UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Rogério César Peixoto Fragoso

Clustering-based dynamic ensemble selection for one-class decomposition

Recife

2022

Rogério César Peixoto Fragoso

Clustering-based dynamic ensemble selection for one-class decomposition

Tese 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 grau de Doutor em Ciência da Computação.

Área de Concentração: Inteligência computacional

Orientador: Prof. Dr. George D. C. Cavalcanti

Coorientador: Prof. Dr. Luiz E. S. Oliveira
Coorientador: Prof. Dr. Roberto H. W. Pinheiro

Recife

2022

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

F811c Fragoso, Rogério César Peixoto
Clustering-based dynamic ensemble selection for one-class decomposition / Rogério César Peixoto Fragoso. – 2022.
81 f.:il., fig, tab.

Orientador: George Darmiton da Cunha Cavalcanti.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022

Inclui referências e apêndices.

1. Inteligência computacional. 2. Sistemas de múltiplos classificadores. I. Cavalcanti, George Darmiton da Cunha (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2022-183

Rogério César Peixoto Fragoso

“Clustering-based Dynamic Ensemble Selection for One-Class Decomposition”

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Inteligência Computacional.

Aprovado em: 24/08/2022.

_____

Orientador: Prof. Dr. George Darmiton da Cunha Cavalcanti

BANCA EXAMINADORA

_____

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

_____

Prof. Dr. Yandre Maldonado e Gomes da Costa
Departamento de Informática / UEM

_____

Prof. Dr. Renato Vimieiro
Departamento de Ciência da Computação / UFMG

_____

Prof. Dr. George Gomes Cabral
Departamento de Computação / UFRPE

_____

Prof. Dr. Péricles Barbosa Cunha de Miranda
Departamento de Computação / UFRPE

ABSTRACT

A natural solution to tackle multi-class problems is employing multi-class classifiers. However, in specific situations, such as imbalanced data or a high number of classes, it is more effective to decompose the multi-class problem into several and easier to solve problems. One-class decomposition is an alternative, where one-class classifiers (OCCs) are trained for each class separately. However, fitting the data optimally is a challenge for classifiers, especially when it presents a complex intra-class distribution. The literature shows that multiple classifier systems are inherently robust in such cases. Thus, the adoption of multiple OCCs for each class can lead to an improvement for the one-class decomposition. With that in mind, in this work, we introduce two methods for multi-class classification using ensembles of OCCs. One-class Classifier Dynamic Ensemble Selection for Multi-class problems (MODES, for short) and Density-Based Dynamic Ensemble Selection (DBDES) provide competent classifiers for each region of the feature space by decomposing the original multi-class problem into multiple one-class problems, segmenting the data from each class, and training a OCC for each cluster. The rationale is to reduce the complexity of the classification task by defining a region of the feature space where the classifier is supposed to be an expert. The classification of a test instance is performed by dynamically selecting an ensemble of competent OCCs and the final decision is given by the reconstruction of the original multi-class problem. Experiments carried out with 25 databases, 4 OCC models, and 3 aggregation methods showed that the proposed techniques outperform the literature. When compared with literature techniques, MODES and DBDES obtained better results, especially for databases with complex decision regions.

Keywords: one-class decomposition; multiple classifier system; dynamic ensemble selection.

RESUMO

Uma solução natural para lidar com problemas multi-classe é empregar classificadores multi-classe. No entanto, em situações específicas, como dados desbalanceados ou grande número de classes, decompor o problema multiclasse em vários problemas mais fáceis de resolver pode ser mais eficaz. A decomposição em uma classe é uma alternativa, onde classificadores de uma classe (OCCs) são treinados para cada classe separadamente. No entanto, ajustar os dados de forma otimizada é um desafio para os classificadores, principalmente quando os dados apresentam uma distribuição intra-classe complexa. A literatura mostra que sistemas de múltiplos classificadores são inerentemente robustos em tais casos. Assim, a adoção de múltiplos OCCs para cada classe pode levar a uma melhoria de desempenho na decomposição de uma classe. Com isso em mente, neste trabalho apresentamos dois métodos para classificação de problemas multi-classe através ensembles de OCCs. One-class Classifier Dynamic Ensemble Selection for Multi-class problems (MODES) e Density-Based Dynamic Ensemble Selection (DBDES) fornecem classificadores competentes para cada região do espaço de características, decompondo o problema multiclasse original em vários problemas de uma classe, segmentam os dados de cada classe e um OCC é treinado para cada cluster. MODES utiliza o algoritmo K-means e um conjunto de índices de validação de cluster enquanto DBDES utiliza o algoritmo OPTICS para a segmentação dos dados. A lógica é reduzir a complexidade da tarefa de classificação definindo uma região do espaço de características onde o classificador deve ser um especialista. A classificação de uma instância de teste é realizada selecionando dinamicamente um conjunto de OCCs competentes e a decisão final é dada pela reconstrução do problema multiclasse original. Experimentos realizados com 25 bancos de dados, 4 modelos OCC e 3 métodos de agregação mostraram que as técnicas propostas superam a literatura. Quando comparado com técnicas da literatura, MODES e DBDES obtiveram melhores resultados, principalmente para bancos de dados com regiões de decisão complexas.

Palavras-chaves: decomposição de uma classe; sistemas de múltiplos classificadores; seleção dinâmica de ensemble.

LIST OF FIGURES

Figure 1 – Decision region of a single Gaussian One-Class Classifier (OCC). Scenario (a) shows a broad decision region, leading to a high false positive rate. Scenario (b) shows a reduced decision region leading to a high false negative rate. 19

Figure 2 – Decision regions using (a) one, (b) two, (c) four and (d) five Gaussian one-class classifiers. 20

Figure 3 – Reachability plot for a database with hierarchical clusters of different sizes and densities. Each "valley" in the plot represents a cluster where the steep areas represent the start and end of clusters. 29

Figure 4 – One-class Dynamic Ensemble Selection for Multi-class problems (MODES) training phase. 39

Figure 5 – MODES test phase. 40

Figure 6 – Toy example of the training phase of MODES for a class label l. The cluster validity indices output a set $D = \{2, 4, 5\}$ containing the numbers of clusters to segment the target data. A one-class classifier λ_j is trained for each cluster c_j. 42

Figure 7 – Toy example of the test phase of MODES for a class label l. MODES computes the euclidean distance between the test instance x (represented by •) and the centroid of each cluster. The OCC trained with data from the closest cluster is is dynamically selected to the ensemble E_l. 43

Figure 8 – Toy example of Density-Based Dynamic Ensemble Selection (DBDES) for a class label l. The figure on the left (a), presents the training phase and the figure on the right presents the test phase. 47

Figure 9 – Result for Nemenyi post hoc test for (a) accuracy and (b) Kappa Statistic. 55

Figure 10 – Frequency of use of OCCs with MODES for Decision Templates (DTs) and Maximum Support (MAX) (a) and Error Correcting Output Codes (ECOC) (b). 56

Figure 11 – Result for Nemenyi *post-hoc* test for (a) accuracy and (b) Kappa Statistic performances of DBDES. 60

Figure 12 – Frequency of use of OCCs with DBDES for DTs and MAX (a) and ECOC (b). 61
Figure 13 – Result for Nemenyi *post-hoc* test for (a) accuracy and (b) Kappa Statistic performances of MODES, DBDES, Dynamic Ensemble Selection with THReshold-based neighborhood pruning (DES_{THR}), and the static combination of OCCs. 63
Figure 14 – Accuracy and Kappa statistic for the best configurations of DBDES, MODES, and DES_{THR} according to the homogeneity of the neighborhood. Databases where more than 25% of the instances present more than one class in the neighborhood are shown in the left side of the figure. The right side shows databases where less than 25% of the instances present more than one class in the neighborhood. 64
Figure 15 – Accuracy and Kappa statistic for the best configurations of DBDES, MODES, and DES_{THR} according to the Imbalance Ratio. Databases with IR < 10 are shown in the left side of the figure. The right side shows databases with IR ≥ 10. 66
Figure 16 – Accuracy and Kappa statistic for the best configurations of DBDES, MODES, and DES_{THR} according to the number of classes of the databases. Databases with less than 7 classes are shown in the left side of the figure. The right side shows databases at least 7 classes. 68

LIST OF TABLES

Table 1 – Comparison involving DBDES, MODES, and other techniques that use one-class decomposition, OCC ensemble, for each one-class problem, clustering-based approach, and/or dynamic selection (DS) 35

Table 2 – Default values for MODES and DBDES hyper-parameters. 48

Table 3 – Databases description. Imbalance Ratio is computed as the division of the cardinality of the largest class by the cardinality of the smallest class. 51

Table 4 – OCCs hyper-parameters. Matlab dd_tools library implementation was used in the experiments. 52

Table 5 – Accuracy performance (in %) of MODES using DTs, ECOC, and MAX for four one-class classifiers. The best result for each database is in bold. The last row represents the number of wins, ties, and losses achieved by each technique. 54

Table 6 – Kappa performance of MODES using DTs, ECOC, and MAX for four one-class classifiers. The best result for each database is in bold. The last row represents the number of wins, ties, and losses achieved by each technique. . 55

Table 7 – Accuracy performance (in %) of DBDES using DTs, ECOC, and MAX aggregation methods for four one-class classifiers. The best result for each database is in bold. The last row represents the number of wins, ties, and losses achieved by each technique. 57

Table 8 – Kappa statistic performance of DBDES using DTs, ECOC, and MAX aggregation methods for four one-class classifiers. The best result for each database is in bold. The last row represents the number of wins, ties, and losses achieved by each technique. 59

Table 9 – Accuracy performance (in %) of MODES, DBDES, DES_{THR}, and Static aggregation of OCCs. The best result for each database is in bold. The last rows represent the mean performance across all databases, the number of wins, ties, and losses achieved by each technique, and the average rankings. 62

Table 10 – Kappa performance of MODES, DBDES, DES_{THR}, and Static aggregation of OCCs. The best result for each database is in bold. The last rows represent the mean performance across all databases, the number of wins, ties, and losses achieved by each technique, and the average rankings. 63
Table 11 – Mean accuracy and Kappa statistic for top-performing configurations of DBDES, MODES, and DES_{THR} according to the homogeneity of the neighborhoods in the databases. 65
Table 12 – Mean accuracy and Kappa statistic for top-performing configurations of DBDES, MODES, and DES_{THR} according to the Imbalance Ratio. 67
Table 13 – Mean accuracy and Kappa statistic for top-performing configurations of DBDES, MODES, and DES_{THR} according to the number of classes of the databases. 67
Table 14 – Wilcoxon signed rank test results comparing the top-performing configurations of DBDES, MODES, and DES_{THR}. Convention adopted: "$>>$" means strong evidence that the technique in the column presents greater effectiveness than the technique in the row; "$>$" means evidence that the technique in the column presents greater effectiveness than the technique in the row; "$\sim$" means no evidence that the technique in the column presents greater effectiveness than the technique in the row. 69
Table 15 – Mean execution time across all databases for the top-performing configurations of MODES, DBDES, and DES_{THR}. The time is presented in seconds and is divided by training, test, and total. 69
Table 16 – Training time of MODES, DBDES, and DES_{THR} top performing configurations and the characteristics of the databases. Imbalance Ratio (IR) is computed as the division of the cardinality of the largest class by the cardinality of the smallest class. Best result for each database in bold. The last rows represent the mean performance across all databases, the number of wins, ties and losses achieved by each technique, and the average rankings. . 78

Table 17 – Prediction time of MODES, DBDES, and DES_{THR} top performing configurations and the characteristics of the databases. IR is computed as the division of the cardinality of the largest class by the cardinality of the smallest class.. Best result for each database in bold. The last rows represent the mean performance across all databases, the number of wins, ties and losses achieved by each technique, and the average rankings. 79
Table 18 – Accuracy performance (in %) of MODES, DBDES, and Random Forest. Best result for each database in bold. The last rows represent the mean performance across all databases, the number of wins, ties and losses achieved by each technique, the average rankings and the $p - value$ for Wilcoxon Signed Ranking Test. 80
Table 19 – Kappa performance of MODES, DBDES, and Random Forest. Best result for each database in bold. The last rows represent the mean performance across all databases, the number of wins, ties and losses achieved by each technique, the average rankings and the $p - value$ for Wilcoxon Signed Ranking Test. 81

LIST OF ABBREVIATIONS AND ACRONYMS

CVI	Cluster Validity Index
DBDES	Density-Based Dynamic Ensemble Selection
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DCS	Dynamic Classifier Selection
DES	Dynamic Ensemble Selection
DES_{THR}	Dynamic Ensemble Selection with THReshold-based neighborhood pruning
DTs	Decision Templates
DTs	Decision Templates
ECOC	Error Correcting Output Codes
GaussianDD	Gaussian Data Descriptor
IR	Imbalance Ratio
MAX	Maximum Support
MCS	Multiple Classifiers Systems
MEAN	Mean Support
MODES	One-class Dynamic Ensemble Selection for Multi-class problems
MSTDD	Minimum Spanning Tree Data Descriptor
OCC	One-Class Classifier
OPTICS	Ordering Points To Identify the Clustering Structure
ParzenDD	Parzen Data Descriptor
SVDD	Support Vector Data Descriptor

LIST OF SYMBOLS

c	A cluster
D_{SEL}	Validation set
E	Ensemble
g	A partition, composed of clusters
L	The set of class labels in a problem
p	Pool of classifiers
x	Test instance
Γ	A training database
λ	A one-class classifier
ξ	OPTICS hyper-parameter for hierarchical cluster extraction
ϵ	The radius of a data instance neighborhood
Ψ	The region of competence of a test instance

CONTENTS

1 INTRODUCTION 15
1.1 PROBLEM STATEMENT 18
1.2 OBJECTIVES 20
1.3 CONTRIBUTIONS 21
1.4 ORGANIZATION 21
2 BACKGROUND 23
2.1 CLUSTERING 23
2.1.1 Partitioning 23
2.1.2 Hierarchical 24
2.1.3 Density-based 25
2.1.4 Grid-based 25
2.1.5 Algorithms 26
2.1.5.1 K-means 26
2.1.5.2 DBSCAN 27
2.1.5.3 OPTICS 27
2.1.6 Cluster validity indices 30
2.2 ONE-CLASS CLASSIFICATION 30
2.2.1 OCC models 31
2.3 MULTIPLE CLASSIFIERS SYSTEMS 32
2.3.1 Aggregation techniques 33
2.4 RELATED WORKS 34
3 PROPOSAL 37
3.1 MODES 38
3.1.1 Training phase 38
3.1.2 Test phase 40
3.1.3 Toy example 42
3.2 DBDES 43
3.2.1 Training phase 44
3.2.2 Test phase 45
3.2.3 Toy example 46

3.3 HYPER-PARAMETERS 47
4 EXPERIMENTS 49
4.1 EXPERIMENTAL PROTOCOL 49
4.2 EXPERIMENTAL RESULTS 53
4.2.1 Experiment 1 53
4.2.2 Experiment 2 57
4.2.3 Experiment 3 60
5 FINAL REMARKS 71
REFERENCES 73
APPENDIX A – EXECUTION TIME 78
APPENDIX B – COMPARISON WITH BENCHMARK MULTI-CLASS CLASSIFIER 80

1 INTRODUCTION

Multi-class classification may be tackled in several ways. The most common approach is to use multi-class classifiers, where the objective is to separate the classes of the problem. However, some difficulties embedded in the nature of the data, for example, class imbalance, complex data distribution, class overlap, a high number of classes, and small data samples, may impair the performance if not carefully treated.

Deep learning (POUYANFAR et al., 2018), which is currently one of the most remarkable machine learning techniques, shows good performance in a variety of multi-class problems and, in general, does not require any special procedure to deal with the aforementioned issues. However, the amount of data required for training a satisfactory model depends on the complexity of the problem, i.e., the more complex is the problem, the more data is needed. Databases presenting complex data distribution, class overlap, and a high number of classes, for example, require more data (CAO; ZHANG; TANG, 2018). Furthermore, class imbalance is still an important issue regarding deep learning methods, mainly for non-image databases (JOHNSON; KHOSHGOFTAAR, 2019). Data sampling, i.e., oversampling and downsampling, may be used to diminish this issue. Oversampling techniques, such as Synthetic Minority Over-sampling TEchnique (SMOTE) (CHAWLA et al., 2002), augment the size of the minority class, creating synthetic data instances to balance the majority class. Downsampling techniques, perform data reduction on the majority class. In oversampling, the original distribution of the minority class is maintained and no data is discarded. However, since the majority class is ignored, synthetic data instances may include noisy data (KIM; JO; SHIN, 2016) and/or may be generated over the majority class increasing class overlap (HE; GARCIA, 2009). On the other hand, downsampling diminishes the computational effort for training, since less data is used, however, the classification performance may be hindered due to information loss caused by the exclusion of useful data. Thus, reducing the data may not be a good option if data is already scarce. Therefore, databases with such impairments may be a difficult to solve problem, even for a powerful technique such as deep learning.

An alternative that is recently gaining attention is the one-class decomposition, where each class is treated as a separate problem (KRAWCZYK; WOŹNIAK; HERRERA, 2015; KRAWCZYK et al., 2018). The training data from each class of the original multi-class problem is fed to an One-Class Classifier (OCC), and each OCC estimates the likelihood that a test instance

belongs to the target class. The final classification decision, i.e., which class the test instance is assigned to, is given by the aggregation of the decisions made by the OCCs. The rationale is that, following the divide and conquer principle, each one-class problem is simpler to solve than the original multi-class problem.

It is important to remark that one-class classifiers are not superior to standard classifiers for most of the multi-class problems. OCCs do not access counter-examples in the training phase, instead, it uses the target data to define a decision border that describes the data and maintains the generalization power. In this case, inter-class information is lost. However, the nature of OCCs makes of one-class decomposition an interesting approach for specific cases, where the data contain the aforementioned issues (KRAWCZYK; WOŹNIAK; HERRERA, 2015). For example, because OCCs treat each class individually, class imbalance is not an issue. Furthermore, for the same reason, a database with class overlapping may be better described by one-class classifiers than with standard classifiers. Even fuzzy classification does not allow extremely overlapped decision regions (CUPERTINO; ZHAO; CARNEIRO, 2015). One-class decomposition permits the generation of overlapped decision borders in the training phase and the aggregation in the test phase is responsible for deciding which class a test instance belongs to based on the proximity or importance to each class, for example. One-class decomposition is also beneficial for problems in which the number of classes may vary, such as incremental learning or open-set problems (KRAWCZYK; WOŹNIAK; CYGANEK, 2014).

The most intuitive approach for one-class decomposition is to use an OCC for each class of the original problem and aggregate their outputs. However, a single one-class classifier, as well as a single standard classifier, hardly ever fits the data distribution optimally (TAX; DUIN, 2001). Recently, Multiple Classifiers Systems (MCS) have been adopted in the context of one-class problems aiming to address this issue. MCS may perform better and be more robust than a single classifier by combining the outputs of distinct classifiers (KUNCHEVA, 2014; ZHOU, 2012). For this reason, MCS are an increasingly adopted approach (CRUZ; SABOURIN; CAVALCANTI, 2018). The most common way of working with MCS is to generate a pool of classifiers using the training data (generation step) and to combine the responses of these classifiers to give the final classification (aggregation step). The classifiers should be complementary, i.e., competent in different regions of the feature space, and individually accurate. Optionally, a selection step may be adopted to select the most competent classifiers instead of combining all of them. The selection may be static (a subset of the classifiers in the pool is selected to classify all test instances) or dynamic (a subset of classifiers is selected on the fly to classify each test

instance).

Relevant studies using Multiple Classifiers Systems with one-class classifiers have shown promising results (KRAWCZYK; WOŹNIAK; CYGANEK, 2014; KRAWCZYK et al., 2018; KRAWCZYK; WOŹNIAK, 2016; LIU et al., 2016). To the best of our knowledge, a single technique adopted dynamic ensemble selection for one-class decomposition (KRAWCZYK et al., 2018). However, this technique does not treat the intra-class complex data distribution, i.e., the presence of remote instances and multi-modal data, as a particular issue.

We propose in this work two methods for multi-class classification using one-class decomposition: One-class Dynamic Ensemble Selection for Multi-class problems (MODES) and Density-Based Dynamic Ensemble Selection (DBDES). Both techniques decompose the multi-class problem into one-class problems, segment the training data of each class into clusters, and train an OCC for each cluster. During the generalization phase, an ensemble (a subset of the whole pool of OCCs) is selected per test instance. This ensemble should be composed of the most competent OCCs to classify the test instance. Afterward, these OCCs in the ensemble are combined to predict the class of the test instance.

Since each OCC is trained with different subsets, we expect a high diversity among the OCCs in the generated pool, which is a desirable characteristic of MCS. The proposed strategy aims at facilitating the classification task because each OCC deals only with part of the whole classification problem, which is the rationale behind the divide and conquer principle. Moreover, this data segmentation strategy is also interesting in dealing with complex data distribution (as shown in the problem statement, in Section 1.1) because, in such situations, locally specialized OCCs may fit better the data (KRAWCZYK; WOŹNIAK; CYGANEK, 2014).

MODES uses K-means algorithm for the clustering step. As the number of clusters k is required and the determination of the ideal value for this hyper-parameter in a database is still an open problem (KOLESNIKOV; TRICHINA; KAURANNE, 2015), we use cluster validity indices (ARBELAITZ et al., 2013) as an alternative to defining the number of clusters for each class.

DBDES, differently, uses Ordering Points To Identify the Clustering Structure (OPTICS) for the clustering step. The primary motivation for this choice is to avoid the problem of defining the number of clusters in the data since this algorithm does not require such parameterization. Furthermore, OPTICS can identify arbitrarily shaped clusters and clusters with different densities (ANKERST et al., 1999), unlike K-means, which leads to more accurate segmentation in the case of non-convex clusters.

1.1 PROBLEM STATEMENT

OCCs try to define a closed boundary around the instances belonging to only one class in the training. In the test, the class used for training, called the target class, has to be distinguished from all other possible instances. The class of non-target instances is known as the outlier class.

A single OCC may not be capable of capturing well the characteristics of the target class. For example, if the training data is multi-modal or contains remote instances, empty regions may appear within the decision boundary due to an over-estimation in training. That is, seeking to encompass the remote instances or the multiple modes present in the training data, the decision boundary may include regions that are not covered by any training instance. In the test, both target and outlier instances may appear in the empty regions (KRAWCZYK et al., 2018).

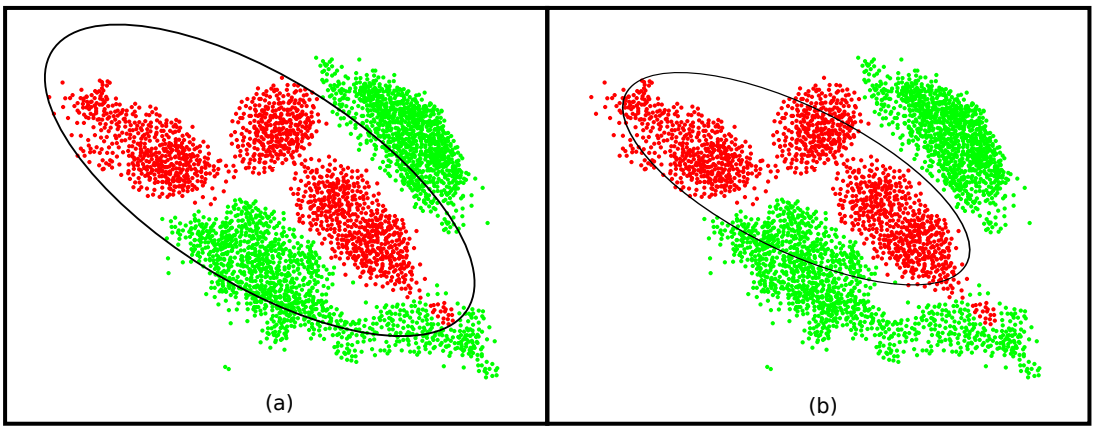
Even successful OCCs, such as Support Vector Data Descriptor (SVDD) suffer from this issue. SVDD defines a spherically shaped decision boundary around the target data (TAX; DUIN, 2004). However, a single hyper-sphere may not be able to best describe the data if there are some distinctive distributions in it (LE et al., 2010). Kernel functions help to optimize the decision region in such a way that most of the target instances are inside it while minimizing its size to diminish the probability of including outliers. The training data is mapped from the input space into a higher dimensional feature space which is easier to distinguish from other distributions. However, SVDD assumes that all training instances (in the target class) come from a single distribution, which is not true in many cases. Hence, in the case of distinctive data distributions in the target class, outliers may be inside the decision region, as well, if only one spherically shaped decision boundary, i.e., a single SVDD, is used (XIAO et al., 2009).

Consider a target class formed by more than one distribution, i.e., the data is multi-modal. If only one OCC is trained for the target class, two scenarios may occur: (a) the OCC will define a broad (overestimated) decision region to contemplate all target instances, or (b) the OCC will define a tight decision region to diminish the proportion of instances from the outlier class inside it. A broad decision region may lead to a high false positive rate while a reduced decision region may lead to a high false negative rate.

Figure 1 shows a hypothetical database where the data distribution presents a complex form and the classes present multi-modal data. The points in red represent the target class and the points in green represent the outlier class. Scenarios (a) and (b), mentioned above,

are depicted using a single Gaussian one-class classifier, for the sake of simplicity, however, the concepts may be extended for more complex OCCs.

Figure 1 – Decision region of a single Gaussian OCC. Scenario (a) shows a broad decision region, leading to a high false positive rate. Scenario (b) shows a reduced decision region leading to a high false negative rate.



Source: The author.

In scenario (a), the decision region is wide enough to embrace all instances from the target class. Only instances from the target class are used in the training process, i.e., outliers are not seen in this step. Thus, a wide decision border may lead to a high false positive rate, since, in the test, a number of outliers may be located inside it, such as shown in Figure 1(a). and be, therefore, erroneously labeled as target class.

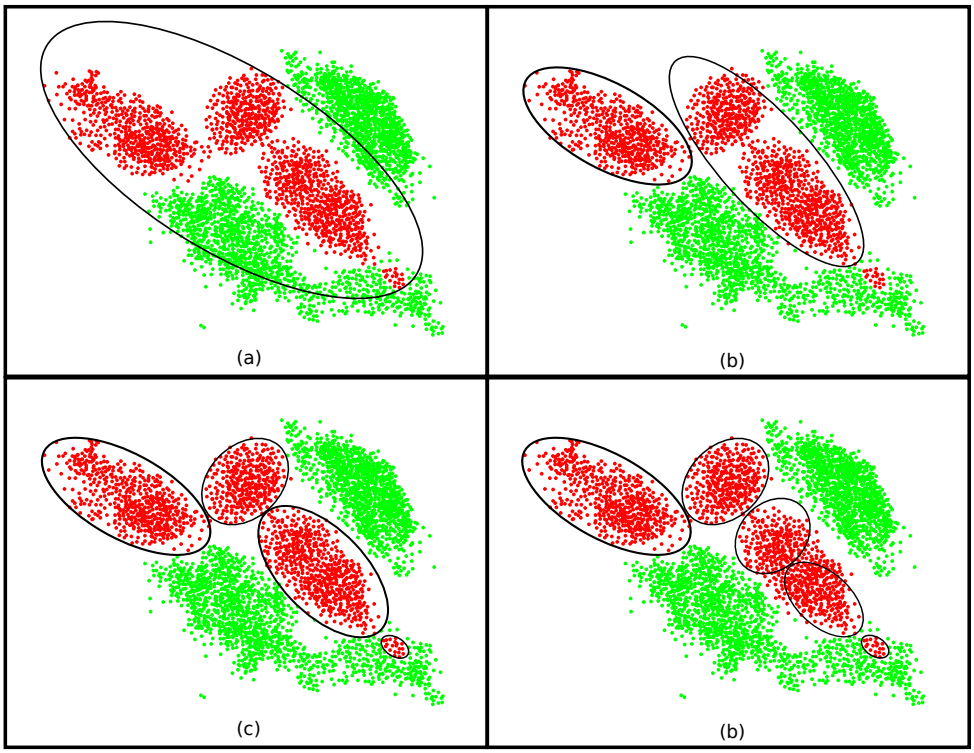
In order to avoid that situation, one may configure a reduced decision border, allowing a percentage of the target training instances outside the decision region, as shown in scenario (b). In the example, besides not being able to achieve such objective, many instances from an specific distribution in the target class are not embraced by the decision region. Thus, instances from this distribution may be erroneously labeled as outliers.

To avoid the problems explained in scenarios (a) and (b), a possible solution is to identify the different data distributions, i.e., modes, present in the target class and train an OCC for each mode. With this approach, the decision region for each OCC tends to be simpler than using the original data, leading to more precise classification.

Figure 2 shows the decision regions for the same problem exposed in Figure 1 using (a) one, (b) two, (c) four and (d) five Gaussian Data Descriptors. The scenario depicted in Figure 2(a) is the same of Figure 1(a). In Figure 2(b), two OCCs are used. In this case, it can be noted that fewer outlier instances are in the decision regions of both OCCs than in Figure 2(a). However, some target instances are not within the decision region of none of the OCCs. The

scenario shown in Figure 2(c) uses four OCCs. In this case, the target class is almost perfectly separated from the outlier class. Figure 2(d) presents the scenario using five OCCs, where some target class instances are not embraced by none of the OCCs.

Figure 2 – Decision regions using (a) one, (b) two, (c) four and (d) five Gaussian one-class classifiers.



Source: The author.

It is important to remark that using a high number of OCCs, i.e., more OCCs than the number of modes present in the data, may lead to inaccurate results. The reason is that training an OCC with data excessively segmented leads to a reduced decision region and a high false negative rate. So an important aspect is to define the number of modes present in the data to avoid such impairments.

1.2 OBJECTIVES

This research aims to improve the classification performance in one-class decomposition tackling the issue of complex intra-class data distribution using multiple OCC for each class. More specifically, we aim to answer the following research questions: (1) Does the adoption of One-Class Classifiers ensembles leverage the classification performance in one-class decomposition? (2) Does the use of data segmentation for the pool generation and OCC ensemble selection lead to good results?

We propose two methods for multi-class classification using one-class decomposition, named MODES and DBDES. Both methods decompose the multi-class problem into one-class problems, separate the training data by class, segment the training data of each class and train an OCC for each cluster. The data segmentation aims to improve the quality of the classifiers for databases that present complex data distribution and presence of noise. With this approach, we aim to improve the classification performance in databases with such peculiarities.

MODES uses K-means algorithm with different numbers of clusters for data segmentation. A set of cluster validity indices (ARBELAITZ et al., 2013) is used to define the number of clusters present in the training data (this subject is better presented in Section 2.1.6). The execution of many cluster validity indices results in different numbers of clusters. The main motivation for this approach is that the number of cluster, commonly, is not known a priori and the determination of the ideal number of clusters for a database is a difficult task and it still is an open problem (KOLESNIKOV; TRICHINA; KAURANNE, 2015).

On the other hand, DBDES uses OPTICS for data segmentation, instead of K-means. This approach aims to improve the quality of the OCCs by capturing clusters with arbitrary shapes and different densities and to avoid the problem of the definition of the number of clusters (DBDES does not require parameterization for the number of clusters). Although OPTICS is a computationally intensive algorithm, it is expected to be faster than executing K-means a series of times and computing the cluster validity indices.

1.3 CONTRIBUTIONS

The main contributions of this work are: (1) two OCC decomposition methods capable of dealing with complex intra-class data distribution; (2) an evaluation of the performance of state-of-the-art one-class decomposition techniques.

1.4 ORGANIZATION

The rest of this text is organized as follows. Chapter 2 presents the background of clustering, one-class classification, MCS, Dynamic Ensemble Selection (DES), and other techniques for a better understanding of the following chapters. Chapter 3 details the proposed techniques, named One-class Dynamic Ensemble Selection for Multi-class problem (MODES) and

Density-Based Dynamic Ensemble Selection (DBDES). Chapter 4 shows the experimental configurations and results obtained by the proposed techniques and a comparison with literature in three experiments. The final remarks and future works proposal are presented in Chapter 5.

2 BACKGROUND

This Chapter presents concepts about clustering, one-class classification, and Multiple Classifiers Systems (MCS) that are necessary for a better understanding of the rest of this document. Additionally, we present important works related to MCS in the context of one-class classification and one-class decomposition.

2.1 CLUSTERING

The main objective of clustering is to create meaningful groups of data objects from an unlabeled database so that elements from a cluster are similar while elements from different clusters are dissimilar. Clustering algorithms can be broadly divided into partitioning, hierarchical, density-based, and grid-based (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001).

Partitioning algorithms divide the data into a set of k disjoint (or flat) clusters, where each data instance belongs to one cluster. Hierarchical algorithms create a hierarchical structure of the data, which enables the extraction of a series of nested clusters. Density-based algorithms search for contiguous areas in the feature space where the density of instances surpasses a given threshold. Grid-based algorithms divide the feature space into a finite number of cells to compute the statistics of each region and, the clustering procedure is executed using the cells instead of the data instances directly.

Sections 2.1.1 to 2.1.4 describe these families of clustering algorithms. In addition, three clustering algorithms that are relevant for better understanding the proposed methods (K-means, DBSCAN, and OPTICS) are explained in Section 2.1.5. Section 2.1.6 presents measures used to evaluate the quality of the partitions, known as Cluster Validity Indices.

2.1.1 Partitioning

Classical partitioning algorithms, such as K-means and K-medoid (JAIN, 2010), segment the data into a predefined number of complementary clusters. These methods, also known as center-based partitioning algorithms, can be interpreted as a parametric approach that assumes that the unknown density of the data is composed of a mixture of k densities, one for each of the k clusters in the data(CAMPELLO et al., 2020). The most adopted approach to

implementing center-based partitioning algorithms is using iterative relocation algorithms. An initial set of k instances is selected - randomly or based on some heuristic - as cluster centroids and, then, the data instances are iteratively relocated among clusters optimizing an objective function (the sum of the squared distances from the instances to the centroids of the clusters, for example) until convergence.

Center-based partitioning algorithms generally are easy to implement, however they may encounter difficulties in identifying clusters of arbitrary shapes. Since these algorithms use the distance from the object to the centroids to determine the membership of that object to the clusters, only ellipsoid-shaped clusters are discovered (ESTER et al., 1996).

Additionally, center-based partitioning algorithms require the configuration of the hyper-parameter k. This is not a simple task and usually involves wrapper procedures, where the clustering algorithm is executed a number of times with different values for k, then, the best value is selected by evaluating the quality of each partition using a predefined criterion(JAIN, 2010). The evaluation criterion can be given by cluster validity indices (ARBELAITZ et al., 2013).

2.1.2 Hierarchical

Unlike partitioning algorithms, hierarchical clustering algorithms do not segment the data into a determined number of clusters at once. A structure of nested clusters is created through a series of partitions that can be performed either in an agglomerative (bottom-up) or divisive (top-down) way. The agglomerative approach starts with each data instance in a single cluster. In the next iterations, the nearest clusters are merged. The process terminates with all data instances in a single cluster. The divisive approach, on the other hand, starts with a single cluster containing all data instances and, iteratively, splits the clusters until there are only clusters with one data instance each (EVERITT et al., 2011).

The hierarchy of clusters can be represented as a merging tree, called dendrogram, which presents each stage of the division/agglomeration process. Each node of the dendrogram represents a cluster, where the root represents the largest cluster, which contains all the data instances and the leaves represent the clusters that contain only one data instance each. The height of the blocks in a dendrogram represents the distance between clusters.

Most hierarchical algorithms construct a representation of the data that expresses the hierarchical clustering structure but do not explicitly build the clusters. Thus, the users of such algorithms must define the number of clusters at some point if they interested in extracting

the clusters - rather than the complete hierarchy of the clustering structure. A commonly used approach is to select one of the partitions in the nested sequences of clusters by cutting the dendrogram at a selected height. Large changes in fusion levels are taken to indicate the best cut (EVERITT et al., 2011).

A drawback of hierarchical algorithms is the fact that cluster divisions or fusions are irreversible, hence, mistaken decisions can not be corrected (EVERITT et al., 2011). However, this inflexibility is important for diminishing the computational costs, since it prevents the need for a combinatorial number of different solutions.

Examples of relevant hierarchical algorithms are Single Link, Complete Link, BIRCH (Balance Iterative Reducing and Clustering using Hierarchies), and CURE (Cluster Using REpresentatives) (XU; WUNSCH, 2005).

2.1.3 Density-based

Density-based algorithms search for clusters identifying dense contiguous regions on the feature space separated by low-density regions. Data instances in dense regions belong to the clusters while data instances in low-density regions are considered noise or outliers (CAMPELLO et al., 2020). The density of a region must surpass a given threshold for that region to be considered a cluster.

Density-based algorithms do not require the number of clusters as an input hyper-parameter. In addition, it is a non-parametric approach, that is, it does not expect the data to follow Gaussian or other parametric distributions and, thus, can identify clusters of arbitrary shapes in databases and deal well with noisy patterns in the data (CAMPELLO et al., 2020).

Some of the most relevant representatives of density-based algorithms are DBSCAN and OPTICS, which will be presented in details in Sections 2.1.5.2 and 2.1.5.3, respectively.

2.1.4 Grid-based

Grid-based clustering segment the data into a finite number of cells to form a grid structure. Then, the clustering operations are made on the cells instead of the raw data. The regions with a higher density than their surroundings are considered clusters. This approach reduces the computational costs of the clustering procedure since the number of cells is smaller than the number of data instances (AGGARWAL; REDDY, 2014). STatistical INformation Grid-based

(STING), Wavelet-based Clustering (WaveCluster), and CLIQUE (Clustering In QUEst) are popular examples of grid-based clustering algorithms.

2.1.5 Algorithms

In this section, we present the clustering algorithms relevant for understanding the proposed methods. K-means is used by MODES (FRAGOSO et al., 2021) for the clustering step and OPTICS is used in DBDES for the same purpose. In addition, we also describe the DBSCAN algorithm to facilitate the understanding of OPTICS, since the latter is an extension of the former.

2.1.5.1 K-means

K-means is one of the simplest, yet powerful, clustering algorithms. Its center-based approach uses the mean of all instances in a cluster as its centroid. It starts selecting k random instances in the database as the initial clusters centroids. Then, the other instances are assigned to the cluster with the nearest centroid. The new centroids of the clusters are computed as the mean of all instances in a cluster. With the new centroids, the instances are relocated among the clusters recomputing the distances to the centroids. The algorithm terminates when a convergence criterion is met, for example, no changes are made in the clusters or the mean squared distance from the instances in the clusters to their centroids stop decreasing significantly after some iterations(JAIN; MURTY; FLYNN, 1999).

The algorithm for K-means is described as follows(JAIN; MURTY; FLYNN, 1999; JAIN, 2010):

1. Select an initial partition with k clusters centers coinciding with k random instances in the database.
2. Create a new partition by assigning the instances to the cluster with the nearest centroid.
3. Re-compute the centroids as the means of all instances in each cluster.
4. If a convergence criterion is not met, go to step 2.

2.1.5.2 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (ESTER et al., 1996) is a clustering algorithm that partitions the database based on the density of the feature space. DBSCAN was one of the earliest density-based clustering algorithms to be proposed and is still one of the most studied algorithms (LUCHI; RODRIGUES; VAREJÃO, 2019).

DBSCAN requires two hyper-parameters to define the density threshold: ε represents the radius of a region and $minPts$ represents the minimum number of data instances that must be in a region (with radius ε) for that region to be considered a cluster.

The algorithm starts with an instance x and evaluates its neighborhood. For an instance x in the database Γ, the ε-neighborhood $N_\varepsilon(x)$ is denoted as $\{x' \in \Gamma \mid distance(x, x') \leq \varepsilon\}$. If $|N_\varepsilon(x)| \geq minPts$, then x is considered a core instance and a new cluster is created containing all the instances in $N_\varepsilon(x)$. This process is repeated for all $x \in \Gamma$ and terminates when all instances are visited. The clusters are merged when a core instance is in the ε-neighborhood of another core instance. If an instance x' is not a core instance, but it is in the ε-neighborhood of any core instance, i.e., $x' \in N_\varepsilon(x)$ and $|N_\varepsilon(x)| \geq minPts$, x' is considered a border instance. Data instances that were not added to any cluster, i.e, that are not core instances or border instances, are considered noise.

DBSCAN is a very powerful algorithm, however, it presents some drawbacks. To name a few, the choice of values for its hyper-parameters is a difficult task and it may encounter difficulties in identifying clusters with different densities (ANKERST et al., 1999; KARAMI; JOHANSSON, 2014; KHAN et al., 2014). With a single density threshold hyper-parameterization for all the database, clusters with distinct densities are hardly discovered. To address this issue, two main approaches may be adopted: use more adaptive distance measures or a hierarchical clustering extraction (CAMPELLO et al., 2020).

2.1.5.3 OPTICS

Ordering Points To Identify the Clustering Structure (OPTICS) (ANKERST et al., 1999) is an extension of DBSCAN that aims to address the problem of clusters with varying densities and the configuration for its hyper-parameters. Unlike DBSCAN, Ordering Points To Identify the Clustering Structure (OPTICS) does not explicitly segment the data but, instead, creates an augmented ordering of the database representing its density-based clustering structure,

from which either flat or hierarchical clusters can be extracted.

A value for $minPts$, which is used in the same way as in DBSCAN is required. The second hyper-parameter, ε, however, is optional and is used only for reducing runtime complexity. OPTICS uses ε as an upper limit for the neighborhood radius. Therefore, OPTICS checks for a spectrum of all different radii of size $\varepsilon' \leq \varepsilon$. The size of the neighborhood increases while there are not at least $minPts$ data instances within it. This allows OPTICS to find clusters with different densities. Low-density clusters have large values for ε' while high-density clusters have low values for ε'. In practice, this is equivalent to extending DBSCAN to simultaneously cluster a database for a multitude of values for ε (AGGARWAL; REDDY, 2014). By default, ε is set to infinite in OPTICS.

Two important concepts are introduced in OPTICS, besides those from DBSCAN: core distance and reachability distance. The core distance of an instance x is computed as the distance from x to its $minPts$-th neighbor, i.e., it is the minimum value of radius ε' required for x to be regarded as a core instance. For an instance x with $|N_\varepsilon(x)| < minPts$, that is, a non-core instance, the core distance is set to UNDEFINED. The reachability distance of an instance x' with respect to an instance x is defined as the maximum between the core distance of x and the distance between x' and x, if x is a core instance. Otherwise, the reachability distance is set to UNDEFINED.

With the order in which the instances were processed and their respective reachability distances, it is possible to draw a reachability plot, which is an interesting graphical representation of the cluster ordering of a database. In the graph, the data instances are disposed along the horizontal axis in the order that they were processed by OPTICS. The vertical axis represents the reachability distances of each data instance to its predecessor. Regions containing contiguous data instances with low reachability distances (or valleys) represent clusters. The lower the reachability distances, the denser is the cluster. Of course, these valleys must have at least $minPts$ instances to be considered clusters.

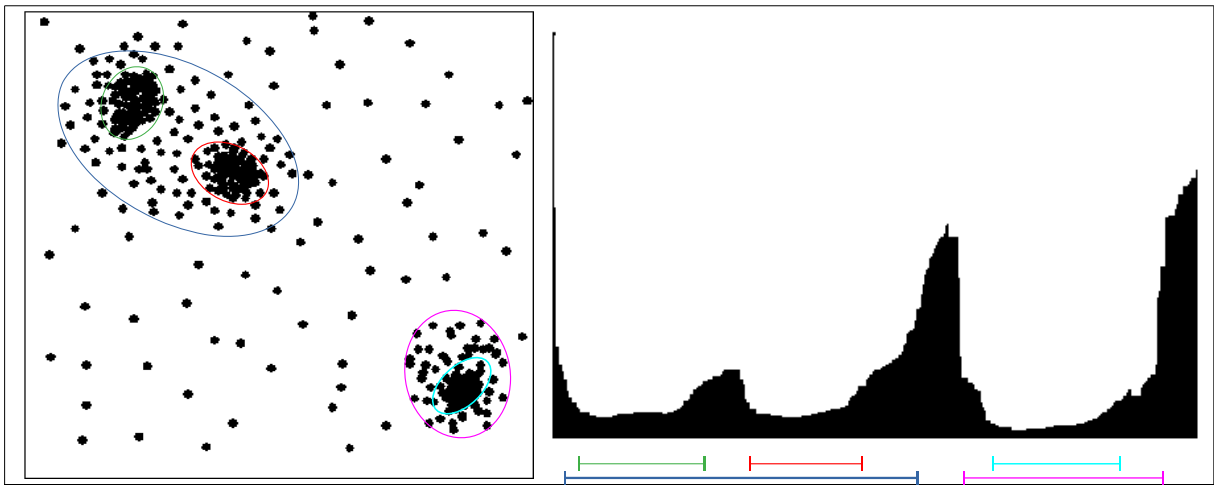
OPTICS is able to extract flat clusters, in a DBSCAN fashion, or hierarchical clusters. For the flat partition, it is necessary to provide a value for $\varepsilon' < \varepsilon$ for the algorithm to select one of the partition solutions. This can be interpreted as a horizontal cut in the reachability plot at $y = \varepsilon'$. Data instances with a higher reachability distances than the cut-off represent noise and separate the clusters(HAHSLER; PIEKENBROCK; DORAN, 2019).

For the hierarchical cluster extraction, the algorithm scans the cluster ordering looking for significant changes between the reachability distances of one instance and its successor in the

ordering. A threshold ξ is defined so that a difference greater than $\xi\%$ is interpreted as the start of a cluster, if the reachability distance is descending, or the end of a cluster, otherwise.

Figure 3 shows a bi-dimensional database with hierarchical clusters and its reachability plot. It is easy to identify the hierarchical clustering structure in the plot. For example, the clusters marked in green and red are sub-clusters of the cluster marked in blue. In the plot, two nested valleys, representing the sub-clusters, are inside the larger valley that represents the cluster marked in blue.

Figure 3 – Reachability plot for a database with hierarchical clusters of different sizes and densities. Each "valley" in the plot represents a cluster where the steep areas represent the start and end of clusters.



Source: Adapted from (ANKERST et al., 1999)

One of the most relevant strengths of OPTICS is the fact that it is not limited by a single hyper-parameter setting. The cluster ordering stores information corresponding to a broad range of values (ANKERST et al., 1999). Another advantage is a low dependency on the values for the hyper-parameters. According to the authors of OPTICS, "the values have just to be large enough to yield a good result". Values between 10 and 20 for $minPts$ produced good results (ANKERST et al., 1999). Moreover, using the default value for ε (infinite), all clustering levels will be considered. Obviously, this increases the computational costs, since the neighborhoods for each object include all data instances. On the other hand, smaller values for ε may result in more data instances having UNDEFINED reachability distance, which leads to lower density clusters (those whose core instances are core instances only for neighborhoods larger than ε) not being detected by OPTICS.

2.1.6 Cluster validity indices

Many clustering algorithms are heavily affected by the characteristics of the data and by their hyper-parameters (KIM; RAMAKRISHNA, 2005). Thus, the definition of values for the hyper-parameters that leads to a data partition that fits the data optimally is a critical task for such algorithms. A Cluster Validity Index (CVI) may be used as a guideline for this task. CVI are methods that provide an evaluation of the clustering quality by defining a relation between intra-cluster cohesion and inter-cluster separation (KIM; RAMAKRISHNA, 2005).

For example, if we aim to find the optimal number of clusters (the k parameter in K-means, for instance), we need to segment the data using a set of different numbers of clusters and evaluate each partition using a CVI.

However, since clustering is an unsupervised learning task, there is no single best validity index for all types of clustering tasks (KIM; LEE; KANG, 2018). Each CVI presents advantages and disadvantages for each specific problem, i.e., different partitions may be considered optimal for an arbitrary database according to different CVIs, and the choice of the CVI depends on the data and the aim of clustering.

Some popular CVIs are Silhouette (KAUFMAN; ROUSSEEUW, 2009), Dunn (DUNN, 1974), and Calinski and Harabasz (CALIŃSKI; HARABASZ, 1974). The R package *NbClust* (CHARRAD et al., 2014) implements all of them (and a total of 30 CVIs).

2.2 ONE-CLASS CLASSIFICATION

One-class classification is a kind of problem where it is difficult or expensive to obtain counter-examples, such as transaction fraud recognition, machine fault detection, anomaly detection, etc. Thus, only instances belonging to one class are available. Objects from this class are called target objects and all other objects are called outliers. An One-Class Classifier (OCC) tries to describe the set of objects from the target class and to identify which (new) objects resemble this training set (TAX; DUIN, 2004).

One-class classifiers are specially categorized into density methods, boundary methods, and reconstruction methods (KHAN; MADDEN, 2009). Density methods compute the probability density function (PDF) of the target class data. In the test, the PDF value for the test instance is compared with a threshold. This technique requires a large amount of data to overcome the curse of dimensionality (DUDA; HART; STORK, 2012). Gaussian and Parzen

OCCs are examples of density methods (DUDA; HART; STORK, 2012). Boundary methods build a model by optimizing a closed boundary around the target data and the acceptance of a test instance is given by the distance to the fitted model. Boundary methods require less training data than density methods. Among the boundary methods, Support Vector Data Descriptor (SVDD) (TAX; DUIN, 2004), One-class Support Vector Machine (OCSVM) (SCHÖLKOPF et al., 2001), and Nearest Neighbor (TAX; DUIN, 2000) can be remarked. Reconstruction methods assume a model of data generation process and estimate the parameters of this model in the training. Self-organizing Maps and Learning Vector Quantization are classified as reconstruction methods (KOHONEN et al., 2001).

One-class classifiers have been successfully applied to a variety of applications such as real-time activity error detection (DAS et al., 2016), text classification (MANEVITZ; YOUSEF, 2001), and authorship verification (KOPPEL; SCHLER, 2004). Recently, one-class decomposition has gained attention from researchers. This technique separates the training data by class and trains one-class classifiers for each class. Then, the classification results are combined. Krawczyk et al. (KRAWCZYK; WOŹNIAK; HERRERA, 2015) showed the usefulness of one-class decomposition in some specific cases, such as complex data distribution, imbalanced data, presence of noise, and a high number of classes. For these cases, the research shows that using an OCC for each class of the original problem and aggregating them is more effective than using classical binary decomposition strategies. Another important application of one-class decomposition is when the number of classes is not known apriori, since one-class classifiers are trained for each class separately (KRAWCZYK; WOŹNIAK; CYGANEK, 2014).

2.2.1 OCC models

This section briefly describes the OCC models used in this work: Gaussian Data Descriptor (GaussianDD), Parzen Data Descriptor (ParzenDD), SVDD, and Minimum Spanning Tree Data Descriptor (MSTDD).

GaussianDD is a density method that surrounds the training data by fitting a simple Gaussian distribution in the feature space. A limitation of this model is that it relies on the assumption that the data is normally distributed. It is a simple and fast OCC model, but it can present better results than more complex OCCs, especially if the data do not contain multiple clusters (LIU et al., 2016).

ParzenDD is a non-parametric density estimator that computes a kernel for each training

instance and, then, estimates the probability density function of the data by defining a linear combination of these kernels. Test instances are classified by checking whether they may be drawn from the estimated distribution (DUDA; HART; STORK, 2012).

MSTDD is a non-parametric model that builds a minimum spanning tree on the training data aiming to capture the underlying structure of the data. The edges of the tree are considered an additional set of virtual target instances that can help to model the target class distribution. MSTDD performs well in high-dimensional and small sample-size problems in comparison to other existing one-class classifiers (JUSZCZAK et al., 2009).

SVDD (TAX; DUIN, 2004) defines its decision boundary, in the training, creating a spherical surface that contains all (or most of) the target training instances within the smallest radius. A test instance is classified by computing its distance to the center of the hyper-sphere. Instances outside the hyper-sphere are considered outliers. The main drawback of SVDD is that it requires the estimation of many parameters.

2.3 MULTIPLE CLASSIFIERS SYSTEMS

The main rationale behind the use of Multiple Classifiers Systems (MCS) is that, following the no free lunch theorem (WOLPERT, 1996), there is not a single classifier that is competent to deal with test instances from all regions of the input space. Thus, MCS combine a set of classifiers expecting to outperform any individual classifier.

MCS are commonly divided into 3 steps: generation, selection, and fusion/aggregation. In the generation step, a pool of classifiers is generated, seeking accuracy and diversity among the classifiers. The purpose of the generation is to form sets of classifiers that complement each other, that is, to generate a pool that contains competent classifiers for different regions of the feature space. The most effective generation techniques train the base classifiers with different feature sets or different training sets or, yet, use different classifier models (heterogeneous pools) (CRUZ; SABOURIN; CAVALCANTI, 2018).

The goal of the selection step is to select from the pool of classifiers the most competent classifiers for the problem. Since this is an optional step, it is omitted in some techniques. It is possible to build MCS skipping this step and performing a static combination of all classifiers in the pool. However, the advantages of applying the selection step are widely known (KUNCHEVA, 2014; CRUZ; SABOURIN; CAVALCANTI, 2018).

The selection task can be static or dynamic (WOŹNIAK; GRAÑA; CORCHADO, 2014). In

static selection, a subset of classifiers is formed in the training phase. Then, in the test phase, the selected ensemble is used to classify all the test instances. Dynamic selection is performed during the classification of test instances. For each test instance x, a region of competence is computed, usually based on the k-Nearest Neighbors of x. The best performing classifiers in the region of competence are selected to classify x. Dynamic selection can be divided into two approaches (CRUZ; SABOURIN; CAVALCANTI, 2018). In Dynamic Classifier Selection (DCS), for each test instance x, the most competent classifier is chosen. In Dynamic Ensemble Selection (DES), a set composed of the most competent classifiers in the pool is selected.

The fusion, or aggregation, step is responsible for combining the output of each individual classifier to provide the final output of the MCS.

2.3.1 Aggregation techniques

In this work, we employ four aggregation techniques: Mean Support (MEAN), Maximum Support (MAX), Decision Templates (DTs), and Error Correcting Output Codes (ECOC).

MEAN - Mean Support, or simple average (KITTLER et al., 1998; ZHOU, 2012), averages the output of the individual learners to compose the aggregated output. It computes the mean scores of each class across the classifiers and assigns the test instance to the class with the maximum mean score.

MAX - The maximum support aggregation (KITTLER et al., 1998; WOŹNIAK; GRAÑA; CORCHADO, 2014) assigns the class with the maximum score among the classifiers to the test instance. In the one-class classification task, this means labeling the test instance as belonging to the class used to train the OCC with maximum support among all.

DTs - Decision Templates (KUNCHEVA; BEZDEK; DUIN, 2001; ZHOU, 2012; KRAWCZYK; WOŹNIAK; HERRERA, 2015) represent typical values for the outputs of classifiers from a pool. In the training phase, the decision templates are encoded as a matrix where the number of columns is equal to the number of classifiers in the pool and the number of rows is equal to the number of classes of the classification problem. Each cell $DT_{l\lambda}$ of that matrix is computed as the average support of a classifier λ for data instances from the class label l. In the test phase, the decision profile of a test instance, that is, the array containing the classification outputs of the classifiers in the pool for that instance, is compared with the decision templates. The test instance is then assigned to the class whose decision template is the most similar to the test instance's decision profile. The similarity can be measured with a distance metric, such

as euclidean distance, for example.

ECOC - Error Correcting Output Codes (ZHOU, 2012; KRAWCZYK; WOŹNIAK; HERRERA, 2015) is a framework originally designed to reconstruct the multi-class classification problems from the decisions of binary classifiers. The idea is to avoid solving the multi-class problem directly and to break it into dichotomies instead (KUNCHEVA, 2014). It is primarily divided into two steps: coding and decoding (ZHOU, 2012). In the coding step, each class of the original multi-class problem is encoded as a codeword (a unique sequence of zeroes and ones) and placed as rows of a matrix (coding matrix). Then, a classifier is trained for each column of the coding matrix, aiming to separate the classes designed with one from those designed with zero in the column. The decoding step starts with a test instance being classified by all trained classifiers. The output of each classifier is concatenated resulting in a binary string. Then, the test instance is assigned to the class whose codeword is closest to the output string of the test instance.

2.4 RELATED WORKS

Multiple Classifiers Systems have recently been adopted in the context of one-class problems. In (KRAWCZYK; WOŹNIAK; CYGANEK, 2014), the authors used a clustering-based approach to generate ensembles of One-class Support Vector Machines. The data from the one-class problem is segmented and, for each cluster, an OCC is trained. This MCS does not include a selection step, i.e., all the classifiers in the pool are used to classify all test instances. A classifier trained with instances from a specific region of the feature space may not be competent to classify test instances from other regions. Hence, the combination of all classifiers in the pool may hinder the performance. Dynamic selection techniques aim to mitigate this problem.

A Dynamic Classifier Selection (DCS) method for OCC was proposed in (KRAWCZYK; WOŹNIAK, 2016). DCS techniques select a single classifier in execution time to classify each test instance. In that work, a single OCC is selected from a pool of heterogeneous OCC models. The work showed that dynamic selection techniques work well in the context of OCC because during the training phase it was possible to generate a diverse pool of classifiers. However, in DCS, the performance depends on the quality of the algorithm that selects the classifier from the pool (KO; SABOURIN; JR, 2008). Since only one classifier is responsible for the classification of a test instance, the performance may be impaired if this selected classifier is not competent for the classification of the current test instance. Complex data distribution or the presence of

noise may worsen this problem since the chances of selecting a non-competent classifier are higher.

This problem may be alleviated using DES, which selects a subset of classifiers from the pool to compose the ensemble responsible for the classification decision for a test instance. To the best of our knowledge, the first DES for one-class decomposition was proposed by Krawczyk et al. (KRAWCZYK et al., 2018).

In that work, an OCC is trained for each class of the original problem. To classify a test instance, the method selects an ensemble composed of the OCCs trained with the data from the classes in the neighborhood of that test instance. Then, the original multi-class problem is recomposed by aggregating the decisions of the selected one-class classifiers. Details of intra-class data distribution are not taken into account when using only one OCC for each class, which may impair the classification performance. However, to the best of our knowledge, literature dynamic selection techniques for one-class decomposition use only one OCC for each class of the original problem.

Table 1 – Comparison involving Density-Based Dynamic Ensemble Selection (DBDES), One-class Dynamic Ensemble Selection for Multi-class problems (MODES), and other techniques that use one-class decomposition, OCC ensemble, for each one-class problem, clustering-based approach, and/or dynamic selection (DS)

Ref	Title	Decomp.	Ensemble	Cluster	DS
(KRAWCZYK; WOŹNIAK; CYGANEK, 2014)	Clustering-based ensembles for one-class classification	Yes	Yes	Yes	No
(KRAWCZYK; WOŹNIAK; HERRERA, 2015)	On the usefulness of one-class classifier ensembles for decomposition of multi-class problems	Yes	No	No	No
(KRAWCZYK; WOŹNIAK, 2016)	Dynamic classifier selection for one-class classification	No	No	No	DCS
(KRAWCZYK et al., 2018)	Dynamic ensemble selection for multi-class classification with one-class classifiers	Yes	No	No	DES
(LIU et al., 2016)	Modular ensembles for one-class classification based on density analysis	No	Yes	Yes	No
(GÖRNITZ et al., 2017)	Support Vector Data Descriptions and k-Means Clustering: One Class?	No	Yes	Yes	No
(WOJCIECHOWSKI; WOŹNIAK, 2020)	Employing Decision Templates to Imbalanced Data Classification	No	Yes	Yes	No
(TSAI; LIN, 2021)	Feature selection and ensemble learning techniques in one-class classifiers	No	Yes[1]	No	No
(FRAGOSO et al., 2021)	One-class Dynamic Ensemble Selection for Multi-class problems (MODES)	Yes	Yes	Yes	DES
	Density-Based Dynamic Ensemble Selection of OCC Ensembles for Multi-class problems (DBDES)	Yes	Yes	Yes	DES

Source: The author.

[1] Only the majority class is used for training

Other methods (LIU et al., 2016; GÖRNITZ et al., 2017) successfully use multiple one-class classifiers, however, they were not used for one-class decomposition. Even so, they represent relevant advances in this research field. A detailed analysis of ensembles of OCCs for binary problems was carried out in (TSAI; LIN, 2021). This work also evaluates the use of feature selection techniques with OCCs ensembles.

Additionally, in (WOJCIECHOWSKI; WOŹNIAK, 2020), clustering-based ensembles of multi-class classifiers are used to tackle the class imbalance problem. The data from each class is segmented, using the OPTICS algorithm, and a Decision Template (KUNCHEVA; BEZDEK; DUIN, 2001) is trained for each cluster. The relation of this work with the proposed techniques is not as explicit as the works previously mentioned in this section since it does not use OCCs. However, it served as an inspiration to the proposed techniques.

Table 1 summarizes the characteristics of the related works compared with the proposed approaches, MODES and DBDES. We evaluate three aspects: (1) Can the technique handle multi-class problems, i.e., can it be used for one-class decomposition? (2) Does the technique adopt OCC ensembles for the one-class problem? (3) Does the technique adopt dynamic selection DS for the one-class problem? MODES and DBDES are the only techniques that exhibit all these characteristics.

3 PROPOSAL

In this chapter, we present the proposed techniques, namely One-class Dynamic Ensemble Selection for Multi-class problems (MODES) and Density-Based Dynamic Ensemble Selection (DBDES). Both techniques aim to tackle multi-class classification by decomposing the original problem into one-class problems, generating pools of classifiers for each one-class problem, and using dynamically selected One-Class Classifier (OCC) ensembles to classify each test instance. They deal with complex intra-class data distribution by segmenting the training data of each original class and training an OCC for each cluster.

MODES segments the data using a center-based clustering algorithm, such as K-means. This kind of algorithm requires a value for the hyper-parameter k, which represents the number of clusters. The definition of the ideal value for k is not a simple task. To tackle this issue, MODES computes a set of 13 cluster validity indices (ARBELAITZ et al., 2013; CHARRAD et al., 2014) for the data from each class. Each index evaluates partitions using from 2 to 10 clusters and outputs the number of clusters that it considers ideal for the data. This range was chosen in experiments, where we identified that, in general, using more than 10 clusters worsens the classification accuracy of MODES. The goal is to minimize the dependency on the determination of the ideal number of clusters and, thus, to approximate the number of partitions to the number of modes present in the data. Moreover, training the pool of OCCs with different input data (different partitions) should increase diversity, which is an important aspect for ensemble methods.

DBDES, on the other hand, uses Ordering Points To Identify the Clustering Structure (OPTICS) algorithm for the clustering step. It is a density-based clustering algorithm that does not require the number of clusters apriori. Furthermore, OPTICS is able to identify clusters with arbitrary shapes, can extract both flat and hierarchical clusters from the data, and presents low dependency on the hyper-parameters (ANKERST et al., 1999). Thus, employing OPTICS for the clustering step of a clustering-based pool generation may represent an improvement both in the quality of the OCCs in the pool and runtime complexity. Additionally, the hierarchical clustering extraction by OPTICS allows non-exclusive partitions, that is, a data instance may be a member of more than one cluster. This increases the diversity of the input data for the generation of the pool of OCCs.

Sections 3.1 and 3.2 detail each of the proposed techniques.

3.1 MODES

MODES is primarily composed of training and test phases. In the training phase, the data is separated by class and segmented into different partitions, then an OCC is trained for each cluster and added to a pool. This phase results in a pool for each different partition of each class. The test phase selects an ensemble of OCCs to classify a test instance for each one-class problem (if x belongs or not to the class). Sections 3.1.1 and 3.1.2 detail the training and test phases, respectively. Additionally, we provide a toy instance for MODES in Section 3.1.3.

3.1.1 Training phase

The training phase consists of separating the training data by class and generating a pool of one-class classifiers for each class. The data is segmented into different partitions and an OCC is trained for each cluster of each partition. Algorithm 1 and Figure 4 describe the training phase, which follows these steps:

1. Separate the training instances by class (lines 2 to 4): let L be the set of class labels, the instances in the training set Γ are separated by class resulting in $|L|$ training sets: $\{\Gamma_1, \Gamma_2, ..., \Gamma_{|L|}\}$.

2. Compute the cluster validity indices for Γ_l (lines 6 and 7): for each class $l \in L$, a set of cluster validity indices is computed. Each index assesses the data segmentation with 2 to 10 clusters and the best number of clusters indicated by the index is added to the set D.

3. Segment Γ_l (lines 8 and 9): the data from each class is segmented into different partitions, i.e., different sets of clusters $g_i = \{c_1, ..., c_j, ..., c_{d_i}\}$ using a partitioning algorithm, such as K-means. Each partition g_i is obtained using different values $d_i \in D$ as the number of clusters k. This results in $|D|$ different partitions $\{g_1, ..., g_i, ..., g_{|D|}\}$, each containing d_i clusters.

4. Train an OCC for each cluster (lines 10 to 13): for each cluster c_j in g_i, a one-class classifier λ_j is trained and added to the pool p_i. MODES discards clusters containing only one instance, since it may represent an outlier.

5. Repeat items 2 to 4 for each class label $l \in L$, adding the pairs $< g_i, p_i >$ to M_l.

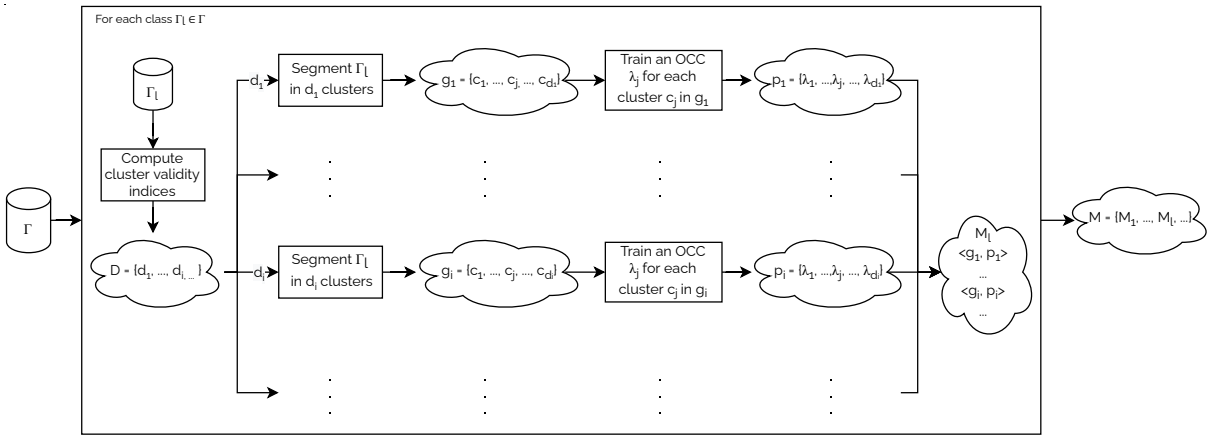
The output of MODES training phase is the array M, which contains $|L|$ entries M_l, one for each class label. Each M_l is an array containing $|D|$ pairs $< g_i, p_i >$, where g_i is Γ_l segmented into clusters $\{c_1, ..., c_j, ..., c_{d_i}\}$ and p_i is a pool composed of d_i OCCs $\{\lambda_1, ..., \lambda_j, ..., \lambda_{d_i}\}$. M_l binds each OCC λ_j to the cluster c_j used to train it.

Algorithm 1: MODES training phase

input : Γ: a training set
output : M: an array with pools of OCCs for each class

$M = \emptyset$
$L =$ labels(Γ) // extract the set of labels
for $l \in L$ **do**
  $\Gamma_l =$ instances(Γ, l) // get instances from class l
  $D = M_l = \emptyset$
  for $index \in$ {*Silhouette, Hartigan, C-index, ...*} **do**
    $D = D \cup$ computeIndex(Γ_l, $index$)
  for $i = 1 : |D|$ **do**
    $g_i = partition(\Gamma_l, d_i)$ // segment Γ_l into $g_i = \{c_1, ..., c_j, ..., c_{d_i}\}$
    $p_i = \emptyset$
    for $j = 1 : d_i$ **do**
      $\lambda_j =$ trainOCC(c_j)
      $p_i = p_i \cup \lambda_j$
    $M_l = M_l \cup < g_i, p_i >$
return M

Figure 4 – MODES training phase.



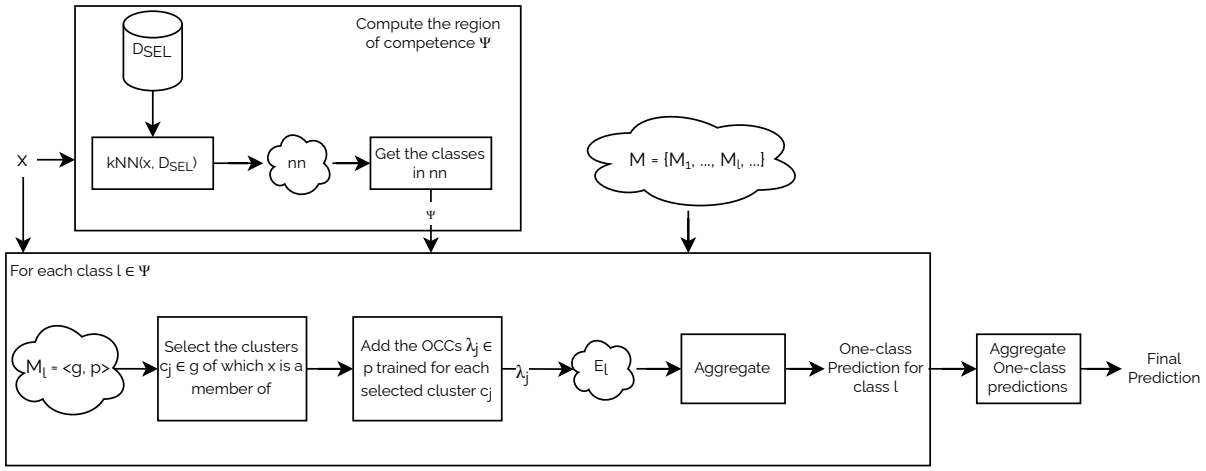
Source: The author.

3.1.2 Test phase

In the test phase, for each test instance x, only the classifiers trained with data belonging to classes present in the neighborhood of x are used. The neighborhood, or region of competence Ψ (with respect to x), is defined as the classes of the k nearest neighbors of x, where k is configured with 3 x $|L|$. For this task, a validation set D_{SEL} is used. Additionally, a threshold is applied so that classes with less than 10% of the instances in the neighborhood are discarded. This approach is based on that proposed in (KRAWCZYK et al., 2018) and aims to remove non-competent classifiers from the ensemble. For each class label l present in the neighborhood of x, an ensemble E_l is dynamically selected to classify x. The classification of x for the l-th one-class problem (i.e, if x belongs or not to class l) is given by the aggregation of the OCCs in E_l. The final classification of x, i.e., the final decision that assigns a label to x, is given by the aggregation of the decisions made by the ensembles E_1, ..., E_l, ..., $E_{|\Psi|}$, generated for each class label l present in the region of competence Ψ with respect to x.

Algorithm 2 and Figure 5 describe the test phase, which follows these steps to classify each test instance x:

Figure 5 – MODES test phase.



Source: The author.

1. Compute the region of competence Ψ (lines 1 and 2): the region of competence is defined as the labels of the k nearest neighbors of x in the validation set D_{SEL}, where k is defined as $3 \times |L|$. A threshold is applied so that classes with less than 10% of the instances in the neighborhood are discarded (KRAWCZYK et al., 2018).

2. Select each class l present in the region of competence Ψ (line 4). If Ψ contains only one class, x is assigned to this class.

3. Select the ensemble E_l (lines 5 to 9): for each pair $< g_i, p_i > \in M_l$, select in g_i the nearest cluster to x, then the OCC in p_i trained with data from that cluster is selected to the ensemble.

4. Aggregate E_l (line 10): aggregate the predictions for the one-class problem using the Mean of probabilities (aggregation technique that presented the best performance in preliminary experiments using simple aggregation techniques, such as Mean of probabilities, Maximum Support (MAX) and Minimum support). This step results in the one-class decision, which is stored in the l-th position of R. At the end of this process, the array R contains the one-class decisions for each class label l in Ψ.

5. Aggregate the decisions in R (line 11): Use an aggregation strategy (Decision Templates (DTs), Error Correcting Output Codes (ECOC), or MAX) to recompose the original multi-class problem from the one-class decisions.

Algorithm 2: MODES test phase

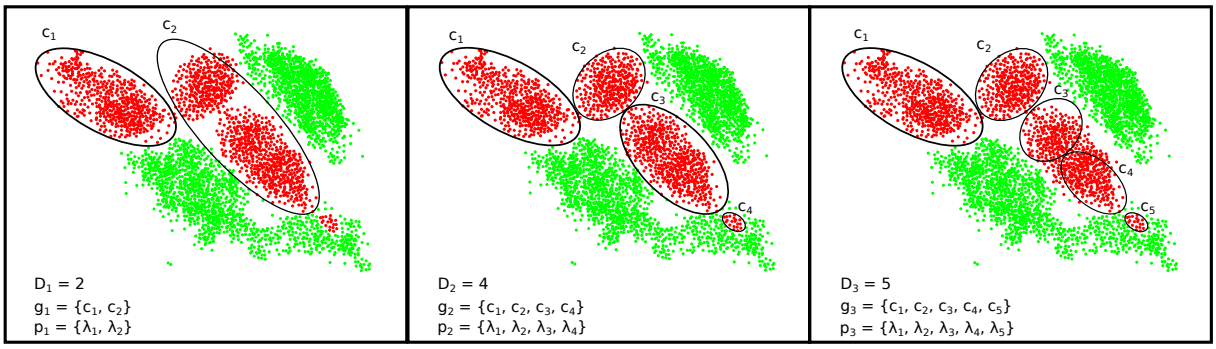
input : x: a test instance
M: an array with pools of OCCs for each class
D_{SEL}: a validation set
output : ω: the predicted class for x

1 $nn =$ kNN(x, D_{SEL})
2 $\Psi =$ labels(nn) // extract the set of labels of the instances in nn
3 $R = \emptyset$
4 **for** $l \in \Psi$ **do**
5 $E_l = \emptyset$
6 **for** $< g_i, p_i > \in M_l$ **do**
7 // $g_i = \{c_1, ..., c_j, ..., c_{|g_i|}\}$ and $p_i = \{\lambda_1, ..., \lambda_j, ..., \lambda_{|g_i|}\}$
8 $j =$ argmin $\{distance(x, g_i)\}$ // index to the nearest cluster
9 $E_l = E_l \cup \lambda_j$
10 $R = R \cup$ mean(E_l, x) // mean aggregation
11 $\omega =$ aggregate(R, x) // DTs, ECOC or MAX
12 **return** ω

3.1.3 Toy example

We present a toy example to demonstrate how MODES works in a visual manner, which helps to understand it. Figures 6 and 7 describe the training and test phases, respectively, for a specific class label $l \in L$ of a hypothetical database. The target class is represented by the red dots while the green dots represent the outliers, i.e., the data from other classes.

Figure 6 – Toy example of the training phase of MODES for a class label l. The cluster validity indices output a set $D = \{2, 4, 5\}$ containing the numbers of clusters to segment the target data. A one-class classifier λ_j is trained for each cluster c_j.

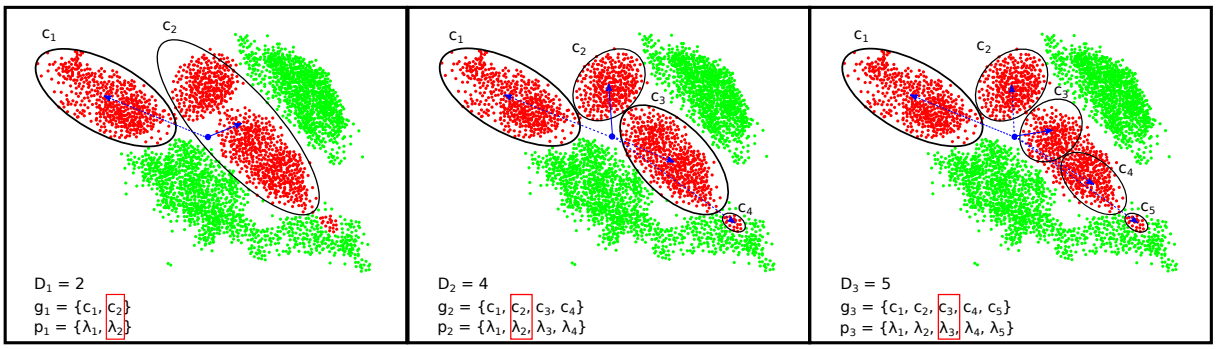


Source: The author.

The first step of the training phase is the computation of the cluster validity indices for the target data, in order to define a set of numbers of clusters to segment the data. Suppose that, in this toy example, the computation of the cluster validity indices outputs the set $D = \{2, 4, 5\}$. Thus, for each $d_i \in D$, MODES segments the target class data into d_i clusters $g_i = \{c_1, ..., c_{D_i}\}$. Then, an OCC λ_j is trained for each cluster $c_j \in g_i$, making up the pool p_i. For instance, for $D_1 = 2$, the target data is segmented into $g_1 = \{c_1, c_2\}$ and the pool of OCCs $p_1 = \{\lambda_1, \lambda_2\}$ is trained. The pairs $< g_i, p_i >$ are added to the array M_l. In this toy example, $M_l = \{< g_1, p_1 >, < g_2, p_2 >, < g_3, p_3 >\}$. Each of these pairs binds a pool of OCCs to the clustered data which it was trained with. The output of the training phase is the array M, composed of the arrays M_l for $l \in L$.

The test phase starts with a test instance x. The region of competence Ψ is defined, based on the neighborhood of x. For each class label $l \in \Psi$, MODES retrieves from the array M (composed in the training phase) the array M_l. MODES then computes, for each g_i, the distance from x to the centers of the clusters. Then, the OCC in p_i trained with the data from the closest cluster is selected to compose the ensemble E_l. In this toy example, since the closest clusters to x are c_2 for D_1, c_2 for D_2 and c_3 for D_3, $E_l = \{\lambda_{12}, \lambda_{22}, \lambda_{33}\}$, where λ_{ij} is the jth OCC from the pool p_i. The one-class classification for the class l is given by the mean

Figure 7 – Toy example of the test phase of MODES for a class label l. MODES computes the euclidean distance between the test instance x (represented by •) and the centroid of each cluster. The OCC trained with data from the closest cluster is is dynamically selected to the ensemble E_l.



Source: The author.

of probabilities for E_l. Suppose that E_l outputs $\{0.75, 0.85, 0.8\}$, then $R_l = 0.8$, where R_l is the mean probability computed by E_l. The final classification is given by the aggregation of each decision R_l for each ensemble E_l, with $l \in \Psi$. Using MAX aggregation, for instance, the test instance would be assigned to the class l which R_l has the highest support in R.

3.2 DBDES

DBDES is a variation of MODES (FRAGOSO et al., 2021), which aims at reducing the runtime complexity by using the OPTICS algorithm for clustering instead of K-means. OPTICS is a density-based clustering algorithm that does not require the number of clusters apriori, can extract both flat and hierarchical clusters from the data, and presents low dependency on the hyper-parameters (ANKERST et al., 1999). Additionally, using OPTICS should improve the quality of the clustering in data with complex distribution (which is the main application for MODES and DBDES), since this algorithm can identify noisy data, clusters with different densities, and clusters with arbitrary shapes.

DBDES can be divided into two main phases: (i) training, where the original multi-class problem, consisting of the set L of classes, is decomposed into $|L|$ one-class problems, the training data for each class is segmented using OPTICS, and the pools of OCCs are generated training an OCC for each cluster; and (ii) test, where, for each test instance, an ensemble of OCCs is dynamically selected based on the membership of that instance to the clusters identified in the training phase, the classification decisions are aggregated for each class to make the decision on the one-class problem (if x belongs or not to the class), and, finally, the original multi-class problem is recomposed aggregating the one-class decisions.

Sections 3.2.1 and 3.2.2 describe in more detail the training and test phases of DBDES, respectively.

3.2.1 Training phase

The training phase of DBDES separates the training data by class and segments the data from each class using OPTICS. Then, an OCC is trained for each cluster and added to a pool.

Algorithm 3: DBDES training phase

input : Γ: the training set
output : M: an array containing maps of pools of OCCs

```
1  M = ∅
2  L = labels(Γ)                          // extract the set of labels
3  for l ∈ L do
4      Γ_l = instances(Γ, l)              // extract the instances of class l
5      g = optics(Γ_l)                    // segment Γ_l into g = {c_1, c_2, ..., c_j, ...}
6      p = ∅
7      for c_j ∈ g do
8          λ_j = trainOCC(c_j)
9          p = p ∪ λ_j
10     M_l = < g, p >
11 return M
```

Algorithm 3 describes the details of the training phase, which follows these steps:

1. Separate the training instances by class (lines 2 to 4): the training instances are separated by class, resulting in $|L|$ training sets: $\{\Gamma_1, ..., \Gamma_l, ..., \Gamma_{|L|}\}$, where L is the set of classes in Γ.

2. Cluster the data in Γ_l (line 5): for each class $l \in L$, Γ_l is segmented using OPTICS resulting in the partition $g = \{c_1, c_2, ..., c_j, ...\}$.

3. Generate the pool of OCCs (lines 6 to 9): for each cluster $c_j \in g$, an OCC λ_j is trained and added to the pool p, and the pair $< g, p >$ is stored in M_l.

4. Repeat items 2 and 3 for each class $l \in L$.

The output of the training phase is the array M, which contains $|L|$ pairs $M_l =< g, p >$, one for each class label l of the original multi-class problem. Each pair $< g, p >$ is composed of a partition, which is Γ_l segmented into clusters $\{c_1, ..., c_j, ...\}$, and p is a pool composed of the OCCs $\{\lambda_1, ..., \lambda_j, ...\}$. M_l binds each OCC λ_j to the cluster c_j used to train it.

The main difference to MODES in this phase is that, here, each map of pools M_l stores only one pair $< g, p >$ while, in MODES, each M_l stores several pairs $< g_i, p_i >$, one for each number of clusters D_i identified by the cluster validity indices. Additionally, as OPTICS allows non-exclusive partitions, a training instance may be a member of more than one cluster in each partition g.

3.2.2 Test phase

In the test phase, DBDES discards OCCs that were not trained with data from classes in the neighborhood of the test instance x, as in (KRAWCZYK et al., 2018) and (FRAGOSO et al., 2021). An ensemble of OCCs E_l is dynamically selected to classify x, for each class label l in the region of competence Ψ with respect to x. The classification of the l-th one-class problem, i.e., if x belongs or not to class l, is given by the aggregation of the classifiers in the ensemble E_l. The final classification (the final decision that assigns a label to x) is given by the aggregation of the decisions made by the ensembles $\{E_1, ..., E_l, ..., E_{|\Psi|}\}$ selected for each class l in the region of competence Ψ. The region of competence is computed using a validation set D_{SEL}.

Algorithm 4: DBDES test phase

```
input  : x: a test instance
         M: an array containing maps of pools of OCCs
         D_SEL: a validation set
output : ω: the predicted label for x
1  nn = kNN(x, D_SEL)
2  Ψ = labels(nn)                    // extract the set of labels of the instances in nn
3  R = ∅
4  for M_l ∈ M do
5      // M_l stores the pair < g, p >
6      if l ∈ Ψ then
7          E_l = ∅
8          for c_j ∈ g do
9              if x belongs to c_j then
10                 E_l = E_l ∪ λ_j
11         R = R ∪ max(E_l, x)       // MAX aggregation
12 ω = aggregate(R, x)               // DTs, MAX or ECOC
13 return ω
```

Algorithm 4 describes the test phase of DBDES, which follows these steps to classify each

test instance x:

1. Compute the region of competence Ψ (lines 1 and 2): the region of competence is defined in the same way as in MODES, i.e., as the classes of the k nearest neighbors of x in a validation set D_{SEL}, with $k = 3*|L|$. Also, the same threshold is applied to ignore classes representing less than 10% of the instances in the neighborhood (KRAWCZYK et al., 2018).

2. Select the maps of pools of OCCs M_l (lines 4 to 6): for each class l in the region of competence Ψ, load M_l, which contains the pair $< g, p >$.

3. Select the ensemble E_l (lines 8 to 10): assess the membership of x to each cluster in g. Then, select in p the OCCs trained with data from clusters to which x belongs and add them to the ensemble E_l. As OPTICS allows non-exclusive partitions, x may be a member of more than one cluster $c_j \in g$. In preliminary experiments we carried out, this strategy yielded better results than selecting only one cluster per partition g.

4. Aggregate E_l (line 11): aggregate the predictions for the one-class problem (if x belongs or not to the class l) using MAX. This aggregation technique was chosen due to its simplicity and because it outperformed the mean of probabilities in the preliminary experiments we carried out.

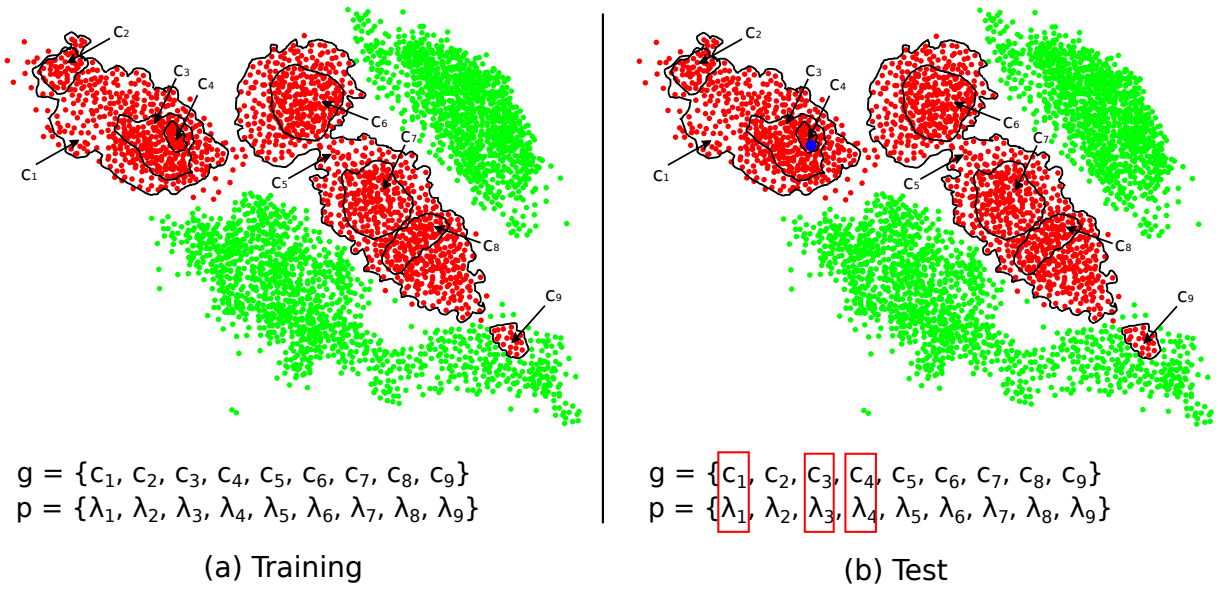
5. Aggregate the decisions in R (line 12): to recompose the original multi-class problem from the one-class problems, use an aggregation technique (DTs, ECOC, or MAX).

3.2.3 Toy example

This section presents a visual description of DBDES intending to facilitate the understanding of the proposed technique. Figure 8 presents an example of the training (a) and test (b) phases of DBDES.

In the training phase, DBDES separates the training data by class and the data from each class l is segmented using the OPTICS algorithm. In the example depicted in Figure 8, OPTICS identifies nine clusters in the data, forming the partition $g = \{g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9\}$. Then, an OCC is trained for each cluster and added to the pool $p = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7, \lambda_8, \lambda_9\}$. Note that OPTICS identifies hierarchical (nested) clusters in the data. This is the fact that provides the distinct data inputs for training the pool of OCCs.

Figure 8 – Toy example of DBDES for a class label l. The figure on the left (a), presents the training phase and the figure on the right presents the test phase.



Source: The author.

In the test phase, DBDES evaluates the membership of a test instance x to each cluster in the partition g. Then, it selects the OCCs trained with data from the clusters of which x is a member. As OPTICS identifies hierarchical non-exclusive clusters, an instance may be a member of more than one cluster. Supposing that the test instance x is represented by the blue dot in the rightmost figure in Figure 8, DBDES selects the OCCs λ_1, λ_3, and λ_4 to the ensemble because they were trained with data from clusters c_1, c_3, and c_4 (the clusters which x is a member of).

This process is repeated for each class label l and, in the end, the outputs of the ensembles created for each class label l are aggregated to give the final result of the classification.

3.3 HYPER-PARAMETERS

MODES and DBDES have three hyper-parameters each. In previous experiments, we encountered efficient values that have lead to enhanced classification performance. Table 2 details these values.

It is important to note that MODES presents an increased computational complexity which may lead to increased memory and processing requirements. However, since the focus of applications for MODES does not include large databases nor data streams, we focused our efforts on enhancing the classification accuracy rather than computational requirements. With

Table 2 – Default values for MODES and DBDES hyper-parameters.

Hyper-parameter	MODES	DBDES
Number of clusters	2 to 10	N/A
minPts	N/A	20
Ψ (region of competence)	3 x $\|L\|$, where L is the set of classes	3 x $\|L\|$, where L is the set of classes
Threshold for Ψ	10%	10%

Source: The author.

DBDES, we aim at reducing the complexity of MODES while maintaining or improving the classification accuracy. In Section 4.2.3 we include a brief discussion about the runtime of the techniques.

4 EXPERIMENTS

In this chapter, we evaluate the performance of the proposed techniques. Three experiments are carried out. The first one evaluates the performance of One-class Dynamic Ensemble Selection for Multi-class problems (MODES) with different configurations, that is, One-Class Classifier (OCC)s models and aggregation techniques. The goal of this experiment is to identify the best configuration of MODES. The same is made with Density-Based Dynamic Ensemble Selection (DBDES) in the second experiment.

Finally, the third experiment compares the best configurations of MODES and DBDES against two different approaches to one-class decomposition: (1) Dynamic Ensemble Selection with THReshold-based neighborhood pruning (DES_{THR}) (KRAWCZYK et al., 2018), a dynamic ensemble selection method with neighborhood pruning based on threshold; (2) the aggregation, without selection, of a pool of one-class classifiers composed of one OCC for each class (KRAWCZYK; WOŹNIAK; HERRERA, 2015). In (2), the training phase consists in dividing the data by class and training an OCC for each class. In the test phase, a test instance is submitted to all trained OCCs and an aggregation technique is used to give the final classification based on the outputs of each OCC. The technique proposed in (2) is further referred to in this work as static combination of OCCs.

In addition, we included in Appendix B the results of a comparison carried out among the proposed techniques and a benchmark multi-class classifier (Random Forest Classifier). Although the proposed techniques do not intend - by no means - to be more efficient than regular multi-class classifiers in all specter of problems, a comparison with this kind of classifier is relevant to evaluate which in cases MODES and DBDES are worth using.

The experimental protocol is described in Section 4.1 and the results are discussed in Section 4.2.

4.1 EXPERIMENTAL PROTOCOL

This section describes the configurations used in the experiments carried out in this paper. Twenty-five databases from Keel datasets repository[1] were used. Most of these databases were used in (KRAWCZYK et al., 2018), which also aims to evaluate the decomposition of the multi-class problem into one-class problems. The databases present a comprehensive set of

[1] Available in: http://www.keel.es/dataset

characteristics (number of instances, dimensionality, number of classes, imbalance ratio). Some of the selected databases are modified versions of other databases. For example, Glass1 and Glass6 are binary versions of the database Glass, where the positive instances belong to classes 1 and 6, respectively, and the negative instances belong to the other classes. Furthermore, the databases Movement Libras, Optidigits, and Texture were modified. Some classes were joined so that the modified databases have 7, 4, and 5 classes respectively. The nearest classes were joined together to simulate classes with multi-modal distributions. In addition, these modifications aimed at increasing the class imbalance. The selected databases present a variable number of features (from 6 to 90), classes (from 2 to 26), and instances (from 148 to 12,960). Furthermore, there are balanced and imbalanced databases. Table 3 describes the characteristics of each database.

As the proposed methods rely on the distance among the training instances (for clustering), in a pre-processing step, the data was scaled using z-score (JAIN; NANDAKUMAR; ROSS, 2005). Non-numerical attributes were transformed using the simple label encoder, where each value is bound to an integer value (for example, a is transformed to 1, b is transformed to 2, and so on).

Four one-class classifiers were adopted in this work (TAX, 2005): Gaussian Data Descriptor (GaussianDD), Parzen Data Descriptor (ParzenDD), Support Vector Data Descriptor (SVDD), and Minimum Spanning Tree Data Descriptor (MSTDD). SVDD, ParzenDD and MSTDD are used in (KRAWCZYK et al., 2018), which is the one of the most important one-class decomposition techniques. MODES uses centroid-based clustering to generate the chunks of data used for training. Thus, we expect that GaussianDD performs well with MODES. For this reason, we also evaluate GaussianDD one-class classifier. We used the implementations of the OCCs available in the Matlab library *dd_tools* (TAX, 2018) and the hyper-parameters used in the experiments are detailed in Table 4. It is important to remark that we are aware that a thorough tuning of the hyper-parameters for the OCCs should enhance the classification performance. However, the objective of the experiments is to compare different one-class decomposition techniques rather than to evaluate the performances of particular OCC models.

The experiments were performed using 5-fold cross-validation and the final performance is given by the average of the 5 folds. The performance was evaluated using accuracy (KUNCHEVA, 2014) and Kappa statistic (COHEN, 1960; KRAWCZYK et al., 2018), both metrics used in (KRAWCZYK et al., 2018). Kappa statistics, or Cohen's Kappa Coefficient, was adopted as a complementary metric to accuracy because it provides a different evaluation for the results, mainly for minority

Table 3 – Databases description. Imbalance Ratio is computed as the division of the cardinality of the largest class by the cardinality of the smallest class.

Name	Instances	Features	Numeric	Nominal	Classes	Imbalance Ratio
Automobile	159	25	15	10	6	16.00
Car	1,728	6	0	6	4	18.62
Cleveland	297	13	13	0	5	12.31
Dermatology	358	34	34	0	6	5.55
Ecoli	336	7	7	0	8	71.5
Flare	1,066	11	9	2	6	7.70
Glass	214	9	9	0	6	8.44
Glass1	214	9	9	0	2	1.82
Glass6	214	9	9	0	2	6.38
Led7digit	500	7	7	0	10	1.54
Letter	2,000	16	16	0	26	1.11
Lymphography	148	18	3	15	4	40.50
Movement Libras	360	90	90	0	7	1.50
Nursery	12,960	8	0	8	5	2,160
Optdigits	5,620	64	64	0	4	1.53
Page-blocks	5,472	10	10	0	5	175.5
Penbased	10,992	16	16	0	10	1.08
Satimage	6,435	36	36	0	6	2.45
Segment	2,310	19	19	0	7	1.00
Shuttle	5,780	9	9	0	7	4,558
Texture	5,500	40	40	0	5	5.00
Vehicle	846	18	18	0	4	1.10
Vehicle2	846	18	18	0	2	2.88
Vowel	990	13	13	0	11	1.00
Yeast	1,484	8	8	0	10	92.6

Source: The author.

classes. This metric is less sensitive to randomness caused by a different number of instances in each class because it scores the successes independently for each class and aggregates them in the end (GALAR et al., 2011). Kappa statistic ranges from -1 (total disagreement) through 0 (random classification) to 1 (perfect agreement).

Equations 4.1 and 4.2 describe the computation of such metrics.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{4.1}$$

where TP, TN, FP, and FN mean true positive, true negative, false positive, and false negative,

Table 4 – OCCs hyper-parameters. Matlab dd_tools library implementation was used in the experiments.

GaussianDD	Regularization parameter = 0.001 Fraction rejected = 0.05
ParzenDD	Kernel type = normal Width parameter optimization = Max. likelihood Fraction rejected = 0.05
SVDD	Kernel type = RBF $C = 5.0$ $\gamma = 0.0045$ Fraction rejected = 0.05
MSTDD	Maximum Support (MAX). path = none Fraction rejected = 0.05

Source: The author.

respectively.

$$\text{Kappa} = \frac{p_o - p_e}{1 - p_e}, \tag{4.2}$$

where p_o is the empirical probability of agreement on the label assigned to any sample (the observed agreement ratio between the predicted class and the actual class), and p_e is the hypothetical probability of chance agreement.

MODES uses center-based clustering, which requires a value for the number of clusters (k). The determination of the values for k was performed using the 13 fastest indexes present in the *R* package *NbClust* (CHARRAD et al., 2014): Calinski and Harabasz (CALIŃSKI; HARABASZ, 1974), PtBiserial (MILLIGAN, 1981), Hartigan (HARTIGAN, 1975), Ball (BALL; HALL, 1965), Mcclain (MCCLAIN; RAO, 1975), KL (KRZANOWSKI; LAI, 1988), Silhouette (KAUFMAN; ROUSSEEUW, 2009), Gap (TIBSHIRANI; WALTHER; HASTIE, 2001), Dunn (DUNN, 1974), SDindex (HALKIDI; VAZIRGIANNIS; BATISTAKIS, 2000), SDbw (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001), C-index (HUBERT; LEVIN, 1976), Davies and Bouldin (DAVIES; BOULDIN, 1979). The evaluation of the numbers of clusters was performed from 2 to 10 clusters. Since the average number of clusters returned by the cluster validity indexes was 5.5 (with standard deviation of 3), we did not evaluate other ranges. The clustering algorithm adopted in the experiments was k-Means due to its simplicity and good results (JAIN, 2010). We used the default hyper-parameters values given by *Scikit Learn*[2] implementation, except for the number of clusters,

[2] <https://scikit-learn.org/>

which was chosen by clustering validity indices, as explained in Section 3.1.

The clustering phase of DBDES was executed using Ordering Points To Identify the Clustering Structure (OPTICS) implementation given by *Scikit Learn*. OPTICS requires a value for the hyper-parameter $minPts$. In the experiments, we use $minPts = 20$, since this value is in the range recommended by the authors of OPTICS (ANKERST et al., 1999). For ξ and ε, we use the default values of the Scikit-learn library in the experiments (0.05 and infinite).

Both MODES and DBDES aggregate the prediction in two levels. The first is performed in the one-class problem. The output of the one-class classification, i.e., whether the test instance resembles the target class or not, is given by the aggregation of the selected classifiers. For this level of aggregation, MODES employs the mean of probabilities (ZHOU, 2012) while DBDES used MAX. The second level is in the re-composition of the multi-class problem, where the predictions of the one-class problems are aggregated to give the final prediction, i.e., which class the test instance belongs to. In the second level, we adopt Decision Templates (DTs) (KUNCHEVA; BEZDEK; DUIN, 2001), Error Correcting Output Codes (ECOC)) (PUJOL; RADEVA; VITRIA, 2006) or MAX (KUNCHEVA, 2014) for all the techniques. These techniques were adopted in literature (KRAWCZYK; WOŹNIAK; HERRERA, 2015; KRAWCZYK et al., 2018).

4.2 EXPERIMENTAL RESULTS

In this section, we present and discuss the results obtained in the experiments. Sections 4.2.1 and 4.2.2 present the experiments carried out to evaluate the different configurations of MODES and DBDES, respectively. In Section 4.2.3, we compare the performances of the best configurations of MODES and DBDES with two techniques for one-class decomposition: DES_{THR} and static combination of OCCs.

4.2.1 Experiment 1

This experiment aims to identify the most effective configuration for MODES. Thus, we evaluate the proposed technique using 4 OCC models and 3 aggregation techniques, making up 12 different configurations for MODES.

Tables 5 and 6 present the accuracy and Kappa statistics scores, respectively, for all configurations of MODES. The best result for each database is in bold and the last rows present the mean performance for all databases, the number of wins, ties, and losses, and the average

rankings. The average rankings are computed using all the 12 configurations.

MODES achieved its best performance using MSTDD and Decision Templates, for both accuracy and Kappa Statistic. This configuration presented the best average accuracy and Kappa statistics scores and the higher number of wins, i.e., the number of databases in which it achieved the best performance among all configurations, for both metrics. The best results were achieved with DTs and the best performing one-class classifiers were MSTDD and GaussianDD.

Table 5 – Accuracy performance (in %) of MODES using DTs, ECOC, and MAX for four one-class classifiers. The best result for each database is in bold. The last row represents the number of wins, ties, and losses achieved by each technique.

	Decision Templates				Error Correcting Output Codes				MAX			
Dataset	GaussianDD	ParzenDD	SVDD	MSTDD	GaussianDD	ParzenDD	SVDD	MSTDD	GaussianDD	ParzenDD	SVDD	MSTDD
Automobile	66.44	57.12	72.46	**78.69**	66.56	62.10	60.79	63.11	65.04	72.95	69.19	71.09
Car	89.12	**90.80**	90.68	84.84	90.05	77.49	85.48	82.00	83.85	89.70	81.71	84.32
Cleveland	58.24	47.43	57.24	53.88	46.46	37.41	48.77	54.18	57.18	55.86	**60.28**	53.85
Dermatology	93.02	84.89	92.72	92.74	**93.28**	91.05	92.75	93.03	91.56	92.20	92.41	91.89
Ecoli	75.90	76.80	78.28	77.69	**78.61**	77.69	77.71	72.93	75.00	76.50	75.30	75.91
Flare	68.76	69.51	66.60	69.61	62.94	68.39	63.41	64.63	64.07	63.60	**70.08**	69.89
Glass	62.18	64.95	59.41	**71.54**	57.50	63.11	61.24	56.57	57.48	65.42	64.98	68.69
Glass1	71.01	75.70	69.15	81.33	62.64	71.54	65.91	70.09	75.27	78.98	75.24	**81.34**
Glass6	94.85	94.85	**96.26**	95.32	93.44	93.44	93.91	95.78	**96.26**	95.78	95.79	95.32
Led7digit	70.80	71.60	71.20	**72.20**	68.00	67.60	69.20	55.00	70.00	71.00	72.00	60.80
Letter	92.48	83.81	68.17	90.52	65.50	44.90	51.55	70.41	**95.34**	94.22	84.65	93.88
Lymphography	74.37	66.25	76.97	74.32	68.94	71.68	77.72	71.68	66.90	68.97	**79.03**	70.92
Movement Libras	77.22	52.78	77.78	**86.67**	79.17	67.22	71.94	70.56	78.06	83.33	76.94	85.56
Nursery	92.23	80.05	83.56	74.58	83.70	54.55	62.82	63.73	**92.56**	79.61	89.00	72.69
Optdigits	98.02	95.23	97.38	98.02	96.69	90.11	96.30	96.80	97.85	**98.11**	97.33	98.01
Page-blocks	**95.18**	94.76	94.32	95.01	93.93	91.25	93.21	92.68	95.14	94.76	94.92	94.96
Penbased	99.36	97.19	97.40	99.27	96.53	81.96	94.13	97.42	**99.42**	99.19	98.45	99.02
Satimage	87.57	86.03	86.62	**90.35**	82.50	82.50	81.29	83.26	88.02	88.56	87.80	88.39
Segment	93.77	91.34	90.82	**95.37**	91.95	94.42	90.43	94.07	94.42	94.72	91.99	95.24
Shuttle	99.23	99.36	99.17	**99.42**	99.33	99.21	99.17	99.33	99.24	99.31	99.24	99.42
Texture	**99.80**	93.35	94.76	98.53	98.60	95.84	94.53	96.49	99.78	98.67	96.80	98.95
Vehicle	**78.72**	62.06	70.44	70.45	77.07	63.24	63.24	63.71	78.60	68.80	69.26	68.44
Vehicle2	**99.17**	94.21	94.45	97.64	94.33	86.88	92.91	93.50	97.05	96.93	94.44	97.64
Vowel	97.68	80.30	90.10	**98.48**	89.39	47.17	66.77	89.60	97.58	97.07	88.99	97.88
Yeast	53.98	56.06	52.16	54.78	44.81	54.92	44.20	44.40	49.93	**56.40**	47.30	51.01
Mean	83.56	78.66	81.12	**84.05**	79.28	73.43	75.98	77.40	82.62	83.23	82.12	82.60
Win/tie/loss	4/0/21	1/0/24	0/1/24	**8/0/17**	2/0/23	0/0/25	0/0/25	0/0/25	3/1/21	2/0/23	3/0/22	1/0/24
Avg ranks	4.74	7.73	6.43	**3.94**	7.44	9.12	9.07	8.51	5.24	4.82	5.94	5.02

Source: The author.

Friedman test was employed to determine if the different configurations present significantly different performance. The test output $p - value = 1.11^{-16}$ for both accuracy and Kappa Statistics, meaning that the difference in performance among the configurations is significant. Then, the Nemenyi *post hoc* test was used to identify which of the configurations differ from each other. Figure 9 shows the results of Nemenyi *post hoc test*.

Again, MSTDD with DTs was identified as the best performing configuration, for both accuracy and Kappa Statistics. It is also interesting to note that the seven best performing configurations (all base OCCs (except for ParzenDD) with DTs and MAX) present quite

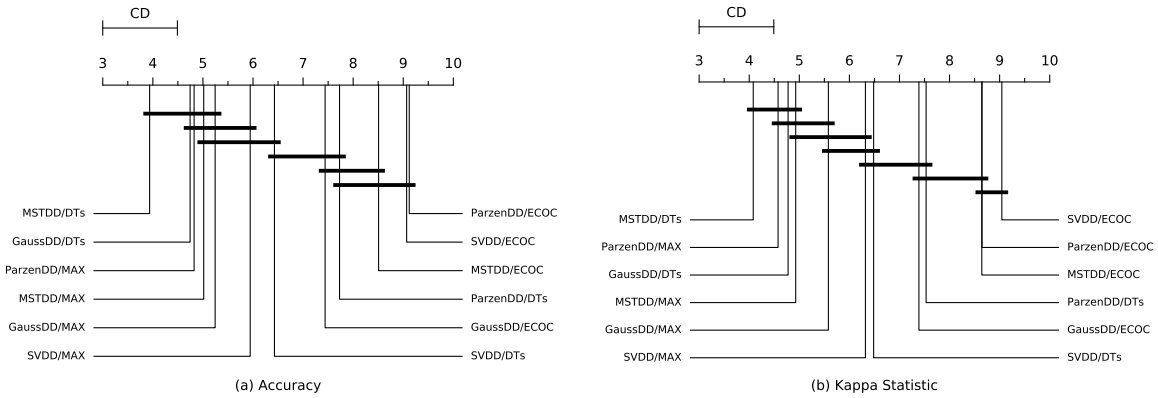
Table 6 – Kappa performance of MODES using DTs, ECOC, and MAX for four one-class classifiers. The best result for each database is in bold. The last row represents the number of wins, ties, and losses achieved by each technique.

	Decision Templates				Error Correcting Output Codes				MAX			
Dataset	GaussianDD	ParzenDD	SVDD	MSTDD	GaussianDD	ParzenDD	SVDD	MSTDD	GaussianDD	ParzenDD	SVDD	MSTDD
Automobile	0.5788	0.5477	0.6346	**0.7199**	0.5453	0.5012	0.4729	0.5283	0.5290	0.6493	0.5899	0.6176
Car	0.7822	**0.8029**	0.8028	0.6808	0.7753	0.4328	0.6855	0.6355	0.5905	0.7746	0.5350	0.6526
Cleveland	0.2735	0.2537	0.3054	0.2996	0.2358	0.1781	0.2208	0.1688	0.1955	**0.3387**	0.2918	0.2984
Dermatology	0.9125	0.8110	0.9082	0.9086	**0.9164**	0.8887	0.9098	0.9125	0.8934	0.9017	0.9040	0.8977
Ecoli	0.6667	0.6815	0.6979	0.6903	0.6943	**0.6980**	0.6782	0.6027	0.6419	0.6697	0.6488	0.6615
Flare	0.6039	0.6144	0.5777	0.6146	0.5350	0.5972	0.5362	0.5531	0.5419	0.5377	**0.6178**	0.6156
Glass	0.4855	0.5314	0.4632	**0.6128**	0.4095	0.4878	0.4626	0.3970	0.3763	0.5250	0.4996	0.5654
Glass1	0.3649	0.4840	0.3626	**0.5810**	0.1601	0.4553	0.2114	0.2395	0.3799	0.5284	0.4473	0.5626
Glass6	0.7705	0.7755	**0.8226**	0.7855	0.7281	0.7281	0.7437	0.8092	0.8134	0.8092	0.7955	0.7855
Led7digit	0.6752	0.6842	0.6797	**0.6909**	0.6447	0.6403	0.6578	0.4996	0.6666	0.6774	0.6884	0.5642
Letter	0.9218	0.8316	0.6689	0.9014	0.6412	0.4267	0.4962	0.6922	**0.9516**	0.9399	0.8404	0.9363
Lymphography	0.5038	0.3845	0.5509	0.4977	0.3910	0.4681	0.5620	0.4483	0.3784	0.4059	**0.5885**	0.4245
Movement Libras	0.7331	0.4518	0.7388	**0.8434**	0.7542	0.6153	0.6690	0.6559	0.7402	0.8043	0.7280	0.8302
Nursery	0.8875	0.7087	0.7616	0.6315	0.7593	0.3380	0.4487	0.4686	**0.8902**	0.7018	0.8373	0.6031
Optdigits	0.9733	0.9357	0.9646	0.9733	0.9552	0.8671	0.9500	0.9567	0.9709	**0.9745**	0.9639	0.9731
Page-blocks	**0.7700**	0.7550	0.6952	0.7564	0.6860	0.6137	0.6337	0.5711	0.7665	0.7585	0.7131	0.7615
Penbased	0.9929	0.9688	0.9711	0.9919	0.9615	0.7994	0.9348	0.9714	**0.9935**	0.9910	0.9828	0.9891
Satimage	0.8464	0.8253	0.8357	**0.8812**	0.7824	0.7856	0.7670	0.7946	0.8511	0.8589	0.8491	0.8570
Segment	0.9273	0.8990	0.8929	**0.9460**	0.9061	0.9348	0.8884	0.9308	0.9348	0.9384	0.9066	0.9444
Shuttle	0.9785	0.9823	0.9771	**0.9837**	0.9813	0.9780	0.9769	0.9813	0.9789	0.9807	0.9788	**0.9837**
Texture	**0.9971**	0.9048	0.9253	0.9789	0.9798	0.9401	0.9213	0.9496	0.9969	0.9809	0.9538	0.9848
Vehicle	**0.7162**	0.4935	0.6066	0.6060	0.6943	0.5103	0.5090	0.5151	0.7145	0.5836	0.5901	0.5789
Vehicle2	**0.9781**	0.8365	0.8593	0.9372	0.8521	0.7071	0.8034	0.8194	0.9197	0.9206	0.8467	0.9370
Vowel	0.9744	0.7833	0.8911	**0.9833**	0.8833	0.4189	0.6344	0.8856	0.9733	0.9678	0.8789	0.9767
Yeast	0.4064	**0.4360**	0.3818	0.4221	0.2471	0.4082	0.2379	0.2396	0.3435	0.4337	0.3065	0.3668
Mean	0.7488	0.6953	0.7190	**0.7552**	0.6848	0.6168	0.6405	0.6491	0.7213	0.7461	0.7193	0.7347
Win/tie/loss	4/0/21	2/0/23	1/0/24	**8/1/16**	1/0/24	1/0/24	0/0/25	0/0/25	3/0/22	2/0/23	2/0/23	0/1/24
Avg ranks	4.78	7.53	6.48	**4.08**	7.39	8.65	9.04	8.64	5.58	4.58	6.32	4.93

Source: The author.

similar performance (mean accuracy between 81.12 and 84.05). The performance of the other 5 configurations (all base OCCs with ECOC and ParzenDD with DTs) is at a level below (mean accuracy between 73.43 and 79.28). This can be interpreted as evidence that MODES is robust to the choice of the base one-class classifier, since, except for ParzenDD with DTs, the base OCCs performed similarly when using the same aggregation technique.

Figure 9 – Result for Nemenyi post hoc test for (a) accuracy and (b) Kappa Statistic.



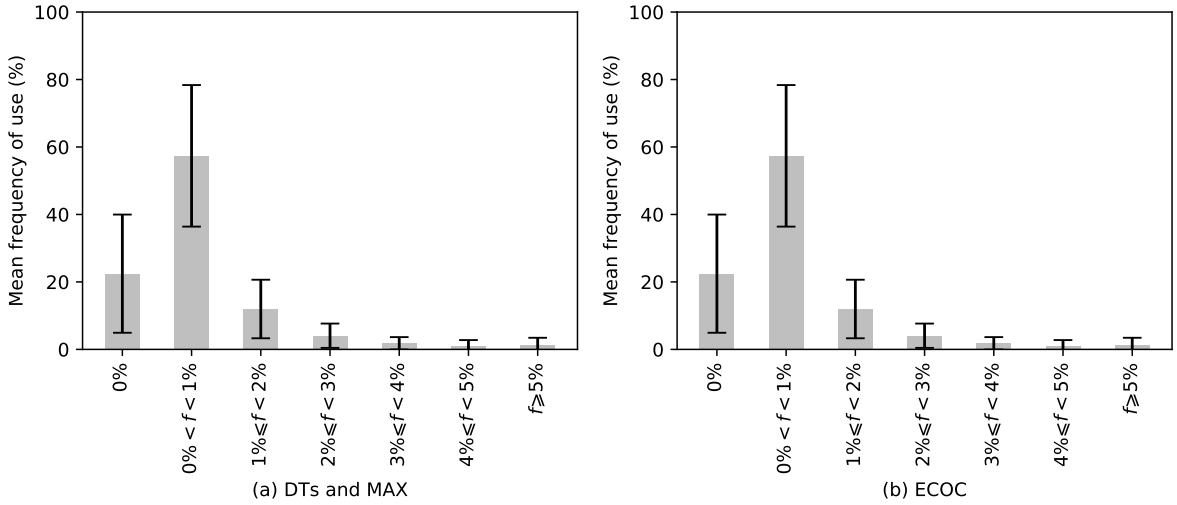
Source: The author.

A quantitative analysis of the generation and selection phases of MODES was carried out. In the training phase, MODES segments the data of each class label l using a center-based clustering algorithm, such as K-means, with a set D of numbers of clusters $\{d_1, d_2, ...\}$ given by the cluster validity indices. For each $d_i \in D$, MODES trains a pool of OCCs, one OCC for each cluster. The average number of OCCs generated by class ranges from 13.60 to 33.74 (average 25.55) using DTs or MAX and from 16.17 to 35.10 (average 28.48) using ECOC.

In the selection phase, MODES selects one OCC for each number $d_i \in D$ to compose the ensemble. Thus, the cardinality of D represents the number of OCCs selected. The distribution for the frequency of selection of the OCCs is presented in Figure 10: never selected, selected for up to 1%, 2%, 3%, 4%, 5% of the test instances and selected for more than 5% of the test instances. The frequencies are equal for DTs and MAX since these aggregation strategies use the original classes. ECOC creates meta-binary problems which use meta-classes, thus, the frequencies for ECOC are shown separately (right side of Figure 10).

For the majority of the databases, the highest frequency of selection is between 0% and 1%, that is, most of the OCCs are selected to classify up to 1% of the test instances. It is worth noting that the majority of the OCCs are selected at least once. Using ECOC, less than 10% of the OCCs are never used while for DTs and MAX, less than 20% of the OCCs are never used. Both figures indicate that the strategy for pool generation (i.e., segment the training data and train an OCC for each cluster) is effective, since few OCCs are never selected and most of the OCCs are used for classifying few instances, which means that they are used for specific regions of the feature space.

Figure 10 – Frequency of use of OCCs with MODES for DTs and MAX (a) and ECOC (b).



Source: The author.

4.2.2 Experiment 2

The goal of this experiment, like the previous one, was to evaluate the different configurations of DBDES, i.e., combinations of OCC models and aggregation techniques, to identify the most effective one. The evaluation of the configurations took into account the accuracy and Kappa statistic performance. The result of each configuration is compared to all the other configurations. Tables 7 and 8 present the results in terms of accuracy and Kappa statistic, respectively. The best result for each database is in bold and the last rows present the mean performance for all databases, the number of wins, ties, and losses and the average rankings computed using all the 12 configurations.

Table 7 – Accuracy performance (in %) of DBDES using DTs, ECOC, and MAX aggregation methods for four one-class classifiers. The best result for each database is in bold. The last row represents the number of wins, ties, and losses achieved by each technique.

	Decision Templates				Error Correcting Output Codes				Maximum Support			
Database	GaussianDD	ParzenDD	SVDD	MSTDD	GaussianDD	ParzenDD	SVDD	MSTDD	GaussianDD	ParzenDD	SVDD	MSTDD
Automobile	67.80	64.13	55.24	74.88	68.74	67.33	51.82	60.55	58.54	73.53	62.75	**79.27**
Car	82.58	81.89	77.14	82.29	78.88	77.72	77.55	81.94	77.89	**89.99**	78.07	84.38
Cleveland	56.55	42.07	54.49	52.18	53.56	37.06	54.52	55.20	**58.90**	47.76	58.48	50.15
Dermatology	**96.36**	81.85	89.39	89.66	96.13	91.63	93.59	95.26	96.05	93.86	95.81	93.31
Ecoli	75.90	71.41	75.60	75.58	74.71	**81.26**	71.73	76.51	74.12	78.88	73.80	76.49
Flare	**69.79**	66.04	68.38	67.63	62.95	65.95	63.23	64.26	65.85	65.20	68.29	68.39
Glass	54.21	55.64	46.73	**65.45**	51.40	59.81	50.96	50.94	59.36	57.01	58.43	58.89
Glass1	71.98	70.07	65.95	75.26	69.16	63.59	68.72	70.56	73.39	72.46	71.53	**80.40**
Glass6	95.33	94.85	95.33	94.85	94.39	93.44	93.91	94.85	**96.26**	95.32	**96.26**	94.85
Led7digit	69.60	72.40	70.20	72.60	69.80	65.40	72.60	51.60	68.80	**73.00**	**73.00**	63.20
Letter	75.19	63.10	60.15	84.03	63.69	64.96	65.56	93.96	89.88	93.99	75.79	**94.17**
Lymphography	76.34	69.59	73.68	69.61	73.68	69.61	**78.41**	70.97	71.01	72.30	77.72	69.61
Movement Libras	75.00	50.28	58.06	85.28	65.28	56.11	72.22	70.83	74.72	**86.11**	78.06	83.33
Nursery	75.56	73.23	72.84	76.60	83.33	75.34	73.59	78.81	**86.57**	79.76	79.97	74.05
Optdigits	97.21	95.21	96.85	97.60	96.01	86.51	95.82	96.80	96.92	**98.15**	97.08	97.37
Page-blocks	95.03	94.28	94.61	94.94	94.70	94.74	94.12	**95.43**	94.24	93.95	95.07	94.34
Penbased	98.19	95.81	95.96	99.03	97.15	95.75	96.95	99.10	98.73	**99.25**	97.46	99.04
Satimage	84.32	78.29	81.24	89.04	86.46	87.79	87.26	**89.43**	87.58	89.34	88.30	88.61
Segment	92.51	90.00	87.01	**95.06**	88.70	93.59	88.66	93.29	91.65	92.68	89.00	93.42
Shuttle	99.14	99.33	99.28	99.38	99.19	99.28	99.17	99.31	99.26	99.24	99.24	**99.42**
Texture	99.22	92.64	92.84	97.58	97.25	94.62	94.02	96.47	**99.71**	98.38	95.33	97.84
Vehicle	81.21	61.70	66.08	70.22	72.10	64.54	61.35	63.36	**82.04**	71.99	66.67	70.10
Vehicle2	98.11	93.02	92.20	95.04	94.56	86.88	91.37	93.50	**98.47**	96.57	93.50	95.39
Vowel	91.11	58.69	54.44	94.95	53.84	50.20	46.87	95.96	91.01	97.58	65.76	**98.38**
Yeast	54.25	56.27	50.94	57.68	48.65	55.93	45.75	50.13	56.47	**58.42**	49.06	52.83
Mean	81.30	74.87	74.99	82.26	77.37	75.16	75.59	79.56	81.90	**82.99**	79.38	82.29
Win/tie/loss	2/0/23	0/0/25	0/0/25	2/0/23	0/0/25	1/0/24	1/0/24	2/0/23	**5/1/19**	**5/1/19**	0/2/23	5/0/20
Avg. ranks	5.31	8.55	8.81	4.49	7.50	8.08	8.70	6.10	5.24	**4.33**	6.02	4.86

Source: The author.

The best mean accuracy performance was achieved by ParzenDD/MAX (82.96). This configuration also presented the best average ranking (4.33). However, GaussianDD/MAX achieved the highest number of wins, six versus five wins achieved by ParzenDD/MAX and MSTDD/MAX. Nonetheless, the mean accuracy performance of GaussianDD/MAX and its average ranking were only the fourth best, behind ParzenDD/MAX, MSTDD/MAX, and

MSTDD/DTs. Hence, we consider that the best accuracy performance was achieved by ParzenDD/MAX.

DTs achieved good results with GaussianDD/DTs and MSTDD/DTs. However, when adopting ParzenDD/DTs and SVDD/DTs, DBDES presented the worst mean accuracy performances among all configurations. These configurations did not achieve any win and their average rankings were among the three worst (with SVDD/ECOC. With ECOC, DBDES produced mean accuracy below 80.00 for all OCCs and obtained four wins: two with MSTDD/ECOC and one with ParzenDD/ECOC and SVDD/ECOC. MAX obtained the best results, with all OCCs accuracy performances above 80.00 (except for SVDD/MAX, which obtained 79.38) and achieving 16 wins (out of 25). SVDD/MAX, however, did not achieve any win.

Looking at the performances of the OCCs model, we notice that MSTDD presented good results for all aggregation techniques. With DTs and ECOC, MSTDD was the OCC that produced the best results. With MAX, MSTDD achieved the second best results, behind ParzenDD. SVDD obtained the worst average rankings for all aggregation techniques. Additionally, it presented the worst mean accuracy for MAX and the second worst mean accuracy for DTs and ECOC. Although ParzenDD/MAX presented the best performance among all configurations, this OCC model achieved bad results with DTs and ECOC. For these aggregation techniques, ParzenDD obtained the worst mean accuracy and the second worst average rankings. GaussianDD presented regular results for all aggregation techniques and achieved eight wins, where GaussianDD/DTs obtained two wins and GaussianDD/MAX obtained six wins.

It is worth noting that the seven best performing configurations (all base OCCs with MAX, GaussianDD/DTs, MSTDD/DTs, and MSTDD/ECOC) present quite similar performance (mean accuracy between 79.38 and 82.99). The performances of the other five configurations are at a level below (mean accuracy between 74.87 and 77.37). This behavior is similar to that observed in MODES and can be interpreted as evidence that DBDES is also robust to the choice of the base one-class classifier, since, except for ParzenDD and SVDD with DTs, the base OCCs performed similarly when using the same aggregation technique.

The behavior of the methods with respect to Kappa statistic performance was similar to that observed with accuracy performance. ParzenDD/MAX also presented the best results: the highest mean performance (0.7411), the highest number of wins (five, tied with GaussianDD/MAX and MSTDD/MAX), and the best average rank (4.06). ParzenDD/DTs, SVDD/DTs, ParzenDD/ECOC and SVDD/ECOC presented the worst performances. These

Table 8 – Kappa statistic performance of DBDES using DTs, ECOC, and MAX aggregation methods for four one-class classifiers. The best result for each database is in bold. The last row represents the number of wins, ties, and losses achieved by each technique.

	Decision Templates				Error Correcting Output Codes				Maximum Support			
Database	GaussianDD	ParzenDD	SVDD	MSTDD	GaussianDD	ParzenDD	SVDD	MSTDD	GaussianDD	ParzenDD	SVDD	MSTDD
Automobile	0.5775	0.5391	0.4086	0.6749	0.5806	0.5784	0.3765	0.5001	0.4220	0.6610	0.5044	**0.7303**
Car	0.6191	0.6330	0.5127	0.6425	0.5802	0.4350	0.5538	0.6353	0.4727	**0.7811**	0.4617	0.6765
Cleveland	0.3127	0.1998	0.2894	0.2996	0.1840	0.1677	0.2156	0.1744	**0.3313**	0.2467	0.3307	0.2729
Dermatology	**0.9543**	0.7745	0.8674	0.8704	0.9516	0.8960	0.9200	0.9406	0.9503	0.9227	0.9472	0.9159
Ecoli	0.6724	0.6080	0.6647	0.6684	0.6524	**0.737**	0.5976	0.6763	0.6472	0.7079	0.6430	0.6778
Flare	**0.616**	0.5697	0.5987	0.5904	0.5372	0.5685	0.5307	0.5495	0.5713	0.5626	0.5929	0.5995
Glass	0.3806	0.4307	0.3090	**0.546**	0.3205	0.4513	0.3071	0.3089	0.4099	0.4157	0.4419	0.4384
Glass1	0.4189	0.3790	0.3071	0.4724	0.2253	0.1904	0.2438	0.2374	0.3586	0.3860	0.3068	**0.5556**
Glass6	0.7896	0.7755	0.7952	0.7728	0.7549	0.7281	0.7387	0.7647	0.8134	0.7911	**0.8197**	0.7728
Led7digit	0.6618	0.6931	0.6687	0.6953	0.6638	0.6146	0.6952	0.4614	0.6528	**0.6997**	**0.6997**	0.5909
Letter	0.7420	0.6162	0.5856	0.8339	0.6224	0.6356	0.6418	0.9372	0.8947	0.9375	0.7482	**0.9394**
Lymphography	0.5375	0.3961	0.4754	0.4377	0.4681	0.4152	**0.5806**	0.4499	0.4349	0.4715	0.5699	0.4404
Movement Libras	0.7057	0.4219	0.5127	**0.8276**	0.5922	0.4863	0.6731	0.6595	0.6992	0.8273	0.7412	0.8046
Nursery	0.6486	0.6142	0.6111	0.6603	0.7547	0.6400	0.6130	0.6912	**0.803**	0.7041	0.7058	0.6219
Optdigits	0.9622	0.9354	0.9574	0.9675	0.9460	0.8197	0.9435	0.9566	0.9584	**0.975**	0.9605	0.9644
Page-blocks	**0.7563**	0.7327	0.7091	0.7469	0.6846	0.7321	0.6282	0.7280	0.7406	0.7303	0.7364	0.7376
Penbased	0.9799	0.9534	0.9551	0.9892	0.9684	0.9528	0.9661	0.9900	0.9858	**0.9917**	0.9718	0.9893
Satimage	0.8083	0.7363	0.7703	0.8656	0.8325	0.8503	0.8407	**0.87**	0.8478	0.8693	0.8542	0.8606
Segment	0.9126	0.8833	0.8485	**0.9424**	0.8682	0.9253	0.8677	0.9217	0.9025	0.9146	0.8717	0.9232
Shuttle	0.9761	0.9813	0.9798	0.9828	0.9773	0.9798	0.9769	0.9808	0.9793	0.9788	0.9788	**0.9837**
Texture	0.9888	0.8956	0.8981	0.9655	0.9606	0.9223	0.9137	0.9495	**0.9958**	0.9768	0.9323	0.9691
Vehicle	0.7496	0.4887	0.5486	0.6029	0.6278	0.5276	0.4836	0.5104	**0.7607**	0.6264	0.5553	0.6011
Vehicle2	0.9510	0.8000	0.8014	0.8763	0.8528	0.7071	0.7514	0.8192	**0.9591**	0.9128	0.8345	0.8847
Vowel	0.9022	0.5456	0.4989	0.9444	0.4922	0.4522	0.4156	0.9556	0.9011	0.9733	0.6233	**0.9822**
Yeast	0.4059	0.4385	0.3748	0.4584	0.3271	0.4226	0.2866	0.3473	0.4386	**0.4634**	0.3585	0.3944
Mean	0.7212	0.6417	0.6379	0.7334	0.6570	0.6334	0.6305	0.6806	0.7172	**0.7411**	0.6876	0.7331
Win/tie/loss	3/0/22	0/0/25	0/0/25	3/0/22	0/0/25	1/0/24	1/0/24	1/0/24	**5/0/20**	4/1/20	1/1/23	**5/0/20**
Avg. ranks	5.22	8.18	8.64	4.19	7.86	8.01	8.94	6.52	5.44	**4.06**	6.32	4.60

Source: The author.

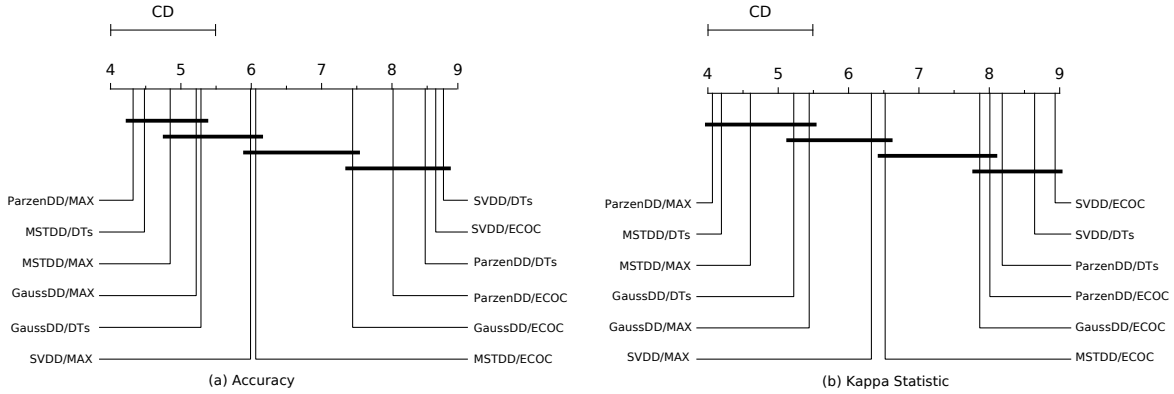
configurations obtained an average rank higher than eight, their mean Kappa statistic performances were the four worst and only ParzenDD/ECOC and SVDD/ECOC achieved one win each. Again, MAX aggregation yielded the best results obtaining 16 wins. DTs achieved six wins and ECOC, only three wins.

Friedman test was employed to determine if the distinct configurations present significantly different performances. The test output was $p - value = 3.18^{-59}$, for accuracy and $p - value = 1.73^{-63}$, for Kappa Statistics, meaning that the difference of performance among the configurations is significant for both metrics. Then, Nemenyi *post-hoc* test was used to identify which of the configurations differ from each other. Figure 11 shows the results for Nemenyi *post-hoc test* for accuracy (a) and Kappa statistic (b).

From the plots, we can see that ParzenDD/MAX presents the best performances, but, some other configurations worth to be mentioned, since their performances do not present significant differences from the best configuration, according to the critical difference (CD) on Nemenyi test: MSTDD/DTs, MSTDD/MAX, GaussianDD/DTs, and GaussianDD/MAX.

A quantitative analysis of the generation and selection phases of DBDES was carried out,

Figure 11 – Result for Nemenyi *post-hoc* test for (a) accuracy and (b) Kappa Statistic performances of DBDES.



Source: The author.

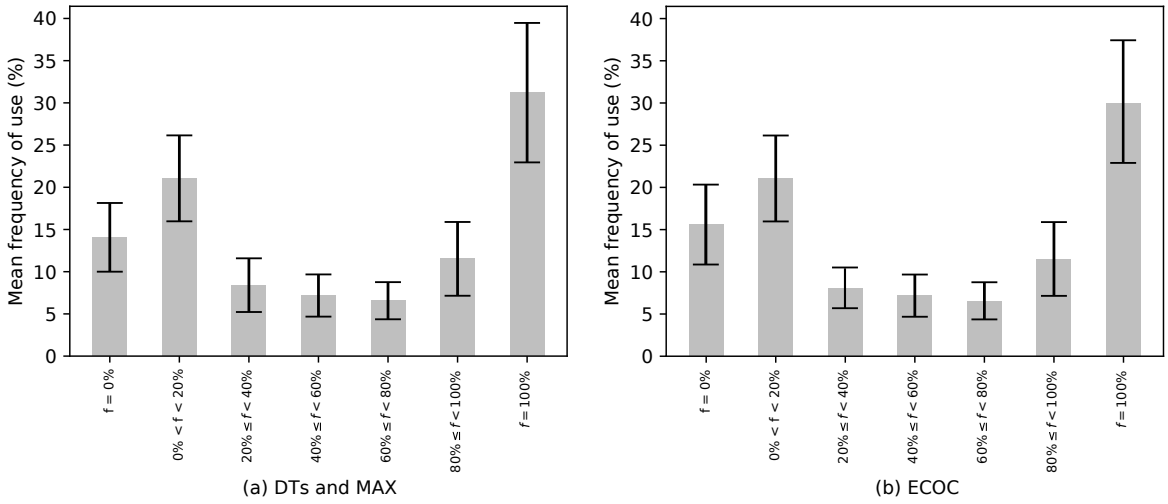
similar to that in Section 4.2.1, for MODES. DBDES segments the training data using the OPTICS algorithm, which extracts hierarchical non-exclusive clusters. An OCC is trained for each cluster. In the selection phase, DBDES selects, to compose the ensemble, an OCC for each cluster of which the test instance is a members. Figure 12 exhibits the distribution for the frequency of selection of OCCs with DBDES: never selected, up to 20%, 40%, 60%, 80%, and 100% of the test instances. Again, the frequencies are equal for Decision Templates (DTs) and MAX, since both techniques use the original classes. ECOC creates meta-binary problems which use meta-classes. For this reason, the frequencies for ECOC are shown separately on the right side of the picture.

The vast majority of the OCCs are selected at least once. Only approximately 15% of the OCC are never selected. More than 20% of the OCCs are selected for at most 20% of the test instances. It is worth noting a high number of OCCs are selected for 100% of the test instances. This can be explained by the fact that OPTICS extracts hierarchical non-exclusive partitions and, in many cases, a large cluster embracing all (or almost all) data instances is identified. Hence, the majority of the test examples belonging to this class are members of such a cluster, which leads to a high frequency of selection.

4.2.3 Experiment 3

In this experiment, we compare MODES and DBDES with DES_{THR} (KRAWCZYK et al., 2018). The comparison involves the most effective configuration for each technique, namely, MSTDD/ DTs for MODES, ParzenDD/MAX for DBDES, and SVDD/DTs for DES_{THR}. We

Figure 12 – Frequency of use of OCCs with DBDES for DTs and MAX (a) and ECOC (b).



Source: The author.

also evaluate DES_{THR} with GaussianDD, using DTs, ECOC, and MAX, since this OCC model was not evaluated in (KRAWCZYK et al., 2018).

Additionally, we compare these techniques with the static combination of one OCC for each class (KRAWCZYK; WOŹNIAK; HERRERA, 2015). For this technique, we carried out a preliminary evaluation involving four OCC models (GaussianDD, ParzenDD, SVDD, and MSTDD) and three aggregation techniques (DTs, ECOC, and MAX). The preliminary evaluation aimed at selecting for Experiment 3 the best performing configuration for each aggregation technique. Considering both Accuracy and Kappa statistic, the evaluation identified that MSTDD/DTs, MSTDD/ECOC, and ParzenDD/DTs were the top-performing configurations for the static combination of OCCs.

Tables 9 and 10 show the results obtained in Experiment 3. The analysis of the mean accuracy and Kappa statistic performances indicates that MODES and DBDES outperform all configurations of DES_{THR} and the static combination of OCCs. MODES was the best performing technique obtaining the highest mean accuracy (84.05) and Kappa statistic (0.7522) and DBDES was the second one, scoring 82.99 for mean accuracy and 0.7411 for Kappa statistic. DES_{THR} presented good results with GaussianDD/MAX and GaussianDD/DTs (third and fourth top-performing). Furthermore, MODES and DBDES obtained the best and second best average rankings, respectively. Regarding the win/tie/loss (number of databases in which a technique performs better than, equals to, or worse than another technique), MODES presented the best results with 8 wins for accuracy and 7 wins for Kappa statistic. DBDES obtained 5 wins for both accuracy and Kappa statistic. DES_{THR} with GaussianDD/MAX obtained

Table 9 – Accuracy performance (in %) of MODES, DBDES, DES_{THR}, and Static aggregation of OCCs. The best result for each database is in bold. The last rows represent the mean performance across all databases, the number of wins, ties, and losses achieved by each technique, and the average rankings.

	MODES	DBDES	DES_{THR}				Static		
Dataset	MSTDD/DTs	ParzenDD/MAX	SVDD/DTs	GaussianDD/DTs	GaussianDD/ECOC	GaussianDD/MAX	MSTDD/DTs	ParzenDD/MAX	MSTDD/ECOC
Automobile	78.69	73.53	57.82	67.80	68.15	58.54	**79.39**	75.48	15.21
Car	84.84	**89.99**	78.42	82.58	78.94	76.56	82.06	89.58	81.54
Cleveland	53.88	47.76	54.83	56.22	53.56	**58.90**	47.76	42.69	53.17
Dermatology	92.74	93.86	93.55	**96.36**	96.12	95.20	92.47	92.18	67.33
Ecoli	77.69	**78.88**	76.49	75.31	69.96	74.42	64.93	41.38	58.92
Flare	69.61	65.20	64.35	**69.98**	61.44	65.01	64.54	61.54	29.08
Glass	**71.54**	57.01	50.93	53.73	51.40	59.36	64.51	54.20	49.07
Glass1	**81.33**	72.46	66.38	71.98	69.16	73.39	74.33	72.46	64.49
Glass6	95.32	95.32	**96.26**	94.85	94.39	**96.26**	93.00	93.47	93.94
Led7digit	72.20	73.00	**74.00**	70.80	68.00	68.80	70.40	72.60	20.60
Letter	90.52	**93.99**	63.73	76.92	48.15	89.64	92.75	78.96	31.68
Lymphography	74.32	72.30	**77.77**	76.34	73.68	70.34	43.56	44.92	4.69
Movement Libras	**86.67**	86.11	78.06	75.28	65.28	74.44	86.11	85.00	48.89
Nursery	74.58	79.76	76.05	74.85	73.75	**86.38**	51.59	46.16	57.39
Optdigits	98.02	**98.15**	96.81	97.21	96.01	96.94	97.31	97.92	84.45
Page-blocks	**95.01**	93.95	93.99	94.99	93.21	94.19	77.50	33.76	90.18
Penbased	**99.27**	99.25	96.12	98.07	91.38	98.60	99.15	99.14	87.91
Satimage	**90.35**	89.34	85.72	85.33	83.10	87.58	89.42	88.83	44.33
Segment	**95.37**	92.68	86.80	92.42	88.70	91.60	93.03	89.22	73.55
Shuttle	**99.42**	99.24	99.28	99.14	99.19	99.28	93.62	88.28	97.64
Texture	98.53	98.38	93.31	99.24	97.25	**99.71**	95.47	97.78	81.67
Vehicle	70.45	71.99	64.54	81.21	72.10	**82.04**	69.98	71.99	43.97
Vehicle2	97.64	96.57	92.79	98.11	94.56	**98.47**	94.92	96.57	84.64
Vowel	98.48	97.58	67.58	90.71	50.51	90.71	**99.60**	98.28	27.78
Yeast	54.78	**58.42**	50.74	54.38	45.08	56.47	47.44	51.62	34.50
Mean	**84.05**	82.99	77.45	81.35	75.32	81.71	78.59	74.56	57.06
Win/tie/loss	**8/0/17**	5/0/20	2/1/22	2/0/23	0/0/25	5/1/19	2/0/23	0/0/25	0/0/25
Avg. ranks	**2.82**	3.34	5.68	4.22	6.31	4.19	5.03	5.26	8.15

Source: The author.

5 wins and 1 tie for both accuracy and Kappa statistic, while DES_{THR} with GaussianDD/DTs obtained 2 wins for both metrics.

We also employed the Friedman test to determine if the performances of the techniques present a statistically significant difference. The test output was $p - value = 1.61^{-72}$, for accuracy, and $p - value = 1.27^{-76}$, for Kappa statistics, meaning that the difference of performance among the configurations is significant for both metrics. Then, Nemenyi *post-hoc* test was used to identify which of the configurations differ from each other. Figure 13 shows the results for Nemenyi *post-hoc test* for accuracy (a) and Kappa statistic (b).

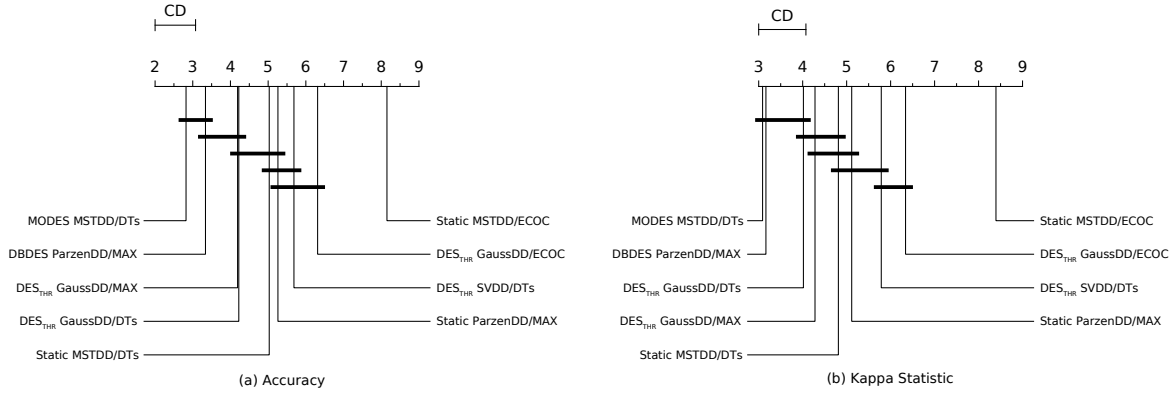
MODES presents the best performance for accuracy and only DBDES achieves accuracy performance that does not present a statistically significant difference to MODES. DES_{THR} with GaussianDD/MAX and GaussianDD/DTs, however, obtained accuracy performance comparable to DBDES, i.e., without statistically significant difference. In terms of Kappa statistic, MODES and DBDES achieved the best performances and DES_{THR} with GaussianDD/DTs achieved performance comparable to both techniques. The performances of the other techniques were worse and the statistical tests showed a significant difference from the top-performing techniques for both accuracy and Kappa statistic. Thus, the next analysis will include only the four top-performing techniques: MODES, DBDES, and DES_{THR} with Gaus-

Table 10 – Kappa performance of MODES, DBDES, DES_{THR}, and Static aggregation of OCCs. The best result for each database is in bold. The last rows represent the mean performance across all databases, the number of wins, ties, and losses achieved by each technique, and the average rankings.

	MODES	DBDES	DES_{THR}				Static		
Dataset	MSTDD/DTs	ParzenDD/MAX	SVDD/DTs	GaussianDD/DTs	GaussianDD/ECOC	GaussianDD/MAX	MSTDD/DTs	ParzenDD/MAX	MSTDD/ECOC
Automobile	0.6460	0.6610	0.4412	0.5775	0.5727	0.4220	**0.7344**	0.6910	0.0307
Car	0.6808	**0.7811**	0.5287	0.6157	0.5728	0.4104	0.6362	0.7704	0.6275
Cleveland	0.2996	0.2467	0.2946	0.3091	0.1840	**0.3313**	0.2581	0.2080	0.0624
Dermatology	0.9086	0.9227	0.9190	**0.9543**	0.9514	0.9395	0.9050	0.9013	0.5647
Ecoli	0.6903	**0.7079**	0.6770	0.6630	0.5573	0.6521	0.5507	0.3230	0.3474
Flare	0.6146	0.5626	0.5503	**0.6183**	0.5190	0.5610	0.5563	0.5236	0.1587
Glass	**0.6128**	0.4157	0.3657	0.3740	0.3202	0.4096	0.5429	0.3860	0.2820
Glass1	**0.5810**	0.3860	0.3371	0.4189	0.2253	0.3586	0.4567	0.3883	0.0234
Glass6	0.7855	0.7911	**0.8197**	0.7699	0.7549	0.8134	0.7514	0.7746	0.7398
Led7digit	0.6909	0.6997	**0.7110**	0.6752	0.6450	0.6528	0.6709	0.6953	0.1255
Letter	0.9014	**0.9375**	0.4845	0.7599	0.4608	0.8923	0.9245	0.7812	0.2893
Lymphography	0.4977	0.4715	**0.5711**	0.5375	0.4681	0.4152	0.2508	0.2882	0.0120
Movement Libras	**0.8434**	0.8273	0.7426	0.7090	0.5922	0.6959	0.8373	0.8241	0.4042
Nursery	0.6315	0.7041	0.5891	0.6391	0.6129	**0.8000**	0.4247	0.3985	0.3759
Optdigits	0.9733	**0.9750**	0.9569	0.9622	0.9460	0.9586	0.9637	0.9719	0.7875
Page-blocks	**0.7564**	0.7303	0.6217	0.7557	0.6070	0.7375	0.3521	0.1078	0.5153
Penbased	**0.9919**	0.9917	0.8765	0.9786	0.9042	0.9844	0.9906	0.9905	0.8656
Satimage	**0.8812**	0.8693	0.7915	0.8202	0.7893	0.8478	0.8703	0.8631	0.2968
Segment	**0.9460**	0.9146	0.8460	0.9116	0.8682	0.9020	0.9187	0.8742	0.6914
Shuttle	0.9460	0.9788	**0.9798**	0.9761	0.9773	**0.9798**	0.8403	0.7112	0.9343
Texture	0.9789	0.9768	0.9046	0.9890	0.9606	**0.9958**	0.9361	0.9683	0.7510
Vehicle	0.6060	0.6264	0.5284	0.7496	0.6278	**0.7607**	0.5997	0.6263	0.2480
Vehicle2	0.9372	0.9128	0.8225	0.9510	0.8528	**0.9591**	0.8736	0.9128	0.5470
Vowel	0.9833	0.9733	0.6433	0.8978	0.4556	0.8978	**0.9956**	0.9811	0.2056
Yeast	0.4221	**0.4634**	0.3707	0.4075	0.2501	0.4386	0.3600	0.3937	0.0707
Mean	**0.7522**	0.7411	0.6550	0.7208	0.6270	0.7127	0.6880	0.6542	0.3983
Win/tie/loss	**7/0/18**	5/0/20	3/1/21	2/0/23	0/0/25	5/1/19	2/0/23	0/0/25	0/0/25
Avg. ranks	**3.09**	3.16	5.79	4.02	6.34	4.28	4.81	5.12	8.40

Source: The author.

Figure 13 – Result for Nemenyi *post-hoc* test for (a) accuracy and (b) Kappa Statistic performances of MODES, DBDES, DES_{THR}, and the static combination of OCCs.



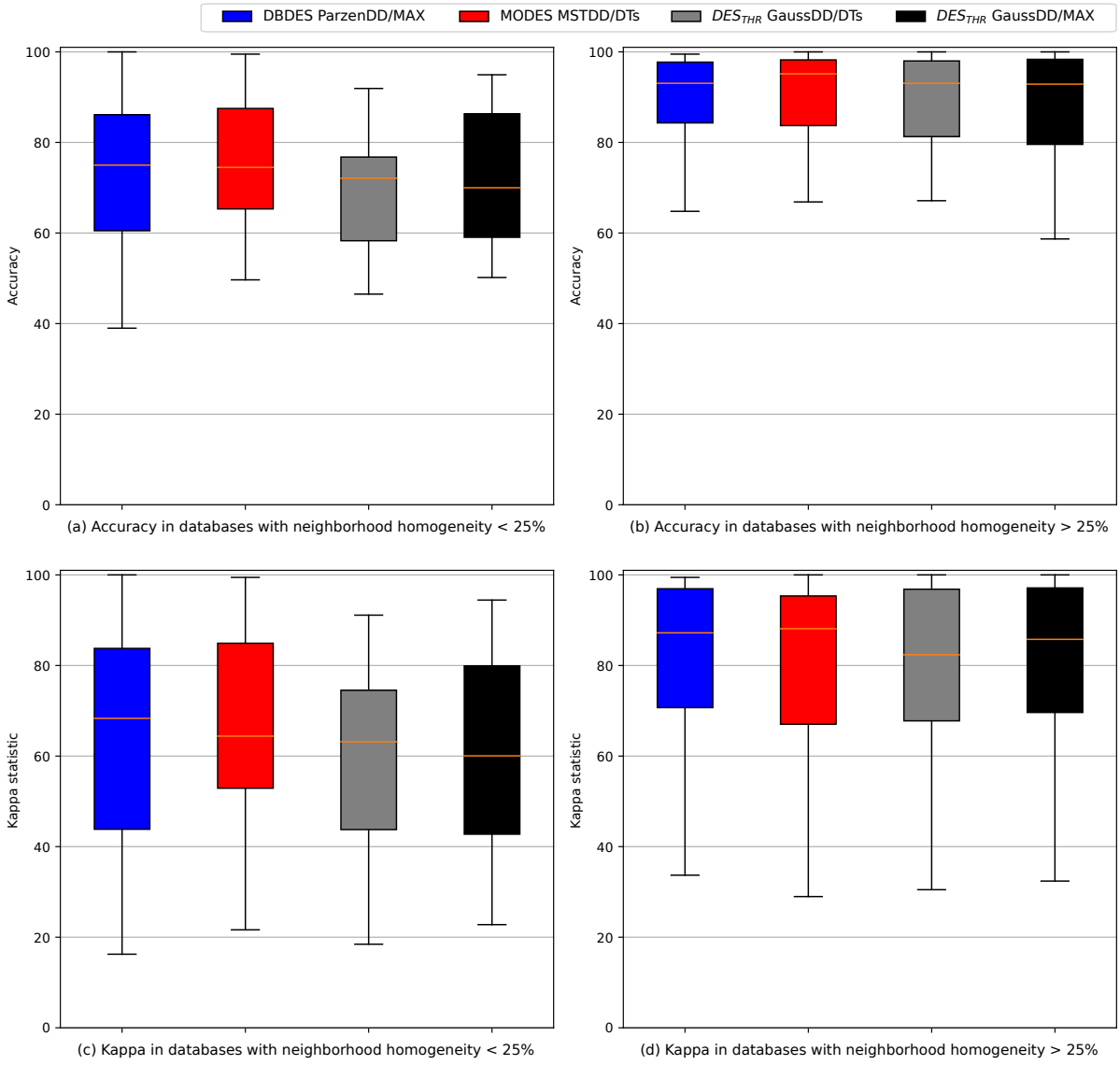
Source: The author.

sianDD/MAX and GaussianDD/DTs.

In order to evaluate the applicability of the proposed techniques, an analysis of the region of competence, i.e., the neighborhood of the test instances, was performed. We identified that, for some databases, a considerable proportion of the instances are in regions where the neighbors belong to only one class. We say that such instances are in a homogeneous neighborhood. When an instance is in a homogeneous neighborhood, the classification is directly given by

the class of the instances in the neighborhood, without using the Dynamic Ensemble Selection (DES) of OCCs.

Figure 14 – Accuracy and Kappa statistic for the best configurations of DBDES, MODES, and DES_{THR} according to the homogeneity of the neighborhood. Databases where more than 25% of the instances present more than one class in the neighborhood are shown in the left side of the figure. The right side shows databases where less than 25% of the instances present more than one class in the neighborhood.



Source: The author.

Figure 14 shows the accuracy and Kappa statistic of the top-performing configurations of DBDES, MODES, and DES_{THR}, which were identified in the previous analysis. The figure shows on the left side databases where up to 25% of the instances are in homogeneous regions and on the right side databases where more than 25% of the instances are in homogeneous regions. Additionally, Table 11 shows the mean accuracy and Kappa statistic of each technique according to the homogeneity of the neighborhood in the databases.

Table 11 – Mean accuracy and Kappa statistic for top-performing configurations of DBDES, MODES, and DES_{THR} according to the homogeneity of the neighborhoods in the databases.

	Accuracy		Kappa	
	Hom. $\geq$ 25%	Hom. <25%	Hom. $\geq$ 25%	Hom. <25%
DBDES ParzenDD/MAX	89.01	73.83	80.85	64.00
MODES MSTDD/DTs	89.70	75.56	81.85	65.29
DES_{THR} GaussianDD/MAX	88.61	71.36	79.07	59.56
DES_{THR} GaussianDD/DTs	89.12	69.70	80.89	58.87

Source: The author.

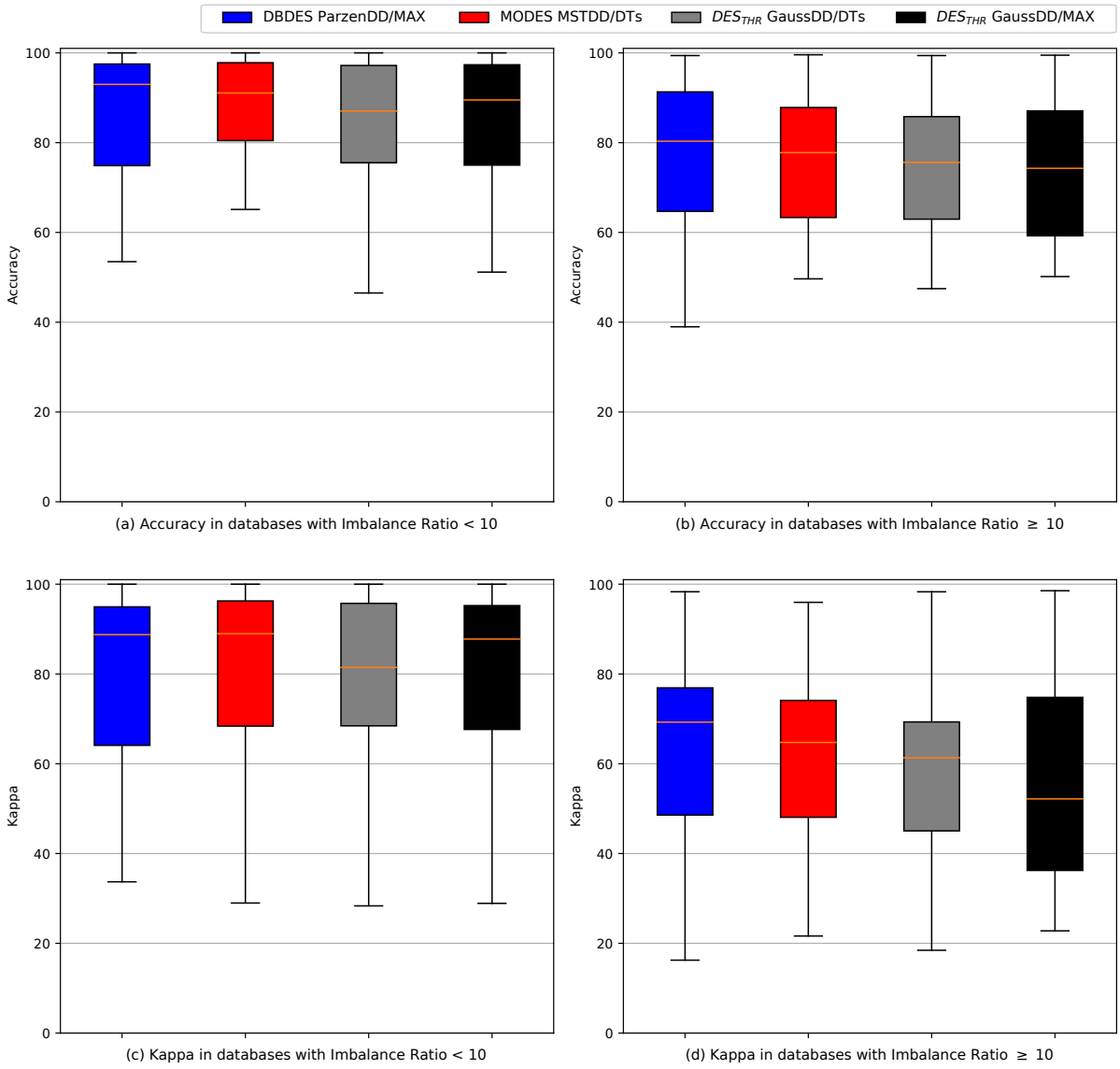
It is noticeable that, for databases with homogeneous neighborhoods, the performances of the techniques are more stable for both accuracy and Kappa statistic. The reason is that, in such databases, more instances are classified in the same way by the techniques (assigning the test instance to the single class in the neighborhood). The performance in databases with homogeneous neighborhoods is higher than in databases with heterogeneous neighborhoods, which is expected since it is known that instances in complex neighborhoods are hard to classify.

MODES presents the best performance for databases with both homogeneous and heterogeneous neighborhoods. Furthermore, DBDES also outperforms DES_{THR} for both homogeneous and heterogeneous neighborhoods. However, in databases with heterogeneous neighborhoods, the difference between the proposed techniques and DES_{THR} is bigger. The behavior of the performances of the techniques in terms of accuracy and Kappa statistic were similar. However, the difference between the Kappa statistic performances of the techniques in databases with heterogeneous neighborhoods to those with homogeneous neighborhoods was bigger than the accuracy, specially for DES_{THR}.

We also carried out a performance analysis regarding the Imbalance Ratio (IR). The imbalance ratio was computed as the cardinality of the largest class divided by the cardinality of the smallest class. The databases were separated into two groups: (a) low IR databases ($IR < 10$), composed of 16 databases; and (b) high IR databases ($IR \geq 10$), composed of 9 databases. Figure 15 and Table 12 show the accuracy and Kappa statistic performances obtained by DBDES, MODES, and DES_{THR} in the two groups of databases.

The boxplots show that the performances of all the techniques are higher and more stable in databases with low IR than in databases with high IR. MODES presented the highest accuracy and Kappa statistic for databases with low IR. For databases with high IR, DBDES obtained

Figure 15 – Accuracy and Kappa statistic for the best configurations of DBDES, MODES, and DES_{THR} according to the Imbalance Ratio. Databases with IR < 10 are shown in the left side of the figure. The right side shows databases with IR ≥ 10.



Source: The author.

the highest scores for both accuracy and Kappa statistic. Furthermore, for both groups of databases and both metrics, the proposed techniques outperform DES_{THR}.

Finally, we also evaluated the performances of the techniques with respect to the number of classes of the databases. We split the databases into two groups, the first containing databases with less than 7 classes and the second with databases with at least 7 classes. Figure 16 and Table 13 show the results of the evaluation.

For both accuracy and Kappa statistic, MODES and DBDES presented less advantage against DES_{THR} in databases with less than 7 classes than in databases with at least 7 classes. In fact, DBDES obtained worse mean performance than both configurations of DES_{THR} in

Table 12 – Mean accuracy and Kappa statistic for top-performing configurations of DBDES, MODES, and DES_{THR} according to the Imbalance Ratio.

	Accuracy		Kappa	
	IR <10	IR ≥ 10	IR <10	IR ≥ 10
DBDES ParzenDD/MAX	86.25	77.09	79.89	63.83
MODES MSTDD/DTs	88.00	77.02	82.72	61.89
DES_{THR} GaussianDD/MAX	85.48	75.01	78.93	57.63
DES_{THR} GaussianDD/DTs	84.51	75.73	78.37	60.90

Source: The author.

Table 13 – Mean accuracy and Kappa statistic for top-performing configurations of DBDES, MODES, and DES_{THR} according to the number of classes of the databases.

	Accuracy		Kappa	
	<7 classes	≥ 7 classes	<7 classes	≥ 7 classes
DBDES ParzenDD/MAX	80.97	86.49	68.96	83.27
MODES MSTDD/DTs	82.93	86.04	71.19	82.39
DES_{THR} GaussianDD/MAX	81.18	82.66	67.00	78.84
DES_{THR} GaussianDD/DTs	81.30	81.45	69.01	77.54

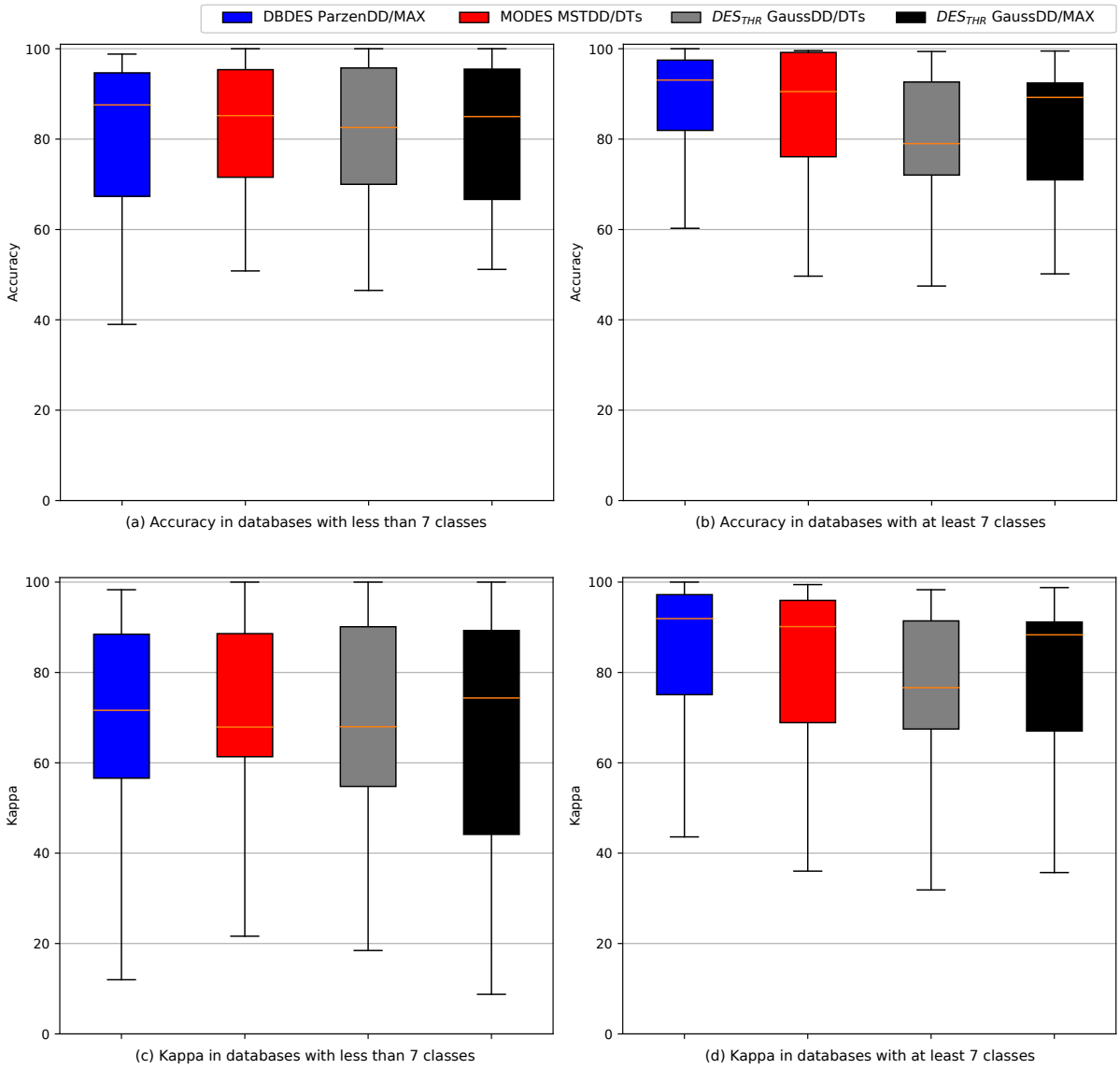
Source: The author.

databases with less than 7 classes while MODES presented a slight advantage. For databases with more than 7 classes, both MODES and DBDES presented better mean performance than the two configurations of DES_{THR}. In addition, DBDES achieved the best results, overcoming MODES for both accuracy and Kappa statistic.

The three evaluations presented above (regarding neighborhood homogeneity, class imbalance, and number of classes) aimed at deepening the understanding of the performance of the proposed techniques in databases with difficulties that may hinder classification performance. The proposed techniques presented better results than literature techniques in the cases of such difficulties, indicating that the approach proposed in this work is a good alternative to tackle problems with the aforementioned issues.

In order to better evaluate the differences between the performances of each technique, we carried out a Wilcoxon signed-rank test (BENAVOLI; CORANI; MANGILI, 2016). The test was performed in a pairwise manner, verifying the null hypothesis that each of the proposed techniques (one at a time) presents similar performance to the other technique and the alternative hypothesis that each of the proposed techniques presents better performance than the other

Figure 16 – Accuracy and Kappa statistic for the best configurations of DBDES, MODES, and DES_{THR} according to the number of classes of the databases. Databases with less than 7 classes are shown in the left side of the figure. The right side shows databases at least 7 classes.



Source: The author.

techniques.

Table 14 summarizes the results of the Wilcoxon signed-rank test. Two levels of significance were adopted: $p - value = 0.1$, meaning evidence that the technique in the column performs better than the technique in the row, and $p - value = 0.05$, meaning strong evidence that technique in the column performs better than the technique in the row. In the table, DBDES ParzenDD/MAX and MODES MSTDD/DTs are represented by DBDES and MODES in the columns, for space reasons.

The Wilcoxon test indicated strong evidence that the proposed techniques present better performance than the two top-performing DES_{THR} configurations. The only exception was

Table 14 – Wilcoxon signed rank test results comparing the top-performing configurations of DBDES, MODES, and DES_{THR}. Convention adopted: "$>>$" means strong evidence that the technique in the column presents greater effectiveness than the technique in the row; "$>$" means evidence that the technique in the column presents greater effectiveness than the technique in the row; "$\sim$" means no evidence that the technique in the column presents greater effectiveness than the technique in the row.

	Accuracy		Kappa statistic	
	DBDES	MODES	DBDES	MODES
DBDES ParzenDD/MAX		$\sim$		$\sim$
MODES MSTDD/DTs	$\sim$		$\sim$	
DES_{THR} GaussianDD/MAX	$\gg$	$\gg$	$\gg$	$\gg$
DES_{THR} GaussianDD/DTs	$>$	$\gg$	$\gg$	$\gg$

Source: The author.

observed in the comparison between DBDES ParzenDD/MAX and DES_{THR} GaussianDD/DTs in terms of accuracy, where the test indicated evidence that the proposed technique performs better than the other technique. In addition, the test also indicated that there was no evidence of a significant difference between the performances of DBDES ParzenDD/MAX and MODES MSTDD/DTs, for both accuracy and Kappa statistic, although MODES presented higher mean scores for both metrics.

Finally, we also evaluated the execution times of the techniques to verify if our supposition that DBDES is faster than MODES is confirmed. In addition, this analysis is relevant to give an outlook of the trade-off between classification accuracy and execution time of the techniques evaluated.

Table 15 – Mean execution time across all databases for the top-performing configurations of MODES, DBDES, and DES_{THR}. The time is presented in seconds and is divided by training, test, and total.

	Training	Test	Total
DBDES ParzenDD/MAX	3.62	0.48	4.10
MODES MSTDD/DTs	97.22	14.78	112.00
DES_{THR} GaussianDD/MAX	0.56	0.33	0.89
DES_{THR} GaussianDD/DTs	0.55	6.81	7.36

Source: The author.

Table 15 shows the mean execution time, in seconds, that each technique took to classify the databases. The experiments were executed using an Intel i5 8th generation with 12 GB RAM. We did not apply any parallelism technique for none of the algorithms. Tables with the results for each database are present in Appendix A. It is noticeable that DES_{THR}

GaussianDD/MAX presented the lowest execution times. It showed the fastest test phase, and its execution time in the training phase was quite similar to the fastest one (DES_{THR} GaussianDD/DTs). DBDES presented the third fastest training phase, second fastest test phase, and second fastest total time. It is important to remark that DBDES presented higher accuracy than both configurations of DES_{THR} but with lower execution time than DES_{THR} GaussianDD/DTs.

MODES, however, was the slowest technique for both phases. This was expected since the execution of the clustering procedure in MODES is computationally intense. Nevertheless, MODES presented the best classification accuracy in the experiments. Hence, it is worth using this technique if execution time is not an issue.

5 FINAL REMARKS

This research proposed two methods for multi-class classification, named One-class Dynamic Ensemble Selection for Multi-class problems (MODES) and Density-Based Dynamic Ensemble Selection (DBDES). Both techniques aim to improve the robustness and classification performance in problems with complex intra-class data distribution. They decompose the original multi-class problem into multiple one-class problems and employ a clustering-based approach to generate pools of One-Class Classifiers. Dynamic Ensemble Selection (DES) is applied to classify each test instance. MODES uses a center-based clustering algorithm, like K-means, and relies on a set of cluster validity indices to define the numbers of clusters in the data. DBDES employs the Ordering Points To Identify the Clustering Structure (OPTICS) algorithm for clustering and extracts hierarchical non-exclusive partitions.

Experiments were carried out in a comprehensive experimental setting. The first experiment identified the best configuration for MODES. The best performance was achieved using the One-Class Classifier (OCC) model Minimum Spanning Tree Data Descriptor (MSTDD) and the aggregation technique Decision Templates (DTs). The second experiment identified the best configuration for DBDES, namely, the OCC model Parzen Data Descriptor (ParzenDD) and Maximum Support (MAX).

In the third experiment, the top-performing configurations of MODES and DBDES were compared with literature techniques, using a variety of configurations. The results showed that MODES' average performance is superior to all the configurations of the other methods. DBDES presented the second-best average performance. Furthermore, statistical tests indicated evidence that the performances of the proposed techniques are superior to the literature (strong evidence in three out of four cases).

In addition, we identified that the proposed techniques perform better for databases containing both complex and simple neighborhoods (high or low presence of instances of other classes in the neighborhood, respectively) and for both balanced and imbalanced databases. However, in databases with such difficulties (complex neighborhood and class imbalance), the advantage of the performances of the proposed techniques is more remarkable. Hence, the experiments indicated that the adoption of a clustering-based approach for the dynamic ensemble of OCCs selection improves the classification performance in the one-class decomposition scheme.

The proposed techniques are composed of two levels: the first is responsible for one-class classification, i.e., whether the test instance belongs or not to the target class; and the second is responsible for the re-composition of the multi-class problem, i.e., aggregating the outputs of the one-class problems to give the final multi-class decision. Hence, the proposed techniques can, optionally, be easily adapted to be used in one-class problems.

For future works, we aim to evaluate the choice of the algorithm (MODES or DBDES) according to characteristics of the database. For instance, Figure 15 and Table 12 indicate that DBDES achieves better results than MODES in databases with high imbalance ratio. As it requires less execution time, DBDES may be more adequate than MODES to large databases or an environment where time constraints are imposed. Thus, the best technique for a database may be determined using these and other heuristics like number of instances, dimensionality, homogeneity of the neighborhoods, and so forth.

Also, we aim to delve deeper into the use of clustering-based dynamic ensemble of OCCs selection for one-class problems. Since this approach presented good results in a multi-class setting, it is worth investigating whether its adoption in one-class problems.

REFERENCES

AGGARWAL, C. C.; REDDY, C. K. Data clustering. *Algorithms and applications. Chapman&Hall/CRC Data mining and Knowledge Discovery series, Londra*, Citeseer, 2014.

ANKERST, M.; BREUNIG, M. M.; KRIEGEL, H.-P.; SANDER, J. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, ACM New York, NY, USA, v. 28, n. 2, p. 49–60, 1999.

ARBELAITZ, O.; GURRUTXAGA, I.; MUGUERZA, J.; PÉREZ, J. M.; PERONA, I. An extensive comparative study of cluster validity indices. *Pattern Recognition*, Elsevier, v. 46, n. 1, p. 243–256, 2013.

BALL, G. H.; HALL, D. J. *ISODATA, a novel method of data analysis and pattern classification*. [S.l.], 1965.

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

CALIŃSKI, T.; HARABASZ, J. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, Taylor & Francis, v. 3, n. 1, p. 1–27, 1974.

CAMPELLO, R. J.; KRÖGER, P.; SANDER, J.; ZIMEK, A. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 10, n. 2, p. e1343, 2020.

CAO, P.; ZHANG, S.; TANG, J. Preprocessing-free gear fault diagnosis using small datasets with deep convolutional neural network-based transfer learning. *Ieee Access*, IEEE, v. 6, p. 26241–26253, 2018.

CHARRAD, M.; GHAZZALI, N.; BOITEAU, V.; NIKNAFS, A.; CHARRAD, M. M. Package 'nbclust'. *Journal of Statistical Software*, v. 61, p. 1–36, 2014.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, v. 16, p. 321–357, 2002.

COHEN, J. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, Sage Publications Sage CA: Thousand Oaks, CA, v. 20, n. 1, p. 37–46, 1960.

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

CUPERTINO, T. H.; ZHAO, L.; CARNEIRO, M. G. Network-based supervised data classification by using an heuristic of ease of access. *Neurocomputing*, Elsevier, v. 149, p. 86–92, 2015.

DAS, B.; COOK, D. J.; KRISHNAN, N. C.; SCHMITTER-EDGECOMBE, M. One-class classification-based real-time activity error detection in smart homes. *IEEE journal of selected topics in signal processing*, IEEE, v. 10, n. 5, p. 914–923, 2016.

DAVIES, D. L.; BOULDIN, D. W. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, n. 2, p. 224–227, 1979.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern classification*. [S.l.]: John Wiley & Sons, 2012.

DUNN, J. C. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, Taylor & Francis, v. 4, n. 1, p. 95–104, 1974.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *kdd*. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231.

EVERITT, B. S.; LANDAU, S.; LEESE, M.; STAHL, D. *Cluster analysis 5th ed*. [S.l.]: John Wiley, 2011.

FRAGOSO, R. C.; CAVALCANTI, G. D.; PINHEIRO, R. H.; OLIVEIRA, L. S. Dynamic selection and combination of one-class classifiers for multi-class classification. *Knowledge-Based Systems*, Elsevier, v. 228, p. 107290, 2021.

GALAR, M.; FERNÁNDEZ, A.; BARRENECHEA, E.; BUSTINCE, H.; HERRERA, F. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, Elsevier, v. 44, n. 8, p. 1761–1776, 2011.

GÖRNITZ, N.; LIMA, L. A.; MÜLLER, K.-R.; KLOFT, M.; NAKAJIMA, S. Support vector data descriptions and k-means clustering: One class? *IEEE transactions on neural networks and learning systems*, IEEE, v. 29, n. 9, p. 3994–4006, 2017.

HAHSLER, M.; PIEKENBROCK, M.; DORAN, D. dbscan: Fast density-based clustering with r. *Journal of Statistical Software*, v. 91, p. 1–30, 2019.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. On clustering validation techniques. *Journal of intelligent information systems*, Springer, v. 17, n. 2-3, p. 107–145, 2001.

HALKIDI, M.; VAZIRGIANNIS, M.; BATISTAKIS, Y. Quality scheme assessment in the clustering process. In: SPRINGER. *European Conference on Principles of Data Mining and Knowledge Discovery*. [S.l.], 2000. p. 265–276.

HARTIGAN, J. A. *Clustering algorithms*. [S.l.]: Wiley, 1975.

HE, H.; GARCIA, E. A. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, Ieee, v. 21, n. 9, p. 1263–1284, 2009.

HUBERT, L. J.; LEVIN, J. R. A general statistical framework for assessing categorical clustering in free recall. *Psychological bulletin*, American Psychological Association, v. 83, n. 6, p. 1072, 1976.

JAIN, A.; NANDAKUMAR, K.; ROSS, A. Score normalization in multimodal biometric systems. *Pattern recognition*, Elsevier, v. 38, n. 12, p. 2270–2285, 2005.

JAIN, A. K. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, Elsevier, v. 31, n. 8, p. 651–666, 2010.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. *ACM computing surveys (CSUR)*, Acm New York, NY, USA, v. 31, n. 3, p. 264–323, 1999.

JOHNSON, J. M.; KHOSHGOFTAAR, T. M. Survey on deep learning with class imbalance. *Journal of Big Data*, Springer, v. 6, n. 1, p. 1–54, 2019.

JUSZCZAK, P.; TAX, D. M.; PE, E.; DUIN, R. P. et al. Minimum spanning tree based one-class classifier. *Neurocomputing*, Elsevier, v. 72, n. 7-9, p. 1859–1869, 2009.

KARAMI, A.; JOHANSSON, R. Choosing dbscan parameters automatically using differential evolution. *International Journal of Computer Applications*, Foundation of Computer Science, v. 91, n. 7, p. 1–11, 2014.

KAUFMAN, L.; ROUSSEEUW, P. J. *Finding groups in data: an introduction to cluster analysis*. [S.l.]: John Wiley & Sons, 2009.

KHAN, K.; REHMAN, S. U.; AZIZ, K.; FONG, S.; SARASVADY, S. Dbscan: Past, present and future. In: IEEE. *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*. [S.l.], 2014. p. 232–238.

KHAN, S. S.; MADDEN, M. G. A survey of recent trends in one class classification. In: SPRINGER. *Irish conference on artificial intelligence and cognitive science*. [S.l.], 2009. p. 188–197.

KIM, B.; LEE, H.; KANG, P. Integrating cluster validity indices based on data envelopment analysis. *Applied Soft Computing*, Elsevier, v. 64, p. 94–108, 2018.

KIM, H.-J.; JO, N.-O.; SHIN, K.-S. Optimization of cluster-based evolutionary undersampling for the artificial neural networks in corporate bankruptcy prediction. *Expert systems with applications*, Elsevier, v. 59, p. 226–234, 2016.

KIM, M.; RAMAKRISHNA, R. New indices for cluster validity assessment. *Pattern Recognition Letters*, Elsevier, v. 26, n. 15, p. 2353–2363, 2005.

KITTLER, J.; HATEF, M.; DUIN, R. P.; MATAS, J. On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 20, n. 3, p. 226–239, 1998.

KO, A. H.; SABOURIN, R.; JR, A. S. B. From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition*, Elsevier, v. 41, n. 5, p. 1718–1731, 2008.

KOHONEN, T.; SCHROEDER, M.; HUANG, T.; MAPS, S.-O. Springer-verlag new york. *Inc., Secaucus, NJ*, v. 43, n. 2, 2001.

KOLESNIKOV, A.; TRICHINA, E.; KAURANNE, T. Estimating the number of clusters in a numerical data set via quantization error modeling. *Pattern Recognition*, Elsevier, v. 48, n. 3, p. 941–952, 2015.

KOPPEL, M.; SCHLER, J. Authorship verification as a one-class classification problem. In: ACM. *Proceedings of the twenty-first international conference on Machine learning*. [S.l.], 2004. p. 62.

KRAWCZYK, B.; GALAR, M.; WOŹNIAK, M.; BUSTINCE, H.; HERRERA, F. Dynamic ensemble selection for multi-class classification with one-class classifiers. *Pattern Recognition*, Elsevier, v. 83, p. 34–51, 2018.

KRAWCZYK, B.; WOŹNIAK, M. Dynamic classifier selection for one-class classification. *Knowledge-Based Systems*, Elsevier, v. 107, p. 43–53, 2016.

KRAWCZYK, B.; WOŹNIAK, M.; CYGANEK, B. Clustering-based ensembles for one-class classification. *Information Sciences*, Elsevier, v. 264, p. 182–195, 2014.

KRAWCZYK, B.; WOŹNIAK, M.; HERRERA, F. On the usefulness of one-class classifier ensembles for decomposition of multi-class problems. *Pattern Recognition*, Elsevier, v. 48, n. 12, p. 3969–3982, 2015.

KRZANOWSKI, W. J.; LAI, Y. A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics*, JSTOR, p. 23–34, 1988.

KUNCHEVA, L. *Combining pattern classifiers. Hoboken*. [S.l.]: New Jersey. John Wiley & Sons, Inc, 2014.

KUNCHEVA, L. I.; BEZDEK, J. C.; DUIN, R. P. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern recognition*, Elsevier, v. 34, n. 2, p. 299–314, 2001.

LE, T.; TRAN, D.; MA, W.; SHARMA, D. A theoretical framework for multi-sphere support vector data description. In: SPRINGER. *International Conference on Neural Information Processing*. [S.l.], 2010. p. 132–142.

LIU, J.; MIAO, Q.; SUN, Y.; SONG, J.; QUAN, Y. Modular ensembles for one-class classification based on density analysis. *Neurocomputing*, Elsevier, v. 171, p. 262–276, 2016.

LUCHI, D.; RODRIGUES, A. L.; VAREJÃO, F. M. Sampling approaches for applying dbscan to large datasets. *Pattern Recognition Letters*, Elsevier, v. 117, p. 90–96, 2019.

MANEVITZ, L. M.; YOUSEF, M. One-class svms for document classification. *Journal of machine Learning research*, v. 2, n. Dec, p. 139–154, 2001.

MCCLAIN, J. O.; RAO, V. R. Clustisz: A program to test for the quality of clustering of a set of objects. *Journal of Marketing Research*, JSTOR, p. 456–460, 1975.

MILLIGAN, G. W. A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, Springer, v. 46, n. 2, p. 187–199, 1981.

POUYANFAR, S.; SADIQ, S.; YAN, Y.; TIAN, H.; TAO, Y.; REYES, M. P.; SHYU, M.-L.; CHEN, S.-C.; IYENGAR, S. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 51, n. 5, p. 1–36, 2018.

PUJOL, O.; RADEVA, P.; VITRIA, J. Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 28, n. 6, p. 1007–1012, 2006.

SCHÖLKOPF, B.; PLATT, J. C.; SHAWE-TAYLOR, J.; SMOLA, A. J.; WILLIAMSON, R. C. Estimating the support of a high-dimensional distribution. *Neural computation*, MIT Press, v. 13, n. 7, p. 1443–1471, 2001.

TAX, D. Ddtools, the data description toolbox for matlab. *Delft University of Technology ed*, 2005.

TAX, D. *DDtools, the Data Description Toolbox for Matlab*. 2018. Version 2.1.3.

TAX, D. M.; DUIN, R. P. Data description in subspaces. In: IEEE. *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. [S.l.], 2000. v. 2, p. 672–675.

TAX, D. M.; DUIN, R. P. Combining one-class classifiers. In: SPRINGER. *International Workshop on Multiple Classifier Systems*. [S.l.], 2001. p. 299–308.

TAX, D. M.; DUIN, R. P. Support vector data description. *Machine learning*, Springer, v. 54, n. 1, p. 45–66, 2004.

TIBSHIRANI, R.; WALTHER, G.; HASTIE, T. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Wiley Online Library, v. 63, n. 2, p. 411–423, 2001.

TSAI, C.-F.; LIN, W.-C. Feature selection and ensemble learning techniques in one-class classifiers: an empirical study of two-class imbalanced datasets. *IEEE Access*, IEEE, v. 9, p. 13717–13726, 2021.

WOJCIECHOWSKI, S.; WOŹNIAK, M. Employing decision templates to imbalanced data classification. In: SPRINGER. *International Conference on Hybrid Artificial Intelligence Systems*. [S.l.], 2020. p. 120–131.

WOLPERT, D. H. The lack of a priori distinctions between learning algorithms. *Neural computation*, MIT Press, v. 8, n. 7, p. 1341–1390, 1996.

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

XIAO, Y.; LIU, B.; CAO, L.; WU, X.; ZHANG, C.; HAO, Z.; YANG, F.; CAO, J. Multi-sphere support vector data description for outliers detection on multi-distribution data. In: IEEE. *2009 IEEE international conference on data mining workshops*. [S.l.], 2009. p. 82–87.

XU, R.; WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on neural networks*, Ieee, v. 16, n. 3, p. 645–678, 2005.

ZHOU, Z.-H. *Ensemble methods: foundations and algorithms*. [S.l.]: Chapman and Hall/CRC, 2012.

APPENDIX A – EXECUTION TIME

Table 16 – Training time of MODES, DBDES, and Dynamic Ensemble Selection with THReshold-based neighborhood pruning (DES_{THR}) top performing configurations and the characteristics of the databases. Imbalance Ratio (IR) is computed as the division of the cardinality of the largest class by the cardinality of the smallest class. Best result for each database in bold. The last rows represent the mean performance across all databases, the number of wins, ties and losses achieved by each technique, and the average rankings.

	Databases characteristics						DBDES	MODES	DES_{THR}	
Name	Instances	Features	Numeric	Nominal	Classes	IR	ParzenDD/MAX	MSTDD/DTs	GaussDD/DTs	GaussDD/MAX
Automobile	159	25	15	10	6	16.0	**0.47**	8.25	**0.47**	**0.47**
Car	1728	6	0	6	4	18.6	1.60	107.02	0.13	**0.11**
Cleveland	297	13	13	0	5	12.3	0.20	10.74	0.08	**0.07**
Dermatology	358	34	34	0	6	5.6	0.25	15.62	**0.15**	0.16
Ecoli	336	7	7	0	8	71.5	0.24	9.72	**0.11**	0.12
Flare	1066	11	9	2	6	7.7	1.06	26.62	0.15	**0.13**
Glass	214	9	9	0	6	8.4	0.14	8.56	**0.06**	**0.06**
Glass1	214	9	9	0	2	1.8	0.15	5.12	**0.04**	**0.04**
Glass6	214	9	9	0	2	6.4	0.15	6.25	**0.03**	0.04
Led7digit	500	7	7	0	10	1.5	0.30	14.19	0.12	**0.11**
Letter	2000	16	16	0	26	1.1	16.52	11.29	**2.08**	2.12
Lymphography	148	18	3	15	4	40.5	0.16	4.54	**0.04**	0.06
Movement Libras	360	90	90	0	7	1.5	**0.38**	24.42	0.40	**0.38**
Nursery	12960	8	0	8	5	2,160.0	18.54	2.96	**0.69**	0.72
Optdigits	5620	64	64	0	4	1.5	7.68	1,224.39	**1.80**	1.87
Page-blocks	5472	10	10	0	5	175.5	7.83	1.89	**0.38**	**0.38**
Penbased	10992	16	16	0	10	1.1	8.13	5.45	**1.78**	1.79
Satimage	6435	36	36	0	6	2.5	6.28	2.99	**1.86**	**1.86**
Segment	2310	19	19	0	7	1.0	2.13	4.87	**0.31**	0.33
Shuttle	5780	9	9	0	7	4,558.0	8.24	1,059.44	**0.38**	0.40
Texture	5500	40	40	0	5	5.0	7.67	33.61	**2.17**	2.20
Vehicle	846	18	18	0	4	1.1	0.54	28.89	**0.12**	0.13
Vehicle2	846	18	18	0	2	2.9	0.57	55.23	**0.10**	0.11
Vowel	990	13	13	0	11	1.0	0.61	25.48	0.19	**0.18**
Yeast	1484	8	8	0	10	92.6	0.78	63.92	**0.17**	**0.17**
Mean							3.62	69.62	**0.55**	0.56
Win/tie/loss							0/2/23	0/0/25	**13/6/6**	5/7/13
Avg. ranks							3.10	3.72	**1.51**	1.66

Source: The author.

Table 17 – Prediction time of MODES, DBDES, and DES_{THR} top performing configurations and the characteristics of the databases. IR is computed as the division of the cardinality of the largest class by the cardinality of the smallest class.. Best result for each database in bold. The last rows represent the mean performance across all databases, the number of wins, ties and losses achieved by each technique, and the average rankings.

	Databases characteristics						DBDES	MODES	DES_{THR}	
Name	Instances	Features	Numeric	Nominal	Classes	IR	ParzenDD/MAX	MSTDD/DTs	GaussDD/DTs	GaussDD/MAX
Automobile	159	25	15	10	6	16.0	**0.12**	5.11	0.65	**0.12**
Car	1728	6	0	6	4	18.6	0.21	6.45	0.67	**0.09**
Cleveland	297	13	13	0	5	12.3	**0.08**	5.79	0.44	**0.08**
Dermatology	358	34	34	0	6	5.6	**0.09**	8.63	0.92	0.10
Ecoli	336	7	7	0	8	71.5	**0.09**	6.53	0.85	0.10
Flare	1066	11	9	2	6	7.7	0.41	9.86	1.01	**0.13**
Glass	214	9	9	0	6	8.4	**0.09**	3.84	0.46	**0.09**
Glass1	214	9	9	0	2	1.8	**0.04**	1.15	0.11	**0.04**
Glass6	214	9	9	0	2	6.4	**0.03**	1.21	0.12	**0.03**
Led7digit	500	7	7	0	10	1.5	0.16	11.26	1.19	**0.13**
Letter	2000	16	16	0	26	1.1	5.14	52.61	83.65	**4.21**
Lymphography	148	18	3	15	4	40.5	0.06	1.72	0.25	**0.05**
Movement Libras	360	90	90	0	7	1.5	**0.22**	16.77	2.58	**0.22**
Nursery	12960	8	0	8	5	2,160.0	1.50	3.23	5.42	**0.44**
Optdigits	5620	64	64	0	4	1.5	**0.53**	94.52	14.56	**0.53**
Page-blocks	5472	10	10	0	5	175.5	0.23	1.05	2.76	**0.11**
Penbased	10992	16	16	0	10	1.1	0.69	9.82	17.58	**0.41**
Satimage	6435	36	36	0	6	2.5	0.55	4.84	13.00	**0.43**
Segment	2310	19	19	0	7	1.0	0.44	10.25	2.70	**0.12**
Shuttle	5780	9	9	0	7	4,558.0	0.23	9.00	4.76	**0.11**
Texture	5500	40	40	0	5	5.0	0.37	46.26	11.65	**0.22**
Vehicle	846	18	18	0	4	1.1	0.13	5.28	0.61	**0.10**
Vehicle2	846	18	18	0	2	2.9	0.08	2.36	0.29	**0.04**
Vowel	990	13	13	0	11	1.0	0.33	36.51	2.07	**0.20**
Yeast	1484	8	8	0	10	92.6	0.21	15.53	1.90	**0.18**
Mean							0.48	14.78	6.81	**0.33**
Win/tie/loss							2/7/16	0/0/25	0/0/25	**16/7/2**
Avg. ranks							1.79	3.72	3.20	**1.29**

Source: The author.

APPENDIX B – COMPARISON WITH BENCHMARK MULTI-CLASS CLASSIFIER

Table 18 – Accuracy performance (in %) of MODES, DBDES, and Random Forest. Best result for each database in bold. The last rows represent the mean performance across all databases, the number of wins, ties and losses achieved by each technique, the average rankings and the $p - value$ for Wilcoxon Signed Ranking Test.

Dataset	DBDES ParzenDD/MAX	MODES MSTDD/DTs	Random Forest
Automobile	73.53	78.69	**86.22**
Car	89.99	84.84	**97.80**
Cleveland	47.76	53.88	**57.17**
Dermatology	93.86	92.74	**97.24**
Ecoli	78.88	77.69	**84.53**
Flare	65.20	69.61	**73.36**
Glass	57.01	71.54	**77.59**
Glass1	72.46	**81.33**	81.33
Glass6	95.32	95.32	**96.28**
Led7digit	**73.00**	72.20	70.20
Letter	93.99	90.52	**96.37**
Lymphography	72.30	74.32	**83.06**
Movement Libras	86.11	**86.67**	80.83
Nursery	79.76	74.58	**98.34**
Optdigits	**98.15**	98.02	97.78
Page-blocks	93.95	95.01	**97.37**
Penbased	99.25	**99.27**	99.13
Satimage	89.34	90.35	**91.66**
Segment	92.68	95.37	**98.05**
Shuttle	99.24	99.42	**99.71**
Texture	98.38	**98.53**	98.24
Vehicle	71.99	70.45	**74.47**
Vehicle2	96.57	97.64	**98.58**
Vowel	97.58	**98.48**	96.87
Yeast	58.42	54.78	**63.00**
Mean	0.8296	0.8405	0.8781
Win/tie/loss	2/0/23	4/1/20	18/1/6
Avg. ranks	2.33	2.15	1.52
Wilcoxon	-	0.0000	0.0000

Source: The author.

Table 19 – Kappa performance of MODES, DBDES, and Random Forest. Best result for each database in bold. The last rows represent the mean performance across all databases, the number of wins, ties and losses achieved by each technique, the average rankings and the $p-value$ for Wilcoxon Signed Ranking Test.

Dataset	DBDES ParzenDD/MAX	MODES MSTDD/DTs	Random Forest
Automobile	0.6610	0.6460	**0.8198**
Car	0.7811	0.6808	**0.9523**
Cleveland	0.2467	**0.2996**	0.2711
Dermatology	0.9227	0.9086	**0.9653**
Ecoli	0.7079	0.6903	**0.7845**
Flare	0.5626	0.6146	**0.6586**
Glass	0.4157	0.6128	**0.6901**
Glass1	0.3860	**0.5810**	0.5784
Glass6	0.7911	0.7855	**0.8262**
Led7digit	**0.6997**	0.6909	0.6687
Letter	0.9375	0.9014	**0.9623**
Lymphography	0.4715	0.4977	**0.6696**
Movement Libras	0.8273	**0.8434**	0.7753
Nursery	0.7041	0.6315	**0.9756**
Optdigits	**0.9750**	0.9733	0.9699
Page-blocks	0.7303	0.7564	**0.8566**
Penbased	0.9917	**0.9919**	0.9903
Satimage	0.8693	0.8812	**0.8966**
Segment	0.9146	0.9460	**0.9773**
Shuttle	0.9788	0.9460	**0.9918**
Texture	0.9768	**0.9789**	0.9746
Vehicle	0.6264	0.6060	**0.6596**
Vehicle2	0.9128	0.9372	**0.9624**
Vowel	0.9733	**0.9833**	0.9656
Yeast	0.4634	0.4221	**0.5173**
Mean	0.7411	0.7522	0.8144
Win/tie/loss	2/0/23	6/0/19	17/0/8
Avg. ranks	2.21	2.22	1.58
Wilcoxon	-	0.0000	0.0000

Source: The author.