 / 33	13 / 0 / 51	18 / 0 / 46	24 / 0 / 40
LCA	7 / 0 / 57	11 / 0 / 53	17 / 0 / 47	16 / 0 / 48	15 / 0 / 49	18 / 0 / 46
KNORAE	18 / 0 / 46	19 / 0 / 45	30 / 0 / 34	16 / 0 / 48	17 / 0 / 47	28 / 0 / 36
KNORAU	48 / 1 / 15	49 / 2 / 13	44 / 1 / 19	28 / 0 / 36	29 / 0 / 35	45 / 0 / 19

Table 32 – Wins, ties and losses for noise level 0.2.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	0.9572	0.7639	0.1500	1.0000	0.9999	0.8058
LCA	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999
KNORAE	1.0000	0.9998	0.3343	1.0000	1.0000	0.4231
KNORAU	0.0000	0.0000	0.0004	0.9507	0.7151	0.0003

Table 33 – p-values for the Wilcoxon signed-rank test for noise level 0.2.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	41 / 0 / 23	45 / 0 / 19	51 / 0 / 13	10 / 0 / 54	16 / 0 / 48	32 / 0 / 32
LCA	57 / 0 / 7	53 / 0 / 11	55 / 0 / 9	39 / 0 / 25	43 / 0 / 21	48 / 0 / 16
KNORAE	46 / 0 / 18	55 / 1 / 8	54 / 0 / 10	30 / 0 / 34	34 / 0 / 30	60 / 0 / 4
KNORAU	15 / 1 / 48	30 / 12 / 22	33 / 3 / 28	23 / 0 / 41	24 / 0 / 40	36 / 1 / 27

Table 34 – Wins, ties and losses for noise level 0.2.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	0.0428	0.0007	0.0000	1.0000	1.0000	0.0781
LCA	0.0000	0.0000	0.0000	0.0173	0.0002	0.0000
KNORAE	0.0000	0.0000	0.0000	0.7396	0.0345	0.0000
KNORAU	1.0000	0.1582	0.5867	0.9994	0.9939	0.1115

Table 35 – p-values for the Wilcoxon signed-rank test for noise level 0.2.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	31 / 0 / 33	51 / 0 / 13	46 / 0 / 18	55 / 0 / 9	49 / 0 / 15	42 / 0 / 22
LCA	17 / 0 / 47	55 / 0 / 9	54 / 0 / 10	42 / 0 / 22	38 / 0 / 26	35 / 0 / 29
KNORAE	30 / 0 / 34	54 / 0 / 10	53 / 0 / 11	50 / 0 / 14	50 / 0 / 14	39 / 1 / 24
KNORAU	44 / 1 / 19	33 / 3 / 28	33 / 4 / 27	40 / 0 / 24	39 / 0 / 25	34 / 0 / 30

Table 36 – Wins, ties and losses for noise level 0.2.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	0.1500	0.0000	0.0001	0.0000	0.0000	0.0013
LCA	1.0000	0.0000	0.0000	0.0002	0.0164	0.5977
KNORAE	0.3343	0.0000	0.0000	0.0000	0.0000	0.1236
KNORAU	0.0004	0.5867	0.7578	0.0010	0.0083	0.7587

Table 37 – p-values for the Wilcoxon signed-rank test for noise level 0.2.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	24 / 0 / 40	32 / 0 / 32	29 / 0 / 35	22 / 0 / 42	62 / 0 / 2	52 / 0 / 12
LCA	18 / 0 / 46	48 / 0 / 16	45 / 0 / 19	29 / 0 / 35	48 / 0 / 16	49 / 0 / 15
KNORAE	28 / 0 / 36	60 / 0 / 4	56 / 0 / 8	24 / 1 / 39	56 / 0 / 8	56 / 0 / 8
KNORAU	45 / 0 / 19	36 / 1 / 27	30 / 0 / 34	30 / 0 / 34	50 / 0 / 14	49 / 0 / 15

Table 38 – Wins, ties and losses for noise level 0.2.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	0.8058	0.0781	0.6003	0.9987	0.0000	0.0000
LCA	0.9999	0.0000	0.0000	0.4023	0.0000	0.0000
KNORAE	0.4231	0.0000	0.0000	0.8764	0.0000	0.0000
KNORAU	0.0003	0.1115	0.4231	0.2413	0.0000	0.0000

Table 39 – p-values for the Wilcoxon signed-rank test for noise level 0.2.

A.4 NOISE LEVEL: 0.3

A.4.1 Average Ranks

Method	Avg. Rank
KNORAU - BagDS	5.335938
KNORAU - MSDS	5.828125
KNORAU - DS	5.914062
KNORAU - Fire + MSDS	6.718750
OLA - MSDS	7.507812
Static	8.507812
KNORAE - MSDS	8.875000
OLA - BagDS	10.203125
KNORAE - Fire + MSDS	10.804688
KNORAU - Fire + BagDS	11.125000
LCA - Fire + MSDS	12.195312
OLA - DS	12.625000
OLA - Fire + MSDS	12.703125
LCA - MSDS	14.437500
KNORAE - BagDS	14.718750
KNORAU - Fire	14.890625
KNORAE - Fire + BagDS	16.140625
KNORAE - DS	16.867188
LCA - Fire + BagDS	17.054688

OLA - Fire + BagDS	17.273438
LCA - BagDS	17.867188
OLA - Fire	18.812500
KNORAE - Fire	19.390625
LCA - DS	19.531250
LCA - Fire	19.671875

Table 40 – Average Ranks for each method, for noise level 0.3.

A.4.2 Win/Tie/Loss comparisons

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	18 / 0 / 46	22 / 0 / 42	27 / 0 / 37	10 / 0 / 54	13 / 0 / 51	22 / 0 / 42
LCA	7 / 0 / 57	9 / 0 / 55	14 / 0 / 50	8 / 0 / 56	13 / 0 / 51	19 / 0 / 45
KNORAE	13 / 0 / 51	14 / 0 / 50	23 / 0 / 41	7 / 0 / 57	15 / 0 / 49	20 / 0 / 44
KNORAU	44 / 1 / 19	47 / 0 / 17	46 / 0 / 18	15 / 0 / 49	25 / 0 / 39	29 / 0 / 35

Table 41 – Wins, ties and losses for noise level 0.3.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	1.0000	0.9998	0.9095	1.0000	1.0000	1.0000
LCA	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999
KNORAE	1.0000	1.0000	0.9776	1.0000	1.0000	0.9994
KNORAU	0.0006	0.0000	0.0005	1.0000	0.9994	0.6944

Table 42 – p-values for the Wilcoxon signed-rank test for noise level 0.3.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	46 / 0 / 18	40 / 0 / 24	54 / 0 / 10	11 / 0 / 53	14 / 0 / 50	29 / 0 / 35
LCA	57 / 0 / 7	50 / 2 / 12	55 / 0 / 9	34 / 0 / 30	45 / 0 / 19	53 / 0 / 11
KNORAE	51 / 0 / 13	49 / 0 / 15	56 / 0 / 8	20 / 0 / 44	39 / 0 / 25	54 / 0 / 10
KNORAU	19 / 1 / 44	44 / 1 / 19	38 / 0 / 26	11 / 0 / 53	11 / 0 / 53	20 / 0 / 44

Table 43 – Wins, ties and losses for noise level 0.3.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	0.0000	0.0039	0.0000	1.0000	1.0000	0.6029
LCA	0.0000	0.0000	0.0000	0.2963	0.0004	0.0000
KNORAE	0.0000	0.0001	0.0000	0.9993	0.0210	0.0000
KNORAU	0.9994	0.0007	0.0653	1.0000	1.0000	0.9990

Table 44 – p-values for the Wilcoxon signed-rank test for noise level 0.3.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	27 / 0 / 37	54 / 0 / 10	49 / 0 / 15	60 / 0 / 4	57 / 0 / 7	52 / 0 / 12
LCA	14 / 0 / 50	55 / 0 / 9	54 / 0 / 10	43 / 0 / 21	38 / 0 / 26	25 / 0 / 39
KNORAE	23 / 0 / 41	56 / 0 / 8	56 / 0 / 8	57 / 0 / 7	55 / 0 / 9	48 / 0 / 16
KNORAU	46 / 0 / 18	38 / 0 / 26	34 / 0 / 30	52 / 0 / 12	48 / 0 / 16	42 / 0 / 22

Table 45 – Wins, ties and losses for noise level 0.3.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	0.9095	0.0000	0.0000	0.0000	0.0000	0.0000
LCA	1.0000	0.0000	0.0000	0.0000	0.0387	0.9725
KNORAE	0.9776	0.0000	0.0000	0.0000	0.0000	0.0000
KNORAU	0.0005	0.0653	0.5373	0.0000	0.0000	0.0010

Table 46 – p-values for the Wilcoxon signed-rank test for noise level 0.3.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	22 / 0 / 42	29 / 0 / 35	18 / 0 / 46	12 / 0 / 52	52 / 0 / 12	51 / 0 / 13
LCA	19 / 0 / 45	53 / 0 / 11	48 / 0 / 16	39 / 0 / 25	55 / 0 / 9	50 / 0 / 14
KNORAE	20 / 0 / 44	54 / 0 / 10	51 / 1 / 12	16 / 0 / 48	60 / 0 / 4	55 / 0 / 9
KNORAU	29 / 0 / 35	20 / 0 / 44	20 / 0 / 44	22 / 0 / 42	54 / 0 / 10	53 / 0 / 11

Table 47 – Wins, ties and losses for noise level 0.3.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	1.0000	0.6029	0.9969	1.0000	0.0000	0.0000
LCA	0.9999	0.0000	0.0000	0.0275	0.0000	0.0000
KNORAE	0.9994	0.0000	0.0000	1.0000	0.0000	0.0000
KNORAU	0.6944	0.9990	0.9999	0.9990	0.0000	0.0000

Table 48 – p-values for the Wilcoxon signed-rank test for noise level 0.3.

A.5 NOISE LEVEL: 0.4

A.5.1 Average Ranks

Method	Avg. Rank
KNORAU - MSDS	5.179688
KNORAU - BagDS	5.609375
KNORAU - DS	5.710938
Static	6.445312
KNORAU - Fire + MSDS	7.281250
KNORAE - MSDS	8.703125
OLA - MSDS	8.992188
KNORAE - Fire + MSDS	11.226562
KNORAU - Fire + BagDS	11.335938
OLA - BagDS	11.500000
LCA - Fire + MSDS	12.046875
OLA - DS	12.945312
OLA - Fire + MSDS	13.382812
KNORAE - BagDS	14.937500
LCA - MSDS	14.968750
KNORAU - Fire	15.125000
LCA - Fire + BagDS	16.328125
KNORAE - Fire + BagDS	16.484375
OLA - Fire + BagDS	17.109375

KNORAE - DS	17.445312
OLA - Fire	17.679688
LCA - BagDS	17.867188
KNORAE - Fire	18.468750
LCA - Fire	19.070312
LCA - DS	19.156250

Table 49 – Average Ranks for each method, for noise level 0.4.

A.5.2 Win/Tie/Loss comparisons

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	13 / 0 / 51	17 / 0 / 47	23 / 0 / 41	7 / 0 / 57	10 / 0 / 54	14 / 0 / 50
LCA	5 / 0 / 59	6 / 0 / 58	11 / 0 / 53	5 / 0 / 59	12 / 0 / 52	15 / 0 / 49
KNORAE	7 / 0 / 57	8 / 0 / 56	20 / 0 / 44	7 / 0 / 57	8 / 0 / 56	13 / 0 / 51
KNORAU	30 / 0 / 34	30 / 1 / 33	32 / 0 / 32	9 / 0 / 55	18 / 0 / 46	28 / 0 / 36

Table 50 – Wins, ties and losses for noise level 0.4.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	1.0000	1.0000	0.9999	1.0000	1.0000	1.0000
LCA	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
KNORAE	1.0000	1.0000	0.9998	1.0000	1.0000	1.0000
KNORAU	0.4023	0.1604	0.1531	1.0000	1.0000	0.9786

Table 51 – p-values for the Wilcoxon signed-rank test for noise level 0.4.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	51 / 0 / 13	43 / 0 / 21	47 / 0 / 17	12 / 0 / 52	15 / 0 / 49	29 / 0 / 35
LCA	59 / 0 / 5	53 / 1 / 10	58 / 0 / 6	36 / 0 / 28	45 / 0 / 19	50 / 0 / 14
KNORAE	57 / 0 / 7	49 / 1 / 14	57 / 0 / 7	27 / 0 / 37	36 / 1 / 27	53 / 0 / 11
KNORAU	34 / 0 / 30	38 / 0 / 26	35 / 0 / 29	7 / 0 / 57	10 / 1 / 53	18 / 0 / 46

Table 52 – Wins, ties and losses for noise level 0.4.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	0.0000	0.0204	0.0000	1.0000	1.0000	0.7791
LCA	0.0000	0.0000	0.0000	0.1131	0.0003	0.0000
KNORAE	0.0000	0.0000	0.0000	0.9248	0.0281	0.0000
KNORAU	0.5977	0.1547	0.0715	1.0000	1.0000	0.9999

Table 53 – p-values for the Wilcoxon signed-rank test for noise level 0.4.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	23 / 0 / 41	47 / 0 / 17	47 / 0 / 17	53 / 0 / 11	56 / 0 / 8	50 / 0 / 14
LCA	11 / 0 / 53	58 / 0 / 6	57 / 0 / 7	40 / 0 / 24	30 / 0 / 34	24 / 0 / 40
KNORAE	20 / 0 / 44	57 / 0 / 7	53 / 0 / 11	54 / 0 / 10	54 / 0 / 10	46 / 1 / 17
KNORAU	32 / 0 / 32	35 / 0 / 29	34 / 1 / 29	58 / 0 / 6	54 / 0 / 10	48 / 0 / 16

Table 54 – Wins, ties and losses for noise level 0.4.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	0.9999	0.0000	0.0001	0.0000	0.0000	0.0000
LCA	1.0000	0.0000	0.0000	0.0137	0.6335	0.9984
KNORAE	0.9998	0.0000	0.0000	0.0000	0.0000	0.0000
KNORAU	0.1531	0.0715	0.2644	0.0000	0.0000	0.0000

Table 55 – p-values for the Wilcoxon signed-rank test for noise level 0.4.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	14 / 0 / 50	29 / 0 / 35	28 / 0 / 36	14 / 0 / 50	45 / 0 / 19	51 / 0 / 13
LCA	15 / 0 / 49	50 / 0 / 14	50 / 0 / 14	40 / 0 / 24	54 / 0 / 10	48 / 0 / 16
KNORAE	13 / 0 / 51	53 / 0 / 11	51 / 0 / 13	17 / 1 / 46	55 / 0 / 9	51 / 0 / 13
KNORAU	28 / 0 / 36	18 / 0 / 46	21 / 0 / 43	16 / 0 / 48	53 / 0 / 11	54 / 0 / 10

Table 56 – Wins, ties and losses for noise level 0.4.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	1.0000	0.7791	0.9895	1.0000	0.0000	0.0000
LCA	1.0000	0.0000	0.0000	0.0016	0.0000	0.0000
KNORAE	1.0000	0.0000	0.0000	1.0000	0.0000	0.0000
KNORAU	0.9786	0.9999	1.0000	1.0000	0.0000	0.0000

Table 57 – p-values for the Wilcoxon signed-rank test for noise level 0.4.

APPENDIX B – TABLES - 50 BAGS

B.1 NOISE LEVEL: 0

B.1.1 Average Ranks

Method	Avg. Rank
KNORAE - Fire	7.007812
KNORAU - Fire	8.328125
KNORAU - Fire + BagDS	9.789062
KNORAE - Fire + MSDS	10.531250
KNORAU - Fire + MSDS	10.632812
KNORAE - BagDS	10.679688
OLA - Fire	10.734375
KNORAE - Fire + BagDS	11.140625
KNORAU - BagDS	11.218750
KNORAE - DS	11.218750
KNORAE - MSDS	11.414062
KNORAU - DS	11.476562
OLA - DS	11.976562
OLA - BagDS	12.304688
KNORAU - MSDS	12.398438

OLA - Fire + BagDS	13.140625
Static	13.281250
OLA - Fire + MSDS	14.148438
OLA - MSDS	15.250000
LCA - BagDS	17.007812
LCA - Fire + BagDS	17.968750
LCA - DS	18.164062
LCA - Fire + MSDS	18.226562
LCA - MSDS	18.375000
LCA - Fire	18.585938

Table 58 – Average Ranks for each method, for noise level 0.

B.1.2 Win/Tie/Loss comparisons

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	32 / 4 / 28	32 / 5 / 27	26 / 4 / 34	34 / 3 / 27	27 / 3 / 34	26 / 3 / 35
LCA	18 / 4 / 42	20 / 4 / 40	17 / 5 / 42	21 / 4 / 39	18 / 4 / 42	21 / 3 / 40
KNORAE	32 / 9 / 23	33 / 9 / 22	26 / 17 / 21	40 / 8 / 16	32 / 11 / 21	32 / 15 / 17
KNORAU	26 / 29 / 9	27 / 29 / 8	23 / 29 / 12	34 / 17 / 13	28 / 25 / 11	26 / 26 / 12

Table 59 – Wins, ties and losses for noise level 0.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	0.0211	0.0244	0.7509	0.0141	0.0856	0.2767
LCA	0.9997	0.9985	1.0000	0.9941	0.9892	0.9964
KNORAE	0.0177	0.0055	0.1731	0.0000	0.0068	0.0030
KNORAU	0.0036	0.0005	0.0950	0.0000	0.0000	0.0002

Table 60 – p-values for the Wilcoxon signed-rank test for noise level 0.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	28 / 4 / 32	27 / 17 / 20	19 / 9 / 36	30 / 3 / 31	24 / 6 / 34	22 / 9 / 33
LCA	42 / 4 / 18	36 / 21 / 7	26 / 8 / 30	31 / 6 / 27	37 / 5 / 22	35 / 4 / 25
KNORAE	23 / 9 / 32	24 / 25 / 15	26 / 13 / 25	40 / 7 / 17	23 / 12 / 29	27 / 10 / 27
KNORAU	9 / 29 / 26	14 / 42 / 8	13 / 31 / 20	28 / 16 / 20	23 / 22 / 19	21 / 24 / 19

Table 61 – Wins, ties and losses for noise level 0.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	0.9789	0.7954	0.9999	0.0389	0.6775	0.9819
LCA	0.0003	0.0002	0.7521	0.2454	0.0178	0.1165
KNORAE	0.9823	0.1785	0.8432	0.0000	0.1095	0.6928
KNORAU	0.9964	0.3189	0.9637	0.0002	0.0030	0.0427

Table 62 – p-values for the Wilcoxon signed-rank test for noise level 0.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	26 / 4 / 34	19 / 9 / 36	16 / 11 / 37	20 / 4 / 40	23 / 7 / 34	25 / 14 / 25
LCA	17 / 5 / 42	26 / 8 / 30	24 / 11 / 29	31 / 3 / 30	30 / 3 / 31	29 / 3 / 32
KNORAE	26 / 17 / 21	26 / 13 / 25	24 / 14 / 26	19 / 6 / 39	28 / 9 / 27	21 / 18 / 25
KNORAU	23 / 29 / 12	13 / 31 / 20	9 / 35 / 20	16 / 17 / 31	16 / 23 / 25	14 / 26 / 24

Table 63 – Wins, ties and losses for noise level 0.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	0.7509	0.9999	1.0000	1.0000	0.9968	0.9558
LCA	1.0000	0.7521	0.9502	0.6549	0.9230	0.8680
KNORAE	0.1731	0.8432	0.9553	1.0000	0.9072	0.9376
KNORAU	0.0950	0.9637	0.9925	1.0000	0.9990	0.9994

Table 64 – p-values for the Wilcoxon signed-rank test for noise level 0.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	26 / 3 / 35	22 / 9 / 33	20 / 12 / 32	25 / 14 / 25	16 / 4 / 44	27 / 7 / 30
LCA	21 / 3 / 40	35 / 4 / 25	29 / 4 / 31	32 / 3 / 29	29 / 10 / 25	24 / 13 / 27
KNORAE	32 / 15 / 17	27 / 10 / 27	26 / 12 / 26	25 / 18 / 21	17 / 7 / 40	29 / 9 / 26
KNORAU	26 / 26 / 12	21 / 24 / 19	22 / 25 / 17	24 / 26 / 14	13 / 21 / 30	18 / 27 / 19

Table 65 – Wins, ties and losses for noise level 0.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	0.2767	0.9819	0.9494	0.0442	1.0000	0.9653
LCA	0.9964	0.1165	0.3870	0.1320	0.2414	0.7273
KNORAE	0.0030	0.6928	0.8270	0.0624	1.0000	0.7833
KNORAU	0.0002	0.0427	0.0215	0.0006	1.0000	0.9580

Table 66 – p-values for the Wilcoxon signed-rank test for noise level 0.

B.2 NOISE LEVEL: 0.1

B.2.1 Average Ranks

Method	Avg. Rank
KNORAU - Fire + MSDS	6.320312
KNORAE - Fire + MSDS	7.968750
KNORAE - MSDS	8.554688
KNORAU - BagDS	8.851562
KNORAU - DS	9.070312
OLA - MSDS	9.734375
OLA - Fire + MSDS	9.812500
KNORAU - Fire	9.960938
KNORAU - MSDS	10.257812
OLA - BagDS	10.460938
KNORAU - Fire + BagDS	10.468750
OLA - DS	11.828125
Static	12.531250
KNORAE - Fire	14.171875
KNORAE - BagDS	14.179688
KNORAE - Fire + BagDS	15.289062
OLA - Fire + BagDS	15.734375
LCA - MSDS	15.867188
KNORAE - DS	16.164062
LCA - Fire + MSDS	17.148438
OLA - Fire	17.195312
LCA - Fire	17.765625
LCA - Fire + BagDS	18.023438
LCA - BagDS	18.054688
LCA - DS	19.585938

Table 67 – Average Ranks for each method, for noise level 0.1.

B.2.2 Win/Tie/Loss comparisons

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	36 / 0 / 28	33 / 0 / 31	40 / 0 / 24	22 / 0 / 42	25 / 0 / 39	33 / 0 / 31
LCA	12 / 0 / 52	13 / 0 / 51	20 / 1 / 43	18 / 0 / 46	18 / 0 / 46	18 / 0 / 46
KNORAE	22 / 0 / 42	25 / 0 / 39	42 / 2 / 20	28 / 0 / 36	22 / 1 / 41	41 / 4 / 19
KNORAU	48 / 5 / 11	47 / 5 / 12	42 / 4 / 18	33 / 0 / 31	33 / 0 / 31	54 / 4 / 6

Table 68 – Wins, ties and losses for noise level 0.1.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	0.2111	0.0743	0.0111	0.9859	0.9116	0.0453
LCA	1.0000	1.0000	0.9994	0.9992	0.9998	1.0000
KNORAE	0.9329	0.7308	0.0006	0.6944	0.9091	0.0000
KNORAU	0.0000	0.0000	0.0018	0.0254	0.0507	0.0000

Table 69 – p-values for the Wilcoxon signed-rank test for noise level 0.1.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	28 / 0 / 36	43 / 0 / 21	42 / 0 / 22	18 / 0 / 46	18 / 0 / 46	43 / 0 / 21
LCA	52 / 0 / 12	51 / 1 / 12	53 / 0 / 11	38 / 0 / 26	35 / 1 / 28	42 / 0 / 22
KNORAE	42 / 0 / 22	49 / 1 / 14	51 / 0 / 13	39 / 0 / 25	35 / 0 / 29	54 / 0 / 10
KNORAU	11 / 5 / 48	19 / 27 / 18	21 / 17 / 26	29 / 0 / 35	27 / 1 / 36	37 / 4 / 23

Table 70 – Wins, ties and losses for noise level 0.1.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	0.7889	0.0107	0.0330	1.0000	0.9998	0.0037
LCA	0.0000	0.0000	0.0000	0.0063	0.0121	0.0055
KNORAE	0.0671	0.0002	0.0000	0.0715	0.1484	0.0000
KNORAU	1.0000	0.5271	0.9824	0.3868	0.6743	0.0000

Table 71 – p-values for the Wilcoxon signed-rank test for noise level 0.1.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	40 / 0 / 24	42 / 0 / 22	36 / 0 / 28	46 / 0 / 18	46 / 0 / 18	32 / 0 / 32
LCA	20 / 1 / 43	53 / 0 / 11	47 / 1 / 16	36 / 0 / 28	36 / 0 / 28	36 / 1 / 27
KNORAE	42 / 2 / 20	51 / 0 / 13	46 / 0 / 18	45 / 0 / 19	48 / 0 / 16	35 / 0 / 29
KNORAU	42 / 4 / 18	21 / 17 / 26	22 / 14 / 28	34 / 0 / 30	36 / 0 / 28	18 / 4 / 42

Table 72 – Wins, ties and losses for noise level 0.1.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	0.0111	0.0330	0.2940	0.0001	0.0010	0.4494
LCA	0.9994	0.0000	0.0001	0.2518	0.1364	0.0844
KNORAE	0.0006	0.0000	0.0001	0.0021	0.0001	0.7428
KNORAU	0.0018	0.9824	0.9726	0.7545	0.5847	1.0000

Table 73 – p-values for the Wilcoxon signed-rank test for noise level 0.1.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	33 / 0 / 31	43 / 0 / 21	34 / 1 / 29	32 / 0 / 32	51 / 0 / 13	48 / 0 / 16
LCA	18 / 0 / 46	42 / 0 / 22	37 / 0 / 27	27 / 1 / 36	38 / 1 / 25	40 / 0 / 24
KNORAE	41 / 4 / 19	54 / 0 / 10	53 / 0 / 11	29 / 0 / 35	49 / 0 / 15	52 / 0 / 12
KNORAU	54 / 4 / 6	37 / 4 / 23	38 / 4 / 22	42 / 4 / 18	42 / 0 / 22	47 / 0 / 17

Table 74 – Wins, ties and losses for noise level 0.1.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	0.0453	0.0037	0.2404	0.5506	0.0000	0.0000
LCA	1.0000	0.0055	0.1515	0.9156	0.3895	0.1043
KNORAE	0.0000	0.0000	0.0000	0.2572	0.0000	0.0000
KNORAU	0.0000	0.0000	0.0001	0.0000	0.0580	0.0036

Table 75 – p-values for the Wilcoxon signed-rank test for noise level 0.1.

B.3 NOISE LEVEL: 0.2

B.3.1 Average Ranks

Method	Avg. Rank
KNORAU - Fire + MSDS	6.140625
OLA - MSDS	7.359375
KNORAU - BagDS	7.585938
KNORAU - DS	7.656250
KNORAU - MSDS	8.203125
KNORAE - MSDS	8.445312
KNORAE - Fire + MSDS	8.945312
OLA - BagDS	9.523438
OLA - Fire + MSDS	10.125000
Static	10.351562
KNORAU - Fire + BagDS	10.718750
KNORAU - Fire	11.500000
OLA - DS	11.843750
LCA - Fire + MSDS	14.210938
LCA - MSDS	14.234375
KNORAE - BagDS	15.132812
KNORAE - Fire + BagDS	15.992188
OLA - Fire + BagDS	17.093750
KNORAE - Fire	17.484375
LCA - Fire + BagDS	17.703125
KNORAE - DS	17.781250
LCA - Fire	18.390625
LCA - BagDS	18.562500
OLA - Fire	19.328125
LCA - DS	20.687500

Table 76 – Average Ranks for each method, for noise level 0.2.

B.3.2 Win/Tie/Loss comparisons

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	23 / 0 / 41	31 / 0 / 33	34 / 0 / 30	13 / 0 / 51	16 / 0 / 48	24 / 0 / 40
LCA	7 / 0 / 57	10 / 0 / 54	18 / 0 / 46	16 / 0 / 48	17 / 0 / 47	18 / 0 / 46
KNORAE	18 / 0 / 46	18 / 0 / 46	33 / 0 / 31	16 / 0 / 48	18 / 0 / 46	27 / 0 / 37
KNORAU	48 / 1 / 15	48 / 1 / 15	43 / 1 / 20	28 / 0 / 36	29 / 0 / 35	44 / 0 / 20

Table 77 – Wins, ties and losses for noise level 0.2.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	0.9572	0.5821	0.0514	1.0000	1.0000	0.8167
LCA	1.0000	1.0000	1.0000	1.0000	1.0000	0.9997
KNORAE	1.0000	0.9996	0.2561	1.0000	0.9999	0.5453
KNORAU	0.0000	0.0000	0.0016	0.9507	0.7352	0.0004

Table 78 – p-values for the Wilcoxon signed-rank test for noise level 0.2.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	41 / 0 / 23	43 / 1 / 20	52 / 0 / 12	10 / 0 / 54	11 / 0 / 53	38 / 0 / 26
LCA	57 / 0 / 7	56 / 0 / 8	57 / 0 / 7	39 / 0 / 25	42 / 0 / 22	47 / 0 / 17
KNORAE	46 / 0 / 18	54 / 0 / 10	57 / 0 / 7	30 / 0 / 34	40 / 1 / 23	60 / 0 / 4
KNORAU	15 / 1 / 48	28 / 13 / 23	32 / 1 / 31	23 / 0 / 41	24 / 0 / 40	34 / 1 / 29

Table 79 – Wins, ties and losses for noise level 0.2.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	0.0428	0.0032	0.0000	1.0000	1.0000	0.0612
LCA	0.0000	0.0000	0.0000	0.0173	0.0007	0.0000
KNORAE	0.0000	0.0000	0.0000	0.7396	0.0006	0.0000
KNORAU	1.0000	0.8793	0.9091	0.9994	0.9966	0.1279

Table 80 – p-values for the Wilcoxon signed-rank test for noise level 0.2.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	34 / 0 / 30	52 / 0 / 12	46 / 0 / 18	57 / 0 / 7	55 / 0 / 9	42 / 0 / 22
LCA	18 / 0 / 46	57 / 0 / 7	54 / 0 / 10	42 / 0 / 22	42 / 0 / 22	33 / 1 / 30
KNORAE	33 / 0 / 31	57 / 0 / 7	54 / 0 / 10	53 / 0 / 11	49 / 0 / 15	44 / 1 / 19
KNORAU	43 / 1 / 20	32 / 1 / 31	28 / 6 / 30	41 / 0 / 23	39 / 0 / 25	30 / 0 / 34

Table 81 – Wins, ties and losses for noise level 0.2.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	0.0514	0.0000	0.0000	0.0000	0.0000	0.0002
LCA	1.0000	0.0000	0.0000	0.0002	0.0075	0.6429
KNORAE	0.2561	0.0000	0.0000	0.0000	0.0000	0.0137
KNORAU	0.0016	0.9091	0.8144	0.0040	0.0146	0.9027

Table 82 – p-values for the Wilcoxon signed-rank test for noise level 0.2.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	24 / 0 / 40	38 / 0 / 26	28 / 0 / 36	22 / 0 / 42	59 / 0 / 5	55 / 0 / 9
LCA	18 / 0 / 46	47 / 0 / 17	44 / 0 / 20	30 / 1 / 33	49 / 0 / 15	50 / 0 / 14
KNORAE	27 / 0 / 37	60 / 0 / 4	60 / 0 / 4	19 / 1 / 44	55 / 0 / 9	56 / 0 / 8
KNORAU	44 / 0 / 20	34 / 1 / 29	31 / 1 / 32	34 / 0 / 30	49 / 0 / 15	53 / 0 / 11

Table 83 – Wins, ties and losses for noise level 0.2.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	0.8167	0.0612	0.7460	0.9998	0.0000	0.0000
LCA	0.9997	0.0000	0.0000	0.3571	0.0000	0.0000
KNORAE	0.5453	0.0000	0.0000	0.9863	0.0000	0.0000
KNORAU	0.0004	0.1279	0.1522	0.0973	0.0000	0.0000

Table 84 – p-values for the Wilcoxon signed-rank test for noise level 0.2.

B.4 NOISE LEVEL: 0.3

B.4.1 Average Ranks

Method	Avg. Rank
KNORAU - MSDS	5.632812
KNORAU - DS	5.914062
KNORAU - BagDS	5.945312
KNORAU - Fire + MSDS	6.328125
OLA - MSDS	7.445312
Static	8.398438
KNORAE - MSDS	8.859375
KNORAE - Fire + MSDS	10.218750
OLA - BagDS	10.710938
KNORAU - Fire + BagDS	11.414062
OLA - Fire + MSDS	11.500000
OLA - DS	12.640625
LCA - Fire + MSDS	12.898438
LCA - MSDS	14.132812
KNORAU - Fire	14.867188
KNORAE - BagDS	15.007812
KNORAE - Fire + BagDS	16.382812
KNORAE - DS	16.976562
LCA - Fire + BagDS	17.242188
LCA - BagDS	17.578125
OLA - Fire + BagDS	17.914062
OLA - Fire	18.585938
KNORAE - Fire	19.296875
LCA - DS	19.375000
LCA - Fire	19.734375

Table 85 – Average Ranks for each method, for noise level 0.3.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	18 / 0 / 46	21 / 0 / 43	27 / 0 / 37	10 / 0 / 54	8 / 0 / 56	24 / 0 / 40
LCA	7 / 0 / 57	8 / 0 / 56	14 / 0 / 50	8 / 0 / 56	13 / 0 / 51	21 / 0 / 43
KNORAE	13 / 0 / 51	14 / 0 / 50	23 / 0 / 41	7 / 0 / 57	12 / 0 / 52	22 / 0 / 42
KNORAU	44 / 1 / 19	45 / 0 / 19	46 / 0 / 18	15 / 0 / 49	22 / 0 / 42	31 / 0 / 33

Table 86 – Wins, ties and losses for noise level 0.3.

B.4.2 Win/Tie/Loss comparisons

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	1.0000	0.9999	0.8869	1.0000	1.0000	0.9999
LCA	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
KNORAE	1.0000	1.0000	0.9818	1.0000	1.0000	0.9991
KNORAU	0.0006	0.0005	0.0007	1.0000	0.9998	0.5080

Table 87 – p-values for the Wilcoxon signed-rank test for noise level 0.3.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	46 / 0 / 18	43 / 0 / 21	54 / 0 / 10	11 / 0 / 53	13 / 0 / 51	32 / 0 / 32
LCA	57 / 0 / 7	54 / 1 / 9	56 / 0 / 8	34 / 0 / 30	44 / 0 / 20	50 / 0 / 14

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	0.0000	0.0011	0.0000	1.0000	1.0000	0.3056
LCA	0.0000	0.0000	0.0000	0.2963	0.0004	0.0000
KNORAE	0.0000	0.0004	0.0000	0.9993	0.0339	0.0000
KNORAU	0.9994	0.1711	0.0288	1.0000	1.0000	0.9951

Table 89 – p-values for the Wilcoxon signed-rank test for noise level 0.3.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	27 / 0 / 37	54 / 0 / 10	49 / 0 / 15	59 / 0 / 5	56 / 0 / 8	52 / 0 / 12
LCA	14 / 0 / 50	56 / 0 / 8	52 / 0 / 12	44 / 0 / 20	41 / 0 / 23	27 / 0 / 37
KNORAE	23 / 0 / 41	57 / 0 / 7	55 / 0 / 9	58 / 0 / 6	54 / 0 / 10	46 / 0 / 18
KNORAU	46 / 0 / 18	39 / 0 / 25	40 / 1 / 23	52 / 0 / 12	47 / 0 / 17	42 / 0 / 22

Table 90 – Wins, ties and losses for noise level 0.3.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	0.8869	0.0000	0.0000	0.0000	0.0000	0.0000
LCA	1.0000	0.0000	0.0000	0.0000	0.0144	0.8750
KNORAE	0.9818	0.0000	0.0000	0.0000	0.0000	0.0003
KNORAU	0.0007	0.0288	0.0813	0.0000	0.0000	0.0015

Table 91 – p-values for the Wilcoxon signed-rank test for noise level 0.3.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	24 / 0 / 40	32 / 0 / 32	23 / 0 / 41	12 / 0 / 52	55 / 0 / 9	55 / 0 / 9
LCA	21 / 0 / 43	50 / 0 / 14	48 / 0 / 16	37 / 0 / 27	54 / 0 / 10	53 / 1 / 10
KNORAE	22 / 0 / 42	54 / 0 / 10	51 / 0 / 13	18 / 0 / 46	58 / 0 / 6	55 / 0 / 9
KNORAU	31 / 0 / 33	21 / 0 / 43	20 / 0 / 44	22 / 0 / 42	56 / 0 / 8	55 / 0 / 9

Table 92 – Wins, ties and losses for noise level 0.3.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	0.9999	0.3056	0.9500	1.0000	0.0000	0.0000
LCA	1.0000	0.0000	0.0000	0.1250	0.0000	0.0000
KNORAE	0.9991	0.0000	0.0000	0.9997	0.0000	0.0000
KNORAU	0.5080	0.9951	0.9971	0.9985	0.0000	0.0000

Table 93 – p-values for the Wilcoxon signed-rank test for noise level 0.3.

B.5 NOISE LEVEL: 0.4

B.5.1 Average Ranks

Method	Avg. Rank
KNORAU - MSDS	5.289062
KNORAU - BagDS	5.390625
KNORAU - DS	5.781250
Static	6.351562
KNORAU - Fire + MSDS	7.703125
OLA - MSDS	8.882812
KNORAE - MSDS	9.281250
OLA - BagDS	10.343750
KNORAU - Fire + BagDS	10.593750
KNORAE - Fire + MSDS	11.468750
LCA - Fire + MSDS	12.132812
OLA - DS	12.937500
OLA - Fire + MSDS	13.515625
KNORAU - Fire	15.046875
LCA - MSDS	15.187500
KNORAE - BagDS	15.257812
KNORAE - Fire + BagDS	15.953125
OLA - Fire + BagDS	17.007812
LCA - Fire + BagDS	17.179688

KNORAE - DS	17.460938
OLA - Fire	17.710938
LCA - BagDS	17.945312
KNORAE - Fire	18.421875
LCA - DS	19.070312
LCA - Fire	19.085938

Table 94 – Average Ranks for each method, for noise level 0.4.

B.5.2 Win/Tie/Loss comparisons

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	13 / 0 / 51	17 / 0 / 47	24 / 0 / 40	7 / 0 / 57	13 / 0 / 51	13 / 0 / 51
LCA	5 / 0 / 59	5 / 0 / 59	9 / 0 / 55	5 / 0 / 59	8 / 0 / 56	13 / 0 / 51
KNORAE	7 / 0 / 57	7 / 0 / 57	19 / 0 / 45	7 / 0 / 57	8 / 0 / 56	15 / 0 / 49
KNORAU	30 / 0 / 34	33 / 0 / 31	32 / 0 / 32	9 / 0 / 55	18 / 1 / 45	25 / 0 / 39

Table 95 – Wins, ties and losses for noise level 0.4.

DS Algorithm	DS vs Static	BagDS vs Static	MSDS vs Static	FireDS vs Static	BagDS + Fire vs Static	MSDS + Fire vs Static
OLA	1.0000	1.0000	0.9996	1.0000	1.0000	1.0000
LCA	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
KNORAE	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
KNORAU	0.4023	0.1230	0.1531	1.0000	1.0000	0.9947

Table 96 – p-values for the Wilcoxon signed-rank test for noise level 0.4.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	51 / 0 / 13	49 / 0 / 15	45 / 0 / 19	12 / 0 / 52	15 / 0 / 49	30 / 0 / 34
LCA	59 / 0 / 5	52 / 0 / 12	56 / 0 / 8	36 / 0 / 28	41 / 0 / 23	51 / 0 / 13
KNORAE	57 / 0 / 7	49 / 0 / 15	56 / 0 / 8	27 / 0 / 37	41 / 2 / 21	52 / 0 / 12
KNORAU	34 / 0 / 30	39 / 2 / 23	37 / 0 / 27	7 / 0 / 57	12 / 0 / 52	18 / 0 / 46

Table 97 – Wins, ties and losses for noise level 0.4.

DS Algorithm	Static vs DS	BagDS vs DS	MSDS vs DS	FireDS vs DS	BagDS + Fire vs DS	MSDS + Fire vs DS
OLA	0.0000	0.0001	0.0000	1.0000	1.0000	0.9276
LCA	0.0000	0.0000	0.0000	0.1131	0.0012	0.0000
KNORAE	0.0000	0.0000	0.0000	0.9248	0.0030	0.0000
KNORAU	0.5977	0.0799	0.0528	1.0000	1.0000	1.0000

Table 98 – p-values for the Wilcoxon signed-rank test for noise level 0.4.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	24 / 0 / 40	45 / 0 / 19	41 / 0 / 23	54 / 0 / 10	57 / 0 / 7	51 / 0 / 13
LCA	9 / 0 / 55	56 / 0 / 8	55 / 0 / 9	37 / 0 / 27	33 / 0 / 31	24 / 0 / 40
KNORAE	19 / 0 / 45	56 / 0 / 8	53 / 0 / 11	55 / 0 / 9	52 / 0 / 12	47 / 0 / 17
KNORAU	32 / 0 / 32	37 / 0 / 27	37 / 0 / 27	57 / 0 / 7	51 / 0 / 13	51 / 0 / 13

Table 99 – Wins, ties and losses for noise level 0.4.

DS Algorithm	MSDS vs Static	MSDS vs DS	MSDS vs BagDS	MSDS vs FireDS	MSDS vs BagDS + Fire	MSDS vs MSDS + Fire
OLA	0.9996	0.0000	0.0020	0.0000	0.0000	0.0000
LCA	1.0000	0.0000	0.0000	0.0167	0.3222	0.9985
KNORAE	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000
KNORAU	0.1531	0.0528	0.0961	0.0000	0.0000	0.0000

Table 100 – p-values for the Wilcoxon signed-rank test for noise level 0.4.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	13 / 0 / 51	30 / 0 / 34	19 / 0 / 45	13 / 0 / 51	48 / 0 / 16	48 / 1 / 15
LCA	13 / 0 / 51	51 / 0 / 13	49 / 0 / 15	40 / 0 / 24	55 / 0 / 9	56 / 1 / 7
KNORAE	15 / 0 / 49	52 / 0 / 12	46 / 0 / 18	17 / 0 / 47	52 / 0 / 12	50 / 0 / 14
KNORAU	25 / 0 / 39	18 / 0 / 46	17 / 0 / 47	13 / 0 / 51	53 / 0 / 11	48 / 0 / 16

Table 101 – Wins, ties and losses for noise level 0.4.

DS Algorithm	MSDS + Fire vs Static	MSDS + Fire vs DS	MSDS + Fire vs BagDS	MSDS + Fire vs MSDS	MSDS + Fire vs FireDS	MSDS + Fire vs BagDS + Fire
OLA	1.0000	0.9276	0.9998	1.0000	0.0000	0.0000
LCA	1.0000	0.0000	0.0000	0.0015	0.0000	0.0000
KNORAE	1.0000	0.0000	0.0000	1.0000	0.0000	0.0000
KNORAU	0.9947	1.0000	1.0000	1.0000	0.0000	0.0000

Table 102 – p-values for the Wilcoxon signed-rank test for noise level 0.4.