Applications of Autonomous Learning Multi Model Systems to Binary Classification on Imbalanced Datasets

Rodrigo Saragoça Boal Ventura

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisor(s): Prof. João Miguel da Costa Sousa
Prof. Susana Margarida da Silva Vieira

Examination Committee

Chairperson: Prof. Carlos Baptista Cardeira
Supervisor: Prof. Susana Margarida da Silva Vieira
Member of the Committee: Prof. José Luís Valente de Oliveira

December 2021

Resumo

Dados desbalanceados representam um dos desafios mais complexos para os métodos de aprendizagem automática. Apesar da prevalência deste tipo de dados em diversas áreas, os métodos tradicionais geralmente apresentam uma clara preferência pelas classes sobre-representadas, apresentando um fraco desempenho na deteção das classes sub-representadas.

Nesta tese, o desafio da classificação de dados desbalanceados é abordado usando o classificador ALMMo-0, um sistema baseado em regras *fuzzy* do tipo AnYa que é não-iterativo, não-paramétrico e totalmente autónomo. Todos os testes são conduzidos usando versões desbalanceadas de dados de referência.

Em primeiro lugar, o desempenho dos classificadores ALMMo-0 é comparado aos desempenhos de métodos de referência. Especificamente, o desempenho é comparado com métodos de classificação tradicionais, assim como com outros métodos baseados em regras *fuzzy*.

Em segundo lugar, partindo dos resultados obtidos com o ALMMo-0 original, duas modificações são propostas, com o propósito de melhorar a deteção da classe minoritária: um classificador de classe única que não implica a otimização de qualquer parâmetro adicional, e uma estratégia de ponderação da confiança relativamente a cada classe, que ajusta os pesos de cada classe utilizando otimização Bayesiana.

Os resultados obtidos para as modificações propostas sugerem uma vantagem significativa relativamente ao ALMMo-0 original no que diz respeito à deteção da classe minoritária. Em particular, os resultados mostram que o classificador de classe única apresenta um desempenho superior em desbalanceamentos elevados, enquanto que a estratégia de ponderação das classes apresenta melhores desempenhos em desbalancemantos moderados e elevados, dependendo da métrica escolhida para a otimização dos pesos.

Palavras-chave: Classificadores ALMMo-0, Dados Desbalanceados, Classificação Binária, Classificadores de Classe Única, Optimização Bayesiana

Abstract

Imbalanced datasets present one of the most complex challenges for modern machine learning classification methods. Furthermore, despite imbalanced datasets being common in many different and relevant areas, traditional classification methods often show a large bias towards the overrepresented classes, and often show a poor prediction performance on the underrepresented classes.

In this thesis, the problem of imbalanced datasets is approached using the Zero-Order Autonomous Learning Multi-Model (ALMMo-0) classifier, an AnYa-type fuzzy rule-based system, which is non-iterative, non-parametric and fully autonomous. All the tests are performed on imbalanced versions of benchmark datasets.

Firstly, the performance of ALMMo-0 classifiers is compared to benchmark classification methods. In particular, the performance is compared to traditional classification methods, as well as other fuzzy rule-based systems, and is shown to out-perform the benchmark methods on highly imbalanced datasets.

Secondly, based on the empirical results obtained with the original ALMMo-0 algorithm, two modifications are proposed that seek to improve the minority class detection: a one class classifier adaptation of the original ALMMo-0 that requires no extra parameters to be estimated, and a weighted class confidence strategy that assigns different weights to each class and optimizes the weight values using Bayesian optimization.

The proposed methods are shown to outperform the original ALMMo-0 regarding the minority class prediction performance. In particular, the one class classifier is shown to outperform for high imbalances, while the weighted class confidence strategy is shown to outperform for low and high imbalances, depending on the cost function chosen for for the weight optimization.

Keywords: ALMMo-0 Classifier, Imbalanced Datasets, Binary Classification, One Class Classification, Bayesian Optimization

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

η	Utility.
λ	Rule Activation.
μ	Mean.
Ω	Learning Rate.
σ^2	Variance.
θ	Cloud.

Roman symbols

A	Consequent parameters.
B	Birth iteration.
C	Co-variance matrix.
D	Density.
f	Focal point.
K	Iteration number.
M	Number of cloud members.
N	Number of samples analyzed.
R	Number of rules.
r	Cloud radius.
u	Extended input.
X	Average scalar product.
x	Sample.
y	Output.

Subscripts

0	Initialization value.
j	Rule index.
k	Discrete time index.
m	Number of features.

Superscripts

i	Model index.
T	Transpose.

Chapter 1

Introduction

1.1 Motivation

The field of machine learning, and computational intelligence in general, have seen massive relevance increase in the last decades [1]. Recent advancements have not only produced new learning systems that can perform increasingly complex tasks, but have simultaneously provided both the researching community and the interested public the tools to develop their own models in an increasingly seamless and more intuitive manner [2].

In recent years, many of the more well-known advancements in learning systems have often been related to breakthroughs in different fields such as image processing [3], video processing [4], and speech synthesizing [5], which have allowed for many promising results for a wide range of practical applications.

One particular type of challenge that is still incredibly relevant in the field of machine learning are classification tasks on highly imbalanced datasets [6]. Imbalanced datasets are classification datasets where some classes are severely underrepresented compared to the remaining classes. As such, they pose a very challenging task to many traditional classification methods, as these often completely disregard the minority classes, which are usually the classes of most interest [7].

Furthermore, the problem of imbalanced datasets is particularly relevant, as such datasets occur in a wide variety of areas and applications [8]. Examples include disease diagnosis [9], fraud detection [10], cybersecurity [11] and image recognition [12]. Therefore, not only is the problem of imbalanced datasets very prevalent, but it also occurs in crucial areas of high relevance and interest.

Attending to the high interest in this particular issue, a lot of attention has been devoted to develop new approaches and strategies that help reducing the effects of high class imbalances. As such, a wide variety of methods have been proposed with different degrees of success and applicability [13].

However, although many of the state of the art methods for addressing high class imbalances have achieved better prediction performances on such datasets, most of these methods still yield black box models that are not explainable in human terms. This problem, known as interpretability [14], is a very pertinent and ongoing issue in machine learning, that refers to obtaining models that not only achieve good prediction performances, but that are also interpretable by a human expert that can understand the model´s reasoning to decide on certain class.

Another problem that affects many of the traditional classification methods is concerned with streaming data environments [15], where online learning and adapting to constantly changing (non-stationary) environments is required. This is due to the complexity of some of the state of the art classification methods that causes the training and prediction times to often be unacceptable for such streaming data applications [16].

Attending to these commons issues that affect traditional, as well as state of the art classification methods, this thesis approaches the problem of imbalanced datasets from the perspective of Evolving Fuzzy Inference Systems (EFIS) [17]. EFIS are fuzzy inference systems (FIS) that incorporate mechanism that allow for adpating to abrupt changes in the steaming data, while remaining computationally lightweight in terms of both memory and processing requirements [18]. Therefore, as fuzzy rule based (FRB) systems, EFIS are much more interpretable than traditional machine learning methods, as they are based on fuzzy logic [19]. Furthermore, they are also well suited for the growing challenge of streaming data environments.

In 2012, a new type of EFIS architecture was introduced [20], using AnYa type fuzzy rules and the principles first introduced with the Empirical Data Analysis (EDA) framework [21]. Contrarily to traditional FRB systems, such as the Mamdani and Takagi-Sugeno fuzzy systems, AnYa type fuzzy rules introduced a significantly simplified alternative to define the antecedent part of the rules, by replacing the parametric membership functions with non-parametric data clouds, that model local regions of the data space. The confidence scores of each rule for a given sample are obtained using the concept of data density [21], and each sample belongs to all clouds with different degrees.

Attending to the non-parametric nature of the rule antecedents, AnYa type systems are usually less computationally expensive to train than other FRB systems, making them very adequate to online learning tasks and streaming data applications [22]. Despite this being the original task for which AnYa type systems were introduced, they have since been successfully applied to static classification and regression problems with promising results.

In 2017, two new AnYa type FRB systems were introduced: the Autonomous Learning Multiple Model (ALMMo) System [23], and the 0-Order Autonomous Learning Multiple Model Classifier (ALMMo-0) Classifier [24]. These new systems were introduced as Autonomous Learning Systems (ALS), being specially designed to tackle dynamic problems such as online learning and streaming data by being lightweight in both memory requirement and processing power.

Ever since their introduction, both ALMMo systems and ALMMo-0 classifiers have been applied to different types of problems and used as the base for more complex, DL inspired architectures, as recently as in 2021 [25].

In this thesis, the 0-Order Autonomous Learning Multi Model (ALMMo-0) classifier, is used to approach the problem of highly imbalanced datasets. The ALMMo-0 classifier is an AnYa-type FIS, meaning that it is non-parametric and lightweight when compared to traditional fuzzy systems, as its antecedents are cloud based. Furthermore, they are also non-iterative and fully autonomous, making them ideal for streaming data environments. Despite its simplicity, ALMMo-0 classifiers have been shown to outperform benchmark classification methods, as well as traditional fuzzy classifiers. More importantly for the topic of this thesis, ALMMo-0 classifiers model each class independently, reducing the bias towards the underrepresented classes

In [26], ALMMo-0 classifiers were used to classify different heart disorders using sound data. The main goal of the proposed approach was to build highly accurate models, while also providing interpretable and explainable rules that may be used by experts to better diagnose hear disorders. Results have shown that ALMMo-0 classfiers were able to obtain better results than state-of-the-art methods, and also that the performance of the original algorithm could be slightly improved by adding a standardization and normalization pre-processing layer.

As mentioned, the simplicity of ALMMo-0 classifiers makes them particularly suitable to be used as the base learners in ensemble learning methods, and also allows the creation of more lightweight and interpretable deep learning architectures.

In [27], the Multi-Layer Multi-Model Images Classifier (MICE) ensemble was first introduced, as a fast deep learning network for handwriting recognition. The MICE ensemble was proposed in order to allow ALMMo-0 classifiers to be applied to more complex tasks such as image recognition, and consists of multiple ALMMo-0 base learners trained in parallel. The ensemble prediction is accomplished using a winner-takes-all voting strategy that takes each one of the learners predictions. The MICE ensemble also introduced the first ALMMo deep learning architecture, by adding a pre-processing layer that extracts GIST [28] and HOG [29] features from the raw image input. Each one of the ensemble base classifiers is then trained using a unique subset of the extracted features. Shortly after, MICE ensembles were modified in order to perform better when multiple classes have high and similar confidence scores [30].

1.2 Objectives

This thesis seeks to approach the problem of binary classification on imbalanced datasets using the ALMMo-0 classifier, by studying its performance and how the class imbalanced affects the minority class detection, and also by proposing modifications to the original algorithm that seek to diminish the effects of high class imbalances.

This thesis proposes the following objectives:

- Study the effect of imbalanced datasets on the minority class prediction performance of ALMMo-0 classifiers
- Propose and implement modifications to the original ALMMo-0 classifier with the goal of mitigating the class imbalance effect
- Assess the performance of ALMMo-0 classifiers relative to benchmark classification methods
- Assess the performance of the proposed modifications relative to the original ALMMo-0 classifier

1.3 Contributions

The main contributions of the work hereby presented are:

- Provides a comprehensive analysis of the effect of different degrees of class imbalance on the performance of ALMMo-0 classifiers and shows the clear advantage that ALMMo-0 models have when compared to commonly used benchmark classification methods
- Introduces a one class classification adaption of the original ALMMo-0 algorithm that is very lightweight and requires no extra parameters to be estimated, while clearly outperforming the original algorithm for highly imbalanced datasets
- Introduces a weighted class confidence modification of the original ALMMo-0 algorithm that compensates for the bias towards the majority class by assigning different weights to the majority and minority class sub-models. Furthermore, the proposed method allows for the choice of different metrics for the class weights optimization procedure, allowing for different models to be obtained, that have different characteristics and are applicable to different tasks with different specific requirements

1.4 Thesis Outline

The work hereby presented is organized in six chapters:

- **Chapter 1 - Introduction:** This chapter introduces the topics of the thesis and how they are presented and organized in this report
- **Chapter 2 - Autonomous Learning Multi-Model Systems:** This chapter provides a brief overview of the essential concepts of fuzzy logic and traditional fuzzy inference systems, and presents a thorough description of Autonomous Learning Multi-Model systems as a new type of fuzzy rule based systems
- **Chapter 3 - Imbalanced Datasets:** This chapter discusses the issues and challenges posed by imbalanced datasets for traditional classification methods. Furthermore, this chapter also presents a brief overview of relevant classification performance measures, as well as different approaches that have been proposed in order to mitigate the effects of class imbalances
- **Chapter 4 - Proposed Methods:** This chapter discusses the impact of imbalanced datasets on the performance of ALMMo-0 classifiers, and suggests modifications to the original algorithm that seek to improve the minority class prediction performance. Two modifications are proposed: a one class classifier adaptation of the original ALMMo-0, and a weighted class confidence approach that replaces the original winner-takes-all classification criteria
- **Chapter 5 - Results:** This chapter presents the results obtained for imbalanced versions of benchmark classification datasets. These results are used to compare the performance of the original ALMMo-0 classifier with benchmark methods, as well as with the proposed methods. Furthermore, this chapter presents a thorough discussion of the different classification measures and how the final application of the models influences the model selection procedure
- **Chapter 6 - Conclusions:** This chapter presents the relevant conclusions that can be taken from the work presented in this thesis, and also discusses not only what was achieved, but also suggests further work that may be followed based on the results obtained in this thesis

Chapter 2

Autonomous Learning Multi-Model Systems

2.1 Fuzzy Logic

Contrarily to classic (Boolean) logic, in which the membership criteria of a given object to a class is either true (1) or false (0), fuzzy logic [31] introduced the concept of membership degree, in which the object membership to a class is a real number between 0 and 1. Therefore, in fuzzy logic, objects can belong to different classes with different membership degrees.

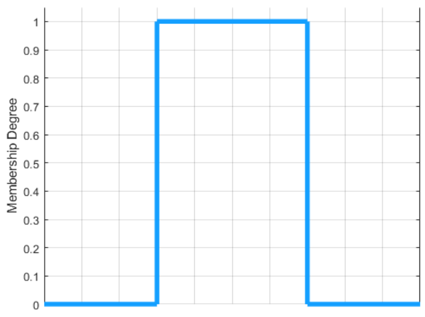
One of the main advantages of fuzzy logic over classic logic is that it acts more closely to the nature of human concepts and thoughts, which tend to be abstract and even imprecise. One clear example is the usage of linguistic terms in communication, which are more often than not vague and imprecise. However, human communication is still successful despite these linguistic terms. Furthermore, one may argue that this vagueness actually improves the communication process, since most linguistic terms are easily interpretable and understandable by humans.

Fuzzy sets are defined as a collection of objects x, defined in some universe of discourse X, such that each element of X is mapped to a membership degree, as defined in 2.1, Where $\mu_A(x)$ is the membership function that characterizes fuzzy set A.

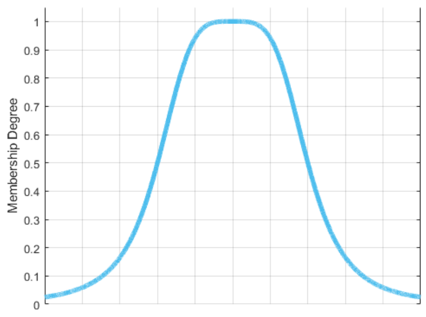
$$A = \{(x, \mu_A(x)) \| x \in X\} \tag{2.1}$$

In contrast to fuzzy sets, classical sets have crisp boundaries, where each element of X either belongs or not to A. In this case, the classical set is characterized by a characteristic function. Figure 2.1 illustrates the difference between characteristic functions and membership functions.

Different membership functions exist, depending on their shape and parameters. Some of the more commonly used one-dimensional membership functions are the Triangular, Trapezoidal, Gaussian and Generalized Bell. Two dimensional and higher dimensional MFs can be defined as AND or OR aggregation of its constituent uni dimensional MFs.



(a) Characteristic Function



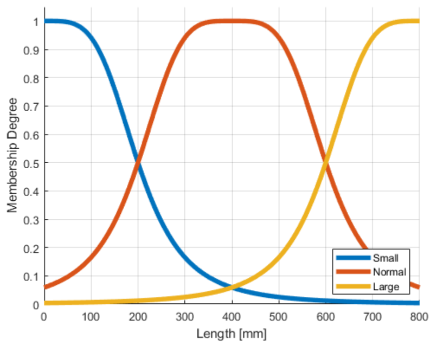
(b) Membership Function

Figure 2.1: Example of a classic set defined by a characteristic function and a fuzzy set defined by a generalized bell membership function

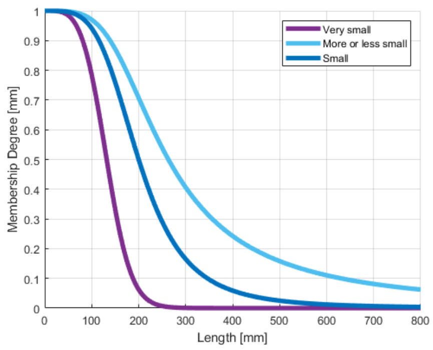
Generally speaking, when the universe of discourse X is a continuous space, X is partitioned into several fuzzy sets whose membership functions cover X in a more or less uniform manner, as shown in Figure 2.2. Usually, each fuzzy set is associated to a certain linguistic value that is more often than not adjectives that appear in daily linguistic usage.

In Figure 2.2(a), the universe of discourse for the length variable (measured in millimeters) is partitioned into three fuzzy sets, associated to the linguistic values ”small”, ”normal” and ”large”, and characterized by the membership functions μ_{small}, μ_{normal} and μ_{large}. The membership functions are also referred to as semantic rules when used in the context of linguistic variables.

Furthermore, as illustrated in Figure 2.2(b), the semantic rules may be modified using linguistic modifiers that alter the membership function shape and the associated linguistic value. In Figure 2.2(b), the semantic rule μ_{small} is modified using a concentration operator to get a new fuzzy set associated to the linguistic term ”very small”, and also modified using a dilation operator to get a new fuzzy set associated to the linguistic term ”more or less small”.



(a) Fuzzy sets partitioning the universe of discourse



(b) Effect of using different linguistic modifiers

Figure 2.2: Fuzzy sets partitioning the length universe of discourse into ”small”, ”normal” and ”large” (a), and the effect of using different linguistic modifiers to obtain the ”very small” and ”more or less small” from the ”small” fuzzy set (b)

One very important fuzzy set operation that is particularly relevant when modelling a system using fuzzy logic is the projection operation. This projection operation is used to project a higher dimensional membership function onto the different problem dimensions. As an example, a two dimensional membership function defined in $X \times Y$ results in two one dimensional membership functions, defined in X and Y respectively.

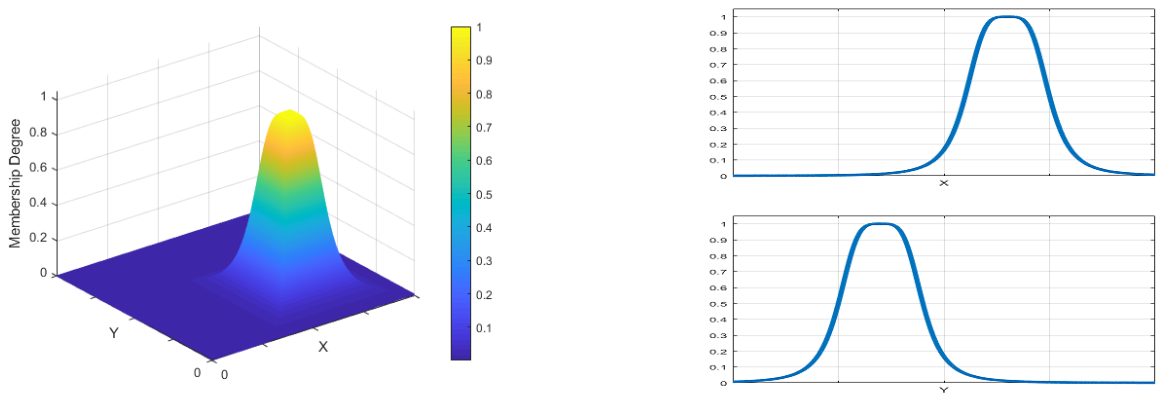


Figure 2.3: Two dimensional membership function and respective projections onto X and Y.

Figure 2.3 illustrates the projections of a two-dimensional Gaussian membership function onto X and Y, resulting in two one-dimensional Gaussian membership functions. As will be discussed in the next sections, if one obtains a n dimensional membership function that characterizes a group of objects with similar characteristic, then one can describe the elements included in the group in terms of the values of each one of the n variables of the problem.

2.2 Fuzzy Inference Systems

Fuzzy reasoning is an inference procedure that derives conclusions from a set of fuzzy rules. Fuzzy rules (also known as fuzzy implications) are IF-THEN statements that describe the relation between the antecedents (IF part) and the consequents (THEN part).

Fuzzy rules may have multiple antecedents. In 2.2, the structure for a two antecedent rule is shown, where A, B and C are fuzzy sets defined in X, Y and Z, respectively.

$$\textbf{IF } x \textit{ is } A \textbf{ AND } y \textit{ is } B \textbf{ THEN } z \textit{ is } C \tag{2.2}$$

The inputs x and y may either be either fuzzy or crisp, in which case they are treated as a fuzzy singletons. The output fuzzy set C may be defuzzified to obtain a crisp value, which is often required for most practical applications.

Each part of the antecedent generates a membership degree to the respective membership function, denoted as ω_1 and ω_2, and defined as $\mu_A(x)$ and $\mu_B(y)$, respectively. Since the two antecedents are connected using the AND operator, the overall firing strength (or activation) of the rule is defined as $\omega_1 \wedge \omega_1$, meaning that its value is equal to the minimum of the two membership degrees.

A single fuzzy rule allows for the modeling of a local region of the data space. Therefore, if multiple rules are obtained for different regions of the data space, and aggregated into a single modelling architecture that provides a reasoning mechanism based on the multiple rules, one can obtain a complete model of the system.

This is precisely the idea of fuzzy inference systems, which combine the knowledge of multiple rules to derive a conclusion from the given input. This is accomplished using an aggregation mechanism (such as the union operation) that combines the individual rules fuzzy outputs into a single fuzzy output. This basic architecture of a FIS is illustrated in Figure 2.4.

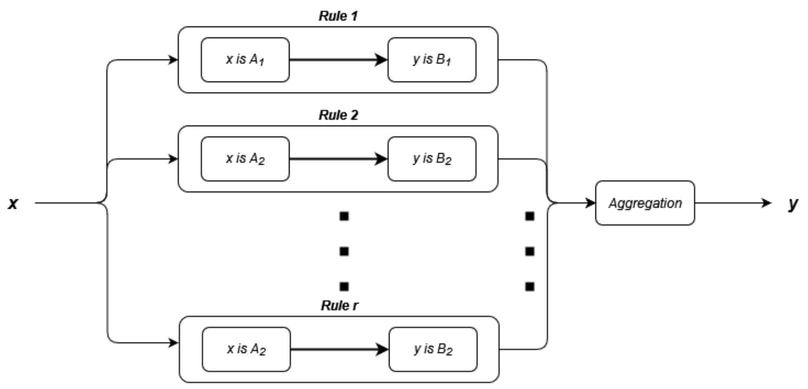


Figure 2.4: General architecture of a fuzzy inference system

FIS are general functions approximators, similar to neural networks, and their mapping accuracy increases as the number of rules increases. However, a large number of rules may diminish the model interpretability, which would defeat one of the main perks of using a fuzzy rule based model.

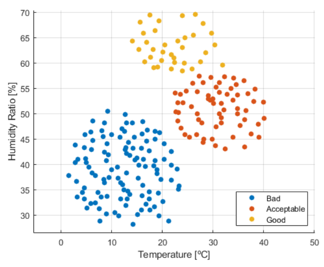
While interpretability is a somewhat vague concept and it is not trivial to measure it, one commonly used metric for fuzzy systems is the FRDP metric, defined as shown in 2.3, where NoR is the number of rules and NoS is the number of samples used to train the fuzzy model.

$$FRDP = \frac{NoR}{NoS} \tag{2.3}$$

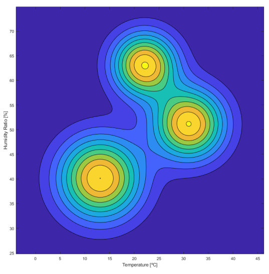
Having defined the general architecture of FIS, as well as the inner workings of its constituent fuzzy rules, it remains to be answered how one would obtain the fuzzy sets that constitute the rules. In other words, it remains to be explained how the fuzzy inference system and its fuzzy rules are obtained from the training data.

This procedure, known as fuzzy modelling, may be sub-divided into two steps. First, the data space must be partitioned and the training samples must be grouped into groups with similar characteristics, characterized by n dimensional membership functions. This step will be further discussed in 2.3. The second step is obtaining the one dimensional fuzzy sets from the n dimensional membership functions.

As already discussed, each n dimensional membership function describes a group of objects with similar characteristics, and can be projected onto each one of the n dimensions using the projection operation. This is the procedure that is usually used to obtain the individual rules that compose a fuzzy inference system.



(a) Samples grouped by the respective growth conditions

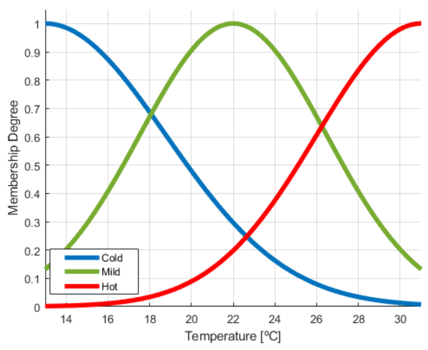


(b) Membership functions describing the different sample groups

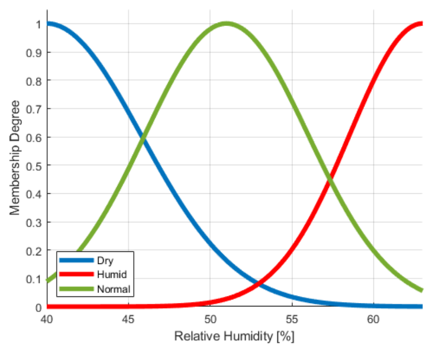
Figure 2.5: Example of a two dimensional problem, describing the growth conditions for a plant in terms of temperature and humidity (a), and the respective membership functions (b)

In order to illustrate the inner working of this procedure, let´s consider the problem of modelling the growth conditions of a plant in terms of the environment temperature and humidity ratio. Figure 2.5(a) shows the training samples already grouped according to the growth conditions they provide, as ”bad”, ”acceptable” or ”good”. Figure 2.5(b) shows the two-dimensional membership functions that define each one of the three groups.

Since there are three distinct groups, and since each one is projected onto the temperature and humidity dimensions, a total of six fuzzy sets are obtained, three for temperature, and three for the humidity. Figure 2.6 shows the resultant one-dimensional membership functions.



(a) Temperature membership functions



(b) Humidity membership functions

Figure 2.6: Projections of the two dimensional fuzzy sets describing the growth conditions onto the temperature (a) and humidity (b) universes of discourse

Attending to what was discussed in 2.1 about linguistic variables, each fuzzy set is associated to a linguistic value. Regarding the temperature, the fuzzy sets correspond to the linguistic values ”cold”, ”mild” and ”hot”. Regarding the humidity, the fuzzy sets correspond to the linguistic values ”dry”, ”normal” and ”humid”. Each group corresponds to a fuzzy rule, meaning that there are three rules in the resultant fuzzy inference system. These rules are as shown in 2.4.

$$\begin{cases} \textbf{IF } \textit{Temperature is Cold } \textbf{AND } \textit{Humidity is Dry } \textbf{THEN } \textit{Condition is Bad} \\ \textbf{IF } \textit{Temperature is Hot } \textbf{AND } \textit{Humidity is Normal } \textbf{THEN } \textit{Condition is Acceptable} \\ \textbf{IF } \textit{Temperature is Mild } \textbf{AND } \textit{Humidity is Humid } \textbf{THEN } \textit{Condition is Good} \end{cases} \tag{2.4}$$

Having established the basic mechanisms of fuzzy inference systems, the remainder of this section presents two widely known traditional fuzzy inference systems (the Mamdani and Takagi-Sugeno fuzzy systems), as well as the recently introduced AnYa type fuzzy inference systems, which are the basis of ALMMo-0 classifiers and ALMMo systems.

2.2.1 Mamdani Fuzzy Inference Systems

Mandani fuzzy systems were first introduced as an attempt to create a control system for a steam engine and boiler combination. Mamdani fuzzy systems have fuzzy inputs and a fuzzy output. The inner workings of a Mamdani fuzzy system are summarized by the following Figure 2.7, where the five main stages are shown.

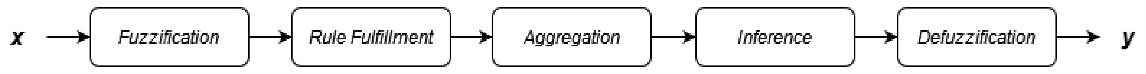


Figure 2.7: Main stages of a Mamdani fuzzy system

The first stage is the fuzzification step, in which the membership degrees of the input are computed for each membership function of the rule antecedents. Then, the fulfillment degree of each rule is computed, and the individual rule outputs are obtained using the previously described inference mechanism. The individual rule fuzzy outputs are then combined using an appropriate aggregation operator, such as the union operator. Finally, the aggregated output fuzzy set is deffuzified, in order to obtain a crisp output. Several deffuzification methods exist, such as centroid of area (CoA) and mean of maxima (MoM).

2.2.2 Takagi-Sugeno Fuzzy Inference Systems

Takagi-Sugeno fuzzy systems are quite simillar to Mamdani fuzzy systems and have a very similar architecture. The key difference between the two systems is that Takagi-Sugeno fuzzy systems have crisp consequents, which may be defined by a constant or a function of the input values.

In equation 2.5, the structure of zero-order rule is shown, where z is the individual rule output and c is a constant.

$$\textbf{IF } x \sim A \textbf{ AND } y \sim B \textbf{ THEN } z = c \tag{2.5}$$

More commonly, linear (first order) functions of the input variables are used to defined the consequent of a Takagi-Sugeno fuzzy system. The structure of a first-order rule is shown in 2.6, where a, b and c are constants.

$$\textbf{IF } x \sim A \textbf{ AND } y \sim B \textbf{ THEN } z = ax + by + c \tag{2.6}$$

Therefore, a first order Takagi-Sugeno fuzzy system rules approximates the local regions of the data space using linear functions. For higher dimensional dimensional systems, the same mechanism applies, using hyper planes defined by the rule consequent parameters.

In order to obtain the overall system output, an aggregation operation is used, as it was the case for Mamdani fuzzy systems. The overall output is defined as in 2.7, where R is the number of fuzzy rules, ω_i are the rules activations, and y_i are the individual rule outputs.

$$y = \sum_{i=1}^{R} \omega_i y_i \tag{2.7}$$

2.2.3 AnYa-type Fuzzy Inference Systems

One of the most recent advancements in fuzzy systems are AnYa-type fuzzy rule based systems, which are the basis of ALMMo-0 classifiers and ALMMo systems. The structure of AnYa-type fuzzy systems is quite similar to the one described for the Takagi-Sugeno FIS, since AnYa-tupe rules have crisp inputs and outputs.

The key difference between AnYa-type fuzzy systems and traditional fuzzy systems is the method used to define the antecedents. Contrarily to traditional fuzzy systems, which define the antecedents using parametric membership functions which are obtained by projecting the data clusters, AnYa-type fuzzy systems use non-parametric data clouds to define the antecedents.

Besides the antecedent definition method, the general structure of AnYa-type rules is essentially the same to the one discussed for Takagi-Sugeno FIS, as shown in 2.8.

$$\textbf{IF } x \sim \Theta \textbf{ THEN } y = f(x) \tag{2.8}$$

Where Θ is a data cloud, and f is a generic function of the individual inputs. For the ALMMo system, which is a regressor, the rule structure is exactly the same as the one shown in 2.8, where f is defined as a linear combination of the individual inputs, similarly to first order Takagi-Sugeno fuzzy systems. The overall output is also obtained in a similar way, as shown in 2.7.

However, as it will be discussed, ALMMo-0 classifiers are not zero order in the sense that they do not output a scalar defined by the consequent parameters, and this would corresponds to a zero order ALMMo regressor. Instead, ALMMo-0 classifiers do not have any consequent parameters since, as a classifier, the output is the label corresponding to the closest data cloud.

2.3 Data Space Partitioning

Having described the basics of fuzzy logic and fuzzy reasoning, as well as the mechanisms and architecture of fuzzy inference systems, the question of how to determine the membership functions remains to be answered. In other words, it remains to be answered how to partition the data space and group samples with similar characteristics.

In traditional fuzzy systems, such as the Mamdani and Tagaki-Suegno inference systems, the membership functions are obtained by projecting data clusters onto the input variables and fitting the membership function parameters to the cluster projections, as already discussed. A broad overview of the different clustering methods is presented in 2.3.1.

As mentioned, AnYa-type fuzzy systems introduced the concept of non-parametric data clouds as the method to define the antecedents. This approach offers obvious advantages, since there are no antecedent parameters to optimize, in contrast to traditional fuzzy systems. A thorough description of data clouds and the key differences to data clustering is presented in 2.3.2.

2.3.1 Data Clustering

Generally speaking, data clustering is the task of dividing data into groups of simillar objects. The objects that belong to a certain cluster share a higher degree of similarity than they do with objects that do not belong to their cluster. Clustering simplifies the data space, as few clusters can describe a large amount of objects, with a consequent loss of finer details [32].

Data clustering is used in several relevant applications, such as image segmentation, genome sequencing, object recognition, character recognition, data mining and data compression. Furthermore, data clustering is an unsupervised learning approach that is be used to extract hidden patters from unlabeled data, and it is often used in semi-supervised learning approaches to label unknown samples.

Attending to the large amount of applications of clustering, several different methods have been introduced throughout the years, and different categorizations have proposed for the different clustering algorithms. Generally speaking, clustering algorithms can be divided into two major groups: hierarchical clustering and partitional clustering [33].

In hierarchical clustering methods [34], clusters are created by iteratively dividing the data space using either a top-down or a bottom-up approach. Agglomerative clustering uses a bottom-up approach, in which clusters are started using a single object and are then successively merged with other clusters to create larger and larger clusters. Divisive clustering follows a top-down approach where a cluster containing all the objects is initialized, and is then successively broken into smaller and smaller clusters.

Partitional clustering methods have a very different approach, in which the data objects are assigned to a specified number of clusters by minimizing some pertinent cost function that encodes some type of similarity criterion. Partitional clustering methods may be further divided into distance based, model based and density based methods.

Distance based clustering methods use some form of distance metric as the similarity criterion. Some of the most commonly used clustering methods such as K-Means [35] and Fuzzy C-Means [36] use the distance (typically the Euclidean distance) between the cluster centers and the training samples to iteratively update the cluster centers until some stopping criteria is met.

It is important to mention that in the mentioned methods the number of clusters must be specified beforehand, which in most tasks is unknown. Broadly speaking, the typical approach to address this issue is to use some type of clustering validity measure to assess the quality of the obtained clusters. Therefore, the optimal number of cluster may be found by using some search procedure to optimize the validity measure value. Many different validity measures have been introduced throughout the years, using different quality notions, such as the clusters volume and their compactness, as well as how well the clusters are separated.

Density based clustering methods [37] divide the data space using some form of density notion. Density, in this context, is defined as the number of samples in a given local region of the data space. Regions with a higher concentration of samples have higher densities, and vice versa. Using this notion of density, density-based clusters are separated from each other by a contiguous low density region. Therefore, density-based clusters can have arbitrary shapes that may better model the local regions of the data space.

2.3.2 Data Clouds

Data clouds were introduced as a non-parametric alternative to define the antecedents of fuzzy rules. In contrast to the previously mentioned method of fitting parametric membership functions to cluster projections, data clouds do not require the assumption of some membership function structure, as the membership degree between an object and a cloud is defined as a data density distribution.

Data clouds are characterized by a focal point, which corresponds to the mean of all the points used to define the cloud, and the variance, which corresponds to how spread apart the cloud points are. Furthermore, ALMMo-0 classifiers also define a cloud radius parameter, which is used as a similarity criterion during the training process to assess if a given sample is within a cloud´s range of influence.

Furthermore, data clouds are defined by simple iteratively updated parameters that naturally arise from the data, as thus no optimization procedure must be used to determine the parameters that would better fit the data, as was the case for the FCM algorithm. Another clear advantage of data clouds and density-based membership degrees is that the number of clusters must not be known beforehand. This is, as discussed before, one of the main reasons why ALMMo regressors and ALMMo-0 classifiers have consistently shown good performances on online learning tasks.

In the context of ALMMo regressors and ALMMo-0 classifiers, density is defined using the unimodal discrete density (UDD) metric, first introduced by the EDA framework. The unimodal density between a sample x and a cloud with focal point μ and variance σ^2 is computed as shown in 2.9.

$$D(x, \mu, \sigma) = \frac{1}{1 + \frac{x-\mu}{\sigma^2}} \tag{2.9}$$

Therefore, the unimodal density increases as the distance between the sample and the focal point decreases. For the same distance, the unimodal density is larger for a clouds with larger variance, as

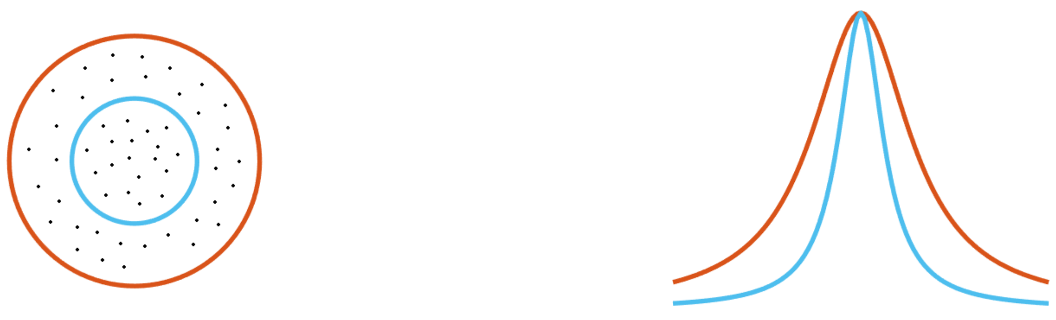

(a) Data clouds (b) Unimodal density functions

Figure 2.8: Example of two clouds with different variances and the respective unimodal density functions

the points are more further apart and the cloud range of influence is also larger. Figure 2.8 shows two clouds with different variances and the respective unimodal density functions.

Data clouds are used to model the antecedent parts of AnYa type fuzzy rules, in contrast with the Mamdani and TSK FRB systems that use parametric membership functions. Therefore, one clear advantage that AnYa type fuzzy rules show when compared to traditional FRB systems is that no parameters must be optimized to model the antecedents.

Similarly to the TSK fuzzy rules, AnYa type fuzzy rules have crisp outputs, which are usually either a constant (zero order) or a linear function of the different inputs. Therefore, the consequent parameters must still be estimated to correctly define the rule. In the case of the ALMMo regressor, this is accomplished using recursive least squares, similarly to first order TSK fuzzy rules.

2.4 ALMMo-0 Classifier

The 0-order Autonomous Learning Multiple Model (ALMMo-0) Classifier is a non-parametric, non-iterative and fully autonomous AnYa type fuzzy rule-based (FRB) multi-class classifier. The structure of ALMMo-0 classifiers is composed of one sub-model per class, each one being trained only on its class samples.

Each one of the class sub-models is based on multiple AnYa type fuzzy rules with one data cloud associated to each rule's antecedent. These rules assume the structure shown in 2.10, where x is a sample, f_j^i is the focal point associated to the j^{th} rule of the i^{th} class classifier, and $Label^i$ is the respective class label.

$$\textbf{IF } x \sim f_j^i \textbf{ THEN } Label^i \tag{2.10}$$

Furthermore, each sub-model is trained independently of each other since only its class samples are used to iteratively create the clouds, meaning that the cloud structure for each class sub-model is not affected by the cloud structures of the remaining sub-models.

Therefore, the fully trained ALMMo-0 classifier is composed of multiple FRB systems, each one with its set of AnYa type rules. When classifying a new sample, every rule in each one of the class-models will output a confidence score, denoted as λ_j^i.

The confidence score is a defined as a function of the distance between one data sample and one cloud focal point, as shown in 2.11. In the original article, the Euclidean distance is used, without any loss of generality.

$$\lambda_j^i = \exp -\frac{1}{2} \left\| x - f_j^i \right\|^2 \tag{2.11}$$

In order to assign a label to the sample, one must devise some form of rule based on the confidence scores of each sub-model. In ALMMo-0 Classifiers, the winner takes all strategy is used. First, the maximum confidence score of each class sub-model λ_{j*}^i is found, and then, the winner takes all principle is used and the class sub-model with the highest confidence score assigns its label:

$$\hat{y} = \underset{i=1,2,\ldots,L}{\arg\max}(\lambda_{j*}^i) \tag{2.12}$$

This classification strategy means that the data clouds effectively create Voronoi tessellation of the data space, dividing it into different sub-regions that belong to different classes. As mentioned, these data clouds are non parametric and no prior assumptions about the data are made.

Furthermore, as will be explained in the next section, the training algorithm that creates the clouds is non iterative and lightweight, meaning that ALMMo-0 Classifiers are particularly adequate for streaming data applications, even though no specific mechanisms to adapt to abrupt changes in the data are proposed, as is the case for ALMMo systems.

2.4.1 Training Algorithm

Let $\mathbf{x_k^i}$ be the k^{th} sample from the i^{th} class. The sample is first norm normalized, as shown in 2.13, effectively removing one degree of freedom from the data space and projecting the sample to a unit radius hyper sphere centered around the origin.

$$\mathbf{x_k^i} \leftarrow \frac{\mathbf{x_k^i}}{\|\mathbf{x_k^i}\|} \tag{2.13}$$

If the sample is the first sample of its class, $\mathbf{x_1^i}$, then the respective sub-model parameters are initialized using equations 2.14.

$$\begin{cases} N^i \leftarrow 1 \\ \mu^i \leftarrow x_1^i \\ X^i \leftarrow \|x_1^i\|^2 \end{cases} \tag{2.14}$$

Where N^i is the number of samples used to train the sub-model, μ^i is the sub-model mean, and X^i is the sub-model average scalar product. Then, the sample is used to create the first cloud of its class sub-model, using equations 2.15.

$$\begin{cases} R^i \leftarrow 1 \\ M_1^i \leftarrow 1 \\ f_1^i \leftarrow x_1^i \\ X_1^i \leftarrow \|x_1^i\|^2 \\ r_1^i \leftarrow r_0 \end{cases} \tag{2.15}$$

Where R^i is the total number of rules in the sub-model, M_1^i is the number of samples used to update the cloud, f_1^i is the cloud's focal point, X_1^i is the cloud's average scalar product, and r_1^i is the cloud´s radius. This radius effectively describes a confidence score threshold around the focal point and its value is not a problem specific parameter. In this thesis, the initial cloud radius r_0 is assumed to be $r_0 = \sqrt{2(1-\cos 15)}$, as specified in the original article.

If the newly arrived sample $\mathbf{x_k^i}$ is not the first sample of its class, the sub-model parameters N^i, μ_k^i, and X_k^i are recursively updated using equations 2.16.

$$\begin{cases} N^i \leftarrow N^i + 1 \\ \mu^i \leftarrow \frac{N^i-1}{N^i}\mu^i + \frac{x_k^i}{N^i} \\ X^i \leftarrow \frac{N^i-1}{N^i}X^i + \frac{1}{N^i}\|x_k^i\|^2 \end{cases} \tag{2.16}$$

The algorithm then checks for density anomalies in the sub-model, ie, regions of the data space where the unimodal density is either too high (high concentration of clouds) or too low (low concentration of clouds). This is accomplished by computing the unimodal density between the sample x_k^i and the sub-model mean μ^i, using equation 2.17, and comparing it to the unimodal densities between each cloud focal point in the sub-model f_j^i, computed using equation 2.18.

$$D(x_k^i, \mu^i) = \frac{1}{1 + \frac{\left\|x_k^i - \mu^i\right\|^2}{X^i - \left\|\mu^i\right\|^2}} \tag{2.17}$$

$$D(f_j^i, \mu^i) = \frac{1}{1 + \frac{\left\|f_j^i - \mu^i\right\|^2}{X^i - \left\|\mu^i\right\|^2}} \tag{2.18}$$

The maximum and minimum focal point densities are compared to the sample density, defining the first condition in the algorithm, shown in equation 2.19.

$$D(x_k^i, \mu^i) \; > \max_{j=1,2,\dots R^i}(D(f_j^i, \mu^i)) \; \vee \; D(x_k^i, \mu^i) \; < \min_{j=1,2,\dots R^i}(D(f_j^i, \mu^i)) \tag{2.19}$$

If Condition 1 is verified, then there is a density anomaly (either too high or too low) and a new cloud is created around the sample x_k^i, using equations 2.20.

$$\begin{cases} R^i \leftarrow R^i + 1 \\ M_j^i \leftarrow 1 \\ f_j^i \leftarrow x_k^i \\ X_j^i \leftarrow \left\|x_k^i\right\|^2 \\ r_j^i \leftarrow r_0 \end{cases} \tag{2.20}$$

Otherwise, if Condition 1 is not verified, then no density anomaly exists and a second condition, based on distance instead of density, is checked to assess if a nearby cloud already exists. First, the distances between the sample x_k^i and each focal point in the sub-model are computed, and then the nearest cloud focal point f_{j*}^i is found, using equation 2.21.

$$f_{j*}^i = \underset{j=1,2,\dots R^i}{\arg\min}\,(\left\|x_k^i - f_j^i\right\|) \tag{2.21}$$

If the distance to the nearest focal point f_{j*}^i is less than its radius r_{j*}^i, then the sample is considered to be close enough and the cloud is updated. This distance criteria is expressed by Condition 2, as shown in 2.22.

$$\left\|x_k^i - f_{j*}^i\right\| \leq r_{j*}^i \tag{2.22}$$

If Condition 2 is met, the nearest cloud parameters are updated using equations 2.23.

$$\begin{cases} M_{j*}^i \leftarrow M_{j*}^i + 1 \\ f_{j*}^i \leftarrow \frac{M_{j*}^i - 1}{M_{j*}^i} f_{j*}^i + \frac{x_k^i}{M_{j*}^i} \\ X_{j*}^i \leftarrow \frac{M_{j*}^i - 1}{M_{j*}^i} X_{j*}^i + \frac{1}{M_{j*}^i} \left\|x_k^i\right\|^2 \end{cases} \tag{2.23}$$

Otherwise, if Condition 2 is not met, a new cloud is created using equations 2.24.

$$\begin{cases} R^i \leftarrow R^i + 1 \\ M_j^i \leftarrow 1 \\ f_j^i \leftarrow x_k^i \\ X_j^i \leftarrow \left\| x_k^i \right\|^2 \\ r_j^i \leftarrow r_0 \end{cases} \tag{2.24}$$

The algorithm then proceeds to the next sample. The complete learning process is summarized below in Algorithm 1.

Algorithm 1 ALMMo-0 Classifier Training Algorithm

```
1:  for $c^i$ in $C$ do
2:      for $x_k^i$ in $c^i$ do
3:          Normalize $x_k^i$ using 2.13
4:          if $c^i$ is empty then
5:              Initialize sub-model using 2.14 and 2.15
6:          else
7:              Update sub-model parameters using 2.16
8:              Compute sample density using 2.17
9:              Compute focal densities using 2.18
10:             if Condition 1 (2.19) is met then
11:                 Create new cloud using 2.20
12:             else
13:                 Find nearest cloud using 2.21
14:                 if Condition 2 (2.22) is met then
15:                     Update nearest cloud using 2.23
16:                 else
17:                     Create new cloud using 2.24
18:                 end if
19:             end if
20:         end if
21:     end for
22: end for
```

2.5 ALMMo Systems

Autonomous Learning Multimodel (ALMMo) Systems are non-parametric and fully data-driven AnYa type fuzzy rule based (FRB) regressors, formulated specifically for data streaming environments by incorporating online data cloud quality monitoring, allowing the model to handle fast changes in the streaming data pattern.

ALMMo Systems are based on multiple AnYa type rules that are empirically extracted from the observed data, without making any prior assumptions concerting the data distribution. All the metaparemeters that form these rules are directly obtained from the data and are updated recursively, improving both memory and calculation efficiencies and further allowing the model to be trained online in streaming applications

ALMMo Systems are based on first order AnYa type rules, where the antecedents are defined by clouds and the consequents are defined by a first-order (linear) function, whose parameters are also recursively estimated using recursive least squares (RLS). These rules have the form shown in 2.25.

$$\textbf{IF}\ x\ \sim\ f_j\ \textbf{THEN}\ y_j = u^T A_j \tag{2.25}$$

Where x is the sample, f_j is the j^{th} rule focal point, y_j is the rule's output, u is the extended input vector $u = [1, x]^T$, and A is the rule´s consequent parameters vector.

The overall output of the system y is computed as the fuzzily weighted sum of the single outputs of each rule, as shown in 2.26.

$$y = \sum_{j=1}^{R} \lambda_j u^T A_j \tag{2.26}$$

Where λ_j is the j^{th} cloud (rule) activation level, which will is defined as the normalized unimodal density computed for each rule.

As regressors, ALMMo systems may be applied to regression tasks, as well as binary classification tasks. For the latter, the overall output y is thresholded at 0.5, allowing for binary class output.

Finally, the already mentioned cloud online monitoring mechanism makes ALMMo systems particularly applicable to streaming data tasks, such as time series forecasting and predictive control. In fact, many of the recent published work on ALMMo Systems has been focused on such tasks.

Nevertheless, ALMMo systems have shown interesting results in more usual classification tasks when compared to other state of the art methods. Attending to the scope of this thesis, ALMMo systems are analyzed purely as binary classifiers.

2.5.1 Training Algorithm

The algorithm starts by initializing the global parameters, using the first training sample x_1, as in equations 2.27.

$$\begin{cases} N \leftarrow 1 \\ \mu \leftarrow x_1 \\ X \leftarrow \|x_1\|^2 \end{cases} \tag{2.27}$$

Where N is the number of samples analyzed by the model, μ is the model global mean and X is the model global average scalar product. Then, the first cloud is created and initialized using 2.28

$$\begin{cases} R \leftarrow 1 \\ B_1 \leftarrow 1 \\ f_1 \leftarrow x_1 \\ X_1 \leftarrow \|x_1\|^2 \\ M_1 \leftarrow 1 \\ \eta_1 \leftarrow 1 \end{cases} \tag{2.28}$$

Where R is the number of clouds (rules), B_j is the cloud birth iteration, f_j is the j^{th} cloud focal point, X_j is the j^{th} cloud average scalar product, M_j is the number of samples used to update the j^{th} cloud and η_j is the j^{th} cloud utility.

Having the antecedent part of the rule initialized, the algorithm then initializes the consequent parameters for a m dimensional input, using 2.29.

$$\begin{cases} \mathbf{A_1} \leftarrow \mathbf{0}_{1\times(m+1)} \\ \mathbf{C_1} \leftarrow \Omega_0 \mathbf{I}_{(m+1)\times(m+1)} \end{cases} \tag{2.29}$$

Where A_j is the consequent parameters vector and C_j is the covariance matrix. Ω_0 is a constant and specifies the initial value of the covariance matrix diagonal when the rule is first created. A value of $\Omega_0 = 10$ is recommended in the original article.

Having the first rule fully initialized, the algorithm then proceeds to read the new training sample. First, the value of K is incremented, and then the model global parameters are recursively updated using 2.30.

$$\begin{cases} N \leftarrow N + 1 \\ \mu \leftarrow \frac{N-1}{N}\mu + \frac{\mathbf{x_k}}{N} \\ X \leftarrow \frac{N-1}{N}X + \frac{\mathbf{x_k}}{N} \end{cases} \tag{2.30}$$

The algorithm then evaluates a global density condition (Condition 1), in which the sample and focal global unimodal densities are computed, in order to assess whether or not the region has a density anomaly, ie, if the global unimodal density is too high (high sample density) or too low (low sample density). The sample global density and the j focal global densities are computed using 2.31 and 2.32, respectively.

$$D(x_k, \mu) = \frac{1}{1 + \frac{\|x_k - \mu\|^2}{X - \|\mu\|^2}} \tag{2.31}$$

$$D(f_j, \mu) = \frac{1}{1 + \frac{\|f_j - \mu\|^2}{X - \|\mu\|^2}} \tag{2.32}$$

The global density condition (Condition 1) is then evaluated, as expressed by 2.33.

$$D(x_k, \mu) \ > \max_{j=1,2,...R}(D(f_j, \mu)) \ \vee \ D(x_k, \mu) \ < \min_{j=1,2,...R}(D(f_j, \mu)) \tag{2.33}$$

If condition is 1 not verified, then the nearest cloud, denoted as $j*$, is found using 2.34.

$$j^* = \arg\min_{j=1,2,...R}(\|x_k - f_j\|) \tag{2.34}$$

The nearest cloud parameters are then updated using 2.35.

$$\begin{cases} M_{j*} \leftarrow M_{j*} + 1 \\ f_{j*} \leftarrow \frac{M_{j*}-1}{M_{j*}} f_{j*} + \frac{\mathbf{x_k}}{M_{j*}} \\ X_{j*} \leftarrow \frac{M_{j*}-1}{M_{j*}} X_{j*} + \frac{\mathbf{x_k}}{M_{j*}} \end{cases} \tag{2.35}$$

Otherwise, if Condition 1 is verified, then the algorithm proceeds to assess if a new cloud around the sample would overlap with an already existent cloud. This is accomplished by first temporarily updating each cloud parameters denoted as M_j', f_j' and X_j', using equations 2.35, and then computing the unimodal local density for each cloud.

Local densities are computed between the sample x_k and each one of the cloud focal points f_j, using the respective local average scalar product X_j, as expressed in 2.36.

$$D(x_k, f_j') = \frac{1}{1 + \frac{\|x_k - f_j'\|^2}{X_j' - \|f_j'\|^2}} \tag{2.36}$$

The cloud with the highest local density, denote as j^* is then found, using 2.37.

$$j^* = \arg\max_{j=1,2,...R}(D(x_k, f_j')) \tag{2.37}$$

The local density condition (Condition 2) is then evaluated, as expressed by 2.38.

$$D(x_k, f_{j*}') \geq \frac{1}{1 + n^2} \tag{2.38}$$

The value of n specifies how close a certain cloud must be to be considered overlapping. In the original article, a value of $n = 0.5$ is used, meaning that the sample x_k is less than $\sigma/2$ away from the cloud focal point, and equivalent to a density value of 0.8.

If Condition 2 is verified, then an overlap exists and the overlapping cloud j^* is replaced by a new one, while the respective rule consequent parameters are kept, as expressed in equations 2.39.

$$\begin{cases} M_{j^*} \leftarrow ceil(\frac{1+M_{j^*}}{2}) \\ F_{j^*} \leftarrow \frac{x_k+F_{j^*}}{2} \\ X_{j^*} \leftarrow \frac{\|x_k\|^2+X_{j^*}}{2} \end{cases} \tag{2.39}$$

Otherwise, if Condition 2 is not verified, then there is no overlapping and a new cloud with focal point x_k is created using 2.40.

$$\begin{cases} R \leftarrow R+1 \\ B_j \leftarrow k \\ M_j \leftarrow 1 \\ f_j \leftarrow x_k \\ X_j \leftarrow \|x_k\|^2 \\ \mathbf{A_j} \leftarrow \frac{1}{N}\sum_{j=1}^{N} A_j \\ \mathbf{C_j} \leftarrow \Omega_0 \mathbf{I}_{(m+1)\times(m+1)} \end{cases} \tag{2.40}$$

The algorithm then proceeds to compute the activation level for each cloud, denoted as λ_j. The cloud activation is defined as the normalized local unimodal densities, computed using the temporarily updated cloud parameters, as expressed in 2.41.

$$\lambda_j(x_k) = \frac{D(x_k, f_j')}{\sum_{j=1}^{R} D(x_k, f_j')} \tag{2.41}$$

The algorithm then applies the cloud quality online monitoring mechanism, in which clouds that have had low activation levels in past iterations (called stale clouds) are removed. Stale clouds are detected by computing the cloud utilities, denoted as η_j, computed using 2.42.

$$\begin{cases} \eta_j \leftarrow \frac{\sum_{B_j}^{K} \lambda^j}{K+1-B_j} \; B_j > K \\ \eta_j \leftarrow 1 \; otherwise \end{cases} \tag{2.42}$$

Thus, the clouds utilities assess how useful a cloud has been to the overall model performance. The criteria to detect stale clouds is expressed by Condition 3 2.43.

$$\eta_j < \eta_0 \tag{2.43}$$

Each cloud that verifies the condition is considered stale and the associated rule is removed from the model. The value of η_0 is a small tolerance constant and it is set to $\eta_0 = 0.1$ as stated in the original article.

Finally, the algorithm proceeds to update the rule consequent parameters, using recursive least squares (RLS), using equations 2.44.

$$\begin{cases} C_j \leftarrow C_j - \frac{\lambda_j C_j u u^T C_j}{1+\lambda_j u^T C_j u} \\ A_j \leftarrow A_j + \lambda_j C_j u (y - u^T A_j) \end{cases} \tag{2.44}$$

The algorithm then proceeds to the next sample. The complete learning process is summarized below in Algorithm 28.

Algorithm 2 ALMMo System Training Algorithm

```
1: for each sample $x_k$ do
2:     if K=1 then
3:         Initialize model using 2.27
4:         Initialize first rule using 2.28 and 2.29
5:     else
6:         Update global parameters using 2.30
7:         Compute sample global density using 2.31
8:         Compute focal global densities using 2.32
9:         if Condition 1 (2.19) is met then
10:            Compute focal local densities using 2.36
11:            Find highest focal density using 2.37
12:            if Condition 2 (2.22) is met then
13:                Replace overlapping cloud using 2.39
14:            else
15:                Create new cloud using 2.40
16:            end if
17:        else
18:            Find nearest cloud using 2.34
19:            Update nearest cloud parameters using 2.35
20:        end if
21:    end if
22:    Compute cloud activations using 2.41.
23:    Update cloud utilities using 2.42.
24:    if Condition 3 2.43 is met then
25:        Remove stale clouds
26:    end if
27:    Update rule consequent parameters using 2.44
28: end for
```

Chapter 3

Imbalanced Datasets

Imbalanced datasets are classification datasets in which the objects are not equally distributed among the problem´s classes. One of the main implications of training classifiers on highly imbalanced datasets is that, broadly speaking, the final model will often show a bias towards classes that have more available samples, and in some cases may even completely ignore the underrepresented classes. This is due to the fact that most classifiers were designed assuming that class imbalance either does not exist, or if it exists, it is not severe enough to affect the classifier´s performance.

Attending to the prevalence of imbalanced datasets in very relevant areas and to the general underperformance that most traditional machine learning methods show on such datasets, a lot of research has been done throughout the years, and many different approaches have been proposed to mitigate the effects of highly imbalanced datasets. Furthermore, in most imbalanced datasets applications, the underrepresented classes are the ones of greater interest, and their correct detection is much more critical than detecting the overrepresented classes.

One particular type of imbalanced datasets that are particularly relevant are imbalanced binary classification datasets, in which one of the classes (typically the negative) is significantly overrepresented and is known as the majority class. The other class (typically the positive) is known as the minority class. Imbalanced binary datasets are particularly relevant since multi-class problems may be decomposed into several binary problems, using either a one-vs-one or a one-vs-all approach.

In 3.1, a selection of relevant classification metrics are presented, and their usage and applicability to imbalanced datasets is discussed. In 3.2, the different types of approaches to deal with imbalanced datasets are discussed. Finally, in section 3.3, the performance of ALMMo systems on imbalanced datasets is discussed, as well as the applicability of the presented methods.

3.1 Classification Metrics

In order to correctly evaluate a classifier's performance on imbalanced classification tasks, an adequate classification metric must be chosen. In fact, when dealing with imbalanced classification, the usage of an inadequate metric may completely hide the model's poor performance [38]. Therefore, the chosen metric is of paramount importance when assessing the performance of a new approach that seeks to mitigate the class imbalance problem.

In order to introduce the different classification metrics, it is helpful to introduce the concept of confusion matrix, which summarizes the classifier prediction performance. Table 3.1 shows the matrix structure, where the rows show the samples true class, and the columns show the classifier´s class predictions.

If a sample belongs to the negative (N) class, and the classifier correctly predicts the negative class, it corresponds to a true negative (TN). Otherwise, if the classifier predicts the positive class, it corresponds to a false positive (FP). For a positive (P) class sample, a correct prediction corresponds to a true positive (TP), and an incorrect negative class prediction corresponds to a false negative (FN).

True Label	**Predicted Label**	
	N	**P**
N	TN	FP
P	FN	TP

Table 3.1: Confusion matrix structure

Imbalanced datasets have a much larger number of negative class samples than positive class samples and as such, it is often observed that the models show a large bias towards the majority (negative) class. This bias translates into a large number of negative class predictions and as such, confusion matrices of highly imbalanced datasets display a large number of true negatives and false negatives. For the extreme case, in which a classifier completely disregards the minority (positive) class, the number of false positives and true positives can be zero.

Therefore, when discussing a classifier´s performance on highly imbalanced datasets, it becomes highly relevant to select classification metrics that do not disregard the minority class prediction performance. In other words, when designing a classifier for a highly imbalanced dataset, one seeks to maximize the number of true positives, while minimizing the number of false negatives, even if that implies an increase in the number of false positives, as these are considered to be far less costlier than allowing for too many false negatives.

Attending to these considerations, some of the most commonly used classification metrics are presented in 3.1.1 and 3.1.2, and their significance for imbalanced datasets is discussed.

3.1.1 Singular Measures

Singular measures are classification metrics that take as inputs the class predictions, without considering the actual classifier's scalar output before being thresholded to the class prediction. As such, singular measures assume a predefined class decision threshold and cannot provide an overall measure of the model's quality. Nonetheless, singular measures still provide good insights about the model's prediction performance, and are used extensively. Furthermore, singular measures are applicable to any classifier since they take the class predictions as the inputs.

Accuracy is possibly the most well known classification metric, as it provides a clear interpretation of the model´s performance as the ratio between the correctly classified samples, and the total number of samples. As shown in 3.1, accuracy takes as inputs all the values in the confusion matrix, providing an overall assessment of the prediction performance.

$$Accuracy = \frac{TP+TN}{P+N} = \frac{TP+TN}{TP+FN+TN+FP} \tag{3.1}$$

However, it is clear that for highly imbalanced datasets, the accuracy value can still be near maximum at 1.0, even if the classifier shows a large bias towards the negative class. For the extreme case in which the classifier completely disregards the minority class, the accuracy becomes equal to the ratio of majority class samples, as not a single one minority class prediction is observed.

Other singular measures exist, that select only some elements of the confusion matrix, in order to evaluate more specific prediction performance characteristics. Regarding the majority class, one commonly used metric is the specificity, computed as shown in 3.2, which seeks to maximize the number of true negative predictions, penalizing a large number of false positives.

$$Specificity = \frac{TN}{TN+FP} \tag{3.2}$$

Specificity is not particular useful for highly imbalanced datasets, since, similarly to accuracy, it still achieves near maximum values even when the minority class is completely disregarded. Regarding the minority class, commonly used metrics include the recall and precision, computed as shown by 3.3 and 3.4, respectively.

$$Recall = \frac{TP}{TP+FN} \tag{3.3}$$

$$Precision = \frac{TP}{TP+FP} \tag{3.4}$$

Both metrics evaluate the positive class prediction performance, as both seek to maximize the number of true positives. However, recall and precision penalize different types of predictions, with recall penalizing false negatives, and precision penalizing false positives. Therefore, one can conclude that for highly imbalanced datasets, the recall score becomes very relevant, as one seeks to minimize the number of false negatives. Nonetheless, the precision score is also relevant, as one may also seek to avoid having an unacceptably large number of false positives.

In practice, however, one may need to obtain a model that offers a compromise between increasing the positive class detection, while not exaggeratedly increasing the number of false positives. One such metric is the Geometric Mean, computed as shown in 3.5, which entails a compromise between recall and specificity. In general, these two goals contradict each other, since they effectively require the classifier to improve both the majority and minority prediction performance.

$$GM = \sqrt{Recall \times Specificity} \tag{3.5}$$

Another commonly used classification metric is the F1-Score, defined as the harmonic mean of the recall and precision scores, and computed as shown in 3.6. As such, the F1-Score has a clear bias towards the positive class which makes it more appropriate for highly imbalanced datasets. A maximum F1-Score implies that no false negatives or false positives were observed, which implies perfect recall and precision scores. However, the F1-Score gives equal importance to recall and precision, which may not be adequate for many tasks, since, as discussed, usually the false negatives are costlier than the false positives. Furthermore, it does not consider all the elements of the confusion matrix, as the true negatives are not considered.

$$F_1 = 2\frac{Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FN + FP} \tag{3.6}$$

Cohen´s Kappa Coefficient is another classification metric, computed as shown in 3.7, which seeks to take into consideration all the confusion matrix elements. Cohen´s Kappa is widely used to assess classifiers performances on highly imbalanced datasets, as it favours correct classifications of the minority class over the majority class.

$$Kappa = \frac{2(TP \times TN - FN \times FP)}{(TP + FP)(FP + TN) + (TP + FN)(FN + TN)} \tag{3.7}$$

Another singular measure generally appropriate for highly skewed data is the Matthews Correlation Coefficient (MC), computed as shown in 3.8, as it also considers all the confusion matrix elements and prioritizes minority class detection.

$$Matthews = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.8}$$

3.1.2 Rank-Based Measures

Rank-based measures allow for a more overall assessment of a model´s performance, as they evaluate the model for different class decision threshold values. For each threshold value, the model predictions are obtained, and singular measure metrics are computed for the predictions.

Therefore, rank-based measures evaluate different prediction performances from the same model, and as such, are generally considered to be a better indicator of the model´s prediction performance. However, since they require the method to have a class decision threshold, they are not applicable to classifiers that only output a class prediction instead of a class confidence scalar value.

By plotting the different performance metrics for each threshold value, one obtains what is known as a performance curve. Performance curves are used to graphically summarize a model's prediction performance and are also used by rank-based measures to summarize the performance in a single scalar value.

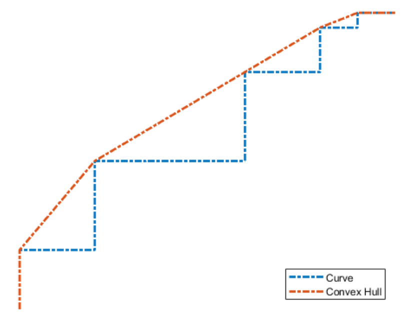
Rank-based classification metrics use the area under a performance curve to summarize the model´s performance across the different threshold values. Formally, this is defined as the integral of the performance curve. However, since in practice there is a finite number of points in the testing set, the performance curve is defined by a finite set of pairs, and the area under the performance curve is defined as a discrete integral. Furthermore, this also means that the performance curve must only be obtained for finite set of threshold values.

For a performance curve computed for thresholds T_t, and defined by the singular metrics $M_x(t)$ and $M_y(t)$ for the x and y axis, respectively, the area under the curve A, computed using the trapezoidal rule, is given by 3.9, where N_t is the number of threshold values.

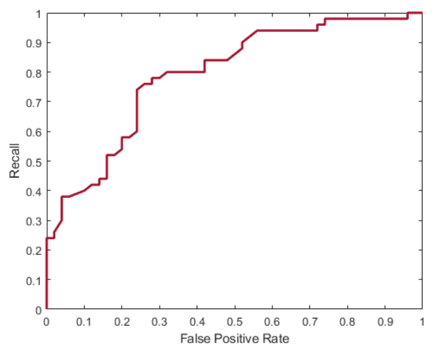
$$A = \sum_{t=1}^{N_t} \frac{[M_y(T_t) + M_y(T_{t-1})][M_x(T_t) - M_x(T_{t-1})])}{2} \tag{3.9}$$

Using the trapezoidal integration rule implies that the convex hull is obtained from the performance curve, as illustrated in Figure 3.1(a). One widely used performance curve is the Receiver Operating Curve (ROC), and the associated rank-based metric is the Area Under the Curve (AUC). The ROC is obtained by plotting the recall against the false positive rate (FPR), defined as one minus the specificity, as shown in Figure 3.1(b).

Although the AUC has been widely used and is considered to be an overall reasonable measure of the model´s performance, it may not be the most adequate metric for highly imbalanced datasets. The reason for this becomes clear by remembering that, as discussed in 3.1.1, highly skewed datasets may cause models to achieve very high specificity values, allowing for good AUC performances, even when the minority class is ignored .



(a) Performance curve and respective convex hull



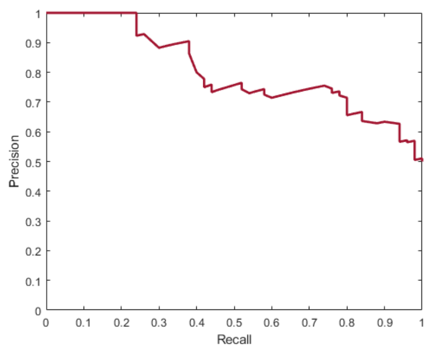
(b) Receiver Operating Curve

Figure 3.1: Convex hull obtained from a performance curve (a) and an example of a receiver operating curve (b)

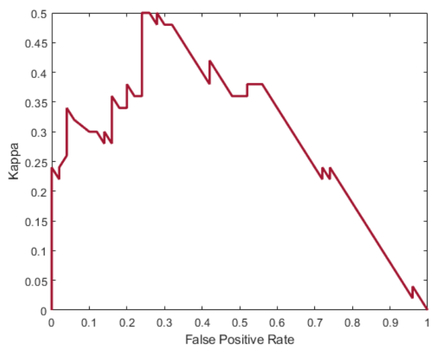
Another widely used performance curve is the Precision-Recall Curve (PRC), which plots the precision against the recall, as shown in Figure 3.2(a). Thus, the area under the PRC (AUPRC) is a generally better performance assessment for highly skewed datasets than the AUC, since it defines an inherent compromise between false positives and false negatives.

However, the AUPRC metric still presents the same issues as the F1-Score, as discussed in 3.1.1, since it gives the same relative importance to precision and recall, which may not be adequate for highly imbalanced datasets.

The Area Under the Kappa Curve (AUK) was introduced to be sensitive to the class distribution, and it is generally considered to be a more correct assessment of a model´s performance for highly imbalanced datasets. As discussed in 3.1.1, the Cohen´s Kappa coefficient gives more emphasis to the minority class in highly skewed datasets and as such, the AUK is considered to be a more statistically robust metric for imbalanced datasets, and has been extensively applied to such problems. The AUK is obtained from the Kappa Curve, which plots the Kappa Coefficient against the FPR, as shown in Figure 3.2(b).



(a) Precision-Recall Curve



(b) Kappa Curve

Figure 3.2: Example of a precision-recall curve (a) and a kappa curve (b)

Although the AUC, AUPRC and AUK provide a good overall assessment of a model´s prediction performance, one is often also interested in using the aforementioned threshold moving method to optimize the model´s threshold value.

In other words, in highly skewed datasets, one can first train the model on a training set, and then evaluate the model´s performance (usually on a separate validation set) using the aforementioned metrics. By evaluating the different threshold values, one can then choose the value that maximizes a chosen performance metric.

Depending on the chosen performance curve, one can use any metric that is a function of the metric pairs that define the curve, in order to summarize the threshold performance. On the ROC, one usually selects the threshold that maximizes the GM metric, which is a function of recall and specificity. On the PRC, one usually chooses the threshold that maximizes the F1-Score, which is a function of recall and precision. On the Kappa curve, one usually chooses the threshold that maximizes the Kappa Coefficient.

As a final note, it is important to mention that changing the model's class decision threshold does not change the value of the rank-based metric, since the area under the respective performance curve stays the same. As such, in order to compare the performance of the resultant model with the original model, one must use one of the singular-measure metrics discussed in 3.1.1.

3.2 Addressing Imbalanced Datasets

Attending to the already discussed prevalence and relevance of imbalanced datasets, a wide variety of approaches have been proposed in the literature [39], and as such, the methods hereby presented pretend to be a broad overview of the different types of approaches that have been successfully applied in different problems.

Broadly speaking, one can define two different types of approaches to highly imbalanced datasets: external (or data-level) approaches, and internal (or algorithm-level) approaches.

External approaches, which are discussed in 3.2.1, are approaches that do not modify in any way the method's learning algorithm, and only modify the training data. External approaches include methods such as data-ressampling methods [40], feature-selection methods [41], and ensemble methods [42].

Internal approaches, which are discussed in 3.2.2, are approaches that modify the method's training algorithm, and include methods such as cost-sensitive methods [43], threshold-moving methods [44], one-class classification methods [45], and hybrid methods [46].

3.2.1 External Approaches

External approaches are methods that are generally applicable to any method as these approaches only modify the training data, leaving the learning algorithm unchanged. Therefore, these methods are usually quite versatile and are often combined with algorithm-level approaches [47].

One such group of methods are data sampling methods, in which the training data is resampled in order to reduce the class imbalance. This can be achieved by either removing samples from the majority class (under-sampling) or by adding minority class samples (over-sampling).

The ressampling process itself may either be a simple random ressampling or a specific algorithmic approach. For random ressampling, the under-sampling and over-sampling methods are referred to as Random Over-Sampling (ROS) [48] and Random Under-Sampling (RUS) [49].

Another common approach is doing the ressampling process using some specific algorithm that either synthetically generates new minority samples or removes majority samples according to some criterion. One widely known approach, the Synthetic Minority Oversampling Technique (SMOTE) [50], creates new minority samples by interpolating the already existent minority samples. Another widely used method, known as Tomek Links [51], undersamples the majority class by removing majority samples that overlap with minority samples.

Generally speaking, data ressampling methods are quite simple to integrate in the modelling procedure since only the training data must be modified. Furthermore, despite the relative simpleness of the presented methods, they can still help achieving significant improvements on the minority class prediction performance, specially when combined with algorithm-level approaches.

However, it is always important to keep in mind that all these methods alter the original data distribution and in some cases may add incorrect information to the training set, specially when using methods that synthetically generate minority samples, such as SMOTE.

Another group of data-level approaches that may be used to address high class imbalances is known as feature selection. Feature selection consists of reducing the number of features by selecting the more relevant subset of features that yield unique knowledge for inter-class discrimination. While feature selection has been extensively applied to different high dimensional problems, its usage in addressing class imbalance has also allowed for promising results [52].

Despite the large variety of different feature selection methods that have been proposed in the literature, all feature selection approaches face the same problem that, for a dataset with a large number features, the number of different feature combinations is generally too large to apply any sort of exhaustive search procedure.

Therefore, most feature selection approaches either avoid searching the different combinations of features by using some feature evaluation criterion to select the relevant features before training the model (filter methods), or search the different feature combinations using a more efficient search algorithm (wrapper methods), typically using some meta heuristic to guide the search [53].

Filter methods [54] are generally considered to be the most lightweight and easy to apply feature selection methods, since they avoid any sort of feature search procedure, taking the feature selection procedure as a pre-processing step. As such, filter methods must use some sort of feature relevance metric to assess the relevance of each feature, and do not require different models to be trained on different subsets of features. However, since filter methods do not assess the feature selection quality by evaluating models on the resultant datasets, they may easily result in a sub-optimal subset of features, since this may depend on the specific learning algorithm being used.

Wrapper methods [55] train different models on different feature subsets, and use the models prediction performances to score each features subset. The feature selection search procedure effectively corresponds to an optimization problem, which implies a compromise between achieving good prediction performances while also minimizing the number of features [56].

This optimization problem is particularly complex since there is a very large number of possible feature combinations, and also since the optimal number of features is unknown and hard to define. Attending to this complexity, the optimization problem is often approached using some type of meta-heuristic, not only because such methods achieve very efficient solution searching, but also because the solution (usually represented as a vector of binary values defining which features are used and which are not) is easily representable in most of widely used meta-heuristic, such as Genetic Algorithms [57], Ant Colony Optimization [58], and Particle Swarm Optimization [59]. Feature selection using wrapper methods and meta-heuristics for the feature selection optimization are considered to be some of the most promising techniques, particularly for highly dimensional, highly imbalanced datasets [60] [61].

Another group of methods that have been widely used to approach imbalanced classification are ensemble learning methods [62]. Ensemble learning combines multiple models, known as base models, to obtain a better predictive performance, when compared to using a single complex model.

Ensemble learning techniques may either use multiple models of the same type, or use multiple types of methods to train the base models. In either case, the core principle of ensemble learning is that by having a set of diverse enough base learners and combining the different model outputs, one can, in general, achieve a better performance than by using a single complex learner [63]. Two of the most common types of ensemble learning methods are Bootstrap Aggregating (Bagging) and Boosting.

Bagging [64] is an ensemble technique created to improve the predictive performance and stability of a classifier. Bagging consists of obtaining multiple training sets (bootstraps) from the original training set, by randomly sampling with replacement, and then training multiple base models using the obtaining training sets. Since some of the samples will be repeated in the training sets, it is guaranteed that each bootstrap is independent from the others, and also that its samples are not dependent on the previously chosen samples. The base models are then aggregated, by combining their individual outputs, usually using some sort of voting procedure.

Boosting [65] is another ensemble meta-algorithm that seeks to improve the predictive performance and reduce variability. Boosting is a sequential procedure based on iteratively learning multiple weak classifiers and using the errors of the previously trained models to influence the creation of the next model. This is usually accomplished by assigning larger weights to the misclassified samples. After iteratively obtaining all the weak learners, these are combined into a final strong ensemble classifier, in which each weak learner has a different weight depending on its predication performance.

3.2.2 Internal Approaches

Internal approaches include different methods that modify in some way the learning algorithm, in order to improve the minority class prediction performance. Therefore, most of the methods that have been proposed are specific to a particular learning algorithm, and hard to generalize for different algorithms. Nonetheless, some of the more widely used and general approaches are still quite direct to adapt to most learning algorithms, and have allowed for improvements in a wide range of classifiers.

Cost-sensitive learning approaches are based on penalizing minority class misclassification, by assigning different costs to each class. In other words, when using cost-sensitive learning, false negatives have a higher cost than false positives, given that the latter is the class of interest. Cost-sensitive approaches may be further divided into direct methods and meta-learning methods [43].

Direct cost-sensitive methods are methods that incorporate cost-sensitive capabilities within the learning algorithm itself. By assigning different costs to each class, the learning algorithm is able to compensate for the small number of minority class objects, since the cost of misclassifying these objects becomes prohibitive. However, defining the class cost is a complex task and often requires expert input. Therefore, many direct cost-sensitive approaches alter the algorithm error/objective function to give more relevance to the minority class, instead of directly defining each class cost [66] [67].

Meta-learning cost-sensitive methods convert cost-insensitive classifiers to cost-sensitive ones, without modifying the learning algorithm itself. Meta-learning approaches use wrapper approaches that either pre-process the training data, or post-process the model output, in order to obtain a more unbiased final model.

One widely used cost-sensitive method is known as thresholding (or threshold-moving) [68]. Thresholding methods are applicable to methods that use a class decision threshold to classify the samples into negative, or positive. Since most traditional learning algorithms neglect the effect of high class imbalance, the standard decision threshold of 0.5 becomes inadequate, since it will result in most model predictions being the negative (majority) class.

Therefore, since cost-insensitive regressors tend to mostly output values lower than the standard 0.5 decision threshold, thresholding techniques tune the decision threshold to a lower value, in order to diminish the bias towards the negative class. In order to tune the value of class decision threshold, typically some classification metric, such as the ones discussed in 3.1.1, is used as the objective function to optimize the threshold value.

Another group of methods for addressing high class imbalance is known as one class classification (OCC) [69]. These techniques are also known as outlier detection or novelty detection, and are useful in domains where one seeks to model the normal system behaviour in order to detect unusual behaviour. Consequently, these techniques are often applied in domains where obtaining data of the normal behaviour is simple, but obtaining rare or anomalous data is costly or simply unfeasible.

One class classification refers to a wide variety of methods that have been proposed in the literature throughout the years, and includes both original OCC algorithms, and OCC adaptations of existent learning algorithms. The basic principle of OCC methods is using only the normal (majority) class samples to train the model, and then learning some type of criterion that allows the classifier to distinguish between normal (majority) and abnormal (minority) samples [70].

One class classification has also shown particularly promising results when combined with cluster-based classifiers [71]. In such methods, the majority class training data is clustered, and then some type of decision region around the clusters is defined. This decision region is typically defined using a distance-measure or a density-measure. If a sample is inside the decision region, it is classified as positive, and negative otherwise [72] [73].

Finally, it is important to mention that many of the recent state-of-the-art approaches combine multiple of the aforementioned approaches into more complex algorithms. Such approaches, often referred to as hybrid approaches, combine external and internal methods in order to further improve the minority class detection performance. Therefore, hybrid approaches must ensure that the differences between the different individual approaches properly complement each other and allow for better performance when compared to the individual methods [13].

3.3 ALMMo and Imbalanced Data

Recalling the architecture described in 2.4, ALMMo-0 Classifiers obtain the class prediction by a winner-takes-all strategy. More specifically, for each class sub-model, the nearest cloud in the sub-model is found, and the respective degree of activation is computed. The class prediction is then obtained by finding the maximum degree of activation and the respective cloud´s class.

Therefore, since there is no decision threshold on ALMMo-0 Classifiers, it is not obvious how one would apply threshold moving methods. Furthermore, since there are no consequent parameters as was the case for ALMMo systems, is it also not obvious on how to apply many of the cost sensitive strategies previously described.

One commonly used set of strategies that was already mentioned in 3.2.2 and is applicable to classifiers such as the ALMMo-0 Classifier is one-class classification. One-class classifiers are trained using only the majority class samples and seek to distinguish between the normal system behaviour and abnormal behaviour. Attending to the wide use of OCC for imbalanced data and to many promising results that have been previously presented in the literature, this thesis introduces a one class adaptation of the original ALMMo-0 Classifier, which is thoroughly discussed in 4.1.

Another proposed modification is based on viewing the ALMMo-0 Classifier as an ensemble of its class sub-models, and each class confidence score as a vote towards that class. As mentioned in 2.4, in the original article, the winner takes all strategy is taken, meaning that the class sub-model with the highest confidence score assigns its class to the sample. However, the bias towards the majority class that arises from the data imbalance causes this strategy to often misclassify the minority samples.

In order to mitigate this bias on the model, a new modification is proposed, in which a weighted winner takes all approach is used, instead of the original unweighted strategy. In the weighted strategy, the confidence scores are first weighted, and only then the highest value gets to assign its class to the sample. This proposed modification is thoroughly discussed in 4.2.

Chapter 4

Proposed Methods

In this chapter, the performance of ALMMo-0 Classifiers on imbalanced binary classification tasks is further analyzed. Based on this analysis, as well as in the empirical results previously obtained with the original algorithm, two different approaches are proposed to improve the performance of the original ALMMo-0 algorithm on highly imbalanced data.

The first approach, presented in 4.1, is based on the methods used in One-Class Classification (OCC), and in particular, on some the work previously done on OCC clustering methods. OCC, also known as outlier detection or novelty detection, is useful in domains where obtaining samples of the normal or common behaviour is quite simple, but obtaining rare, or anomalous samples is costly or even unfeasible.

A lot of research has been conducted in the topic, and plenty of different approaches have been proposed in the literature. However, all methods are based on training the model using only the data from one class, in order to obtain a model that distinguishes between that class and the other class. The proposed modifications to use such a training strategy on ALMMo-0 classifiers is thoroughly explained in 4.1.

The second approach, presented in 4.2, is based on looking at the ALMMo-0 not as a single classifier, but as a ensemble of its class classifiers. In particular, ALMMo-0 Classifiers may be viewed as an ensemble of one class classifiers, each one trained on its class samples. Thus, the original classification process and its winner-takes-all strategy may be viewed as a voting process, where each classifier votes with a score of confidence on its class, and the voter with the highest score assigns its class to the sample.

Therefore, in this thesis, a modification of the voting strategy proposed in the original ALMMo-0 algorithm is presented. In the proposed modification, a weighted majority voting strategy is introduced, replacing the original winner-takes-all strategy. In this modification, the scores of confidence of each class are first weighted, and only then the class with the largest weighted score is determined. A thorough discussion of the impacts of this strategy on the classifier performance, as well as the detailed required changes to the original training algorithm are presented in 4.2.

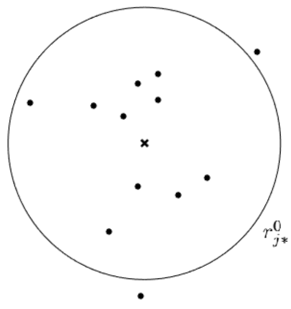
4.1 ALMMo-0 - One Class Classifier

In one-class classification, models are trained using only the normal (majority) class data with the goal of training a model that distinguishes between normal system behaviour and abnormal behaviour. This task, also known as outlier detection or novelty detection, has been successfully applied to highly imbalanced datasets. Concretely, several clustering-based one-class classification models have been introduced, and their general principles can be applied to ALMMo-0 classifiers.

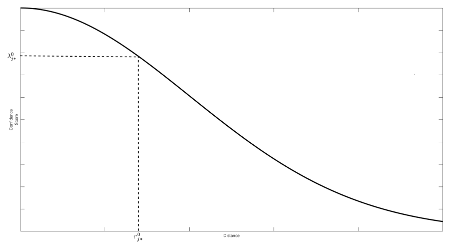
Clustering-based one-class classifiers methods must define an appropriate criterion to distinguish between normal and abnormal samples. Generally speaking, two types of criterion may be defined: boundary-based and density-based. Boundary-based classifiers rely on some type of boundary (either hard or soft) that delimits the norm class clusters. Density-based classifiers use the training samples to estimate a probability density function, which is used to compute the likelihood that the sample belongs to the normal class. If the likelihood is above a certain threshold, the model classifiers the sample as being normal, and abnormal otherwise.

Returning to ALMMo-0 classifiers, one could conceive a one-class adaptation of the original algorithm, by only training the majority class sub-model, and therefore creating only one set of rules (clouds), that describe the normal behaviour of the system. However, one must also define some classification criterion, that is based on the cloud structure of ALMMo-0 classifiers.

Recalling the training process described by Algorithm 1, Condition 2 defines a proximity criterion, in which the cloud radius is set as a decision boundary to assess whether or not a sample is within the a cloud´s region of influence. Therefore, one could use this criterion to classify a sample, by classifying samples within a cloud's region of influence as normal, and abnormal otherwise. Figure 4.1 illustrates this classification strategy.



(a) Focal point, radius and support samples



(b) Confidence score with threshold at the cloud radius

Figure 4.1: Illustration of a cloud radius (a) and the correspondent confidence score threshold (b)

Furthermore, the clouds radius are already iteratively updated, as part of the training algorithm, meaning that no extra parameters must be estimated to apply this classification criterion. Formally, this classification criterion is as shown in 4.1.

$$\begin{cases} \hat{c} = 0 & \|x - f^0_{j*}\| < r^0_{j*} \\ \hat{c} = 1 & otherwise \end{cases} \tag{4.1}$$

Where $\hat{c}$ is the class prediction, x is a sample, f^0_{j*} is the closest majority class focal point, and r^0_{j*} is its respective cloud´s radius. This boundary-based criterion may also be interpreted as a density-based criterion, since the cloud radius effectively acts as a the activation threshold for the cloud activation λ^0_{j*}

This simple modification to the original algorithm seeks to be a lightweight approach to improve the performance of the original classifier on imbalanced binary classification. Comparing to the original classifier, no extra parameters must be estimated, and the minority class sub-model is not trained, meaning that the resultant model has fewer rules (clouds) and is slightly less complex.

Training Procedure

Attending to the simpleness of the proposed modification, the only required change to the training algorithm is to only training the majority class sub-model, using the original training algorithm.

The algorithm starts by finding the number of samples for each class. The class with the largest number of samples (majority class) is set as the default (normal) class. Thus, the minority class sub-model is not trained and it is effectively removed from the model.

4.2 ALMMo-0 - Weighted Class Confidence

Recalling the architecture described in 2.4, ALMMo-0 classifiers are defined by a set of sub-models, each one modelling its class with a set of AnYa type fuzzy rules. The classification strategy is, as already described, accomplished using the winner takes all approach. This approach takes each one of the sub-models confidence levels, and assigns the class of the sub-model with the highest confidence.

However, as already mentioned, ALMMo-0 also suffer from a bias towards the majority class because of its larger number of clouds that will often capture minority samples and misclassify the sample as false negatives.Furthermore, the misclassified minority samples often have a confidence level λ^1 that is only ever so slightly smaller than the confidence level for the closest majority cloud λ^0.

This suggests that one could introduce a weighting strategy that assigns different weights to the different class sub-models. Returning to the specific case of binary classification, if one introduces two complementary weights, denoted as ω_0 and ω_1, that weight the class confidence levels, λ^0 and λ^1, a winner takes all strategy may defined as in 4.2.

$$\hat{c} = \arg\max(\omega_0 \lambda^0_{j^*}, \omega_1 \lambda^1_{j^*}) \tag{4.2}$$

For the case where $\omega_0 = \omega_1$, this approach is effectively equivalent to the original winner takes all strategy. Even though only the relative values of the weights is important, for clarity and intrepretability sake, the weights are defined as in 4.3.

$$\begin{cases} 0 \leq \omega_0 \leq 1 \\ 0 \leq \omega_1 \leq 1 \\ \omega_0 + \omega_1 = 1 \end{cases} \tag{4.3}$$

Attending to this definition, only one of weights needs to be estimated, since the other weight is simply its complement. This means that the weighted confidence strategy has similarities to threshold moving approaches applied to regressors, in which also one parameter, the decision threshold, is estimated.

In fact, if one defines the normalized activations, denoted as γ_0 and γ_1,

$$\begin{cases} \gamma_0 = \frac{\lambda^0}{\lambda^0 + \lambda_1} \\ \gamma_1 = \frac{\lambda^1}{\lambda^0 + \lambda_1} \end{cases} \tag{4.4}$$

The classification criterion is, therefore, defined as in 4.5.

$$\begin{cases} \hat{c} = 0 & \gamma_1 < 0.5 \\ \hat{c} = 1 & otherwise \end{cases} \tag{4.5}$$

Therefore, the normalized majority class confidence level, γ_1, acts as the classification decision threshold, meaning that the proposed weighted confidence strategy is effectively equivalent to adapt the ALMMo-0 classifier to a regressor, using a similar procedure to the one used for ALMMo systems, and then optimizing a decision threshold.

Nonetheless, since ALMMo-0 classifiers have been discussed as such, the approach hereby presented is always discussed from the weighted confidence perspective, in line with weighted voting strategies used in ensemble learning. Figure 4.2 illustrates this ensemble interpretation of the ALMMo-0 architecture.

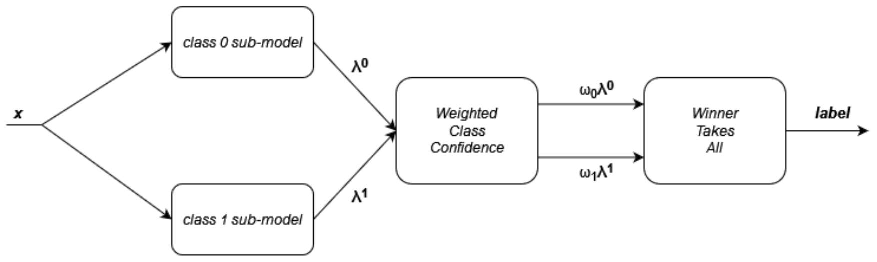


Figure 4.2: Diagram of the modified ALMMo-0 Classifier, in which the class confidence levels are weighted before the class decision

Furthermore, the proposed approach, when interpreted using the knowledge of the ALMMo-0 cloud structure, offers some possible advantages when compared to the one class classification approach presented in section 4.1. Contrarily to the original model, the OCC approach does not segment the data space using Voronoi tessellation, using the cloud radius as the classification criterion instead.

However, this results in a loss of information, particularly in data regions that would be populated with minority class clouds, meaning that the proposed one class classifier will often misclassify a significant number of negative samples that are incorrectly captured by the majority class clouds. Using Voronoi tesselation to segment the data space between both class clouds allows for the details of both classes to be locally captured by each cloud.

The weighted confidence strategy also uses Voronoi tessellation to segment the data space, but the class weighting modifies the segmentation cells dimensions, by expanding all the minority clouds cells around the focal point, proportionally to the minority class confidence weight.

Finally, the question of how to optimize the class weights still remains to be answered. Similarly to what is usually done in threshold moving methods, the training algorithm is divided into two stages. In the first stage, the model is trained normally, using both class samples. In the second stage, the threshold value is optimized, usually using the classifier performance on a separate validation set, that was not used during the first training stage.

One crucial aspect that must be defined beforehand is the criterion used to assess the classifier performance on the validation set. Generally speaking, one may choose any of the classification metrics already presented in 3.1.1, to define a cost function that is only a function of ω_1. This cost function, denoted as F, is defined as in 4.6.

$$F(\omega_1) = F(\hat{y}(\omega_1), y) \tag{4.6}$$

Where F encodes some type of singular metric that is itself only a function of the predictions $\hat{y}$ and the actual labels y. Thus, the cost function is computed by first computing the predictions $\hat{y}$ for a given minority class weight ω_1, and then computing the classification metrics with the obtained results.

The cost function F may multiple local minimums, and therefore gradient methods may get stuck at one of the local minimums and yield a sub-optimal solution. Furthermore, in order to compute the cost function for a given ω_1, the class predictions $\hat{y}$ must first be computed, which even though is not prohibitively slow, may be slow enough to make grid search methods not applicable to online training environments.

Attending to the cost function characteristics and also to the need to minimize the number of times the cost function is sampled, the proposed method for the weight optimization is Baeysian optimization, which has been widely applied to a wide range of problems and, in particular, to hyper parameter optimization in machine learning models.

Bayesian optimization is an optimization technique that uses Bayes Theorem to direct the search in an efficient and effective manner, compromising between exploring unexplored regions of the search space, and refining the search in regions with high likelihood of containing a new minimum. A thorough discussion of the principles behind Baeysian optimization is presented in Appendix A.

A detailed explanation of the training algorithm required modifications is presented in the following section.

Training Procedure

As mentioned, the training algorithm can be divided in two stages. The first stage, in which the training set, denoted as X_{train}, is used to train both class sub-models, is completely similar to the one described for the original algorithm, as described in Algorithm 1.

The second stage concerns the optimization of the class confidence weights. In order to define the optimization problem, first the search interval for the value of ω_1 must be defined. As mentioned, in order to avoid over fit of the training data, a separate validation set is used to assess the performance of the unweighted model and optimize the weight values.

Since, as mentioned, the proposed algorithm is meant for the specific case of imbalanced binary classification, and since the minority class is known beforehand, in order to increase the bias towards the minority class, the minority class weight must be greater than 0.5, which corresponds to the unweighted model. If no performance improvements are found in the validation set, both weights keep the same value of 0.5.

Therefore, only the misclassified minority class samples of the validation set are used to define the search space. First, the validation set class predictions, denoted as $\hat{y}_{valid}$, are computed for the unweighted model. Then, for each false negative, both class confidence levels, denoted as λ^0 and λ^1, respectively, are used to compute the minimum minority class weight that would correctly classify the misclassified sample, denoted as ω_1^*, and computed as shown in 4.7.

$$\omega_1^* = \frac{\lambda^0}{\lambda^0 + \lambda^1} \tag{4.7}$$

These candidate minority class weights are computed for each false negative and grouped in a vector of candidate weights, denoted as ω_1^{cands}. The search interval is then defined as in 4.8.

$$\omega_1 \in [0.5\ ,\ \max(\omega_1^{cands})] \tag{4.8}$$

By setting the maximum search value as the maximum of all the minority class candidate weights, its is guaranteed that the optimization can lead to a weight that correctly classifiers all the minority class samples that were misclassified by the unweighted model. Furthermore, since the initial cost is computed for the unweighted model, it is guaranteed that the performance of the model will never degrade relative to the original model.

Having the search interval defined, it is now possible to formulate the optimization problem, as shown in 4.9.

$$\begin{aligned} \text{Minimize} \quad & F(\omega_1) = F(\hat{y}(\omega_1), y) \\ \text{subject to} \quad & 0.5 \ \leq \ \omega_1 \ \leq \max(\omega_1^{cands}) \end{aligned}$$

As already mentioned, the cost function $F(\omega_1)$ should encode an appropriate classification metric, or a combination of different classification metrics. However, it is crucial to mention that the goal of this optimization is not necessarily to improve the performance of the model in terms of the metrics encoded by the cost function. Since the validation set is used for the weight optimization, the optimal value will only be optimal for the validation data, meaning that it does not guarantee optimal in the test set.

Algorithm 3 ALMMo-0 Classifier - Weighted Class Confidence

```
1: Train class sub-models using training samples and Algorithm 2
2: Compute validation samples activations
3: Find false negative samples and respective candidate weights
4: Initialize minority class weight search space using the candidate weights
5: while stop criterion not met do
6:     Select a new weight candidate by optimizing the acquisition function
7:     Evaluate sample using the objective function
8:     Update searched values and minimum
9:     Update surrogate function
10: end while
11: Update model class weights
```

Chapter 5

Results

5.1 Validation Procedure

In order to assess the performance of both the original algorithms, as well as the proposed methods, six widely known and commonly used benchmark datasets were chosen. The chosen benchmark datasets pretend to be as diverse as possible, particularly regarding both the number and type of features. Furthermore, as will be thoroughly explained in 5.2, in order to study performances for different degrees of class imbalance, the datasets were ressampled to artificially obtain datasets with incrementally low minority class ratios. Thus, the chosen original datasets could not have too few samples or be overly imbalanced.

Attending to these restrictions the chosen datasets are the Australian Credit Approval (Australian) [74], German Credit (German) [75], Mammographic Mass (Mammographic) [76], Pima Indians Diabetes (Pima) [77], Diagnostic Wisconsin Breast Cancer (WBCD) [78] and Original Wisconsin Breast Cancer (WBCO) [79]. The general characteristics of the chosen datasets are shown in Table 5.1.

Dataset	**Imbalance**	**Features**		
		Real	**Integer**	**Categorical**
Australian Credit Approval	44.5%	3	5	6
German Credit	30.0%	0	7	13
Mammographic Mass	48.6%	0	5	0
Pima Indians Diabetes	34.9%	2	6	0
Wisconsin Breast Cancer (Diagnostic)	37.3%	30	0	0
Wisconsin Breast Cancer (Original)	35.0%	0	9	0

Table 5.1: Benchmark datasets characteristics

It is important to recall that throughout this thesis, class imbalance is defined as the percentage of minority samples, and it is expressed as shown in 5.1, where N_{min} is the number of minority class samples, and N_{maj} is the number of majority class samples.

$$Imbalance = \frac{N_{min}}{N_{maj} + N_{min}} \tag{5.1}$$

In order to study the performance for the different methods and for different class imbalances, the benchmark datasets were ressampled to obtain class imbalances of 20%, 10%, 5% and 1%. The ressampling procedure had to be carefully chosen, respecting certain restrictions.

Firstly, attending to the relatively small number of samples of the chosen datasets, one must make sure that the 1% ressampled datasets still have a significant number of minority class samples.

Secondly, while obtaining the desired class ratios strictly using majority class overssample would not remove any minority samples, the resultant datasets would not be very useful to study the performance for different class imbalances, since we instered in how the performance of the different methods changes with an incresing loss of information on the minority class data. Furthermore, if one were to use solely oversampling of the majority class to achieve the class ratio, the data sets would get unnecessarily large at lower ratios as a large number of majority class samples would be needed.

In order to achieve a reasonable compromise for all these restrictions, each dataset was first ressampled by randomly oversampling the majority class samples, in order to achieve a 5% class imbalance. Then, the minority class samples of the resultant dataset are randomly oversampled to achieve a 20% class imbalance.

This procedure creates a new 20% imbalanced dataset that is significantly larger than the original one, guaranteeing that the 1% imbalance still has a large enough number of minority class samples. The 10%, 5% and 1% class imbalances are then obtained by randomly undersampling the minority class samples.

Therefore, the lower imbalances of 20% and 10% have a larger number of minority samples than the original dataset, meaning that while a moderate class imbalance exists, there still is a large number of minority samples that should allow for the correct modelling of the minority class. Class imbalances of 1% have a smaller number of minority samples than the original datasets, meaning that they not only have a high class imbalance, but also have significant loss of information on the minority class. Class imbalances of 5% have the same number of minority samples as the original datasets, meaning that they show high class imbalance, while having, in general, just enough minority class data to correctly model it. Furthermore, since the 5% imbalance datasets are obtained from the resultant 20% randomly ressampled dataset, the 5% imbalance datasets have different minority class samples from the original dataset, despite having the same number of minority samples.

Using the described ressampling procedure, the obtained datasets are described in Table 5.2.

For each test, 5-fold validation was used, meaning that 5 tests are performed, each one using a different fold as the testing set, and the remaining ones (80%) as the training set. Recalling the training algorithm proposed for the weighted class confidence ALMMo-0, the validation set was randomly sampled from the training folds, for each test. In order to achieve a 20% test, 10% validation, 70% train ratio, the validation set corresponds to 12.5% of the training set.

Attending to the random nature of the described resampling procedure, the resampling procedure was repeated 5 times for each one of the original datasets, creating 5 different datasets, that were then used to obtain the different class imbalances, as already described. Thus, a total of 25 tests were performed for each method and for each resampled dataset.

Regarding the choice of the benchmark methods, and since this thesis seeks to compare the original

Dataset	Imbalance	Majority Samples	Minority Samples
Australian	20%	5831	1458
	10%	5831	648
	5%	5831	307
	1%	5831	58
German	20%	5698	1424
	10%	5698	633
	5%	5698	300
	1%	5698	57
Mammographic	20%	7655	1914
	10%	7655	850
	5%	7655	403
	1%	7655	77
Pima	20%	5090	1272
	10%	5090	565
	5%	5090	268
	1%	5090	51
WBCD	20%	4026	1006
	10%	4026	447
	5%	4026	212
	1%	4026	40
WBCO	20%	4539	1135
	10%	4539	504
	5%	4539	239
	1%	4539	45

Table 5.2: Ressampled benchmark datasets characteristics

ALMMo-0 performance to both common classification benchmark methods and other FRB systems, two groups of methods were chosen as benchmarks.

The first group of methods, composed of Logistic Regression (LR), Support Vector Machine (SVM), and shallow Neural Network (NN), are commonly used benchmark methods used for classification tasks. The chosen methods are all relatively simple methods compared to state of the art methods, since the ALMMo-0 classifier is also a fairly simple and lightweight structure.

The second group of methods was chosen with the intent of comparing the original ALMMo-0 to both traditional FIS, as well as other more complex AnYa type FRB systems. For tradition FIS, the adaptive neuro-fuzzy inference system (ANFIS) was chosen. ANFIS are a type of artificial neural network based on a first-order Takagi-Sugeno inference system. Regarding AnYa type FRB systems, the natural choice is the already discussed ALMMo regressor, which is also a first-order FIS. Since ALMMo regressors share a simmilar antecedent structure, and since they are also a less lightweight and more complex algorithm, their performance compared to the ALMMo-0 classifier is of particular interest.

Regarding the choice of cost functions for the proposed ALMMo-0 weighted class method, four pertinent and commonly used classification metrics were selected. These classification metrics are the already described geometric mean (GM), F1-Score (F1), Cohen's kappa coefficient (KC), and the Matthews correlation coefficient (MC).

5.2 Benchmark Dataset Results

Tables 5.4 to 5.15 present detailed results for all the methods, datasets and class imbalances. A complete list of all the nomenclature used throughout the remainder of this chapter is presented in Table 5.3

Legened	**Meaning**
LR	Logistic Regression
SVM	Support Vector Machine
NN	Shallow Neural Network
ANFIS	First-Order Adaptive Neuro-Fuzzy Inference System
ALMMo	First-Order Autonomous Learning Multi-Model System
ALMMo-0	Zero-Order Autonomous Leraning Multi-Model Classifier
ALMMo-0-1C	ALMMo-0 One Class Classifier
ALMMo-0-W	ALMMo-0 Weighted Class Confidence
(GM)	Cost = 1 - Geometric-Mean
(F1)	Cost = 1 - F1-Score
(KC)	Cost = 1 - Cohen´s Kappa Coefficient
(MC)	Cost = 1 - Matthew's Correlation Coefficient

Table 5.3: Nomenclature used in presented results

Tables 5.4 to 5.15 are organized by class imbalance (20%, 10%, 5% and 1%). For each class imbalance, the different methods are divided into three groups to facilitate the reading of the results. For each one of the column metrics, the best result in each group is shown in bold, and the best result among all methods is underlined.

In order to more easily analyze the performance of the different methods for the different datasets, and attending to the results presented in Tables 5.4 to 5.15, the performance discussion will be conducted referring using the term ”low imbalance” for 20% and 10% class imbalances, and the term ”high imbalance” for the 5% and 1% class imbalances.

Imbalance	Method	Accuracy	Recall	Precision	Specificity
	LR	0.903 ± 0.009	0.701 ± 0.029	0.793 ± 0.032	0.954 ± 0.009
	SVM	**0.920 ± 0.007**	**0.776 ± 0.027**	0.816 ± 0.023	0.956 ± 0.006
	NN	0.914 ± 0.008	0.731 ± 0.032	**0.821 ± 0.028**	**0.960 ± 0.008**
	ANFIS	0.900 ± 0.007	0.666 ± 0.030	0.803 ± 0.027	0.959 ± 0.007
	ALMMo	0.838 ± 0.039	**0.898 ± 0.042**	0.569 ± 0.069	0.823 ± 0.055
20%	ALMMo-0	**0.932 ± 0.013**	0.781 ± 0.045	**0.868 ± 0.042**	**0.970 ± 0.010**
	ALMMo-0-1C	0.918 ± 0.010	0.671 ± 0.045	**0.894 ± 0.032**	**0.980 ± 0.007**
	ALMMo-0-w (GM)	0.924 ± 0.022	**0.775 ± 0.049**	0.844 ± 0.087	0.961 ± 0.029
	ALMMo-0-w (F1)	**0.930 ± 0.011**	0.766 ± 0.050	0.871 ± 0.046	0.971 ± 0.013
	ALMMo-0-w (KC)	**0.930 ± 0.011**	0.766 ± 0.050	0.871 ± 0.046	0.971 ± 0.013
	ALMMo-0-w (MC)	**0.930 ± 0.011**	0.766 ± 0.050	0.871 ± 0.046	0.971 ± 0.013
	LR	0.935 ± 0.007	0.548 ± 0.054	0.739 ± 0.054	0.978 ± 0.006
	SVM	0.941 ± 0.007	**0.627 ± 0.042**	0.749 ± 0.055	0.976 ± 0.008
	NN	**0.942 ± 0.007**	0.586 ± 0.048	0.779 ± 0.053	0.981 ± 0.005
	ANFIS	0.921 ± 0.007	0.251 ± 0.061	**0.848 ± 0.092**	**0.995 ± 0.003**
	ALMMo	0.882 ± 0.066	**0.778 ± 0.136**	0.526 ± 0.163	0.894 ± 0.085
10%	ALMMo-0	**0.947 ± 0.009**	0.566 ± 0.077	**0.863 ± 0.086**	**0.989 ± 0.008**
	ALMMo-0-1C	**0.956 ± 0.007**	**0.699 ± 0.049**	**0.846 ± 0.077**	**0.985 ± 0.009**
	ALMMo-0-w (GM)	0.864 ± 0.090	0.632 ± 0.110	0.539 ± 0.256	0.890 ± 0.108
	ALMMo-0-w (F1)	0.942 ± 0.010	0.559 ± 0.083	0.801 ± 0.076	0.984 ± 0.007
	ALMMo-0-w (KC)	0.942 ± 0.010	0.559 ± 0.083	0.801 ± 0.076	0.984 ± 0.007
	ALMMo-0-w (MC)	0.942 ± 0.010	0.559 ± 0.083	0.801 ± 0.076	0.984 ± 0.007
	LR	0.957 ± 0.004	0.282 ± 0.071	0.683 ± 0.112	0.993 ± 0.004
	SVM	**0.965 ± 0.005**	**0.365 ± 0.109**	**0.870 ± 0.141**	0.996 ± 0.004
	NN	0.961 ± 0.005	0.357 ± 0.063	0.738 ± 0.110	0.993 ± 0.004
	ANFIS	0.951 ± 0.002	0.030 ± 0.027	0.670 ± 0.461	**0.999 ± 0.002**
	ALMMo	0.891 ± 0.182	**0.595 ± 0.298**	0.498 ± 0.254	0.906 ± 0.199
5%	ALMMo-0	**0.967 ± 0.006**	0.463 ± 0.060	**0.800 ± 0.128**	**0.993 ± 0.005**
	ALMMo-0-1C	0.964 ± 0.008	**0.684 ± 0.080**	0.634 ± 0.091	0.979 ± 0.007
	ALMMo-0-w (GM)	0.908 ± 0.111	0.491 ± 0.138	0.587 ± 0.288	0.930 ± 0.122
	ALMMo-0-w (F1)	**0.965 ± 0.005**	0.442 ± 0.080	**0.753 ± 0.105**	**0.992 ± 0.004**
	ALMMo-0-w (KC)	**0.965 ± 0.005**	0.442 ± 0.080	**0.753 ± 0.105**	**0.992 ± 0.004**
	ALMMo-0-w (MC)	**0.965 ± 0.005**	0.442 ± 0.080	**0.753 ± 0.105**	**0.992 ± 0.004**
	LR	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	SVM	**0.990 ± 0.000**	0.009 ± 0.031	0.080 ± 0.277	**1.000 ± 0.000**
	NN	0.990 ± 0.001	**0.018 ± 0.053**	**0.120 ± 0.332**	**1.000 ± 0.000**
	ANFIS	0.989 ± 0.002	0.004 ± 0.022	0.040 ± 0.200	0.999 ± 0.002
	ALMMo	**0.990 ± 0.001**	0.018 ± 0.053	0.120 ± 0.332	**1.000 ± 0.000**
1%	ALMMo-0	0.990 ± 0.003	**0.209 ± 0.117**	**0.687 ± 0.362**	0.998 ± 0.003
	ALMMo-0-1C	**0.980 ± 0.008**	**0.756 ± 0.140**	0.324 ± 0.111	**0.982 ± 0.008**
	ALMMo-0-w (GM)	0.682 ± 0.240	0.427 ± 0.248	0.127 ± 0.281	0.685 ± 0.244
	ALMMo-0-w (F1)	0.833 ± 0.251	0.298 ± 0.224	0.354 ± 0.415	0.838 ± 0.255
	ALMMo-0-w (KC)	0.844 ± 0.252	0.289 ± 0.227	**0.355 ± 0.414**	0.850 ± 0.256
	ALMMo-0-w (MC)	0.801 ± 0.290	0.351 ± 0.305	0.349 ± 0.418	0.805 ± 0.295

Table 5.4: Accuracy, Recall, Precision and Specificity results for the Australian dataset

Imbalance	Method	GMean	F1Score	Kappa	Matthews
	LR	0.818 ± 0.018	0.744 ± 0.025	0.684 ± 0.030	0.687 ± 0.030
	SVM	**0.861 ± 0.015**	**0.796 ± 0.019**	**0.746 ± 0.024**	**0.747 ± 0.024**
	NN	0.838 ± 0.019	0.773 ± 0.023	0.720 ± 0.028	0.722 ± 0.027
	ANFIS	0.799 ± 0.018	0.728 ± 0.022	0.668 ± 0.026	0.672 ± 0.025
	ALMMo	0.859 ± 0.021	0.693 ± 0.045	0.592 ± 0.067	0.623 ± 0.051
20%	ALMMo-0	**0.870 ± 0.026**	**0.821 ± 0.034**	**0.780 ± 0.042**	**0.782 ± 0.042**
	ALMMo-0-1C	0.811 ± 0.027	0.766 ± 0.033	0.717 ± 0.039	0.729 ± 0.036
	ALMMo-0-w (GM)	**0.863 ± 0.026**	0.805 ± 0.045	0.758 ± 0.059	0.761 ± 0.058
	ALMMo-0-w (F1)	0.862 ± 0.026	**0.813 ± 0.030**	**0.770 ± 0.036**	**0.774 ± 0.035**
	ALMMo-0-w (KC)	0.862 ± 0.026	**0.813 ± 0.030**	**0.770 ± 0.036**	**0.774 ± 0.035**
	ALMMo-0-w (MC)	0.862 ± 0.026	**0.813 ± 0.030**	**0.770 ± 0.036**	**0.774 ± 0.035**
	LR	0.731 ± 0.036	0.628 ± 0.045	0.593 ± 0.048	0.602 ± 0.047
	SVM	**0.782 ± 0.025**	**0.681 ± 0.031**	**0.649 ± 0.034**	**0.653 ± 0.035**
	NN	0.758 ± 0.032	0.668 ± 0.041	0.637 ± 0.045	0.645 ± 0.044
	ANFIS	0.496 ± 0.063	0.385 ± 0.078	0.356 ± 0.077	0.433 ± 0.074
	ALMMo	**0.827 ± 0.052**	0.595 ± 0.091	0.537 ± 0.115	0.570 ± 0.082
10%	ALMMo-0	0.746 ± 0.051	**0.679 ± 0.062**	**0.651 ± 0.066**	**0.671 ± 0.063**
	ALMMo-0-1C	**0.829 ± 0.028**	**0.762 ± 0.035**	**0.738 ± 0.039**	**0.744 ± 0.040**
	ALMMo-0-w (GM)	0.743 ± 0.044	0.531 ± 0.143	0.467 ± 0.179	0.494 ± 0.159
	ALMMo-0-w (F1)	0.740 ± 0.054	0.655 ± 0.067	0.624 ± 0.072	0.638 ± 0.069
	ALMMo-0-w (KC)	0.740 ± 0.054	0.655 ± 0.067	0.624 ± 0.072	0.638 ± 0.069
	ALMMo-0-w (MC)	0.740 ± 0.054	0.655 ± 0.067	0.624 ± 0.072	0.638 ± 0.069
	LR	0.525 ± 0.071	0.393 ± 0.080	0.375 ± 0.079	0.418 ± 0.074
	SVM	**0.596 ± 0.091**	**0.498 ± 0.101**	**0.484 ± 0.100**	**0.540 ± 0.079**
	NN	0.593 ± 0.054	0.478 ± 0.072	0.460 ± 0.073	0.495 ± 0.074
	ANFIS	0.144 ± 0.101	0.058 ± 0.049	0.054 ± 0.048	0.132 ± 0.100
	ALMMo	0.671 ± 0.252	0.417 ± 0.165	0.385 ± 0.170	0.435 ± 0.150
5%	ALMMo-0	**0.677 ± 0.044**	**0.582 ± 0.067**	**0.566 ± 0.070**	**0.591 ± 0.074**
	ALMMo-0-1C	**0.817 ± 0.050**	**0.655 ± 0.072**	**0.636 ± 0.076**	**0.638 ± 0.076**
	ALMMo-0-w (GM)	0.663 ± 0.063	0.456 ± 0.154	0.426 ± 0.176	0.461 ± 0.159
	ALMMo-0-w (F1)	0.660 ± 0.061	0.551 ± 0.074	0.534 ± 0.076	0.558 ± 0.073
	ALMMo-0-w (KC)	0.660 ± 0.061	0.551 ± 0.074	0.534 ± 0.076	0.558 ± 0.073
	ALMMo-0-w (MC)	0.660 ± 0.061	0.551 ± 0.074	0.534 ± 0.076	0.558 ± 0.073
	LR	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	SVM	0.027 ± 0.092	0.016 ± 0.055	0.016 ± 0.055	0.027 ± 0.092
	NN	**0.046 ± 0.128**	**0.031 ± 0.089**	**0.030 ± 0.088**	**0.045 ± 0.127**
	ANFIS	0.013 ± 0.067	0.008 ± 0.040	0.006 ± 0.040	0.011 ± 0.067
	ALMMo	0.046 ± 0.128	0.031 ± 0.089	0.030 ± 0.088	0.045 ± 0.127
1%	ALMMo-0	**0.428 ± 0.163**	**0.294 ± 0.145**	**0.290 ± 0.146**	**0.354 ± 0.161**
	ALMMo-0-1C	**0.858 ± 0.080**	**0.447 ± 0.124**	**0.439 ± 0.127**	**0.483 ± 0.120**
	ALMMo-0-w (GM)	0.480 ± 0.157	0.072 ± 0.099	0.056 ± 0.105	0.078 ± 0.131
	ALMMo-0-w (F1)	0.415 ± 0.178	0.154 ± 0.140	0.145 ± 0.146	0.185 ± 0.182
	ALMMo-0-w (KC)	0.409 ± 0.180	0.156 ± 0.140	0.147 ± 0.145	0.187 ± 0.181
	ALMMo-0-w (MC)	0.414 ± 0.190	0.147 ± 0.143	0.138 ± 0.148	0.182 ± 0.181

Table 5.5: Geometric Mean, F1-Score, Cohen´s Kappa and Matthews Correlation Coefficient results for the Australian dataset

Imbalance	Method	Accuracy	Recall	Precision	Specificity
	LR	0.807 ± 0.008	0.216 ± 0.032	0.547 ± 0.049	0.955 ± 0.009
	SVM	**0.910 ± 0.009**	**0.584 ± 0.045**	**0.945 ± 0.028**	**0.991 ± 0.005**
	NN	0.857 ± 0.011	0.408 ± 0.063	0.774 ± 0.064	0.969 ± 0.012
	ANFIS	0.809 ± 0.007	0.138 ± 0.027	0.608 ± 0.090	0.976 ± 0.010
	ALMMo	0.811 ± 0.007	0.145 ± 0.033	0.624 ± 0.081	**0.977 ± 0.010**
20%	ALMMo-0	**0.897 ± 0.014**	**0.662 ± 0.056**	**0.791 ± 0.042**	0.956 ± 0.010
	ALMMo-0-1C	0.812 ± 0.013	0.292 ± 0.044	0.560 ± 0.062	0.942 ± 0.015
	ALMMo-0-w (GM)	0.835 ± 0.059	**0.759 ± 0.090**	0.607 ± 0.135	0.854 ± 0.092
	ALMMo-0-w (F1)	**0.883 ± 0.018**	0.634 ± 0.073	**0.749 ± 0.062**	**0.945 ± 0.022**
	ALMMo-0-w (KC)	0.883 ± 0.020	0.639 ± 0.060	0.747 ± 0.071	0.944 ± 0.025
	ALMMo-0-w (MC)	0.881 ± 0.026	0.634 ± 0.074	0.747 ± 0.075	0.943 ± 0.037
	LR	0.901 ± 0.003	0.041 ± 0.020	0.623 ± 0.249	0.997 ± 0.002
	SVM	**0.920 ± 0.004**	**0.209 ± 0.041**	**0.977 ± 0.041**	**1.000 ± 0.000**
	NN	0.908 ± 0.004	0.100 ± 0.039	0.834 ± 0.146	0.998 ± 0.002
	ANFIS	0.899 ± 0.001	0.000 ± 0.000	0.000 ± 0.000	0.999 ± 0.001
	ALMMo	0.900 ± 0.000	0.001 ± 0.003	0.080 ± 0.277	**1.000 ± 0.000**
10%	ALMMo-0	**0.901 ± 0.015**	**0.310 ± 0.083**	**0.509 ± 0.116**	0.967 ± 0.010
	ALMMo-0-1C	0.881 ± 0.015	0.251 ± 0.050	0.376 ± 0.092	0.951 ± 0.016
	ALMMo-0-w (GM)	0.675 ± 0.092	**0.718 ± 0.149**	0.221 ± 0.091	0.670 ± 0.117
	ALMMo-0-w (F1)	0.872 ± 0.064	0.396 ± 0.136	0.453 ± 0.139	0.925 ± 0.083
	ALMMo-0-w (KC)	0.898 ± 0.017	0.342 ± 0.078	0.500 ± 0.102	0.959 ± 0.021
	ALMMo-0-w (MC)	**0.899 ± 0.013**	0.337 ± 0.076	**0.508 ± 0.093**	**0.962 ± 0.017**
	LR	0.950 ± 0.001	0.007 ± 0.012	0.280 ± 0.458	**1.000 ± 0.000**
	SVM	**0.953 ± 0.002**	**0.061 ± 0.034**	**0.960 ± 0.200**	**1.000 ± 0.000**
	NN	0.950 ± 0.001	0.007 ± 0.013	0.240 ± 0.436	**1.000 ± 0.000**
	ANFIS	0.949 ± 0.001	0.000 ± 0.000	0.000 ± 0.000	0.999 ± 0.001
	ALMMo	**0.950 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
5%	ALMMo-0	0.942 ± 0.007	**0.240 ± 0.067**	**0.390 ± 0.096**	0.979 ± 0.009
	ALMMo-0-1C	**0.915 ± 0.010**	0.327 ± 0.073	0.243 ± 0.056	0.946 ± 0.010
	ALMMo-0-w (GM)	0.664 ± 0.152	**0.616 ± 0.203**	0.127 ± 0.104	0.667 ± 0.169
	ALMMo-0-w (F1)	0.881 ± 0.107	0.295 ± 0.195	0.261 ± 0.139	0.912 ± 0.122
	ALMMo-0-w (KC)	0.915 ± 0.066	0.236 ± 0.112	**0.324 ± 0.187**	**0.950 ± 0.074**
	ALMMo-0-w (MC)	0.869 ± 0.150	0.309 ± 0.225	0.287 ± 0.190	0.898 ± 0.169
	LR	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	SVM	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	NN	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	ANFIS	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	ALMMo	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
1%	ALMMo-0	0.984 ± 0.005	**0.080 ± 0.094**	**0.134 ± 0.177**	0.993 ± 0.005
	ALMMo-0-1C	**0.934 ± 0.011**	0.267 ± 0.154	**0.043 ± 0.026**	**0.940 ± 0.011**
	ALMMo-0-w (GM)	0.568 ± 0.181	**0.551 ± 0.223**	0.015 ± 0.013	0.568 ± 0.185
	ALMMo-0-w (F1)	0.703 ± 0.272	0.364 ± 0.325	0.019 ± 0.027	0.706 ± 0.278
	ALMMo-0-w (KC)	0.691 ± 0.277	0.373 ± 0.321	0.018 ± 0.026	0.694 ± 0.282
	ALMMo-0-w (MC)	0.635 ± 0.277	0.440 ± 0.334	0.017 ± 0.023	0.637 ± 0.283

Table 5.6: Accuracy, Recall, Precision and Specificity results for the German dataset

Imbalance	Method	GMean	F1Score	Kappa	Matthews
	LR	0.453 ± 0.033	0.309 ± 0.037	0.221 ± 0.036	0.254 ± 0.038
	SVM	**0.760 ± 0.028**	**0.721 ± 0.034**	**0.671 ± 0.037**	**0.699 ± 0.032**
	NN	0.627 ± 0.049	0.530 ± 0.056	0.456 ± 0.056	0.490 ± 0.050
	ANFIS	0.365 ± 0.034	0.222 ± 0.034	0.160 ± 0.029	0.219 ± 0.036
	ALMMo	0.375 ± 0.040	0.233 ± 0.041	0.171 ± 0.034	0.231 ± 0.037
20%	ALMMo-0	**0.795 ± 0.035**	**0.720 ± 0.043**	**0.658 ± 0.051**	**0.662 ± 0.050**
	ALMMo-0-1C	0.523 ± 0.039	0.382 ± 0.045	0.284 ± 0.048	0.306 ± 0.049
	ALMMo-0-w (GM)	**0.800 ± 0.027**	0.657 ± 0.058	0.555 ± 0.092	0.573 ± 0.075
	ALMMo-0-w (F1)	0.773 ± 0.041	0.684 ± 0.049	0.613 ± 0.058	0.618 ± 0.057
	ALMMo-0-w (KC)	0.776 ± 0.035	**0.686 ± 0.047**	**0.615 ± 0.058**	**0.620 ± 0.058**
	ALMMo-0-w (MC)	0.771 ± 0.039	0.681 ± 0.052	0.610 ± 0.065	0.616 ± 0.063
	LR	0.196 ± 0.048	0.076 ± 0.035	0.064 ± 0.033	0.138 ± 0.062
	SVM	**0.455 ± 0.046**	**0.343 ± 0.058**	**0.319 ± 0.056**	**0.431 ± 0.050**
	NN	0.310 ± 0.063	0.177 ± 0.062	0.159 ± 0.058	0.265 ± 0.066
	ANFIS	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	ALMMo	0.008 ± 0.028	0.002 ± 0.005	0.001 ± 0.005	0.008 ± 0.026
10%	ALMMo-0	**0.543 ± 0.075**	**0.384 ± 0.094**	**0.334 ± 0.100**	**0.347 ± 0.102**
	ALMMo-0-1C	0.487 ± 0.049	0.298 ± 0.057	0.236 ± 0.063	0.244 ± 0.067
	ALMMo-0-w (GM)	**0.680 ± 0.056**	0.313 ± 0.034	0.189 ± 0.055	0.256 ± 0.041
	ALMMo-0-w (F1)	0.593 ± 0.074	0.392 ± 0.073	0.328 ± 0.090	0.344 ± 0.083
	ALMMo-0-w (KC)	0.569 ± 0.061	**0.400 ± 0.068**	0.347 ± 0.074	0.357 ± 0.075
	ALMMo-0-w (MC)	0.566 ± 0.061	0.399 ± 0.066	**0.348 ± 0.070**	**0.359 ± 0.071**
	LR	0.043 ± 0.072	0.013 ± 0.023	0.013 ± 0.022	0.042 ± 0.070
	SVM	**0.235 ± 0.078**	**0.113 ± 0.059**	**0.108 ± 0.057**	**0.229 ± 0.076**
	NN	0.040 ± 0.074	0.013 ± 0.026	0.013 ± 0.025	0.039 ± 0.072
	ANFIS	0.000 ± 0.000	0.000 ± 0.000	-0.002 ± 0.003	-0.005 ± 0.005
	ALMMo	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
5%	ALMMo-0	**0.481 ± 0.066**	**0.290 ± 0.059**	**0.262 ± 0.059**	**0.274 ± 0.060**
	ALMMo-0-1C	0.553 ± 0.064	**0.278 ± 0.060**	**0.234 ± 0.064**	**0.237 ± 0.064**
	ALMMo-0-w (GM)	**0.613 ± 0.074**	0.174 ± 0.063	0.100 ± 0.081	0.154 ± 0.068
	ALMMo-0-w (F1)	0.484 ± 0.116	0.220 ± 0.070	0.177 ± 0.082	0.198 ± 0.076
	ALMMo-0-w (KC)	0.458 ± 0.089	0.238 ± 0.081	0.202 ± 0.092	0.220 ± 0.099
	ALMMo-0-w (MC)	0.481 ± 0.109	0.227 ± 0.083	0.186 ± 0.097	0.209 ± 0.095
	LR	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	SVM	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	NN	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	ANFIS	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	ALMMo	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
1%	ALMMo-0	**0.199 ± 0.204**	**0.094 ± 0.116**	**0.086 ± 0.117**	**0.092 ± 0.124**
	ALMMo-0-1C	0.474 ± 0.164	**0.073 ± 0.044**	**0.057 ± 0.044**	**0.085 ± 0.064**
	ALMMo-0-w (GM)	**0.522 ± 0.098**	0.028 ± 0.015	0.009 ± 0.017	0.027 ± 0.031
	ALMMo-0-w (F1)	0.347 ± 0.243	0.028 ± 0.030	0.012 ± 0.031	0.022 ± 0.039
	ALMMo-0-w (KC)	0.357 ± 0.232	0.027 ± 0.029	0.012 ± 0.030	0.020 ± 0.039
	ALMMo-0-w (MC)	0.388 ± 0.217	0.027 ± 0.026	0.010 ± 0.027	0.022 ± 0.036

Table 5.7: Geometric Mean, F1-Score, Cohen´s Kappa and Matthews Correlation Coefficient results for the German dataset

Imbalance	Method	Accuracy	Recall	Precision	Specificity
	LR	0.880 ± 0.008	0.619 ± 0.036	0.738 ± 0.028	0.945 ± 0.009
	SVM	**0.891 ± 0.009**	**0.694 ± 0.042**	**0.746 ± 0.023**	0.941 ± 0.007
	NN	0.849 ± 0.018	0.438 ± 0.129	0.675 ± 0.159	0.952 ± 0.021
	ANFIS	0.877 ± 0.008	0.597 ± 0.031	0.739 ± 0.031	**0.947 ± 0.008**
	ALMMo	0.839 ± 0.039	**0.819 ± 0.064**	0.584 ± 0.082	0.844 ± 0.063
20%	ALMMo-0	**0.891 ± 0.010**	0.720 ± 0.039	**0.734 ± 0.039**	**0.934 ± 0.014**
	ALMMo-0-1C	0.847 ± 0.010	0.323 ± 0.047	**0.789 ± 0.063**	**0.978 ± 0.008**
	ALMMo-0-w (GM)	0.858 ± 0.031	**0.784 ± 0.071**	0.628 ± 0.077	0.877 ± 0.049
	ALMMo-0-w (F1)	0.879 ± 0.015	0.736 ± 0.051	0.687 ± 0.047	0.915 ± 0.019
	ALMMo-0-w (KC)	**0.880 ± 0.015**	0.732 ± 0.048	0.690 ± 0.045	0.917 ± 0.018
	ALMMo-0-w (MC)	0.879 ± 0.015	0.737 ± 0.052	0.686 ± 0.048	0.915 ± 0.020
	LR	0.919 ± 0.008	0.286 ± 0.080	**0.733 ± 0.091**	0.989 ± 0.004
	SVM	**0.923 ± 0.009**	**0.555 ± 0.067**	0.629 ± 0.051	0.963 ± 0.006
	NN	0.901 ± 0.002	0.021 ± 0.029	0.326 ± 0.360	0.998 ± 0.003
	ANFIS	0.901 ± 0.001	0.009 ± 0.009	0.600 ± 0.500	**1.000 ± 0.000**
	ALMMo	0.906 ± 0.015	0.407 ± 0.251	0.599 ± 0.147	**0.961 ± 0.033**
10%	ALMMo-0	**0.918 ± 0.011**	**0.549 ± 0.065**	**0.606 ± 0.072**	0.959 ± 0.015
	ALMMo-0-1C	**0.916 ± 0.006**	0.331 ± 0.055	**0.673 ± 0.073**	**0.982 ± 0.006**
	ALMMo-0-w (GM)	0.809 ± 0.069	**0.731 ± 0.118**	0.344 ± 0.105	0.818 ± 0.086
	ALMMo-0-w (F1)	0.909 ± 0.012	0.534 ± 0.078	0.555 ± 0.074	0.951 ± 0.014
	ALMMo-0-w (KC)	0.909 ± 0.013	0.537 ± 0.080	0.554 ± 0.076	0.950 ± 0.015
	ALMMo-0-w (MC)	0.907 ± 0.012	0.543 ± 0.095	0.546 ± 0.075	0.948 ± 0.017
	LR	0.949 ± 0.003	**0.050 ± 0.041**	0.351 ± 0.252	0.996 ± 0.003
	SVM	**0.951 ± 0.002**	0.047 ± 0.065	**0.369 ± 0.420**	**0.999 ± 0.002**
	NN	0.950 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1.000 ± 0.000
	ANFIS	0.950 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1.000 ± 0.000
	ALMMo	0.949 ± 0.004	0.036 ± 0.051	0.251 ± 0.345	**0.997 ± 0.005**
5%	ALMMo-0	**0.954 ± 0.008**	**0.447 ± 0.073**	**0.563 ± 0.112**	0.980 ± 0.009
	ALMMo-0-1C	**0.945 ± 0.008**	0.318 ± 0.072	0.441 ± 0.117	**0.978 ± 0.008**
	ALMMo-0-w (GM)	0.809 ± 0.098	**0.736 ± 0.129**	0.206 ± 0.087	0.813 ± 0.109
	ALMMo-0-w (F1)	0.945 ± 0.013	0.450 ± 0.079	**0.479 ± 0.116**	0.971 ± 0.015
	ALMMo-0-w (KC)	0.943 ± 0.017	0.454 ± 0.080	0.475 ± 0.126	0.969 ± 0.020
	ALMMo-0-w (MC)	0.940 ± 0.020	0.473 ± 0.110	0.460 ± 0.125	0.965 ± 0.024
	LR	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	SVM	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	NN	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	ANFIS	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	ALMMo	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
1%	ALMMo-0	0.981 ± 0.008	**0.209 ± 0.168**	**0.236 ± 0.250**	0.989 ± 0.008
	ALMMo-0-1C	**0.977 ± 0.010**	0.356 ± 0.147	0.211 ± 0.128	**0.983 ± 0.010**
	ALMMo-0-w (GM)	0.780 ± 0.149	**0.627 ± 0.242**	0.042 ± 0.031	0.782 ± 0.152
	ALMMo-0-w (F1)	0.956 ± 0.067	0.307 ± 0.218	**0.234 ± 0.274**	0.962 ± 0.069
	ALMMo-0-w (KC)	0.953 ± 0.065	0.329 ± 0.219	**0.234 ± 0.274**	0.959 ± 0.067
	ALMMo-0-w (MC)	0.939 ± 0.096	0.338 ± 0.241	0.231 ± 0.277	0.946 ± 0.099

Table 5.8: Accuracy, Recall, Precision and Specificity results for the Mammographic dataset

Imbalance	Method	GMean	F1Score	Kappa	Matthews
	LR	0.765 ± 0.021	0.673 ± 0.024	0.600 ± 0.028	0.604 ± 0.027
	SVM	**0.808 ± 0.024**	**0.718 ± 0.027**	**0.651 ± 0.032**	**0.652 ± 0.031**
	NN	0.628 ± 0.146	0.523 ± 0.131	0.444 ± 0.118	0.462 ± 0.115
	ANFIS	0.752 ± 0.019	0.660 ± 0.024	0.586 ± 0.029	0.591 ± 0.028
	ALMMo	**0.829 ± 0.017**	0.675 ± 0.040	0.574 ± 0.062	0.594 ± 0.046
20%	ALMMo-0	0.820 ± 0.020	**0.726 ± 0.024**	**0.658 ± 0.030**	**0.659 ± 0.030**
	ALMMo-0-1C	0.560 ± 0.041	0.456 ± 0.050	0.385 ± 0.051	0.438 ± 0.047
	ALMMo-0-w (GM)	**0.827 ± 0.027**	0.691 ± 0.040	0.602 ± 0.058	0.613 ± 0.049
	ALMMo-0-w (F1)	0.820 ± 0.027	**0.709 ± 0.034**	**0.633 ± 0.042**	**0.635 ± 0.042**
	ALMMo-0-w (KC)	0.820 ± 0.027	**0.709 ± 0.033**	**0.633 ± 0.042**	**0.635 ± 0.042**
	ALMMo-0-w (MC)	0.820 ± 0.027	**0.709 ± 0.034**	**0.633 ± 0.043**	**0.635 ± 0.042**
	LR	0.526 ± 0.081	0.408 ± 0.096	0.374 ± 0.095	0.423 ± 0.090
	SVM	**0.730 ± 0.045**	**0.588 ± 0.054**	**0.546 ± 0.058**	**0.548 ± 0.057**
	NN	0.099 ± 0.107	0.038 ± 0.051	0.032 ± 0.044	0.069 ± 0.081
	ANFIS	0.071 ± 0.063	0.017 ± 0.018	0.016 ± 0.016	0.067 ± 0.060
	ALMMo	0.580 ± 0.222	0.416 ± 0.196	0.375 ± 0.183	0.405 ± 0.153
10%	ALMMo-0	**0.724 ± 0.040**	**0.571 ± 0.043**	**0.526 ± 0.048**	**0.530 ± 0.048**
	ALMMo-0-1C	0.568 ± 0.047	0.440 ± 0.052	0.401 ± 0.051	0.432 ± 0.047
	ALMMo-0-w (GM)	**0.767 ± 0.048**	0.449 ± 0.074	0.361 ± 0.097	0.407 ± 0.077
	ALMMo-0-w (F1)	0.710 ± 0.049	**0.540 ± 0.053**	**0.490 ± 0.058**	0.492 ± 0.060
	ALMMo-0-w (KC)	0.712 ± 0.050	0.540 ± 0.055	0.490 ± 0.060	**0.493 ± 0.062**
	ALMMo-0-w (MC)	0.715 ± 0.055	0.538 ± 0.051	0.487 ± 0.056	0.491 ± 0.059
	LR	**0.192 ± 0.116**	**0.086 ± 0.069**	**0.077 ± 0.066**	**0.117 ± 0.095**
	SVM	0.142 ± 0.168	0.079 ± 0.105	0.074 ± 0.099	0.114 ± 0.133
	NN	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	ANFIS	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	ALMMo	0.125 ± 0.146	0.058 ± 0.078	0.052 ± 0.071	0.078 ± 0.098
5%	ALMMo-0	**0.660 ± 0.054**	**0.490 ± 0.060**	**0.467 ± 0.063**	**0.474 ± 0.065**
	ALMMo-0-1C	0.554 ± 0.066	0.365 ± 0.081	0.338 ± 0.084	0.345 ± 0.087
	ALMMo-0-w (GM)	**0.765 ± 0.044**	0.305 ± 0.079	0.246 ± 0.093	0.316 ± 0.067
	ALMMo-0-w (F1)	0.658 ± 0.055	**0.453 ± 0.062**	**0.424 ± 0.067**	**0.431 ± 0.068**
	ALMMo-0-w (KC)	0.660 ± 0.054	0.449 ± 0.061	0.420 ± 0.067	0.428 ± 0.066
	ALMMo-0-w (MC)	0.671 ± 0.069	0.447 ± 0.060	0.417 ± 0.066	0.427 ± 0.064
	LR	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	SVM	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	NN	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	ANFIS	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	ALMMo	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
1%	ALMMo-0	**0.392 ± 0.234**	**0.190 ± 0.153**	**0.182 ± 0.155**	**0.196 ± 0.165**
	ALMMo-0-1C	0.577 ± 0.130	**0.249 ± 0.120**	**0.240 ± 0.123**	**0.255 ± 0.122**
	ALMMo-0-w (GM)	**0.676 ± 0.141**	0.075 ± 0.049	0.057 ± 0.051	0.117 ± 0.072
	ALMMo-0-w (F1)	0.493 ± 0.219	0.209 ± 0.184	0.199 ± 0.188	0.220 ± 0.195
	ALMMo-0-w (KC)	0.512 ± 0.221	0.213 ± 0.184	0.203 ± 0.187	0.227 ± 0.193
	ALMMo-0-w (MC)	0.507 ± 0.226	0.204 ± 0.187	0.194 ± 0.190	0.218 ± 0.194

Table 5.9: Geometric Mean, F1-Score, Cohen´s Kappa and Matthews Correlation Coefficient results for the Mammographic dataset

Imbalance	Method	Accuracy	Recall	Precision	Specificity
	LR	0.841 ± 0.006	0.370 ± 0.021	0.690 ± 0.030	0.958 ± 0.005
	SVM	**0.865 ± 0.007**	**0.429 ± 0.025**	**0.809 ± 0.042**	**0.974 ± 0.007**
	NN	0.841 ± 0.007	0.354 ± 0.052	0.709 ± 0.043	0.963 ± 0.010
	ANFIS	0.838 ± 0.007	0.297 ± 0.031	0.735 ± 0.033	0.973 ± 0.004
	ALMMo	0.696 ± 0.169	0.722 ± 0.210	0.455 ± 0.148	0.689 ± 0.260
20%	ALMMo-0	**0.950 ± 0.007**	**0.823 ± 0.032**	**0.920 ± 0.027**	**0.982 ± 0.007**
	ALMMo-0-1C	0.895 ± 0.013	0.583 ± 0.048	0.845 ± 0.046	**0.973 ± 0.009**
	ALMMo-0-w (GM)	0.913 ± 0.021	**0.900 ± 0.034**	0.735 ± 0.069	0.916 ± 0.029
	ALMMo-0-w (F1)	0.938 ± 0.014	0.825 ± 0.042	0.864 ± 0.057	0.966 ± 0.018
	ALMMo-0-w (KC)	**0.940 ± 0.013**	0.824 ± 0.038	0.872 ± 0.058	0.968 ± 0.017
	ALMMo-0-w (MC)	**0.940 ± 0.013**	0.818 ± 0.041	**0.876 ± 0.054**	0.970 ± 0.016
	LR	0.902 ± 0.005	0.116 ± 0.036	0.549 ± 0.102	0.990 ± 0.003
	SVM	**0.912 ± 0.005**	**0.132 ± 0.046**	**0.907 ± 0.117**	0.998 ± 0.002
	NN	0.901 ± 0.003	0.035 ± 0.041	0.345 ± 0.309	0.997 ± 0.003
	ANFIS	0.901 ± 0.001	0.011 ± 0.009	0.616 ± 0.441	**0.999 ± 0.001**
	ALMMo	0.878 ± 0.072	0.246 ± 0.239	0.536 ± 0.254	0.948 ± 0.102
10%	ALMMo-0	**0.947 ± 0.009**	**0.589 ± 0.063**	**0.840 ± 0.077**	**0.987 ± 0.007**
	ALMMo-0-1C	0.934 ± 0.007	0.554 ± 0.054	0.729 ± 0.062	0.977 ± 0.007
	ALMMo-0-w (GM)	0.837 ± 0.055	**0.800 ± 0.079**	0.376 ± 0.075	0.841 ± 0.065
	ALMMo-0-w (F1)	0.937 ± 0.012	0.565 ± 0.100	0.762 ± 0.088	**0.979 ± 0.014**
	ALMMo-0-w (KC)	**0.937 ± 0.011**	0.568 ± 0.089	0.752 ± 0.081	0.978 ± 0.012
	ALMMo-0-w (MC)	0.937 ± 0.015	0.557 ± 0.086	**0.770 ± 0.100**	0.979 ± 0.018
	LR	**0.951 ± 0.001**	**0.017 ± 0.020**	**0.453 ± 0.480**	**1.000 ± 0.000**
	SVM	0.950 ± 0.001	0.003 ± 0.007	0.160 ± 0.374	**1.000 ± 0.000**
	NN	0.950 ± 0.001	0.001 ± 0.004	0.040 ± 0.200	**1.000 ± 0.000**
	ANFIS	0.950 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	ALMMo	0.950 ± 0.002	0.021 ± 0.038	0.218 ± 0.346	**0.999 ± 0.003**
5%	ALMMo-0	**0.952 ± 0.006**	**0.200 ± 0.056**	**0.577 ± 0.152**	0.991 ± 0.005
	ALMMo-0-1C	**0.956 ± 0.007**	0.569 ± 0.069	**0.567 ± 0.078**	**0.977 ± 0.007**
	ALMMo-0-w (GM)	0.718 ± 0.093	**0.640 ± 0.138**	0.123 ± 0.058	0.722 ± 0.104
	ALMMo-0-w (F1)	0.933 ± 0.024	0.257 ± 0.115	0.390 ± 0.163	0.968 ± 0.031
	ALMMo-0-w (KC)	0.939 ± 0.020	0.235 ± 0.112	0.428 ± 0.151	0.976 ± 0.026
	ALMMo-0-w (MC)	0.929 ± 0.063	0.250 ± 0.155	0.452 ± 0.185	0.965 ± 0.073
	LR	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	SVM	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	NN	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	ANFIS	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
	ALMMo	**0.990 ± 0.000**	0.000 ± 0.000	0.000 ± 0.000	**1.000 ± 0.000**
1%	ALMMo-0	0.988 ± 0.002	**0.095 ± 0.083**	**0.267 ± 0.312**	0.997 ± 0.003
	ALMMo-0-1C	**0.969 ± 0.009**	**0.589 ± 0.134**	**0.196 ± 0.080**	**0.973 ± 0.009**
	ALMMo-0-w (GM)	0.747 ± 0.153	0.484 ± 0.233	0.021 ± 0.012	0.749 ± 0.156
	ALMMo-0-w (F1)	0.905 ± 0.162	0.277 ± 0.250	0.170 ± 0.266	0.911 ± 0.166
	ALMMo-0-w (KC)	0.889 ± 0.178	0.293 ± 0.254	0.175 ± 0.264	0.894 ± 0.182
	ALMMo-0-w (MC)	0.913 ± 0.140	0.249 ± 0.231	0.166 ± 0.267	0.919 ± 0.143

Table 5.10: Accuracy, Recall, Precision and Specificity results for the Pima dataset

Imbalance	Method	GMean	F1Score	Kappa	Matthews
	LR	0.595 ± 0.017	0.481 ± 0.023	0.397 ± 0.025	0.424 ± 0.026
	SVM	**0.646 ± 0.019**	**0.560 ± 0.025**	**0.489 ± 0.028**	**0.524 ± 0.029**
	NN	0.582 ± 0.042	0.469 ± 0.047	0.388 ± 0.044	0.422 ± 0.036
	ANFIS	0.537 ± 0.028	0.423 ± 0.034	0.348 ± 0.034	0.396 ± 0.032
	ALMMo	0.664 ± 0.088	0.511 ± 0.065	0.345 ± 0.134	0.393 ± 0.087
20%	ALMMo-0	**0.899 ± 0.017**	**0.868 ± 0.019**	**0.838 ± 0.023**	**0.840 ± 0.023**
	ALMMo-0-1C	0.752 ± 0.032	0.689 ± 0.043	0.628 ± 0.049	0.645 ± 0.047
	ALMMo-0-w (GM)	**0.908 ± 0.016**	0.807 ± 0.037	0.752 ± 0.051	0.760 ± 0.046
	ALMMo-0-w (F1)	0.893 ± 0.021	0.842 ± 0.032	0.804 ± 0.041	0.805 ± 0.041
	ALMMo-0-w (KC)	0.893 ± 0.019	**0.845 ± 0.030**	**0.808 ± 0.038**	**0.810 ± 0.038**
	ALMMo-0-w (MC)	0.890 ± 0.021	0.844 ± 0.030	0.807 ± 0.038	0.809 ± 0.038
	LR	0.335 ± 0.052	0.190 ± 0.054	0.161 ± 0.052	0.218 ± 0.059
	SVM	**0.358 ± 0.065**	**0.228 ± 0.070**	**0.208 ± 0.066**	**0.322 ± 0.068**
	NN	0.137 ± 0.129	0.062 ± 0.070	0.052 ± 0.062	0.091 ± 0.095
	ANFIS	0.087 ± 0.059	0.021 ± 0.017	0.018 ± 0.015	0.071 ± 0.055
	ALMMo	0.406 ± 0.223	0.241 ± 0.157	0.200 ± 0.133	0.246 ± 0.119
10%	ALMMo-0	**0.761 ± 0.040**	**0.690 ± 0.054**	**0.662 ± 0.058**	**0.676 ± 0.058**
	ALMMo-0-1C	0.734 ± 0.037	0.627 ± 0.045	0.592 ± 0.048	0.600 ± 0.047
	ALMMo-0-w (GM)	**0.818 ± 0.036**	0.505 ± 0.067	0.426 ± 0.084	0.472 ± 0.068
	ALMMo-0-w (F1)	0.741 ± 0.063	0.641 ± 0.069	**0.608 ± 0.074**	**0.620 ± 0.069**
	ALMMo-0-w (KC)	0.743 ± 0.057	**0.641 ± 0.062**	0.607 ± 0.066	0.618 ± 0.062
	ALMMo-0-w (MC)	0.736 ± 0.055	0.639 ± 0.067	0.606 ± 0.073	0.620 ± 0.070
	LR	**0.092 ± 0.095**	**0.033 ± 0.038**	**0.031 ± 0.036**	**0.083 ± 0.091**
	SVM	0.022 ± 0.051	0.006 ± 0.014	0.005 ± 0.013	0.021 ± 0.050
	NN	0.005 ± 0.027	0.001 ± 0.007	0.001 ± 0.007	0.004 ± 0.027
	ANFIS	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	ALMMo	0.082 ± 0.122	0.036 ± 0.062	0.032 ± 0.057	0.057 ± 0.087
5%	ALMMo-0	**0.441 ± 0.063**	**0.292 ± 0.073**	**0.273 ± 0.074**	**0.317 ± 0.080**
	ALMMo-0-1C	**0.744 ± 0.045**	**0.565 ± 0.061**	**0.542 ± 0.065**	**0.544 ± 0.065**
	ALMMo-0-w (GM)	0.669 ± 0.059	0.193 ± 0.033	0.119 ± 0.043	0.182 ± 0.037
	ALMMo-0-w (F1)	0.487 ± 0.096	0.273 ± 0.052	0.242 ± 0.053	0.265 ± 0.060
	ALMMo-0-w (KC)	0.467 ± 0.097	0.272 ± 0.058	0.245 ± 0.057	0.272 ± 0.061
	ALMMo-0-w (MC)	0.470 ± 0.107	0.271 ± 0.066	0.244 ± 0.071	0.279 ± 0.076
	LR	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	SVM	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	NN	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	ANFIS	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
	ALMMo	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
1%	ALMMo-0	**0.249 ± 0.185**	**0.125 ± 0.102**	**0.121 ± 0.101**	**0.142 ± 0.122**
	ALMMo-0-1C	**0.752 ± 0.088**	**0.288 ± 0.092**	**0.277 ± 0.094**	**0.324 ± 0.092**
	ALMMo-0-w (GM)	0.550 ± 0.187	0.040 ± 0.021	0.022 ± 0.020	0.058 ± 0.040
	ALMMo-0-w (F1)	0.392 ± 0.261	0.128 ± 0.144	0.118 ± 0.147	0.141 ± 0.154
	ALMMo-0-w (KC)	0.407 ± 0.246	0.133 ± 0.143	0.123 ± 0.146	0.144 ± 0.153
	ALMMo-0-w (MC)	0.367 ± 0.267	0.123 ± 0.146	0.114 ± 0.149	0.134 ± 0.158

Table 5.11: Geometric Mean, F1-Score, Cohen´s Kappa and Matthews Correlation Coefficient results for the Pima dataset

Imbalance	Method	Accuracy	Recall	Precision	Specificity
	LR	0.985 ± 0.004	0.926 ± 0.019	0.997 ± 0.008	0.999 ± 0.002
	SVM	**0.994 ± 0.002**	**0.970 ± 0.010**	**1.000 ± 0.000**	**1.000 ± 0.000**
	NN	0.989 ± 0.003	0.948 ± 0.014	0.999 ± 0.003	**1.000 ± 0.000**
	ANFIS	0.963 ± 0.007	0.822 ± 0.032	0.991 ± 0.011	0.998 ± 0.002
	ALMMo	0.946 ± 0.063	0.916 ± 0.195	0.841 ± 0.234	0.954 ± 0.074
20%	ALMMo-0	**0.980 ± 0.005**	**0.917 ± 0.017**	**0.983 ± 0.018**	**0.996 ± 0.004**
	ALMMo-0-1C	0.906 ± 0.021	0.667 ± 0.076	0.831 ± 0.068	0.965 ± 0.017
	ALMMo-0-w (GM)	0.971 ± 0.012	**0.949 ± 0.029**	0.914 ± 0.064	0.976 ± 0.019
	ALMMo-0-w (F1)	**0.981 ± 0.006**	0.930 ± 0.025	0.973 ± 0.027	**0.993 ± 0.007**
	ALMMo-0-w (KC)	**0.981 ± 0.006**	0.929 ± 0.025	**0.973 ± 0.026**	**0.993 ± 0.007**
	ALMMo-0-w (MC)	0.980 ± 0.007	0.930 ± 0.026	0.969 ± 0.034	0.992 ± 0.009
	LR	0.986 ± 0.005	0.863 ± 0.050	**1.000 ± 0.000**	**1.000 ± 0.000**
	SVM	**0.994 ± 0.002**	**0.944 ± 0.025**	**1.000 ± 0.000**	**1.000 ± 0.000**
	NN	0.989 ± 0.005	0.895 ± 0.046	**1.000 ± 0.000**	**1.000 ± 0.000**
	ANFIS	0.973 ± 0.006	0.731 ± 0.059	**1.000 ± 0.000**	**1.000 ± 0.000**
	ALMMo	**0.984 ± 0.010**	**0.902 ± 0.047**	0.944 ± 0.089	0.993 ± 0.013
10%	ALMMo-0	0.983 ± 0.003	0.851 ± 0.028	**0.975 ± 0.021**	**0.997 ± 0.002**
	ALMMo-0-1C	0.929 ± 0.021	0.582 ± 0.191	0.654 ± 0.106	0.967 ± 0.010
	ALMMo-0-w (GM)	0.951 ± 0.031	**0.895 ± 0.048**	0.744 ± 0.170	0.957 ± 0.038
	ALMMo-0-w (F1)	0.980 ± 0.006	0.846 ± 0.040	0.948 ± 0.047	0.995 ± 0.005
	ALMMo-0-w (KC)	**0.980 ± 0.005**	0.845 ± 0.039	**0.952 ± 0.040**	**0.995 ± 0.004**
	ALMMo-0-w (MC)	**0.980 ± 0.005**	0.847 ± 0.040	0.951 ± 0.040	**0.995 ± 0.004**
	LR	0.989 ± 0.003	0.787 ± 0.051	**1.000 ± 0.000**	**1.000 ± 0.000**
	SVM	**0.995 ± 0.002**	**0.907 ± 0.033**	**1.000 ± 0.000**	**1.000 ± 0.000**
	NN	0.991 ± 0.003	0.817 ± 0.066	**1.000 ± 0.000**	**1.000 ± 0.000**
	ANFIS	0.956 ± 0.007	0.156 ± 0.090	0.799 ± 0.308	0.998 ± 0.004
	ALMMo	**0.987 ± 0.004**	**0.746 ± 0.081**	**1.000 ± 0.000**	**1.000 ± 0.000**
5%	ALMMo-0	0.985 ± 0.005	0.739 ± 0.067	0.955 ± 0.058	0.998 ± 0.003
	ALMMo-0-1C	0.948 ± 0.014	0.592 ± 0.149	0.489 ± 0.118	0.966 ± 0.012
	ALMMo-0-w (GM)	0.942 ± 0.040	**0.807 ± 0.080**	0.561 ± 0.237	0.950 ± 0.044
	ALMMo-0-w (F1)	0.984 ± 0.005	0.732 ± 0.068	0.947 ± 0.067	**0.998 ± 0.003**
	ALMMo-0-w (KC)	**0.985 ± 0.005**	0.734 ± 0.067	**0.948 ± 0.068**	**0.998 ± 0.003**
	ALMMo-0-w (MC)	0.984 ± 0.005	0.734 ± 0.069	0.943 ± 0.068	**0.998 ± 0.003**
	LR	0.996 ± 0.002	0.550 ± 0.210	**1.000 ± 0.000**	**1.000 ± 0.000**
	SVM	**0.998 ± 0.001**	**0.780 ± 0.146**	**1.000 ± 0.000**	**1.000 ± 0.000**
	NN	0.996 ± 0.002	0.560 ± 0.181	**1.000 ± 0.000**	**1.000 ± 0.000**
	ANFIS	0.990 ± 0.001	0.010 ± 0.035	0.027 ± 0.092	0.999 ± 0.001
	ALMMo	0.995 ± 0.002	0.460 ± 0.164	**1.000 ± 0.000**	**1.000 ± 0.000**
1%	ALMMo-0	**0.996 ± 0.002**	**0.645 ± 0.233**	0.967 ± 0.078	**1.000 ± 0.000**
	ALMMo-0-1C	0.966 ± 0.013	0.565 ± 0.188	0.174 ± 0.070	0.970 ± 0.013
	ALMMo-0-w (GM)	0.982 ± 0.038	**0.685 ± 0.217**	0.745 ± 0.348	0.985 ± 0.038
	ALMMo-0-w (F1)	**0.995 ± 0.004**	0.645 ± 0.236	**0.847 ± 0.238**	**0.998 ± 0.003**
	ALMMo-0-w (KC)	0.994 ± 0.004	0.650 ± 0.242	0.808 ± 0.256	0.998 ± 0.004
	ALMMo-0-w (MC)	0.993 ± 0.012	0.650 ± 0.242	0.821 ± 0.269	0.996 ± 0.012

Table 5.12: Accuracy, Recall, Precision and Specificity results for the WBCD dataset

Imbalance	Method	GMean	F1Score	Kappa	Matthews
	LR	0.962 ± 0.010	0.960 ± 0.011	0.951 ± 0.013	0.952 ± 0.013
	SVM	**0.985 ± 0.005**	**0.985 ± 0.005**	**0.981 ± 0.007**	**0.981 ± 0.006**
	NN	0.973 ± 0.007	0.973 ± 0.008	0.966 ± 0.009	0.967 ± 0.009
	ANFIS	0.906 ± 0.018	0.899 ± 0.020	0.876 ± 0.024	0.882 ± 0.022
	ALMMo	0.914 ± 0.193	0.866 ± 0.202	0.838 ± 0.214	0.848 ± 0.205
20%	ALMMo-0	**0.955 ± 0.009**	**0.949 ± 0.012**	**0.936 ± 0.015**	**0.937 ± 0.015**
	ALMMo-0-1C	0.801 ± 0.046	0.737 ± 0.060	0.681 ± 0.071	0.689 ± 0.070
	ALMMo-0-w (GM)	**0.962 ± 0.010**	0.929 ± 0.026	0.910 ± 0.033	0.912 ± 0.031
	ALMMo-0-w (F1)	0.961 ± 0.012	**0.951 ± 0.015**	**0.939 ± 0.019**	0.939 ± 0.019
	ALMMo-0-w (KC)	0.961 ± 0.012	0.950 ± 0.015	0.938 ± 0.018	**0.939 ± 0.018**
	ALMMo-0-w (MC)	0.960 ± 0.012	0.948 ± 0.016	0.936 ± 0.021	0.937 ± 0.020
	LR	0.929 ± 0.027	0.926 ± 0.029	0.918 ± 0.031	0.922 ± 0.029
	SVM	**0.972 ± 0.013**	**0.971 ± 0.013**	**0.968 ± 0.014**	**0.969 ± 0.014**
	NN	0.946 ± 0.025	0.944 ± 0.026	0.938 ± 0.028	0.940 ± 0.027
	ANFIS	0.854 ± 0.035	0.843 ± 0.040	0.829 ± 0.042	0.842 ± 0.037
	ALMMo	**0.946 ± 0.022**	**0.918 ± 0.040**	**0.909 ± 0.046**	**0.912 ± 0.043**
10%	ALMMo-0	0.921 ± 0.015	0.908 ± 0.016	0.899 ± 0.017	0.901 ± 0.017
	ALMMo-0-1C	0.740 ± 0.130	0.607 ± 0.150	0.569 ± 0.160	0.575 ± 0.155
	ALMMo-0-w (GM)	**0.924 ± 0.018**	0.798 ± 0.094	0.771 ± 0.110	0.784 ± 0.095
	ALMMo-0-w (F1)	0.917 ± 0.022	0.893 ± 0.030	0.882 ± 0.033	0.884 ± 0.033
	ALMMo-0-w (KC)	0.917 ± 0.021	**0.895 ± 0.028**	**0.884 ± 0.031**	**0.886 ± 0.031**
	ALMMo-0-w (MC)	0.918 ± 0.022	**0.895 ± 0.028**	**0.884 ± 0.031**	**0.886 ± 0.031**
	LR	0.887 ± 0.029	0.880 ± 0.033	0.874 ± 0.034	0.882 ± 0.030
	SVM	**0.952 ± 0.018**	**0.951 ± 0.018**	**0.948 ± 0.019**	**0.950 ± 0.018**
	NN	0.903 ± 0.037	0.898 ± 0.041	0.893 ± 0.042	0.899 ± 0.038
	ANFIS	0.374 ± 0.127	0.254 ± 0.134	0.242 ± 0.134	0.330 ± 0.153
	ALMMo	**0.862 ± 0.048**	**0.852 ± 0.056**	**0.845 ± 0.058**	**0.857 ± 0.050**
5%	ALMMo-0	0.858 ± 0.039	0.832 ± 0.055	0.824 ± 0.057	0.832 ± 0.056
	ALMMo-0-1C	0.750 ± 0.101	0.529 ± 0.118	0.502 ± 0.124	0.508 ± 0.125
	ALMMo-0-w (GM)	**0.874 ± 0.038**	0.627 ± 0.149	0.601 ± 0.164	0.631 ± 0.139
	ALMMo-0-w (F1)	0.854 ± 0.040	0.824 ± 0.056	0.816 ± 0.058	0.824 ± 0.057
	ALMMo-0-w (KC)	0.855 ± 0.039	**0.825 ± 0.055**	**0.817 ± 0.058**	**0.826 ± 0.057**
	ALMMo-0-w (MC)	0.855 ± 0.040	0.823 ± 0.056	0.815 ± 0.059	0.823 ± 0.057
	LR	0.728 ± 0.143	0.687 ± 0.175	0.685 ± 0.175	0.727 ± 0.143
	SVM	**0.879 ± 0.083**	**0.869 ± 0.092**	**0.868 ± 0.093**	**0.879 ± 0.083**
	NN	0.738 ± 0.126	0.701 ± 0.155	0.699 ± 0.155	0.737 ± 0.126
	ANFIS	0.028 ± 0.098	0.015 ± 0.050	0.014 ± 0.049	0.015 ± 0.055
	ALMMo	0.666 ± 0.128	0.613 ± 0.161	0.611 ± 0.161	0.665 ± 0.128
1%	ALMMo-0	**0.787 ± 0.161**	**0.745 ± 0.190**	**0.744 ± 0.191**	**0.772 ± 0.159**
	ALMMo-0-1C	0.730 ± 0.124	0.258 ± 0.092	0.247 ± 0.093	0.296 ± 0.101
	ALMMo-0-w (GM)	**0.808 ± 0.146**	0.644 ± 0.268	0.640 ± 0.273	0.670 ± 0.251
	ALMMo-0-w (F1)	0.785 ± 0.169	**0.697 ± 0.214**	**0.694 ± 0.215**	**0.717 ± 0.199**
	ALMMo-0-w (KC)	0.787 ± 0.172	0.682 ± 0.220	0.679 ± 0.221	0.701 ± 0.207
	ALMMo-0-w (MC)	0.787 ± 0.172	0.678 ± 0.232	0.675 ± 0.234	0.701 ± 0.214

Table 5.13: Geometric Mean, F1-Score, Cohen´s Kappa and Matthews Correlation Coefficient results for the WBCD dataset

Imbalance	Method	Accuracy	Recall	Precision	Specificity
	LR	0.966 ± 0.009	0.908 ± 0.043	**0.923 ± 0.035**	**0.981 ± 0.010**
	SVM	**0.973 ± 0.007**	**0.947 ± 0.034**	0.922 ± 0.035	0.979 ± 0.010
	NN	0.967 ± 0.008	0.922 ± 0.048	0.917 ± 0.039	0.978 ± 0.012
	ANFIS	0.965 ± 0.009	0.907 ± 0.050	0.920 ± 0.034	0.980 ± 0.010
	ALMMo	**0.956 ± 0.017**	**0.989 ± 0.011**	0.831 ± 0.062	0.948 ± 0.022
20%	ALMMo-0	0.950 ± 0.015	0.779 ± 0.069	**0.968 ± 0.042**	**0.993 ± 0.010**
	ALMMo-0-1C	0.915 ± 0.040	0.875 ± 0.061	0.770 ± 0.133	0.925 ± 0.053
	ALMMo-0-w (GM)	0.950 ± 0.023	**0.916 ± 0.063**	0.861 ± 0.101	0.958 ± 0.036
	ALMMo-0-w (F1)	**0.958 ± 0.015**	0.880 ± 0.063	**0.916 ± 0.066**	**0.978 ± 0.020**
	ALMMo-0-w (KC)	0.956 ± 0.016	0.875 ± 0.068	0.913 ± 0.077	0.976 ± 0.025
	ALMMo-0-w (MC)	0.956 ± 0.016	0.874 ± 0.076	0.915 ± 0.077	0.977 ± 0.024
	LR	0.974 ± 0.008	0.860 ± 0.066	0.877 ± 0.045	0.986 ± 0.006
	SVM	**0.990 ± 0.005**	**0.920 ± 0.049**	**0.977 ± 0.033**	**0.997 ± 0.004**
	NN	0.973 ± 0.008	0.871 ± 0.068	0.861 ± 0.047	0.984 ± 0.007
	ANFIS	0.975 ± 0.008	0.879 ± 0.065	0.870 ± 0.041	0.985 ± 0.006
	ALMMo	0.959 ± 0.018	**0.982 ± 0.020**	0.728 ± 0.086	0.957 ± 0.021
10%	ALMMo-0	**0.977 ± 0.006**	0.769 ± 0.059	**0.998 ± 0.007**	**1.000 ± 0.000**
	ALMMo-0-1C	0.982 ± 0.009	**0.967 ± 0.024**	0.875 ± 0.072	0.984 ± 0.010
	ALMMo-0-w (GM)	0.973 ± 0.015	0.922 ± 0.068	0.850 ± 0.115	0.979 ± 0.020
	ALMMo-0-w (F1)	**0.982 ± 0.008**	0.882 ± 0.075	**0.938 ± 0.057**	**0.993 ± 0.007**
	ALMMo-0-w (KC)	0.980 ± 0.010	0.883 ± 0.078	0.927 ± 0.082	0.991 ± 0.013
	ALMMo-0-w (MC)	0.981 ± 0.007	0.881 ± 0.077	0.934 ± 0.062	0.992 ± 0.008
	LR	0.986 ± 0.004	0.814 ± 0.074	0.900 ± 0.043	0.995 ± 0.002
	SVM	**0.997 ± 0.002**	**0.934 ± 0.042**	**1.000 ± 0.000**	**1.000 ± 0.000**
	NN	0.981 ± 0.005	0.803 ± 0.086	0.810 ± 0.051	0.990 ± 0.003
	ANFIS	0.980 ± 0.005	0.805 ± 0.086	0.804 ± 0.059	0.989 ± 0.004
	ALMMo	0.951 ± 0.095	**0.972 ± 0.014**	0.616 ± 0.137	0.950 ± 0.101
5%	ALMMo-0	**0.988 ± 0.003**	0.768 ± 0.066	**1.000 ± 0.000**	**1.000 ± 0.000**
	ALMMo-0-1C	**0.998 ± 0.002**	**0.997 ± 0.008**	0.974 ± 0.043	0.999 ± 0.003
	ALMMo-0-w (GM)	0.975 ± 0.014	0.925 ± 0.066	0.715 ± 0.142	0.977 ± 0.016
	ALMMo-0-w (F1)	0.990 ± 0.003	0.848 ± 0.074	0.960 ± 0.061	0.998 ± 0.004
	ALMMo-0-w (KC)	0.990 ± 0.004	0.844 ± 0.069	0.966 ± 0.067	0.998 ± 0.005
	ALMMo-0-w (MC)	0.990 ± 0.003	0.823 ± 0.075	**0.978 ± 0.034**	**0.999 ± 0.002**
	LR	0.995 ± 0.001	0.610 ± 0.141	0.857 ± 0.135	0.999 ± 0.001
	SVM	**1.000 ± 0.000**	**0.958 ± 0.038**	**1.000 ± 0.000**	**1.000 ± 0.000**
	NN	0.995 ± 0.002	0.648 ± 0.129	0.890 ± 0.144	0.999 ± 0.001
	ANFIS	0.992 ± 0.001	0.173 ± 0.054	**1.000 ± 0.000**	**1.000 ± 0.000**
	ALMMo	0.985 ± 0.011	**0.898 ± 0.085**	0.488 ± 0.229	0.986 ± 0.012
1%	ALMMo-0	**0.998 ± 0.001**	0.761 ± 0.066	**1.000 ± 0.000**	**1.000 ± 0.000**
	ALMMo-0-1C	0.989 ± 0.016	**1.000 ± 0.000**	0.721 ± 0.352	0.989 ± 0.016
	ALMMo-0-w (GM)	0.976 ± 0.018	0.938 ± 0.058	0.378 ± 0.245	0.976 ± 0.019
	ALMMo-0-w (F1)	0.998 ± 0.001	0.828 ± 0.049	0.987 ± 0.053	**1.000 ± 0.000**
	ALMMo-0-w (KC)	0.998 ± 0.001	0.817 ± 0.048	0.988 ± 0.048	**1.000 ± 0.000**
	ALMMo-0-w (MC)	**0.998 ± 0.000**	0.817 ± 0.053	**0.998 ± 0.013**	**1.000 ± 0.000**

Table 5.14: Accuracy, Recall, Precision and Specificity results for the WBCO dataset

Imbalance	Method	GMean	F1Score	Kappa	Matthews
	LR	0.943 ± 0.021	0.914 ± 0.023	0.893 ± 0.028	0.894 ± 0.028
	SVM	**0.963 ± 0.015**	**0.934 ± 0.018**	**0.917 ± 0.022**	**0.918 ± 0.022**
	NN	0.949 ± 0.022	0.918 ± 0.021	0.897 ± 0.025	0.898 ± 0.025
	ANFIS	0.942 ± 0.024	0.912 ± 0.024	0.891 ± 0.029	0.892 ± 0.029
	ALMMo	**0.968 ± 0.011**	**0.902 ± 0.036**	**0.874 ± 0.047**	**0.881 ± 0.042**
20%	ALMMo-0	0.879 ± 0.040	0.861 ± 0.047	0.831 ± 0.055	0.840 ± 0.050
	ALMMo-0-1C	0.899 ± 0.033	0.811 ± 0.074	0.758 ± 0.099	0.767 ± 0.092
	ALMMo-0-w (GM)	**0.936 ± 0.027**	0.882 ± 0.047	0.850 ± 0.062	0.856 ± 0.057
	ALMMo-0-w (F1)	0.927 ± 0.030	**0.894 ± 0.036**	**0.868 ± 0.045**	**0.871 ± 0.044**
	ALMMo-0-w (KC)	0.923 ± 0.031	0.889 ± 0.038	0.862 ± 0.047	0.866 ± 0.044
	ALMMo-0-w (MC)	0.923 ± 0.036	0.889 ± 0.039	0.861 ± 0.049	0.866 ± 0.046
	LR	0.920 ± 0.035	0.866 ± 0.042	0.852 ± 0.046	0.853 ± 0.046
	SVM	**0.957 ± 0.025**	**0.946 ± 0.028**	**0.941 ± 0.031**	**0.942 ± 0.030**
	NN	0.925 ± 0.036	0.864 ± 0.040	0.849 ± 0.044	0.850 ± 0.044
	ANFIS	0.930 ± 0.034	0.873 ± 0.039	0.859 ± 0.043	0.860 ± 0.043
	ALMMo	**0.969 ± 0.010**	0.833 ± 0.055	0.811 ± 0.064	0.824 ± 0.053
10%	ALMMo-0	0.876 ± 0.034	**0.867 ± 0.039**	**0.855 ± 0.042**	**0.864 ± 0.036**
	ALMMo-0-1C	**0.975 ± 0.011**	**0.916 ± 0.037**	**0.906 ± 0.042**	**0.909 ± 0.040**
	ALMMo-0-w (GM)	0.949 ± 0.032	0.877 ± 0.058	0.862 ± 0.066	0.868 ± 0.060
	ALMMo-0-w (F1)	0.935 ± 0.039	0.906 ± 0.041	0.896 ± 0.045	0.898 ± 0.042
	ALMMo-0-w (KC)	0.934 ± 0.039	0.899 ± 0.043	0.888 ± 0.048	0.891 ± 0.044
	ALMMo-0-w (MC)	0.934 ± 0.040	0.902 ± 0.037	0.892 ± 0.041	0.895 ± 0.038
	LR	0.899 ± 0.041	0.853 ± 0.043	0.845 ± 0.045	0.848 ± 0.044
	SVM	**0.966 ± 0.022**	**0.965 ± 0.023**	**0.964 ± 0.024**	**0.965 ± 0.023**
	NN	0.890 ± 0.048	0.804 ± 0.059	0.794 ± 0.061	0.795 ± 0.061
	ANFIS	0.891 ± 0.048	0.802 ± 0.055	0.791 ± 0.057	0.793 ± 0.057
	ALMMo	**0.959 ± 0.058**	0.743 ± 0.134	0.725 ± 0.148	0.750 ± 0.126
5%	ALMMo-0	0.876 ± 0.038	**0.867 ± 0.044**	**0.861 ± 0.045**	**0.870 ± 0.040**
	ALMMo-0-1C	**0.998 ± 0.004**	**0.985 ± 0.022**	**0.984 ± 0.024**	**0.984 ± 0.023**
	ALMMo-0-w (GM)	0.950 ± 0.033	0.795 ± 0.081	0.782 ± 0.088	0.796 ± 0.075
	ALMMo-0-w (F1)	0.919 ± 0.040	0.896 ± 0.038	0.891 ± 0.039	0.895 ± 0.036
	ALMMo-0-w (KC)	0.917 ± 0.037	0.897 ± 0.041	0.892 ± 0.043	0.897 ± 0.039
	ALMMo-0-w (MC)	0.906 ± 0.041	0.891 ± 0.038	0.886 ± 0.040	0.891 ± 0.035
	LR	0.775 ± 0.091	0.693 ± 0.080	0.690 ± 0.080	0.710 ± 0.069
	SVM	**0.979 ± 0.020**	**0.978 ± 0.020**	**0.978 ± 0.020**	**0.979 ± 0.020**
	NN	0.801 ± 0.082	0.737 ± 0.098	0.735 ± 0.099	0.751 ± 0.094
	ANFIS	0.411 ± 0.070	0.292 ± 0.081	0.290 ± 0.080	0.409 ± 0.070
	ALMMo	**0.940 ± 0.043**	0.594 ± 0.159	0.588 ± 0.162	0.634 ± 0.128
1%	ALMMo-0	0.872 ± 0.038	**0.863 ± 0.044**	**0.862 ± 0.044**	**0.871 ± 0.039**
	ALMMo-0-1C	**0.995 ± 0.008**	0.784 ± 0.277	0.780 ± 0.281	0.815 ± 0.236
	ALMMo-0-w (GM)	0.956 ± 0.026	0.497 ± 0.179	0.490 ± 0.182	0.562 ± 0.150
	ALMMo-0-w (F1)	0.909 ± 0.027	**0.898 ± 0.030**	**0.897 ± 0.030**	**0.902 ± 0.028**
	ALMMo-0-w (KC)	0.903 ± 0.026	0.892 ± 0.030	0.891 ± 0.030	0.896 ± 0.029
	ALMMo-0-w (MC)	0.903 ± 0.029	0.897 ± 0.029	0.896 ± 0.030	0.901 ± 0.027

Table 5.15: Geometric Mean, F1-Score, Cohen´s Kappa and Matthews Correlation Coefficient results for the WBCO dataset

5.3 ALMMo-0 Classifier and Benchmark Methods

In this section, the performance of the original ALMMo-0 classifier is compared with the selected benchmark methods. In order to better summarize the results for the different datasets, Table 5.16 presents the Win/Tie/Loss counts for the different models, by class imbalance and metric.

Metric	Class Imbalance	Method				
		LR	SVM	NN	ANFIS	ALMMo
Accuracy	LOW	3/3/6	6/2/4	3/3/6	1/2/9	0/3/9
	HIGH	5/4/3	8/4/0	5/3/4	4/2/6	4/6/2
Recall	LOW	2/3/7	5/3/4	4/1/7	2/0/10	6/2/4
	HIGH	1/2/9	3/1/8	1/2/9	0/1/11	2/2/8
Precision	LOW	3/2/7	4/3/5	3/2/7	1/4/7	0/2/10
	HIGH	1/3/8	2/5/5	1/3/8	0/3/9	1/1/10
Specificity	LOW	5/2/5	5/2/5	7/0/5	7/1/4	2/3/7
	HIGH	8/2/2	8/4/0	8/2/2	7/3/2	8/3/1
G-Mean	LOW	3/2/7	5/3/4	4/1/7	2/0/10	4/3/5
	HIGH	1/2/9	4/0/8	1/2/9	0/1/11	2/2/8
F1-Score	LOW	3/1/8	4/5/3	3/2/7	1/1/10	1/2/9
	HIGH	1/2/9	4/0/8	1/1/10	0/0/12	0/1/11
Kappa	LOW	3/1/8	4/5/3	3/2/7	1/1/10	1/2/9
	HIGH	1/2/9	4/0/8	1/1/10	0/0/12	0/1/11
Matthews	LOW	3/1/8	6/3/3	3/2/7	1/1/10	1/2/9
	HIGH	1/2/9	4/2/6	1/1/10	0/0/12	0/1/11

Table 5.16: Win/Tie/Loss counts of the benchmark methods versus the ALMMo-0 classifier

Starting with the minority class prediction performance for low imbalance datasets, the results clearly suggest that, in general, ALMMo-0 models outperform LR, NN and ANFIS, while showing more mixed results when comparing to SVM and ALMMo.

Supporting this conclusion are the recall performances, which show that LR, NN and ANFIS under-perform on most of the tests. Recall performances are more mixed for SVM and ALMMo, showing a slight advantage.

Regarding the impact of the better recall performances on the majority class detection, recall results show that the ALMMo-0 classifier out-performs all the benchmark models. This might seem surprising, since better recall performances often result in worse precision performances. However, observing the specificity results, which show that the ALMMo-0 either matches or under-performs the performance of all the other methods with the exception of the ALMMo regressor, it becomes clear that the the benchmark methods generally ignored the minority class, meaning that near perfect specificity results were achieved even without any positive class predictions.

Another interesting set of results are the accuracy performances, which show a general advantage of the ALMMo-0 classifier, outperforming all the benchmark methods except SVM. This further suggests that the better minority class detection verified for the ALMMo-0 did not significantly impact the overall prediction performance.

Regarding the remaining metrics, the results clearly suggest that the ALMMo-0 classifier generally outperforms the benchmarks methods, with the exception of the SVM, which shows more mixed results. This is expectable, since the geometric mean and F1-Score benefit from higher recall scores. Regarding the Kappa coefficient and the Matthews coefficient, the better results are also not surprising, since these metrics benefit from better minority class detection. Furthermore, these results once again suggest that the ALMMo-0 classifier is able to achieve better minority class detection without overly compromising the overall classification performance.

Regarding the minority class prediction performance for high imbalance datasets, it is very clear that the ALMMo-0 classifier generally outperforms all the benchmark methods. Once again, this conclusion is clearly supported by the recall results, which show that ALMMo-0 outperforms all the other models.

Furthermore, the results also show that the ALMMo-0 classifier outperforms all the benchmark models in terms of precision, while underperfoming in terms of specificity. The reason for this is that once again, the benchmark models disregard the minority class samples, meaning the number of positive class predictions is generally very low, or even zero.

Regarding the accuracy results, the ALMMo-0 classifier is out-performed by all the benchmark methods. The reason for this is simply that for highly imbalanced datasets, the accuracy is far from being a good assessment of the overall prediction performance of the models.

Regarding the remaining metrics, the results further support the overall better minority class prediction performance of ALMMo-0 models, since it outperforms all the benchmark methods, for each one of the metrics. Therefore, it once again becomes clear that ALMMo-0 models achieve better minority class prediction performance without compromising the overall prediction performance.

Attending to these results, it is clear that the proposed method is particularly well suited for highly imbalanced datasets, as it is able to achieve considerably better minority class prediction performance without overly compromising the overall classification performance.

5.4 ALMMo-0 Classifier and Proposed methods

In this section, the performance of the original ALMMo-0 classifier is compared with the proposed methods. In order to better summarize the results for the different datasets, Table 5.17 presents the Win/Tie/Loss counts for the different models, by class imbalance and metric.

Metric	Class Imbalance	Method				
		ALMMo-0-1C	ALMMo-0-W			
			GM	F1	KC	MC
Accuracy	LOW	1/2/9	0/3/9	1/6/5	0/6/6	0/6/6
	HIGH	1/3/8	0/1/11	0/6/6	0/7/5	0/8/4
Recall	LOW	3/1/8	10/2/0	3/9/0	2/10/0	2/10/0
	HIGH	9/1/2	10/2/0	4/8/0	4/8/0	4/8/0
Precision	LOW	2/2/8	0/1/11	0/4/8	0/4/8	0/3/9
	HIGH	0/3/9	0/0/12	0/6/6	0/6/6	0/6/6
Specificity	LOW	3/1/8	0/1/11	0/6/6	0/7/5	0/8/4
	HIGH	0/3/9	0/1/11	0/4/8	0/6/6	0/7/5
G-Mean	LOW	2/2/8	6/6/0	2/10/0	2/10/0	2/10/0
	HIGH	9/1/2	8/4/0	2/10/0	4/8/0	3/9/0
F1-Score	LOW	2/0/10	0/3/9	2/7/3	1/8/3	1/8/3
	HIGH	5/4/3	0/1/11	1/8/3	1/8/3	1/7/4
Kappa	LOW	2/0/10	0/3/9	2/7/3	1/8/3	1/7/4
	HIGH	5/4/3	0/1/11	2/7/3	1/7/4	1/7/4
Matthews	LOW	2/0/10	0/3/9	1/8/3	0/9/3	1/7/4
	HIGH	4/5/3	0/2/10	1/7/4	1/9/2	1/7/4

Table 5.17: Win/Tie/Loss counts of the proposed methods versus the ALMMo-0 classifier

5.4.1 One Class Classifier

Regarding the accuracy, it is clear that, in general, the one class classifier significantly under performs for all datasets and class imbalances. Furthermore, the results show no significant relation between accuracy performance and the imbalance ratio.

In terms of recall, it is clear that the proposed method tends to outperform at higher imbalances, while underperforming at lower imbalances. Furthermore, the results show a clear relation between recall performance and class imbalance, as the recall performance tends to improve as the class imbalance becomes more pronounced.

Precision performance results clearly show that the proposed method significantly under performs for all datasets and class imbalances. This is expected, as the increase in recall performance implies a decrease in precision performance, as the reduction of false negatives leads to an increase of false positives. Furthermore, results show that the precision performance deteriorates for more pronounced class imbalances, following the opposite behaviour that was verified for the recall performance.

Regarding specificity, results show a slight decrease in performance for all datasets and class imbalances. Furthermore, there is also no clear relation between the specificity performance and the imbalance ratio. However, it is very clear that the decrease in specificity performance is always less substantial than the decrease in precision performance, since the number of false positives is much smaller than the number of true negatives.

Performance in terms of the geometric mean follows a similar behaviour to the one discussed for recall, as the proposed method only outperforms the original algorithm for high imbalances. These results are simple to justify, since, as discussed, no substantial changes in specificity performance were found and, therefore, the geometric mean performance is essentially controlled by the recall performance.

Regarding the F1-Score, the results show a similar behaviour to what was observed for the geometric mean, although the results are more mixed and dependent on the dataset. These results are justified by the recall and precision performances, clearly suggesting that for high imbalances, the recall improvements outweigh the drops in precision performance. Furthermore, the fact that the F1-Score gives equal importance to recall and precision further suggests that the proposed method provides a good compromise between minimizing false negatives and not excessively increasing the number of false positives.

Regarding the Kappa and Matthews coefficients, the results shows that the proposed method underperforms for low imbalances, while showing mixed and highly dataset dependent results for high imbalances. Nonetheless, since there is no significant performance change regarding these metrics, and since the recall performance clearly improved, it is clear that for high imbalances the proposed method provides a clear improvement on the minority class prediction performance without compromising the overall classification performance.

Attending to the results presented so far for the individual metrics, one can arrive at some general conclusions about the proposed one class classifier. Firstly, it is clear that there is an improvement in the minority class detection for high class imbalances, as is clearly shown by the recall improvements. It is also clear that the minimization of false negatives does not exaggeratedly increase the number of false positives. However, one could still argue that, in certain contexts and applications, the increase in false positives may still too large and may not justify the reduction of false negatives.

5.4.2 Weighted Class Confidence

Accuracy performance results show that, similarly to the one class classifier, the proposed method generally under performs for all datasets and class imbalances. However, contrarily to what was observed for the one class classifier, the results also show a clear relation between higher class imbalances and higher accuracy penalties.

Furthermore, the accuracy results also suggest that the performance for GM is clearly distinct from the other cost functions, as it is clear that GM yields the highest drops in accuracy, not only when compared to original ALMMo-0, but also when compared to the one class classifier. Cost functions F1, KC and MC show a general small under performance, with a overall moderate drop for high imbalances.

Recall results show that the proposed method generally outperforms across all datasets and, in particular, high class imbalances. This is in contrast with what was observed for the one class classifier, which has shown moderate recall drops for low class imbalances. Therefore, the results suggest that the weighted class confidence method is, in general, more adequate for lower class imbalances than the proposed one class classifier.

Furthermore, the recall performances show once again a clear difference between GM and the remaining cost functions, as GM not only significantly outperforms the other cost functions, but also significantly outperforms the one class classifier. This is expected, as GM has also shown the largest drop in accuracy, implying that GM achieves the largest reduction in false negatives, at the cost of having the largest increase in false positives. Cost functions F1, KC and MC all show comparable results, achieving a more subtle false negative minimization, while also not exaggeratedly increasing the number of false positives.

As expected, precision results show that the proposed method under-performs for all cost functions, and also support what was concluded from the recall performances, as GM shows the largest drops, particularly at higher imbalances. Cost functions F1, KC and MC show less significant drops in precision since, as already mentioned, they also achieve more modest recall improvements.

The specificity results further support what was concluded from the previous metrics, showing that, as it was the case for the one class classifier, there is a general under performance for all datasets and class imbalances. Furthermore, cost functions F1, KC and MC once again show comparable results which are also comparable to the one class classifier. Cost function GM is the only method that shows a significant drop in specificity, particular at higher class imbalances since, as was already discussed, it significantly increases the number of false positives.

Geometric mean performances suggest a very similar behaviour to the one observed for the one class classifier. Cost functions F1, KC and MC achieve moderate improvements at higher class imbalances, in line with their recall performance, while still under performing the one class classifier. Cost function GM shows once again the largest improvement increase, since it also achieves the largest recall improvements.

It is important to mention that, despite GM using the geometric mean to optimize the class weighting, this is most likely not the reason why it outperforms the other methods, since, as it will be discussed for F1, KC and MC, the results show no clear correlation between the cost function metric and the actual model performance on that same metric. Thus, the most likely explanation for the substantial performance increase that GM shows for the geometric mean is that, since the specificity suffers only moderate drops, using the geometric mean as the cost function is essentially equivalent to optimizing only the recall performance, meaning that no compromise between false negatives and false positives is considered.

Observing the results for F1-Score, Kappa and Matthews, it once again becomes clear that cost function GM and the other cost functions yield different results. Cost function GM shows larger drops than the other cost functions and the one class classifier, once again due to the large increase in false positives that causes an overall classification performance penalization.

Attending to these results, it is clear that the proposed method is not only able to achieve better minority class detection, but also that the different cost functions yield models with different biases towards the minority class. In particular, the results suggest that the GM cost function shows the strongest performance on low imbalances, while the remaining cost functions may be better suited for high imbalances, as they provide a more balanced compromise between recall and precision performances.

5.5 Training Performance and Model Complexity

While up until now, the performance difference between the different methods has been discussed only in terms of classification performance metrics, the training times, as well as the number of rules that define the models, are also very relevant metrics that may help the model choice decision.

In this section, we are particularly interested in the performance differences between the original ALMMo-0 classifier, the ALMMo regressor, and the proposed methods since all of these methods use cloud-based antecedent structures. Regarding the training times, the analysis presented in this section is also restrained to these methods, as all of them were implemented from scratch using the same coding routines and libraries, and therefore a fair comparison can be made.

Table 5.18 presents the model interpretability expressed using the FRDP metric, computed for the overall model, as well as for the negative and positive class sub-models. The training times per sample (including the weight optimization times) are also shown in Table 5.18.

Imbalance	Method	FRDP			Train Time Per Sample
		All	Class 0	Class 1	
20%	ALMMo-N	0.005 ± 0.001	—	—	2.25E-3 ± 4.26E-4
	ALMMo-0	0.086 ± 0.002	0.076 ± 0.002	0.123 ± 0.006	4.41E-4 ± 4.75E-6
	ALMMo0-1C	0.076 ± 0.002	0.076 ± 0.002	—	3.44E-4 ± 1.56E-5
	ALMMo-0-W (GM)	0.093 ± 0.002	0.082 ± 0.003	0.133 ± 0.007	9.27E-4 ± 4.39E-4
	ALMMo-0-W (F1)				8.15E-4 ± 3.94E-4
	ALMMo-0-W (KC)				8.32E-4 ± 3.14E-4
	ALMMo-0-W (MC)				7.58E-4 ± 1.52E-4
10%	ALMMo-N	0.005 ± 0.001	—	—	2.24E-3 ± 3.83E-4
	ALMMo-0	0.087 ± 0.002	0.077 ± 0.002	0.169 ± 0.009	4.40E-4 ± 4.79E-6
	ALMMo0-1C	0.077 ± 0.002	0.077 ± 0.002	—	3.79E-4 ± 1.01E-5
	ALMMo-0-W (GM)	0.092 ± 0.003	0.082 ± 0.003	0.179 ± 0.011	1.56E-3 ± 8.32E-4
	ALMMo-0-W (F1)				8.51E-4 ± 4.39E-4
	ALMMo-0-W (KC)				8.11E-4 ± 1.81E-4
	ALMMo-0-W (MC)				8.93E-4 ± 4.47E-4
5%	ALMMo-N	0.004 ± 0.001	—	—	1.99E-3 ± 2.46E-4
	ALMMo-0	0.083 ± 0.002	0.073 ± 0.002	0.221 ± 0.015	4.42E-4 ± 4.76E-6
	ALMMo0-1C	0.073 ± 0.002	0.073 ± 0.002	—	4.05E-4 ± 1.56E-5
	ALMMo-0-W (GM)	0.089 ± 0.003	0.079 ± 0.002	0.226 ± 0.016	1.33E-3 ± 8.08E-4
	ALMMo-0-W (F1)				8.37E-4 ± 3.56E-4
	ALMMo-0-W (KC)				8.46E-4 ± 3.28E-4
	ALMMo-0-W (MC)				8.60E-4 ± 3.07E-4
1%	ALMMo-N	0.005 ± 0.001	—	—	2.53E-3 ± 5.21E-4
	ALMMo-0	0.075 ± 0.002	0.072 ± 0.002	0.333 ± 0.046	4.46E-4 ± 3.55E-6
	ALMMo0-1C	0.072 ± 0.002	0.072 ± 0.002	—	4.20E-4 ± 1.02E-5
	ALMMo-0-W (GM)	0.081 ± 0.002	0.079 ± 0.002	0.361 ± 0.058	1.30E-3 ± 7.91E-4
	ALMMo-0-W (F1)				9.83E-4 ± 6.12E-4
	ALMMo-0-W (KC)				9.85E-4 ± 4.74E-4
	ALMMo-0-W (MC)				1.07E-3 ± 7.42E-4

Table 5.18: FRDP and training times

While the different metrics showed lots of variability between datasets, the results presented in Table 5.18 show apparent performance differences between the different methods.

Starting with the FRDP metric, it is clear that the ALMMo regressor consistently achieves dramatically lower values, meaning that the resultant models have fewer rules than the ALMMo-0 classifier, as well as the proposed methods. This is because, as discussed before, ALMMo-0 classifiers norm normalize the samples, effectively meaning that the modelling data space loses one dimension and that the samples are projected closer together. Furthermore, the FRDP metric stays relatively constant for the different class imbalances, meaning that the number of rules is generally proportional to the number of samples.

Regarding the ALMMo-0 classifier and the proposed methods, it is evident that the weighted class confidence method consistently achieves slightly higher FRDP values for the complete model, as well as for the individual class sub-models. This is because since a part of the training samples is exclusively used as the validation set to optimize the class weights, fewer samples are used to model the clouds, and therefore the FRDP increases. Regarding the proposed one-class classifier, nothing particularly relevant was found since it only models the majority class sub-model and, therefore, has the same FRDP metrics as the original ALMMo-0 majority class sub-model.

One interesting result that is clearly visible in Table 5.18 is that, while the majority class sun-model FRDP stays reasonably constant for the different class imbalances, the minority class sub-model FRDP significantly increases for higher class imbalances since there are fewer positive class samples. Furthermore, one may notice that although the number of minority samples is twenty times smaller at the 1% class imbalances, the positive class FRDP only increases about three times, clearly illustrating that most of the positive class samples at more moderate imbalances were captured by already existent clouds.

Nevertheless, it is essential to mention that the difference in FRDP metrics between the original ALMMo-0 and the proposed variants is relatively small. Therefore, performance differences due to model complexity, i.e., a high number of rules, should be negligible and may not be particularly relevant to help to choose the most adequate method.

Regarding the training times, it is evident that the ALMMo regressor shows the slowest performance when compared to the ALMMo-0 classifier, as well as the proposed methods. While this may seem odd since ALMMo regressors display, in general, the least amount of rules, it is important to recall that they also require consequent parameters optimization. As discussed, this is achieved using recursive least squares (RLS), which, although reasonably efficient when implemented in its vectorized form, is still more computationally expensive than the training algorithm of the ALMMo-0, which requires updating fewer parameters since there are no consequent parameters.

Regarding the proposed one-class classifier, the training times are similar but slightly shorter since only the majority class sub-model is trained. Regarding the proposed weighted class confidence method, it is clear that there exists a notable increase in the training times. This is entirely expected, as the weight optimization loop is pretty computationally expensive and takes about as much time as the cloud training stage, even though it only uses the validation samples. Nonetheless, and as mentioned, the proposed method is still faster than the ALMMo regressor, meaning that, in principle, it should also be feasible for online learning tasks.

Still regarding the weighted class confidence method and its performance, the mean number of iterations of the optimization loop, as well as the mean minority class weights, are shown in Table 5.19. Once again, the results showed high variability between the different datasets, as the confidence intervals suggest. Thus, the results presented in Table 5.19 are more valuable to compare the different cost functions performances at the different class imbalances.

Regarding the mean number of iterations, it is evident that it stays pretty constant for the different class imbalances. Since it is clearly below the maximum number of iterations (50), it suggests that, in general, the optimization either converged to an optimum weight value or was not able to achieve any improvements on the validation set. Furthermore, it is clear that, once again, the GM cost functions show a different behaviour from the remaining cost functions, as it shows a moderately higher number of iterations. As discussed, this is most likely because the GM cost function effectively prioritizes increasing the recall score, meaning that, in general, there is more room to increase the minority class weight and explore more values that may increase the performance, increasing the iterations until convergence is achieved.

Regarding the minority class weight variations, an immediate conclusion is that the weight variations are very subtle for all class imbalances, falling between 10-4 and 10-7 orders of magnitude. This was completely expected since the class confidences show pretty small (10-3) differences for the misclassified minority class samples.

Furthermore, it is clear that once again the GM cost functions show a different behaviour from the other cost functions, displaying significantly higher weight variations. The reason for this is again the fact that by prioritizing the recall score, the GM cost functions allows for a larger increase in giving priority to the positive class.

Cost Function	**Imbalance**	**Iterations**	**Weight Variation**
GM	20%	16.9 ± 2.3	1.71E-04 ± 4.27E-05
	10%	23.9 ± 3.3	1.87E-04 ± 7.79E-05
	5%	22.7 ± 2.9	2.17E-04 ± 8.28E-05
	1%	25.7 ± 1.5	4.90E-04 ± 7.12E-05
F1	20%	16.7 ± 5.2	7.51E-07 ± 6.40E-07
	10%	16.6 ± 3.5	8.06E-07 ± 5.05E-07
	5%	16.0 ± 1.6	1.31E-06 ± 6.08E-07
	1%	17.5 ± 3.5	3.77E-05 ± 1.81E-05
KC	20%	16.9 ± 5.7	6.69E-07 ± 4.01E-07
	10%	15.8 ± 2.8	9.71E-07 ± 6.75E-07
	5%	16.1 ± 2.4	9.79E-07 ± 7.01E-07
	1%	17.8 ± 2.5	3.48E-05 ± 9.98E-06
MC	20%	15.7 ± 5.5	8.97E-07 ± 3.34E-07
	10%	17.0 ± 3.9	1.00E-06 ± 3.79E-07
	5%	16.6 ± 3.6	1.04E-06 ± 4.04E-07
	1%	18.8 ± 9.4	4.27E-06 ± 1.48E-06

Table 5.19: Mean number of iterations, minority class weight variations and respective validation set cost function improvement

5.6 Model Selection Discussion

In the previous sections, the performance of the original ALMMo-0 classifier was compared to benchmark methods, as well as the proposed methods. This discussion has been based on either classification metrics or other relevant metrics such as training time and FRDP. However, it is crucial to mention that choosing the most adequate model for a certain applications will not only depend on the specific domain of the model, but also on expert input that may establish task-specific restrictions on the prediction performance.

Furthermore, although some classification metrics such as Cohen´s Kappa and Matthews Correlation Coefficient are generally considered to be adequate metrics that summarize a model´s performance on highly imbalanced datasets, it is entirely possible that the model that achieves better results at such metrics may not be the most adequate for the specific task. Therefore, the most adequate model is ultimately chosen based on the balance between false positives and false negatives, which is typically very dependent on the specific task. As such, simpler metrics such as accuracy, recall, precision and specificity are more useful for expert input to assess the most adequate model. In particular, and as thoroughly discussed in this thesis, the balance between recall and precision are crucial to assess the prediction performance of the method.

Having all these notions in mind, it is still clear that the original ALMMo-0 classifier outperforms the selected benchmark methods in terms of minority class detection and, in particular, for higher class imbalances. As discussed, these results are not only suggested by the overall better recall performances, but also by the overall better geometric-mean, F1-Score, Cohen´s Kappa and Matthews Correlation Coefficient scores. Furthermore, the benchmark methods tend to completely ignore the minority class, particularly at higher class imbalances, yielding poor precision results as no positives are ever predicted by the models.

Therefore, it is clear that benchmark regression methods tend to show poor minority class detection performances if no extra measures are considered, such as data sampling techniques or threshold moving approaches, as even a relatively simple and lightweight method such as ALMMo-0 outperforms the benchmark models.

Regarding the proposed ALMMo-0 based methods, the presented results clearly show that both the one class classifier and the weighted class confidence showed different degrees of improvement for the different class imbalances. Furthermore, it was also shown that the different cost functions yield different models with relevant performances differences.

For low-class imbalances (20% and 10%), the weighted class confidence method using the GM cost function is arguably the best choice, as it consistently achieves significant recall improvements while also having an acceptable drop in precision.

For high-class imbalances, there is a more diversified choice of methods, each one with different compromises between recall and precision.

Starting with the one class classifier, it may argued that it displays the best overall compromise, since it shows a good compromise between recall and precision, even achieving slight improvements on the

GM, F1, KC and MC metrics.

However, observing the results for the weighted class confidence, it is clear that the GM cost function displays the largest recall improvements, at the cost of also showing large drops in precision. The remaining cost functions yield recall results that are lower than the one class classifier, but also show more subtle precision drops.

Therefore, it may be argued that the proposed methods offer three distinct solutions for high class imbalances. For overall improvements that offer a good compromise for both classes detection, the one class classifier seems to be the better choice. However, if the recall improvements are still to low and/or there is a high tolerance for false positives, the weighted class method using the GM cost function may be more appropriate. On the contrary, if the precision drops are too large and unacceptable, the F1, KC and MC cost functions may be more appropriate, as they all yield moderate recall improvements with relatively small precision drops.

Chapter 6

Conclusions

This thesis proposed two main goals: studying and comparing the performance of ALMMo-0 classifiers to other relevant benchmark methods, and proposing methods based on the original ALMMo-0 that seek to mitigate the effect of highly imbalanced datasets.

Regarding the first point, the results discussed in 5.3 clearly suggest that ALMMo-0 classifiers achieves better minority class prediction performance when compared to benchmark methods such as LR, SVM and NN, when compared to traditional FRB systems such as the ANFIS, and also when compared to the thoroughly discussed first-order ALMMo regressor.

Furthermore, the selected benchmark methods were left unmodified, meaning that the original algorithms, as well as the datasets, were left unmodified. Thus, the results also suggest that these original algorithms in general show very poor performances on highly imbalanced datasets, and, in many cases, completely ignore the minority class.

Regarding the second point, two modifications of the original ALMMo-0 algorithm were proposed: a one class classifier, and a weighted class confidence approach. The results discussed in 5.4 and 5.5 suggest that both methods achieve, to different extents, better minority class detection that the original ALMMo-0 classifier.

Regarding the proposed one class classifier, the results suggest that it outperforms the original ALMMo-0 on datasets that display high class imbalances. This conclusion is supported by the overall higher recall scores, as well as the higher GM, F1, KC and MC values. Furthermore, the overall better minority class detection is achieved without increasing too much the number of false positives.

The proposed one class classifier is also a remarkably simple modification of the original algorithm, as it simply removes the minority class sub-model, leaving the rest of the training algorithm unmodified. Since it uses the clouds radius as the classification decision threshold, which are parameters that already must be estimated during the training process, the proposed method does not introduce any complexity to the original algorithm.

Regarding the proposed weighted class confidence approach, the results suggest that the different cost functions yield models with remarkably different characteristics. In particular, and as thoroughly discussed in 5.4, the GM cost function tends do yield models with higher biases towards the minority class, meaning that it gives more weight to positive class samples. As such, the weighted class confidence approach using the GM cost function is particularly adequate for datasets that show a moderate class imbalance, as the resultant models show significant increases in recall and acceptable drops in precision. However, on highly imbalanced, a large drop in precision is often observed, meaning that the models may not be applicable to some domains.

The remaining cost function, F1, KC and MC, yield slightly different results but, in general, generate models with a more subtle bias towards the minority class. In particular, for highly imbalanced datasets, these cost functions yield models that still achieve significant recall improvements, but also do not exaggeratedly compromise the majority class detection.

Still regarding the proposed weighted class confidence approach, some issues were observed, particularly in the optimization process. As evidenced by the relatively similar results for the F1, KC and MC cost functions, it is clear that for small validation sets, the optimisation procedure will not be able to differentiate between the different cost functions, as the small number of validation samples will often force the process to converge towards the same weight value that minimizes the different cost functions.

Furthermore, the optimization results also show that for smaller validation sets, it is often observed that the model remains unweighted, meaning that increasing the minority class weight did not translate into a minimization of the cost function.

Nonetheless, the results obtained for the proposed methods are still very encouraging, as they offer, in general, different models that outperform the original ALMMo-0, which itself also was shown to outperform the selected benchmark methods.

6.1 Achievements

Attending to the obtained results, this thesis presents three notable achievements:

- The ALMMo-0 Classifier was shown to have a significantly better minority class prediction performance when compared to benchmark classification methods, as well as both traditional fuzzy systems and cloud-based fuzzy systems

- The proposed one class classifier adaptation of the original ALMMo-0 algorithm was shown to have better minority class detection performance than the original classifier, for highly imbalanced datasets, even improving the Kappa and Matthews coefficient scores

- The proposed weighted class confidence approach was shown to have a better minority class prediction performance, particularly for highly imbalanced datasets. Furthermore, it was shown that the different cost functions yield models with different levels of bias towards the minority class, meaning that one may choose the model that better suits the problem and its particular domain-specific restrictions

6.2 Future Work

As mentioned throughout this thesis, the problem of imbalanced datasets is a very prevalent problem that affects most traditional classification methods. Furthermore, cloud-based fuzzy systems such as ALMMo systems are a fairly recent category of fuzzy systems and as such, it is also a fairly unexplored area. Therefore, plenty of different strategies and approaches could have been explored in this thesis, not only regarding the original ALMMo-0 classifier, but also regarding the proposed methods. As such, the following further work proposals are suggested:

- Combine the proposed methods with external imbalanced data approaches, such as data ressampling, feature selection and ensemble learning meta-algorithms, in order to further improve the results
- Compare the results obtained with both the original ALMMo-0, as well as the proposed variants, with threshold-tuned benchmark methods
- Generalize the proposed methods for imbalanced multi-class datasets
- Evaluate if the proposed methods could be realistically applied to streaming data applications, or if the optimization method makes it prohibitively slow
- Integrate the proposed methods into some of the more recently proposed ALMMo-0 architectures
- Test the proposed methods with different datasets, either by using different strategies to ress-sample the original datasets (such as using SMOTE to generate new samples instead of simply randomly ressampling the data)
- Test the proposed methods with real imbalanced datasets

Bibliography

[1] R. Elshawi, M. Maher, and S. Sakr. Automated Machine Learning: State-of-The-Art and Open Challenges. 2019.

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. pages 1–4, 2016.

[3] S. S. Yadav and S. M. Jadhav. Deep convolutional neural network based medical image classification for disease diagnosis. *Journal of Big Data*, 6(1), 2019. ISSN 21961115. doi: 10.1186/s40537-019-0276-2.

[4] D. Liu, Y. Cui, Y. Chen, J. Zhang, and B. Fan. Video object detection for autonomous driving: Motion-aid feature calibration. *Neurocomputing*, 409:1–11, 2020. ISSN 18728286. doi: 10.1016/j.neucom.2020.05.027.

[5] Y. Ning, S. He, Z. Wu, C. Xing, and L. J. Zhang. Review of deep learning based speech synthesis. *Applied Sciences (Switzerland)*, 9(19):1–16, 2019. ISSN 20763417. doi: 10.3390/app9194050.

[6] B. Pes. Learning from high-dimensional biomedical datasets: The issue of class imbalance. *IEEE Access*, 8:13527–13540, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.2966296.

[7] B. Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016. ISSN 21926360. doi: 10.1007/s13748-016-0094-0.

[8] H. Kaur, H. S. Pannu, and A. K. Malhi. A systematic review on imbalanced data challenges in machine learning: Applications and solutions. *ACM Computing Surveys*, 52(4), 2019. ISSN 15577341. doi: 10.1145/3343440.

[9] R. B. Rao, S. Krishnan, and R. S. Niculescu. Data mining for improved cardiac care, 2006. ISSN 1931-0145.

[10] W. Wei, J. Li, L. Cao, Y. Ou, and J. Chen. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, 16(4):449–475, 2013. ISSN 1386145X. doi: 10.1007/s11280-012-0178-0.

[11] D. A. Cieslak, N. V. Chawla, and A. Striegel. Combating imbalance in network intrusion datasets. *2006 IEEE International Conference on Granular Computing*, pages 732–737, 2006. doi: 10.1109/grc.2006.1635905.

[12] M. Kubat, R. C. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2-3):195–215, 1998. ISSN 08856125. doi: 10.1023/a:1007452223027.

[13] J. M. Johnson and T. M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1), 2019. ISSN 21961115. doi: 10.1186/s40537-019-0192-5. URL `https://doi.org/10.1186/s40537-019-0192-5`.

[14] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. *Proceedings - 2018 IEEE 5th International Conference on Data Science and Advanced Analytics, DSAA 2018*, pages 80–89, 2019. doi: 10.1109/DSAA.2018.00018.

[15] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1–21, 2015. ISSN 21961115. doi: 10.1186/s40537-014-0007-7.

[16] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani. Deep learning for IoT big data and streaming analytics: A survey. *IEEE Communications Surveys and Tutorials*, 20(4):2923–2960, 2018. ISSN 1553877X. doi: 10.1109/COMST.2018.2844341.

[17] I. Škrjanc, J. Iglesias, A. Sanchis, D. Leite, E. Lughofer, and F. Gomide. Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A Survey. *Information Sciences*, 490:344–368, 2019. ISSN 00200255. doi: 10.1016/j.ins.2019.03.060.

[18] P. Angelov and X. Zhou. Evolving fuzzy systems from data streams in real-time. *Proceedings of the 2006 International Symposium on Evolving Fuzzy Systems, EFS'06*, 00:29–35, 2006. doi: 10.1109/ISEFS.2006.251157.

[19] M. J. Gacto, R. Alcalá, and F. Herrera. Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Information Sciences*, 181(20):4340–4360, 2011. ISSN 00200255. doi: 10.1016/j.ins.2011.02.021. URL `http://dx.doi.org/10.1016/j.ins.2011.02.021`.

[20] P. Angelov and R. Yager. A new type of simplified fuzzy rule-based system. *International Journal of General Systems*, 41(2):163–185, 2012. ISSN 03081079. doi: 10.1080/03081079.2011.634807.

[21] P. Angelov, X. Gu, D. Kangin, and J. Principe. Empirical data analysis: A new tool for data analytics. *2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016 - Conference Proceedings*, pages 52–59, 2017. doi: 10.1109/SMC.2016.7844219.

[22] X. Gu. Multi-Layer Ensemble Evolving Fuzzy Inference System. *IEEE Transactions on Fuzzy Systems*, XX(XX):1–1, 2020. ISSN 1063-6706. doi: 10.1109/tfuzz.2020.2988846.

[23] X. Gu and P. Angelov. Autonomous Learning Multimodel Systems From Data Streams. *Encyclopedia of Data Warehousing and Mining, Second Edition*, 26(4):2213–2224, 2017. doi: 10.4018/9781605660103.ch088.

[24] P. Angelov and X. Gu. Autonomous learning multi-model classifier of 0-Order (ALMMo-0). *IEEE Conference on Evolving and Adaptive Intelligent Systems*, 2017-May:22–28, 2017. ISSN 24734691. doi: 10.1109/EAIS.2017.7954832.

[25] X. Gu, P. Angelov, and Z. Zhao. Self-organizing fuzzy inference ensemble system for big streaming data classification. *Knowledge-Based Systems*, 218:106870, 2021. ISSN 09507051. doi: 10.1016/j.knosys.2021.106870. URL `https://doi.org/10.1016/j.knosys.2021.106870`.

[26] E. Soares, P. Angelov, and X. Gu. Autonomous Learning Multiple-Model zero-order classifier for heart sound classification. *Applied Soft Computing Journal*, 94:106449, 2020. ISSN 15684946. doi: 10.1016/j.asoc.2020.106449. URL `https://doi.org/10.1016/j.asoc.2020.106449`.

[27] P. Angelov and X. Gu. MICE: Multi-layer Multi-model Images Classifier Ensemble. 2017.

[28] J. Yin, H. Li, and X. Jia. Crater detection based on gist features. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(1):23–29, 2015. ISSN 21511535. doi: 10.1109/JSTARS.2014.2375066.

[29] H. S. Dadi and G. K. Mohan Pillutla. Improved Face Recognition Rate Using HOG Features and SVM Classifier. *IOSR Journal of Electronics and Communication Engineering*, 11(04):34–44, 2016. ISSN 22788735. doi: 10.9790/2834-1104013444.

[30] P. Angelov and X. Gu. A cascade of deep learning fuzzy rule-based image classifier and SVM. *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017*, 2017-Janua: 746–751, 2017. doi: 10.1109/SMC.2017.8122697.

[31] L. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(65)90241-X. URL `https://www.sciencedirect.com/science/article/pii/S001999586590241X`.

[32] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C. T. Lin. A review of clustering techniques and developments. *Neurocomputing*, 267:664–681, 2017. ISSN 18728286. doi: 10.1016/j.neucom.2017.06.053. URL `http://dx.doi.org/10.1016/j.neucom.2017.06.053`.

[33] P. Rai and S. Singh. A Survey of Clustering Techniques. *International Journal of Computer Applications*, 7(12):1–5, 2010. doi: 10.5120/1326-1808.

[34] F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: an overview. *WIREs Data Mining and Knowledge Discovery*, 2(1):86–97, 2012. doi: https://doi.org/10.1002/widm.53. URL `https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.53`.

[35] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. ISSN 00359254, 14679876. URL `http://www.jstor.org/stable/2346830`.

[36] L. Zhu, F.-L. Chung, and S. Wang. Generalized fuzzy c-means clustering algorithm with improved fuzzy partitions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39 (3):578–591, 2009. doi: 10.1109/TSMCB.2008.2004818.

[37] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek. Density-based clustering. *WIREs Data Mining and Knowledge Discovery*, 1(3):231–240, 2011. doi: https://doi.org/10.1002/widm.30. URL `https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.30`.

[38] S. S. Mullick, S. Datta, S. G. Dhekane, and S. Das. Appropriateness of performance indices for imbalanced data classification: An analysis. *Pattern Recognition*, 102:107197, 2020. ISSN 00313203. doi: 10.1016/j.patcog.2020.107197. URL `https://doi.org/10.1016/j.patcog.2020.107197`.

[39] A. Singh and A. Purohit. A Survey on Methods for Solving Data Imbalance Problem for Classification. *International Journal of Computer Applications*, 127(15):37–41, 2015. doi: 10.5120/ijca2015906677.

[40] A. More. Survey of resampling techniques for improving classification performance in unbalanced datasets. 10000:1–7, 2016. URL `http://arxiv.org/abs/1608.06048`.

[41] M. Lango and J. Stefanowski. Multi-class and feature selection extensions of Roughly Balanced Bagging for imbalanced data. *Journal of Intelligent Information Systems*, 50(1):97–127, 2018. ISSN 15737675. doi: 10.1007/s10844-017-0446-7.

[42] O. Sagi and L. Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):1–18, 2018. ISSN 19424795. doi: 10.1002/widm.1249.

[43] C. X. Ling and V. S. Sheng. Cost-Sensitive Learning and the Class Imbalance Problem. *Encyclopedia of Machine Learning*, (January 2010):231–235, 2008. URL `http://www.springer.com/computer/ai/book/978-0-387-30768-8%5Cnhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.164.4418&rep=rep1&type=pdf`.

[44] G. Collell, D. Prelec, and K. Patil. Reviving Threshold-Moving: a Simple Plug-in Bagging Ensemble for Binary and Multiclass Imbalanced Data. 2016. URL `http://arxiv.org/abs/1606.08698`.

[45] S. S. Khan and M. G. Madden. One-class classification: Taxonomy of study and review of techniques. *Knowledge Engineering Review*, 29(3):345–374, 2014. ISSN 14698005. doi: 10.1017/S026988891300043X.

[46] R. Harliman and K. Uchida. Data- and algorithm-hybrid approach for imbalanced data problems in deep neural network. *International Journal of Machine Learning and Computing*, 8(3):208–213, 2018. ISSN 20103700. doi: 10.18178/ijmlc.2018.8.3.689.

[47] S. Nejatian, H. Parvin, and E. Faraji. Using sub-sampling and ensemble clustering techniques to improve performance of imbalanced classification. *Neurocomputing*, 276:55–66, 2018. ISSN 18728286. doi: 10.1016/j.neucom.2017.06.082. URL `https://doi.org/10.1016/j.neucom.2017.06.082`.

[48] Y. Pang, Z. Chen, L. Peng, K. Ma, C. Zhao, and K. Ji. A signature-based assistant random oversampling method for malware detection. *Proceedings - 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE 2019*, pages 256–263, 2019. doi: 10.1109/TrustCom/BigDataSE.2019.00042.

[49] J. Prusa, T. M. Khoshgoftaar, D. J. DIttman, and A. Napolitano. Using Random Undersampling to Alleviate Class Imbalance on Tweet Sentiment Data. *Proceedings - 2015 IEEE 16th International Conference on Information Reuse and Integration, IRI 2015*, pages 197–202, 2015. doi: 10.1109/IRI.2015.39.

[50] A. Fernández, S. García, F. Herrera, and N. V. Chawla. SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. *Journal of Artificial Intelligence Research*, 61:863–905, 2018. ISSN 10769757. doi: 10.1613/jair.1.11192.

[51] Q. Ning, X. Zhao, and Z. Ma. A novel method for Identification of Glutarylation sites combining Borderline-SMOTE with Tomek links technique in imbalanced data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, (2):1–1, 2021. ISSN 1545-5963. doi: 10.1109/tcbb.2021.3095482.

[52] L. Yin, Y. Ge, K. Xiao, X. Wang, and X. Quan. Feature selection for high-dimensional imbalanced data. *Neurocomputing*, 105:3–11, 2013. ISSN 09252312. doi: 10.1016/j.neucom.2012.04.039. URL `http://dx.doi.org/10.1016/j.neucom.2012.04.039`.

[53] B. H. Nguyen, B. Xue, and M. Zhang. A survey on swarm intelligence approaches to feature selection in data mining. *Swarm and Evolutionary Computation*, 54(February):100663, 2020. ISSN 22106502. doi: 10.1016/j.swevo.2020.100663. URL `https://doi.org/10.1016/j.swevo.2020.100663`.

[54] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang. Benchmark for filter methods for feature selection in high-dimensional classification data. *Computational Statistics and Data Analysis*, 143:106839, 2020. ISSN 01679473. doi: 10.1016/j.csda.2019.106839. URL `https://doi.org/10.1016/j.csda.2019.106839`.

[55] M. Mafarja and S. Mirjalili. Whale optimization approaches for wrapper feature selection. *Applied Soft Computing*, 62:441–453, 2018. ISSN 15684946. doi: 10.1016/j.asoc.2017.11.006. URL `http://dx.doi.org/10.1016/j.asoc.2017.11.006`.

[56] S. M. Vieira, J. M. Sousa, and U. Kaymak. Fuzzy criteria for feature selection. *Fuzzy Sets and Systems*, 189(1):1–18, 2012. ISSN 01650114. doi: 10.1016/j.fss.2011.09.009. URL `http://dx.doi.org/10.1016/j.fss.2011.09.009`.

[57] S. Sivanandam and S. Deepa. Genetic Algorithm Optimization Problems. *Introduction to Genetic Algorithms*, pages 165–209, 2007. doi: 10.1007/978-3-540-73190-0_7.

[58] M. Dorigo and C. Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243–278, 2005. ISSN 03043975. doi: 10.1016/j.tcs.2005.05.020.

[59] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization: An overview. *Swarm Intelligence*, 1(1):33–57, 2007. ISSN 1935-3820. doi: 10.1007/s11721-007-0002-0.

[60] S. M. Vieira, L. F. Mendonça, G. J. Farinha, and J. M. Sousa. Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients. *Applied Soft Computing Journal*, 13(8):3494–3504, 2013. ISSN 15684946. doi: 10.1016/j.asoc.2013.03.021.

[61] S. M. Vieira, U. Kaymak, and J. M. Sousa. Cohen's kappa coefficient as a performance measure for feature selection. *2010 IEEE World Congress on Computational Intelligence, WCCI 2010*, 2010. doi: 10.1109/FUZZY.2010.5584447.

[62] S. Oh, M. S. Lee, and B. T. Zhang. Ensemble learning with active example selection for imbalanced biomedical data classification. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(2):316–325, 2011. ISSN 15455963. doi: 10.1109/TCBB.2010.96.

[63] N. Thomas Rincy and R. Gupta. Ensemble learning techniques and its efficiency in machine learning: A survey. *2nd International Conference on Data, Engineering and Applications, IDEA 2020*, 2020. doi: 10.1109/IDEA49133.2020.9170675.

[64] L. Breiman. Bagging predictors: Technical Report No. 421. *Department of Statistics University of California*, (2):19, 1994. URL `https://www.stat.berkeley.edu/%7B$\sim$%7Dbreiman/bagging.pdf`.

[65] R. E. Schapire. *Explaining AdaBoost*, pages 37–52. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-41136-6. doi: 10.1007/978-3-642-41136-6_5. URL `https://doi.org/10.1007/978-3-642-41136-6_5`.

[66] A. Iranmehr, H. Masnadi-Shirazi, and N. Vasconcelos. Cost-sensitive support vector machines. *Neurocomputing*, 343:50–64, 2019. ISSN 18728286. doi: 10.1016/j.neucom.2018.11.099. URL `https://doi.org/10.1016/j.neucom.2018.11.099`.

[67] I. D. Mienye and Y. Sun. Performance analysis of cost-sensitive learning methods with application to imbalanced medical data. *Informatics in Medicine Unlocked*, 25:100690, 2021. ISSN 23529148. doi: 10.1016/j.imu.2021.100690. URL `https://doi.org/10.1016/j.imu.2021.100690`.

[68] G. Collell, D. Prelec, and K. R. Patil. A simple plug-in bagging ensemble based on threshold-moving for classifying binary and multiclass imbalanced data. *Neurocomputing*, 275:330–340, 2018. ISSN 18728286. doi: 10.1016/j.neucom.2017.08.035. URL https://doi.org/10.1016/j.neucom.2017.08.035.

[69] L. Ruff, R. A. Vandermeulen, N. Görnitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft. Deep one-class classification. *35th International Conference on Machine Learning, ICML 2018*, 10: 6981–6996, 2018.

[70] D. Wang, S. Member, D. S. Yeung, E. C. C. Tsang, and A. Member. Structured One-Class Classification. 36(6):1283–1295, 2006.

[71] Y. Kim and H. K. Kim. Cluster-based deep one-class classification model for anomaly detection. *Journal of Internet Technology*, 22(4):903–911, 2021. ISSN 20794029. doi: 10.53106/160792642021072204017.

[72] B. Krawczyk, M. Woźniak, and B. Cyganek. Clustering-based ensembles for one-class classification. *Information Sciences*, 264:182–195, 2014. ISSN 00200255. doi: 10.1016/j.ins.2013.12.019.

[73] N. Lipka, B. Stein, and M. Anderka. Cluster-based one-class ensemble for classification problems in information retrieval. *SIGIR'12 - Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1041–1042, 2012. doi: 10.1145/2348283.2348459.

[74] Australian dataset, . https://archive.ics.uci.edu/ml/datasets/statlog+(australian+credit+approval).

[75] German dataset, . https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data).

[76] Mammographic dataset, . https://archive.ics.uci.edu/ml/datasets/mammographic+mass.

[77] Pima dataset, . https://www.kaggle.com/uciml/pima-indians-diabetes-database.

[78] Wbcd dataset, . https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic).

[79] Wbco dataset, . https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original).

Appendix A

Bayesian Optimization

Global optimization is a branch of applied mathematics that studies how to find the global minima or maxima of a function, or a set of functions, for a given range of values. Contrarily to analytical methods that seek to obtain an algebraic solution describing the optimal values, global optimization methods are numerical methods, and as such, are applicable to any problem, even for black-box objective functions that have no known algebraic formulation.

The general global optimization problem for a black-box function $f(x)$ is shown in A.1, where $\mathcal{A}$ is a bounded compact set define in a d dimensional space.

$$\min_{x \in \mathcal{A} \subset \mathbb{R}^d} f(x) \tag{A.1}$$

Bayesian optimization is a global optimization strategy particularly well suited for problems in which the objective function is unknown (no algebraic formulation is known) and expensive to evaluate. Bayesian optimization is also well suited for non-convex functions with multiple local minimums, since it allows for a compromise between exploration of the search space and exploitation of zones that may contain a more optimal value.

The two main components of Bayesian optimization are the surrogate function, which is a Bayesian approximation of the black box objective function that can be efficiently sampled, and the acquisition function, which is used to efficiently select which values to sample next.

The optimization strategy is based on Bayes Theorem, which defines the conditional probability of some event A given that an event B. The Bayes Theorem is defined as shown in A.2, where $P(A|B)$ is the posterior, $P(B|A)$ is the likelihood, and $P(A)$ is the prior.

$$P(A|B) = P(B|A) \times P(A) \tag{A.2}$$

If one considers a group of n samples ($x_1, x_2, ..., x_n$) and evaluates their respective objective function (f) values ($f(x_1), f(x_2), ..., f(x_n)$), and use these pairs to define the known objective function data D, then one can write the Bayes Theorem as shown in A.3.

$$P(f|D) = P(D|f) \times P(f) \tag{A.3}$$

The prior $P(f)$ is sequentially updated, as more samples are evaluated, and as such, is captures the updated beliefs about the objective function behaviour. The likelihood $P(D|f)$ is also sequentially updated as more samples are evaluated, and it may be interpreted as the probability of observing the collected data D given the updated function f.

Therefore, the posterior may be interpreted as the conditional probability of the objective function f given the available data D, and represents everything that is known about the function f given the collected data D. As such, it is an approximation of the black box objective function and it is used to define the surrogate function.

Many different different approaches exist to model the surrogate function, the most common one being Gaussian Processes (GP), that construct a joint probability distribution over the variables, assuming a multivariate Gaussian distribution. As such, GP models are capable of efficient summarization of a large number of functions and have a smooth structure without abrupt transitions, which are very desirable properties.

In order to efficiently evaluate the surrogate function and avoid sampling values that are unlikely to lead to a lower minimum, Bayesian Optimization uses a acquisition function that can be efficiently optimized. This acquisition function optimization is used to find the values that may lead to a more optimal value on the objective function. Many different types of acquisition functions exist, the most common ones being the Probability of Improvement (PI), Expected Improvement (EI), and the Lower Confidence Bounds (LCB).

Having presented all the elements used in Bayesian Optimization, the complete algorithm can be summarized as shown in Algorithm 4. The first step is evaluating N_0 points, which may either be randomly sampled from the search space or chosen using a more advanced criteria. These values are then used to initialize the sample data D, as well as the surrogate function.

The optimization loop consists of first maximizing the acquisition function in order to find the next sample that may lead to a more optimal value. After finding the candidate sample, it is evaluated using f, and the obtained pair $(x, f(x))$ is added to the collected data D. The updated data is then used to update the surrogate function. Multiple stopping criteria may be defined, such as the maximum number of iterations and the minimum improvement after a certain number of iterations.

Algorithm 4 Pseudo-Code for Bayesian optimization

```
1: Evaluate $f$ at $N_0$ points defined in the search space
2: Initialize the collected data $D$
3: Initialize the surrogate function
4: while stop criterion not met do
5:     Find next sample to evaluate by maximizing the acquisition function
6:     Evaluate the found sample using the objective function $f$
7:     Augment the collected data $D$
8:     Update the surrogate function
9: end while
```

In order to better visualize the described optimization algorithm, Figure A.1 show the algorithm evolution for the iterations 1, 10 and 20. The first column shows the real function (unknown for a real problem), the GP surrogate function approximation, as well as the evaluated samples. The second column shows the acquisition function, as well as the maximum value which is used to select the next candidate sample.

Observing the results, it is clear that the surrogate function shows a good fit to the real objective function as more samples are evaluated, and that the global minimum is found. Furthermore, it is also clear that there are more sampled points near the minimums, showing that the search procedure is clearly more efficient than grid search methods.

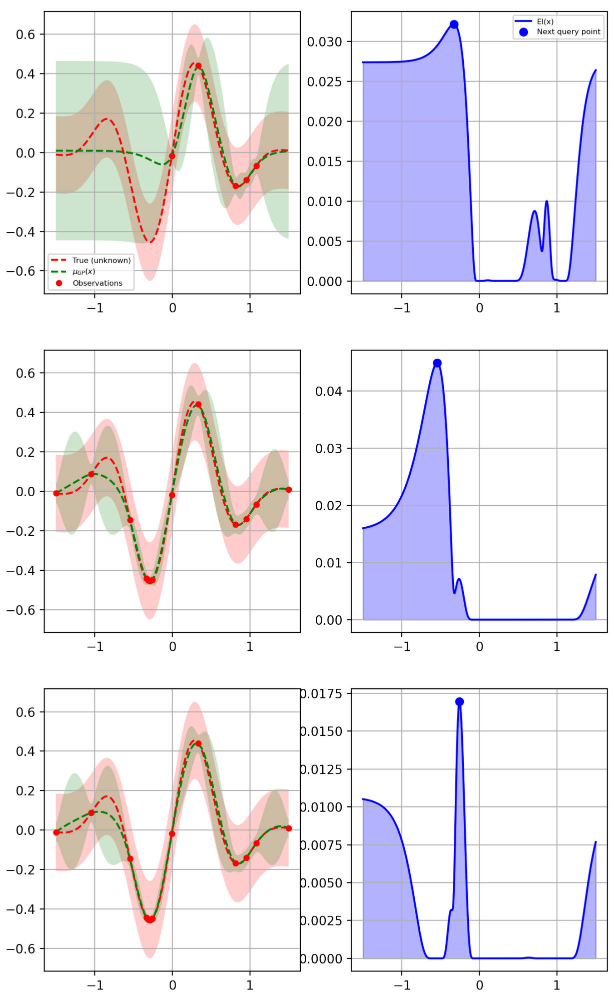


Figure A.1: Bayesian optimization example, showing the evolution of the objective function approximation, as well as the candidate sample selection using the acquisition function, for iterations 1, 10 and 20