A Full Meta-Learning approach to assist in the Full Model Selection problem in high volume datasets

By:

Angel Díaz Pacheco

A dissertation submitted in partial fulfillment
of the requeriments for the degree of:

DOCTOR OF SCIENCE IN COMPUTER SCIENCE

at

Instituto Nacional de Astrofísica, Óptica y Electrónica

February, 2019
Tonantzintla, Puebla

Advisor:
Carlos A. Reyes García, INAOE

Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisor Dr. Carlos for the continuous support of my Ph.D study and related research, for his patience, motivation, and knowledge. His friendly guidance and expert advice have been invaluable throughout all stages of the work.

Besides my advisor, I would like to thank the rest of my thesis committee: Dra. Claudia, Dra. Alicia, Dr. Felipe, Dr. Hugo and Dr. Rene, for their comments and valuable suggestions which have contributed greatly to the improvement of the thesis.

Also, I'm grateful for the support from CONACyT in the development of this research work (scholarship 428581).

Finally, I must express my very profound gratitude to my parents, brother and my fiancée for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

Abstract

Data by itself is not information, it becomes information only when it is analyzed and, a big part of such analysis is performed through machine learning techniques. Choosing the right technique for a dataset is not a trivial task because it requires to test the performance of the available alternatives and, taking into account that many of these techniques possess a set of configurable parameters, the process becomes harder. With the advent of social networks, the information in ambits outside the scientific research grew to unprecedented scales, this situation favored the storing of bigger quantities of data potentially rich in information and economical value, but the challenge of selecting the most adequate technique for those datasets got worsened.

The full model selection analysis emerged as a way to address this issue finding the best combination of a learning algorithm, a subset of features and a combination of data-preparation techniques to a dataset. Full model selection is capable to obtain models with high predictive accuracy and key information about datasets where there is no prior knowledge but, is not the first alternative when datasets become bigger. This analysis supposes to perform a series of transformations in the dataset and the construction of a classifier when a single model is evaluated but, bearing in mind the combination of all factors involved in the problem, the number of possible models is huge or even infinite. Consequently, this problem cannot be addressed through simpler methods as a grid search. Furthermore, the time of this process grows proportionally to the size of the datasets under analysis, therefore, with bigger datasets, the problem becomes intractable.

Several approaches as the use of: proxy models, meta-learning techniques and tools from the Big Data paradigm were explored to be capable to address the huge search space of the full model selection problem and to enable the analysis of high-volume datasets within an affordable computing time. The obtained results of this work showed an important reduction of the time in the search process in comparison with a robust algorithm for model selection and with models of higher predictive accuracy. The contributions of this work were: a framework to perform the full model selection analysis in datasets of any size, based in the MapReduce programming model. A new paradigm for the creation of proxy models based on classification algorithms and employing the full model selection analysis to create better proxy models. The use of the meta-learning paradigm to address the full model selection problem enabling the storing of the knowledge gained with each analysis and moving the problem from huge datasets to smaller meta-datasets. The synergy of the new

proxy models and the meta-learning paradigm obtained the best models faster than all the explored approaches in this work, therefore, the main contribution of this work was the introduction of the full meta-learning full model selection paradigm.

Resumen

Los datos en sí mismos no son información, se convierten en información solo cuando son analizados y gran parte de tal análisis es realizado a través de técnicas de aprendizaje automático. Elegir la técnica adecuada para un conjunto de datos no es una labor trivial ya que requiere de probar el desempeño de cada alternativa disponible y tomando en cuenta que muchas de estas poseen un conjunto de parámetros configurables, el proceso se hace más complicado. Con el advenimiento de las redes sociales, la información en ámbitos ajenos a la investigación científica creció a escalas sin precedentes, esta situación favoreció el almacenamiento de grandes cantidades de datos potencialmente ricos en información y valor económico, pero, el desafío de seleccionar la técnica más adecuada para un conjunto de datos se hizo más difícil.

El análisis de selección de modelo completo emergió como una forma de afrontar el problema de encontrar la mejor combinación de un algoritmo de aprendizaje, un subconjunto de características y una combinación de técnicas de preprocesamiento para un conjunto de datos. La selección de modelo completo es capaz de obtener modelos de gran poder predictivo e información de interés en conjuntos de datos que no han sido analizados, pero, no es la primera alternativa cuando los conjuntos de datos se hacen más grandes. Este análisis supone realizar una serie de transformaciones en el conjunto de datos y la construcción de un clasificador cuando solo un modelo es evaluado, pero, teniendo en mente la combinación de todos los factores involucrados en el problema, el número de modelos posibles es enorme e incluso infinito. En consecuencia, este problema no puede ser enfrentado a través de métodos más simples como la búsqueda en rejilla. Además, el tiempo de dicho proceso crece en proporción al tamaño del conjunto de datos bajo análisis, por lo tanto, con conjuntos de datos más grandes el problema se hace intratable.

Varios enfoques como el uso de: modelos proxy, técnicas de meta aprendizaje y herramientas provenientes del paradigma de Big Data fueron exploradas para tener la capacidad de enfrentar el enorme espacio de búsqueda del problema de selección de modelo completo y habilitar el análisis de conjuntos con gran volumen de datos dentro de un tiempo de computo razonable. Los resultados obtenidos en este trabajo mostraron una importante reducción del tiempo empleado en el proceso de búsqueda en comparación con un robusto algoritmo para selección de modelo completo y con modelos de mayor precisión predictiva. Las contribuciones de este trabajo fueron: un marco de trabajo para realizar el análisis de selección de modelo completo en conjuntos de datos de cualquier tamaño, basado en el modelo de programación MapReduce.

Un nuevo paradigma para la creación de modelos proxy basado en algoritmos de clasificación y empleando el análisis de selección de modelo completo para crear mejores modelos proxy. El uso del paradigma de meta aprendizaje para enfrentar el problema de selección de modelo completo posibilitando el almacenamiento del conocimiento obtenido con cada análisis y moviendo el problema de enormes conjuntos de datos a pequeños meta conjuntos de datos. La sinergia de los nuevos modelos proxy y el uso del paradigma de meta aprendizaje obtuvo los mejores modelos y más rápido que todos los enfoques explorados en este trabajo, por lo tanto, la principal contribución de este trabajo fue la introducción del paradigma de selección de modelo completo asistida por el uso completo del paradigma de meta aprendizaje.

Contents

List of Figures

List of Tables

Part I

Beginning

Chapter 1

Introduction

In the last years, emergence of social networks and development and lowering cost of information technologies made possible the massive generation and storing of data. Just from 2012 to 2014, 2.5 quintillion of bytes of data were daily created, that means that 90% of information in the planet was created in a two years interval [160]. On the other hand, the European Institute of Bioinformatic stores around 20 petabytes of information and currently is one of the main biologic information repositories in the world, while the European Organization for Nuclear Research or CERN (by the acronym in French) each year produces around 15 petabytes of information [104]. Data by itself has no value until information is extracted, then it becomes a profitable asset. Big part of data analysis is performed through machine learning techniques, but to choose the right technique to a dataset is not a trivial task, especially when a non-expert is on charge of such labor.

Although more data could be considered as a universal solution for machine learning problems, data by itself is not the answer. Considering a Boolean function of one hundred variables from a million examples, there are $2^{100} - 10^6$ examples with unknown class. Similarly, a linear classifier trained on a non-linear dataset, no matter how many data is available, the accuracy of the classifier is not going to improve. The "No Free Lunch" theorem of Wolpert and Macready [1997] refers to this situation saying that no machine learning algorithm is the best for all problems.

The model selection paradigm arose to solve the problem of to find the best model for a given dataset but, if we add the fact that parameters that govern behavior and performance of learning techniques must be tuned, the situation is worsened. Model selection has been faced through a wide range of methods like grid searches ([25, 12, 23, 143]), Bayesian methods ([149, 87]) and different bio-inspired mechanisms [70, 8, 29, 92].

Performance of a learning algorithm is also influenced by other factors as preparation of data into a better suited format for the learning algorithm through data-preparation techniques and selection of a subset of features that describes better the dataset using feature selection algorithms. The combination of the aforementioned factors constitutes the Full Model Selection (FMS) paradigm and is defined as: given a pool of pre-processing methods, feature selection and learning algorithms, to select the combination of these that obtains the lowest classification error for a given dataset; the task also includes the selection of hyper-parameters for the considered methods [57]. FMS can be considered as a black box fed with a dataset and, as output, a highly effective classification or regression model is obtained. In absence of this method, potentially useful classifiers are not considered and are just selected by guessing or by the popularity of a given algorithm.

FMS has been employed in diverse tasks as: region labeling on images [75], to help in diagnostic decisions for acute leukemia [58], and for the classification of infant crying patterns to aid in the health assessment of a baby [132], among others. The versatility of FMS to assist in multiple domains without the need of expert knowledge, makes relevant to equip this powerful tool with the ability to deal with the huge data sets of nowadays.

One of the main drawbacks of this paradigm is the vast search space generated by all the possible models that can be built given the considered methods and their hyper-parameters even if restrictions are imposed to the values of that models can take [57]. In conventional size datasets, in other words those that fit easily in the main memory of a standard personal computer, full model selection has been approached through bio-inspired algorithms ([57, 9]) and multi-objective genetic algorithms ([130, 131]). With the current trend of bigger datasets, transforming large amounts of data into actionable knowledge in a feasible time frame is an important task to map large investments in database storage into an actual advantage, however, despite its advantages, full model selection is not the first option.

To provide this paradigm with the capacity to deal with datasets of any size, the efforts of this work are focused in the use of tools from the meta-learning field to be capable to address the exploration of the big search space and intractable computing-times.

The use of Proxy models, that is to say, a less expensive (computationally speaking) alternative to a full numerical simulation [3], have been used ([131]) as a way to compress the time a process takes, but in this work their use was investigated as a way to guide the search process. Employing the proxy model in this way, models of higher predictive accuracy can be found sooner. Regarding meta-learning techniques, they are an important addition to FMS because they provide a way to store the

knowledge gained with each model selection task performed, moving the problem from huge datasets to smaller meta-datasets. The synergy of meta-learning techniques and proxy models introduces a new approach to assist in a time-consuming process where the fitness measure of a potential solution is just an estimation and not a real measure (expected misclassification rate), therefore, the main contribution of this work is the introduction of this new paradigm to assist in FMS and that can be extended to other similar problems.

Part of the objectives in the full model selection paradigm is to find a subset of features with high predictive power and if the dataset dimensionality is reduced before the model selection task, the quality of the final models could be compromised. Although the reduction of the dimensionality of datasets could be beneficial to make them easier to handle with standard PC's, in this work, the reduction of the dimensionality and the reduction in the number of instances in the dataset would not be performed to ensure that the analysis is capable to handle datasets of any size and to obtain the best possible models.

1.1 Objective

"Developing methods to perform a better exploration of the search space imposed by the full model selection problem in high volume datasets in terms of computing-time and models accuracy".

This will be accomplished using proxy models and meta-learning techniques optimized through the FMS paradigm. Despite the ideas presented in this document are focused to binary classification problems, they can be extended to other domains, more dimensions, and learning problems. To accomplish this objective, the next particular objectives are investigated.

- To gain knowledge about which is the best method to perform the FMS analysis in high volume datasets between a swarm-based method and an evolutive algorithm.
- To expand the knowledge in the use of the full model selection paradigm to build better proxy models.
- To expand the knowledge in the development of efficient explorations of the search space in terms of lower computing-times and more accurate models through the aforementioned proxy models.

- To broaden the knowledge of the use of meta-learning to solve the full model selection problem. Since the number of potential models in the full model selection paradigm can be infinite, the efforts to adjust the meta-learning paradigm to our problem will expand the knowledge in this field.

1.2 Hypothesis

The models obtained through the full model selection analysis in high volume datasets assisted by efficient proxy models and the meta-learning paradigm obtain a lower misclassification rate and are obtained performing a lower number of fitness evaluations than those obtained without the use of the aforementioned techniques.

1.3 Contributions

The present work provides the following contributions to the solution of the full model selection problem on high volume datasets:

- A framework to adapt population-based search algorithms to perform the full model selection analysis on high volume datasets. This framework is based on the MapReduce programming model and can be employed by non-expert users. To the best of the authors knowledge this approach is the first one to address the FMS problem in high volume datasets and obtaining models with higher accuracy in most of the evaluated datasets than the available alternatives for model selection.

- A method to use the full model selection paradigm to build proxy models. Importance of different aspects of the full model selection paradigm as feature selection and data preparation has been explored in some related works of proxy models field, however, not all the aspects of this paradigm has been taken into account. Taking as the starting point the good models obtained by the FMS, we extended this idea to proxy models. The evidence presented in this work shows the suitability of the FMS paradigm in the creation of better proxy models and to the best of our knowledge this idea was presented first in this work.

- A method to build proxy models using classification algorithms. Due the good results obtained by the previous contribution and the good classifiers obtained through FMS, we transformed the problem of predict the expected fitness of a potential solution into a binary classification problem (promising and not

promising). This approach is especially useful when the expected fitness of a solution is an estimation and not a true measure of a phenomenon as in model selection. This idea was also presented first in this work.

- A new algorithm based on fuzzy rules that can be used as proxy model or a multi-class classification algorithm. This algorithm was a side contribution of the approach explored in the use of classification algorithms in the construction of proxy models. The developed algorithm exploits similarities among the data points that represent solutions in an optimization algorithm and provides in addition to target class (promising and not promising) the membership degree to each one. To our knowledge this multi-class algorithm was not proposed in another work before.

- A method to use meta-learning techniques to solve the full model selection problem and a new meta-learner algorithm alternative to K Nearest Neighbors (K-NN). Due the high number of possible models, full model selection problem has never been addressed through a meta-learning approach, that is why this first approach is useful to practitioners and researchers as starting point to handle this problem. Furthermore, similarly to proxy models, the meta-learning literature has used some elements from the FMS paradigm to improve the outcome of the process, however, in this work the use of the entire FMS paradigm was proposed through a new meta-learner algorithm that obtained better models than the ones of K-NN.

- A new paradigm to address the FMS problem useful in both, regular size and high-volume datasets. The Full Meta-learning Full Model Selection provides a way to preserve and improve the knowledge gained with each model selection performed. The good quality of the initial models obtained through meta-learning and the efficient guiding of proxy models based on fuzzy-rules in the optimization step assures that FMS process can be performed in a smaller amount of time and obtaining better models that with a common optimization algorithm. This new approach is especially convenient for non-expert users, that cannot reduce the search space by picking a subset of learning algorithms based in their expertise, extending the black box metaphor used to describe the FMS process.

1.4 Products

As a result of contributions of last section, the following scientific papers have been published:

Conference

- Díaz-Pacheco,A. González-Bernal,J.,Reyes-García, C. A., *"Full model selection in huge datasets under the mapreduce paradigm"*, Proceedings of the International Conference on Big Data Analytics, Data Mining and Computational Intelligence", p. 271, 2017.
- Díaz-Pacheco,A. González-Bernal,J.,Reyes-García, C. A., *"Full model selection in Big Data"*, "Proceedings of the Mexican International Conference on Artificial Intelligence MICAI 2017".
- Díaz-Pacheco A., and Reyes-García, C. A., *"Full model selection in huge datasets through a meta-learning approach"*, Proceedings of the International Conference on Big Data Analytics, Data Mining and Computational Intelligence", p. 19, 2018.
- Díaz-Pacheco,A. ,Reyes-García, C. A., *"Full model selection in huge datasets and for proxy models construction"*, "Proceedings of the Mexican International Conference on Artificial Intelligence MICAI 2018".

Journal

- Díaz-Pacheco, A., Gonzalez-Bernal, J. A., and Reyes-García, C. A. (2018). *A mapreduce based framework to perform full model selection in very large datasets.* IADIS INTERNATIONAL JOURNAL ON COMPUTER SCIENCE AND INFORMATION SYSTEMS, ISSN: 1646-3692, 13(1), p. 1-13.
- Díaz-Pacheco and Reyes-García, C. A. *Facing the full model selection problem in high volume datasets employing intelligent proxy models.*, Intelligent Data Analysis, ISSN:1571-4128. *(To be published in fall of 2019)*

1.5 Document organization

The present work has been organized in five parts besides the introduction, which are described below:

- In Part II, all the background concepts needed for a better understanding are provided. This part consists of the three chapters described below.
  - Chapter 2 describes two important categories of the machine learning field: supervised and unsupervised learning. The elements under each one of these categories are of great importance in the proposed approaches in this work. The FMS problem and the parts that made it up are described in this Chapter too.
  - Chapter 3 describes relevant aspects of the FMS problem in high volume datasets as its link with the concept of complex systems and some relevant features of the data. The software tools used in this work as the MapReduce programming model and Apache Spark are also described.
  - Chapter 4 several tools from the soft computing field were employed in the contributions. Some of them are: Genetic Algorithms, fuzzy-rules systems, fuzzy classification and fuzzy clustering. Those concepts are described in this Chapter.
- In Part III, the contributions and the experiments are contained. Since each contribution explores different approaches of the meta-learning field and of the soft-computing area, each chapter has its own related works section, to avoid confusions. There are three Chapters in this part and are as follows.
  - Chapter 5, according the main trends in the state of the art, two bio-inspired optimization algorithms are tested and adapted to the MapReduce programming model as a first solution of the FMS problem in high volume datasets.
  - Chapter 6 explores the use of proxy models not as a way to reduce the time employed in process, instead they are employed to guide efficiently the search.
  - Chapter 7 explores the use of the meta-learning paradigm to provide a way to move the problem of high-volume datasets to small meta-datasets.
- In Part IV, the general conclusions of this work are outlined.
- Part V is for the appendixes.
  - In Appendix A, the MapReduce based algorithms are described.

– In Appendix B, experiments are conducted to test the employed datasets in order to have evidence that they represent different problems, regarding their Intrinsic Dimension. Also, the shape of this datasets is tested to provide variety in the shapes where the different approaches are tested.

Part II

Theoretical framework

Chapter 2

Supervised and unsupervised learning

In the machine learning field there are two important categories for the analysis tasks developed by this discipline: supervised and unsupervised learning. Under a **supervised learning** task, every sample in a dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$ has both, the input in $x \in X$ and the corresponding expected response $y \in Y$. Here, Y represents the set of possible outputs. The expected output is also referred to as the label. Formally, given a training set D, the goal of a supervised learning algorithm is to train a function $f : X \to Y$ where $f \in F$. Here F is some predefined family of functions. Each training sample (x_i, y_i) is assumed to be sampled independently of a joint distribution $P(x, y), x \in X, y \in Y$, which is not revealed to the learning algorithm. Once with a trained model, $f^*(x)$, given a new input $x^{'} \in X$, the label $\hat{y} = f^*(x^{`})$ can be predicted.

On the other hand, the training data in the case of **unsupervised learning** contains no supervisory information. Here, $D = \{x_1, \ldots, x_n\}$ and is assumed that each x_i is sampled independently of a distribution $P(x), x \in X$. Thus, these samples are independent and identically distributed [144].

As stated earlier, the FMS paradigm was designed to face the problem of to choose the right model for a dataset, a task that falls under both aforementioned categories, but it has been mainly investigated for classification jobs. For this reason, some topics of the supervised learning area, the FMS problem and some key elements related to it were outlined in this section. Despite this, the FMS paradigm can be extended to other machine learning problems and similar tasks whit a huge search space. Regarding unsupervised learning topics addressed in this section, they were employed in the experiments made to investigate and to adapt the meta-learning paradigm to our problem. In the same spirit, the meta-learning paradigm is also outlined in this chapter.

2.1 Regression and classification tasks

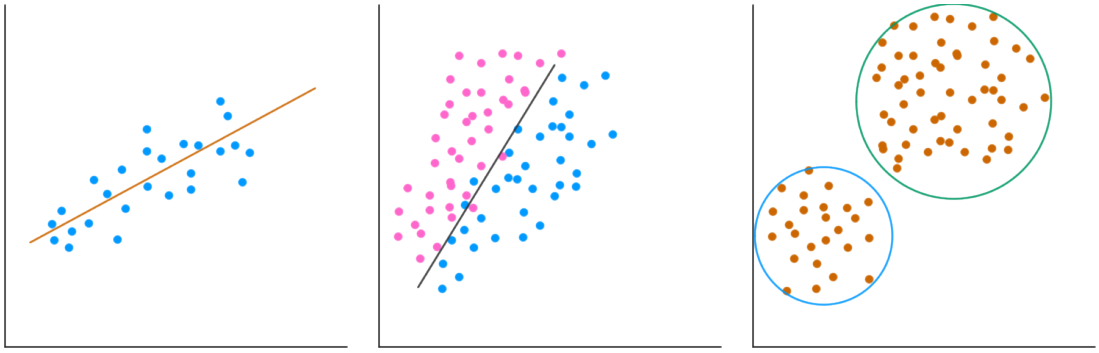
The classification and regression tasks are examples of the supervised learning category mentioned above. A **regression** analysis is a conceptually simple method for investigating functional relationships among variables. The relationship is expressed in the form of an equation or a model connecting the response or dependent variable and one or more explanatory or predictor variables. Let y be the response variable and the set of predictor variables $x_1, x_2, \ldots, x_p$ with (p = number of predictor variables). The true relationship between y and $x_1, x_2, \ldots, x_p$ can be approximated by the regression model $y = f(x_1, x_2, \ldots, x_p) + \epsilon$ where ϵ is assumed to be a random error representing the discrepancy in the approximation [26].

The **classification** analysis is a subtype of the regression analysis because the response variable is a discrete one instead of a continuous one. In a classification analysis the problem is defined by a distribution D over $x \times y$, where $y = \{0, 1, \ldots, n\}$ (with n = number of labels). The goal is to find a classifier $h : x \rightarrow y$ minimizing the error rate on D [98].

It is important to describe both approaches because, in this work the FMS paradigm was employed to build models for classification problems in high volume datasets whereas with the regression approach in synergy with the FMS paradigm, the construction of proxy models was explored. In Figures 2.2.1(a) and 2.2.1(b) a visual example of a regression and a classification task is showed.

2.2 Clustering analysis

A clustering analysis is a task from the unsupervised learning category and consist in the grouping of set of objects in a manner that the objects in the same group are more similar among them than to those in other groups. Formally speaking, given a dataset $D = \{t_1, t_2, \ldots, t_n\}$ of tuples and an integer number of clusters to create k partitions, to define a mapping $f : D \rightarrow (1, 2, \ldots, k)$ where each t_i is assigned to one cluster $K_j, 1 \leq j \leq k$. A cluster K_j, contains precisely those tuples mapped to it; that is $K_j = \{t_i | f(t_i) = K_j, 1 \leq j \leq n\}$ & $t_i \in D$ [80]. As will be seen below, the clustering analysis was employed in this work for the creation of an alternative to K-NN employed as meta-learner. In Fig. 2.2.1(c) an example of a clustering task is shown.



(a) Regression task. (b) Classification task. (c) Clustering task

Figure 2.2.1: Examples of supervised and unsupervised learning tasks.

2.3 Meta-learning and Proxy models

Meta-learning is the study of principled methods that exploits meta-knowledge to obtain efficient models and solutions by adapting machine learning and data mining processes [17]. Meta-learning uses a general machine learning approach to generate meta-knowledge that maps the characteristics of a dataset, captured by meta-features, to the relative performance of the available algorithms [120].

In this work, the meta-learning approach was used as an advisor to recommend models or full models to a not analyzed dataset. In an advisory mode, the meta-learner (the algorithm employed to learn in a meta-learning task) increases its efficiency as it accumulates meta-knowledge. The lack of experience at the beginning of the learner's life compels the meta-learner to use one or more learning strategies without a clear preference for one of them. However, as more training sets have been examined, it is expected that the expertise of the meta-learner to dominate in deciding which learning strategy best suits the characteristics of the training set. The meta-features extracted from the dataset (Fig. 2.3.1 b) are matched with the meta-knowledge base (Fig. 2.3.1 f) to produce a recommendation regarding the best available learning strategy [154].

Proxy-modeling on the other hand, is a subtype of the tasks performed through the meta-learning paradigm. A proxy model is a computationally less expensive alternative to a full numerical simulation in assisted history matching, production optimization, and forecasting. A proxy model is defined mathematically, statistically, or data driven model defined function that replicates the simulation model output for selected input parameters. The proxy model's results are not to mimic the numerical

simulations with 100% accuracy, but the outputs generated with the amount of time to run these models, give a reasonable range of error [108]. In a task that involves the use of proxy models, the knowledge base (Fig. 2.3.1 f) consist of a set of solutions solved through the full mathematical simulation. In an evolutive or a bio-inspired search the meta-dataset is constructed with the particles or individuals with their fitness value, where each part of the vector a meta-feature and the fitness is the response variable. In this work, the use of proxy models was investigated as a way to guide the search process in a FMS analysis.

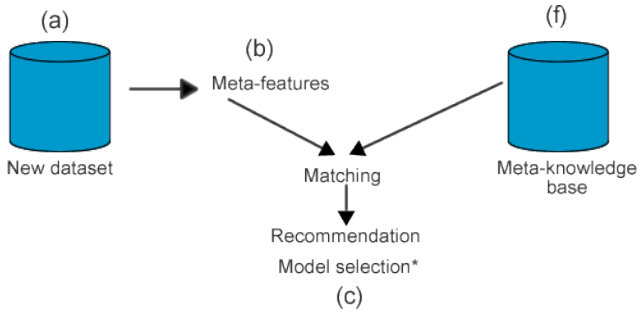


Figure 2.3.1: Meta-learning process in advisory mode.

Source: Based on figure 1.2 of [154]

2.4 Full model selection

The full model selection analysis looks for the best combination of several factors in order to obtain the lowest misclassification rate in the dataset under analysis. These factors are feature selection, data preparation, and the selection of a learning algorithm with its hyper-parameters tuned [57]. As this analysis is play a central role in this work, Eq. 2.1 is provided as a formal definition and is based in the one presented in [149]. In the same way all the involved factors and the performance metrics employed in this work are described in the following sections of this chapter.

$$a^*_{wA}, p^*, f^* \in argmin \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(a_{wA}, p, f, D^{(i)}_{train}, D^{(i)}_{validation}) \\ a^{(i)} \in A, p^{(i)} \in P, f^{(i)} \in F, w_A \in W_A \tag{2.1}$$

Where:

A = Set of available learning algorithms
P = Set of available data-preparation algorithms including $\varnothing$
F = Set of available feature selection algorithms including $\varnothing$
W_A = Set of hyperparameters of a learning algorithm a
D_{train}, D_{val} = Disjoint partitions of the dataset under analysis
$\mathcal{L}$ = Loss function calculated on the validation set (misclassification rate)
$argmin$ = Values that obtains the lowest misclassification rate on the loss function $\mathcal{L}$

Given a set of learning algorithms A, data preparation techniques P and feature selection algorithms F, the goal of the FMS is to determine the combination of algorithms: $a^*_{wA} \in A$ (a machine learning algorithm with an specific configuration in its hyper-parameter values), $p^* \in P$ and $f^* \in F$ with the lowest misclassification rate. The misclassification rate is estimated over the dataset D and this dataset is split in two disjoint partitions ($D^{(i)}_{train}$ and $D^{(i)}_{validation}$ for $i = 1, 2, ..., k$). The misclassification rate is calculated with the loss function $\frac{1}{k}\sum_{i=1}^{k}\mathcal{L}(a_{wA}, p, f, D^{(i)}_{train}, D^{(i)}_{validation})$, training the algorithm a_{wA} in the partition $D^{(i)}_{train}$, and evaluated in the partition $D^{(i)}_{validation}$. The data partitions are previously transformed by p and f.

2.4.1 Factors that make up the FMS analysis

As mentioned before, the FMS analysis involves the combination of important factors that can lead to models with higher accuracy. In addition to such factors, it is necessary to employ metrics to measure the performance of the models evaluated during the search stage and that depend on what kind of learning task is performed (classification or regression). The rest of this section describes each one of the associated factors and the performance metrics employed in this work.

2.4.1.1 Data preparation or data preprocessing

Data preparation is an important step in a data mining process, if there is much irrelevant and redundant information present or noisy and unreliable data, the knowledge discovery during the training phase is more difficult. The set of techniques that comprehends a data preparation task are used to pre-process the data to a suitable form for building models [38]. There are different ways to perform the data-preparation: from linear and non-linear scaling of the input and outputs, homogenization of the variability, principal component analysis, among others [84]. This stage includes data cleaning operations, data integration, data transformation and information reduction (includes feature and instance selection) [64]. A visual representation of

the aforementioned task is shown in Fig. 2.4.1.

Data Cleaning

Includes operations that correct bad data, filter some incorrect data out of the dataset and reduce the unnecessary detail of data. Treatment of missing and noise data is included here. Other cleaning data tasks involve the detection of discrepancies and dirty data (fragments of the original data which do not make sense) [63].

Data Transformation

In this pre-processing step, the data is converted or consolidated so that the mining process result could be applied or may be more efficient. Sub-tasks inside data transformation are the smoothing, the feature construction, aggregation or summarization of data, normalization, discretization and generalization [63].

Data Integration

It comprises the merging of data from multiple data stores. This process must be carefully performed in order to avoid redundancies and inconsistencies in the resulting dataset. Typical operations accomplished within the data integration are the identification and unification of variables and domains, the analysis of attribute correlation, the duplication of tuples and the detection of conflicts in data values of different sources [63].

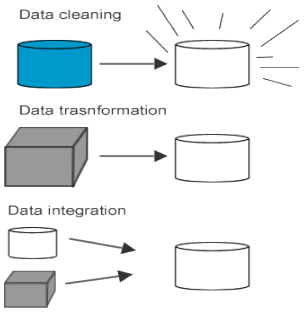


Figure 2.4.1: Forms of data preparation.

Source: Based on figure 1.3 of [63].

2.4.1.2 Feature selection

Feature selection consists of choosing a subset of features that ideally is enough to describe the target class. Considering this definition, the redundant and irrelevant features should be removed [139]. The feature selection problem can be defined as to find a projection of X to X', where X is a binary feature vector $X = \{x_1, x_2, ..., x_n\}$ with $x_i \in \{0, 1\}$ for each $1 \leq i \leq n$ and $X' \subseteq X$. Usually feature selection is performed in consideration to the fitness of a function $p(X')$ [54]. Feature selection methods can be classified in two approaches: individual evaluation and subset evaluation. Individual feature evaluation assesses each feature individually according to its relevance which leads at the end to a feature ranking. The drawback of individual evaluation is that it is incapable of eliminating redundant features because they have the same rank. Differently, subset feature evaluation can overcome the inconvenience of individual evaluation, it uses certain search strategies to select and evaluate a subset of features according to certain evaluation measures and then compares it with the previous best one. From this classification, three main approaches can be identified based on the relationship among the inductive learning method and the feature selection algorithm: filters, wrappers, and embedded methods [85].

- **Filter methods** are feature selection algorithms totally independent of any predictors. Applied directly on the training data, the filter approach is based on feature ranking techniques that use an evaluation criterion and a threshold to determine the feature relevance and decide whether to keep it or discard it. The feature relevance is determined by its capability to provide useful information about different classes.

- **Wrapper methods** are feature selection based on three components: a search strategy, a predictor, and an evaluation function. The search strategy determines the subset of features to be evaluated. The predictor can be any classification method and its performance is used as the objective function to evaluate the subset of features defined by the search strategy to find the optimum subset that gives the best accuracy of it. The wrapper approach outperforms the filter approach but is more time-consuming and requires more computational resources.

- **Embedded methods** incorporate an interaction between feature selection and the learning process. Therefore, the solution is reached faster than wrappers because they make better use of the available data and avoid retraining the predictors for every selected feature subset. Embedded methods integrate a

regularized risk function that is optimized taking into account the features designating parameters and the predictor parameters.

2.4.1.3 Hyper-parameter optimization

In Bayesian statistics, a hyper parameter is a parameter of a prior distribution; the term is used to distinguish them from parameters of the model for the underlying system under analysis. For example, when we use the beta distribution to model the distribution of the parameter p of a Bernoulli distribution, then p is a parameter of the underlying system, and α and β are parameters of the prior distribution (beta distribution), hence hyper-parameters [128]. In the machine learning context, the hyper-parameters are those parameters that define a set of learning functions $\{f(x, w_A), w_A) \in A\}$, where x is an input feature vector, w_A is an input vector of hyper-parameters that depends on the models belonging to A, which is a set of models [84].

Hyper-parameter optimization on the other hand, is the process of tweaking all parameters of a model not learned by gradient descent. As an example, considering a fully connected neural network, its weights can be learned from data, the other settings of the network can't. These hyper-parameters include the number of hidden layers, the number of neurons per hidden layer, the learning rate, and more. Hyper-parameter optimization methods systematically try multiple choices for hyper-parameters on the validation set. The best performing set of hyper-parameter values is then evaluated on a second held-out "test" set to gauge the true model performance. Different hyper-parameter optimization methods differ in the algorithm they used to propose new hyper-parameter settings.

2.4.1.4 Evaluation metrics

The right evaluation of the learning models in supervised learning is one of the main topics in the field of pattern recognition. Once the model is built, it is necessary to evaluate its generalization capacity in unknown samples and to choose the most adequate to a dataset. Some of the most important metrics for both classification and regression [138] are defined below.

Metrics employed in classification tasks

Some of the most popular metrics to evaluate the performance of classification models are detailed below. In Figure 2.4.2 an example of a confusion matrix employed in binary classification is shown, and from it the presented fitness metrics are taken:

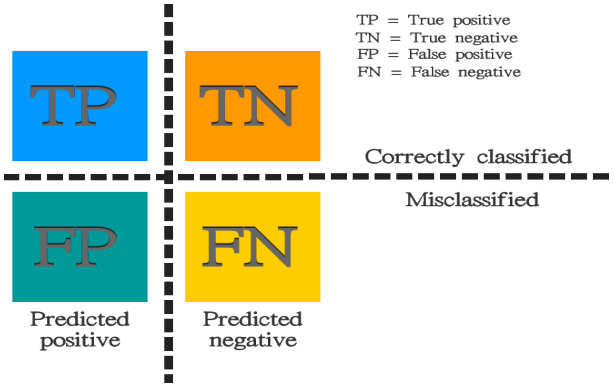


Figure 2.4.2: Confusion matrix for a binary classification problem.

Accuracy

This is the most common evaluation metric for a classification model. Is defined as the degree of right predictions of the model (in its inverse form can be understood as the misclassification rate).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

$$Error = \frac{FP + FN}{TP + TN + FP + FN} \tag{2.3}$$

Balanced Error Rate (BER)

The BER is the average of the ratio of the incorrectly classified samples per label set over all label sets [37].

$$BER = 1 - \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \tag{2.4}$$

Metrics employed in regression tasks

In the regression problems the output of a model is continuous or real instead of discrete, for this reason, the performance metrics are different. Some of the most

popular metrics for regression models are detailed below.

Mean absolute error (MAE)

This metrics measures the mean magnitude of the error in a set of predictions without considering its direction. It is the mean over the test samples of the absolute differences among the predictions and the real observations where all the individual differences have the same weight.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y_j}| \tag{2.5}$$

Root mean square error (RMSE)

This metric is a quadratic punctuation rule that also measures the magnitude of the average error, It is the square root of the differences among predictions and real observations.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y_j})^2} \tag{2.6}$$

2.5 Summary

In this chapter relevant concepts of the central axis of this work were analyzed. As the full model selection analysis can be applied to a wide range of problems in the machine learning field, the two best known categories in this area: supervised, and unsupervised learning were described. The regression and classification tasks were described because both approaches were employed in the experiments performed in this Thesis, the classification approach in the selection of models for high volume datasets and the regression approach for the construction of proxy models. On the other hand, the use of clustering analysis was explored in order to solve the FMS problem through the meta-learning approach. This last approach along with proxy models were also briefly described in this section. Finally, the FMS problem was defined in conjunction with the factors involved in it (data-preparation, feature selection, hyper-parameter optimization and evaluation metrics).

Chapter 3

High volume datasets, tools of the Big Data paradigm and characteristics of those datasets

As mentioned in Chapter one, more data has been created between 2012 and 2014 than in the entire history of the human race. Some possible causes of this phenomenon are the emergence of paradigms as social networks and the internet. Despite science has traditionally dealt with challenges handling large volumes of data in complex systems, the massive data generations in scopes outside scientific research has favored the popularization of the analysis of bigger datasets under the paradigm know as Big Data. This paradigm refers to data objects that are so large and/or complex that it cannot be perceived, acquired, managed, and processed by traditional information technology and software/hardware tools within a tolerable time [28]. The term Big Data was coined to represent growing volumes of data. Along with volume, the term also incorporates three more attributes, velocity, variety, and value [48] as follows:

- Volume: represents the ever-increasing and exponentially growing amount of data.
- Velocity: represents the amount of data generated with respect to time and a need to analyze that data in near-real time for some mission critical operations.
- Variety: represents variety in data formats.
- Value: This is the most important aspect of Big Data. The data is only as valuable as its utilization in the generation of actionable insight

This definition is the most extended one of Big Data, and once in a while a new definition is added to literature but with more V's. This definition is ambiguous and does not provide rules to identify a dataset that belongs to the Big Data paradigm, therefore in order to avoid confusions and for the sake of clarity we prefer the term of high-volume datasets for those that are big enough to not fit on the main memory (the training set without reductions and the learning algorithm) of a common personal computer. In our experiments we explored the use of tools coming from the Big Data paradigm, therefore, along this chapter were described as well as some important characteristics of datasets.

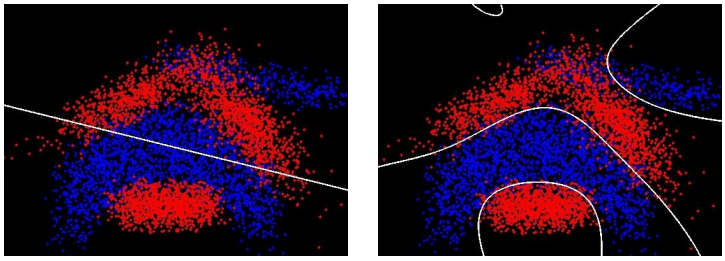
3.1 Model selection and high-volume datasets

The choice of a learning algorithm to a specific dataset is not a trivial task, it involves the testing of several models until to find the most suitable. Though more data could be thought as a silver bullet to machine learning problems, data alone is not enough, no matter how much of it you have. Let's consider learning a Boolean function of 100 variables from a million examples. There are $2^{100} - 10^6$ examples with unknown class. In the absence of further information, there is no way to do this [50]. Similarly, if a linear classifier is trained on a non-linear dataset, no matter how many data is available, the performance of such classifier is not going to improve. An example of that is provided in Fig. 3.1.1. This situation is known as the "No Free Lunch" theorem. It states that given a set of all functions F and a set of benchmark functions $F1$, if algorithms $A1$ is better on average than algorithm $A2$ on $F1$, then algorithm $A2$ must be better than algorithm $A1$ on $F - F1$ [51]. In simply words, basically states that no machine learning algorithm is the best for all problems. Thus, the correct choice of algorithm often remains unclear unless the algorithms are tested through trial and error [53].

From this starting point, the exploration of the search space imposed by the FMS problem is infinite even if restrictions are imposed on the hyper-parameter values of the model [57]. Let's consider a reduced example of a model selection exploration with L algorithms, M parameters and N levels for each parameter. Supposing that the complexity of a model λ is bounded by λ_0, the complexity of the search will be bounded by $L \times M \times N \times \lambda_0$. Taking into account that complexity and processing time of many machine learning algorithms are related to the number of examples in the training set, an adequate model search in large datasets could take too much time that the search becomes meaningless.

Although, the size of the dataset can be reduced through sampling and instance selection techniques, it could suppose the loss of model's quality and an exploration

in a different direction of the FMS problem. Therefore, to expand the FMS analysis capabilities and its utility to non-expert practitioners, we explored different ways to be capable to analyze sets with high-volume of data, using all the instances and features.



(a) Linear SVM on Banana dataset.

(b) RBF SVM on Banana dataset.

Figure 3.1.1: Performance of different models in the same dataset.

Source: https://qph.fs.quoracdn.net/main-qimg-9017c48fbc223cdcf414ce59ee07136f-c

3.2 MapReduce and the divide and conquer paradigm

One of the most powerful techniques for solving problems is to break them down into smaller, more easily to solve pieces. Smaller problems are less overwhelming and, they permit us to focus on details that are lost when we are studying the entire problem. In a divide-and-conquer algorithm, a problem is divided into smaller subproblems that are solved, and then, merged into one solution of the full problem [141].

In the same vein, the MapReduce programming model was introduced by Dean & Ghemawat in 2004 to enable the parallelization and distribution of big scale computation to analyze huge datasets. MapReduce has become almost a synonym of the Big Data paradigm, even though is one of the many available tools to handle the tasks of this field. This programming model was designed to work under the master-slave communication model. In the MapReduce programming model a computing task is specified as a sequence of stages: map, shuffle and reduce that works on a dataset $X = \{x_1, x_2, ..., x_n\}$. The map step applies a function μ to each value x_i to produce a finite set of key-value pairs (k, v). To allow for parallel execution, the computation of function $\mu(x_i)$, must depend only on x_i. The shuffle step collects all the key-value pairs produced in the previous map step, and produces a set of lists, $L_k = (k; v_1, v_2, ..., v_n)$ where each of such lists consists of all values v_i, such that $k_i = k$ for a key k assigned in the map step. The reduce stage applies a function ρ to

each list $L_k = (k; v_1, v_2, ..., v_n)$, created during the shuffle step, to produce a set of values $y_1, y_2, ..., y_n$. The reduce function ρ is defined to work sequentially on L_k but should be independent of other lists L_k, where $k^{'} \neq k$ [67].

Examined in detail, MapReduce is nothing, but the divide-and-conquer technique mentioned earlier [140]. In the high-volume datasets analysis, the main problem is the size of the data; now with the MapReduce paradigm, we can work on this data in parallel and get the desired results in the required timeframe. The logical steps involved in the process are as follows:

1. Divide the dataset into many chunks.
2. Execute our map function on them in parallel.
3. Start conquering and group the map outputs as per their key.
4. Execute our reduce function on them in parallel.

Figure 3.2.1 shows a representation of the **map, shuffle** and **reduce** stages on a dataset represented by colors (for an easier visual interpretation).

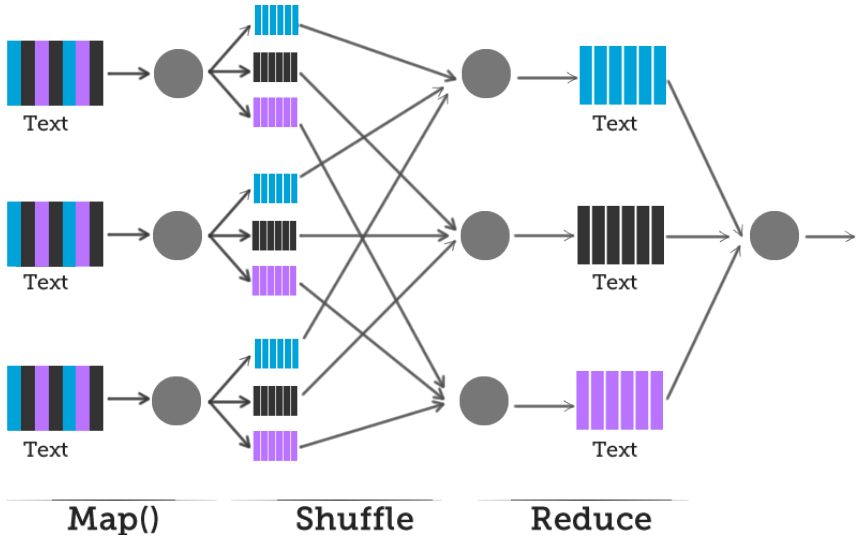


Figure 3.2.1: The MapReduce Programming Model: Map, Shuffle, and Reduce.

As it can be seen in Fig. 3.2.1, MapReduce is an instance of the divide-and-conquer technique for algorithms design as in other parallelization techniques. Some advantages that MapReduce incorporates in its design over other platforms for data analysis are [134]:

- **Low-cost unreliable commodity hardware:** Instead of using expensive, high-performance, reliable symmetric multiprocessing or massively parallel processing machines equipped with high-end network and storage subsystems, the MapReduce framework is designed to run on large clusters of commodity hardware.
- **Extremely Scalable RAIN (Redundant Array of Independent Nodes) cluster:** Instead of using centralized RAID (Redundant Array of Independent Disks) based SAN (Storage Area Network) or NAS (Network-Attached Storage) storage systems, every MapReduce node has its own local off-the-shelf hard drives. These nodes can be taken out of service with almost no impact to still-running MapReduce jobs.
- **Fault-tolerant:** In case of failure, MapReduce applies a mechanism to replicate the information in order to keep running the MapReduce tasks. To handle crashed nodes, system administrators simply take crashed hardware off-line. New nodes can be plugged in at any time without hassle.
- **Highly Parallel:** The most important contribution of the MapReduce framework is its ability to automatically support the parallelization of task executions. MapReduce's shared architecture makes it scalable and ready for parallelization.

3.2.1 Apache Spark

Apache Spark is an open-source distributed general-purpose cluster computing framework with (mostly) in-memory data processing engine that can do ETL (Extract, Transform and Load), analytics, machine learning and graph processing on large volumes of static data (batch processing) or dynamic data (streaming processing). In contrast to other MapReduce-based frameworks (such as Hadoop), most of the processing stages are done in memory and hence most of the time provides better performance for certain applications as iterative algorithms and interactive data mining [90], therefore is appropriated to be used in the FMS problem.

The RDD (Resilient Distributed Dataset) is the cornerstone of Apache Spark. A RDD is a resilient and distributed collection of records spread over one or many

partitions. Using RDD Spark hides data partitioning and so distribution that in turn allowed them to design parallel computational framework with a higher-level programming interface (API). The features of RDDs (decomposing the name):

- **Resilient**, i.e. fault-tolerant with the help of RDD lineage graph and so able to recompute missing or damaged partitions due to node failures.
- **Distributed** with data residing on multiple nodes in a cluster.
- **Dataset** is a collection of partitioned data with primitive values or values of values, e.g. tuples or other objects (that represent records of working data).

RDDs support two kinds of operations:

- **Transformations** - lazy operations that return another RDD.
- **Actions** - operations that trigger computation and return values.

3.3 Full model selection, high-volume datasets, MapReduce and its relation with complex systems

Over the past few years there has been an exponential growth in the rate of available data sets obtained from complex systems, ranging from the interconnection of millions of users in social media data, chemo-informatics, hydro-informatics to the information contained in the complex biological data sets [73]. A complex system is a system composed of many interacting parts, such that the collective behavior of the parts together is more than the sum of their individual behaviors. Classic examples of complex systems are: Condensed matter systems, ecosystems, economic and financial markets, the brain, the immune system, insect colonies, flocking or schooling behavior in birds or fish, among others [112]. An initial definition of a complex system can be given as:

- A system that exhibits behavior that is often non-linear due to large number of interacting components. The interactions between the components, e.g. cells, may well be non-linear too.
- A system whose behavior cannot be predicted merely by investigating the behavior of the individual components in isolation.

A component of a complex system is any part of the system that can be reasonably compartmentalized, either by physical or abstract partitions, on the basis that it forms a functional part of the whole. Linearity on the other hand is defined mathematically as a behavior that satisfies the properties:

- *additivity:* $f(x) + f(y) = f(x + y)$
- *homogeneity:* $a \cdot f(x) = f(ax)$

Neither of these properties is present in pure non-linearity: Let's consider the combination of behaviors in complex systems as mathematical functional outputs, as defined above, which produce other observable behavior where additivity and homogeneity do not hold. To define non/linearity more clearly, a linear behavior, in terms of system states, can be predicted with knowledge of any previous state and then aggregating a fixed amount over a parametric index to the desired index, whereas this is not always possible with non-linear behavior. In the case of complex systems non-linearity is a key property. Open complex systems invariably interact with the environment, which provides a context to their behavior through feedback mechanisms. Conversely, closed complex systems have no such interaction and are therefore independently complex. **Interaction** in the context of complex systems can be defined as any communicative action between two or more entities where the entities can be either physical objects, such as molecules or cells, or abstract concepts, such as behavior. Complex systems exhibit certain characteristics [118] as:

- **Complexity in behavior:** The behavior is defined as a pattern of state changes where in the states may refer to both environmental and system states. The states of the system as well as individual models that mimic the system can be highly nonlinear. The integration of models must preserve the behavioral complexity of the original individual entities.
- **Complexity of component parts:** Complex systems are composed of components that may well exhibit a degree of complexity themselves.
- **Compartmentalization:** Modeling often involves compartmentalization of the system, which can be based on both real compartments and artificial ones. A model integration strategy must respect any compartmentalization that exists within the original models, since these constitute implicit assumptions about what the models represent.
- **Environmental context:** Complex systems often interact within the environment which can affect overall behavior.

The explosion in the quantity of information in the world enabled the creation of the Big Data paradigm in order to handle and analyze data of complex systems. This paradigm also has allowed to researchers to make more accurate models that mimics the behavior of such systems. The tools to face the Big Data problem as MapReduce, are based on the divide and conquer programming technique and, therefore, MapReduce makes use of the superposition property as in a linear non-complex systems [121]. Taking this into account, we can see that Big Data is not a complex system, but a tool to model them. In the other hand, the full model selection problem is composed by all the possible models that can be built given the considered methods and, as was stated earlier, the search space of the FMS problem is almost infinite. With the given definition of complex systems, it can be seen that the FMS problem can be considered as a complex system, because the output of the system cannot be calculated adding the outputs of each one of its components (selection of learning algorithm, hyper-parameter optimization, feature selection and selection of data-preparation techniques). In summary, an infinite search space of a complex system with a high computational cost to explore each possible solution, makes of the FMS problem in high volume datasets a hard problem and a complex system.

3.4 Relevant characteristics of the analyzed datasets

As was discussed earlier, the exploration of a vast search space and the time-consuming process of the model construction with high-volume datasets impose restrictions to the experiments and the number of datasets to analyze. Despite that, the big quantity of different types of available datasets makes impossible to select a representative sample of such sets. In order to be capable to consider that the employed datasets represent a variety of problems, two important characteristics of them were considered: The Intrinsic dimension and the Shape of the datasets.

The **Intrinsic Dimension (ID)** of a given dataset $X_N \equiv x_i{}_{i=1}^{N} \subset R^D$ is the minimum number of parameters needed to capture, and describe, all the information carried by the data. X_N is said to have an intrinsic dimension equal to $d \in 1, ..., D$ if its elements lie entirely within a d-dimensional subspace of R^D [99]. Information about the ID of a dataset is relevant in many contexts, for instance in molecular simulations, where often a dimensionality reduction is required, or in bioinformatics, or in image analysis where the ID is a suitable descriptor to distinguish between different kinds of image structures [60]. According to the statistical learning theory, the capacity and generalization capability of a given classifier may depend on the ID. More specifically, in the particular case of linear classifiers where the data are drawn from a manifold embedded through an identical map, the Vapnik-Chervonenkis

dimension of the separation hyperplane is $d+1$. Since the generalization error depends on the Vapnik-Chervonenkis dimension, it follows that the generalization capability may depend on the ID [21]. Therefore, ID can be employed as a way to assure that each dataset represents a different problem.

Shape on the other hand, is a crucial component in many of scientific analysis, with examples including geomorphology, powder particle characterization, and biology. A convexity analysis can be used for a variety of applications, for instance shape decomposition which in turn can be used to compute shape similarity and has been applied to object indexing [168]. A dataset with convex shape is the one that haves the shape of a simple polygon whose interior is a convex set [127]. The convex set is defined as follows: given a pair of points x and y in E^n, the line segment $\overline{xy}$ joining x and y is the set of all points of the form $\alpha x + \beta y$ where $\alpha \geq 0$, $\beta \geq 0$ and $\alpha + \beta = 1$, a set S is convex if for each pair of points x and y in S it is true that $\overline{xy} \subset S$ [91]. On the other hand, a non-convex dataset is the one that cannot meet the previous criteria. In Figure 3.4.1 a convex shape and a non-convex shape are shown. As this property is capable to separate the datasets in two categories, with the shape analysis we can know to which category belongs a dataset and therefore, to know the performance of the FMS analysis in both categories.

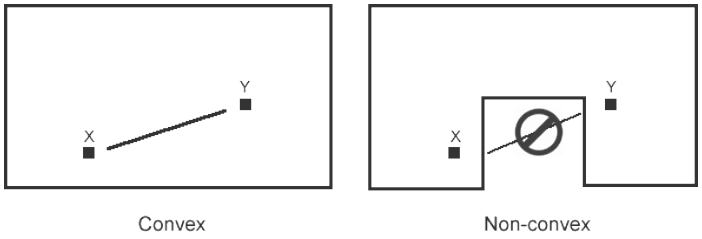


Figure 3.4.1: Example of complex and non-convex sets.

3.5 Summary

In this chapter, the problem of performing the FMS analysis in high-volume datasets was described. The concept of high-volume datasets was defined and separated from the popular paradigm of Big Data. Despite this fact, in order to be capable to perform the analysis in the entire datasets, the use of tools from the Big Data paradigm as MapReduce and Apache Spark were explored and described. The relation of the Complex systems with the FMS problem was described as well as some important characteristics of the datasets employed in this work.

Chapter 4

Soft computing techniques

To be capable to face the search space imposed by the FMS problem, the use of several techniques from the soft computing field were investigated. For the exploration, two well-known bio-inspired algorithms were employed: The Genetic Algorithm (GA) and the Particle Swarm Optimization (PSO) algorithm. The use of the fuzzy logic paradigm was used in the experiments that involved the use of proxy models and to investigate the use of the meta-learning paradigm to solve the FMS problem. Along this chapter, these key elements that are part of the performed experiments were described.

4.1 Genetic Algorithms

Genetic algorithms are a type of optimization algorithm, meaning they are employed to finding the optimal solution to a given computational problem that maximizes or minimizes a particular function. Genetic algorithms imitate the biological processes of reproduction and natural selection to solve for the 'fittest' solutions. Like in evolution, many of a genetic algorithm's process are random, however this optimization technique allows us to set the level of randomization and the level of control.

Components and structure

Since genetic algorithms are designed to simulate a biological process, much of the relevant terminology is borrowed from biology. The basic components common to almost all genetic algorithms are:

- A fitness function for optimization.

- A population of chromosomes.
- Selection of which chromosomes will reproduce.
- Crossover to produce next generation of chromosomes.
- Random mutation of chromosomes in new generation.

The **fitness function** is the function the algorithm is trying to optimize. "Fitness" concept is taken from evolutionary theory. It is used because the fitness function tests how "fit" each potential solution is. The term **chromosome** refers to a numerical value or values that represent a candidate solution to the problem that the genetic algorithm is trying to solve. Each candidate solution is encoded as an array of parameter values, a process that is also found in other optimization algorithms. If a problem has $N_{parameter}$ dimensions, then typically each chromosome is encoded as an $N_{parameter}$-element array: chromosome $= [p_1, p_2, \ldots, p_{N_{parameter}}]$ where each p_i is a particular value of the i^{th} parameter. A genetic algorithm begins with a randomly chosen assortment of chromosomes, which serves as the first generation (initial population). Then each chromosome in the population is evaluated by the fitness function to test how well it solves the problem at hand. Now the **selection operator** chooses some chromosomes for reproduction based on a probability distribution defined by the user. The fitter a chromosome is, the more likely it is to be selected. The **crossover operator** resembles the biological crossing over and recombination of chromosomes to create two offspring. **Mutation operator** randomly flips individual elements in the new chromosomes, typically, mutation happens with a very low probability. Genetic algorithms are iterated until the fitness value of the "best-so-far" chromosome stabilizes and does not change for many generations. This means the algorithm has converged to a solution [19].

4.2 Particle Swarm Optimization

The PSO algorithm is a biologically inspired computational search and optimization method developed in 1995 by Eberhart and Kennedy. The algorithm emulates the behavior of animal societies that don't have any leader in their group or swarm, such as bird flocking and fish schooling. Typically, a flock of animals that have no leaders will find food by random, follow one of the members of the group that has the closest position with a food source (potential solution). The flocks achieve their best condition simultaneously through communication among members who already have a better situation. The animal which has a better condition will inform it to its flocks

and the others will move simultaneously to that place. This would happen repeatedly until the best conditions or a food source is discovered. The process of PSO algorithm in finding optimal values follows the work of this animal society. Particle swarm optimization consists of a swarm of particles, where a particle represents a potential solution [129]. PSO makes use of a velocity vector to update the current position of each particle in the swarm. The position of each particle is updated based on the social behavior that a population of individuals, the swam in the case of PSO, adapts to its environment by returning to promising regions that were previously discovered. The process is stochastic in nature and makes use of the memory of each particle, as well as the knowledge gained by the swarm as a whole [153]. The outline of a basic PSO algorithm is as follows:

1. Start with an initial set of particles, typically randomly distributed throughout the design space.

2. Calculate a velocity vector for each particle in the swarm.

3. Update the position of each particle, using its previous position and the updated velocity vector.

4. Go to step 2 and repeat until convergence.

4.3 Fuzzy logic

The fuzzy set theory covers an important quantity of methods and techniques to capture human knowledge and to deal with the uncertainty in a wide range of problems. Due its usefulness, part of this paradigm was employed in some experiments that made use of proxy models based in fuzzy rules and in the design of an alternative to the meta-learner algorithm K-NN. This last algorithm made use of fuzzy clustering techniques and therefore of validity indexes for those. All these concepts are described below. The principal objective of Fuzzy Logic (FL) is the formalization/mechanization of the capacity of human beings to reason and make decisions in an environment of uncertainty. There are many misconceptions about fuzzy logic. To begin with, fuzzy logic is not fuzzy. In large measure, fuzzy logic is precise. Another source of confusion is the duality of meaning of fuzzy logic. In a narrow sense, fuzzy logic is a logical system. But in much broader sense which is in dominant use today, FL is much more than a logical system. More specifically, fuzzy logic has many facets [164].

Fuzzy-set-theoretic facet

This facet is focused on fuzzy sets, that is, on classes whose boundaries are unsharp, e.g., the class of beautiful women, the class of honest men and the class of tall mountains. In more detail, fuzzy sets are graduated in the sense that membership in a fuzzy set is a matter of degree. A fuzzy set, A, in a universe of discourse, U, is defined by a membership function which associates with each object, $u \in U$, the degree to which u is a member of A. A fuzzy set is basic if its membership function takes values in the unit interval. More generally, the membership function may take values in a partially ordered set [164].

Logical facet

The logical facet of FL (FLl), is fuzzy logic in its narrow sense. FLl may be viewed as a generalization of multivalued logic. Truth values in FLl are allowed to be fuzzy sets [164].

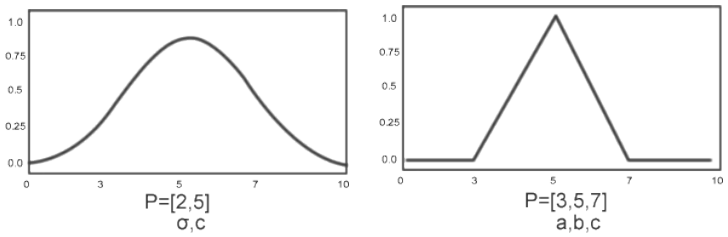
Relational facet

The relational facet (FLr), is focused on fuzzy relations and, more generally, on fuzzy dependencies. In FLr, a granulated function, $f*$, is described as a collection of fuzzy if-then rules of the form: if X is A then Y is B , where A and B are fuzzy sets carrying linguistic labels like small, medium, and large. In this sense, X and Y are linguistic variables. The concept of a linguistic variable and the associated calculi of fuzzy if-then rules play pivotal roles in almost all applications of fuzzy logic. A granulated function, $f*$, may be viewed as a summary of f, with $f*$ being a granular value of f. In this perspective, perception of a probability distribution may be described as a granular probability distribution [164].

4.3.1 Fuzzy sets

The concept of fuzzy sets was introduced as a generalization of the classical set theory. In the real world, many classes of objects are not as well-defined as a regular set theory would suggest. For example a set of all tall persons includes of course all persons taller than 1.9m, but what about persons of length 1.8m?. The definition of a fuzzy set has what is needed to define a set that has inexact boundaries [94]. A fuzzy set is a generalization of a crisp set (traditional set). A fuzzy set $A^{'}$ in a universe of discourse X is characterized by a **membership function** $\mu_{A'}$ which assigns to each

element $x \in X$ a real number $\mu_{A'}(x) \in [0, 1]$ expressing the membership grade of x in the fuzzy set $A^{'}$ [148]. A membership function (μ) is a curve that defines how each point in the input space is mapped to a membership value. The shape and equations of two of the most commonly used membership functions are shown below.



(a) Gaussian. (b) Triangular.

Figure 4.3.1: Examples of membership functions.

Source: https://la.mathworks.com/help/fuzzy/foundations-of-fuzzy-logic.html

Equation of the Gaussian membership function (fig 4.3.1 (a)).

$$f(x; \sigma, c) = e^{\frac{-(x-c)^2}{2\sigma^2}} \tag{4.1}$$

Equation of the triangular membership function (fig 4.3.1 (b)).

$$f(x; a, b, c) = max(min(\frac{x-a}{b-a}, \frac{c-x}{c-b}), 0) \tag{4.2}$$

4.3.2 Fuzzy rules

Fuzzy rules are key tools for expressing pieces of knowledge in fuzzy logic [52], while a collection of rules is called a Fuzzy rule system. Those rule systems are tools that allow storing and represent the expert knowledge [156]. Such rules control the behavior of any fuzzy system. Fuzzy expert systems are usually involved when process cannot be described by exact algorithms or when these processes are difficult to model with conventional mathematical models. A fuzzy rule-base is a set of “IF-THEN" rules that can be expressed as below:

$$IF \quad x_1 \quad is \quad A_1^k \quad Then \quad y_k \quad is \quad B^k \tag{4.3}$$

Where:

$i = 1, ..., n$; = 1 to number of antecedents
x_i = Input value of the i-th antecedent of the rule
A_i^k = K-th fuzzy set of the i-th antecedent
y_k = Output value of the consequent
B^k = K-th fuzzy set of the consequent

The part of the rule, $If \quad x_1 \quad is \quad A_1^k$, is known as antecedent or premise and the terms

A_1^k and B^k are linguistic variables defined through fuzzy sets in the universe of discourse X and Y respectively. The part ,$y_k \quad is \quad B^k$, is known as consequent or conclusion [106].

4.3.3 Fuzzy classification

Fuzzy classification is a natural extension of the traditional classification, the same way fuzzy sets extend classical sets. In a sharp classification, each object is assigned to exactly one class, meaning that the membership degree of the objects is 1 in this class and 0 in all the others. The belonging of the objects in the classes is therefore mutually exclusive. In contrast, fuzzy classification allows the objects to belong to several classes at the same time; furthermore, each object has membership degrees which express to what extent this object belongs to the different classes. Formally described, let O be an object characterized by a t-dimensional feature vector x of a universe of discourse U. Let $C_1, \ldots, C_n$ be a set of classes which is given a priori or has to be discovered. Fuzzy classification calculates a membership vector $M = \{m_1, \ldots, m_n\}$ for the object O. The vector element $m_i \in [0, 1]$ is the degree of membership of O in the class C_i [156].

4.3.4 Fuzzy clustering

The objective of clustering is to partition the data set X into c clusters. For the time being, assume that c is known, based on prior knowledge. Fuzzy partitions can be seen as a generalization of hard partition. A fuzzy partition of the data set X can be represented by a $c \times N$ matrix $U = [\mu_i, k]$, where $\mu_{i,k}$ denotes the degree of membership that the k-th observation belongs to the c-th cluster ($1 \leq k \leq N, 1 \leq i \leq c$). Therefore, the i-th row of U contains values of the membership function of the i-th fuzzy subset of X. The matrix U is called the fuzzy partition matrix [1]. Conditions for a fuzzy partition matrix are given by:

$$\mu_{i,k} \in [0,1], 1 \leq i \leq c, 1 \leq k \leq N \tag{4.4}$$

$$\sum_{i=1}^{c} \mu_{i,k} = 1, 1 \leq k \leq N \tag{4.5}$$

$$0 \leq \sum_{k=1}^{N} \mu_{i,k} < N, 1 \leq i \leq c \tag{4.6}$$

4.3.5 Fuzzy clustering validity indices

Clustering validity is an evaluation metric in order to determine how good is the result of the clustering process. There are several indices to determine the validity of crisp and fuzzy clusters [41]. Regarding fuzzy clustering, there are indices that make use only of the membership values of the partitions such as **Partition coefficient** and the **Entropy partition**. Those indexes are easy to calculate but are just useful for few well separated clusters and don't take into account the geometrical properties of the information. In order to overcome such problems, Xie & Beni (1992) defined a validity index that measures both the compactness and the cluster separation [161].

$$V_{XB}(U, V; X) = \frac{\sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^{m} \|x_k - v_i\|^2}{n(min(v_i - v_j))} \tag{4.7}$$

From equation 4.7, c is the number of clusters, n is instances number of the dataset, m is the fuzzifier term, $V = [v_1, v_2, \ldots, v_n]$ is a vector of cluster centers and $U = (u_{ij})$ is the fuzzy partition matrix composed by the membership degrees of each object in the dataset regarding the $i-$th cluster. The Fukuyama-Sugeno validity index is another popular index proposed by Fukuyama & Sugeno in 1989. This index is presented in equation 4.8 and simultaneously measures the compaction and the cluster separation. The term $\overline{v}$ is the geometric mean of the cluster centers $V = [v_1, v_2, \ldots, v_n]$.

$$V_{FS}(U, V; X) = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^{m} (\|x_k - v_i\|^2 - \|v_i - \overline{v}\|^2) \tag{4.8}$$

4.4 Summary

The key concepts and techniques employed to face the FMS problem in high volume datasets were described in this chapter. The two principal search strategies employed: the GA and PSO, were briefly described. On the other hand, due its usefulness and suitability, some methods and techniques from the fuzzy logic paradigm were employed. Fuzzy sets, membership functions, fuzzy rules systems and fuzzy classification techniques were of great interest in the construction of a proxy model based in a fuzzy-rules classification algorithm, therefore were also covered. Regarding the meta-learning approach followed in this work, the concepts as fuzzy clustering and fuzzy validity indexes were employed for the design of a new meta-learner algorithm, and there were explained above.

Part III

Experiments and contributions

Chapter 5

First approach to solve the FMS problem in high-volume datasets

The FMS problem has been widely investigated in datasets where the search of models does not represent a challenge in terms of time and the use of specialized algorithms and hardware to handle them. As it was explained in previous chapters, the FMS problem involves the exploration of a huge or almost infinite search space, even if restrictions are imposed to the hyper-parameter values. In order to find a first solution to this problem, two important bio-inspired algorithms were compared to obtain evidence of their suitability to solve the FMS problem in high volume datasets.

Although similar comparisons were made in the past ([130, 131]), the algorithms for model selection were built under different packages or frameworks. Those frameworks employed distinct algorithms for the feature selection and data-preparation step as well as learning algorithms. As an example, in [131] just SVM was contemplated, reducing in this way the search space explored by the proposed algorithm. The algorithms compared in this Chapter, a GA and a PSO were built under the MapReduce programming model in order to deal with datasets of any size, and in order to make fair comparisons, the GA and PSO used the same algorithms for all the steps of the FMS process.

In like manner, at the time the experiments were made, just one of the works in the literature was able to perform model selection in high volume datasets [142], while the K-NN algorithm proposed by Yu et al. [2002] was adapted to MapReduce. Those algorithms were employed to make comparisons and the acronyms used in the rest of this work are **Grid** for the work of Sparks et al. [2015] and **K-NN** for the work of Yu et al. [2002].

This first set of experiments are related to the first objective defined in this

document, in order to gain knowledge of which algorithm is better suited to FMS in high volume datasets. Furthermore, the contributions obtained were the developed framework to adapt population-based optimization algorithms to perform FMS under **MapReduce** and, of course the information obtained from the comparison itself.

The organization of this chapter is as follows: in Section 5.1 a revision of the relevant work for this experiment is provided, in Section 5.2 the proposed solution is described, Section 5.3 present the experiments performed and in Section 5.4 are the final considerations of this chapter.

5.1 Related work

From the analysis of the literature, some parts of the FMS problem has been addressed as an optimization task, using mainly four groups of search techniques: bio-inspired methods, grid searches, gradient descent and Bayesian methods methods.

Two important examples of bio-inspired methods, that is, PSO and GA, were introduced in Chapter 4 and, although the Artificial Immune System (used in [8]) and the Bat algorithm (present in Bansal and Sahoo [2015]) were not described, their functioning follows almost the same principles that in all bio-inspired methods but trying to mimic the behavior of Bats and of the Immune system. In the works within this category, the hyper-parameter optimization problem [70, 29, 8, 92] is addressed, coding the values of the hyper-parameters of a learning algorithm in a vector that represents a particle, bat or individual depending on the method used. These potential solutions are evaluated and according to their fitness and the mechanism of each technique, the search space is explored until the termination criterion is met or the quality of the solutions don't improve.

Regarding fitness evaluation, most of the works relie on a single criterion [70, 57, 58, 29, 92] and just a few, conducted the search evaluation using more than one criterion [9, 130, 131]. Concerning to multicriteria optimization, two main methods are employed to select the fittest solutions: weighted sum [9] and the Pareto front [130, 131]. Despite the usefulness of multicriteria optimization methods, the evaluation of additional objectives, increments considerably the time of the process, for this reason, the use of proxy models was proposed in Rosales-Pérez et al. [2015].

The FMS problem has been investigated, [57, 58, 9, 130, 131] just through bio-inspired methods. The search space of this problem is so huge that, a grid search could become easily intractable if a good exploration is performed or too poor if the levels of the factors are considerably reduced to make possible a faster exploration.

Grid search is the simplest way to optimize the hyper-parameters of a learning algorithm, evaluating all combinations of contemplated factors [25, 12, 23, 143, 83, 13,

86, 152, 76, 103]. Values of such factors must be necessarily discretized to be capable to explore the combination that otherwise could result unaffordable. Within the works of this category, most of them are focused on comparing the efficiency of grid search against other methods (random search [12, 13]), different fitness evaluation techniques (Leve one out vs K-fold cross validation [86, 152]) and the advantages and disadvantages of the model re-train vs re-search of models [23]. Some works of this group stand out of the rest by considering ways of reducing the time employed in the search process but without decreasing the quality of models using nested grids [76] and the hybridization of the grid search and a theoretical decision technique [83]. Another outstanding work deals with the hyper-parameter optimization of algorithms but considering the use of the MapReduce programming model to handle high volume datasets [143].

Gradient methods are mainly focused in the hyper-parameter optimization problem of learning algorithms as the Support Vectors Machine [10, 24] and Neural networks [89]. These methods are simpler and faster but, they are fallible because are strongly influenced by their starting point and are easily captured by local optima.

Bayesian methods are also among the popular alternatives to solve the hyper-parameter optimization problem, and they have been extended to address model-selection with hyper-parameter optimization [149, 87, 77]. The Sequential Model-Based Optimization or SMBO is the core optimization algorithm of many works in the literature. It constructs a regression model to predict the performance of the true model employed (a proxy model). SMBO iterates between fitting the model and gathering additional data based on the predicted performance. In the context of parameter optimization, the proxy model is fitted to a training set where an instance is made up with the parameters of the true model and its performance with this configuration. SMBO uses a criterion known as Expected Improvement, which is employed to select new regions in the search space to explore. SMBO has inherited a range of limitations inappropriate to the automated algorithm configuration setting. These limitations include a focus on deterministic target algorithms; use of costly initial experimental designs; reliance on computationally expensive models; and the assumption that all target algorithm runs have the same execution costs [77].

From the previous paragraphs, some key elements for the FMS methods presented in this Chapter were obtained. The first one is the preponderance of the bio-inspired methods to address the vast search space of the FMS problem, because a good exploration employing a Grid search could easily become untractable and the gradient-based methods are not suitable to deal with this problem.

As stated earlier, the FMS problem was explored only through bio-inspired methods and, among them, two stood out from the rest: PSO and GA. For this

reason, is of big interest to compare their suitability to the FMS problem in high volume datasets.

Although the use of MapReduce programming model has been proposed for hyper-parameter optimization of learning algorithms in high volume datasets, hyper-parameter optimization is just one part of the whole FMS problem. Synergy of a better suited search algorithm as the ones mentioned above and the use of MapReduce programming model could provide a good starting point to solve the problem addressed in this work.

Finally, just in two works of the ones that address the FMS problem [130, 131], was remarked the difficulty of selecting a single model from a population as a final model. Even though in those works, their multi-objective optimization nature makes hard to assess the quality of each solution to pin point the best of all, it is also true that even with a single fitness measure, choosing a possible model is also hard. Since the fitness measure is just an estimation of the true fitness in the test set, the best of all models could be the third and not the first in the ranking. Taking the experience provided by the aforementioned works, a new idea to explore to solve this problem is through an ensemble of all the best models.

To gain knowledge about which is the best method to perform the FMS analysis in high volume datasets between the two most important approaches in the literature, that is, **genetic algorithms** and **swarm-based methods**, a comparison was performed. In both approaches it was sought to cover the weak points of some works (The use of methods unable to explore a vast search space, inability to handle high volume datasets, selection of a single final model, among others) with the strongest points of others (the use of **MapReduce** programming model and classifiers ensemble) in order to find the best suited algorithm for this problem.

5.2 Bio-inspired algorithms to perform FMS in high volume datasets

The most commonly used optimization algorithms in the literature to solve the FMS problem are those based in swarms (PSO in Escalante et al. [2009] and Bat in Bansal and Sahoo [2015]) and Genetic Algorithms (present in Rosales-Pérez et al., Rosales-Pérez et al. [2014, 2015]), therefore, in the interest of seeking the most suitable algorithm to our problem, a GA and PSO were investigated. In the following sections, aspects as codification scheme, operators employed to update solutions and other key factors of both methods are described. In order to be able to analyze datasets of any size, the **MapReduce** programming model **was employed**. Algorithms developed

under this paradigm are shown in Appendix-A. Since two different search methods were analyzed, the search algorithm presented is a generic one and can be extended to any population-based search method. For an easier understanding, a flowchart of the search process performed by both methods is presented in Fig. 5.2.1.

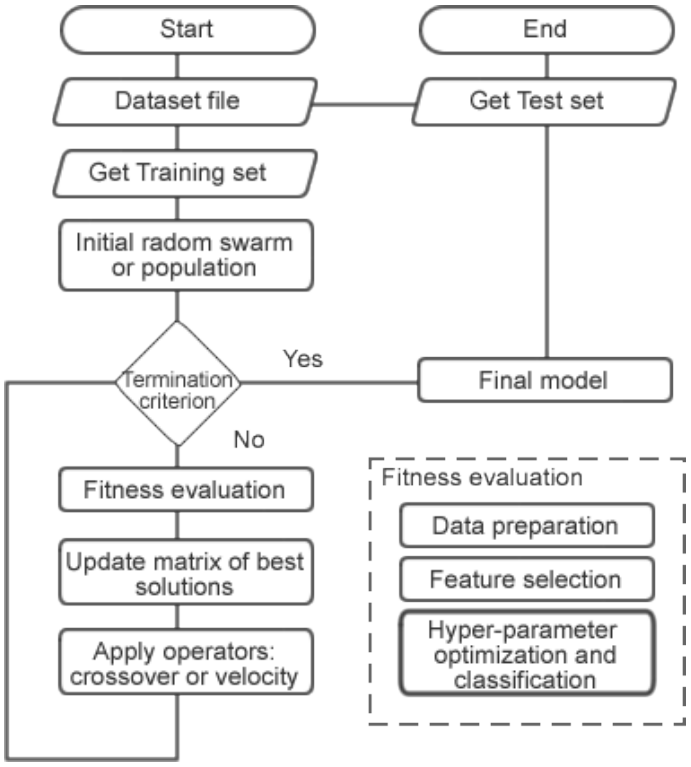


Figure 5.2.1: Flowchart of a generic FMS process.

5.2.1 Codification of solutions

The solutions encoded in a population based meta-heuristic method needs to be codified in a vector. This vector receives different names according to the algorithm employed, **individual** for a GA and **particles** for PSO. Depending on the nature of the meta-heuristic method, the codification scheme of such vector varies between real and binary. A real codification scheme was chosen because its adaptability of a wide range of meta-heuristic methods. This step, fitness calculation and final model construction of the optimization process involved in FMS **are the same for PSO** and the **GA** as well as other population-based algorithms that can be adapted to this framework. Is in the evolutive or Bio-inspired strategies as cross-over, selection,

replacement and mutation for the GA and velocity and trajectory adjustment for PSO, where there are important variations and are explained in the following sections.

The solution vector $x = [x_1, x_2, ..., x_{16}]$ is encoded as follows: In position 1 the fitness of the potential models is stored. Position 2 allows determining which operation will be done first: data-preparation or feature selection. Position 3 indicates if the data-preparation step will be done. Positions 4 to 6 are parameters for the data-preparation step (method identifier, parameter 1 and parameter 2). Position 7 determines if the feature selection step will be done. Positions 8 and 9 are for the feature selection step (Method identifier and number of features to be selected respectively). Positions 10 to 16 are for the machine learning algorithm construction. The range of values that every element in the vector can take is as follows: [0-100]; [0,1], [0,1], [1,30], [1,NF], [1,50], [0,1], [1,5], [1,NF], [1,6], [1,2], [1,4], [1,100], [1,60], [1,400], [-20,20] with NF = Number of Features. This range of values was selected in accordance with the maximum and minimum values that algorithms employed (see Table 5.2.1) for feature selection, data preparation and classification algorithms can take. Such algorithms are the ones available in the Apache Spark 1.6.0 (**MapReduce**) programming framework.

Table 5.2.1: Data-preparation methods, feature selection and classification algorithms used in this work.

Data-preparation algorithms	Feature standardization Normalization Principal component analysis (PCA) Shift and scale Discretization
Feature selection algorithms	Joint Mutual Information (JMI) Minimum Redundancy Maximum Relevance (mrMR) Interaction Capping (ICAP) Conditional Mutual Information Maximization (CMIM) Informative Fragments (IF)
Classification algorithms	Support Vector Machine (SVM) Logistic Regression (LR) Multilayer Perceptron (MLP) Decision Tree (DT) Random Forest (RF) Gradient-Boosted Trees (GBT)

5.2.2 Crossover, mutation and trajectory adjustment operators

Particle Swarm Optimization

One of the most popular and successful algorithms to perform the FMS analysis is the Particle Swarm Model Selection (PSMS) proposed by Escalante et al. [2009]. PSMS is based on the PSO algorithm, which is a population-based search inspired by the behavior of biological communities that exhibit both individual and social behavior [57]. PSMS is faster and easier to implement because it is based on a single operator. A set of potential solutions, in this case particles $S = \{x_1^t, x_2^t, ..., x_m^t\}$, is called a swarm. Each particle has a related velocity value that is used to explore the search space and the velocity of such particle at time t is as follows $V_i^t = [v_{i,1}^t, v_{i,2}^t, ..., v_{i,16}^t]$ where $v_{i,k}^t$ is the velocity for dimension k of the particle i at time t. **The search trajectories** are adjusted employing the following equations:

Velocity

$$v_{i,j}^{t+1} = W \times v_{i,j}^t + c1 \times r1 \times (p_{i,j} - x_{i,j}^t) + c2 \times r2 \times (p_{g,j} - x_{i,j}^t) \tag{5.1}$$

Adjustment of trajectory

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1} \tag{5.2}$$

From Equation 5.1, $p_{i,j}$ is the value in dimension j of the best solution found so far, also called personal best. $p_{g,j}$ is the value in dimension j of the best particle found so far in the swarm. The constants $c1, c2 \in \mathbb{R}$ are used to weight the influence of local and global best solutions. The values $r1, r2 \sim U[0, 1]$ introduce randomness into the search process. Inertia weight W controls the impact of the past velocity of a particle over the current one, influencing the local and global exploration. As in the original paper, the inertia weight is adaptive and specified by the triplet $W = (w_{start}, w_f, w_{end})$; where w_{start} and w_{end} are the initial values of W, w_f indicates the fraction of iterations in which W is decreased. W is decreased by $W = W - w_{dec}$ from the first iteration where $W = W_{start}$ to last iteration where $W = w_{end}$ and $w_{dec} = \frac{w_{start} - w_{end}}{Number\_of\_iterations}$ [57].

Genetic Algorithm

The genetic algorithm used, was designed as a fusion of some aspects of the approach known as CHC (Cross-generational elitist selection, Heterogeneous recombination, Cataclysmic mutation) proposed in Eshelman [1991] and others from the Eclectic GA shown in Morales and Quezada [1998]. From the CHC, the crossover operator HUX (Half Uniform Crossover) was obtained and adapted to real codification. From the Eclectic approach, the **selection** and **replacement** model known as **Vasconcelos Model (VM)** was taken. In VM, the best individual is mated with the worst in the population, in other words, the "k" individuals in the population are ranked by their fitness. Subsequently, VM will select individuals as follows: $(1, k), (2, k-1), ..., (k/2, k/2+1)$. The traditional mutation operator was preserved as well as the cataclysmic mutation operator (from the CHC); this last mutation operator was used as a way to avoid the stagnation in the search process. The use of real codification makes it necessary to use ad hoc crossover and mutation operators adapted from Haupt et al. [1998].

Crossover operator

$$Off_i = \beta P_{mn} + (1-\beta)P_{fn} \tag{5.3}$$

In Eq. 5.3, the $i-$th offspring (off_i) is obtained multiplying the $n-$th allele of parent 1 (P_{mn}) by a random number (β) in the range $[0, 1]$ added to the multiplication of the $n-$th allele of parent 2 (P_{fn}) by $1-\beta$.

Mutation operator

$$Off_i = Off_i + \sigma N_n(0, 1) \tag{5.4}$$

In Eq. 5.4, the $i-$th mutated offspring (Off_i) is obtained by adding the value of the $i-$th offspring (Off_i) to a normally distributed number ($\sigma N_n(0, 1)$) in the interval $[0, 1]$.

5.2.3 Fitness measure and final model construction

For both methods, the evaluation metric used was the Balanced Error Rate or BER employed in the original work that proposes the FMS paradigm ([57]), and

that was explained in Section 2.4 of this work. To asses this metric, the popular technique for model evaluations, K-fold cross validation was employed. In **K-fold cross-validation**, the data is randomly divided in K equal sized and mutually exclusive subsets (or folds). The model is estimated and validated k times; each subset in turn is reserved for the validation and the remaining data are used for estimation. The k prediction errors from different iterations are averaged to provide the overall prediction error estimate [15]. In the same way that in Escalante et al. [2009], experiments were conducted with different number of folds (2...10) without significant differences and, taking into account the computing time factor, the choice was the use of the 2-fold cross validation.

Final model construction

Regarding the choice of a final particle or individual as a final model, it is not a trivial task. Since the fitness measure of each potential model is just an estimation of the misclassification rate in the test set and not the real measure, there is no certainty in the choice based on their fitness that one particle is the best of the final swarm. As both methods are population-based, with an elitist approach, (what means that best solutions found along all process are stored in a data structure), is possible to take advantage of this situation and to create an ensemble as a final model. Using an ensemble as final model, it is possible to reduce the mean error rate over that of the individual classifier and often, the ensemble outperforms even the strongest individual member of the ensemble [79]. In our case, having a set with the best classifiers and a fitness measure to weight their influence in an ensemble, the best option is the use of a weighted voting ensemble. If the individual classifiers are with unequal performance, intuitively, it is reasonable to give more power to the stronger classifiers in voting. The output class label of the ensemble is:

$$H(x) = C_{arg_j max} \Sigma_{i=1}^{T} w_i h_i^j(x) \tag{5.5}$$

Where w_i is the weight assigned to the classifier h_i, x is a set of instances, $C = \{c_1, c_2, \ldots, c_l\}$ is a set of l possible class labels, and $H(x)$ is the final set of labels [167].

5.3 Experiments and results

In this section, the model selection experiments conducted to test the performance of PSMS and the GA in the datasets shown in Chapter B are presented. To be capable to conduct statistical tests, replications of each dataset were created. Each replication was created with a random sort of the instances and different among replications. Following the recommendations proposed to avoid optimistic bias in the performance evaluation, the Internal protocol for model selection experiments described in Cawley and Talbot [2010] was employed. In the internal protocol, the model selection step is performed separately in each replication of each dataset. In each experimental run, following the conventions in the literature, the dataset (the replication) is divided in two disjoint sets, one partition to perform the model selection process (60% of the samples for training set) and the other partition (40% for test set) is employed to test the performance of the models constructed in the previous partition. To obtain a statistical power of 90% in a Student's t-test, 44 replications were performed. The number of replications may vary according to the test performed and the number of methods to be compared. The guidelines presented above are the ones that were followed in all the experiments presented in this work.

Configuration parameters of each algorithm are as follows. For the GA different population sizes (5,10,20,30,40,50 individuals) and mutation rates (0.05,0.07,0.1) were evaluated experimentally in the 10% of the RLCP dataset, concerning the crossover rate, this was fixed and determined by the Vasconcelos Model (also used for selection and replacement) where all offspring is made by crossover. The best combination of factors was a population of 30 individuals and a Mutation rate of 0.1%. Regarding PSMS, the original configuration parameters of the original paper were preserved in the same way that the adaptive inertia weight mechanism show in section 5.2.2. The swarm size was changed from 5 to 30 to compete in similar conditions to the GA. The stopping criterion was set for both methods to perform 47 iterations (in order to be able to make fair comparisons with the Grid search), what means 1,412 model evaluations. In Table 5.3.1 the average misclassification rates in the test partition over 44 replications obtained by GA and PSMS is shown.

Table 5.3.1, shows that PSMS obtained the lowest error in four (Higgs, Synthetic 2, Epsilon and Non-convex 2) of the nine evaluated datasets, in the datasets Non-convex 1 and Non-convex 3, PSMS obtained the same results as the GA. In datasets RLCP, KDD and Synthetic 1, PSMS obtained a biger error than the GA. As mentioned above, to obtain evidence of significant differences, a Student's t-test was performed and the results are shown in Table 5.3.2.

Table 5.3.1: Average misclassification rate obtained in the test partition by GA and PSMS over 44 replications. The lowest error rates are in **bold**.

Dataset	GA	PSMS
RLCP	**0.009±0.001**	0.052±0.001
KDD	**0.025±0.007**	0.166±0.148
Synthetic 1	**15.862±0.004**	15.864±0.004
Higgs	29.507±0.143	**28.304±0.064**
Synthetic 2	6.684±0.002	**6.683±0.005**
Epsilon	54.300±1.391	**53.928±0.995**
Non-convex 1	**0.000±0.000**	**0.000±0.000**
Non-convex 2	51.881 ± 1.462	**51.675±2.181**
Non-convex 3	**0.000±0.000**	**0.000±0.000**
Average	17.585±22.427	**17.408±22.216**

Table 5.3.2: Results obtained from the Student's t-test for the comparison of the performance of GA and PSMS in the FMS analysis in high volume datasets. Cases whose p-vale is below $\boldsymbol{\alpha = 0.05}$ are in **bold**.

Dataset	p-value
RLCP	$\mathbf{2.027 \times 10^{-64}}$
KDD	$\mathbf{1.445 \times 10^{-07}}$
Synthetic 1	0.054
Higgs	$\mathbf{1.639 \times 10^{-40}}$
Synthetic 2	0.324
Epsilon	0.275
Non-convex 1	-
Non-convex 2	0.622
Non-convex 3	-

As can be seen in Table 5.3.2, there were significant differences in three datasets: RLCP, KDD and Higgs. It's important to note that in datasets Non-convex 1 and Non-convex 3 a comparative cannot be performed because both algorithms obtained similar results. In datasets RLCP and KDD the differences in the performance of the search techniques favors GA over PSMS.

Regarding Higgs dataset, PSMS is the one with the best performance. To this point, it may be thought that both search techniques are similar in performance but

considering the analysis of datasets shown in Annex-B about the intrinsic dimension, Higgs is a harder problem than RLCP and KDD with an intrinsic dimension of 12/15 against 2 for RLCP and 1 for KDD. Taking this into account, the performance of PSMS is superior compared to the GA. In order to compare the performance of both techniques, two well-known classification/model-selection techniques were employed. The Kernel Nearest-Neighbour algorithm (K-NN) present in Yu et al. [2002] and a grid search based on MapReduce available in the libraries of Apache Spark (Grid) for tuning machine learning algorithms [7]. Concerning the Grid search the number of models to be evaluated was established to 1,412 (in order to be similar to the number of models evaluated by PSMS and the GA) using the algorithms mentioned in Table 5.2.1 and for K-NN we set k=9,999 to allow to K-NN exploit the high number of samples to improve its performance.

The average misclassification rate obtained by all the algorithms mentioned above are shown in Table 5.3.3. In order to obtain a statistical power of 90% in an ANOVA test and a Tukey HSD test (pos hoc) that compared four methods, 20 replications were performed.

Table 5.3.3: Average misclassification rate obtained in the test partition by GA, PSMS the Grid search and K-NN over 20 replications. The lowest error rates are in **bold**.

Dataset	GA	PSMS	Grid	K-NN
RLCP	0.009±0.001	0.052±0.001	**0.001±0.000**	0.500±0.098
KDD	0.025±0.007	0.156±0.134	**0.001±0.000**	19.535±0.261
Synthetic 1	**15.862±0.004**	**15.862±0.004**	17.011±0.120	50.088±0.028
Higgs	29.507±0.143	**28.299±0.057**	30.955±0.228	46.916±0.408
Synthetic 2	6.684±0.002	**6.681±0.005**	22.152±0.541	50.126±0.115
Epsilon	54.334±1.396	54.008±0.925	**29.664±0.174**	50.673±3.323
Non-convex 1	**0.000±0.000**	**0.000±0.000**	**0.000±0.000**	**0.000±0.000**
Non-convex 2	52.007±1.642	51.573±1.955	50.783±1.723	**49.041±2.110**
Non-convex 3	**0.000±0.000**	**0.000±0.000**	**0.000±0.000**	**0.000±0.000**
Average	17.585±22.427	17.408±22.216	**16.729±18.297**	29.653±24.143

Table 5.3.3 shows that the Grid search obtained the best performance in three datasets: RLCP, KDD and Epsilon. All methods obtained the same results in datasets Non-convex 1 and Non-convex 3, while in datasets Synthetic 1, Higgs, Synthetic 2 and Non-convex 2, the best results were obtained by PSMS, GA and K-NN.

Regarding all Non-convex datasets, they are useful for several reasons. The first one is to obtain evidence that the proposed FMS methods are capable to analyze datasets of different shapes. The second reason is that datasets: Non-convex 1 and

Non-convex 3 represent a relatively trivial classification task and, if the evaluated algorithms cannot obtain an error rate near to zero or zero, those algorithms must be discarded. On the contrary, Non-convex 2 is an impossible classification job, therefore, no classifier should surpass an accuracy of 50%, which would mean that the classifier was overfitted.

From Table 5.3.3, it can be seen that, the second-best method in the comparative was PSMS with the lowest misclassification rates in two datasets: Higgs and Synthetic 2. The GA and PSMS obtained the same performance in Synthetic 1, Non-convex 1 and Non-convex 3. To find statistical evidence of significant differences, both methods were compared against the Grid search and to K-NN. The Dunnett pos hoc test was employed to compare each treatment with a single control (the GA or PSMS). 20 replications were performed to obtain a statistical power of 90% in an ANOVA test. The results of these comparisons are shown in Tables 5.3.4 (PSMS vs Grid and K-NN) and Table 5.3.5 (GA vs Grid and K-NN).

Table 5.3.4: Results obtained from the ANOVA test for the comparison of the performance of GA vs the Grid search and K-NN. As a pos hoc analysis, the Dunnett test was performed. The p-values for each comparison and for the ANOVA test are shown. Cases whose p-vale is below $\boldsymbol{\alpha = 0.05}$ are in **bold**.

Dataset	ANOVA p-value	PSMS vs Grid	PSMS vs K-NN
RLCP	$\mathbf{3.724 \times 10^{-36}}$	**0.012**	$\mathbf{2.647 \times 10^{-06}}$
KDD	$\mathbf{3.967 \times 10^{-100}}$	**0.010**	$\mathbf{2.647 \times 10^{-06}}$
Synthetic 1	$\mathbf{3.898 \times 10^{-135}}$	$\mathbf{2.647 \times 10^{-06}}$	$\mathbf{2.647 \times 10^{-06}}$
Higgs	$\mathbf{8.479 \times 10^{-86}}$	$\mathbf{2.647 \times 10^{-06}}$	$\mathbf{2.647 \times 10^{-06}}$
Synthetic 2	$\mathbf{3.674 \times 10^{-101}}$	$\mathbf{2.647 \times 10^{-06}}$	$\mathbf{2.647 \times 10^{-06}}$
Epsilon	$\mathbf{1.625 \times 10^{-43}}$	$\mathbf{2.647 \times 10^{-06}}$	$\mathbf{6.709 \times 10^{-06}7.476*}$
Non-convex 1	-	-	-
Non-convex 2	$\mathbf{4.139 \times 10^{-04}}$	0.335	**0.0002**
Non-convex 3	-	-	-

Table 5.3.4 shows that, there were significant differences between PSMS and the other methods. This test in conjunction with Table 5.3.3, provide evidence that the performance of PSMS is the best in datasets Synthetic 1, Higgs, and Synthetic 2 and there were no significant differences when is compared against the Grid search in the dataset Non-convex 2. Regarding datasets RLCP, KDD and Epsilon, the Grid search obtained the best performance on those datasets and, although from the perspective of the ID analysis, the datasets RLCP and KDD are the easiest problems in the experiments, the Epsilon dataset is the hardest one. Regarding K-NN, this algorithm was surpassed by PSMS in datasets RLCP KDD, Synthetic 1, Higgs, Synthetic 2 and

Epsilon. Concerning to Non-convex datasets, no comparisons can be made, because the performance of both methods was the same in Non-convex 1 and 3, and the best performance in Non-convex 2 was obtained by the K-NN algorithm.

Table 5.3.5: Results obtained from the ANOVA test for the comparison of the performance of GA vs the Grid search and K-NN. As a pos hoc analysis, the Dunnett test was performed. The p-values for each comparison and for the ANOVA test are shown. Cases whose p-vale is below $\boldsymbol{\alpha = 0.05}$ are in **bold**.

Dataset	ANOVA p-value	GA vs Grid	GA vs K-NN
RLCP	$\mathbf{4.316 \times 10^{-37}}$	0.867	$\mathbf{2.647 \times 10^{-06}}$
KDD	$\mathbf{4.137 \times 10^{-103}}$	0.841	$\mathbf{2.647 \times 10^{-06}}$
Synthetic 1	$\mathbf{3.880 \times 10^{-135}}$	$\mathbf{2.647 \times 10^{-06}}$	$\mathbf{2.647 \times 10^{-06}}$
Higgs	$\mathbf{4.553 \times 10^{-84}}$	$\mathbf{2.647 \times 10^{-06}}$	$\mathbf{2.647 \times 10^{-06}}$
Synthetic 2	$\mathbf{3.677 \times 10^{-101}}$	$\mathbf{2.647 \times 10^{-06}}$	$\mathbf{2.647 \times 10^{-06}}$
Epsilon	$\mathbf{1.128 \times 10^{-42}}$	$\mathbf{2.647 \times 10^{-06}}$	$\mathbf{4.161 \times 10^{-06}}$
Non-convex 1	-	-	-
Non-convex 2	$\mathbf{1.986 \times 10^{-05}}$	0.072	$\mathbf{1.049 \times 10^{-05}}$
Non-convex 3	-	-	-

Regarding the analysis of the performance of the GA, in Table 5.3.5 it can be seen that there were significant differences in the datasets Synthetic 1, Higgs and Synthetic 2, the datasets where the GA obtained the best performance. However, despite in datasets RLCP, KDD and Non-convex 2, there were no significant differences compared to the best method in this comparative (the Grid search), the best performance with significant differences in the Epsilon dataset was obtained by the Grid search. The K-NN algorithm was surpassed by the GA in almost all datasets except in the Non-convex 2, where it obtained the best performance.

5.3.1 Discussion

The experiments conducted, showed that both search algorithms were capable to obtain models of good quality and the size of the datasets was not a problem, performing the search process in a tractable time. From the characteristics of the proposed datasets, the algorithms compared were capable to deal with a wide range of dataset sizes and with different shapes and, considering the performance obtained in the dataset Non-convex 2, both algorithms perform a search without overfitting the models.

Despite the fact that their performance was very similar, the PSMS algorithm was capable to obtain a performance with significant differences in one of the hardest

datasets, the Higgs dataset, whose ID are of 12/15, while the GA obtained the best results in the easiest datasets RLCP with an ID of 2 and KDD with an ID of 1.

Both, the GA and PSMS where compared to two techniques for model selection and classification techniques, also based in **MapReduce**. From the comparison, GA and PSMS were capable to obtain the best performance in three datasets: Synthetic 1, Higgs and Synthetic 2 but, they were outperformed by the Grid search in the hardest problem, the Epsilon dataset. This situation suggests that the huge search space imposed by the FMS problem requires granting more time to the search process or to guide the search more efficiently. Regarding the K-NN algorithm, both proposed algorithms were capable to outperform it with significant differences in almost all datasets.

From the evidence obtained, and due its capacity to solve a harder problem (with significant differences) than the ones where the GA obtained the best performance, it can be argued that PSMS is the best of both methods and is better suited to solve the FMS problem in high volume datasets than the GA. However, also from the evidence, it is clear that PSMS must be refined to perform better searches in a larger search space in order to obtain better results than the ones obtained by a simpler search technique as the Grid search.

5.4 Final considerations

In this Chapter, the use of bio-inspired methods to perform the FMS analysis in high volume datasets was investigated. The analysis of the relevant work in the literature showed that the FMS problem has been addressed only for smaller datasets and, it has been faced only through bio-inspired methods due its vast search space. Also, from the analysis of the related works, just the hyper-parameter optimization problem in high volume datasets has been investigated but only through grid-based searches. The strengths of the analyzed work were taken into account in the algorithms proposed in this Chapter and are as follows:

- The vast search space imposed by the FMS problem cannot be handled by gradient-based methods and grid-based searches. The use of bio-inspired methods has been recommended to these types of problems, that is why the two most popular methods (Genetic Algorithms and Swarm based algorithms) were employed.

- To be capable to analyze high volume datasets without reducing the dataset in number of features or instances, the **MapReduce** programming model has been used. The algorithms in this chapter were developed under this paradigm.

- In the population-based searches as the explored in this chapter, the selection of a single particle or individual as final model is not easy. In the literature, the use of ensembles of the best models found during the search has been proposed. The algorithms of this Chapter employed weighted voting ensembles as final model.

This first set of experiments was related to the first objective proposed in this work. This objective pursued to know which algorithm between a GA and PSMS was the most adequate for FMS in high volume datasets. Although, similar comparisons were made in the past, the evaluated algorithms didn't use the same algorithms for all the stages involved in FMS, that is why the information obtained in this comparison represents a contribution to this field. Furthermore, the developed framework to adapt population-based optimization algorithms to **MapReduce** contributes equipping the FMS paradigm with the ability to deal with high volume datasets.

The experiments provided evidence that the Swarm-based search, PSMS outperformed the GA in one of the hardest problems analyzed and therefore, it can be argued that PSMS is the best of both algorithms. Despite this fact, both algorithms were surpassed by a simpler Grid search. This situation suggests that, the bigger search space in the FMS problem needs a way to guide the search process more efficiently in order to obtain models of better quality. The use of proxy models has been suggested as a way to reduce the time of a search process, but it could be of great interest, to explore their use as a compass to guide the search in a more efficiently way and, to improve the quality of our models.

Chapter 6

Use of proxy models as a way to guide the search of the FMS problem

Traditionally, proxy models have been used as a tool to speed up time-consuming optimization processes as: calibration of hydrological models [82], petroleum reservoir modeling [163], natural gas fields reservoir simulations [3], and wind turbines design [157], among others. However, proxy models also can be used to guide efficiently search process as in FMS. This efficiency is given in terms of models of higher accuracy found in shorter search processes.

In this Chapter, the second and third objectives of this work are addressed. The central importance of proxy models in this set of experiments, lead the authors to look for the best way to create accurate proxy models. From the insights of the related work and from the authors point of view, FMS paradigm is the best alternative for creation of high-quality proxy models. To the best of the authors knowledge, this is the first work that considers the use of this paradigm in the construction of proxy models.

Different strategies were considered in the construction of proxy models, and another groundbreaking contribution of this work, is the use of classification algorithms to build them. In order to provide richer information about proxy models and their use to guide the search in the FMS problem, all approaches that take advantage of this paradigm are addressed in this Chapter.

The organization of this Chapter is as follows, in Section 6.1 a revision of the related work is provided, Section 6.2 describes the proposed methods to build proxy models under the FMS paradigm and their integration with the search of models in high volume datasets. The experimental results and the discussion are in Section 6.3 and in Section 6.4 are the final considerations.

6.1 Related work

A wide range of problems where the search space is huge, the model to evaluate a possible solution is "proprietary technology" (the model is unknown), and the time that a single evaluation takes is prohibitive have been addressed through the use of proxy models. As explained in Section 2.3, a proxy model is an inexpensive alternative to a full numerical simulation. Proxy models range from data-fit models, statistical models and reduced order models [3] but in most works this alternative comes from a regression model trained in a set of solutions evaluated with the real model (meta-dataset). In order to get a better understanding of the approach proposed in this work to build proxy models and its difference with other approaches in the literature, Fig. 6.1.1 is provided.

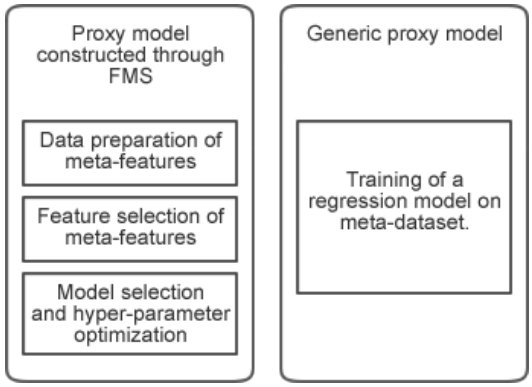


Figure 6.1.1: Proxy models built under FMS (left) and generic proxy models (right).

A significant amount of work relies on a single regression algorithm previously selected to be employed as proxy model [131, 119, 155, 97, 81, 111, 115, 157, 116]. The most commonly used algorithms as proxy models are: Neural networks [131, 111, 119], Gaussian process[1] [155, 97, 81, 116] and reduced mathematical models [115].

In the past, the use of proxy models to solve the FMS problem has been explored [131], however, in the same way that in several works, their use is limited to the reduction of the time the optimization process takes. A proxy model is a useful tool that also can be employed to guide the search process. From this perspective, a proxy model can be thought as a compass to explore the search space, therefore a good idea is to find a way to build the best compass possible.

[1]Gaussian process is a probabilistic, non-parametric model with uncertainty predictions. It can be used for the modelling of complex, non-linear systems. The output of the GP is a normal distribution expressed in terms of mean and variance [16]

From the related work, three strategies have been used to improve the performance of proxy models. The first one is **the use of model selection techniques** (not to be confused with the FMS paradigm) in the construction of proxy models [157, 31, 32, 68, 122, 124]. As the "No free lunch" theorem indicates, there is no best model for all problems, this principle applies for proxy models too. In Wilson et al. [2017] and [122] this idea is explored using different algorithms for the proxy model construction (Regression Random Forest, Logistic Regression, Neural networks, among others) but without hyper-parameter optimization. The remaining work under this category consider a pool of algorithms and, the hyper-parameter optimization step. For hyper-parameter optimization and model selection, it has been employed genetic algorithms [31, 68] and gradient-based methods [32]. The ensemble of models [124] also have been considered.

The second strategy considers **instance selection in the construction of proxy models** [3, 66, 147, 97]. Selection of instances in the meta-dataset (instances evaluated with the true model) can be used as a way to reduce the time employed for the construction of the proxy model [97] or to improve the quality of the proxy model through the selection of the instances that leads to a better replication of the true model [147, 66]. Moreover, the sampling of data points has been employed to keep the dataset heterogeneity for a better training [3].

Finally, a third strategy considers use of **fuzzy sets theory in the construction of proxy models** [34, 35, 45, 82, 6, 145, 117, 65]. This paradigm was employed because, there is big interest in capturing expert knowledge [6] and simulating human behavior [145]. Hyper-parameter optimization and model selection is also performed in works under this category to improve their performance [117, 65], however, a big part of these works relies just on neuro-fuzzy networks as proxy models [34, 35, 45, 82] to exploit the use of granule computing to group similar possible solutions instead of performing individual evaluations.

From above, it can be seen that, proxy models are applied to a wide range of problems, using different types of algorithms and employing different methods to improve their performance. However, there is another important division in the proxy model literature: those works where main efforts are focused in the construction of the proxy model and works where the proxy model is a tool build during the optimization process and then discarded. In FMS, each dataset represents (ideally) a different computational problem and for this reason, there is no prior information to build a proxy model. Even if the information is available, the proxy model constructed from a meta-dataset with the classifiers and their performance on a linear dataset could have a low accuracy to predict the performance of classifiers in a non-linear dataset, therefore FMS falls within the second category.

Different strategies analyzed show great interest in the improvement of accuracy in proxy models through instance selection, model selection and, hyper-parameter optimization techniques. In FMS problem and similar, proxy model provides a mechanism to decide if a potential solution is promising enough to be evaluated with a true model or if it should be discarded. Along progression of search of models, meta-dataset changes continuously and, in the same way, importance of the meta-features and the suitability of learning algorithms. A way to build highly adaptable proxy models is through the use of the FMS paradigm and, due in past this paradigm has proved its usefulness in classification tasks, next logical step is to take FMS paradigm to proxy models field.

Upon a foundation of good models, the evolutive mechanisms of bio-inspired searches lead the process to better regions and, therefore, to obtain better models. Thinking outside the box, discrimination process between promising and not promising solutions can be addressed also as a classification task. In this Chapter, three approaches were explored in order **to gain information** about the use of the FMS paradigm **to build better proxy models, and using them to guide efficiently through the vast search space of the FMS problem.** Those approaches are described below:

- Proxy models construction employing FMS paradigm. That is, performing data-preparation, feature selection, selection of a learning algorithm and hyper-parameter optimization on the meta-dataset. The success of FMS paradigm to build highly accurate models, was the starting point to employ this paradigm to build better proxy models that can guide the search process and finding better models in a shorter search process. To the authors best knowledge, this is the first work that proposes the use of FMS to build proxy models.

- Using classification algorithms in conjunction with FMS paradigm. The approach mentioned above performs model selection step among regression algorithms. In this approach instead of predicting the expected fitness of a model, a discrimination between promising and not promising models is provided. Taking advantage of the success of FMS to build highly accurate classification models, the next logical step was to use those classifiers to assist in FMS process. With a better discrimination between potential and not potential solutions, the time of the optimization process is invested in the search of finer models. At the time of writing this document, this approach was considered only in this work.

- Using a fuzzy classification algorithm build under FMS paradigm. Fuzzy classification algorithms provide a membership degree of an object to considered classes.

This membership degree can be employed to perform a finer discrimination. Investing the time of the process in potential solutions enables to bio-inspired mechanisms to use them as a cornerstone to obtain better models in a fraction of the time of the unassisted process. This approach was also considered first in this work.

6.2 Highly adaptive proxy models to guide the search in the FMS problem

In FMS problem and similar, proxy models cannot replace true model entirely, and both models must be used together. Since fitness of each potential solution in a model selection process is just an estimation of true performance of the model, proxy model has no way to replicate accurately the true model, that is why, the proxy model cannot replace time-consuming fitness-function. In order to understand the use of proxy models in a model selection process, Fig. 6.2.1 shows a flowchart of this process.

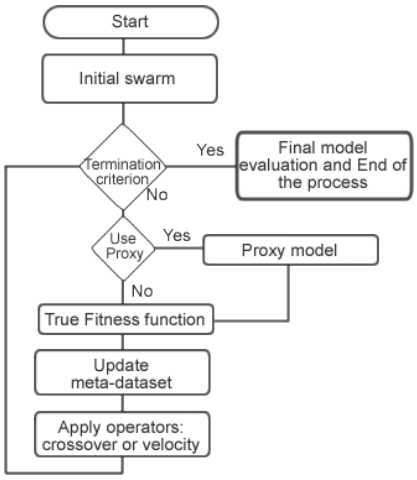


Figure 6.2.1: Use of proxy models in the full model selection problem.

Although the use of proxy models can be extended to any search algorithm, to avoid confusions, this process will be described in the context of a PSO algorithm.

1. During a given number of iterations, all the particles in the swarm are evaluated with the true fitness function to collect data for the meta-dataset. The meta-

features in this meta-dataset are each one of the elements in the particles and as target, their fitness is employed (a regression task).

2. A learning algorithm is trained in this meta-dataset and from this point it becomes the Proxy model. The particles are evaluated with the proxy model and according their expected fitness, the promising ones are evaluated with the true fitness function, the rest are discarded.

3. The meta-dataset is updated with the particles evaluated with the true fitness function.

4. Return to step 2 until termination criterion is met.

6.2.1 Considerations of the proxy models developed through FMS

In the previous Chapter, experimental results provided evidence of the superiority of PSMS in the FMS task compared to the GA, that is why, this algorithm was employed as cornerstone of all subsequent experiments. In this case, PSMS was adapted to perform the construction of proxy models and to perform FMS analysis in high volume datasets. This situation can be quite confusing, then, it is necessary to use different acronyms to distinguish these elements. PSMS algorithm employed for model selection in high volume datasets was referred along this Chapter as Main PSMS (M-PSMS). For the search assisted by proxy models that uses regression algorithms, (S-PSMSReg) was employed and regarding PSMS assisted by proxy models constructed with classification algorithms, the acronym (S-PSMSClass) was used.

For S-PSMSReg, and S-PSMSClass the learning algorithms shown in Table 5.2.1, in their regression and classification versions were employed. As proxy models produced by S-PSMSReg are for regression tasks, traditional feature selection algorithms cannot be employed. In a regression task there are no discretized classes, which is a requirement for feature selection algorithms in Table 5.2.1. Such task was addressed through the Principal Component Analysis.

In the case of S-PSMSClass is necessary to transform the learning process on the meta-dataset from a regression problem to a classification problem. Although this transformation is to a certain extent an easy job, in a model selection process is not that simple. Taking the scenario where there is no prior information about a dataset and information about the best performance that a learning algorithm can achieve, it is not easy to decide when a particle is promising and when not. To overcome

this situation, the fuzzy sets theory was used, and three fuzzy sets[2] were created (shown in Fig. 6.2.2): Low, Medium and High. In previous Chapter was explained that the Balanced Error Rate or BER was employed for the fitness evaluation of the particles, then a small BER means a model with high accuracy, therefore the particles whose BER has the highest membership degree to the "Low" fuzzy set, are marked as promising and the rest are discarded. The class unbalance was another challenge faced by this approach. At the beginning of the search the number of promising solutions is lower than the not-promising ones. To overcome this problem, the well-known strategies of over-sampling and under-sampling were employed in order to get an equal proportion of both classes.

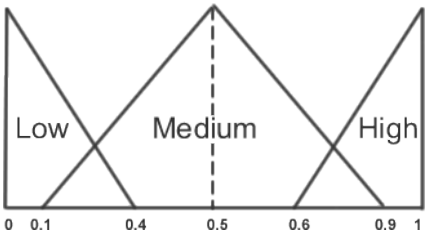


Figure 6.2.2: Fuzzy sets employed in the S-PSMSClass approach.

6.2.2 A proxy model based on fuzzy classification rules and constructed under the FMS paradigm

The use of a fuzzy classification algorithm at this point does not look like as a logical choice for a proxy model. However, in the experiments section of this Chapter, such connection will be stablished. As explained above, from a fuzzy classifier we can obtain the class of an object and its membership degree to that class. This membership degree can be used as a lower bound to discriminate more effectively between classes. Whit this in mind, the idea of a fuzzy-rules classifier provided in Ishibuchi and No & [1994] was employed, but to exploit the benefits of granular computing to find similarities among object, the classifier was constructed using fuzzy granules with the well-known clustering algorithm Fuzzy C-Means (FCM) [14]. Let $X = \{x_1, x_2, ..., x_n\} \subset R^S$ be an unlabeled dataset with n samples. The objective

[2]The Eq. 4.2 is employed to calculate the membership degree of an element to a triangular fuzzy set.

function of FCM was defined as:

$$J_m(U,V) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^m d_{ij}^2 \tag{6.1}$$

From eq. 6.1, c is the number of clusters, n is the number of instances or data points, the exponent m is a fuzzifier for controlling the fuzzy degree of the clustering result, $V = [v_1, v_2, ..., v_n]$ is a vector of cluster centers, d_{ij} is the distance of object x_j to v_i, $U = (u_{ij})$ is a fuzzy partition matrix composed of the membership degrees of each object x_j with respect to ith cluster, where u_{ij} satisfies $\sum_{i=1}^{c} u_{ij} = 1$, $0 \leq u_{ij} \leq 1$ and $\sum_{j=1}^{n} u_{ij} > 0\,(\forall i)$. The condition for minimizing $J_m(U,V)$ are as follows:

$$u_{ij} = \frac{(1/d_{ij}^2)^{1/(m-1)}}{\sum_{k=1}^{c}(1/d_{kj}^2)^{1/(m-1)}}, i = 1, ..., c, j = 1, ..., n. \tag{6.2}$$

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m}, i = 1, ..., c. \tag{6.3}$$

With the granules obtained from the FCM algorithm, we can turn them into fuzzy membership functions for the construction of a fuzzy-rules classifier. Each cluster is turned into one rule with a grade of certainty and a consequent class in the form of:

$$\text{Rule } R_i^c : If\ x_p\ is\ Cluster_i^c\ Then\ x_p\ belongs\ to\ C_t^M\ with\ CF = CF_i^c$$

Where R_i^c is the label of the rule, x_p is a data point or pattern, $Cluster_i^c$ is the i-th cluster, and in this case the membership function, C_t^M is the t-th class of M total classes and CF_i^c is the grade of certainty of the rule. Next step is the construction of these rules using the procedure in [78] but adapted to our membership functions. First the sum of the compatibility of x_p in Class T or β_{Ct} is calculated using the fuzzy partition matrix U obtained in the clustering stage as follows:

$$\beta_{Ct} = \sum_{p \in Ct} u_{ip},\ t = 1, 2, ..., M\ and\ i = 1, 2, ..., c \tag{6.4}$$

Subsequently, the Class X (CX) of R_i^c is determined as:

$$\beta_{Cx} = max\{\beta_{c1}, \beta_{c2}, ..., \beta_{cm}\}, t = 1, ..., M. \tag{6.5}$$

If two or more classes take the maximum value, consequent $C_t^M = \emptyset$ because, consequent class cannot be determined uniquely. Subsequently, the certainty degree of the rule is calculated as follows:

$$CF_i^c = \frac{|\beta_{Cx} - \beta|}{\sum_{t=1}^{M} \beta_{Ct}} \tag{6.6}$$

$$\beta = \sum_{t=1, Ct \notin Cx}^{M} \beta_{Ct}/(M-1) \tag{6.7}$$

From the foregoing, there are several factors that cannot be established in advance as: number of clusters c, fuzzifier parameter m, similarity measure, features employed, among others. The optimal values of these parameters can be obtained through the FMS paradigm and according to trends in the literature, instance selection was also considered. A PSO algorithm was employed. The particle vector $x_i^t = [x_{i,1}^t, x_{i,2}^t, ..., x_{i,nf}^t, x_{i,nf+1}^t, ..., x_{i,ni}^t]$ codification is as follows: Position 1 stores the fitness of particle (BER of 2-fold cross validation). Position 2 is for the Fuzzifier term m of the FCM algorithm. Position 3 is the identifier of similarity measure. The following similarity measures were considered: 1) Euclidean distance, 2) Manhattan distance, 3) Pearson coefficient, 4) Cosine measure and 5) Chebyshev distance. Position 4 is for the number of iterations of FCM. Position 5 is for number of clusters. Position 6 is the identifier of data preparation method. The methods employed were: 1) Feature standardization, 2) data normalization and 3) do nothing. Alternative methods were considered without differences in the performance of the algorithm. The feature selection process is performed as a wrapper approach (position 7 to nf or number of features). The same approach was employed in the instance selection process (position $nf + 1$ to ni or number of instances). Choosing a single rule set as surrogate model can be a difficult task, because the real performance could be underestimated or overestimated. For this reason, the final model is also a weighted-voting ensemble of the best rule sets found in the search. To the synergy of PSMS and this fuzzy-rules algorithm is referred along this work as Fuzzy Rules based proxy model for PSMS (FR-PSMS).

6.3 Experiments and results

The first searches assisted by FMS-based proxy models (S-PSMSReg and S-PSMSClass) were compared to PSMS (the unassisted search) and to a surrogate assisted search, that is PSMS assisted by a Multilayer Perceptron (PSMS-MLP). In the PSMS-MLP

version, hyper-parameters of the MLP were adjusted through a grid search at the beginning of FMS analysis and during the rest of the process, the MLP was just re-trained with the updated meta-dataset. Regarding FR-PSMS, this approach was proposed at a later stage of this research due to the success of the Classification-based proxy models. Performance of the methods mentioned above and the remarkable performance of S-PSMSClass will be shown in the next sections.

The termination criterion of PSMS was changed from 47 iterations (used in the previous Chapter) to 50. As swarm size was of 30 particles, the number of particles evaluated was of 1,500. Although this search process explores an important quantity of potential models, not all evaluated particles are employed to build the final model, therefore, evaluation of models considered as "bad" is just a waste of time. In order to know the average number of "good" models evaluated by PSMS and to provide evidence that proposed approaches perform a more efficient[3] search, fitness of all the evaluated particles (in some datasets) was stored and, following the procedure to transform a regression task into a classification one (show in Section 6.2.1), this quantity was determined. To know about how many potential models are found in a conventional FMS process is essential to determine the value of termination criterion in all proxy assisted searches. This value must be chosen carefully in order to perform a smaller search than the one performed by the unassisted search, measured in the number of evaluations performed with the time-consuming fitness function, and big enough to obtain models of similar or even superior quality. In Table 6.3.1 these results are shown.

Table 6.3.1: Average amount of "good" models found by PSMS exploring 1,500 particles over 20 replications.

Datasets	Number of promising models
RLCP	830.70 ± 38.16
KDD	709.60 ± 192.57
Synthetic 1	404.85 ± 116.57
Higgs	508.50 ± 53.07
Synthetic 2	290.15 ± 63.53
Epsilon	310.05 ± 10.34

The average amount of promising models in the datasets is 508, for this reason the

[3]The Merriam-Webster dictionary defines the efficiency as: effective operation as measured by a comparison of production with cost (as in energy, time, and money).

total number of models evaluated will be of 500. In Table 6.3.2, the mean classification error obtained by the different algorithms in this chapter is shown.

Table 6.3.2: Average misclassification rate obtained in the test partition by PSMS, PSMS-MLP, S-PSMSReg and S-PSMSClass over 43 replications. The lowest error rates are in **bold**.

Dataset	PSMS	PSMS-MLP	S-PSMSReg	S-PSMSClass
RLCP	0.052 ± 0.001	0.426 ± 1.593	0.059 ± 0.123	**0.028±0.067**
KDD	0.165±0.150	5.071±7.537	**0.078±0.002**	0.159±0.136
Synthetic 1	**15.864±0.005**	**15.861±0.004**	**15.864±0.006**	**15.865±0.006**
Higgs	**28.307±0.062**	29.651±1.781	30.156±0.635	28.325±0.171
Synthetic 2	6.683±0.006	6.682±0.004	**6.681±0.003**	**6.681±0.004**
Epsilon	54.046±1.029	33.121±5.085	28.813±2.564	**28.418±1.107**
Non-convex 1	**0.000±0.000**	4.279±3.852	0.365±0.529	**0.000±0.000**
Non-convex 2	51.626±2.182	**48.803±2.309**	50.903±1.561	51.364±2.441
Non-convex 3	**0.000±0.000**	1.117±1.858	0.392±0.588	**0.000±0.000**
Average	17.467±22.186	16.112±17.186	14.812±18.209	**14.537±18.174**

Table 6.3.2 shows that S-PSMSClass obtained the best performance in six datasets: RLCP, Synthetic 1, Synthetic 2, Epsilon and Non-convex 3. The second-best algorithm in the comparative was PSMS with the best performance in four datasets: Synthetic 1, Higgs, Non-convex 1 and Non-convex 3. Regarding S-PSMSReg, it obtained the best performance just in three datasets (KDD, Synthetic 1 and Synthetic 2), while PSMS-MLP was the best just in two (Synthetic 1 and Non-convex 2).

It is important to remember that all surrogate-based approaches evaluated only a third part of the particles assessed by PSMS and despite that, they were capable to obtain a performance near to the obtained by PSMS and even, this performance was surpassed in the Epsilon dataset. As was mentioned in previous sections of this Chapter, the proxy models were employed with the main purpose of to guide the search process, therefore is interesting to known how this was done. The following figures are employed to provide some insight of how this process was performed in two datasets: Higgs (Fig. 6.3.1) and Synthetic 1 (Fig. 6.3.2). Those figures show the estimated BER of the best particle in the Y axis and in X axis the number of evaluated models is used to show the progression of the search.

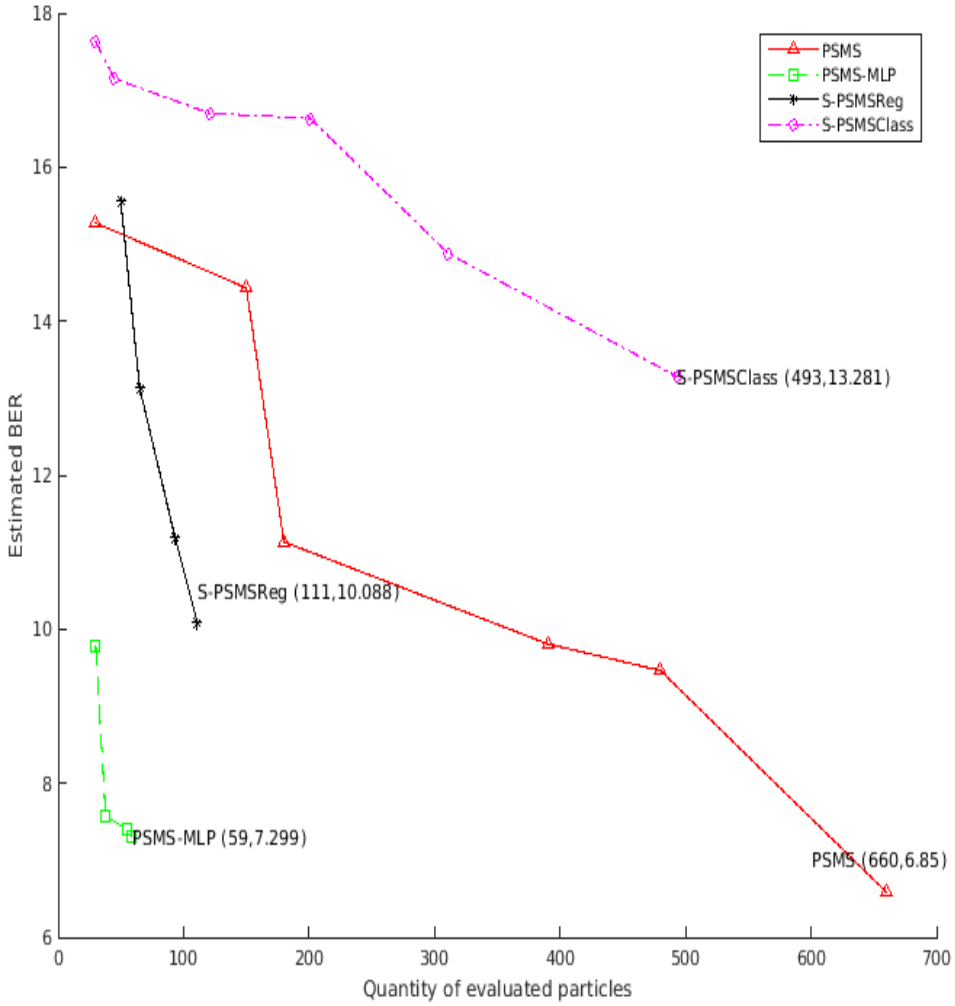


Figure 6.3.1: Search process performed by PSMS, PSMS-MLP, S-PSMSReg and S-PSMSClass in dataset Synthetic 1. In X axis the quantity of evaluated particles is shown; the Y axis shows the estimated BER of the best particle in the swarm.

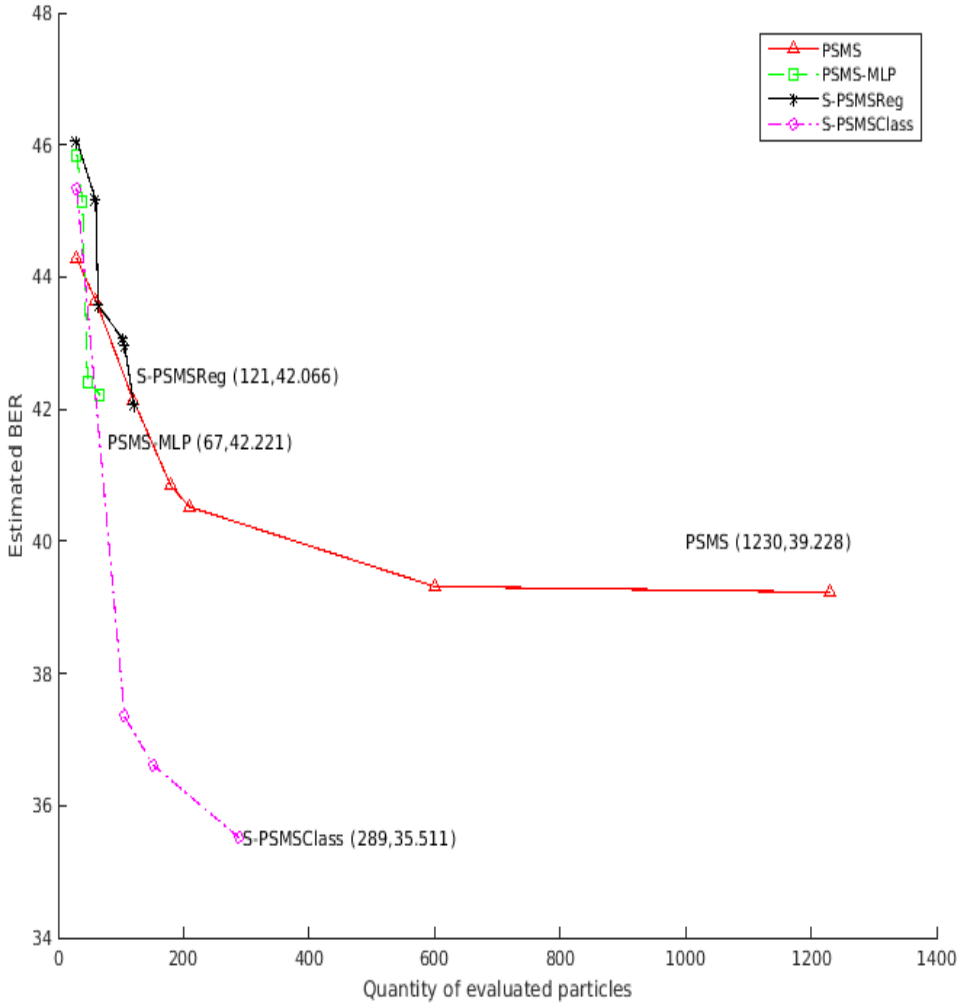


Figure 6.3.2: Search process performed by PSMS, PSMS-MLP, S-PSMSReg and S-PSMSClass in dataset Higgs. In X axis the quantity of evaluated particles is shown; the Y axis shows the estimated BER of the best particle in the swarm.

Fig. 6.3.1 shows that, PSMS-MLP was the fastest search-method in the comparison. The most important model of PSMS-MLP, the one with the highest weight in its final ensemble was found after 59 evaluated particles. The lower estimated BER found by PSMS-MLP was around of 7%, near to the lowest BER found by PSMS but after

680 evaluations. Regarding the FMS-based proxy model approaches, S-PSMSReg obtained an estimated BER of 10% after 110 evaluations, while S-PSMSClass obtained an estimated BER around of 13% after 490 evaluations. Since all methods obtained a similar performance in this dataset, it can be argued that PSMS-MLP is the most efficient method of the comparison, performing a faster process and obtaining similar and slightly better results than those obtained by PSMS.

Regarding Fig. 6.3.2, the PSMS-MLP algorithm was again the fastest method with an estimated BER of 42% after 67 evaluations. The second best was S-PSMSReg with 120 evaluations and a BER also of 42%. PSMS obtained its best particle after 1,200 evaluations and its BER was of 39%. S-PSMSClass obtained its best particle after 290 evaluations and with a BER below of 36%. From the results in Table 6.3.2 shows that the best performance was obtained by PSMS followed by S-PSMSClass, PSMS-MLP and S-PSMSReg. To find evidence of significant differences in the performance of the compared strategies, an ANOVA test and a Dunnett test was performed, in Table 6.3.3, the reported p-values of the mentioned tests with confidence level of 95% are shown.

Table 6.3.3: Results obtained from the ANOVA test for the comparison of the performance of PSMS, PSMS-MLP, S-PSMSReg and S-PSMSClass. As a pos hoc analysis, the Dunnett test was performed. The p-values for each comparison and for the ANOVA test are shown. Cases whose p-vale is below $\boldsymbol{\alpha = 0.05}$ are in **bold**.

Dataset	Anova F	PSMS vs PSMS-MLP	PSMS vs S-PSMSReg	PSMS vs S-PSMSClass
RLCP	**0.000011**	$\mathbf{8.705 \times 10^{-05}}$	0.997	0.998
KDD	$\mathbf{2.175 \times 10^{-10}}$	$\mathbf{2.717 \times 10^{-06}}$	0.999	1
Synthetic 1	0.073	0.055	0.994	0.511
Higgs	$\mathbf{2.518 \times 10^{-20}}$	$\mathbf{2.689 \times 10^{-06}}$	$\mathbf{2.687 \times 10^{-06}}$	0.999
Synthetic 2	0.075	0.495	0.053	0.087
Epsilon	$\mathbf{3.576 \times 10^{-96}}$	$\mathbf{2.687 \times 10^{-06}}$	$\mathbf{2.687 \times 10^{-06}}$	$\mathbf{2.687 \times 10^{-06}}$
Non-convex 1	N/A	N/A	N/A	N/A
Non-convex 2	$\mathbf{7.487 \times 10^{-09}}$	$\mathbf{2.709 \times 10^{-06}}$	0.279	0.896
Non-convex 3	N/A	N/A	N/A	N/A

The results reported in Table 6.3.3 shows that all surrogate-based methods surpassed the performance of PSMS with significant differences in the Epsilon dataset, however just S-PSMSClass obtained a performance without significant differences in almost all datasets, therefore S-PSMSClass with around one third of the evaluations, is as good as PSMS and was capable to achieve a better performance in the hardest dataset, Epsilon. Regarding PSMS-MLP and S-PSMSReg, even in those datasets

where they obtained the best performance, there were no significant differences (KDD, Synthetic1 and Synthetic 2 for S-PSMSReg and Synthetic 1 for PSMS-MLP). Regarding Non-convex 2 dataset, PSMS-MLP was overfitted in this hard classification task, while both regression-based searches (PSMS-MLP and S-PSMSReg) had problems with the non-convex datasets. From the definition of the concept "efficiency" it can be say that an efficient method for our problem, obtain similar or even better models, performing a lower quantity of evaluations or employing a smaller amount of time. The computing time of all methods was registered and are shown as a bar chart in Fig. 6.3.3. The performance of the methods in the Non-convex datasets is shown in Fig. 6.3.4 due to differences in time scales. The experiments were performed in a workstation using 12 threads with an Intel(R) Xeon(R) CPU E5-2695 at 2.40GHz and 30 GB in RAM.

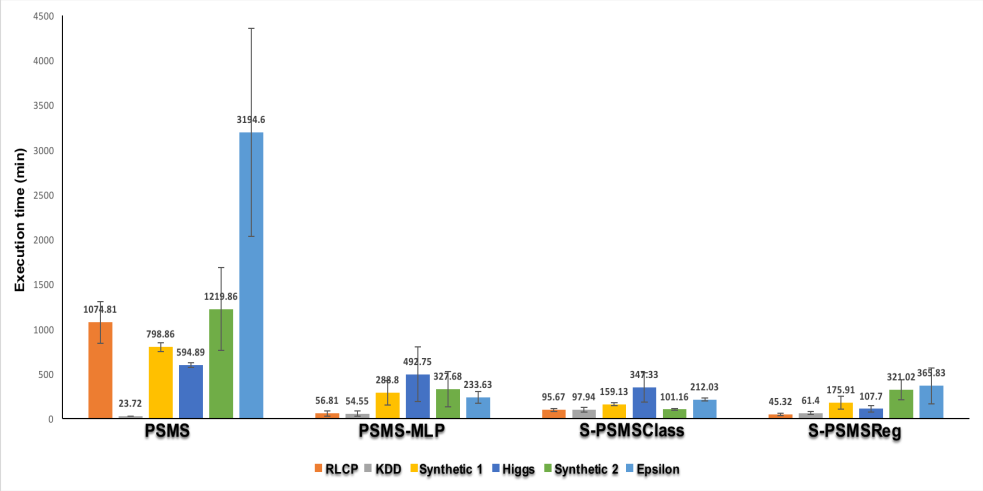


Figure 6.3.3: Bar chart with the average execution times (in minutes) obtained by PSMS, PSMS-MLP, S-PSMSReg and S-PSMSClass over 20 replications in the convex datasets. Each color represents a different dataset and the bars are grouped by algorithm. The standard deviation is depicted as a solid black line over the bars.

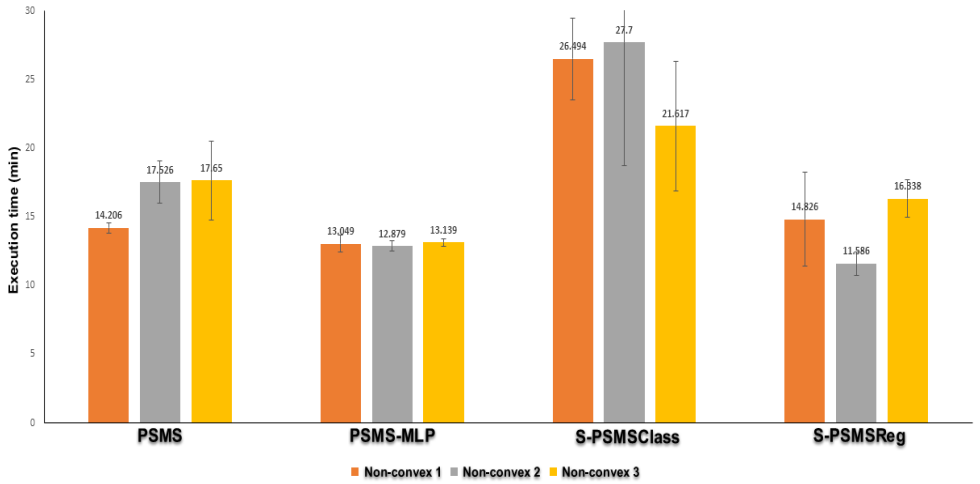


Figure 6.3.4: Bar chart with the average execution times (in minutes) obtained by PSMS, PSMS-MLP, S-PSMSReg and S-PSMSClass over 20 replications in the Non-convex datasets. Each color represents a different dataset and the bars are grouped by algorithm. The standard deviation is depicted as a solid black line over the bars.

Figure 6.3.3 shows that the highest execution times were obtained by PSMS in the convex datasets while, in the non-convex ones (Fig. 6.3.4) S-PSMSReg was the one that took more time. Is not a surprise that all surrogate-based searches obtained a lower execution time than the one obtained by PSMS in the convex datasets. Nonetheless, is uncommon that with a lower number of evaluations, PSMS-MLP got higher execution times than S-PSMSClass in the Higgs dataset. Fig. 6.3.2 shows that PSMS-MLP and S-PSMSReg obtained its best performance before S-PSMSClass, however, the longer execution times are due to another termination criterion used for the surrogate-based searches. All these searches were conditioned to perform an evaluation of 500 particles with the true model (training and model evaluation in the test set) or to perform 1000 iterations in the search algorithm, whatever occurs first. This measure was taken in order to avoid performing a very poor exploration and obtaining bad models. Without this measure, the performance of PSMS-MLP and S-PSMSReg may have been worse. Regarding to Non-convex datasets, the execution times of both regression-based proxy-assisted searches are not important, despite PSMS-MLP obtained the lowest time in this category, because both methods obtained the worst performance in all datasets. From both figures, it can be seen that S-PSMSClass obtained an average classification time compared to the other surrogate based methods and considerably lower compared to the execution times obtained by PSMS.

6.3.1 Discussion of the results obtained in the first proxy-based methods

From all the average misclassification rates, the statistical analyses, the way the search process is performed and the execution times, there is evidence to support that the best method to this point is S-PSMSClass. This algorithm does not fit in the strict definition of a proxy model, because this mechanism does not provide the expected fitness of a particle. The discrimination of the particles evaluated in the categories promising and not-promising is capable to produce a swarm composed by models with high potential. The devices in the bio-inspired searches are capable to evolve them (so to speak) to promising regions in the search space and obtaining better results faster. This method was capable to obtain results as good as the ones obtained by PSMS with a third of the evaluations (500 against 1,500) and in a fraction of the time. From the figures about the exploration process, it can be seen that all regression-based proxy-assisted search performs a more hurried search than the one performed by S-PSMSClass. This is because the regression-based proxy models become more meticulous and, at early stages in the search, no particle is evaluated as promising again.

This approach is an excellent way to address the FMS problem in high volume dataset, because it can find models of high quality, as good as the ones obtained by a longer search and even surpassing the results of PSMS in the hardest problem (Epsilon) according the ID analysis. The method exploited efficiently the time the search process takes due to the good guidance of the classification-based proxy models to explore the vast search space.

6.3.2 Comparison of FR-PSMS against S-PSMSClass

Due to the good results obtained by S-PSMSClass, it was compared against FR-PSMS. As mentioned earlier in this Chapter, FR-PSMS was a sub-product of the idea of using classification-based proxy models. Taking as starting point that the discrimination provided by this new device is good, the discrimination provided by a classifier that also give information about the membership of an object to the different classes could be better. This membership degree can be used as a lower bound in order to perform a stricter discrimination of the particles.

This new device was allowed to decide the number of evaluations needed until the final model construction and, as termination criterion, to complete 50 iterations. In Table 6.3.4 the average misclassification rates are shown and in the third column the average evaluations performed by FR-PSMS over 44 replications.

Table 6.3.4: Average misclassification rate obtained in the test partition by S-PSMSClass and FR-PSMS over 44 replications. PSMS is shown just for demonstrative purposes. The lowest error rates are in **bold**.

Dataset	S-PSMSClass	FR-PSMS	Mean number of function evaluations in FR-PSMS	PSMS
RLCP	0.035±0.056	**0.013±0.016**	201±75.214	0.052 ± 0.001
KDD	0.160±0.134	**0.001±0.000**	245±62.147	0.165±0.150
Synthetic 1	15.865±0.006	**15.863±0.004**	530±20.045	15.864±0.005
Higgs	28.320±0.172	**27.656±0.981**	334±130.198	28.307±0.062
Synthetic 2	**6.681±0.004**	**6.681±0.003**	210±60.75	6.683±0.006
Epsilon	28.821±1.095	**27.467±0.322**	254±107.706	54.046±1.029
Non-convex 1	**0.000±0.000**	**0.000±0.000**	230 ± 60.147	**0.000±0.000**
Non-convex 2	**51.333±2.421**	51.721±2.129	232±24.547	51.626±2.182
Non-convex 3	**0.000±0.000**	0.067±0.085	223±43.334	0.000±0.000
Average	14.579±18.203	**14.385±18.125**	273.222±18.125	17.467±22.186

Table 6.3.4 shows that FR-PSMS was capable to obtain the best performance in seven datasets but specially in datasets Higgs and Epsilon, where its performance was remarkable, even compared to PSMS. From the third column in Table 6.3.4, it can be seen that FR-PSMS was capable to reduce considerably the number of evaluations performed in almost all datasets with a performance of 266 evaluations (the average plus the standard deviation) in Non-convex 3 (the best case) but performing 550 evaluations in the worst case (Synthetic 1), surpassing to S-PSMSClass by 50 evaluations. S-PSMSClass was the best method from the previous comparison, because it was able to achieve a similar performance than PSMS but with a lower number of evaluations and smaller execution times. Therefore, in order to know if FR-PSMS is capable to outperform the best approach from the previous stage, a Student's t-test was performed with 43 replications to obtain a power of 90%. In Table 6.3.5 the reported p-values of the test are shown.

According to Table 6.3.5, it can be seen that there are significant differences in almost all convex datasets, except in Synthetic 2. However, in datasets RLCP and Synthetic 1, those p-vales are slightly below the threshold (0.05) and it was considered as that there were no differences in the performance of both methods in those datasets (RLCP, Synthetic 1 and Synthetic 2). Despite that, there were differences in two important datasets: Higgs and Epsilon, and also, in the KDD dataset. Regarding Non-convex datasets, the performance of both methods was similar in the datasets Non-convex 1 and Non-convex 2, while in Non-convex 3, no comparisons could be made. FR-PSMS got the worst performance in Non-convex 3, however, from the other surrogate-based alternatives (PSMS-MLP and S-PSMSReg), its performance

was better and near to a misclassification rate of zero. The computing time factor was also considered and, a bar chart with the performance of both methods along with PSMS is shown in Fig. 6.3.5 for the convex sets and, in Fig. 6.3.6 for the non-convex sets. The hardware configuration was the one mentioned above.

Table 6.3.5: Results obtained from the Student's t-test for the comparison of the performance of FR-PSMS and S-PSMSClass in the FMS analysis in high volume datasets. Cases whose p-vale is below $\boldsymbol{\alpha = 0.05}$ are in **bold**.

Dataset	p-value
RLCP	$\mathbf{0.022}$
KDD	$\mathbf{7.281 \times 10^{-10}}$
Synthetic 1	$\mathbf{0.039}$
Higgs	$\mathbf{9.188 \times 10^{-05}}$
Synthetic 2	0.630
Epsilon	$\mathbf{6.303 \times 10^{-09}}$
Non-convex 1	-
Non-convex 2	0.405
Non-convex 3	-

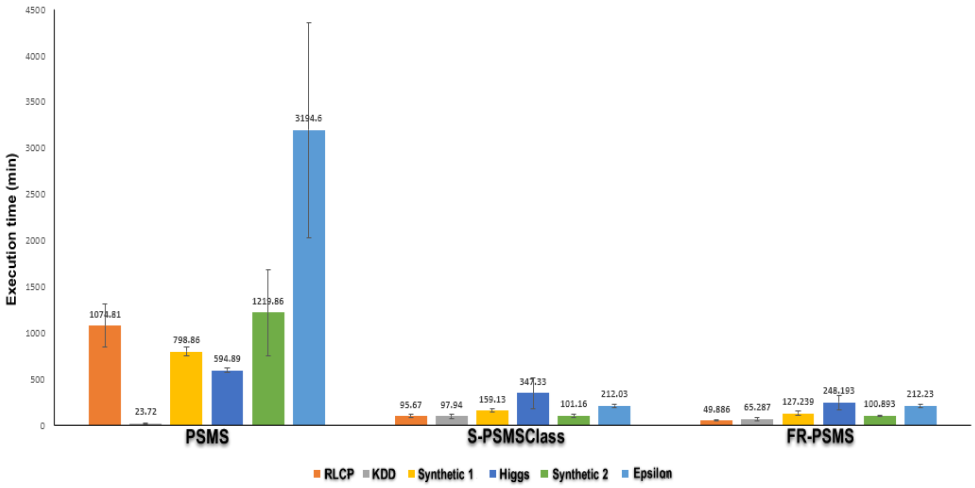


Figure 6.3.5: Bar chart with the average execution times (in minutes) obtained by PSMS, S-PSMSClass and FR-PSMS over 20 replications in the convex datasets. Each color represents a different dataset and the bars are grouped by algorithm. The standard deviation is depicted as a solid black line over the bars.

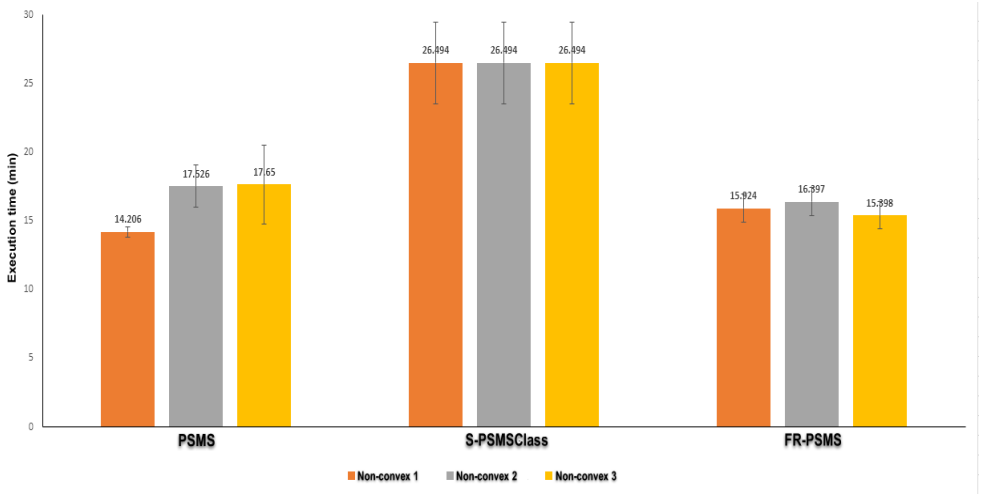


Figure 6.3.6: Bar chart with the average execution times (in minutes) obtained by PSMS, S-PSMSClass and FR-PSMS over 20 replications in the Non-convex datasets. Each color represents a different dataset and the bars are grouped by algorithm. The standard deviation is depicted as a solid black line over the bars.

The incorporation of PSMS to both figures provides further insight into the reduction of the time employed in the search process by both surrogate-assisted methods. From the comparison in the convex sets (Fig. 6.3.5) shows that FR-PSMS obtained a similar performance as the one obtained by S-PSMSClass but with a small reduction of time in the Higgs dataset. However, compared to PSMS, the computing time of FR-PSMS represents a considerable reduction without decreasing the models quality. Regarding Non-convex sets (Fig. 6.3.6), performance of FR-PSMS was near to the one obtained by PSMS and smaller than the one obtained by S-PSMSClass. From all tests performed, there is evidence to support that FR-PSMS is the best approach so far, therefore, it was compared against the Grid search and K-NN. In Table 6.3.6 the average misclassification errors of FR-PSMS, the Grid search and K-NN is shown. To find statistical evidence, an ANOVA with a Dunnett test was performed and the obtained results are shown in Table 6.3.7.

Table 6.3.6: Average misclassification rate obtained in the test partition by FR-PSMS, the Grid search and K-NN over 20 replications. The lowest error rates are in **bold**.

Dataset	FR-PSMS	Grid	K-NN
RLCP	0.011±0.019	**0.001±0.000**	0.500±0.098
KDD	0.156±0.134	**0.001±0.000**	19.535±0.261
Synthetic 1	**15.863±0.003**	17.011±0.120	50.088±0.028
Higgs	**27.258±0.641**	30.955±0.228	46.916±0.408
Synthetic 2	**6.681±0.003**	22.152±0.541	50.126±0.115
Epsilon	**27.232±0.920**	29.664±0.174	50.673±3.323
Non-convex 1	**0.000±0.000**	**0.000±0.000**	**0.000±0.000**
Non-convex 2	51.575±2.167	50.783±1.723	**49.041±2.110**
Non-convex 3	0.031±0.089	**0.000±0.000**	**0.000±0.000**
Average	**14.385±18.125**	16.729±18.297	29.653±24.143

Table 6.3.7: Results obtained from the ANOVA test for the comparison of the performance of FR-PSMS vs the Grid search and K-NN. As a pos hoc analysis, the Dunnett test was performed. The p-values for each comparison and for the ANOVA test are shown. Cases whose p-vale is below $\boldsymbol{\alpha = 0.05}$ are in **bold**.

Dataset	ANOVA p-value	FR-PSMS vs Grid	FR-PSMS vs K-NN
RLCP	$\mathbf{1.237 \times 10^{-36}}$	0.862	$\mathbf{9.560 \times 10^{-10}}$
KDD	$\mathbf{4.326 \times 10^{-95}}$	0.055	$\mathbf{9.560 \times 10^{-10}}$
Synthetic 1	$\mathbf{9.055 \times 10^{-143}}$	$\mathbf{9.560 \times 10^{-10}}$	$\mathbf{9.560 \times 10^{-10}}$
Higgs	$\mathbf{1.245 \times 10^{-77}}$	**0.005**	$\mathbf{9.560 \times 10^{-10}}$
Synthetic 2	$\mathbf{6.438 \times 10^{-83}}$	$\mathbf{9.560 \times 10^{-10}}$	$\mathbf{9.560 \times 10^{-10}}$
Epsilon	$\mathbf{1.634 \times 10^{-43}}$	**0.002**	$\mathbf{9.560 \times 10^{-10}}$
Non-convex 1	-	-	-
Non-convex 2	$\mathbf{6.775 \times 10^{-04}}$	0.432	**0.0005**
Non-convex 3	-	-	-

Table 6.3.6 shows that FR-PSMS obtained the best performance in five datasets (Synthetic 1, Higgs, Synthetic 2, Epsilon and Non-convex 1) while the Grid search, got the best performance in four (RLCP, KDD, Non-convex 1 and Non-convex 3) and K-NN just in three (all the Non-convex). Concerning Table 6.3.7, FR-PSMS outperformed to K-NN in all convex sets and obtained significant differences regarding the Grid search in all datasets where FR-PSMS is the best and there were no differences

in those datasets where the Grid search obtained the best performance. In the non-convex datasets, it can be seen that K-NN was slightly overfitted and FR-PSMS obtained the worst performance in Non-convex 3.

6.3.3 Discussion of the results obtained by FR-PSMS

FR-PSMS was compared against the best method of the previous experimental stage: S-PSMSClass. Both methods employed proxy models based in classification algorithms and built under the FMS paradigm. The mechanisms present in the fuzzy-rules algorithm of FR-PSMS were capable to guide the search of models efficiently, finding models of similar quality of those obtained through a larger search process and even better in two hard datasets: Epsilon and Higgs. The computing time of FR-PSMS was slightly lower compared to the one obtained by S-PSMSClass but the quality of its models surpassed to all previous approaches (GA,PSMS,PSMS-MLP, S-PSMSReg and S-PSMSClass). It is important to mention that FR-PSMS was the first approach capable to surpass the performance of the Grid search in the Epsilon dataset. Despite its performance was not the best in the non-convex datasets, FR-PSMS was capable to obtain good results in two of them, without overfitting in Non-convex 2 and with lower misclassification rates in Non-convex 3, compared to other surrogate based approaches of this Chapter (PSMS-MLP and S-PSMSReg).

The better results and the smaller computing times of FR-PSMS, makes of this approach an excellent alternative to address the FMS problem in high volume datasets. The mechanism provided in this method are capable to make a better use of the time assigned to perform the search of models. With the metaphor of the compass in mind, in this experimental stage arose the idea of to obtain a tool equivalent to a Map for the search space. This idea directed the efforts on this work to explore the use of the meta-learning paradigm.

6.4 Final considerations

In this Chapter, the use of proxy models to assist the search in FMS problem was explored. As in this problem, ideally each dataset represents a different problem, the efforts cannot be directed to the construction of a proxy model to be used in all the datasets, as opposed to other works. In FMS problem, proxy models must be built during the search process and taking into account that fitness evaluation is not a true measure but just an estimation of performance of the models, proxy models are also slightly inaccurate. In the previous Chapter, proposed methods were not able to outperform a simpler Grid search because exploration of the vast search space of

the FMS was harder than the exploration of the hyper-parameter optimization of the grid search. Unlike other approaches, in this work, the use of proxy models as a way to guide the search process instead of a time reducer was explored. Using a proxy model to guide a search can be seen as turning it into a compass and a way to build the best compass is through the FMS paradigm. Following this path, three FMS-based proxy model alternatives were constructed.

- A version of PSMS assisted by proxy models constructed under the FMS paradigm. This alternative used regression algorithms to build a device that predicts the fitness of a particle in order to decide if it is worth assessing this particle with the time-consuming fitness function or not. Proxy models constructed under this paradigm were able to guide efficiently the search process in FMS, obtaining models of higher quality that those obtained by the unassisted search and to a search assisted by a proxy model built with a regression algorithm (traditional approach in the literature). Although some aspects of the FMS paradigm have been considered in the literature as meta-feature selection and hyper-parameter optimization, the use of the entire paradigm obtained accurate proxy models that achieved good solutions performing only a third of the evaluations performed by the unassisted search.

- A version of PSMS assisted by proxy models that employed classification algorithms. This approach was not traditional because there is no prediction of the expected fitness. The transformation of this regression problem into a classification one permitted to choose particles whose fitness was not as good as expected at the moment of its evaluation but, during the evolution of the search process, several of these particles evolved to good quality models. This approach performed a better exploration and achieved superior models than those obtained with traditional approaches based on regression algorithms. The proposed method is especially useful to guide in optimization process where the fitness of a solution is just an estimation and not a real measure as in FMS. Use of classification algorithms in proxy models was first proposed in this work.

- A version of PSMS assisted by a fuzzy-rules classification algorithm developed under the FMS paradigm. This alternative was most atypical but was investigated following the logic that if a classification algorithm as surrogate function was good, a classification algorithm that provides the membership degree of an object to the class "promising" was better. Using the membership degree of the particles as a lower bound, the discrimination process can be finer. Despite that traditional approach (a regression algorithm as proxy model)

achieved an important reduction of the computing-time in the FMS problem, the fuzzy-classification based proxy models achieved both, reduction of the computing-time and highly accurate models. For this reason, this was the best approach so far in this work, well suited for the FMS problem in high volume datasets. To the best of the authors knowledge, this approach was proposed first in this work.

To our best knowledge, this is the first work that proposes the use of the FMS paradigm to build proxy models and the use of classification and fuzzy-classification algorithms. From all the above alternatives, those that employed classification algorithms proved to be the best. The evidence collected in the experiments suggested that algorithm FR-PSMS, the one that uses a fuzzy-classification algorithm as proxy model, was the best of all the explored methods. This algorithm was capable to outperform to the unassisted search obtaining models of high quality, conducting a better exploration of the search space and within a fraction of the time employed by the complete search. FR-PSMS was the first method that surpassed the performance of the Grid search in almost all datasets.

Chapter 7

Use of the meta-learning paradigm to address the FMS problem in High volume datasets.

In the previous Chapter, a device to be employed as a compass to lead through the search space of FMS problem was proposed. As this search space is so big, it is a good idea to find some kind of map of such space. In this Chapter, the use of meta-learning paradigm was explored to assist in the search of models. Meta-learning has been employed in many problems as: instance selection [93], recommendation of under-sampling algorithms [40], image segmentation algorithms [18], and classification algorithms in gene expression classification [43], among others, for that reason it is well suited for the FMS problem. The set of experiments presented in this Chapter is related to the last objective in this work. This objective seeks to gather information about the use of meta-learning to solve the FMS problem in high volume datasets. Unlike other problems in the literature (as model selection) that have been addressed as a supervised learning problem in the meta-learning stage, in FMS, the number of potential models is huge or even infinite, therefore, meta-learning step cannot be approached as a supervised learning task. For that reason, in this Chapter is also proposed a new meta-learner alternative to traditional K-NN algorithm employed in similar tasks. FMS has never been addressed by a meta-learning approach, therefore, the information collected is of special interest, to know if FMS problem can be moved from huge datasets to small meta-datasets. On the other hand, the new meta-learner proposed tries to replicate benefits of a supervised classification process, it is adequate to be employed to solve the FMS problem and supposes an alternative to K-NN, the most employed meta-learner.

The organization of this Chapter is as follows: in Section 7.1 the related work are presented, Section 7.2 describes the proposed methods to solve the FMS problem through the meta-learning paradigm. Section 7.3 is for the experiments and in Section 7.4 the final considerations are presented.

7.1 Related work

In Section 2.3, the meta-learning paradigm was briefly described but, for the task at hand it can be said that, the key idea behind meta-learning is to employ the knowledge acquired from previous similar problems to recommend one or more techniques successfully applied, now for the dataset under analysis [18].

In a technique-recommendation scenario using meta-learning paradigm, dataset must be characterized. This set of characteristics are the meta-features that will be employed in a learning process. This last step is addressed with a learning algorithm, in this case referred as the meta-learner. Several algorithms have been used as meta-learner, some examples are: Decision Trees, Neural Networks, SVM and Naive Bayes but the most popular meta-learner employed is the K-NN algorithm. Analyzed work can be split in two different categories: those that **addresses the meta-learning step as a supervised learning task** [4, 159, 93, 102, 36, 88, 40, 33, 18, 136, 113, 137, 123, 43, 27, 166, 30, 146] and, the ones that **approached this stage as an unsupervised learning one** [126, 39, 107].

The first category is preferred in problems where it is required to know the performance of several techniques in a dataset in the form of a ranking [4, 40, 137, 43, 5, 30], in tasks where there is interest in to analyze the expected performance of an approach [159, 36, 136, 113, 123], and for processes of techniques suggestion [93, 102, 88, 33, 18, 166, 146]. All these works have in common that there is a small quantity of different techniques to recommend, to rank and to predict their performance. With few different methods, these problems can be addressed as a classification problem with one class per algorithm. With sharper categories, classes or groups, success of these approaches can easily be evaluated and if a meta-learner has a low performance it can be replaced by another with higher accuracy. In this way, it can be obtained an outcome nearest to the true result (expected accuracy of a classifier), that is why, the meta-learning step is mostly addressed in this manner.

Regarding second category, all analyzed works [126, 39, 107] make use of meta-learning paradigm to recommend individuals, particles or data-points with high potential for an optimization algorithm. Those potential solutions are evolved, and then better solutions are obtained, in a fraction of the time as opposite of starting the search with an initial set of random solutions. These approaches are mostly employed

in problems where there is a high number of potential solutions to assign a label for each one and, the meta-learning step is performed through similarity measures to find the closest problems to the one under analysis.

To improve the quality of the meta-learning process, several approaches has been followed as model selection for the meta-learner [93, 27], feature selection, in the meta-dataset [159, 36, 33, 123], discretization of the meta-features [159], and the use more meta-features [4, 27]. From this perspective, it can be seen that the use of the FMS paradigm is well suited to improve the meta-learning process. However, although better results can be obtained in a supervised-based meta-learning process, the nature of the FMS problem cannot be adjusted to this paradigm.

In order **to investigate the effect of the FMS paradigm in the meta-learning field** and **to find a way to solve the FMS problem in high volume datasets through meta-learnig**, in this Chapter the following approaches were explored:

- The use of the meta-learning paradigm was investigated to solve FMS problem. In this approach, the employed datasets for experiments were characterized and a meta-dataset was built from previous FMS analyzes. The K-NN algorithm was employed as meta-learner. Although this is the traditional approach employed in the literature, it was not explored before for the FMS problem, therefore is important to know the suitability of meta-learning to a problem with a huge search space that have always been addressed through optimization algorithms. A better performance of meta-learning over a typical approach for FMS (PSMS) would provide evidence that the problem can be moved from bigger datasets to smaller meta-datasets.

- An alternative meta-learner algorithm to K-NN was proposed to improve the quality of the meta-learning process. From above, it can be seen that is preferred to address meta-learning stage as supervised learning task. As the nature of FMS cannot be adapted to this paradigm, a possible solution was to propose a new meta-learner but with some advantages of a supervised learning task, as the discrimination in classes. Following some main trends in the literature, this new algorithm was developed under the FMS paradigm to improve the quality of the process. This new meta-learner and the ability to store knowledge of each FMS task performed, are powerful tools that improves the quality of the models obtained and reducing the time employed for a time-consuming process.

- The use of solutions found through meta-learning as initial population of an optimization algorithm. Since the nature of FMS problem, allows the use of

meta-learning paradigm only through this schema, potential solutions can be employed as an initial swarm better placed in the search space and, obtaining models of high quality in a fraction of the time than using an initial random population. This approach was tested with PSMS and FR-PSMS. The use of the meta-learning paradigm in conjunction with the search assisted by a new kind of proxy models of FR-PSMS, produced a new paradigm not explored before, the Full Meta-Learning Full Model Selection, the fastest and the best approach in this work.

7.2 Use of the meta-learning paradigm to solve the FMS problem in high volume datasets

As detailed above, in the meta-learning paradigm employed in this work, there are two key elements: the meta-dataset construction and the meta-learner algorithm. The meta-dataset consists of meta-features extracted from several datasets and in the case of a supervised based meta-learning approach, a target class. As the FMS paradigm does not fit well in a supervised approach, instead of an identifier of the best technique for a dataset or the expected fitness of such technique, a file with the best models found for a specific dataset was employed. As it can be seen, in order to build a meta-dataset, it is required to analyze several datasets with the FMS analysis. Although this situation can be seen as an important weakness of this paradigm, it is also an important advantage of meta-learning. As has been mentioned through this work, the FMS analysis is a time-consuming process, and in high volume datasets, the process takes more time. Without the use of the meta-learning paradigm, the knowledge gained with each analysis could be lost, therefore, to store useful information and to improve the quality of the models with each analysis performed, the meta-learning approach was considered as an advantage.

From the analysis of the related works, information about the meta-features was obtained. In this work, the most popular meta-features employed across all those works are used (Table 7.2.1). The meta-features are divided in three groups: simple features, statistical and information theory properties. The first group consist of features of the dataset as the number of descriptive variables and the number of instances. Second group is for the statistical properties that describes the dataset as the skewness and kurtosis. Third group describes the dataset in terms of information theory measures as the entropy and the mutual information. Regarding the datasets employed to build the meta-dataset, they are show in Table 7.2.2 and were obtained from Lichman [2013], Fan [2018] and Alcalá-Fdez et al. [2011]. As most of them can

be considered as "small" or that can be analyzed with a conventional desktop PC, the time needed to analyze them all was lower than the time needed to perform a single replication of the Epsilon dataset with PSMS.

Table 7.2.1: Meta-features employed.

Type of meta-feature	Meta-feature
Simple	Proportion of attributes with outliers
	Number of continuous attributes
	Number of discrete attributes
	Proportion of examples and number of attributes
Statistical	Mean absolute correlation coefficient
	Standard deviation ratio
	Bhattacharyya distance
	Mean skewness
	Mean kurtosis
	Fisher discriminant ratio
Information theory	Mean mutual information
	Noise-signal ratio
	Mean entropy

Table 7.2.2: Datasets employed in the creation of the meta-dataset.

banana	breast-cancer	diabetis	flare-solar	german
heart	image	ringnorm	splice	thyroid
titanic	twonorm	waveform	ionosphere	sonar
adult	australian	colon	codrna	fourclass
abalone9vs10	ijcnn1	liver-disorders	madelon	mushrooms
phishing	skinNoskin	susy2	svmguide	svmguide3
w1a	haberman	pima	tictactoe	bupa
crx	monk2	mamographic	houseofvotes	bands
magic	appendicitis	chess	coil2000	phoneme
saheart	fars	spambase	spectfheart	wdbc

7.2.1 An alternative to the K-NN algorithm in an unsupervised meta-learning process

The K-NN algorithm is simple and effective, because it is capable to find similar datapoints (in this case, datasets) to the one under analysis through a distance measure, commonly the Euclidean distance. As mentioned above, in problems as the

one analyzed here, there are no known categories or classes to be able to perform a good discrimination process. To overcome this limitation, the employ of the well-known algorithm for fuzzy clustering, FCM (described in Section 6.2.2) was proposed. Using the instance of the meta-dataset that describes the dataset under analysis as one of the centroids of FCM, we can obtain different groups and the membership degree of each object to each centroid. With this data, the process of finding similar problems can be limited to the objects that belongs to the cluster of interest and, if we need more similar points than the ones available in that cluster, other data points with a higher membership degree to our "centroid" can be obtained. Allowing the competition, the process becomes more reliable assuring that the obtained data points are really more similar to the one under analysis. However, although FCM offers advantages over K-NN, is plain to see that several factors need to be chosen carefully as the distance function, the value of fuzzifier "m", and the number of groups. These and other important factors addressed in the meta-learning literature as the meta-feature selection and pre-processing of the meta-dataset, can be approached through the FMS paradigm. The use of more than one similarity measure and the meta-feature selection process, allow discovering new relations among the objects under analysis that otherwise would be ignored. This optimization is explained below.

7.2.1.1 An FMS-based meta-learner

In order to select the best values and combinations of the factors mentioned above, FCM is optimized through the PSO algorithm. This synergistic combination will be onwards referred as FCM optimized through PSO (PS-FCM). The particles of PSO were codified as a vector $x^i = [x^i_1, x^i_2, ..., x^i_{7+nf}]$. The elements of the particles and the range of values are as follows: Position 1 stores the fitness of particle [0-100]. The fitness function is explained in Section 7.2.1.2. Position 2 is for the Fuzzifier term m of the FCM algorithm [2-10]. Position 3 is the identifier of the similarity measure. Were considered the following similarity measures: 1) Euclidean distance, 2) Manhattan distance, 3) Pearson coefficient, 4) Cosine measure and 5) Chebyshev distance [1-5]. Position 4 is for the number of iterations of FCM [100-500]. Position 5 is for number of clusters [2-10]. Position 6 is the identifier of the data preparation method. The methods employed were: 1) Feature standardization, 2) data normalization and 3) do nothing [1-3]. Alternative methods were considered without differences in the performance of the algorithm. Feature selection process was performed as a wrapper approach, that is why the values that position 7 to nf (number of features) can take are in the interval 0-1, that indicates if the feature will be used or removed.

7.2.1.2 Fitness function for PS-FCM

To evaluate the particles in PS-FCM, it is required to use fuzzy-clustering quality measures. Several quality measures were tried (Dun Index, Bezdek partition entropy, Xie & Beni index, Fukuyama-Sugeno index, Fuzzy hyper-volume) obtaining contradictory results (some of them suggesting the maximum number of clusters and some of them suggesting the minimum). In experiments with toy-datasets, those quality measures were not able to determine the correct number of clusters, even when are easy to identify by plain sight. In order to overcome this obstacle, the Fukuyama-Sugeno and Xie & Beni indexes were employed (they are shown in Eq. 4.7 and 4.8). Those indexes were selected because, it was observed in the experiments that, the Fukuyama-Sugeno index always obtained the highest number of possible clusters (in this case 10), while the Xie & Beni index obtained the lowest (2). To find the most accurate number of fuzzy partitions, both indexes were combined through a fuzzy inference system. The results of each index are scaled in the interval [0-1] and used as input of a fuzzy-rule set. The rules are specified to obtain the maximum value, where both indexes obtain the maximum membership degree in the fuzzy set Medium (the same of Fig. 6.2.2). That is $R_i : If\ XB\ is\ Medium\ And\ FS\ is\ Medium\ Then\ fitness\ is\ High$. In Fig. 7.2.1 the performance of proposed combined indexes in toy-datasets is shown. As PS-FCM is a population-based algorithm, when it converges, all solutions in the swarm are of similar quality. To take advantage of this fact, a voting scheme was employed. In this way, the variance in the list of similar problems is reduced and all the different relations found among the objects is exploited.

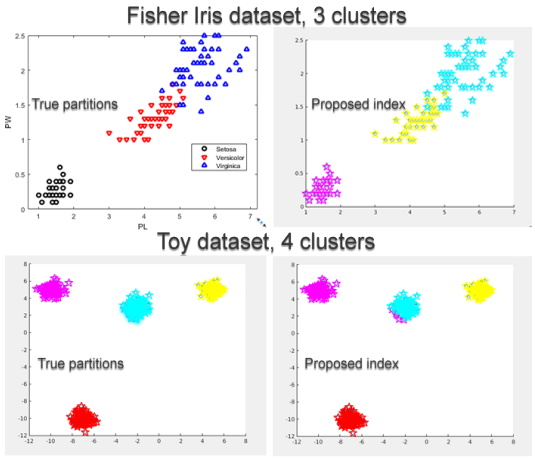


Figure 7.2.1: Performance of the combined quality index in toy-datasets.

7.3 Experiments and results

Both meta-leaners, that is PS-FCM and K-NN, were compared to find differences in their performance. With the best models employed in the 30 most similar problems (due to the size of the swarm) to the one under analysis a weighted voting ensemble was built with the procedure mentioned in Section 5.2.3. Table 7.3.1 shows the average misclassification rates of both meta-leaners and the performance of the best approach so far (FR-PSMS). The performance of FR-PSMS was show as a way to know how near to, or far from, are the meta-learning approaches from FR-PSMS.

Table 7.3.1 shows that PS-FCM obtained the best performance in seven datasets: KDD, Synthetic 1, Higgs, Synthetic 2, Epsilon, Non-convex 1 and Non-convex 3. Comparing the results of PS-FCM against the ones of FR-PSMS, it can be seen that they are pretty close except in the datasets: Higgs, Epsilon and Non-convex 3. In order to find significant differences in the performance of both meta-learners, a Student's t-test was performed (shown in Table 7.3.2) with a statistical power of 90%.

Table 7.3.1: Average misclassification rate obtained in the test partition by PS-FCM and K-NN over 44 replications. The performance of FR-PSMS is just for demonstrative purposes. The lowest error rates are in **bold**.

Dataset	PS-FCM	K-NN for meta-learning	FR-PSMS
RLCP	0.013±0.005	**0.004±0.003**	0.013±0.016
KDD	**0.026±0.005**	0.030±0.004	0.001±0.000
Synthetic 1	**15.863±0.003**	**15.863±0.003**	15.863±0.004
Higgs	**31.041±0.212**	31.046±0.246	27.656±0.981
Synthetic 2	**6.683±0.003**	**6.683±0.003**	6.681±0.003
Epsilon	**31.316±0.876**	31.476±1.740	27.467±0.322
Non-convex 1	**0.000±0.000**	4.161±1.361	0.000±0.000
Non-convex 2	51.490±2.095	**50.268±2.600**	51.721±2.129
Non-convex 3	**1.644±0.120**	3.600±1.992	0.067±0.085
Average	**15.314±18.628**	15.903±17.789	14.385±18.125

Table 7.3.2: Results obtained from the Student's t-test for the comparison of the performance of PS-FCM and K-NN in the FMS analysis in high volume datasets. Cases whose p-vale is below $\boldsymbol{\alpha = 0.05}$ are in **bold**.

Dataset	p-value
RLCP	$\mathbf{4.708 \times 10^{-14}}$
KDD	$\mathbf{3.106 \times 10^{-06}}$
Synthetic 1	0.201
Higgs	0.934
Synthetic 2	0.938
Epsilon	0.586
Non-convex 1	-
Non-convex 2	**0.012**
Non-convex 3	$\mathbf{4.588 \times 10^{-08}}$

From the results in Table 7.3.2, concerning to convex datasets, both meta-leaners were almost similar except in datasets RLCP where the performance of K-NN was the best, and KDD where PS-FCM outperformed to K-NN. From the ID analysis, it can be argued that RLCP is a problem harder than KDD with dimensions of 2 and 1 respectively, therefore, from the convex datasets, K-NN was the best approach. Regarding non-convex sets, it can be seen that the performance of PS-FCM was the best in Non-convex 1 (although no comparisons could be made) and Non-convex 3. Despite there was a significant difference in Non-convex 2, both meta-leaners were not overfitted and, obtaining lower misclassification rates in this dataset, is evidence of an overfitted classifier in an impossible classification task. Consequently, with the best performance in three datasets (KDD, Non-convex 1 and Non-convex 3), the best meta-leaner was PS-FCM.

It is worth to note that, both meta-learners with just 30 models, obtained a performance close to FR-PSMS in the datasets: RLCP, KDD, Synthetic 1, Synthetic 2, Non-convex 1 (only FR-PSMS) and Non-convex 3. Using the meta-learning paradigm in this way, can provide an initial population better placed in the search space, so, the models obtained by PS-FCM were used to feed to PSMS in the datasets Higgs, Epsilon and Non-convex 3. Since this population was better placed, lesser model evaluations were needed, the best performance was obtained after the first 10 iterations without further improvement and even, with deteriorate performance after 20 iterations. The obtained results after 44 replications were: 29.647±0.101 in Higgs, 29.538±0.402 in Epsilon and 0.000±0.000 in Non-convex 3. A Student's t-test was employed to compare the performance of FR-PSMS against PS-FCM, in Table 7.3.3, these results are shown.

Table 7.3.3: Results obtained from the Student's t-test for the comparison of the performance of FR-PSMS and PS-FCM in the FMS analysis in high volume datasets. Cases whose p-vale is below $\boldsymbol{\alpha = 0.05}$ are in **bold**.

Dataset	p-value
RLCP	0.941
KDD	$\mathbf{1.123 \times 10^{-31}}$
Synthetic 1	0.863
Higgs	$\mathbf{6.056 \times 10^{-17}}$
Synthetic 2	0.701
Epsilon	$\mathbf{2.211 \times 10^{-27}}$
Non-convex 1	-
Non-convex 2	0.614
Non-convex 3	-

The information in Table 7.3.3 showed that, despite the refining process performed through PSMS, there were significant differences in three datasets: KDD, Higgs and Epsilon. However, the misclassification rate in the dataset Non-convex 3 was reduced and outperformed the performance of FR-PSMS. Although, this approach was not the best compared to FR-PSMS, its performance was close to the performance of PSMS in the Higgs dataset, where PS-FCM obtained an average misclassification rate of 29.647 ± 0.101 performing 300 evaluations and PSMS obtained 28.299 ± 0.057 after 1,500 evaluations. Regarding Epsilon, the performance of PS-FCM was superior with an average misclassification rate of 31.316 ± 0.876 with just 30 models against 54.008 ± 0.926 of PSMS. Since in the pure meta-learning approach (that is, with no further optimization), the development of the search process cannot be tracked, the figures with this process are not shown, instead, to know the time employed by this approach, Fig. 7.3.1 and Fig. 7.3.2 show the computing times of FR-PSMS and PS-FCM.

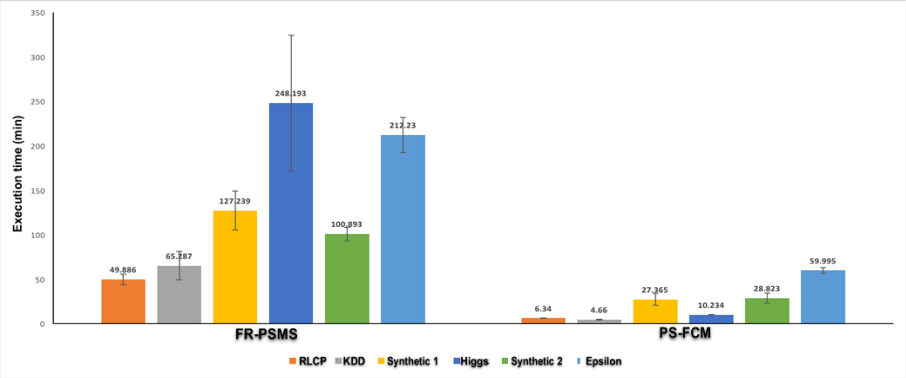


Figure 7.3.1: Bar chart with the average execution times (in minutes) obtained by FR-PSMS and PS-FCM over 20 replications in the convex datasets. Each color represents a different dataset and the bars are grouped by algorithm. The standard deviation is depicted as a solid black line over the bars.

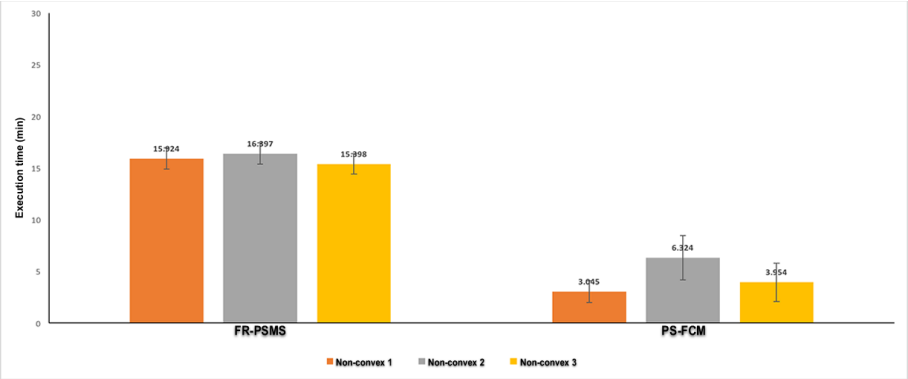


Figure 7.3.2: Bar chart with the average execution times (in minutes) obtained by FR-PSMS and PS-FCM over 20 replications in the Non-convex datasets. Each color represents a different dataset and the bars are grouped by algorithm. The standard deviation is depicted as a solid black line over the bars.

Although best performance in three datasets (Higgs, Epsilon and Non-convex 3) was not obtained by PS-FCM, the computing times shown in the previous figures was dramatically lower than the obtained by all approaches in this document in all datasets. Despite its excellent time reduction, the performance obtained by PS-FCM need to be improved to achieve a good compromise between time and accuracy. In the same way that in previous Chapters, the performance of PS-FCM was compared against the Grid search and K-NN (for classification in high volume datasets). In

Table 7.3.4 are the average misclassification rates and, in Table 7.3.5, the reported results of an ANOVA and a Dunnett test are shown.

Table 7.3.4: Average misclassification rate obtained in the test partition by PS-FCM, the Grid search and K-NN over 20 replications. The lowest error rates are in **bold**.

Dataset	PS-FCM	Grid	K-NN
RLCP	0.012±0.006	**0.001±0.000**	0.500±0.098
KDD	0.026±0.005	**0.001±0.000**	19.535±0.261
Synthetic 1	**15.863±0.003**	17.011±0.120	50.088±0.028
Higgs	**29.633±0.113**	30.955±0.228	46.916±0.408
Synthetic 2	**6.683±0.002**	22.152±0.541	50.126±0.115
Epsilon	**29.461±0.449**	29.664±0.174	50.673±3.323
Non-convex 1	**0.000±0.000**	**0.000±0.000**	**0.000±0.000**
Non-convex 2	51.753±2.042	50.783±1.723	**49.041±2.110**
Non-convex 3	**0.000±0.000**	**0.000±0.000**	**0.000±0.000**
Average	**15.314±18.628**	16.729±18.297	29.653±24.143

Table 7.3.5: Results obtained from the ANOVA test for the comparison of the performance of PS-FCM vs the Grid search and K-NN. As a pos hoc analysis, the Dunnett test was performed. The p-values for each comparison and for the ANOVA test are shown. Cases whose p-vale is below $\boldsymbol{\alpha = 0.05}$ are in **bold**.

Dataset	ANOVA p-value	PS-FCM vs Grid	PS-FCM vs K-NN
RLCP	$\mathbf{5.473 \times 10^{-37}}$	0.771	$\mathbf{2.647 \times 10^{-06}}$
KDD	$\mathbf{4.146 \times 10^{-103}}$	0.812	$\mathbf{2.647 \times 10^{-06}}$
Synthetic 1	$\mathbf{3.866 \times 10^{-135}}$	$\mathbf{2.647 \times 10^{-06}}$	$\mathbf{2.647 \times 10^{-06}}$
Higgs	$\mathbf{3.761 \times 10^{-84}}$	$\mathbf{2.647 \times 10^{-06}}$	$\mathbf{2.647 \times 10^{-06}}$
Synthetic 2	$\mathbf{3.675 \times 10^{-101}}$	$\mathbf{2.647 \times 10^{-06}}$	$\mathbf{2.647 \times 10^{-06}}$
Epsilon	$\mathbf{2.748 \times 10^{-42}}$	0.922	$\mathbf{2.647 \times 10^{-06}}$
Non-convex 1	-	-	-
Non-convex 2	$\mathbf{2.245 \times 10^{-04}}$	0.214	**0.0001**
Non-convex 3	-	-	-

From the results in Table 7.3.4, PS-FCM was able to obtain the best performance in six datasets (Synthetic 1, Higgs, Synthetic 2, Epsilon, Non-convex 1 and Non-convex 3), while the Grid search was the best in four (RLCP, KDD, Non-convex 1 and Non-convex 3), and K-NN jus in three (all the non-convex sets). Table 7.3.5 showed that

there were significant differences in those datasets where PS-FCM obtained the best performance (compared to the Grid search) and in almost all datasets compared to K-NN. Although, PS-FCM was not the best approach from all the explored methods, the evidence above supports that PS-FCM was able to outperform to the Grid search, PSMS and K-NN (both meta-leaner and for classification).

7.3.1 Discussion of the results in the first meta-learning based strategies to perform the FMS analysis in high volume datasets

Two strategies to perform the FMS selection analysis through a meta-learning approach were compared. The K-NN algorithm employed as meta-leaner was able to obtain excellent models in almost all the convex sets, while, PS-FCM, the meta-learner approach designed under the FMS paradigm, was capable to obtain an excellent performance in both, the convex and the non-convex datasets. For this reason, PS-FCM was elected to be compared against the best approach so far, FR-PSMS. Considering that, the final model of PS-FCM was built with the best models obtained from the 30 most similar problems to the dataset under analysis, their results were good but needed an improvement. The way, the meta-learning step was performed in this work, allowed to use these 30 models as an initial swarm for an optimization algorithm as PSMS. Through this optimization step, were obtained better models for the datasets where PS-FCM obtained a higher misclassification rate than FR-PSMS. However, although this optimization was performed, the models of PS-FCM were not capable to outperform the ones of FR-PSMS. Despite this fact, PS-FCM was capable to outperform to the Grid search and K-NN in those datasets where PSMS was not able to outperform the Grid search. Considering that, the information gained with each FMS analysis stored, and the meta-learning process improves with each analysis, this alternative is of special interest and more for searches of models in high volume datasets, where powerful hardware is not available, the meta-learning paradigm allows to move the problem from bigger datasets to smaller meta-datasets.

Taking into account that the models obtained through the meta-learning approach can be refined, a good alternative to do that is through a good search strategy, therefore, the FR-PSMS algorithm and PS-FCM were employed in the same way that the metaphor used in this work of a compass and map to address the big search space of the FMS analysis in high volume datasets. Below are the experiments and results of the synergy of these approaches.

7.3.2 Synergy of the meta-learning paradigm and the proxy models to address the FMS problem in high volume datasets

The experiments above showed the flexibility of the meta-learning paradigm, that can be used as a final answer or as a seed of better searches. However, the combination of this initial population with high potential and an optimization algorithm was not enough to surpass the performance of the best approach so far, FR-PSMS. In pursuance of better models, the idea of combining the two meta-learning techniques used in this work arose. This paradigm or Full Meta-learning to assist in the Full Model Selection problem (FML-FMS), is the synergy of the initial population provided by PS-FCM and the best approach in this work, FR-PSMS. This approach was tested in all the proposed datasets, but there was not important improvement (neither in time nor accuracy) except in datasets: Higgs, Epsilon and Non-convex 3. The performance of FR-PSMS and FML-FMS is shown in Table 7.3.6. In the third and fourth columns are the average evaluations performed by each method. Also, to find significant differences, in Table 7.3.7 the reported p-values of a Student's t-test are shown.

Table 7.3.6: Average misclassification rates obtained in the test partition by FML-FMS and FR-PSMS over 44 replications. The lowest error rates are in **bold**.

Dataset	FML-FMS	FR-PSMS	Mean number of function evaluations in FML-FMS	Mean number of function evaluations in FR-PSMS
Higgs	**27.648±0.409**	27.656±0.981	**269.636±26.145**	323.500±114.157
Epsilon	**26.918±1.712**	27.467±0.322	**212.909±46.896**	247.000±87.706
Non-convex 3	**0.000±0.000**	0.031±0.089	**177.045±44.467**	203.500±54.311
Average	**18.188±15.756**	18.384±15.895	**219.863±46.685**	258.000±60.751

Table 7.3.7: Results obtained from the Student's t-test for the comparison of the performance of FML-FMS and FR-PSMS in the FMS analysis in high volume datasets. Cases whose p-vale is below $\boldsymbol{\alpha = 0.05}$ are in **bold**.

Dataset	p-value
Higgs	0.956
Epsilon	**0.049**
Non-convex 3	-

Table 7.3.6 showed that FML-FMS was capable to obtain the lowest error in the three datasets and performing a lower number of evaluations than FR-PSMS. Regarding the statistical test, FML-FMS outperform to FR-PSMS in the Epsilon dataset, however, the p-value is slightly lower than 0.05, therefore was considered as there was not difference at all. Despite that FML-FMS did not outperformed to FR-PSMS in the convex sets, it was capable to obtain similar results that the best approach with a lower number of evaluations. Regarding the dataset Non-convex 3, FML-FMS outperformed to FR-PSMS. In order to show how the search process was conducted, the Fig. 7.3.3 is provided.

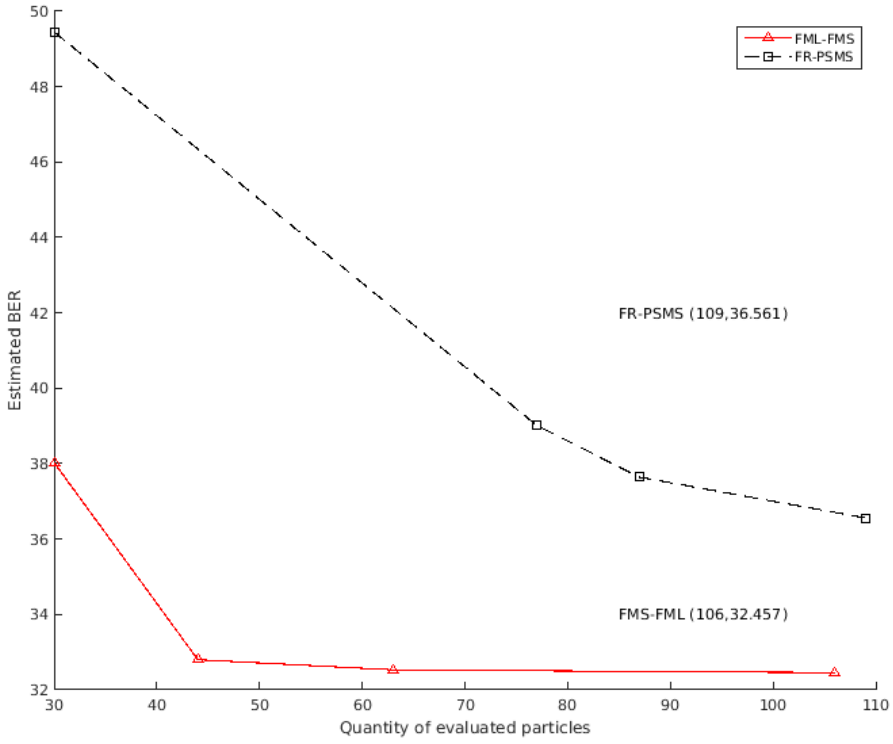


Figure 7.3.3: Search process performed by FML-FMS and FR-PSMS in dataset Higgs. In X axis the quantity of evaluated particles is shown; the Y axis shows the estimated BER of the best particle in the swarm.

Fig. 7.3.3 shows that FML-FMS started the search in a better place than FR-PSMS with an estimated BER of 38% for FML-FMS and around of 50% for FR-PSMS. After 44 evaluations, FML-FMS was capable to outperform to FML-FMS with an estimated BER of 33% that was not obtained by FR-PSMS even after 110 evaluations. From this information, is plain to see that the combination of both meta-learning approaches is able to equip with new capabilities the FMS paradigm. These additional devices provided the capacities to search efficiently a huge search space that involves higher training times. Regarding the time factor, execution-times of FML-FMS and FR-PSMS are shown in Fig 7.3.4 for the convex datasets and, due the differences in the time-scale, the execution-times of both methods in Non-convex 3 are shown in Fig. 7.3.5.

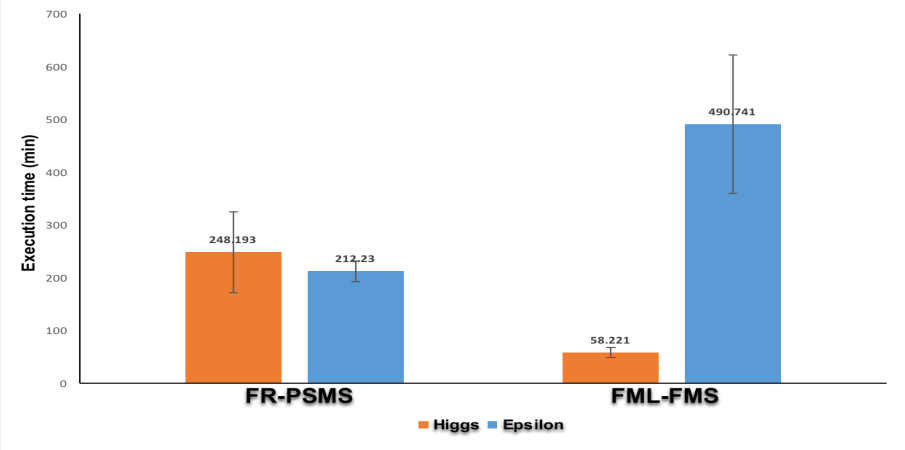


Figure 7.3.4: Bar chart with the average execution times (in minutes) obtained by FML-FMS and FR-PSMS over 20 replications in the convex datasets. Each color represents a different dataset and the bars are grouped by algorithm. The standard deviation is depicted as a solid black line over the bars.

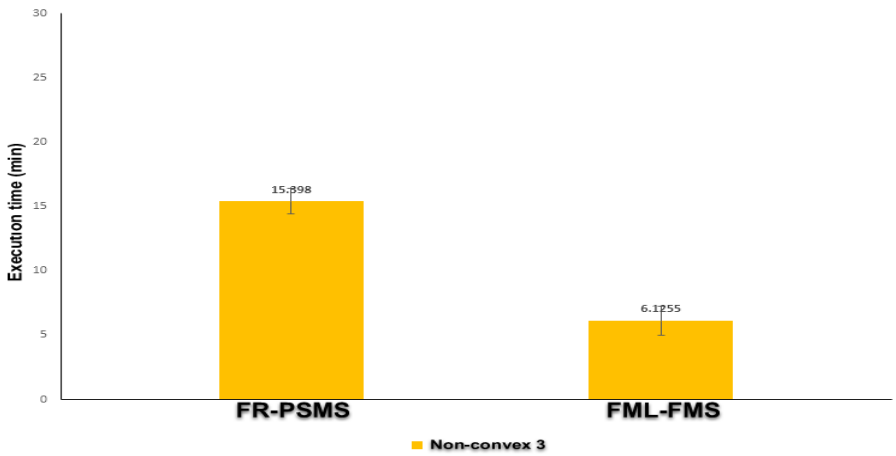


Figure 7.3.5: Bar chart with the average execution times (in minutes) obtained by FML-FMS and FR-PSMS over 20 replications in the Non-convex datasets. Each color represents a different dataset and the bars are grouped by algorithm. The standard deviation is depicted as a solid black line over the bars.

Fig. 7.3.4 shows that FML-FMS considerably reduced the time employed in the search process in datasets Higgs and the Non-convex 3, however, exceeded by far the time of FR-PSMS in the Epsilon dataset. This could be because the initial particles of FML-FMS represents more complex models than the ones of FR-PSMS and therefore, the training of those models takes more time.

FML-FMS was also compared to the Grid search and K-NN, the obtained results are shown in Table 7.3.8 and in Table 7.3.9, the reported p-values of an ANOVA and a Dunnett test are presented.

Table 7.3.8: Average misclassification rate obtained in the test partition by FML-FMS, the Grid search and K-NN over 20 replications. The lowest error rates are in **bold**.

Dataset	FML-FMS	Grid	K-NN
Higgs	**27.594±0.455**	30.955±0.228	46.916±0.408
Epsilon	**26.825±1.904**	29.664±0.174	50.673±3.323
Non-convex 3	**0.000±0.000**	**0.000±0.000**	**0.000±0.000**
Average	**18.139±15.714**	20.206±17.511	32.529±28.234

Table 7.3.9: Results obtained from the ANOVA test for the comparison of the performance of FML-FMS vs the Grid search and K-NN. As a pos hoc analysis, the Dunnett test was performed. The p-values for each comparison and for the ANOVA test are shown. Cases whose p-vale is below $\boldsymbol{\alpha = 0.05}$ are in **bold**.

Dataset	ANOVA p-value	FML-FMS vs Grid	FML-FMS v K-NN
Higgs	$\mathbf{2.482 \times 10^{-78}}$	$\mathbf{2.647 \times 10^{-06}}$	$\mathbf{2.647 \times 10^{-06}}$
Epsilon	$\mathbf{1.030 \times 10^{-40}}$	**0.0003**	$\mathbf{2.647 \times 10^{-06}}$
Non-convex 3	-	-	-

According to Table 7.3.8, FML-FMS was the best approach from all the compared methods in all datasets and from Table 7.3.9, it can be seen that, there were significant differences in all datasets except in Non-convex 3, where all methods obtained the same results.

7.3.3 Discussion of the results obtained by FML-FMS

The synergistic combination of both meta-learning approaches was capable to obtain excellent results in the experiments performed, however, the obtained models did not outperform the ones obtained by FR-PSMS. Despite this fact, is important to note that performance of FML-FMS did not decrease, even considering that the time employed in the process was lower.

This new approach equips with new capacities the FMS analysis, guiding the search process efficiently, storing the information obtained during the process and improving the results with each analysis performed without increase the computing times.

It is worth to mention that were presented the results of only three datasets, because in the remining six, the best obtainable results with the search strategy employed (PSMS) were found using just the suggested solutions offered by PS-FCM and, there was not necessary to use the full meta-learning approach. Nevertheless, the decision of when to use the FML-FMS approach and when use only the models obtained through the meta-learning method (PS-FCM), is not trivial and, it is part of the future work.

From all the evidence obtained, it can be argued that, FML-FMS is the best approach of this work because according the definition of efficiency, the quality of the product (the models in this case) is preserved but without increasing the cost or time. In the FMS problem in high volume dataset, the search of models employing the FML-

FMS approach can be addressed according the budget of time or resources (equipment) of the practitioner, searching for models through the MapReduce paradigm or moving the problem from huge datasets to small meta-datasets.

7.4 Final Considerations

In this Chapter, the meta-learning paradigm to address the FMS problem in high volume datasets was proposed. Nature of FMS makes it harder to address as a supervised meta-learning task, the most common approach in the literature. This approach has been preferred because is easier to address the problem in this way and, if the accuracy of the employed meta-learner is low, it can easily be exchanged by other with better performance, among other factors. Since search space of FMS is huge and each combination of factors represents a model, it cannot be addressed as a supervised meta-learning task. In this work, the meta-learning stage was addressed as a search of most similar problems to the one under consideration. A meta-dataset was built employing the most popular meta-features in the literature and three strategies were followed.

- In the first strategy, the search of most similar problems was performed employing the K-NN algorithm as meta-learner, in the same way that most works in meta-learning field. Although this approach is the most traditional, this is the first work that addresses the FMS problem by means of meta-learning. The meta-learning paradigm empowers FMS with the ability to preserve the information gained with each model selection task and re-use it in new analyses.

- Second strategy employed a new algorithm based on FMS paradigm (PS-FCM). This new algorithm was constructed considering some of the most important trends in the literature to improve the quality of meta-learning process, performing meta-feature selection and data-preparation. As these processes are part of FMS paradigm, it was employed to construct a meta-leaner in conjunction with the FCM algorithm. This new algorithm offers an alternative to K-NN in a meta-learning process. PS-FCM was able to outperform K-NN and although it was not the best approach in this work, the obtained results showed that using the meta-learning paradigm, FMS problem can be moved from huge datasets to smaller meta-datasets.

- The third strategy improved the solutions obtained by a meta-learner through an optimization algorithm. Solutions provided by PS-FCM are in a better region of the search space, enabling in this way a shorter search of models with

a higher quality than those drawn at random. The combination of PS-FCM and FR-PSMS produced a new paradigm, the Full Meta-Learning Full Model Selection, the best approach found in this work to address the search of models in high volume datasets. This new approach was capable to make a better use of the time assigned to the search process, obtaining highly accurate models in a fraction of the time employed by a traditional optimization algorithm, the conventional procedure in this kind of problems.

Both meta-leaners (K-NN and PS-FCM) were compared and PS-FCM was the only one with a good performance in convex and non-convex datasets. The models obtained by PS-FCM were compared against the best approach so far (FR-PSMS) but, the performance of the last was the best. In the third strategy the PSMS algorithm was used to improve the models obtained by PS-FCM. Despite the improvement of solutions, the results did not surpass those of FR-PSMS. As part of the third strategy, models were also improved through the best approach in this work, FR-PSMS. The quality of models slightly improves but without significant differences to the results obtained by FR-PSMS but with an important reduction of the time. FML-FMS was capable to obtain good results in six datasets without the optimization step, therefore, there is evidence that is more efficient than previous approaches. These new tools equipped the FMS paradigm with new capacities to store and improve with each analysis the information obtained of problems addressed.

Part IV

General conclusion

Chapter 8

Conclusions

The development of new technologies has marked a trend in the generation of high amounts of data in several and distinct fields of human endeavor. In the period between 2012 and 2014 was created the 90% of the data (to that time) in the world [160], while the CERN and the European Institute of Bioinformatic generated around 20 and 15 petabytes each per year [104].

This data becomes information and justify its storage expenditure when it is analyzed and, big part of this analysis is performed by means of machine learning algorithms. However, the choice of the right learning algorithm is not a trivial task, even when data fits easily in the main memory of a standard personal computer.

Selection of the right algorithm that fits better a dataset is a problem known as model selection and although the availability of more data could be thought as an accuracy booster for any learning algorithm, the reality is that there is now evidence that a given learning is superior to others in all datasets, no matter how much data we have. This situation is addressed by the "No Free Lunch Theorem" of Wolpert and Macready [1997] and basically states that no machine learning algorithm is the best for all problems.

Accuracy of a learning algorithm is influenced by other factors as the selection of a subset of features that describes better the phenomenon in the dataset and a combination of pre-processing techniques to transform data into a more effective format for the learning algorithm. This combination of feature selection, data-preparation techniques, the selection of a learning algorithm and the tuning of its hyper-parameters is known as Full Model Selection [57].

This paradigm is capable to obtain models with higher accuracy and it has been employed in several problems as region labeling on images [75], assist in the diagnosis for acute leukemia [58], and classification of infant crying patterns [132], among others. FMS can be considered as a Black Box whose input is a dataset and their output is a highly accurate model, therefore it can be extremely useful for both experts and practitioners.

The search space of this problem is huge or infinite even if restrictions are imposed to the hyper-parameter values of the considered techniques. FMS in conventional size datasets involves the use of time-consuming optimization techniques but with the bigger datasets of nowadays, this optimization process becomes intractable.

In order to equip this paradigm with the capacity to deal with high volume datasets, the objective of this work was *"to develop methods to perform a better exploration of the search space imposed by this problem in high volume datasets in terms of computing-time and models accuracy"*. The followed hypothesis was that *"models obtained through the full model selection analysis in high volume datasets assisted by efficient proxy models and the meta-learning paradigm obtain a lower misclassification rate and are obtained performing a lower number of fitness evaluations than those obtained without the use of the aforementioned techniques"*.

For easier understanding, the followed approaches and contributions are described in the next sections as well as the future work.

8.1 Approaches followed in this work

Three sets of experiments were performed related to the specific objectives in this work. In the next sections they are briefly described.

8.1.1 First approach to solve the FMS problem in high-volume datasets

To be able to explore the vast search space of FMS and the big computing times of high-volume datasets, two important bio-inspired optimization algorithms were taken to MapReduce. In the literature revision was highlighted that all the works that addresses FMS problem employed bio-inspired optimization algorithms because

the enormous number of combinations cannot be tried through simpler grid searches. On the other hand, just one work addressed the problem of bigger computing times through the MapReduce programming model.

Combining these ideas, a GA and a model selection algorithm based on PSO (PSMS) were adapted to MapReduce and comparisons were made. Although similar comparisons were made in the past, such algorithms were developed employing different frameworks and algorithms in the stages involved in the FMS process (feature selection, data-preparation and learning step).

The **contributions obtained** from this set of experiments was a framework to adapt population-based search algorithms for FMS in high volume datasets. To the best of the authors knowledge, this is the first approach to address this problem in high volume dataset. Moreover, the information obtained from the comparison in similar conditions of the aforementioned algorithms was also a contribution of this work.

Conclusions extracted from these experiments were that PSMS outperformed the GA, therefore PSMS is better suited for FMS in high volume datasets, however both approaches were outperformed by a simpler grid search (also based in MapReduce) in a similar amount of time. Search space of FMS makes necessary to seek for tools to guide efficiently through optimization process in order to obtain models of higher quality and reduce the computing time.

8.1.2 Use of proxy models to guide in FMS process

The second set of experiments explored the use of proxy models. Although, proxy models have been used to speed up time-consuming optimization processes as calibration of hydrological models [82], petroleum reservoir modeling [163], natural gas fields reservoir simulations [3], and wind turbines design [157], it was analyzed its suitability to guide efficiently the search process of FMS in terms of models accuracy and reduction of the time employed.

The state-of-the-art analysis revealed that to improve the accuracy of proxy models, some steps of the FMS paradigm are used in the creation of such models as feature selection and data-preparation. With this in mind and from the evidence in the literature of the capacity of FMS to obtain highly accurate models, this paradigm was employed in the proxy model construction. This approach addressed the second

objective in this work to gain information about the use of FMS to build better proxy models.

Three strategies to build proxy models were followed to address the third objective in this work. The first one employed FMS to build proxy models, that is, performing data-preparation, feature selection, selection of a learning algorithm and hyper-parameter optimization on the meta-dataset.

The second one used classification algorithms in conjunction with the FMS paradigm. First approach performed the model selection step among regression algorithms. In this approach instead of predicting the expected fitness of a model, a discrimination between promising and not promising models was provided.

In the third strategy a fuzzy classification algorithm built under FMS paradigm was employed. Fuzzy classification algorithms provide a membership degree of an object to considered classes. This membership degree can be employed to perform a finer discrimination.

The **contributions** obtained from this set of experiments are as follows:

- Introduction of the use of FMS paradigm to build proxy models. Some parts of FMS paradigm have been employed in the literature to boost the accuracy of proxy models, however, this is the first work that proposes the use of this paradigm to build proxy models based on the evidence of the high-quality classification models obtained by this paradigm. Proxy models built in this way have a higher accuracy and obtain models of higher quality in comparison to traditional approaches based only in one regression algorithm.

- Introduction of the use of FMS and classification algorithms in the proxy model construction. This approach is especially useful when the expected fitness of a solution is an estimation and not a true measure of a phenomenon as in model selection. This idea was also presented first in this work.

- A new algorithm based on fuzzy rules and the FMS paradigm that can be used as proxy model or a multi-class classification algorithm. This algorithm exploits similarities among the data points that represent solutions in an optimization algorithm and provides in addition to target class (promising and not promising) the membership degree to each one.

Conclusions from these experiments are:

- The use of FMS paradigm obtained accurate proxy models that achieved good solutions performing only a third of the evaluations performed by the unassisted search PSMS.
- Turning proxy models from a regression problem into a classification one permitted to choose particles whose fitness was not as good as expected at the moment of its evaluation but, during the evolution of the search process, several of these particles evolved to good quality models. Using FMS in conjunction with classification algorithms for proxy models, performed a better exploration and achieved superior models than those obtained by the regression-based proxy models.
- Using a fuzzy-classification algorithm built under the FMS paradigm performed a better discrimination process and achieved an important reduction of the computing-time in the FMS problem. Despite that traditional approach (a regression algorithm as proxy model) achieved an important reduction of the computing-time in the FMS problem, the fuzzy-classification based proxy models achieved both, reduction of the computing-time and models of higher accuracy than the aforementioned approaches.

8.1.3 Use of meta-learning to address FMS problem in high-volume datasets

The use of meta-learning paradigm was explored to assist in the search of models. Meta-learning has been employed in many problems as: instance selection [93], recommendation of under-sampling algorithms [40], image segmentation algorithms [18], and classification algorithms in gene expression classification [43], among others, that is why it is well suited for the FMS problem.

This set of experiments was related to the last objective in this work. That is, to gather information about the use of meta-learning to solve the FMS problem in high volume datasets. Unlike other problems in the literature that have been addressed as a supervised learning problem in the meta-learning stage, in FMS, the number of potential models is huge, therefore, meta-learning step cannot be approached as a supervised learning task.

A revision of the literature showed that most works addressed the meta-learning step as a classification problem with one class per algorithm or technique. With sharper categories, classes or groups, success of these approaches can easily be evaluated and if a meta-learner has a low performance, it can be replaced by another with higher accuracy. On the other hand, few works approached meta-learning step as the recommendation of particles or data-points with high potential for an optimization algorithm through a meta-learner based on similarity measures, K-NN most of the time.

A meta-dataset was built employing the most popular meta-features present in all the considered works. Also, the traditional K-NN algorithm was employed as meta-learner. Although this is the traditional approach in the literature, it was not explored before for the FMS problem, therefore was important to know the suitability of meta-learning to a problem with a huge search space that have always been addressed through optimization algorithms.

A second strategy followed was to propose an alternative meta-learner (PS-FCM) to K-NN in order to improve the quality of the meta-learning process. Following some main trends in the literature, this new algorithm was developed under the FMS paradigm.

The last strategy proposed was to use the solutions found through meta-learning as initial population of an optimization algorithm. This approach was tested with PSMS and FR-PSMS.

The **contributions** obtained from this set of experiments are as follows:

- A method to use meta-learning techniques to solve the full model selection problem. Although this is the traditional approach employed in their literature, it was not explored before for the FMS problem.
- A new meta-learner algorithm alternative to K-NN. According main trends in the literature, this new algorithm was developed under the FMS paradigm to improve the quality of the process. The new meta-learner and the ability to store knowledge of each FMS task performed, are powerful tools that improves the accuracy of the models obtained and reducing the time employed in a time-consuming process.
- A new paradigm to address FMS in high volume datasets, that is, the synergy between meta-learning along with the search assisted by a new kind of proxy

models of FR-PSMS. This paradigm is known as Full Meta-Learning Full Model Selection.

Conclusions from these experiments are:

- Meta-learning paradigm empowers FMS with the ability to preserve the information gained with each model selection task and re-use it in new analyses. Meta-learning makes possible to move FMS problem from high volume datasets to smaller meta-datasets.
- The new meta-learner proposed offers an alternative to K-NN and is capable to outperform it obtaining better models in a fraction of the time required by PSMS.
- Full Meta-Learning Full Model Selection was the best approach found in this work to address the search of models in high volume datasets. This approach outperformed all previous methods in terms of model accuracy and computing-time.

8.2 Future work

Several aspects from the explored strategies were not addressed in this work. They are part of the future work and are mentioned below:

- As mentioned above, the FML-FMS approach is the combination of the models provided by PS-FCM and the optimization performed by FR-PSMS. However, although the use of the meta-learning was investigated to know when use just PS-FCM and when to use both methods in combination, the results did not provide accurate information, therefore this should be further developed.
- Meta-learning can be employed in different levels of granularity, that is, from recommending the right technique for the whole dataset to recommendation by each instance. As finer level of granularity is not an option for huge datasets, it could be of great interest to explore this paradigm in subsets of a dataset to obtain more robust models.
- Since the FMS analysis provides information about datasets, it would be interesting to study the incorporation of the information collected through this method in the meta-features employed by PS-FCM. This is known in the

literature as landmarking but to the best of the authors knowledge it was not performed through the FMS paradigm. FMS can bring very useful information as the number of features with better discriminant power, the best combination of preprocessing techniques, the best learning algorithms for a dataset, among others.

- Although the use of a fuzzy inference system was a useful solution to combine the performance of two fitness measures in PS-FCM, it would be interesting to investigate the utility of this method compared to the well-known Pareto front.

Bibliography

[1] J. Abonyi and B. Feil. *Cluster Analysis for Data Mining and System Identification*. Birkhäuser Basel, 2007. ISBN 9783764379889. URL `https://books.google.com.mx/books?id=T9TDO9eaG58C`.

[2] Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, Salvador García, Luciano Sánchez, and Francisco Herrera. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17, 2011.

[3] Faisal Alenezi and Shahab Mohaghegh. A data-driven smart proxy model for a comprehensive reservoir simulation. In *Information Technology (Big Data Analysis)(KACSTIT), Saudi International Conference on*, pages 1–6. IEEE, 2016.

[4] Kalousis Alexandros and Hilario Melanie. Model selection via meta-learning: a comparative study. *International Journal on Artificial Intelligence Tools*, 10 (04):525–554, 2001.

[5] Rahman Ali, Aasad Masood Khatak, Francis Chow, and Sungyoung Lee. A case-based meta-learning and reasoning framework for classifiers selection. In *Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication*, page 31. ACM, 2018.

[6] Laya Aliahmadipour and Esfandiar Eslami. A hybrid approach to optimize fuzzy if-then rules. In *Intelligent Systems (ICIS), 2014 Iranian Conference on*, pages 1–5. IEEE, 2014.

[7] Apacheorg. Ml tuning: model selection and hyperparameter tuning. `http://spark.apache.org/docs/latest/ml-tuning.html`, Aug 2016.

[8] Ilhan Aydin, Mehmet Karakose, and Erhan Akin. A multi-objective artificial immune algorithm for parameter optimization in support vector machine. *Applied soft computing*, 11(1):120–129, 2011.

[9] Bhavna Bansal and Anita Sahoo. Full model selection using bat algorithm. In *Cognitive Computing and Information Processing (CCIP), 2015 International Conference on*, pages 1–4. IEEE, 2015.

[10] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

[11] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[12] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012. ISSN 1532-4435.

[13] James Bergstra, Dan Yamins, and David D Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference*, pages 13–20, 2013.

[14] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[15] R.C. Blattberg, B.D. Kim, and S.A. Neslin. *Database Marketing: Analyzing and Managing Customers*. International Series in Quantitative Marketing. Springer New York, 2008. ISBN 9780387725789. URL `https://books.google.com.mx/books?id=-JwptfFItaoC`.

[16] C. Borgelt, M.Á. Gil, J.M.C. Sousa, and M. Verleysen. *Towards Advanced Data Analysis by Combining Soft Computing and Statistics*. Studies in Fuzziness and Soft Computing. Springer Berlin Heidelberg, 2012. ISBN 9783642302787. URL `https://books.google.com.mx/books?id=qri5BQAAQBAJ`.

[17] P. Brazdil, C.G. Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining*. Cognitive Technologies. Springer Berlin Heidelberg, 2008. ISBN 9783540732624. URL `https://books.google.com.mx/books?id=-Gsi_cxZGpcC`.

[18] Gabriel FC Campos, Sylvio Barbon, and Rafael G Mantovani. A meta-learning approach for recommendation of image segmentation algorithms. In *Graphics, Patterns and Images (SIBGRAPI), 2016 29th SIBGRAPI Conference on*, pages 370–377. IEEE, 2016.

[19] Jenna Carr. An introduction to genetic algorithms. *Senior Project*, 1:40, 2014.

[20] Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul):2079–2107, 2010.

[21] Claudio Ceruti. Novel techniques for intrinsic dimension estimation. 2014.

[22] Claudio Ceruti, Simone Bassis, Alessandro Rozza, Gabriele Lombardi, Elena Casiraghi, and Paola Campadelli. Danco: Dimensionality from angle and norm concentration. *arXiv preprint arXiv:1206.3881*, 2012.

[23] Simon Chan, Philip Treleaven, and Licia Capra. Continuous hyperparameter optimization for large-scale recommender systems. In *Big Data, 2013 IEEE International Conference on*, pages 350–358. IEEE, 2013.

[24] Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine learning*, 46(1-3):131–159, 2002.

[25] Clément Chatelain, Sébastien Adam, Yves Lecourtier, Laurent Heutte, and Thierry Paquet. A multi-model selection framework for unknown and/or evolutive misclassification cost problems. *Pattern Recognition*, 43(3):815–823, 2010.

[26] S. Chatterjee and A.S. Hadi. *Regression Analysis by Example*. Wiley Series in Probability and Statistics. Wiley, 2006. ISBN 9780470055458. URL `https://books.google.com.mx/books?id=uiu5XsAA9kYC`.

[27] Lena Chekina, Lior Rokach, and Bracha Shapira. Meta-learning for selecting a multi-label classification algorithm. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 220–227. IEEE, 2011.

[28] Min Chen, Shiwen Mao, Yin Zhang, and Victor CM Leung. Big data: related technologies, challenges and future prospects. 2014.

[29] Jeoung-Nae Choi, Young-Il Lee, and Sung-Kwun Oh. Fuzzy radial basis function neural networks with information granulation and its genetic optimization. *Advances in Neural Networks–ISNN 2009*, pages 127–134, 2009.

[30] Andrew Collins, Joeran Beel, and Dominika Tkaczyk. One-at-a-time: A meta-learning recommender-system for recommendation-algorithm selection on micro level. *arXiv preprint arXiv:1805.12118*, 2018.

[31] Ivo Couckuyt, Filip De Turck, Tom Dhaene, and Dirk Gorissen. Automatic surrogate model type selection during the optimization of expensive black-box problems. In *Simulation Conference (WSC), Proceedings of the 2011 Winter*, pages 4269–4279. IEEE, 2011.

[32] Karel Crombecq, Luciano De Tommasi, Dirk Gorissen, and Tom Dhaene. A novel sequential design strategy for global surrogate modeling. In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 731–742. IEEE, 2009.

[33] Rafael MO Cruz, Robert Sabourin, and George DC Cavalcanti. Meta-des. oracle: Meta-learning and feature selection for dynamic ensemble selection. *Information Fusion*, 38:84–103, 2017.

[34] Israel Cruz-Vega, Mauricio Garcia-Limon, and Hugo Jair Escalante. Adaptive-surrogate based on a neuro-fuzzy network and granular computing. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 761–768. ACM, 2014.

[35] Israel Cruz-Vega, Carlos Alberto Reyes García, Pilar Gómez Gil, Juan Manuel Ramírez Cortés, and José de Jesús Rangel Magdaleno. Genetic algorithms based on a granular surrogate model and fuzzy aptitude functions. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 2122–2128. IEEE, 2016.

[36] Can Cui, Mengqi Hu, Jeffery D Weir, and Teresa Wu. A recommendation system for meta-modeling: A meta-learning based approach. *Expert Systems with Applications*, 46:33–44, 2016.

[37] W. Daelemans and K. Morik. *Machine Learning and Knowledge Discovery in Databases: European Conference, Antwerp, Belgium, September 15-19, 2008, Proceedings*. Number pt. 2 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008. ISBN 9783540874812. URL `https://books.google.com.mx/books?id=dHNqCQAAQBAJ`.

[38] S. Das. *Data Science Using Oracle Data Miner and Oracle R Enterprise: Transform Your Business Systems into an Analytical Powerhouse*. Apress, 2016. ISBN 9781484226148. URL https://books.google.com.mx/books?id=rejIDQAAQBAJ.

[39] Péricles BC de Miranda, Ricardo BC Prudêncio, Andre Carlos PLF de Carvalho, and Carlos Soares. Combining a multi-objective optimization approach with meta-learning for svm parameter selection. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pages 2909–2914. IEEE, 2012.

[40] Romero FAB de Morais, Péricles BC Miranda, and Ricardo MA Silva. A meta-learning method to select under-sampling algorithms for imbalanced data sets. In *Intelligent Systems (BRACIS), 2016 5th Brazilian Conference on*, pages 385–390. IEEE, 2016.

[41] J.V. de Oliveira and W. Pedrycz. *Advances in Fuzzy Clustering and its Applications*. Wiley, 2007. ISBN 9780470061183. URL https://books.google.com.mx/books?id=Pn0e1xm4YBgC.

[42] Vin De Silva and Gunnar E Carlsson. Topological estimation using witness complexes. *SPBG*, 4:157–166, 2004.

[43] Bruno F de Souza, André de Carvalho, and Carlos Soares. Metalearning for gene expression data classification. In *Hybrid Intelligent Systems, 2008. HIS'08. Eighth International Conference on*, pages 441–446. IEEE, 2008.

[44] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[45] Amir Hossein Saeedi Dehaghani and Mohammad Hasan Badizad. A soft computing approach for prediction of p- Ï-t behavior of natural gas using adaptive neuro-fuzzy inference system. *Petroleum*, pages –, 2016. ISSN 2405-6561. doi: https://doi.org/10.1016/j.petlm.2016.12.004. URL http://www.sciencedirect.com/science/article/pii/S2405656116301213.

[46] Sara del Río, Victoria López, José Manuel Benítez, and Francisco Herrera. On the use of mapreduce for imbalanced big data using random forest. *Information Sciences*, 285:112–137, 2014.

[47] Sara del Río, Victoria López, José Manuel Benítez, and Francisco Herrera. A mapreduce approach to address big data classification problems based on

the fusion of linguistic fuzzy rules. *International Journal of Computational Intelligence Systems*, 8(February):422–437, 2015. ISSN 1875-6891. doi: 10.1080/18756891.2015.1017377. URL http://www.tandfonline.com/doi/abs/10.1080/18756891.2015.1017377.

[48] A. Deshpande and M. Kumar. *Artificial Intelligence for Big Data: Complete guide to automating Big Data solutions using Artificial Intelligence techniques*. Packt Publishing, 2018. ISBN 9781788476010. URL https://books.google.com.mx/books?id=pF9dDwAAQBAJ.

[49] R.M. Díaz, F. Pichler, and A.Q. Arencibia. *Computer Aided Systems Theory - EUROCAST 2009: 12th International Conference, Las Palmas de Gran Canaria, Spain, February 15-20, 2009, Revised Selected Papers*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009. ISBN 9783642047725. URL https://books.google.com.mx/books?id=FBJtCQAAQBAJ.

[50] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[51] K.L. Du and M.N.S. Swamy. *Neural Networks and Statistical Learning*. SpringerLink : Bücher. Springer London, 2013. ISBN 9781447155713. URL https://books.google.com.mx/books?id=wzK8BAAAQBAJ.

[52] Didier Dubois and Henri Prade. What are fuzzy rules and how to use them. *Fuzzy Sets and Systems*, 84(2):169 – 185, 1996. ISSN 0165-0114. doi: https://doi.org/10.1016/0165-0114(96)00066-8. URL http://www.sciencedirect.com/science/article/pii/0165011496000668. Dedicated to the Memory of Professor Arnold Kaufmann.

[53] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley, 2012. ISBN 9781118586006. URL https://books.google.com.mx/books?id=Br33IRC3PkQC.

[54] Sašo Džeroski, Panče Panov, Dragi Kocev, and Ljupčo Todorovski. *Discovery Science: 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014, Proceedings*, volume 8777. Springer, 2014.

[55] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on information theory*, 29 (4):551–559, 1983.

[56] Max J Egenhofer. A formal definition of binary topological relationships. In *International conference on foundations of data organization and algorithms*, pages 457–472. Springer, 1989.

[57] Hugo Jair Escalante, Manuel Montes, and Luis Enrique Sucar. Particle swarm model selection. *Journal of Machine Learning Research*, 10(Feb):405–440, 2009.

[58] Hugo Jair Escalante, Manuel Montes-y Gómez, Jesús A González, Pilar Gómez-Gil, Leopoldo Altamirano, Carlos A Reyes, Carolina Reta, and Alejandro Rosales. Acute leukemia classification by ensemble particle swarm model selection. *Artificial intelligence in medicine*, 55(3):163–175, 2012.

[59] Larry J Eshelman. The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of genetic algorithms*, volume 1, pages 265–283. Elsevier, 1991.

[60] Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):12140, 2017.

[61] R. Fan. Libsvm data: Classification, regression, and multi-label. `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`, 2018.

[62] Y Fukuyama and M Sugeno. A new method of choosing the number of clusters for the fuzzy c-means method. 01 1989.

[63] S. García, J. Luengo, and F. Herrera. *Data Preprocessing in Data Mining*. Intelligent Systems Reference Library. Springer International Publishing, 2014. ISBN 9783319102474. URL `https://books.google.com.mx/books?id=SbFkBAAAQBAJ`.

[64] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Springer, 2015.

[65] Hossein Ghalkhani, Saeed Golian, Bahram Saghafian, Ashkan Farokhnia, and Asaad Shamseldin. Application of surrogate artificial intelligent models for real-time flood routing. *Water and Environment Journal*, 27(4):535–548, 2013.

[66] Aliakbar Golzari, Morteza Haghighat Sefat, and Saeid Jamshidi. Development of an adaptive surrogate model for production optimization. *Journal of Petroleum Science and Engineering*, 133:677–688, 2015.

[67] Michael T Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In *International Symposium on Algorithms and Computation*, pages 374–383. Springer, 2011.

[68] Dirk Gorissen, Tom Dhaene, and Filip De Turck. Evolutionary model type selection for global surrogate modeling. *Journal of Machine Learning Research*, 10(Sep):2039–2078, 2009.

[69] Norma Van Surdam Graham. *Visual pattern analyzers*. Oxford University Press, 1989.

[70] XC Guo, JH Yang, CG Wu, CY Wang, and YC Liang. A novel ls-svms hyper-parameter selection based on particle swarm optimization. *Neurocomputing*, 71 (16):3211–3215, 2008.

[71] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[72] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[73] Aboul Ella Hassanien, Ahmad Taher Azar, Vaclav Snasel, Janusz Kacprzyk, and J Abawajy. *Big Data in Complex Systems*. Springer, 2015.

[74] Randy L Haupt, Sue Ellen Haupt, and Sue Ellen Haupt. *Practical genetic algorithms*, volume 2. Wiley New York, 1998.

[75] Escalante-Balderas H.J. *Cohesión semántica para la anotación y recuperación de imágenes*. PhD thesis, Instituto Nacional de Astrofísica, Óptica y Electrónica, Marzo 2010.

[76] Chien-Ming Huang, Yuh-Jye Lee, Dennis KJ Lin, and Su-Yun Huang. Model selection for support vector machines via uniform design. *Computational Statistics & Data Analysis*, 52(1):335–346, 2007.

[77] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

[78] Hisao Ishibuchi and Ken No &. Selection of fuzzy if-7kn rules by a genetic method. 1994.

[79] Thomas Martini Jørgensen and Christian Linneberg. Feature weighted ensemble classifiers–a modified decision scheme. In *International Workshop on Multiple Classifier Systems*, pages 218–227. Springer, 2001.

[80] J. Jose. *Customer Payment Trend Analysis Based on Clustering for Predicting the Financial Risk of Business Organizations*. Anchor Academic Publishing, 2017. ISBN 9783960671046. URL `https://books.google.com.mx/books?id=WT7vDQAAQBAJ`.

[81] J-C Jouhaud, Pierre Sagaut, Marc Montagnac, and Julien Laurenceau. A surrogate-model based multidisciplinary shape optimization method with application to a 2d subsonic airfoil. *Computers & Fluids*, 36(3):520–529, 2007.

[82] M Kamali, P Ponnambalam, and ED Soulis. Hydrologic model calibration using fuzzy tsk surrogate model. In *Fuzzy Information Processing Society, 2005. NAFIPS 2005. Annual Meeting of the North American*, pages 799–803. IEEE, 2005.

[83] Hiromasa Kaneko and Kimito Funatsu. Fast optimization of hyperparameters for support vector regression models with highly predictive ability. *Chemometrics and Intelligent Laboratory Systems*, 142:64–69, 2015. doi: 10.1016/j.chemolab.2015.01.001.

[84] Mikhail Kanevski, Alexei Pozdnoukhov, and Vadim Timonin. *Machine learning for spatial environmental data: theory, applications, and software*. EPFL press, 2009.

[85] Chaouki Khammassi and Saoussen Krichen. A ga-lr wrapper approach for feature selection in network intrusion detection. *computers & security*, 70: 255–277, 2017.

[86] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[87] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *The Journal of Machine Learning Research*, 18(1):826–830, 2017.

[88] Mirko Kück, Sven F Crone, and Michael Freitag. Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 1499–1506. IEEE, 2016.

[89] Jan Larsen, Claus Svarer, Lars Nonboe Andersen, and Lars Kai Hansen. Adaptive regularization in neural network modeling. In *Neural Networks: Tricks of the Trade*, pages 113–132. Springer, 1998.

[90] Jacek Laskowski. Mastering apache spark, 2016.

[91] S.R. Lay. *Convex Sets and Their Applications*. Dover Books on Mathematics Series. Dover Publications, 2007. ISBN 9780486458038. URL `https://books.google.com.mx/books?id=U9eOPjmaH90C`.

[92] Stefan Lessmann, Robert Stahlbock, and Sven F Crone. Genetic algorithms for support vector machine model selection. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 3063–3069. IEEE, 2006.

[93] Enrique Leyva, Yoel Caises, Antonio González, and Raúl Pérez. On the use of meta-learning for instance selection: An architecture and an experimental study. *Information Sciences*, 266:16–30, 2014.

[94] H.H. Li and M.M. Gupta. *Fuzzy Logic and Intelligent Systems*. International Series in Intelligent Technologies. Springer Netherlands, 2007. ISBN 9780585280004. URL `https://books.google.com.mx/books?id=gChdVKmk6OUC`.

[95] M. Lichman. Uci machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[96] Jimmy Lin and Chris Dyer. Data-intensive text processing with mapreduce. *Synthesis Lectures on Human Language Technologies*, 3(1):1–177, 2010.

[97] Bo Liu, Qingfu Zhang, and Georges GE Gielen. A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. *IEEE Transactions on Evolutionary Computation*, 18(2):180–192, 2014.

[98] Z. Liu and C.H. Xia. *Performance Modeling and Engineering*. Springer US, 2008. ISBN 9780387793610. URL `https://books.google.com.mx/books?id=Zda0LxbnCgoC`.

[99] Gabriele Lombardi, Alessandro Rozza, Claudio Ceruti, Elena Casiraghi, and Paola Campadelli. Minimum neighbor distance estimators of intrinsic dimension. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 374–389. Springer, 2011.

[100] Victoria López, Sara del Río, José Manuel Benítez, and Francisco Herrera. Cost-sensitive linguistic fuzzy rule based classification systems under the mapreduce framework for imbalanced big data. *Fuzzy Sets and Systems*, 258:5–38, 2014. ISSN 01650114. doi: 10.1016/j.fss.2014.01.015. URL `http://dx.doi.org/10.1016/j.fss.2014.01.015`.

[101] Victoria López, Sara Del Río, José Manuel Benítez, and Francisco Herrera. On the use of mapreduce to build linguistic fuzzy rule based classification systems for big data. *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2014.

[102] Rafael G Mantovani, André LD Rossi, Joaquin Vanschoren, Bernd Bischl, and André CPLF Carvalho. To tune or not to tune: recommending when to adjust svm hyper-parameters via meta-learning. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE, 2015.

[103] Oded Maron and Andrew W Moore. The racing algorithm: Model selection for lazy learners. In *Lazy learning*, pages 193–225. Springer, 1997.

[104] Vivien Marx. Biology: The big challenges of big data. *Nature*, 498(7453): 255–260, 2013.

[105] Eduardo Massad, Neli Regina Siqueira Ortega, Laecio Carvalho de Barros, and Claudio J Struchiner. *Fuzzy logic in action: Applications in epidemiology and beyond*, volume 232. Springer Science & Business Media, 2009.

[106] Mathworks. Foundations of fuzzy logic. `https://www.mathworks.com/help/fuzzy/foundations-of-fuzzy-logic.html`, 2017. URL `https://www.mathworks.com/help/fuzzy/foundations-of-fuzzy-logic.html`.

[107] Péricles BC Miranda, Ricardo BC Prudêncio, André PLF De Carvalho, and Carlos Soares. A hybrid meta-learning architecture for multi-objective optimization of svm parameters. *Neurocomputing*, 143:27–43, 2014.

[108] S.D. Mohaghegh. *Shale Analytics: Data-Driven Analytics in Unconventional Resources*. Springer International Publishing, 2017. ISBN 9783319487533. URL `https://books.google.com.mx/books?id=9gwbDgAAQBAJ`.

[109] Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 306–313. IEEE, 2002.

[110] Angel Kuri Morales and Carlos Villegas Quezada. A universal eclectic genetic algorithm for constrained optimization. In *Proceedings of the 6th European congress on intelligent techniques and soft computing*, volume 1, pages 518–522, 1998.

[111] Juliane MüLler, Christine A Shoemaker, and Robert Piché. So-mi: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. *Computers & Operations Research*, 40(5): 1383–1400, 2013.

[112] Mark EJ Newman. Complex systems: A survey. *arXiv preprint arXiv:1112.1440*, 2011.

[113] Mohammad Nozari and Carlos Manuel Soares. Metalearning to choose the level of analysis in nested data: A case study on error detection in foreign trade statistics. 2015.

[114] Randal S. Olson, William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, 10(1):36, Dec 2017. ISSN 1756-0381. doi: 10.1186/s13040-017-0154-4. URL `https://doi.org/10.1186/s13040-017-0154-4`.

[115] Carolina Osorio and Michel Bierlaire. A surrogate model for traffic optimization of congested networks: an analytic queueing network approach. Technical report, 2009.

[116] Pramudita Satria Palar and Koji Shimoyama. On multi-objective efficient global optimization via universal kriging surrogate model. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 621–628. IEEE, 2017.

[117] Priyanka Devi Pantula, Srinivas Soumitri Miriyala, and Kishalay Mitra. Simultaneous knowledge discovery and development of smart neuro-fuzzy surrogates for online optimization of computationally expensive models. In *Control Conference (ICC), 2017 Indian*, pages 260–267. IEEE, 2017.

[118] Manish Patel and Sylvia Nagl. *The role of model integration in complex systems modelling: An example from cancer biology*. Springer, 2010.

[119] Lucas M Pavelski, Myriam R Delgado, Carolina P Almeida, Richard A Gonçalves, and Sandra M Venske. Extreme learning surrogate models in multi-objective optimization based on decomposition. *Neurocomputing*, 180: 55–67, 2016.

[120] D.N. Pham and S.B. Park. *PRICAI 2014: Trends in Artificial Intelligence: 13th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2014, Gold Coast, QLD, Australia, December 1-5, 2014, Proceedings*. Lecture Notes in Computer Science. Springer International Publishing, 2014. ISBN 9783319135601. URL `https://books.google.com.mx/books?id=1vdWBQAAQBAJ`.

[121] T. Piessens and M. Steyaert. *Design and Analysis of High Efficiency Line Drivers for xDSL*. The Springer International Series in Engineering and Computer Science. Springer US, 2006. ISBN 9781402025181. URL `https://books.google.com.mx/books?id=zgHpBwAAQBAJ`.

[122] Martin Pilat and Roman Neruda. Meta-learning and model selection in multi-objective evolutionary algorithms. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 1, pages 433–438. IEEE, 2012.

[123] Rattan Priya, Bruno Feres de Souza, André LD Rossi, and André CPLF de Carvalho. Predicting execution time of machine learning tasks using metalearning. In *Information and Communication Technologies (WICT), 2011 World Congress on*, pages 1193–1198. IEEE, 2011.

[124] Haobo Qiu, Liming Chen, Chen Jiang, Xiwen Cai, and Liang Gao. Ensemble of surrogate models using sign based cross validation error. In *Computer Supported Cooperative Work in Design (CSCWD), 2017 IEEE 21st International Conference on*, pages 526–531. IEEE, 2017.

[125] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, mar 2001. also NeuroCOLT Technical Report NC-TR-1998-021.

[126] Matthias Reif, Faisal Shafait, and Andreas Dengel. Meta-learning for evolutionary parameter optimization of classifiers. *Machine learning*, 87(3):357, 2012.

[127] Zhou Ren, Junsong Yuan, and Wenyu Liu. Minimum near-convex shape decomposition. *IEEE transactions on pattern analysis and machine intelligence*, 35(10):2546–2552, 2013.

[128] CTI Reviews. *Probability and Statistics, The Science of Uncertainty*. Cram101, 2017. ISBN 9781497027473. URL `https://books.google.com.mx/books?id=E3lwBAAAQBAJ`.

[129] Dian Palupi Rini, Siti Mariyam Shamsuddin, and Siti Sophiyati Yuhaniz. Particle swarm optimization: technique, system and challenges. *International journal of computer applications*, 14(1):19–26, 2011.

[130] Alejandro Rosales-Pérez, Jesus a. Gonzalez, Carlos a. Coello Coello, Hugo Jair Escalante, and Carlos a. Reyes-Garcia. Multi-objective model type selection. *Neurocomputing*, 146:83–94, 2014. ISSN 09252312.

[131] Alejandro Rosales-Pérez, Jesus A Gonzalez, Carlos A Coello Coello, Hugo Jair Escalante, and Carlos A Reyes-Garcia. Surrogate-assisted multi-objective model selection for support vector machines. *Neurocomputing*, 150:163–172, 2015.

[132] Alejandro Rosales-Pérez, Carlos A Reyes-García, Jesus A Gonzalez, Orion F Reyes-Galaviz, Hugo Jair Escalante, and Silvia Orlandi. Classifying infant cry patterns by the genetic selection of a fuzzy model. *Biomedical Signal Processing and Control*, 17:38–46, 2015.

[133] L. Saitta and J.D. Zucker. *Abstraction in Artificial Intelligence and Complex Systems*. SpringerLink : Bücher. Springer New York, 2013. ISBN 9781461470526.

[134] Sherif Sakr, Anna Liu, and Ayman G Fayoumi. The family of mapreduce and large-scale data processing systems. *ACM Computing Surveys (CSUR)*, 46(1): 11, 2013.

[135] Javier Sánchez-Monedero, Pedro Antonio Gutiérrez, María Pérez-Ortiz, and César Hervás-Martínez. An n-spheres based synthetic data generator for supervised classification. In *International Work-Conference on Artificial Neural Networks*, pages 613–621. Springer, 2013.

[136] Samantha Corinne Sanders. Informing the use of hyper-parameter optimization through meta-learning. 2017.

[137] Eugene Santos Jr, Alex Kilpatrick, Hien Nguyen, Qi Gu, Andy Grooms, and Chris Poulin. Flexible algorithm selection framework for large scale metalearning. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 496–503. IEEE Computer Society, 2012.

[138] scikit learn.org. 3.3. model evaluation: quantifying the quality of predictions — scikit-learn 0.19.0 documentation. `http://scikit-learn.org/stable/modules/model_evaluation.html`, 2017.

[139] Yun Q Shi. *Transactions on data hiding and multimedia security III*, volume 4920. Springer, 2008.

[140] A. Singh and V. Rayapati. *Learning Big Data with Amazon Elastic MapReduce*. Packt enterprise. Packt Publishing, 2014. ISBN 9781782173441. URL `https://books.google.com.mx/books?id=-twkBQAAQBAJ`.

[141] S.S. Skiena. *The Algorithm Design Manual*. Springer London, 2009. ISBN 9781848000704. URL `https://books.google.com.mx/books?id=7XUSn0IKQEgC`.

[142] Evan R Sparks, Ameet Talwalkar, Daniel Haas, Michael J Franklin, Michael I Jordan, and Tim Kraska. Automating model search for large scale machine learning. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 368–380. ACM, 2015.

[143] Evan R Sparks, Ameet Talwalkar, Daniel Haas, Michael J Franklin, Michael I Jordan, and Tim Kraska. Automating model search for large scale machine learning. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 368–380. ACM, 2015.

[144] A. Subramanya and P.P. Talukdar. *Graph-Based Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2014. ISBN 9781627052023. URL `https://books.google.com.mx/books?id=fzKNBQAAQBAJ`.

[145] Xiao Yan Sun, Dunwei Gong, and Subei Li. Classification and regression-based surrogate model-assisted interactive genetic algorithm with individual's fuzzy fitness. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 907–914. ACM, 2009.

[146] Lisheng Sun-Hosoya, Isabelle Guyon, and Michèle Sebag. Activmetal: Algorithm recommendation with active meta learning. In *IAL 2018 workshop, ECML PKDD*, 2018.

[147] VS Sundar and Michael D Shields. Surrogate-enhanced stochastic search algorithms to identify implicitly defined functions for reliability analysis. *Structural Safety*, 62:1–11, 2016.

[148] E. Szmidt. *Distances and Similarities in Intuitionistic Fuzzy Sets*. Studies in Fuzziness and Soft Computing. Springer International Publishing, 2013. ISBN 9783319016405. URL `https://books.google.com.mx/books?id=Djq5BQAAQBAJ`.

[149] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855. ACM, 2013.

[150] Mania Tlili and Tarek M Hamdani. Big data clustering validity. In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, pages 348–352. IEEE, 2014.

[151] José María Valencia-Ramírez, Julio A Raya, José R Cedeño, Ranyart R Suárez, Hugo Jair Escalante, and Mario Graff. Comparison between genetic programming and full model selection on classification problems. In *Power, Electronics and Computing (ROPEC), 2014 IEEE International Autumn Meeting on*, pages 1–6. IEEE, 2014.

[152] Sudhir Varma and Richard Simon. Bias in error estimation when using cross-validation for model selection. *BMC bioinformatics*, 7(1):91, 2006.

[153] Gerhard Venter and Jaroslaw Sobieszczanski-Sobieski. Particle swarm optimization. *AIAA journal*, 41(8):1583–1589, 2003.

[154] Ricardo Vilalta, Christophe Giraud-Carrier, and Pavel Brazdil. Meta-learning-concepts and techniques. In *Data mining and knowledge discovery handbook*, pages 717–731. Springer, 2009.

[155] Loris Vincenzi and Paola Gambarelli. A proper infill sampling strategy for improving the speed performance of a surrogate-assisted evolutionary algorithm. *Computers & Structures*, 178:58–70, 2017.

[156] N. Werro. *Fuzzy Classification of Online Customers*. Fuzzy Management Methods. Springer International Publishing, 2015. ISBN 9783319159706. URL `https://books.google.com.mx/books?id=OXrdBgAAQBAJ`.

[157] B. Wilson, S. Wakes, and M. Mayo. Surrogate modeling a computational fluid dynamics-based wind turbine wake simulation using machine learning. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Nov 2017. doi: 10.1109/SSCI.2017.8280844.

[158] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[159] Jun won Lee and Christophe Giraud-Carrier. Predicting algorithm accuracy with a small set of effective meta-features. In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*, pages 808–812. IEEE, 2008.

[160] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. Data mining with big data. *IEEE transactions on knowledge and data engineering*, 26(1):97–107, 2014.

[161] Xuanli Lisa Xie and Gerardo Beni. A validity measure for fuzzy clustering. *IEEE Transactions on pattern analysis and machine intelligence*, 13(8):841–847, 1991.

[162] Kai Yu, Liang Ji, and Xuegong Zhang. Kernel nearest-neighbor algorithm. *Neural Processing Letters*, 15(2):147–156, 2002.

[163] Tina Yu and Dave Wilkinson. Coevolution of simulator proxies and sampling strategies for petroleum reservoir modeling. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 2677–2684. IEEE, 2009.

[164] L. A. Zadeh. Fuzzy logic. *Scholarpedia*, 3(3):1766, 2008. doi: 10.4249/scholarpedia.1766.

[165] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[166] Yang Zhao, Qinneng Xu, Yupeng Chen, and Kwok Leung Tsui. Using baidu index to nowcast hand-foot-mouth disease in china: a meta learning approach. *BMC infectious diseases*, 18(1):398, 2018.

[167] Z.H. Zhou. *Ensemble Methods: Foundations and Algorithms*. CHAPMAN & HALL/CRC MACHINE LEA. Taylor & Francis, 2012. ISBN 9781439830031. URL `https://books.google.com.mx/books?id=BDB50Ev2ur4C`.

[168] Jovia Zunic and Paul L Rosin. A convexity measurement for polygons. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Citeseer, 2002.

Part V

Appendix

Appendix A

Algorithms

In this section, the algorithms that make up the framework presented in Chapter 5 are described. This framework can be adapted to any population-based search method and therefore the search algorithm is a generic one and does not incorporate the operations of any specific method. These algorithms were developed to work under the MapReduce programming model, the Apache Spark 1.3 framework specifically, that is why, the operations to obtain an RDD from the file that contains the dataset is shown. For an easier understanding, the algorithms are remarked and, the remarks are indicated by the symbol ▷.

Algorithm 1 Get the RDD

```
1: procedure GETRDD(PathDataset,numparts)
2:     RowRDD = Load(PathDataset,numparts)
3:   ▷ Obtains RDD[String] and divide it among each node
4:     RDDcol = RowRDD.map(row → row.split(","))
5:   ▷ The split function is applied to every row in the RDD[String] and then it is transformed into
       RDD[Array[String]] (separated by columns)
6:     RDDVect = RDDcol.map(row → Vector(row.map(ColInR → ColInR.toDouble)))
7:   ▷ Every column is transformed to Double type and RDD[Vector[Double]] is obtained
8:     Return(RDDvVect)
9: end procedure
```

Algorithm 2 Generic search algorithm

```
1: procedure searchMethod(TestSet,TrainSet,NIt)
2:     Population = CreateInitialRandomPopulation()
3:
4:     fitness = MRfitness(Population,TrainSet,labels)
5:   ▷ MRfitness evaluates the performance of the solutions
6:     ItCount=0;
7:
8:     while GenCount < NIt do
9:         UpdatedPop = updatesPopulation(Population)
10:
11:        fitness = MRfitness(Population,TrainSet,labels)
12:
13:        Population = replacement(Population,UpdatedPop)
14:  ▷ Elitistic replacement by the solution fitness
15:        ItCount +=1
16:
17:    end while
18:    fModel = buildFM(Population)                          ▷ Builds final model
19:    finalFitness = evalFinalModel(fModel,TestSet)
20:
21: end procedure
```

The model evaluation stage is made up by data preparation, feature selection, and the training of a classification algorithm. In Algorithm 3 this process is shown.

Algorithm 3 Fitness calculation

```
1: procedure MRfitness(Population,TrainSet)
2:     fitness = Array[Double](Population.length)
3:     for i = 0; i < Population.length; i++ do
4:         solution = Population(i)
5:         precedence = solution(2)
6:         if precedence == 0 then
7:             RDDPrep = DataPrep(TrainSet,solution)          ▷ Performs data preparation
8:             RDDFS = FeatSelection(RDDPrep,solution)        ▷ Performs feature selection
9:             fitness(i) = Classification(RDDFS,solution)        ▷ Performs classification
10:        else
11:            RDDFS = FeatSelection(TrainSet,solution)       ▷ Performs feature selection
12:            RDDPrep = DataPrep(RDDFS,solution)             ▷ Performs data preparation
13:            fitness(i) = Classification(RDDPrep,solution)      ▷ Performs classification
14:        end if
15:    end for
16:    Return(fitness)
17: end procedure
```

The constitutive parts of Algorithm 3 are shown in the following algorithms for data preparation, feature selection and classification under the MapReduce programming model.

Algorithm 4 Data preparation

```
1: procedure DataPrep(DataSet,solution)
2:     Return(DataSet.map(row → row.toArray.map(col → Transform(col,solution))))
3:   ▷ The Transform function is applied to every column of each row in the RDD according to the
       parameters encoded in the particle
4: end procedure
```

Algorithm 5 Feature Selection

```
1: procedure FeatSelection(DataSet,solution)
2:     numFeat = solution(9)
3:     rankRDD = DataSet.map(row → RankingCalculation(row))
4:   ▷ The RankingCalculation function obtains the ranking of the features of the dataset
5:     reducedRDD = rankRDD.map(row → getF(row,numFeat))
6:   ▷ The function getF is applied to every row in rankRDD and returns a reduced dataset
7:     Return(reducedRDD)
8: end procedure
```

Algorithm 6 Classification

```
1: procedure Classification(DataSet,solution)
2:     NumFolds = 2
3:     kFolds = createFold(DataSet,NumFolds)
4:   ▷ The createFold function creates an RDD for k-Fold Cross validation
5:     error=kFolds.map {
6:       case(Training,Validation)
7:   ▷ The dataset is separated in Training and Validation partitions
8:       model = createModel(Training, solution)
9:   ▷ The createModel function creates a model using the parameters codified in the particle
10:      PredictedTargets = Validation.map(Instance → model.predict(Instance.features))
11:  ▷ Performs the predictions in the validation set
12:      accuracy= getAcc(PredictedTargets,Validation.targets)
13:  ▷ Obtains the accuracy in each fold
14:      error = 100-accuracy
15:      Return(error)
16:    }
17:    meanError=error.sum/error.length
18:    Return(meanError)
19: end procedure
```

Appendix B

Analysis of the datasets employed for the experiments

The huge quantity of data available for analysis makes it very difficult the choice of a group of datasets to represent the entire data universe. For the model selection analysis in smaller datasets, some benchmarks have been proposed as the IDA [125] benchmark and PMLB [114], just to mention a few. In order to have evidence of the variety of problems that methods proposed in this work can handle, the data properties mentioned in Section 3.4 were investigated in the employed datasets. Those datasets were obtained from the UCI repository [95]. The "Synthetic 1" and "Synthetic 2" datasets were created using the tool for synthetic datasets generation in the context of ordinal regression: "Synthetic Datasets Nspheres" provided in [135]. Despite having been developed for ordinal regression, the tool can be properly adjusted for traditional binary or multi-class problems and provides mechanisms to control overlaps and class balance. Along this chapter the datasets were analyzed to know their Intrinsic Dimension (ID) and their Shape. In Table B.0.1 the proposed datasets are shown.

Table B.0.1: Proposed datasets

Datasets	Data points	Attributes	Samples by class	Type of variables	File size
RLCP	5749111	11	(5728197;20915)	Real	261.6 MB
KDD	4856150	41	(972780;3883369)	Categorical	653 MB
Synthetic 1	200000000	3	(100000000;100000000)	Real	5.5 GB
Higgs	11000000	28	(5170877;5829123)	Real	7.5 GB
Synthetic 2	49000002	30	(24500001;24500001)	Real	12.7 GB
Epsilon	500,000	2000	(249778;250222)	Real	15.6 GB

B.1 Topology of the datasets

In order to determine the topology of the datasets, analyses were performed employing the alpha-shape algorithm presented in Edelsbrunner et al. [1983] and with the calculation of the Betti-numbers from [42]. The Betti numbers represent the number of $n-$D holes in a space, and they are an important topological feature. Betti numbers are determined by analyzing the simplicial complex like simplices. The simplices or simplex are spatial objects classified according to their spatial dimensions. For each dimension, a minimal object exists, the $0-$simplices are represented by nodes, $1-$simplices stand for edges, $2-$simplices stand for triangles, $3-$simplices are tetrahedrons. A simplicial complex is a finite collection of simplices and its faces [56]. Regarding Betti-numbers, Betti 0 represents the number of connected components. If everything is connected by an edge, Betti 0 is 1. Betti 1 represents the number of holes in a surface. For example, a $1-$Complex can be represented by a triangle. It has 3 edges and 1 hole. Betti 2 represents voids (Empty volumes) [42]. Taking into account the definition of shape in a dataset provided in section 3.4 and the aforementioned techniques employed to analyze the datasets, it can be seen that if there are holes in a dataset we can conclude that is a non-convex dataset, if there are holes, then is a convex dataset. The analysis of a torus is provided in Figure B.1.1 as example of a non-convex dataset. It can be seen that Betti $0 = 1$ meaning that there is a single component, Betti $1 = 1$ stands for the central hole in the dataset, and finally Betti $2 = 1$ stands for the cavity in the interior of the torus. On the other hand, the alpha-shape algorithm was used to find the contour and to draw a polygon of the approximated shape of the dataset. A simple visual analysis allows us to find that there is a single component with a hole in the middle (coinciding with B1=1 and B2=1), therefore is a non-convex dataset. The following figures represent the class distribution (represented as blue crosses and red dots), the alpha-shapes and the Betti numbers of the datasets in Table B.0.1.

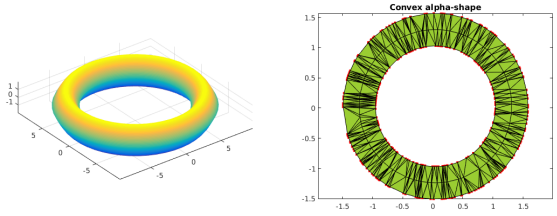


Figure B.1.1: A Torus with its alpha-shape and Betti numbers: $B0$=1, $B1$=2, $B2$=1.

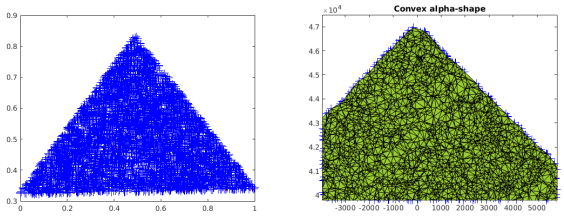


Figure B.1.2: Class distributions, alpha-shape and Betti numbers of the RLCP dataset. $B0$=1, $B1$=0, $B2$=18060.

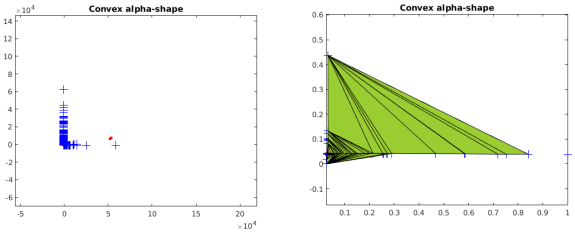


Figure B.1.3: Class distributions, alpha-shape and Betti numbers of the KDD dataset. $B0$=1, $B1$=0, $B2$=18060.

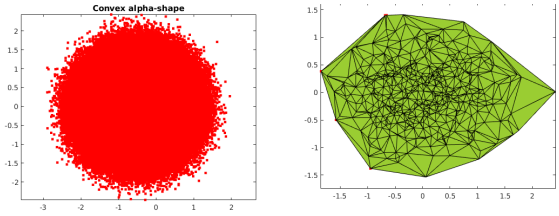


Figure B.1.4: Class distributions, alpha-shape and Betti numbers of the Synthetic 1 dataset. $B0$=1, $B1$=0, $B2$=18424.

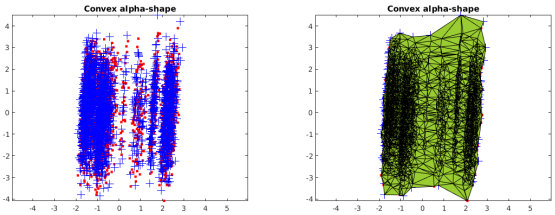


Figure B.1.5: Class distributions, alpha-shape and Betti numbers of the Higgs dataset. $B0$=1, $B1$=0, $B2$=18424.

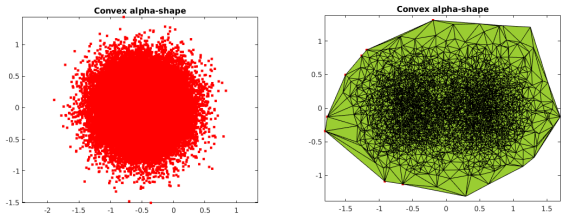


Figure B.1.6: Class distributions, alpha-shape and Betti numbers of the Synthetic 2 dataset. $B0$=1, $B1$=0, $B2$=18424.

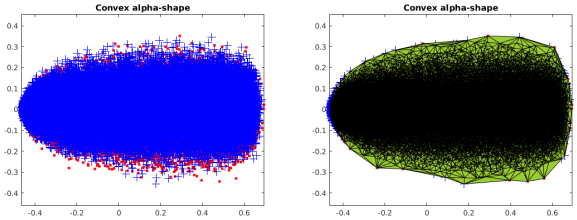


Figure B.1.7: Class distributions, alpha-shape and Betti numbers of the Epsilon dataset. $B0$=1, $B1$=0, $B2$=18377.

To this point, it can be seen that the topology of all the datasets is convex, because their Betti number values are the same in the dimensions $B0$ and $B1$ (number of components and holes). The dimension $B2$ is employed to find holes in higher dimensions and is not useful for this analysis. On the other hand, a visual analysis of the polygons provided by the alpha-shape algorithm shows a single component and no holes in all figures, therefore those datasets are convex. In order to explore the capabilities of the algorithms proposed in this work, new synthetic datasets with non-convex topology and with separated components were created. To ensure its non-convex topology, the datasets were created with two dimensions. Referring to the other characteristics, all the datasets have equally distributed samples by class (5000 and 5000) with a total of 10,000 samples and 1 GB of file size. In the following figures, their class distribution and alpha-shape are shown.

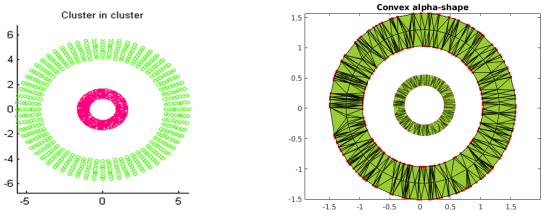


Figure B.1.8: Class distributions and alpha-shape of the non-convex 1 dataset.

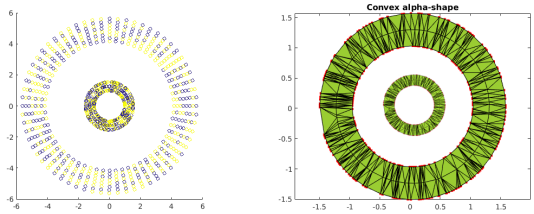


Figure B.1.9: Class distributions and alpha-shape of the non-convex 2 dataset.

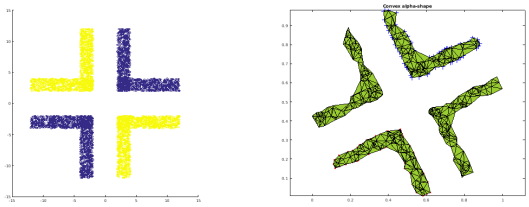


Figure B.1.10: Class distributions and alpha-shape of the non-convex 3 dataset.

B.2 Intrinsic dimension analysis

The ID is the minimum number of parameters needed to represent the data without information loss. The ID of the employed datasets was estimated with the “Minimum neighbor distance estimator" (MNDE) [99] and the “Dimensionality from angle and norm concentration" (DANCO) estimator [22]. The importance of the estimation of the “id” of each dataset is to ensure that each dataset represents a different computational problem and, therefore, that proposed algorithms have the capability to deal with a wide range of problems and in the context of this work also with datasets of different sizes. In the Table B.2.1, the calculated intrinsic dimension using

the aforementioned estimators is shown. In the case of the non-convex datasets, this dimension is not shown because they were created with two dimensions in order to ensure their non-convex topology.

Table B.2.1: Intrinsic dimension of the datasets.

Datasets	MNDE	DANCO
RLCP	2	2
KDD	1	1
Synthetic 1	3	3
Higgs	12	15
Synthetic 2	22	28
Epsilon	160	78

Table B.2.1 shows that all evaluated datasets have a wide range of values from 1 to 160 or 78, depending on the estimator employed. Considering the evidence provided by the ID analysis, it can be argued that all employed datasets are different among each other. Regarding the Shape analysis, although all datasets in Table B.0.1 are of convex shape, the new datasets created (non-convex 1 to 3) provide variety to our group for data analysis. These two properties in combination provide us evidence that evaluated datasets can be employed to test the algorithms proposed in this work because they represent a variety of problems to analyze.

B.3 Summary

This chapter presented the datasets that were employed to test our algorithms. In order to provide evidence of the variety of problems that these datasets represent, two important analysis were performed, the ID analysis and the Shape analysis. The results obtained showed that all datasets represent a variety of different problems. Concerning the data size, the datasets are of incremental size starting from 261 MB to 15.6 GB. Taking into account all these factors, the employed datasets can be used to test the capability of the proposed algorithms to deal with datasets of a wide range of size, different intrinsic dimensions and different topologies.

Acronyms

FCM Fuzzy C-Means. 61–63, 84, 97, 135

FL Fuzzy Logic. 32, 33, 135

FML-FMS Full Meta-learning to assist in the Full Model Selection problem. 92–96, 98, 106, 135

FMS Full Model Selection. 3–6, 8, 11, 12, 14, 15, 20, 22, 23, 25, 28–30, 37, 39–43, 45, 46, 49, 50, 53–60, 63, 64, 71, 76–82, 84, 91, 94, 96–98, 101–107, 135

FR-PSMS Fuzzy Rules based proxy model for PSMS. 63, 64, 71, 72, 74–76, 78, 82, 86–88, 91–96, 98, 105, 106, 135

GA Genetic Algorithm. 30, 37, 39–44, 46, 48–54, 60, 76, 102, 135

ID Intrinsic Dimension. 28, 29, 51, 53, 71, 87, 130, 134, 135

K-NN K Nearest Neighbors. 6, 12, 32, 39, 50–53, 74–76, 79–81, 83, 84, 86, 87, 89–91, 95–98, 105, 106, 135

M-PSMS Main PSMS. 60, 135

PS-FCM FCM optimized through PSO. 84–92, 96–98, 105–107, 135

PSMS Particle Swarm Model Selection. 45, 48–54, 60, 63–65, 68, 70–74, 76, 77, 81–83, 87, 88, 91, 96, 98, 102, 104–106, 135

PSMS-MLP PSMS assisted by a Multilayer Perceptron. 63, 65, 67–70, 72, 76, 135

PSO Particle Swarm Optimization. 30–32, 37, 39–45, 59, 63, 84, 102, 135

S-PSMSClass PSMS assisted by proxy models constructed with classification algorithms, the acronym. 60, 63–65, 68, 70–72, 74, 76, 135

S-PSMSReg the search assisted by proxy models that uses regression algorithms,. 60, 63, 65, 68–70, 72, 76, 135