UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Use of meta-learning for hyperparameter tuning of classification problems

Rafael Gomes Mantovani

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Rafael Gomes Mantovani

Use of meta-learning for hyperparameter tuning of classification problems

Doctoral dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho
Co-advisor: Prof. Dr. Joaquin Vanschoren

USP – São Carlos
July 2018

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

G633u

Gomes Mantovani, Rafael
Use of meta-learning for hyperparameter tuning of classification problems / Rafael Gomes Mantovani; orientador André Carlos Ponce de Leon Ferreira de Carvalho; coorientador Joaquin Vanschoren. -- São Carlos, 2018.
155 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2018.

1. Meta-learning. 2. Hyperparameter tuning. 3. Classification problems. I. Carlos Ponce de Leon Ferreira de Carvalho, André , orient. II. Vanschoren, Joaquin, coorient. III. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Rafael Gomes Mantovani

Uso de meta-aprendizado para ajuste de hiper-parâmetros em problemas de classificação

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho
Coorientador: Prof. Dr. Joaquin Vanschoren

USP – São Carlos
Julho de 2018

ACKNOWLEDGEMENTS

Firstly, I would like to thank everyone who contributed to this work, my parents, family, friends from Brazil, friends from abroad, and colleagues. In particular, my former professor Maria Angélica, who believed in me when I was giving up the academic career and indicated me to professor André. It all started thanks to you.

I lovely thank my fiancée Larissa, not just once but as many times as I can. During this Ph.D. process you've been by my side at all the times I've needed. I could not have done it without your support and love.

I also especially thank professor André for the opportunity to work with him, for his contributions to this research, for being my advisor and friend. It was a pleasure be your Ph.D. student in the last five years. I wish we continue to collaborate for many years.

I thank professor Joaquin Vanschoren for his contributions to the research carried out during this thesis, for being my co-advisor, and especially for receiving me in Eindhoven during a whole year. This experience changed me so much. I am happy I've met you and wish we can work together on many new research projects.

I also thank my friends and professors Ricardo Cerri, André Rossi, Sylvio Barbon and Tomáš Horváth for their contributions during this Ph.D., and for the opportunity to keep working with them in related projects.

I thank my friend Paulo Pisani. We have worked as Ph.D. students, but now I hope we can collaborate more soon. Thank you for all the support and help to finish this thesis.

Finally, I would like to thank São Paulo Research Foundation (FAPESP), grants #2012/23114-9 and #2015/03986-0, for the financial support of this thesis.

“What we know is a drop,
what we don’t know is an ocean.”
(Isaac Newton)

ABSTRACT

MANTOVANI, R. G. **Use of meta-learning for hyperparameter tuning of classification problems**. 2018. 155 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

Machine learning solutions have been successfully used to solve many simple and complex problems. However, their development process still relies on human experts to perform tasks such as data preprocessing, feature engineering and model selection. As the complexity of these tasks increases, so does the demand for automated solutions, namely Automated Machine Learning (AutoML). Most algorithms employed in these systems have hyperparameters whose configuration may directly affect their predictive performance. Therefore, hyperparameter tuning is a recurring task in AutoML systems. This thesis investigated how to efficiently automate hyperparameter tuning by means of Meta-learning. To this end, large-scale experiments were performed tuning the hyperparameters of different classification algorithms, and an enhanced experimental methodology was adopted throughout the thesis to explore and learn the hyperparameter profiles for different classification algorithms. The results also showed that in many cases the default hyperparameter settings induced models that are on par with those obtained by tuning. Hence, a new Meta-learning recommender system was proposed to identify when it is better to use default values and when to tune classification algorithms for each new dataset. The proposed system is capable of generalizing several learning processes into a single modular framework, along with the possibility of assigning different algorithms. Furthermore, a descriptive analysis of model predictions is used to identify which data characteristics affect the necessity for tuning in each one of the algorithms investigated in the thesis. Experimental results also demonstrated that the proposed recommender system reduced the time spent on optimization processes, without reducing the predictive performance of the induced models. Depending on the target algorithm, the Meta-learning recommender system can statistically outperform the baselines. The significance of these results opens a number of new avenues for future work.

Keywords: Meta-learning, Hyperparameter tuning, Classification problems.

RESUMO

MANTOVANI, R. G. **Uso de meta-aprendizado para ajuste de hiper-parâmetros em problemas de classificação**. 2018. 155 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

Soluções de aprendizado de máquina tem sido cada vez mais usadas com sucesso para resolver problemas dos mais simples aos complexos. Entretanto, o processo de desenvolvimento de tais soluções ainda é um processo que depende da ação de especialistas humanos em tarefas como: pré-processamento dos dados, engenharia de *features* e seleção de modelos. Consequentemente, quando a complexidade destas tarefas atinge um nível muito alto, há a necessidade de soluções automatizadas, denominadas por Aprendizado de Máquina automatizado (AutoML). A maioria dos algoritmos usados em tais sistemas possuem hiper-parâmetros cujos valores podem afetar diretamente o desempenho preditivo dos modelos gerados. Assim sendo, o ajuste de hiper-parâmetros é uma tarefa recorrente no desenvolvimento de sistems de AutoML. Nesta tese investigou-se a automatização do ajuste de hiper-parâmetros por meio de Meta-aprendizado. Seguindo essa linha, experimentos massivos foram realizados para ajustar os hiper-parâmetros de diferentes algoritmos de classificação. Além disso, uma metodologia experimental aprimorada e adotada ao lngo da tese perimtiu identificar diferentes perfis de ajuste para diferentes algoritmos de classificação. Entretanto, os resultados também mostraram que em muitos casos as configurações *default* destes algoritmos induziram modelos mais precisos do que os obtidos por meio de ajuste. Assim, foi proposto um novo sistema de recomendação baseado em Meta-learning para identificar quando é melhor realizar o ajuste de parâmetros para os algoritmos de classificação ou apenas usar os valores *default*. O sistema proposto é capaz de generalizar várias etapas do aprendizado em um único framework modular, juntamente com a possibilidade de avaliar diferentes algoritmos de aprendizado de máquina. As análises descritivas das predições obtidas pelo sistema indicaram quais características podem ser responsáveis por determinar quando o ajuste se faz necessário para cada um dos algoritmos investigados na tese. Os resultados também demonstraram que o sistema recomendador proposto reduziu o tempo gasto com a otimização mantendo o desempenho preditivo dos modelos gerados. Além disso, dependendo do algoritmo de classificação modelado, o sistema foi estatisticamente superior aos baselines. A significância desdes resultados abre um novo número de oportunidades para trabalhos futuros.

Palavras-chave: Meta-aprendizado, Ajuste de hiper-parâmetros, Problemas de classificação.

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

Acc	Accuracy
ANN	Artificial Neural Network
AT	Active Testing
AUC	Area Under the ROC curve
AutoML	Automated Machine Learning
CASH	Combined Algorithm Selection and Hyperparameter Optimization
CD	Critical Difference
CHAID	Chi-squared Automatic Interaction Detector
CV	Cross-Validation
DF	Default values
DL	Deep Learning
DS	Decision Stump
DT	Decision Tree
fANOVA	Functional ANOVA
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
GS	Grid Search
HPs	hyperparameters
Irace	Iterated F-Race
kNN	k-Nearest Neighbors
LMT	Logistic Model Tree
LOO	Leave One Out
MAD	Mean Absolute Deviation
MAP	Mean Average Precision
ML	Machine Learning
MLP	Multilayer Perceptron
MtL	Meta-learning
NAE	Normalized Absolute Error
NBTree	Naïve-Bayes Tree
NMSE	Normalized Mean Squared Error

OpenML	Open Machine Learning
PILS	Iterated Local Search in Parameter Configuration Space
PMCC	Pearson Product Moment Correlation Coefficient
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
RF	Random Forest
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
ROAR	Random Online Adaptive Racing
RS	Random Search
SFS	Sequential Forward Selection
SMBO	Sequential Model-based Optimization
SMOTE	Synthetic Minority Over-sampling Technique
SVM	Support Vector Machine
TAF	Transfer Acquisition Function
TS	Tabu Search
Xgboost	Extreme Gradient Boosting

CONTENTS

CHAPTER

1

INTRODUCTION

With the expansion of data generation, Machine Learning (ML) has achieved high popularity in recent years. Progress in this area allowed a large uptake of ML solutions by the industry. Building ML solutions often involve several tasks, which include comparing many algorithms, optimizing their "hyperparameters (HPs)", and exploring different feature representations. However, very rarely an algorithm works satisfactorily at the first try. As a consequence, a large amount of time is spent tuning HPs to generate models with high predictive performance. Thus, as the complexity of these tasks increases, it demands ML solutions that can be easily used, without the need for human intervention, which have been named "Automated Machine Learning (AutoML)" (AutoML) solutions.

AutoML focuses on users with little or no expert knowledge in ML, also providing new tools and functionalities to ML experts to advance the state-of-the-art in the area. It includes many topics related with ML, such as Bayesian optimization (SNOEK; LAROCHELLE; ADAMS, 2012), Transfer learning and Meta-learning (MtL) (BRAZDIL *et al.*, 2009).

The use of AutoML may relieve data scientists from the repetitive and time-consuming steps in the design of a full data science solution, including the algorithm design and optimization tasks. Hence, they can allocate their time on more difficult tasks. Thus, the goal of AutoML is to facilitate and increase the use of data science techniques by non-experts and to support data scientists in their work, not to replace them (FEURER *et al.*, 2015). In this thesis, AutoML is considered for predictive tasks, in particular, supervised classification tasks using the Support Vector Machine (SVM) and Decision Tree (DT) induction algorithms. However, the issues investigated in this thesis can be easily extended to other tasks.

Most of the ML algorithms employed in AutoML systems have HPs, free parameters/options that need to be configured beforehand, and which directly affect the predictive performance of the induced models. A recurrent subtopic of research on AutoML systems is the HP tuning of ML algorithms.

Many ML applications use default HP settings suggested by libraries and tools implementing these algorithms, even though many studies have shown that their predictive performance greatly depends on using suitable HP values (THORNTON *et al.*, 2013; FEURER *et al.*, 2015). In predictive tasks, suitable HP values are those that lead to the induction of models with high predictive accuracy. In early works, these values were tuned according to previous experiences of data scientists by trial and error. To overcome this problem, optimization techniques are often employed to automatically search for suitable HP settings (BERGSTRA *et al.*, 2011; BARDENET *et al.*, 2013).

Although HP tuning may lead to more accurate models, the optimization process for finding the most promising setting is usually very time-consuming. Depending on the data problem being solved and the number of HPs to be optimized, tuning may become computationally unfeasible. A recent alternative to overcome these limitations has been the use of MtL (GOMES *et al.*, 2012; REIF *et al.*, 2014; FEURER; SPRINGENBERG; HUTTER, 2015).

Different MtL systems have been proposed to select (ALI; SMITH-MILES, 2006), rank (SUN; PFAHRINGER, 2013), or predict (REIF *et al.*, 2014) the best HP settings of ML algorithms. Other works combine MtL with optimization techniques, where MtL recommends the initial points for optimization techniques to start the search for suitable HP settings [1]. The use of MtL by itself or in combination with optimization techniques tends to be more computationally efficient when compared with the use of only optimization techniques. However, it must be observed that the quality of MtL recommendations directly depends on the quality of the meta-examples used (GOMES *et al.*, 2012; FEURER *et al.*, 2015).

1.1 Problem investigated and hypothesis assumed

The AutoML area is relatively new, and there are still many questions to be addressed. This fact, and the emerging attention it has attracted from important research groups (FEURER *et al.*, 2015; KOTTHOFF *et al.*, 2016) and large companies (GOOGLE, 2018; RAPIDMINER, 2018), highlights the importance of new studies in the area. The problem investigated in this thesis **is the automation of the HP tuning in the AutoML context**. An important aspect for successful AutoML is to have an automatic and robust HP tuning system, which emphasizes the importance of the problem investigated.

Some ML algorithms are more sensitive than others to HP tuning (BERGSTRA; BENGIO, 2012). Besides, different tuning methods may find suitable HP settings from different hyperspace regions. However, the obtained predictive performance improvement may not be statistically better than simply adopting the default HP values. This indicates that each algorithm has its own *HP profile*. In this thesis, this term refers to a set of characteristics associated with the HP tuning of a ML algorithm, such as: its sensitivity to the tuning process, the number

[1] Related studies will be discussed in more details in Chapter 3.

of evaluations required to find suitable settings; the most suitable optimization technique for different budget sizes; and which HPs are the most relevant for the tuning. This profile leads to the first hypothesis investigated in this thesis:

H1: each classification algorithm has its own *HP profile*, which can impact the results obtained by tuning techniques. Further, it would be possible to identify the most sensitive HPs and perform a more efficient model tuning.

Previous works have pointed out that HP tuning is not mandatory to achieve good predictive performance for some classification tasks (RIDD; GIRAUD-CARRIER, 2014). Therefore, when computational resources are limited, a commonly adopted alternative is to use the default HP values suggested by ML packages and tools. The use of default values largely reduces the overall computational cost, but, depending on the dataset, it can result in models whose predictive performance is significantly worse than models produced by using HP tuning. The ideal situation would be to recommend the best alternative, default or tuned HP values, for each new dataset.

Based on the recent success of the MtL applications, the second hypothesis studied in this work is:

H2: by exploiting the *HP profile*, meta-learning can support the identification of situations where HP tuning can improve the predictive performance of ML algorithms;

1.2 Objective

The main objective of this thesis is **to investigate and develop efficient MtL-based recommender systems able to recommend the best HP values for classification algorithms**. This objective is accomplished by proposing a recommender system able to predict when HP tuning is expected to lead to models with statistically significant improvements after identifying the HP profile of the ML algorithm. The experiments performed in this thesis can be generalized for several algorithms. Thus, the results from the current research can be used in the future to develop a unified framework to be added to ML tools.

1.3 Main contributions to the research area

The thesis deals with several aspects of HP tuning systems, and makes contributions to ML, MtL and HP tuning of ML algorithms. The main contributions of this thesis can be summarized as:

- A clear comparison of different HP tuning techniques for Support Vector Machines (SVMs) and Decision Tree (DT) induction algorithms on an unprecedented scale. The use of SVMs as study case is justified by their sensitivity to tuning, while DT algorithms were selected due to the lack of investigations in the literature;

- We unveil when is better to use HP tuning and when it is sufficient to use default HPs;
- Development of a meta-learning framework to predict when default HP values provide accurate models, saving computational cost that would otherwise be wasted on optimization with no significant improvement;
- A comprehensive analysis of the effect of HPs on the predictive performance of the induced models and the relationship between them;
- A comparison of the effectiveness of different meta-features sets and preprocessing methods for MtL;
- Reproducibility of experiments: all code and experimental results are available to reproduce experiments, analyses, and allow further analysis. Code is available at `Github` repositories, while experimental results are available on `Open Machine Learning (OpenML)` (VANSCHOREN *et al.*, 2014).

1.4 Thesis organization

Throughout the thesis, several HP tuning scenarios for classification algorithms are experimentally evaluated and discussed. An overview of the covered topics and their relationship is shown in Figure 1:

- Chapter 2 introduces the HP tuning problem and the techniques often applied to deal with this problem;
- Chapter 3 presents MtL concepts and definitions, followed by a review of related work on MtL for HP tuning of classification algorithms. Several questions regarding the design of experiments for MtL are also discussed;
- Chapter 4 briefly describes the common experimental methodology aspects adopted throughout the thesis: classification algorithms, tuning techniques, resampling strategies, statistical tests, and ML tools used to allow the reproducibility of experiments and analyses;
- Chapter 5 analyzes the *HP profile* of SVMs, while
- Chapter 6 analyzes the *HP profile* of DT induction algorithms;
- Chapter 7 investigates the use of SVM and DT generated meta-knowledge by a MtL recommender system to predict in which situations the HP tuning will lead to a significant improvements in predictive performance;
- Finally, Chapter 8 presents the conclusions of this thesis, and discusses final considerations, limitations, and suggestions for future studies.

Thesis Structure

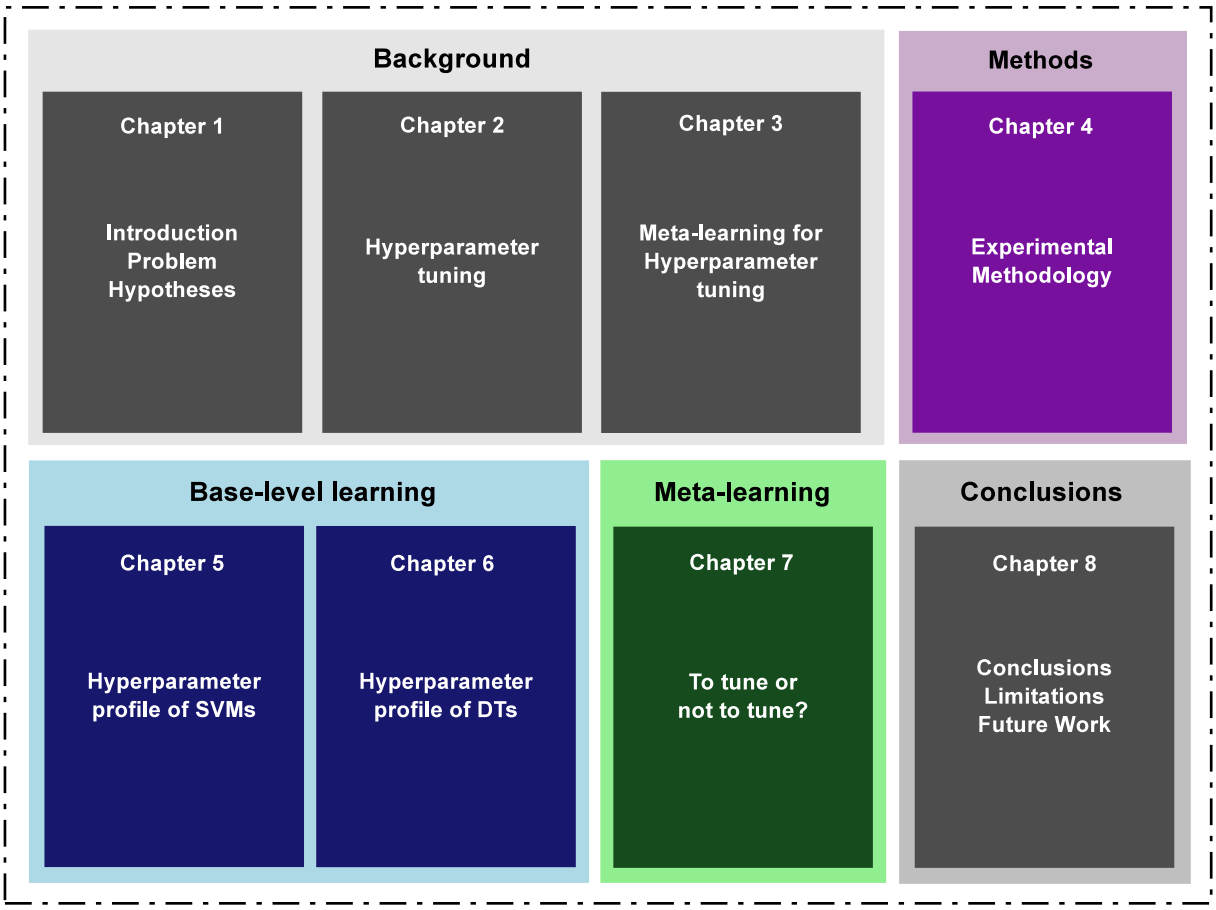


Figure 1 – Thesis structure.

The full code generated for the experiments in this thesis is publicly available to reproduce the reported analysis - and extend it to other classifiers. Links are available in Section 4.8, where the methodology is described. All the experiments are also available at OpenML (VANSCHOREN *et al.*, 2014)[2].

[2] OpenML will be described in Section 4.6

CHAPTER

2

HYPERPARAMETER TUNING

In a supervised classification problem, ML algorithms are trained with labeled data to identify which label a new observation belongs to. These algorithms have free "hyperparameters" (HPs) whose values directly affect their predictive performance. Finding a suitable HP setting requires specific knowledge, intuition, and mostly: trial and error experiments.

Figure 2 illustrates the general HP tuning schema. In many situations, the HP tuning is carried out manually by some expert, progressively refining a grid of values over the desired space. However, literature provides a set of techniques ranging from simple to complex, able to accomplish this task. From a theoretical point of view, selecting the ideal HP values requires an exhaustive search over all possible subsets of HPs. Depending on the number and type of the

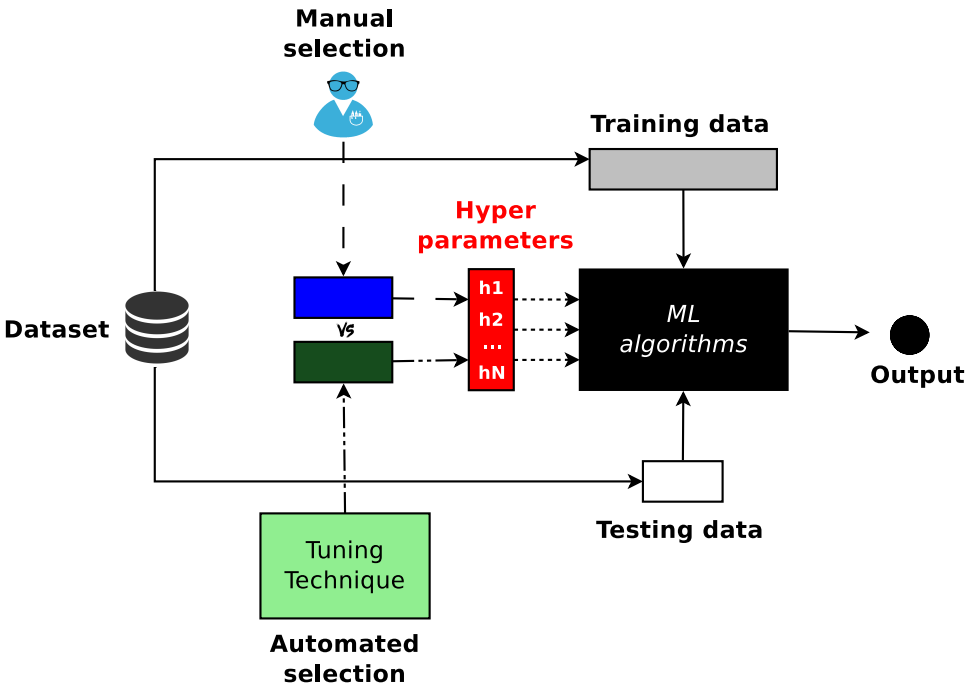


Figure 2 – General hyperparameter tuning schema.

HPs, this task becomes impractical. Therefore, for the ML community, it is often accepted to search reduced HP space instead of the complete space (BERGSTRA; BENGIO, 2012).

According to Bardenet *et al.* (2013), the use of automated techniques for HP tuning presents benefits, such as:

- Freeing the users from the task of manually selecting the HP values, thus they can concentrate efforts in other aspects referring to the use of the algorithm itself;
- Reducing the computation time spent for training and use of the algorithm; and
- Improving the predictive performance of the induced models.

2.1 Definition

The HP tuning process is usually treated as a black-box optimization problem whose objective function is associated with the predictive performance of the model induced by a ML algorithm. Formally it is defined as follows:

Definição 1. Let $H = H_1 \times H_2 \times \cdots \times H_k$ be the HP space for an algorithm $a \in A$, where A is the set of ML algorithms. Each H_i represents a set of possible values for the ith HP of a ($i \in \{1, \ldots, k\}$) and can be usually defined by a set of constraints.

Definição 2. Additionally, let D be a set of datasets where $\mathbf{D} \in D$ is a dataset from D. The function $f : A \times D \times H \rightarrow \mathbb{R}$ measures the predictive performance of the model induced by the algorithm $a \in A$ on the dataset $\mathbf{D} \in D$ given a HP configuration $\mathbf{h} = (h_1, h_2, \ldots, h_k) \in H$. Without loss of generality, higher values of $f(a, D, h)$ mean higher predictive performance.

Definição 3. Given $a \in A$, H and $\mathbf{D} \in D$, together with the previous definitions, the goal of a HP tuning task is to find $\mathbf{h}^\star = (h_1^\star, h_2^\star, \ldots, h_k^\star)$ such that

$$\mathbf{h}^\star = \underset{\mathbf{h} \in H}{arg\,max}\, f(a, \mathbf{D}, \mathbf{h}) \tag{2.1}$$

The optimization of the HP values can be based on any performance measure f, which can even be defined by multi-objective criteria. Further aspects can make the tuning more difficult, like:

- HP configurations that lead to a model with high predictive performance for a given dataset may not lead to high predictive performance for other datasets;
- HP values often depend on each other[1]. Hence, independent tune of HPs may not lead to a good set of HP values;
- the exhaustive evaluation of several HP configuration can be very time-consuming.

[1] This is the case of SVMs (BEN-HUR; WESTON, 2010).

2.2 Tuning techniques

In addition to simple approaches, like Grid Search (GS) and Random Search (RS), in the last decades, population-based optimization techniques have been successfully used for HP tuning of classification algorithms (BERGSTRA *et al.*, 2011; LÓPEZ-IBÁÑEZ *et al.*, 2011; SNOEK; LAROCHELLE; ADAMS, 2012; BARDENET *et al.*, 2013). When applied to tuning, these techniques (iteratively) build a *population* $P \subset H$ of HP settings for which $f(a, \mathbf{D}, \mathbf{h})$ are being computed for each $\mathbf{h} \in P$. By doing so, they can simultaneously explore different regions of a search space. There are various population-based HP tuning strategies, which differ in how they update P at each iteration. Some of these methods will be discussed in the following Subsections.

2.2.1 Grid Search

Grid Search (GS) is the most straightforward approach to HP tuning. It is an exhaustive search through a subset of the HP space to select the best of a family of models, parametrized by a grid of values. Its simple execution is illustrated in Algorithm 1.

Algorithm 1 – GS pseudocode

1: $Best_{global} \leftarrow$ `NULL`
2: **for** each HP $h_i \in H$ **do**
3: Sample a set of values $V_i = v_{i1}, v_{i2}, \ldots, v_{iN}$ from h_i
4: **end for**
5: **for** each $\lambda \in (h_1, \ldots, hk) \in V_1 \times \cdots \times V_k$ **do**
6: $Best_{Local} \leftarrow f(a, D, \lambda)$
7: **end for**
8: $Best_{global} \leftarrow max\{Best_{Local}\}$
9: **return** $Best_{global}$

GS may be a good choice for spaces with few HPs. However, it suffers from the dimensionality of the problem: the higher the number of HPs evaluated, the higher the computational cost required to solve the problem. Even so, the manual selection of the grid values that precede the search may provide some tips on how the HP space surface behaves (BERGSTRA; BENGIO, 2012).

2.2.2 Random Search

Random Search (RS) is a simple technique that performs random trials in a search space. Its use can reduce the computational cost when there is a large number of possible settings being investigated (ANDRADOTTIR, 2015). Usually, RS performs its search iteratively in a predefined number of iterations. $P(i)$ is extended (updated) by a randomly generated HP setting $\mathbf{h} \in H$ in each (ith) iteration of the HP tuning process. RS has obtained efficient results in optimization for

Deep Learning (DL) algorithms (BERGSTRA; BENGIO, 2012; BARDENET *et al.*, 2013). The RS simple workflow is described in Algorithm 2.

Algorithm 2 – RS pseudocode

```
t ← 1
Best_global ← NULL
while Stopping criteria not satisfied do
    Generate a population P(t) randomly
    for each p_i ∈ P(t) do
        f_{p_i} ← f(a, D, p_i)
    end for
    Best_Local ← max{f_p}
    if Best_local > Best_global then
        Best_global ← Best_local
    end if
    t ← t + 1
end while
return Best_global
```

2.2.3 Sequential Model Based Optimization

Sequential Model-based Optimization (SMBO) (SNOEK; LAROCHELLE; ADAMS, 2012; BROCHU; CORA; FREITAS, 2010) is a sequential method that starts with a small initial population $P(0) \neq \emptyset$ which, at each new iteration $i > 0$, is extended by a new HP configuration $\mathbf{h}'$, such that the expected value of $f(a, \mathbf{D}, \mathbf{h}')$ is maximal according to an induced meta-model $\hat{f}$ approximating f on the current population $P(i-1)$. In the experiments reported in Bergstra *et al.* (2011), Snoek, Larochelle and Adams (2012), Bergstra, Yamins and Cox (2013), SMBO performed better than GS and RS and matched or outperformed state-of-the-art techniques in several HP optimization tasks. The general SMBO framework is presented in the Algorithm 3.

Algorithm 3 – SMBO pseudocode

```
t ← 0
Generate initial population P(0)
Evaluate the current population P(t)
while Stopping Criteria not satisfied do
    t ← t + 1
    Fit a Surrogate model M(t) to all points in P(t)
    An infill criteria I proposes new n points N(t) from M(t)
    The new points N(t) are evaluated and added to P(t)
end while
Best_global ← best individual in P(t)
return Best_global
```

After evaluating the initial points, the technique fits a surrogate regression model on the available data. Then, it queries the model for a new candidate HP setting using an acquisition

function (or *infill criteria*). This function searches for a point at the hyperspace which yields the best infill value (for example, the expected improvement regarding performance) and then adds this value to the population for the next iteration.

2.2.4 Genetic Algorithm

Bio-inspired techniques, such as a Genetic Algorithm (GA), based on natural processes, have also been largely used for HP tuning (GOMES *et al.*, 2012; FRIEDRICHS; IGEL, 2005; KALOS, 2005). In these techniques, the initial population $P(0) = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{n_0}\}$, generated randomly or according to background knowledge, is changed in each iteration according to operators based on natural selection and evolution. The GA general pseudocode is presented in the Algorithm 4.

Algorithm 4 – GA pseudocode

```
t ← 0
Generate initial population P(0)
Evaluate the current population P(t)
while Stopping criteria not satisfied do
    t ← t + 1
    Select population P(t) from P(t − 1)
    Apply crossover operators in P(t)
    Apply mutation operators in P(t)
    for each new individual i in the current population P(t) do
        Evaluate individual i fitness
    end for
end while
Best_global ← best individual i from P(t)
return Best_global
```

2.2.5 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a bio-inspired technique relying on the swarming and flocking behaviors of animals (SIMON, 2013). In case of PSO, each particle $\mathbf{h} \in P(0)$ is associated with its position $\mathbf{p} = (h_1, \ldots, h_k) \in H$ in the search space H, a velocity $\mathbf{v}_h \in \mathbb{R}^k$ and also its so far best found position $\mathbf{b}_h \in H$. During iterations, the movements of each particle are changed according to its so far best-found position as well as the so far best-found position $g_{Best} \in H$ of the entire swarm (recorded through the computation). The Algorithm 5 outlines PSO behavior.

2.2.6 Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs) (HAUSCHILD; PELIKAN, 2011) are a family of algorithms that lie on the boundary of GA and SMBO by combining the advantages of

Algorithm 5 – PSO pseudocode

```
1:  t ← 0
2:  Generate initial population P(0)
3:  Evaluate the current population P(t)
4:  while Stopping Criteria not satisfied do
5:      t ← t + 1
6:      Update weights
7:      Select p_Best for each particle h
8:      Select g_Best from P(t − 1)
9:      Update velocity v for each particle h
10:     Update position b for each particle h
11:     Evaluate the current population of particles P(t)
12: end while
13: Best_global ← g_Best from P(t)
14: return Best_global
```

both approaches such that the search is guided by iteratively updating an explicit probabilistic model of promising candidate solutions. In other words, the implicit crossover and mutation operators used in GA are replaced by an explicit probabilistic model M. The Algorithm 6 describes EDA pseudocode. EDAs has some advantages when compared with other meta-heuristics: over the iterations, it reduces considerably the number of functions evaluated, and it is not sensitive to its parameters as GAs.

Algorithm 6 – EDA pseudocode

```
1:  t ← 0
2:  Generate initial population P(0)
3:  Evaluate the current population P(t)
4:  while Stopping Criteria not satisfied do
5:      Select population of promising solutions S(t) from P(t)
6:      Build a probabilistic model M(t) from S(t)
7:      Sample M(t) to generate new candidate solutions O(t)
8:      Incorporate O(t) into P(t)
9:      t ← t + 1
10: end while
11: Best_global ← P(t)
12: return Best_global
```

2.2.7 Iterated F-Race

The Iterated F-Race (Irace) (BIRATTARI *et al.*, 2010) technique was designed for algorithm configuration and optimization problems (LANG *et al.*, 2015; MIRANDA; SILVA; PRUDÊNCIO, 2014) based on '*racing*' procedures. A race starts with a population $P(t)$ of HP settings and iteratively selects the most promising candidates considering the HP distributions, comparing them by statistical tests. Configurations that are statistically worse than at least one of

other configuration candidates are discarded from the racing. Based on the surviving candidates, HP distributions are updated. This process is repeated until a stopping criterion is not satisfied. An outline of the Irace algorithm is given in Algorithm 7.

Algorithm 7 – Irace pseudocode

```
t ← 0
Generate initial population P(0)
Evaluate the current population P(0)
Estimate the number of races N^races
Do a Race R(0) with the initial population P(0)
while Stopping Criteria not satisfied do
    t ← t + 1
    Calculate the limited budget B(t) of evaluations
    Generate new candidates N(t) from P(t) distributions
    Add N(t) into P(t)
    Do a Race R(t) with the population P(t)
end while
Best_global ← best individual in P(t)
return Best_global
```

2.3 Chapter remarks

In this chapter, the formal definition of the HP tuning problem is presented. Subsequent sections also described the main techniques often used to solve it: ranging from the most straightforward techniques GS and RS, meta-heuristic like GA, PSO, and EDA; to more complex approaches like SMBO and Irace. All these techniques can improve the predictive performance of the final induced models, but they can also be very time consuming to find suitable HP settings. Moreover, it is not guaranteed that the tuning process will lead neither to improved ML models nor significant improvements. As stated in the thesis' hypothesis, MtL can be useful to make HP tuning less costly. MtL is the subject of the next chapter.

CHAPTER

3

META-LEARNING ON HYPERPARAMETER TUNING

Several ML algorithms have been proposed for prediction tasks. However, since each algorithm has its inductive bias, some of them can be more appropriate for a particular dataset. When applying a ML algorithm to a dataset, a higher predictive performance can be obtained if an algorithm whose bias is most appropriate to the datasets is used. The recommendation of the most adequate ML algorithm for a new dataset is investigated in an research area known as Meta-learning (MtL) (VILALTA; DRISSI, 2002; SMITH-MILES, 2009; BRAZDIL *et al.*, 2009; LEMKE; BUDKA; GABRYS, 2015).

This chapter provides an overview of MtL on HP tuning. The next sections are organized as follow: Section 3.1 introduces MtL concepts; Section 3.2 describes several approaches using MtL on HP proposed in the literature; Section 3.3 discusses the general picture of the research area; Section 3.4 discusses the gaps presented in the literature; and, Section 3.5 presents the final remarks of the chapter.

3.1 Meta-learning

In recent years, MtL has been largely used for algorithm selection (ALI; SMITH-MILES, 2006), algorithm ranking (KANDA *et al.*, 2011) and prediction of the performance of ML algorithms (REIF *et al.*, 2014). In general words, MtL learns from many previous experiences, i.e., applying different learning algorithms to many datasets, inducing a model capable of selecting the most promising algorithm for a new dataset.

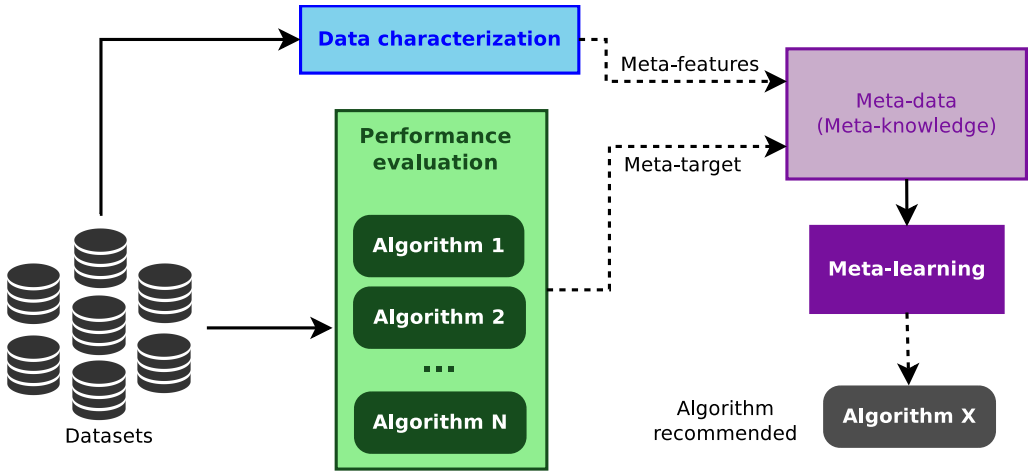


Figure 3 – Main steps of Meta-learning (Adapted from Brazdil *et al.* (2009)).

3.1.1 Definition

In this approach, each application of a learning algorithm to a dataset is named as *base-learning* and the knowledge extracted during the process is commonly designated as *meta-knowledge*. According to Brazdil *et al.* (2009), MtL can be used to continually improve the learning mechanism after each training process.

Thus, the MtL investigates the relation between tasks/problem domains and learning strategies. The goal is to select the most promising algorithm for a given task and to understand when a particular learning strategy is more suitable than others. This task is also commonly referred to as "Algorithm Selection" (SOARES; BRAZDIL; KUBA, 2004; BRAZDIL *et al.*, 2009). Figure 3 illustrates the main steps of these typical MtL process.

The first subtask is to select a set of datasets and ML algorithms. Both sets will be used to create a meta-dataset. Each example in the meta-dataset corresponds to a conventional dataset used in the ML task. Datasets would also be characterized by a set of descriptors, named "meta-features". At the same time, ML algorithms will perform over these datasets, and their predictive performances will be used to define which one performs best, the "meta-target". Meta-features and meta-target compose the meta-knowledge about the recommendation problem.

As such, it is important to learn what makes a learning algorithm successful or unsuccessful when applied to a given dataset. Besides, some criterion must be obeyed to guarantee a learning task at the meta-level:

- for each dataset, at least one algorithm should overcome the adopted baseline;
- given a set of algorithms A, their performance at the base level cannot be significantly improved by the addition of a new algorithm;
- when comparing algorithms at the base level, each algorithm should overcome the others at least once;
- an algorithm should not overcome another algorithm in all the datasets.

Following these requirements, a suitable meta-dataset is generated, and as result of the MtL process, a meta-model is induced. This meta-model represents a mapping between meta-features describing a dataset, and the predictive performance obtained by the group of learning algorithms when applied to these datasets. Therefore, the quality of the meta-features is essential for the predictive performance of the meta-models. Hence, it can be used to predict the best algorithm for a new unseen problem.

3.1.2 Data characterization

The meta-features extracted from each dataset must be sufficient to describe the main aspects of the dataset necessary to distinguish the predictive performance obtained by different learning algorithms when applied to this dataset and, as a result, allow the induction of a meta-model with a good predictive performance. According to Vilalta *et al.* (2004) three different approaches can be used to extract meta-features:

(i) *Simple*, *Statistical* and *Information-theoretic* meta-features (BRAZDIL; GAMA; HENERY, 1994; BRAZDIL; HENERY, 1994): consist of simple measures about the input dataset, such as the number of attributes, examples and classes, skewness, kurtosis and entropy. They are the most explored subset of meta-features in literature (REIF *et al.*, 2014; MANTOVANI *et al.*, 2015c; SOARES; BRAZDIL; KUBA, 2004; GOMES *et al.*, 2012; MIRANDA *et al.*, 2012; REIF; SHAFAIT; DENGEL, 2012; FEURER; SPRINGENBERG; HUTTER, 2015);

(ii) *Model-based* meta-features (BENSUSAN; GIRAUD-CARRIER; KENNEDY, 2000; PENG *et al.*, 2002): are a set of properties of a model induced by a ML algorithm for the dataset at the hand. For instance, if a decision tree induction algorithm is applied to the dataset, statistics about nodes, leaves and branches can be used to describe the dataset. They have also been used frequently in literature (REIF *et al.*, 2014; REIF; SHAFAIT; DENGEL, 2012);

(iii) *Landmarking* (PFAHRINGER; BENSUSAN; GIRAUD-CARRIER, 2000; VANSCHOREN, 2010): the predictive performance obtained by models induced by simple learning algorithms, named landmarkers, are used to characterize a dataset. These measures were

explored in studies such as (REIF *et al.*, 2014; FEURER *et al.*, 2015; FEURER; SPRINGENBERG; HUTTER, 2015).

Recently, new sets of measures have been proposed and explored in literature, like:

(iv) *Data complexity* (HO; BASU, 2002; ORRIOLS-PUIG; MACIA; HO, 2010): is a set of measures which analyze the complexity of a problem considering the overlap in the attributes values, the separability of the classes, and geometry/topological properties. They have been explored in (MANTOVANI *et al.*, 2015c; GARCIA; de Carvalho; LORENA, 2015; GARCIA; CARVALHO; LORENA, 2016; NOJIMA; NISHIKAWA; ISHIBUCHI, 2011); and

(v) *Complex networks* (MORAIS; PRATI, 2013; GARCIA; de Carvalho; LORENA, 2015): measures based on complex network properties are extracted from a network built with the data instances. These measures can only be extracted from numerical data. Thus, preprocessing procedures are required for their extraction. They were explored in (GARCIA; de Carvalho; LORENA, 2015; GARCIA; CARVALHO; LORENA, 2016).

An alternative for dataset characterization is the use of "*relative landmarks*" measures (LEITE; BRAZDIL; VANSCHOREN, 2012). In this approach, datasets are characterized based on the pairwise performance differences of the algorithms run on them: if the performance difference of two algorithms keeps similar across datasets, then data distributions of these datasets are likely to be similar regarding the algorithms' performance. Relative landmarking meta-features are often explored with Active Testing (AT) systems, a kind of MtL (MIRANDA *et al.*, 2012; ABDULRAHMAN *et al.*, 2018).

The meta-features' selection can be considered as an *ad hoc* process. Pinto, Soares and Mendes-Moreira (2016) present a framework to systematically generate meta-features in the MtL context. The idea is to decompose meta-features into different components and use post-processing functions to aggregate values over several data attributes. Experiments showed that systematically generated sets could be more informative than non-systematic meta-features for algorithm selection problem.

Using any of these meta-feature sets, the recommendation returned by a meta-model for a new dataset may be the most promising learning algorithm(s), a set of the N best learning algorithms or a ranking of learning algorithms according to their predictive potential for this data set.

3.2 Meta-learning on HP tuning

As previously mentioned, there is a large number of studies investigating the use of MtL to automate one or more steps in the application of ML algorithms to data analysis tasks. These

studies can be roughly grouped into the following approaches, depending on which aspect that MtL is applied:

- Recommends HP settings;
- Predicts training runtime;
- Recommends initial values for HP optimization;
- Generates rules for extraction of meta-features;
- Predicts HP tuning necessity;
- Estimates predictive performance for a given HP setting.

Tables 1 and 2 present a comprehensive list of studies that either embedded or used MtL to cope with the HP tuning problem. Next, these works are described in detail.

3.2.1 Recommendation of HP settings

The first approach considered HP settings as independent algorithm configurations and predicted the best setting based on characteristics of the dataset under analysis. In this approach, the HP settings are predicted without actually evaluating the model on the new dataset (SOARES; BRAZDIL; KUBA, 2004). In Soares, Brazdil and Kuba (2004) and Soares and Brazdil (2006), the authors predicted the width (γ) of the SVM Gaussian kernel for regression problems. A finite set of γ values was investigated for 42 regression problems, and the predictive performance was assessed using 10-fold Cross-Validation (CV) and the Normalized Mean Squared Error (NMSE) evaluation measure.The recommendation of γ values for new datasets used a k-Nearest Neighbors (kNN) meta-learner.

Ali and Smith-Miles (2006) presented a similar study but selecting one among five different SVM kernel functions for 112 classification datasets. They assessed model predictive performance for different HP settings using 10-CV procedure and the simple Accuracy (Acc) measure. Miranda and Prudêncio (2013) adapted the AT MtL approach' (LEITE; BRAZDIL; VANSCHOREN, 2012) to select the γ and the soft margin (C) SVM HPs. Experiments performed on 60 classification datasets assessed the settings using a single 10-CV and the Acc measure.

Nojima, Nishikawa and Ishibuchi (2011) employed MtL to recommend the number of fuzzy functions (3,5 or 7) to a Chi-RW fuzzy algorithm. Experiments were performed with 24 Keel original datasets and 300 artificial generated from them. Meta-examples were characterized by data complexity meta-features, and a multiobjective genetic fuzzy rule selection meta-learners induced meta-models with a Leave One Out (LOO)-CV assessing them with the simple Acc measure.

Table 1 – Summary of related studies applying to MtL to HP tuning. Fields without information in the related study received an hyphen (1 of 2).

Reference	Meta-learning	Base Learner(s)	Tuning Techniques	Meta Learner(s)	N. of Datasets	Source of Datasets	Evaluation Procedure	Evaluation Measure
Soares, Brazdil and Kuba (2004)	Recommends HP settings	SVM	GS	kNN	42	UCI, METAL	10-CV	NMSE
Soares and Brazdil (2006)	Recommends HP settings	SVM	GS	kNN	42	UCI, METAL	10-CV	NMSE
Ali and Smith-Miles (2006)	Recommends HP settings	SVM	GS	C5.0	112	UCI, KDC	10-CV	Acc
Nojima, Nishikawa and Ishibuchi (2011)	Recommends HP settings	Chi-RW	GS	MoGFRS	24	Keel	Holdout	Acc
Pinto *et al.* (2017)	Recommends HP settings	Bagging	GS	Xgboost	146	OpenML	-	Kappa
Zhongguo *et al.* (2017)	Recommends HP settings	C4.5 SVM kNN	GS	kNN	100	UCI	Holdout	MAE
Miranda and Prudêncio (2013)	Recommends HP settings	SVM	GS	AT	60	UCI	10-CV	Acc
Lorena *et al.* (2018)	Recommends HP settings	SVM	GS	kNN	39	UCI	10-CV	NMSE
Reif, Shafait and Dengel (2011)	Predicts training runtime	SVM, kNN, MLP, Ripper DT*	GS	SVM	123	UCI	-	PMCC NAE
Priya *et al.* (2012)	Predicts training runtime	SVM	GA	J48, SVM Bagging, NB kNN, Jrip	78	UCI	5-CV	MAD
Gomes *et al.* (2012)	Recommends initial values for HP optimization	SVM	GS PSO,TS	kNN	40	WEKA	10-CV	NMSE
Miranda *et al.* (2012)	Recommends initial values for HP optimization	SVM	GS, PSO	kNN	40	UCI, WEKA	10-CV	Acc
Reif, Shafait and Dengel (2012)	Recommends initial values for HP optimization	SVM RF	GS,GA	kNN	102	UCI, Statlib	10-CV	Acc
Miranda, Silva and Prudêncio (2014)	Recommends initial values for HP optimization	SVM	GS,PSO	kNN	100	UCI	10-CV	Acc
Feurer *et al.* (2015) Feurer, Springenberg and Hutter (2015)	Recommends initial values for HP optimization	SVM, RF CASH	GS, SMBO	kNN	140	OpenML	Nested-CV	Acc

Table 2 – Summary of related studies applying to MtL to HP tuning. Fields without information in the related study received an hyphen (2 of 2).

Reference	Meta-learning	Base Learner(s)	Tuning Techniques	Meta Learner(s)	N. of Datasets	Datasets' Source	Evaluation Procedure	Evaluation Measure
Sun and Pfahringer (2013)	Generates rules for extraction of meta-features	CASH	PSO	Ripper ATR Forests	466	UCI, WEKA Statlib, KDD	10-CV	AUC
Molina *et al.* (2012)	Predicts HP tuning necessity	J48	GS	NB, SVM LR, PART JRip, CART RBF, NNge LADTree, MLP REPTree, J48 Riddo BayesNet	14	-	10-CV	Acc
Ridd and Giraud-Carrier (2014)	Predicts HP tuning necessity	CASH	PSO	J48, RF SVM	326	UCI, WEKA Statlib, KDD	-	AUC
Gunasekara, Pang and Kasabov (2010)	Estimates predictive performance	SVM	GS	SVM	4	-	10-CV	RMSE
Reif *et al.* (2014)	Estimates predictive performance for a HP setting	SVM, DT*, kNN, MLP RF, OneR NB, LR FLD	GS	SVM	54	UCI, Statlib	10-CV	PMCC RMSE
Wistuba, Schilling and Schmidt-Thieme (2015)	Estimates predictive performance for a HP setting	AdaBoost SVM	GS	SMFO	25	UCI	Holdout	CANE
Wistuba, Schilling and Schmidt-Thieme (2016) Wistuba, Schilling and Schmidt-Thieme (2018)	Estimates predictive performance for a HP setting	SVM CASH	GS	GP	109	UCI, WEKA	Holdout	-
Eggensperger *et al.* (2018)	Estimates predictive performance for a HP setting	SVM Xgboost	ROAR, Irace RS, PILS	RF	11	AClib	Holdout	RMSE SCRR

Pinto *et al.* (2017) proposed the "autoBagging" tool, an AutoML system that automatically ranks Bagging work-flows considering four different Bagging HPs by exploring past performance and dataset characterization. Experiments were carried out with 140 OpenML datasets and 146 meta-features (extracted with post-processing aggregation functions (PINTO; SOARES; MENDES-MOREIRA, 2016)). They used an XGBoost as meta-learner to predict ranking of work-flows, and evaluated results at the meta-level using a Mean Average Precision (MAP) measure in a LOO-CV.

Zhongguo *et al.* (2017) employed MtL to recommend algorithm and its HP. The authors performed experiments with C4.5, kNN and SVM classifiers covering few of their HP by a grid design. Performances were evaluated over 100 UCI datasets regarding AUC. In the end, a kNN meta-model was applied to recommend the best HP setting for new unseen datasets.

Lorena *et al.* (2018) proposed a set of new complexity meta-features for regression problems. One of the case studies evaluated was the SVM HP tuning problem. The authors generated a finite grid of γ, C and ε (margin of tolerance for regression SVMs) values, assessing them with a single 10-fold CV and NMSE measure, over 39 regression problems. A kNN distance-based meta-learner made the recommendation of HP for new unseen datasets.

3.2.2 Prediction of training runtime

A different approach is to use MtL to estimate the training runtime of classification algorithms when induced by different HP settings. In Reif, Shafait and Dengel (2011), the authors predicted the training runtime of several classifiers: kNN, SVM, Multilayer Perceptron (MLP), a DT and RIPPER[1]. They defined a discrete grid of HP settings, evaluating these settings on 123 classification datasets. The performance measures used for HP assessment were the Pearson Product Moment Correlation Coefficient (PMCC) and Normalized Absolute Error (NAE).

In Priya *et al.* (2012), the authors conducted a similar study with SVMs but using a Genetic Algorithm (GA) to optimize HP and perform feature selection of six meta-learners (kNN, SVM, J48, JRip, NB, and Bagging). Experiments were carried out over 78 classification datasets assessing HP settings using 5-CV and the Mean Absolute Deviation (MAD) evaluation measure.

3.2.3 Recommendation of initial values for HP optimization

MtL has also been used to speed up the optimization of HP values for classification algorithms (GOMES *et al.*, 2012; MIRANDA *et al.*, 2012; REIF; SHAFAIT; DENGEL, 2012; FEURER *et al.*, 2015). In Gomes *et al.* (2012) MtL is used to recommend HP settings that will compose the initial population of the Particle Swarm Optimization (PSO) and Tabu Search (TS)

[1] Repeated Incremental Pruning to Produce Error Reduction (RIPPER) is a propositional rule algorithm which employs association rules with reduced error pruning.

optimization techniques. Experiments were conducted in 40 regression datasets adjusting the C and γ SVM HPs to reduce the NMSE value. A kNN meta-learner was used to recommend the initial search values.

Reif, Shafait and Dengel (2012) and Miranda *et al.* (2012) investigated, respectively, the use of GAs and different versions of PSOs for the same task. In Miranda, Silva and Prudêncio (2014), the authors used multi-objective tuning to optimize the HPs increasing predictive performance and the number of support vectors. These studies generated meta-knowledge evaluating SVMs for a grid of $\gamma \times C$ values. These studies used simple accuracy measure and 10-fold CV to optimize $\gamma \times C$ HP values.

The same approach is explored in a tool to automate the use of ML algorithms, the `Auto-skLearn` (FEURER *et al.*, 2015; FEURER; SPRINGENBERG; HUTTER, 2015). In this tool, the MtL is used to recommend HP settings for the initial population of the Sequential Model-based Optimization (SMBO) technique. The authors explored a Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem including ten algorithms and their HPs. Experiments were performed over 140 OpenML classification datasets, exploring meta-feature selection and dimensionality reduction techniques (PCA), assessing HP settings with the simple Acc measure. It is also the first work that uses nested-CVs for evaluating HP settings.

3.2.4 Generation of rules for the extraction of meta-features

Sun and Pfahringer (2013) presented a novel meta-feature generation method for MtL, inducing meta-feature extraction rules. They performed experiments on 466 binary classification datasets with all the available WEKA classifiers. The new meta-rules are used to produce a ranking of the most suitable algorithms for new unseen datasets. At the base level, a PSO method searches the space evaluating HP settings using 10-CV and Area Under the ROC curve (AUC) measure.

3.2.5 Prediction of HP tuning necessity

In the current thesis, we investigate a way of predicting the HP tuning necessity. In literature, there are two studies with proposals close to our approach. Molina *et al.* (2012) employed MtL to predict the improvement obtained in a DT algorithm varying it HPs. They selected the C4.5 algorithm and defined a grid of values for the HPs C and M. A total of 14 educational datasets were characterized by means of 5 simple meta-features. Thus, the MtL was used to predict whether the use of different HP settings increase, decrease or maintain the predictive performance of the induced DTs.

Ridd and Giraud-Carrier (2014) used MtL to identify when HP tuning would lead to significant increase in accuracy. They carried out experiments using a PSO technique to search

the hyperspace of several ML algorithms in 326 binary classification datasets. Different than us, they considered the tuning of all the algorithms as a unique scenario. The analysis performed by the authors considered different thresholds to determine when an improvement was obtained or not in the data collection, but no statistical analysis was performed.

Unlike these two related studies, the MtL experiments described in this thesis[2] explored a large number of heterogeneous datasets, an unbiased tuning methodology, and induced meta-models for different target algorithms. The experimental setup also included meta-features from several categories and different learning processes.

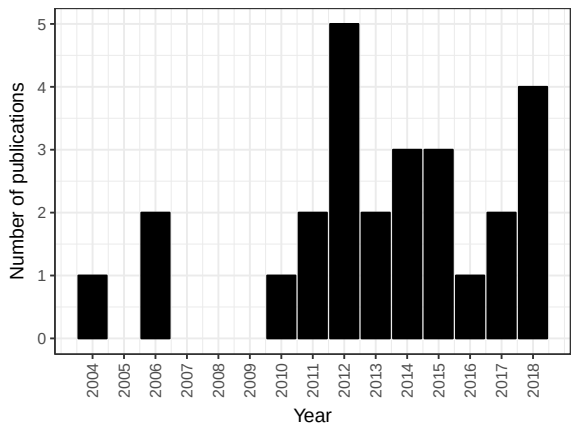
3.2.6 Estimation of predictive performance for a given HP setting

A more recent approach uses MtL to estimate ML algorithms performance considering their HPs. In Gunasekara, Pang and Kasabov (2010) the MtL is used to predict the performance of SVMs when a kernel n-gram is used to handle four different string datasets. The HP values are discretized, then all the combinations evaluated using a 10-CV and the RMSE performance measure. With this meta-data, HP settings are further recommended for new unseen data.

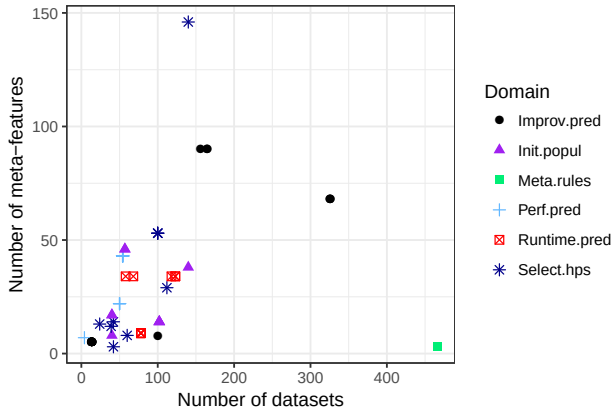
In Reif *et al.* (2014), the authors evaluated different ML algorithms over 54 datasets and used the performance predictions to develop a MtL system for automatic algorithm selection. Wistuba, Schilling and Schmidt-Thieme (2015), Wistuba, Schilling and Schmidt-Thieme (2016) adapted the acquisition function of surrogate models by one optimized meta-model. They evaluated several HP configurations in a holdout fashion procedure over 105 datasets and used the meta-knowledge to predict the performance of new HP settings for new datasets. The base-level algorithms explored were the AdaBoost and SVM. The authors also proposed a new Transfer Acquisition Function (TAF) that extended the previous studies predicting the performance of HP settings for surrogate models (WISTUBA; SCHILLING; SCHMIDT-THIEME, 2018).

Eggensperger *et al.* (2018) proposed a benchmarking approach of "surrogate scenarios", which extracts meta-knowledge from HP optimization and algorithm configuration problems, approximating the performance surface by RF regression models. Two of the 11 meta-datasets explored in the experimental setup are HP tuning problems: SVMs HP settings assessed in the `MNIST` dataset, and Extreme Gradient Boosting (Xgboost) algorithm's settings assessed on the `covertype` dataset. These settings were obtained executing a simple RS method and three optimizers: Random Online Adaptive Racing (ROAR) (HUTTER *et al.*, 2009), Irace (LóPEZ-IBáñEZ *et al.*, 2016), and Iterated Local Search in Parameter Configuration Space (PILS) (HUTTER *et al.*, 2009).

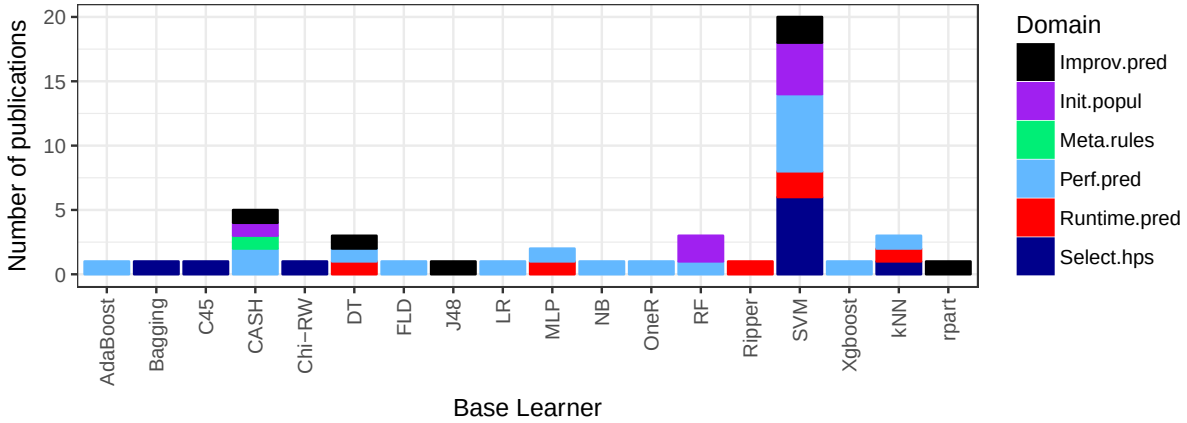
[2] See Section 3.3 and Chapter 7.



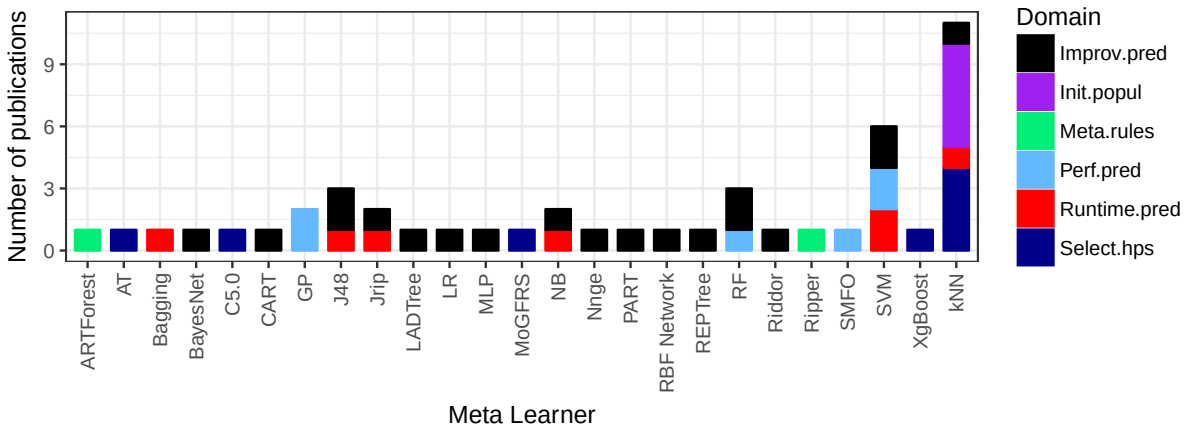
(a) Related studies published per year.



(b) Number of datasets and meta-features used by related studies.



(c) Algorithms used as base-learners by related studies.



(d) Algorithms used as meta-learners by related studies.

Figure 4 – Statistics from studies which employed MtL for HP tuning.

3.3 General picture

Figure 4 shows the main statistics from MtL studies discussed above. The literature review indicates that the use of MtL for HP tuning has grown in the last years (figure 4a). Since 2010, on average, two papers covering this main subject are published per year. However, they

are still just a few, and the problem could be more deeply explored in the literature. A total of **25** studies were found during the bibliographic review.

Regarding the base-level problems, datasets come mostly from the same sources. Very first studies used UCI, Keel and WEKA repositories, but, since 2014, more publications are using OpenML. OpenML provides an extensive data collection, which currently contains most of the datasets from UCI, Keel, and WEKA, allowing greater reproducibility of the experiments.

Figure 4b relates the number of datasets and meta-features explored by the related studies. The legend at the right side depicts each application domain discussed in the beginning of the chapter: i) Predicts HP tuning necessity (`Improv.pred`); ii) Recommends initial values for HP optimization (`Init.popul`) ; iii) Generates rules for extraction of meta-features (`Meta.rules`); iv) Estimates predictive performance for a HP setting (`perf.pred`); iv) Predicts training runtime (`Runtime.pred`); and vi) Recommends HP settings (`Select.hps`).

With few exceptions (SUN; PFAHRINGER, 2013; RIDD; GIRAUD-CARRIER, 2014), most of the studies were conducted with less than 100 datasets. Since the HP optimization of the base-level algorithm is required to generate the meta-knowledge, it limits the number of examples explored by the MtL systems. The number of datasets increased just in most recent studies when researchers could use more powerful computational architectures, like clusters and GPUs. For this, as the number of datasets processes increased, different sets of meta-features were also proposed and investigated. The exceptions are the studies proposed by Sun and Pfahringer (2013), Ridd and Giraud-Carrier (2014): the first generated 466 binary datasets, which were later curated, reduced, and used in the experiments described by the last.

Figure 4c shows the algorithms whose HP were optimized/recommended by the MtL systems. Most of the studies consider SVMs as the subject of research, while few studies investigate kNN and DTs tuning. It is interesting that SVMs are explored by almost all of the domains. The use of SVMs as study case may be justified by their sensitivity to tuning: defaults are not recommended for most of the problems and tuning often improves significantly their performance (RIDD; GIRAUD-CARRIER, 2014). The SVM tuning problem is also often modelled as a low-dimensional problem with few HPs to adjust, being an interesting first study case to evaluate tuning approaches. An important observation should be made about the CASH problems; they describe mixed hyper-spaces that cover more than 10 learning algorithms and, therefore, were considered as a category different from the others.

Regarding the choice of the meta-learner algorithm, Figure 4d shows the general picture of the related studies. Most of the approaches prioritized the use of an instance-based algorithm (kNN). Relating recommendations with the most similar training example is a natural way to search for patterns, and then identify which features explain the decision making process. However, it does not always work, and more recent studies have been exploring different algorithms at the meta-level. Curiously, MtL systems used to predict the improvement provided

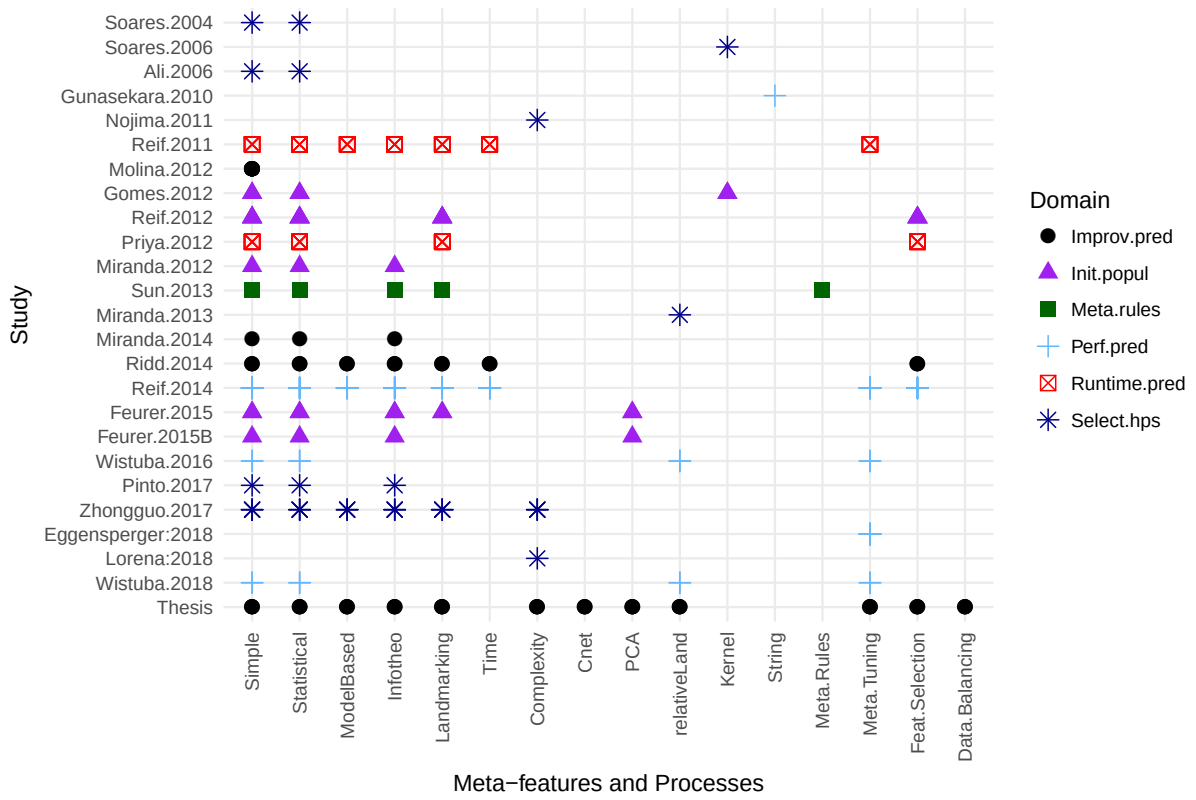


Figure 5 – Meta-features' sets and additional processes used in previous work with MtL and HP tuning.

by tuning techniques are those who employed most instance-based meta-learners[3].

Figure 5 shows all sets of meta-features and data processes explored in the literature. All the related studies are shown chronologically in the y-axis, while meta-features' sets and data processes are listed at the x-axis. The application domains are also highlighted by different shapes and colors. Simple, Statistical, Model-based and Information-theoretic meta-features are not surprisingly the most explored in literature (BRAZDIL *et al.*, 2009). There are some specific sets of meta-features used in one study only. It happens with the string-based (GUNASEKARA; PANG; KASABOV, 2010) and pairwise meta-rules meta-features (SUN; PFAHRINGER, 2013). Other sets are rarely explored depending on the application domain: kernel-based meta-features were used when the object of study are SVMs (SOARES; BRAZDIL, 2006; GOMES *et al.*, 2012); relative landmarking is the representation adopted by AT algorithms (MIRANDA; PRUDÊNCIO, 2013); while data complexity meta-features were tried by studies selecting HP settings (NOJIMA; NISHIKAWA; ISHIBUCHI, 2011; ZHONGGUO *et al.*, 2017; LORENA *et al.*, 2018).

Dimensionality reduction was first and only handled by the `Auto-skLearn` system (FEURER; SPRINGENBERG; HUTTER, 2015; FEURER *et al.*, 2015). The HP tuning of the meta-learners has been explored only by more recent studies predicting the performance of the algorithms (`perf.pred`). The meta-feature selection process was explored just by few studies (REIF; SHAFAIT; DENGEL, 2012; PRIYA *et al.*, 2012; RIDD; GIRAUD-CARRIER, 2014; REIF *et*

[3] Meta-learners used by each one of the studies are also listed in Tables 1 and 2

al., 2014), but those who present a higher number of meta-examples. Finally, none of the related studies explored data balancing techniques.

3.4 Literature overview

The literature review permitted to identify some interesting aspects. Overall:

- most of the studies created the meta-data using GS to tune the algorithms' HPs;
- most of them also evaluated the resultant models with a single CV resampling procedure and the simple Acc evaluation measure;
- most of the studies were conducted with at most 100 datasets (Figure 4b). In (RIDD; GIRAUD-CARRIER, 2014; SUN; PFAHRINGER, 2013) the authors used more than 300 datasets, but all of them are binary classification problems;
- at the meta-level, most of the studies considered instance based meta-learners (kNN). Different algorithms were tried at the meta-learning level but mostly concentrated by the same study;
- experimental setups considering meta-feature selection, dimensionality reduction or HP tuning of the meta-learners were considered just by few studies (FEURER; SPRINGENBERG; HUTTER, 2015; FEURER *et al.*, 2015; REIF *et al.*, 2014; RIDD; GIRAUD-CARRIER, 2014);
- different approaches to generate meta-features are not well explored in literature: data complexity meta-features (Complexity, Cnet) are used just by three of the related studies; the same for relative landmarking meta-features;
- few of them provide the complete resources for the reproducibility of experiments;
- None of the studies found by the authors combined all these six previous issues.

3.5 Chapter remarks

This chapter presented the literature review of MtL on HP tuning. Related studies were split into six categories according to the final goal of the MtL system. The main aspects of these studies are presented in Tables 1 and 2 and illustrated in Figures 4 and 5. Although the number of related papers in the last years has been increasing, there are still several aspects that need further investigation. Literature presents some patterns: most of the studies produced meta-knowledge through GS executions with a Holdout or single CV resampling, characterizing datasets using simple and statistical meta-features, and recommending HPs with a kNN meta-learner. The reproducibility of the experiments and the sharing of results are two key aspects that

have not been explored yet. These aspects would benefit the research community with valuable meta-knowledge for further work. Thus, exploring different experimental setups, meta-features, and procedures applied to the learning task may open up new horizons in the MtL research area.

CHAPTER

4

EXPERIMENTAL METHODOLOGY

As shown in the previous chapter, different evaluation methodologies have been used to assess the HP tuning of ML algorithms in MtL recommender systems. There is no standard in the literature, and most of the studies prefer the simplest approaches. However, it is not always the simplest solution that can describe a correct surface of the HP spaces. An important aspect considered in this thesis is how HP settings are assessed during the optimization, to minimize the chance of overfitting. Next sections briefly present the main experimental and methodological choices adopted through the thesis.

The next sections are organized as follows: Section 4.1 briefly presents target ML algorithms investigated in this thesis; Section 4.2 discusses the use of nested-CVs for HP tuning; Sections from 4.3 to 4.6 presents datasets, evaluation metrics, statistical tests and AutoML tools adopted in experiments; Section 4.7 enumerates tuning techniques' parameters adopted throughout the thesis; Section 4.8 lists repositories with the experimental results; and, Section 4.9 presents the final remarks of the chapter.

4.1 Classification algorithms

Many ML algorithms for solving supervised classification tasks can be found in the literature. Two of the most important approaches are: Support Vector Machines and Decision Trees' induction algorithms (MITCHELL, 1997).

4.1.1 Support vector machines

Support Vector Machines (SVMs) (VAPNIK, 1995) are kernel-based algorithms that perform non-linear classification using a hyperspace transformation, i.e., they map data inputs into a high-dimensional feature space where the problem is possibly linearly separable. They are known to show robust performance across a wide variety of problems (WAINBERG; ALI-

PANAHI; FREY, 2016). A typical kernel function used by SVMs is the Radial Basis Function (RBF) which contains the HP γ (the width of the kernel). The penalty HP C controls the trade-off between minimizing the number of wrongly labeled examples and maximizing the margin[1].

4.1.2 Decision tree induction algorithms

Although high predictive accuracy is the most frequently used measure to evaluate ML algorithms, in many applications, easy interpretation of the induced models is also a necessary requirement. Good predictive performance and model interpretability are found in one of the most successful sets of classification algorithms: Decision Tree (DT) induction algorithms (ROKACH; MAIMON, 2014).

They also have several advantages over many ML algorithms, such as robustness to noise, tolerance against missing information, handling of irrelevant and redundant predictive attribute values, and low computational cost (ROKACH; MAIMON, 2014). Their importance is demonstrated by a wide range of well-known algorithms proposed in the literature, such as Breiman et al.'s Classification and Regression Tree (CART) (BREIMAN *et al.*, 1984) and Quinlan's C4.5 algorithm (QUINLAN, 1993), as well as some hybrid-variants of them, like Naïve-Bayes Tree (NBTree) (KOHAVI, 1996), Logistic Model Tree (LMT) (LANDWEHR; HALL; FRANK, 2005) and Conditional Inference Trees (CTree) (HOTHORN; HORNIK; ZEILEIS, 2006).

In this thesis, the HP tuning of DT induction algorithms was also investigated, as well as the impact of their HPs in the predictive performance. For such, three DT induction algorithms were selected as study-cases:

- two of the most popular algorithms in ML (WU; KUMAR, 2009) - the J48 algorithm, a `WEKA` (WITTEN; FRANK, 2005) implementation for the Quinlan‘s C4.5 (QUINLAN, 1993); the Breiman et al.'s CART algorithm (BREIMAN *et al.*, 1984); and

- the CTree (HOTHORN; HORNIK; ZEILEIS, 2006), an algorithm that works similarly to the traditional CHAID algorithm (KASS, 1980), using statistical tests, but it provides a more recent implementation which handles different types of data attributes[2].

Experiments were focused on these algorithms due to the interpretability of their induced models and widespread use. All of them generate simple models, are robust for specific domains, and allow non-experts users to understand how the classification decision is made.

[1] A mode detailed discussion regarding SVM can be found in Ben-Hur and Weston (2010).
[2] The CHAID algorithm processes just categorical data attributes.

4.2 Nested-CV resamplings

Krstajic *et al.* (2014) compared different resampling strategies for selecting and assessing the predictive performance of regression/classification models induced by ML algorithms. In Cawley and Talbot (2010) the authors also discussed overfitting in the evaluation methodologies when assessing ML algorithms. They describe an "*unbiased performance evaluation methodology*", which correctly accounts for any overfitting that may occur in the model selection. The internal protocol described by the authors performs the model's selection independently within each fold of the resampling procedure. In fact, most of the current studies on HP tuning have adopted nested-CVs, including important AutoML tools, like `Auto-WEKA`[3] (THORNTON *et al.*, 2013; KOTTHOFF *et al.*, 2016) and `Auto-skLearn`[4] (FEURER; SPRINGENBERG; HUTTER, 2015; FEURER *et al.*, 2015). Thus, this thesis adopts the nested-CV methodology in the experiments.

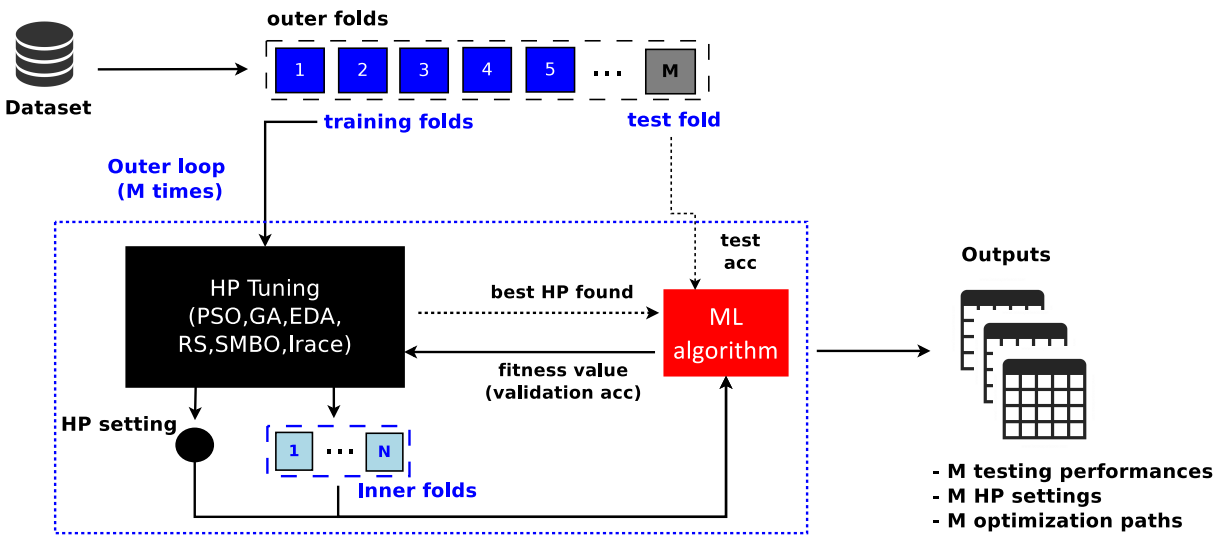


Figure 6 – Experimental methodology used to adjust ML HPs. The tuning is conducted via nested cross-validation: 3-fold CV for computing fitness values and 10-fold CV for assessing performances. The outputs are the HP settings, the predicted performances and the optimization paths of each technique. Figure adapted from Mantovani *et al.* (2018a).

The nested-CV (CAWLEY; TALBOT, 2010; KRSTAJIC *et al.*, 2014) methodology is illustrated by Figure 6. For each dataset, data are split into M outer-folds: the training folds are

[3] <http://www.cs.ubc.ca/labs/beta/Projects/autoweka/>
[4] <https://github.com/automl/auto-sklearn>

used by the tuning techniques to find good HP settings, while the test fold is used to assess the 'optimal' solution found. Internally, tuning techniques split each of the M training folds into N inner-folds to measure the fitness value of each new HP setting. At the end of the process, a set of M optimization paths, M settings, and their predictive performances are returned. During the experiments, all the tuning techniques were run on the same data partitions, with the same seeds and data to properly allow their comparison. In (KRSTAJIC *et al.*, 2014), the authors used $M = N = 10$. However, they argued that there is no study suggesting the number of folds in the outer and inner CV loops. Here, the same value used in the original paper was used for $M = 10$. Due to time constraints and the size of datasets used in the experiments, $N = 3$ was adopted.

4.3 Datasets

A total of 169 binary and multiclass classification datasets from `OpenML` (VANSCHOREN *et al.*, 2014) were selected for the experiments. From all the available and active datasets, those attending the following criteria were selected:

(a) number of features does not exceed 1500;

(b) number of instances between 100 and 50.000;

(c) must not be a reduced, modified or binarized version of the original classification problem;

(d) must not be an adaptation of a regression dataset;

(e) all the classes must have at least 10 examples, enabling the use of stratified 10-fold CV.

All datasets, and their main characteristics, meeting these criteria are presented in Tables 18 to 20 at Appendix A. They were selected to cover a wide range of classification tasks, with different characteristics. The `OpenML` (CASALICCHIO *et al.*, 2017)[5] package was used to select and download datasets from the `OpenML` website, while functions from `mlr` (BISCHL *et al.*, 2016)[6] package were used to preprocess the collected datasets.

In order to allow their use by SVMs, the datasets were preprocessed: constant or identifier attributes were removed; the logical attributes (TRUE/FALSE) were converted to values $\in \{0, 1\}$; missing values were imputed by the median for numerical attributes, and a new category for categorical attributes; all categorical attributes were converted to the 1-N encoding; all attributes were normalized with $\mu = 0$ and $\delta = 1$. When performing HP tuning on DTs the original datasets were used, i.e., no preprocessing step required, since DTs can handle any missing information or data from different types.

[5] <https://github.com/openml/openml-r>
[6] <https://github.com/mlr-org/mlr>

4.4 Evaluation measures

Since the HP tuning experiments contain several and diverse datasets, many of them may have unbalanced classes. Hence, the Balanced Per Class Accuracy (BAC) measure (BRODERSEN *et al.*, 2010) was used as the fitness value during tuning, as well as for the final model assessment. For each dataset, each tuning techniques was repeated several times using different seeds. Thus, all results regarding BAC were averaged over these repetitions.

In MtL experiments described in Chapter 7, several binary classification problems were generated. Thus, their predictions were assessed using the Area Under the ROC Curve (AUC) performance measure, a more robust metric than BAC for binary problems. Besides that, using the AUC, it is also possible to evaluate the influence that different threshold values have in the predictions.

4.5 Statistical tests

The Friedman test (DEMŠAR, 2006), with significance levels at $\alpha = 0.05$ and $\alpha = 0.1$, was also used to compare the hyperparameter tuning techniques, evaluating the statistical significance of the experimental results. The null hypothesis states that all classifiers induced with the hyperparameter settings found by the tuning techniques, and the classifier induced by default values, are equivalent concerning predictive BAC performance. If the null hypothesis were rejected, the Nemenyi post-hoc test was applied, stating that the performances of two different techniques are significantly different if the corresponding average ranks differ by at least a Critical Difference (CD) value.

Another statistical test used in this thesis is the Wilcoxon paired-test (SANTAFE; INZA; LOZANO, 2015). In both base and meta-level experiments, it was used to compare tuning techniques performances with those obtained by default HP settings, indicating in which situations the improvement was statistically significant.

4.6 OpenML and mlr

Two experimental frameworks/tools were also adopted in this thesis:

- the `OpenML.org` (*Open Machine Learning*) (VANSCHOREN *et al.*, 2014): is a free scientific platform for standardization of ML experiments and sharing empirical results[7]. It works as a place for researchers and students share and organize data and experimental results, in a way they can collaborate with each other to handle harder problems. OpenML links to data available anywhere online and is being integrated in popular Data Mining plat-

[7] <http://openml.org/>

forms such as Weka (HALL *et al.*, 2009), R (R Core Team, 2016) and Python (ROSSUM, 1995)[8]; and

- the mlr R package (*Machine Learning in R*) package[9] (BISCHL *et al.*, 2016): a complete framework in R that provides several ML tools to generate experiments efficiently. The package is an interface to a large number of learning techniques (single and multi-target classification and regression algorithms, survival analysis, clustering and general example-specific cost sensitive learning), preprocessing methods (imputation, data balancing), feature selection, HP tuning, benchmarking experiments (plots, statistical analysis) and so on[10] or blog web pages[11].

These two tools work together through the OpenML[12] R package (CASALICCHIO *et al.*, 2017), an interface that provides functions to interact with the OpenML server. For example, users can download and upload files, run their implementations on a specific task (problem) and get predictions in the correct form using R commands[13].

Since AutoML is one of the primary motivations of the thesis, these tools provide ways to make it real. Therefore, it was possible to setup a large number of automatic experiments to generate meta-information used to feed the meta-learning recommender systems described in further chapters.

4.7 Setup of the tuning techniques

Table 3 presents the setup of the tuning techniques described in Chapter 2. Except for budget-dependent parameters, all parameters of the techniques are presented in this table. These values are the default values provided by each R package implementation. In SMBO, Irace and PSO cases, the use of the default values have shown to be robust enough to save time and resources (ZAMBRANO-BIGIARINI; CLERC; ROJAS, 2013; LÓPEZ-IBÁÑEZ *et al.*, 2016; BISCHL *et al.*, 2017). For EDA and GA (and evolutionary methods in general) there is no *standard* values for their parameters (MILLS; FILLIBEN; HAINES, 2015). So, to keep fair comparisons, the default parameter values provided by the correspondent R packages were adopted in the experiments.

The GA, PSO and EDA meta-heuristics were implemented using the GA[15](SCRUCCA, 2013),

[8] More details can be found in (VANSCHOREN *et al.*, 2014)
[9] <https://github.com/mlr-org/mlr>
[10] More details can be found in the official documentation <http://mlr-org.github.io/mlr-tutorial/release/html/index.html>
[11] <http://mlr-org.github.io/>
[12] <https://github.com/openml/openml-r>
[13] More detailed descriptions and examples can be found in the tutorial web page[14]
[15] <https://github.com/luca-scr/GA>

Table 3 – Parameters of the tuning techniques used in this thesis.

Technique	Parameter	Option	R Package
GS	stopping criteria	budget size	mlr
RS	stopping criteria	budget size	mlr
PSO	stopping criteria algorithm implementation	budget size SPSO2007 (CLERC, 2012)	`pso`
EDA	stopping criteria EDA implementation copula function margin function	budget size GCEDA normal truncnorm	`copulaedas`
GA	stopping criteria selection operator crossover operator crossover probability mutation operator mutation probability elitism rate	budget size proportional selection with linear scaling local arithmetic crossover 0.8 random mutation 0.05 0.05	`GA`
SMBO	initial design method surrogate model stopping criteria infill criteria	Random LHS Random Forest budget size expected improvement	`mlrMBO`
Irace	stopping criteria	budget size	irace

`pso`[16](BENDTSEN., 2012), and `copulaedas`[17](GONZALEZ-FERNANDEZ; SOTO, 2014) R packages, respectively. The J48, CART and CTree algorithms were implemented using the `RWeka`[18](HORNIK; BUCHTA; ZEILEIS, 2009), `rpart`[19](THERNEAU; ATKINSON; RIPLEY, 2015) and `party`[20](HOTHORN; HORNIK; ZEILEIS, 2006) packages, respectively, wrapped into the `mlr` package. The SMBO technique was implemented using the `mlrMBO`[21] (BISCHL *et al.*, 2017) R package, with its Random Forest (RF) surrogate models implemented by the `randomForest`[22] R package (LIAW; WIENER, 2002). The Irace technique was implemented using the `irace`[23] (LÓPEZ-IBÁÑEZ *et al.*, 2016) R package.

4.8 Repositories for the coding used in this thesis

Experimental results are available at the OpenML studies as described at Table 4. In the corresponding pages, all datasets, classification tasks, algorithms/flows, and results are available for reproducibility.

The code used for HP tuning process (`HpTuning`), extracting meta-features (`MfeatExtractor`),

[16] <https://cran.r-project.org/web/packages/pso/index.html>
[17] <https://github.com/yasserglez/copulaedas>
[18] <https://cran.r-project.org/web/packages/RWeka/index.html>
[19] <https://cran.r-project.org/web/packages/rpart/index.html>
[20] <https://cran.r-project.org/web/packages/party/index.html>
[21] <https://github.com/mlr-org/mlrMBO>
[22] <https://cran.r-project.org/web/packages/randomForest/index.html>
[23] <http://iridia.ulb.ac.be/irace/>

Table 4 – OpenML studies with results generated by experiments.

Level	Experiment	Repository
Base	SVM HP tuning	<https://www.openml.org/s/52>
Base	DT HP tuning	<https://www.openml.org/s/50>
Meta	When to tune SVM	<https://www.openml.org/s/58>
Meta	When to tune DT	<https://www.openml.org/s/94>

running meta-learning (`mtlSuite`), and performing the graphical analyses (`TuningAnalysis`, `MtlAnalysis`) are hosted at `GitHub`. All of these repositories are also listed at Table 5.

Table 5 – Repositories with tools developed by the authors.

Task	Repository
HP tuning process	<https://github.com/rgmantovani/HpTuning>
Meta-feature extraction	<https://github.com/rgmantovani/MfeatExtractor>
Meta-learning	<https://github.com/rgmantovani/mtlSuite>

Instructions to run each project can be found directly at the correspondent pages.

4.9 Chapter remarks

Evaluation methodologies adopted in previous work are not entirely suitable for HP setting assessment in a MtL recommender system. Most of the studies searched the HP space by sweeps or a simple GS with a single CV, and are evaluated in few datasets. To solve this issue, nested-CVs have been discussed in the literature to avoid any overfitting that may occur in the model selection (CAWLEY; TALBOT, 2010; KRSTAJIC *et al.*, 2014). In this sense, the experimental methodology adopted exploring different techniques, hundred of datasets, nested resamplings and AutoML tools makes experiments more complete and reproducible. All the concepts described in this chapter are applied in the experiments described in next chapters.

CHAPTER

5

TUNING OF SVMS

Support Vector Machines (SVMs) have been successful in numerous applications (KOCH *et al.*, 2012). However, despite their good predictive performance, they are highly sensitive to their HP values. Due to that, SVM HP tuning is still widely studied in literature (BRAGA *et al.*, 2013; DUARTE; WAINER, 2017; HORN *et al.*, 2016; PADIERNA *et al.*, 2017). As mentioned before, different tuning techniques may obtain different HP settings from different hyperspace regions. So, in this chapter, the *HP profile* of the SVMs is investigated. The experiments described in this chapter were published in Mantovani *et al.* (2015a), Mantovani *et al.* (2015b). Experimental results are further used to feed the MtL recommender system described in the Chapter 7.

The next sections are organized as follows: Section 5.1 presents the SVM HP space investigated in experiments; Section 5.2 investigates the budget required for tuning SVMs; Section 5.3 defines the setup for the tuning task; Section 5.4 discusses SVMs HP tuning results; Section 5.5 compare the different tuning techniques when performed on SVMs; Section 5.6 shows a new approach for the SVMs default HP values; and Section 5.7 presents the final remarks of the chapter.

5.1 SVM HP space

The SVM HP space used in the experiments is presented in Table 6. For each HP, the table shows its symbol, name, range, scale (when applied to values) and default values provided by LibSVM (CHANG; LIN, 2011). Here, only the Radial Basis Function (RBF) kernel is considered since it achieves good performances in general, may handle nonlinear decision boundaries, and has less numerical difficulties than other kernel functions (e.g., the values of the polynomial kernel may be infinite) (HSU; CHANG; LIN, 2007). For C and γ, the selected range covers the HP space also investigated in studies as Gomes *et al.* (2012), Reif, Shafait and Dengel (2012), Ridd and Giraud-Carrier (2014). LibSVM default values are $C = 1$, and $\gamma = 1/N$, where N is the number of features of the dataset under analysis.

Table 6 – SVM hyperspace used in experiments. For each hyperparameter is shown its description, range, and scale that values are calculated.

Symbol	hyperparameter	Range	Scale	Default
k	kernel	{RBF}	-	RBF
C	regularized constant	$[2^{-15}, 2^{15}]$	log	1
γ	width of the kernel	$[2^{-15}, 2^{15}]$	log	$^1/_N$

5.2 Defining budget size

The first crucial experimental choice is the budget size (b), i.e., how many HP settings the techniques should evaluate to find good solutions. As such, a different progressive number of budget sizes were investigated to check whether increasing the number of evaluations (individuals in the case of population-based methods) leads to a higher predictive performance. Since just a simple estimation was desired, only 20 datasets (a heterogeneous control group) and GA technique were used in this preliminary experiment. Four scenarios were generated: budget size of 200, 2.500, 5.000 and 10.000 evaluations per execution. These scenarios are illustrated in Table 7.

Table 7 – Budget scenarios investigated for SVM HP tuning.

Scenario	Number of Evaluations	Population Size	Number of Generations
sc-200	200	10	20
sc-2.5k	2.500	100	25
sc-5k	5.000	100	50
sc-10k	10.000	100	100

Figure 7 summarizes the experimental results obtained in each budget scenario. Performance values are averaged over 30 repetitions. This figure also shows *validation* and testing (test) accuracies of the induced models. Datasets at the x-axis are identified by their correspondent OpenML ids[1].

Results show that similar BAC values were obtained in all scenarios, with a low standard deviation associated with the resultant models. The biases (the difference between validation and testing accuracies) are also very small even in the first scenario with fewer evaluations. Increasing the number of evaluations does not necessarily improve the performance obtained, and few evaluations are enough to find suitable solutions. A possible reason for this general behavior is the small number of HPs being tuned and simple landscape of the hyperspaces.

Figure 8 illustrates GA tuning behavior in all budget scenarios for the dataset "`LED-display-domain-7dig`
Left side figures show the SVM HP space covered by the GA executions, with default HP values

[1] Datasets' descriptions may be consulted in Tables 18, 19 and 20 at Appendix A.
[2] LED-display-domain-7digit dataset has: 10 classes, 500 examples and 8 attributes - <https://www.openml.org/d/40496>.

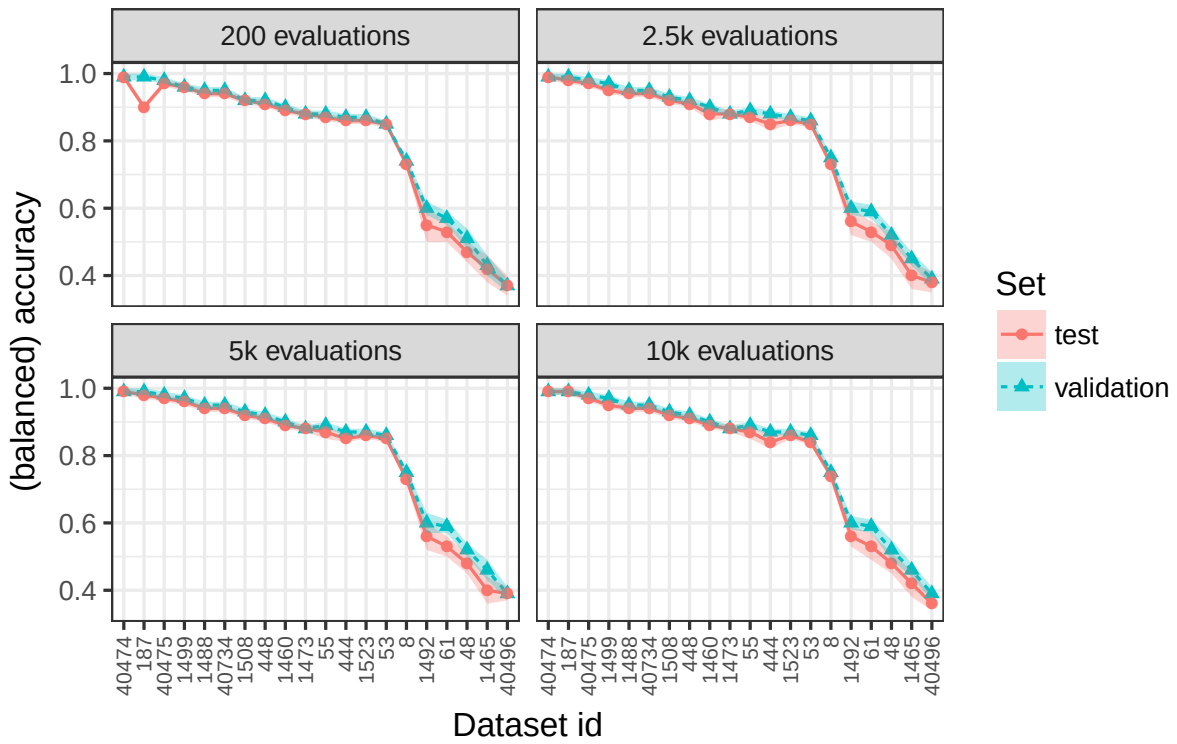


Figure 7 – BAC values of the induced models obtained in each budget scenario. Results are averaged over 30 repetitions. Figure adapted from (MANTOVANI *et al.*, 2015a).

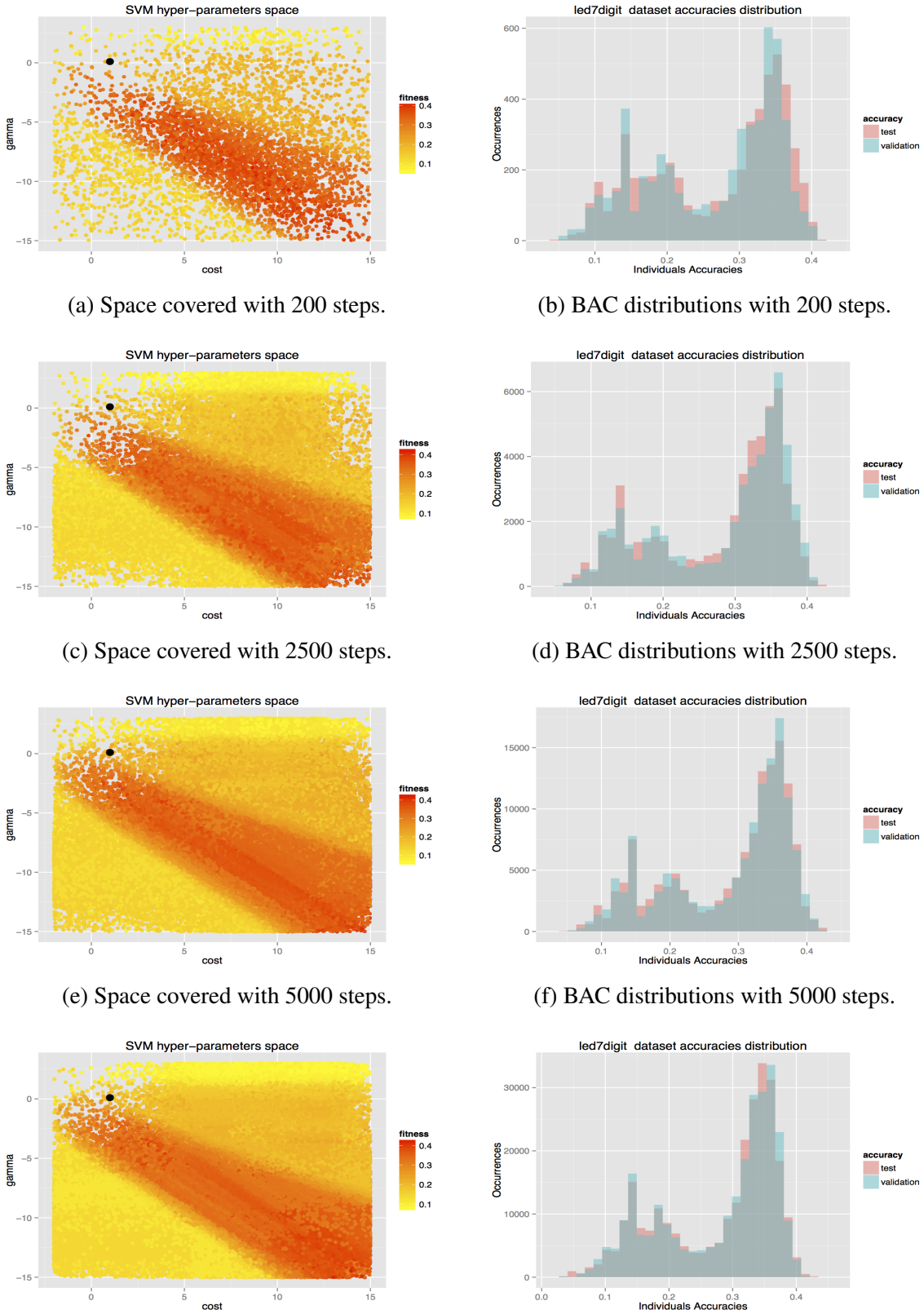
represented by a black point. The darker the point, the better is the predictive performance reached on it.

Figures at the right side depict the predictive performances obtained in validation and test sets during the optimization process. Bars in pink indicate the number of individuals of the testing set that reached a certain level of performance. The light blue bars denote those from the validation set. The region in gray is where the two distributions overlap. The higher the overlap, the lower is the bias technique to estimate the performance of models in both evaluation sets.

Increasing the number of evaluations provides a better understanding of the hyperspace and where the best solutions to the problem are located. SVM HPs are dependent on each other, so there is not a single but a region of "best" HP settings (identified by orange). For this particular dataset, default HP values stay out this region but close to the border. Hence, an improvement of 0.278 concerning BAC is observed when tuning is performed.

Regarding performance distributions, when the number of evaluations is increased, distributions tend to overlap and the bias in the CVs decreases. However, there is a greater number of points that achieve the same predictive performance. This can be observed mainly in executions up to 2000 evaluations. This behavior can be seen by both the density of dots in the heatmaps and the number of individuals with the same accuracy in histograms.

Since good HP settings were obtained by a reduced budget size with neither loss of performance nor biasing results, a budget size of $b = 200$ evaluations was adopted for all the



(a) Space covered with 200 steps.

(b) BAC distributions with 200 steps.

(c) Space covered with 2500 steps.

(d) BAC distributions with 2500 steps.

(e) Space covered with 5000 steps.

(f) BAC distributions with 5000 steps.

(g) Space covered with 10000 steps.

(h) BAC distributions with 10000 steps

Figure 8 – HP space covered by GAs in the "LED-display-domain-7digit" dataset. Figure from Mantovani *et al.* (2015a).

tuning tasks with SVMs.

5.3 Tuning setup

A total of five techniques were selected to check how tuning affects the predictive performance of the induced models: GS, the simplest approach for tuning; RS (ANDRADOTTIR, 2015), baseline recommended by Bergstra and Bengio (2012); and three meta-heuristics commonly explored in related literature - GA (GASCóN-MORENO *et al.*, 2011), PSO (LIN *et al.*, 2008) and EDAs (PADIERNA *et al.*, 2017). Experiments were performed in a subset of 70 from the datasets randomly selected from data collection.

Performance assessment was done using Nested-CVs: an inner 3-CV for the fitness evaluation, and a 10-CV outer loop for model assessment. As discussed before, since data collection contains a wide variety of binary and multiclass classification problems, the BAC measure was used to assess induced models and also guide the search of the tuning techniques. Default HPs values provided by `LibSVM` were also used as baselines to measure the improvement obtained by tuning techniques.

Table 8 – Setup of the SVM HP tuning experiments.

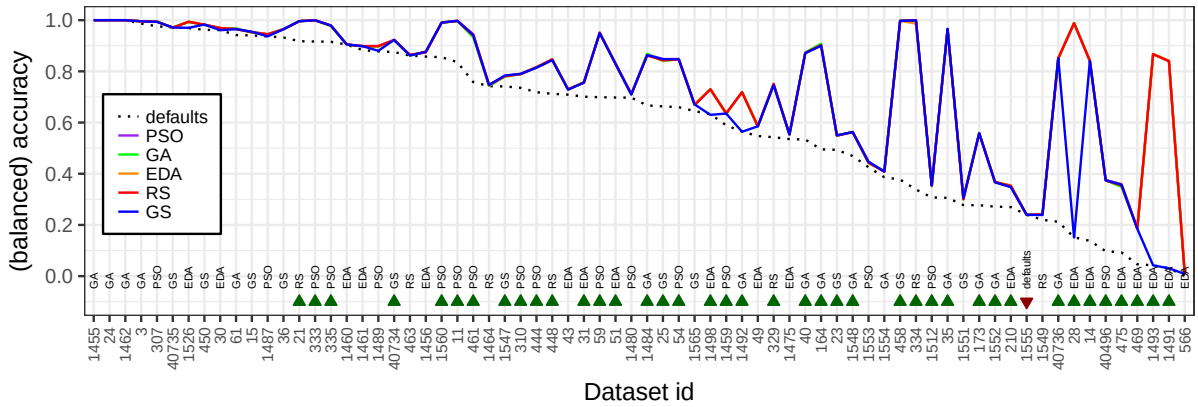
Element	**Method**	**R package**
HP-tuning techniques	Grid Search (GS)	`mlr`
	Random Search (RS)	`mlr`
	Genetic Algorithm (GA)	`GA`
	Particle Swarm Optimization (PSO)	`PSO`
	Estimation of Distribution Algorithm (EDA)	`copulaedas`
Algorithm	SVM algorithm	`e1071`
Inner resampling	3-fold cross-validation	`mlr`
Outer resampling	10-fold cross-validation	`mlr`
Optimized measure	BAC	`mlr`
Evaluation measure	{BAC, Optimization paths }	`mlr`
Repetitions	30 times with different seeds	-
	seeds = $\{1,\ldots,30\}$	-
Baseline	Default values (DF)	`e1071`

5.4 Performance improvement

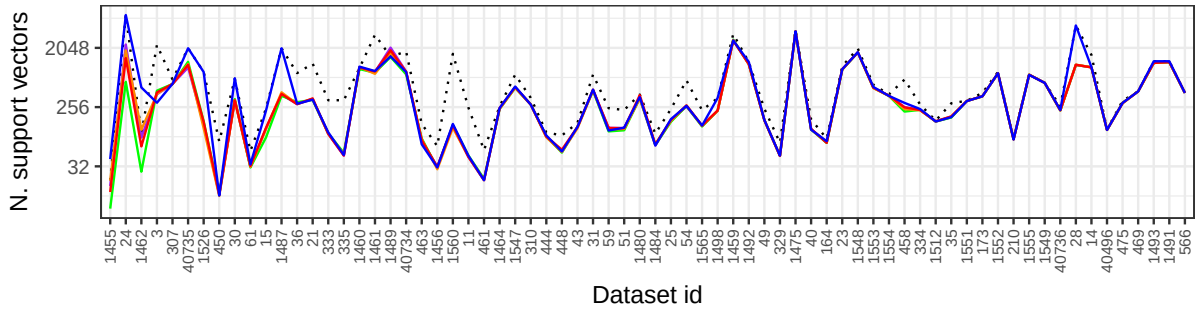
HP tuning results for SVMs in 70 datasets are depicted in Figure 9. Top figure (9a) shows the average BAC values obtained by the tuning techniques and defaults averaged over 30 executions. The x-axis identifies datasets by their `OpenML` ids, listing them decreasingly by the BAC performances obtained using default HP values[3]. For each dataset, the name of the tuning technique that resulted in the best predictive performance is also shown above the x-axis.

[3] The corresponding dataset names may be seen in Tables 18, 19 and 20 at Appendix A.

The Wilcoxon paired-test was applied to assess the statistical significance of the results obtained by this *best* technique when compared to the results using default HP values. The test was applied to the solutions obtained from the 30 repetitions (with $\alpha = 0.05$). An upper green triangle (▲) at x-axis identifies datasets where statistically significant improvements were detected after applying the HP tuning technique. On the other hand, every time a red down triangle (▼) is presented, the use of defaults was statistically better than the use of tuning techniques.



(a) Average BAC performance.



(b) Average number of support vectors. Y-axis values are in log_2 scale.

Figure 9 – HP tuning results for the SVM algorithm.

In the figure, one may notice that all the techniques performed similarly, with their curves mostly overlapped. It should be noted that there are some cases where the tuning does not provide significant improvement. This mainly happened for some artificial and biological datasets, like those with ids = {49, 1459, 1464, 1551, 1555}. Furthermore, in these datasets when the improvement was significative, GS often presented performances visually inferior to other techniques in at least five problems (as may be observed at the blue curve behavior). Overall, the HP tuning statistically improved SVMs' performances in $15/94$ ($\approx 57\%$) of the datasets. Default HP values were significantly better in a single dataset, and in the remaining $29/70$ ($\approx 41\%$) datasets, the approaches "tied"[4], i.e., there was a waste of time trying to optimize the HPs.

[4] A tie is considered when tuned models are not statistically better than those obtained by default HP settings.

Figure 9b shows the average number of support vectors of the models obtained by each tuning technique. Values at the y-axis are in log_2 scale. X-axis datasets are presented in the same order as in the figure above. In the figure, it can be noticed that default HP values induced models with a higher number of support vectors. These models required more points in the hyperplane to define the decision boundaries. On the other hand, when tuning improved SVMs, the HP settings found generated models with a reduced number of support vector. In summary, when tuning improved SVMs, it also reduced the number of required support vectors.

5.5 Comparing techniques

Results from the previous section show that no technique was the best on all datasets. Figure 10 shows a simple win-tie-loss comparison, highlighting this fact by a high number of ties between techniques. The experiments have shown that, overall, default values are good initial values for the SVM HPs. Nevertheless, in most cases, tuning has resulted in a substantial improvement regarding predictive performance.

Figure 11 presents the CD diagram comparing tuning techniques applied in SVMs. Techniques are connected when there is *no* statistically significant differences between them. Both scenarios, with $\alpha = 0.05$ and $\alpha = 0.1$ presented the same analysis. All the tuning techniques performed better than HP values considering the selected datasets. PSO and EDA were ranked first, followed by RS, GA and GS. In general, techniques did not present statistical differences regarding performance between each other. The only exception is when PSO and GS are compared, with the first surpassing the last.

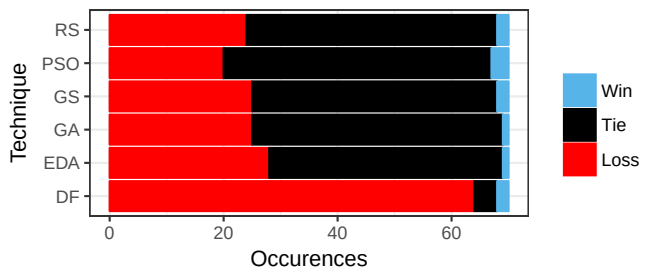
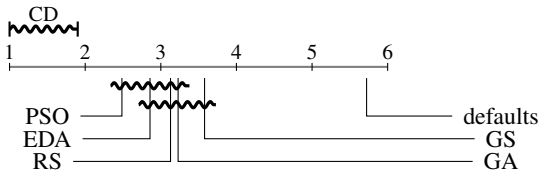
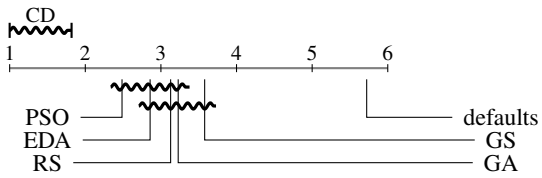


Figure 10 – Win-Tie-Loss of the tuning techniques over the datasets.

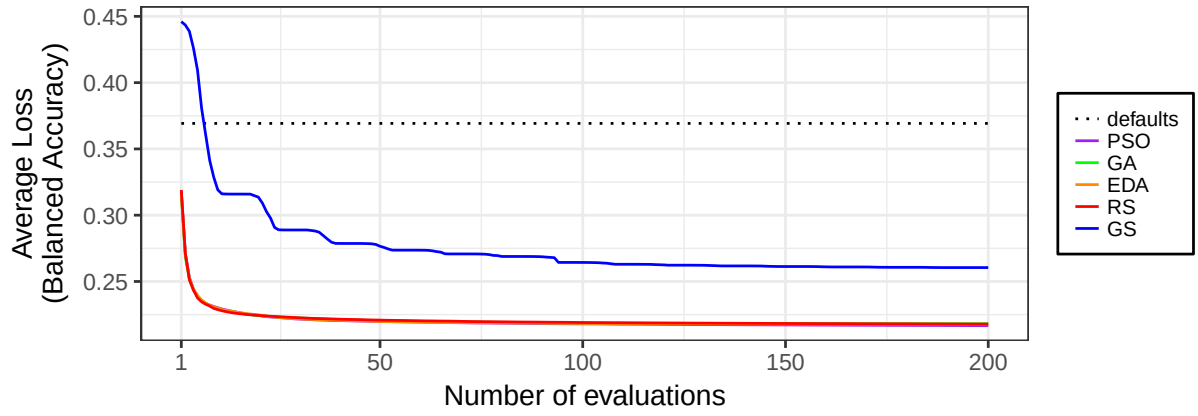


(a) SVM CD diagram with $\alpha = 0.05$.

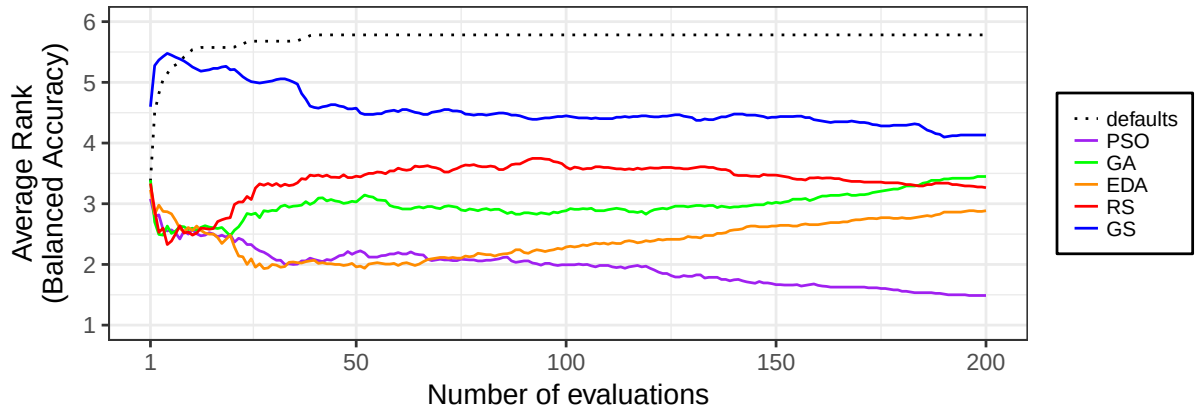


(b) SVM CD diagram with $\alpha = 0.1$.

Figure 11 – Comparison of the BAC values of the HP tuning techniques for SVMs according to the Nemenyi test. Groups of techniques that are not significantly different are connected.



(a) Average loss curve.



(b) Average ranking curve.

Figure 12 – Loss curves for SVMs across datasets.

These results suggest that, for SVMs, a simple and fast technique such as RS can obtain similar results compared to more sophisticated techniques (e.g., population-based methods) and the traditional GS. The reason could be due to the low dimensional space investigated since just two HPs are being adjusted.

Figure 12 shows the average loss curves over the number of evaluations of the five tuning techniques evaluated. The top figure shows the average loss regarding predictive performance, while the bottom picture shows the average rank of the techniques. Results are averaged over the 70 datasets. As expected, defaults are the worst approach. Figure 12a highlights the GS *deficiency*, with all the other techniques performing better than it. Among each other, meta-heuristics and RS presented similar gain of performance across data collection.

When the measure is the average rank (Figure 12b) it is possible to see which technique is best for each budget size. GS was always the worst ranked since the beginning. PSO and EDA produced better models earlier and stayed close until 100 evaluations. After this period, PSO improved more using the shared information between particles to reach new regions of the hyperspace, while EDA got stuck in a common distribution.

GA and RS always stay in the middle. It might be the case GA failed to change its individuals sufficiently, and its parametrization may have affected it. In the end, RS approaches the EDA and GA performances, randomly selecting new settings at the space. Overall, loss curves go in the direction of what was presented in CD diagrams, showing that even RS would be an interesting technique for the SVM HP tuning problem.

5.6 Optimization of new default HP values

Although HP tuning may result in more accurate models, the optimization of these HPs usually has a high computational cost, since a large number of candidate solutions needs to be evaluated. An interesting alternative would be to generate a default HP setting by optimizing these HPs over several datasets rather than doing that individually. The optimized common values may improve the model performance when compared with the use of the default values. These optimized values can also reduce the computation cost to induce models when compared with the optimization for each dataset.

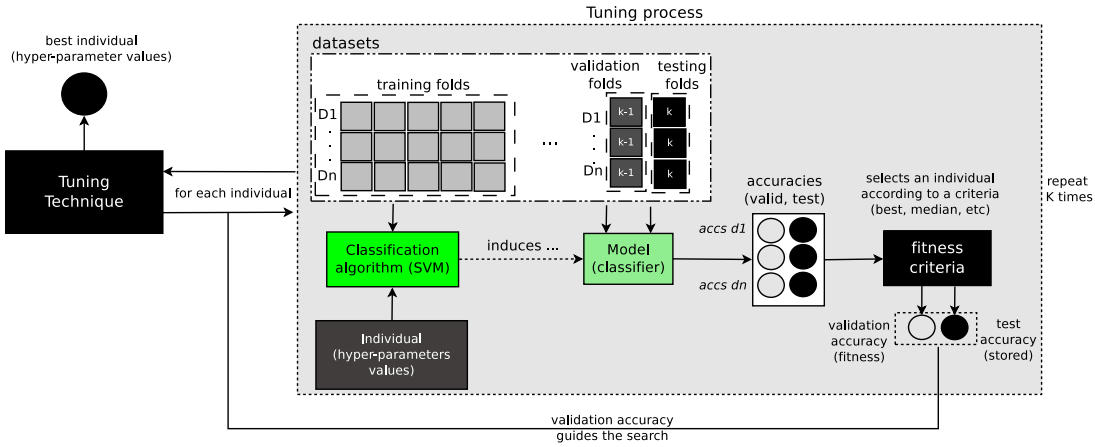


Figure 13 – SVM HP tuning process with multiple datasets. Figure adapted from Mantovani *et al.* (2015b)

Following this hypothesis, the predictive performance of SVMs induced by a PSO algorithm generating these common HPs were evaluated. Figure 13 illustrates the experimental tuning methodology adapted to handle multiple datasets at the same time. In general, there are no big differences from the experimental methodology presented before in Section 4.2. All the datasets were into training and testing partitions.

The main difference is how the tuning technique evaluates candidate solutions and guides the search. Every time a new HP setting is evaluated, SVMs are induced for each one of the datasets, generating an array of performance values. Then, a criterion is applied in this array to select a HP setting which will designate the fitness value. The criterion may be any function, but in experiments, the median value of the predictive performances was defined as the fitness value of the individual. The tuning technique selected was the PSO, since it was top-ranked when compared to the other techniques. The same control group of 20 datasets used before was

selected to optimize the common HP values. The remaining 70 datasets were used to evaluate the approach. New default optimized HP values ("def.Opt')' were compared with those provided by `LibSVM` and `WEKA` tools[5].

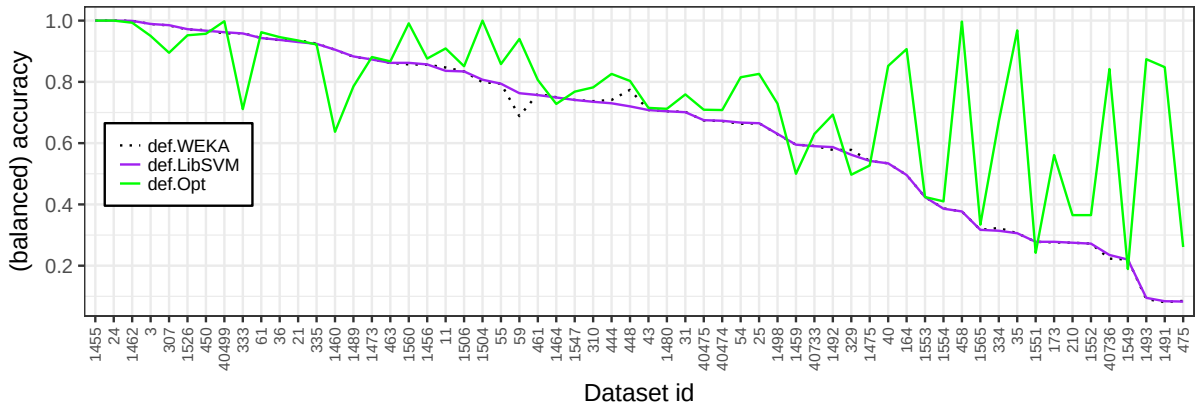


Figure 14 – HP tuning results for the default optimized HP settings.

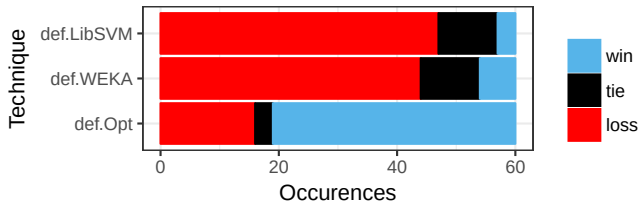


Figure 15 – Win-Tie-Loss of the default HP values over the validation datasets.

Figure 14 shows the HP tuning results for the default HP optimized approach. Datasets on the x-axis are decreasingly ordered according to the `LibSVM` default HP performances. Unsurprisingly, the optimized common HP values reached the best predictive performance. As expected, the new optimized settings fell in the optimum region for most of the datasets, surpassing the traditional approaches for default values. When compared with the individual tuning processes, in most of the datasets the tuning is still required. However, the new optimized default values could increase the number of cases where tuning could be avoided.

Figure 15 presents the win-tie-loss statistics regarding these three approaches. Default HP settings provided by ML tools performed quite similar, with the `WEKA` approach prevailing in a few more cases. Moreover, models induced by the optimized default HPs were the best in 60% of the cases. Figure 16 complements the analysis presenting the CD diagram comparing the HP default approaches. Using different α values did not change results. The new approach was the top-ranked approach in the datasets selected for evaluation, presenting statistical differences with both baselines.

The `WEKA` default HP settings are constant for any dataset. Thus, they may not be necessarily located in the region with the best solutions. The `LibSVM`, although it uses a formula

[5] The default HP values for SVMs provided by `WEKA` can be find at the following page: <http://weka.sourceforge.net/doc.dev/wekfollowingfiers/functions/SMO.html>

(a) SVM CD diagram with $\alpha = 0.05$.

(b) SVM CD diagram with $\alpha = 0.1$.

Figure 16 – Comparison of the BAC values of the default HP approaches for SVMs according to the Nemenyi test. Groups of techniques that are not significantly different are connected.

to define γ, it falls into the same problem. The optimized default settings, even being optimized simultaneously for many datasets, are mostly near the optimum regions because it considers the dependent relationship of the SVM HPs. It is true they will not always be the best choice, but its performance comes similar to the tuning techniques. Once SVMs are very sensitive algorithm, it is an interesting alternative when more accurate models are desirable with a lower computational cost.

5.7 Chapter Remarks

In this chapter, the *HP profile* of SVMs was investigated. SVMs are highly sensitive to their HP values, and thus still widely studied in recent literature (PADIERNA *et al.*, 2017; LORENA *et al.*, 2018), but few of them compare different techniques when performing HP tuning (HORN *et al.*, 2016).

Experiments were carried with simple (RS, GS) and population-based techniques (GA, PSO, EDA) over a total of 90 public datasets. The experimental results provided the *HP profile* of the SVMs, showing that only a few iterations are required to reach good solutions in the SVM hyperspace.

Further, all the techniques performed statistically better than models induced with default HP settings. Overall, there were no statistical differences among each other. Thus, a simple RS performed similarly than meta-heuristics regarding of the average loss over data collection. This study was the first showing the effectiveness of a simple RS technique for the SVM HP tuning (MANTOVANI *et al.*, 2015a).

Besides the default values proposed by ML tools, an optimization technique to define new default HP values based on a group of datasets was developed (MANTOVANI *et al.*, 2015b). The use of this new set of HP values, referred to as optimized defaults, produced significantly better models than the default values suggested by ML tools.

All the data generated from tuning tasks is meta-knowledge that will be used by the MtL recommender system proposed in Chapter 7. In the next chapter, the *HP profile* of DT induction algorithms will be investigated.

CHAPTER

6

TUNING OF DECISION TREES

The HP tuning task is usually investigated for "black-box" algorithms, such as Artificial Neural Network (ANN) and SVMs (as reported in the previous chapter), but not for DTs. There are some prior studies investigating the evolutionary design of new DT induction algorithms (BARROS *et al.*, 2012; BARROS; CARVALHO; FREITAS, 2015), but only a few on HP tuning for them (REIF; SHAFAIT; DENGEL, 2011; MOLINA *et al.*, 2012; REIF *et al.*, 2014).

Many ML algorithms able to deal with classification tasks can be found in the literature. Although high predictive accuracy is the most frequently used measure to evaluate these algorithms, in many applications, easy interpretation of the induced models is also a requirement. Good predictive performance and model interpretability are found in one of the most successful sets of classification algorithms: DT induction algorithms (ROKACH; MAIMON, 2014).

In this chapter, the *HP profile* of the DT induction algorithms is investigated. The experiments described in this chapter were published in Mantovani *et al.* (2016), Mantovani *et al.* (2018a). Experimental results are further used to feed the MtL recommender system described in the Chapter 7.

The next sections are organized as follows: Section 6.1 presents DT HP spaces investigated in the experiments; Section 6.2 studies the budget required for tuning DTs; Section 6.3 describes the setup for the tuning task; Section 6.4 discusses the results obtained from the DT HP tuning; Section 6.5 compares different tuning techniques on DTs; Section 6.6 uses a fANOVA analysis to estimate DT HPs relative importance; Section 6.7 investigates how reducing HP spaces affect the performance of induced DTs; and, Section 6.8 presents the final remarks of the chapter.

6.1 DT HP spaces

The experiments were performed considering the HP tuning of three DT induction algorithms:

1. the `J48` algorithm, a `WEKA`[1] (WITTEN; FRANK, 2005) implementation of the C4.5 algorithm;
2. the `rpart` implementation of the CART (BREIMAN *et al.*, 1984) algorithm; and
3. and the CTree algorithm (HOTHORN; HORNIK; ZEILEIS, 2006).

These algorithms were selected due to their wide acceptance and use in many ML applications (BARROS *et al.*, 2012; JANKOWSKI; JACKOWSKI, 2014; ROKACH; MAIMON, 2014). The first two algorithms are among the most used in ML, especially by non-expert users (WU; KUMAR, 2009), and the third one is a more recent implementation that uses statistical tests for splits, like the classical Chi-squared Automatic Interaction Detector (CHAID) algorithm (KASS, 1980). The correspondent HP spaces investigated are described in Table 9.

Originally, J48 has ten tunable HPs[2]: all presented at Table 9 and the HP "U", which enables the induction of unpruned trees. Since pruned trees look for the most interpretable models without loss of predictive performance, this HP was removed from the experiments, and just pruned trees were considered. For CTree, all the statistically dependent HPs were kept out, since their effects were previously studied and the default choices were robust for a wide range of problems (HOTHORN; HORNIK; ZEILEIS, 2006), thus the non-statistically dependent HPs were selected. Regarding CART, all the tunable HPs in `rpart` were selected.

For each HP, Table 9 also shows the allowed range of values, default values provided by the correspondent packages, and its constraints for setting new values. The "M" HP values were the same used in Reif, Shafait and Dengel (2011). The range of the pruning confidence (C) HP was adapted from Reif *et al.* (2014), because the algorithm internally controls the parameter values, does not allow some values near zero or $C \geq 0.5$.

6.2 Defining budget

Following the same principle adopted for SVMs, the first experiment investigates whether it is possible to find suitable HP settings by using a reduced budget[3] when tuning DTs. Thus, experiments were performed with the J48 algorithm and the tuning techniques previously evaluated for SVMs, except GS.

[1] <http://www.cs.waikato.ac.nz/ml/weka/>
[2] <http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html>
[3] In this thesis, the budget corresponds to the number of evaluations.

Table 9 – DT HP spaces explored in experiments. The `J48` nomenclature is based on the `RWeka` package, the CART terms is based on the `rpart` package, and the CTree terms based on the `party` package.

Algo	Symbol	Hyperparameter	Range	Type	Default	Conditions
J48	C	pruning confidence	$(0.001, 0.5)$	real	0.25	R = False
J48	M	minimum number of instances in a leaf	$[1, 50]$	integer	2	-
J48	N	number of folds for reduced error pruning	$[2, 10]$	integer	3	R = True
J48	O	do not collapse the tree	{False, True}	logical	False	-
J48	R	use reduced error pruning	{False, True}	logical	False	-
J48	B	use binary splits only	{False, True}	logical	False	-
J48	S	do not perform subtree raising	{False, True}	logical	False	-
J48	A	Laplace smoothing for predicted probabilities	{False, True}	logical	False	-
J48	J	do not use MDL correction for info gain on numeric attributes	{False, True}	logical	False	-
CART	cp	complexity parameter	$(0.0001, 0.1)$	real	0.01	-
CART	minsplit	minimum number of instances in a node for a split to be attempted	$[1, 50]$	integer	20	-
CART	minbucket	minimum number of instances in a leaf	$[1, 50]$	integer	7	-
CART	maxdepth	maximum depth of any node of the final tree	$[1, 30]$	integer	30	-
CART	usesurrogate	how to use surrogates in the splitting process	$\{0, 1, 2\}$	factor	2	-
CART	surrogatestyle	controls the selection of the best surrogate	$\{0, 1\}$	factor	0	-
CTree	mincriterion	the value of the statistic test (1 - p-value) to be exceed for a split occurrence	$(0.9, 0.999)$	real	0.95	-
CTree	minsplit	minimum sum of weights in a node for a split occurrence	$[1, 50]$	integer	20	-
CTree	minbucket	minimum sum of weights in a leaf	$[1, 50]$	integer	7	-
CTree	mtry	number of input variables randomly sampled as candidates at each node for random forest like algorithms	$[p^{0.1}, p^{0.9}]$	real	0	-
CTree	maxdepth	maximum depth of any node of the final tree	$[1, 30]$	integer	no restriction	-
CTree	stump	a stump (a tree with three nodes only) is to be computed	{False, True}	logical	False	-

The J48 algorithm was chosen for this task because it presents a higher number of HPs among the DT induction algorithms. Hence, it would require a higher number of iterations to find suitable HP values. As the J48 hyperspace has nine HPs, GS can become infeasible due to the size of the search space. Consequently, it was removed from the experimental setup. The tuning methodology adopted is the same described in Chapter 4: tuning is performed with nested-CV resamplings - an inner 3-CV loop to evaluate candidate HP settings, and an outer 10-CV loop to assess models induced by the recommended HP settings. Experiments were performed over 94

datasets, duly identified in the tables at Appendix A.

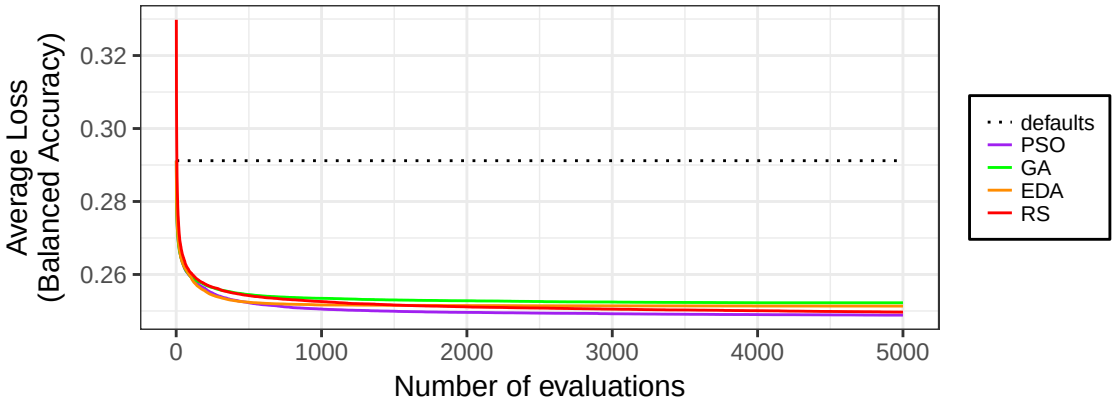


Figure 17 – Loss curves for J48 algorithm across datasets with 5000 evaluations.

Figure 17 presents loss curves when tuning the J48 algorithm with a budget of $b = 5000$ evaluations. Results are averaged over the 94 datasets. The experimental results suggested that all the considered techniques required only around $900 - 1000$ evaluations to converge. The convergence here means the tuning techniques could not reduce their average loss more than $x = 10^5$ until the budget was consumed. Actually, in most cases, the tuning reached its maximum performance after 300 steps. Thus, a budget size of $b = 900$ evaluations is deemed sufficient to perform HP tuning in DT induction algorithms.

6.3 Tuning setup

Once the budget size was defined, experiments were performed to tune all the three DT induction algorithms. A total of six HP tuning techniques were selected to check how tuning affects the predictive performance of the induced DTs:

- the three meta-heuristics explored before with SVMs: GA, PSO and EDA;
- a simple RS technique: as suggested in Bergstra and Bengio (2012), it is a suitable baseline;
- Irace (BIRATTARI *et al.*, 2010): a *racing* technique designed for algorithm configuration problems; and
- SMBO (SNOEK; LAROCHELLE; ADAMS, 2012): a state of the art technique for optimization that employs statistical and/or ML techniques to predict distributions over labels.

Irace and SMBO were added to the experimental setup since they natively handle mixed HP spaces (as it is the case) and allows a direct optimization with many dimensions. Table 10 summarizes the choices made to accomplish the general HP tuning procedure. The experiments

Table 10 – Setup of the DT HP tuning experiments.

Element	Method	R package
HP-tuning techniques	Random Search (RS)	`mlr`
	Genetic Algorithm (GA)	`GA`
	Particle Swarm Optimization (PSO)	`PSO`
	Estimation of Distribution Algorithm (EDA)	`copulaedas`
	Sequential Model Based Optimization (SMBO)	`mlrMBO`
	Iterated F-race (Irace)	`irace`
Decision Trees	J48 algorithm	`RWeka`
	CART algorithm	`rpart`
	CTree algorithm	`party`
Inner resampling	3-fold cross-validation	`mlr`
Outer resampling	10-fold cross-validation	`mlr`
Optimized measure	{Balanced per class accuracy}	`mlr`
Evaluation measure	{Balanced per class accuracy, Optimization paths }	`mlr`
Budget	{900} iterations	
Repetitions	30 times with different seeds	-
	seeds = $\{1,\ldots,30\}$	-
Baseline	Default values (DF)	`RWeka` `rpart` `party`

were carried out using 94 public datasets from `OpenML`[4,5].). Performance assessment was done using Nested-CVs: an inner 3-CV for the fitness evaluation, and a 10-CV outer loop for model assessment. As previously discussed, since data collection contains a wide variety of binary and multiclass classification problems, the BAC measure was used to assess induced models and also guide the search of the tuning techniques. Besides, the default HP values provided by the '`RWeka`', '`rpart`' and '`party`' packages were used as baseline for the experimental comparisons.

6.4 Performance improvements

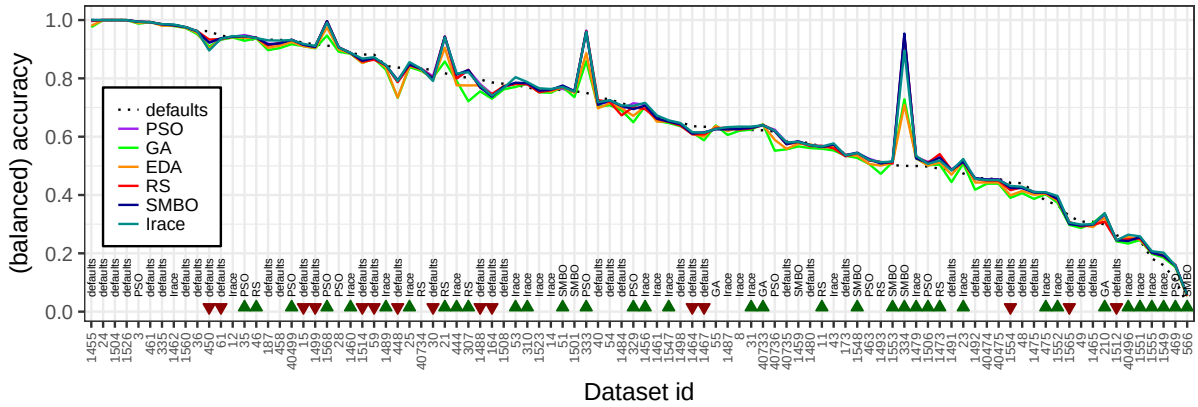
Since all the adopted tuning techniques are stochastic, each one was executed 30 times for each dataset using different seed values. It results in a total of 270.000 = 30 (repetitions) $\times$ 10 (outer-folds) $\times$ 900 (budget) HP settings generated during the search process per dataset.

6.4.1 J48 improvements

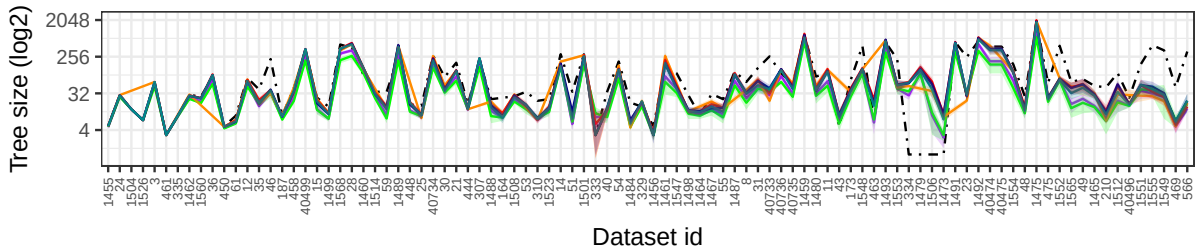
HP tuning results for J48 are depicted in Figure 18. Figure 18a shows the average BAC values obtained by the tuning techniques and default HP values over the datasets. Similarly to the SVMs' plots, the datasets at the x-axis are placed in decreasing order according to their predictive performances using default HP values[6].

For each dataset, the name of the tuning technique that resulted in the best predictive performance is shown above the x-axis. Similarly to the SVM plots, the Wilcoxon paired-test

[4] <http://www.openml.org/>
[5] All datasets. Their main characteristics are specified in Tables 18 to 20 at Appendix A
[6] The corresponding dataset names may be seen in Tables 18, 19 and 20 at A.



(a) Average BAC performance.



(b) Average tree size. X-axis values are in log_2 scale.

Figure 18 – HP tuning results for the J48 algorithm. Figure adapted from Mantovani *et al.* (2018a).

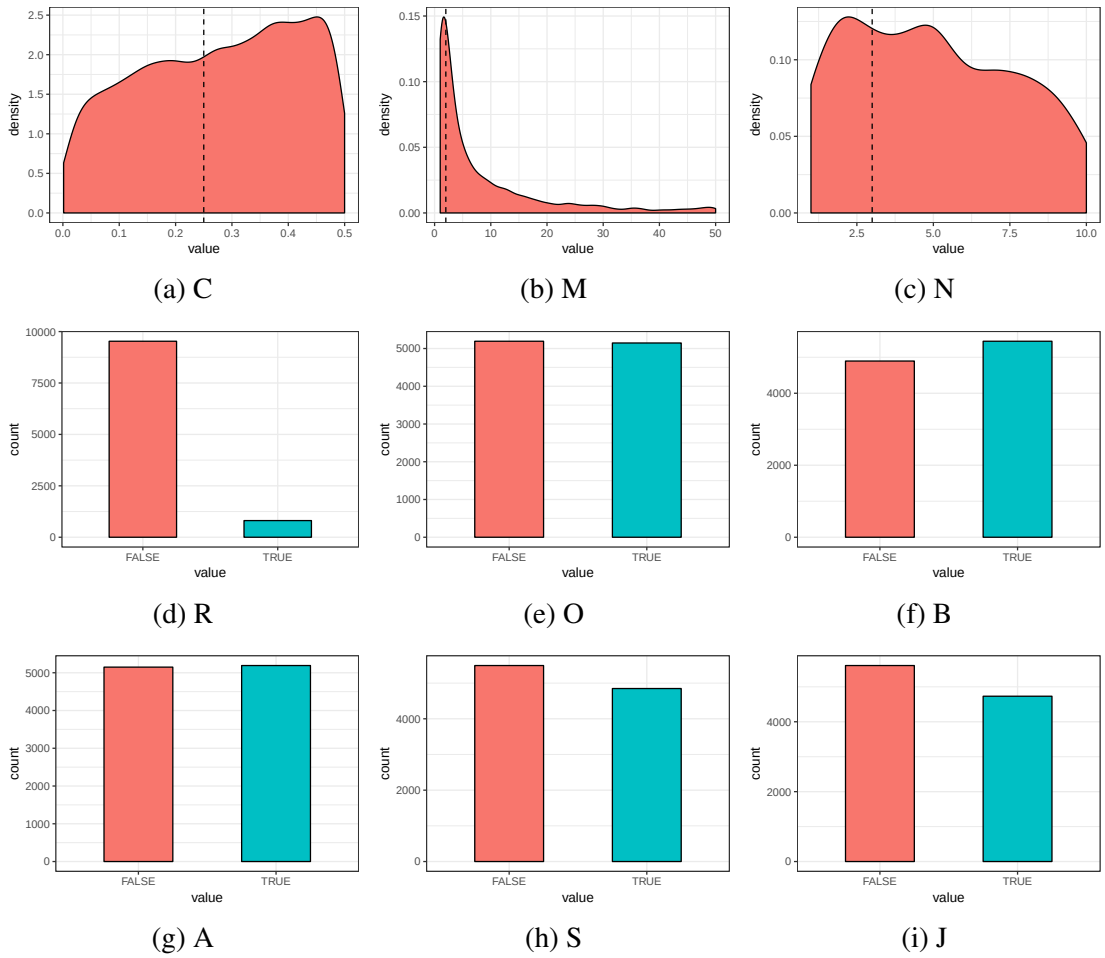
was applied to assess the statistical significance of the results obtained by this *best* technique when compared to the results using default HP values. The test was applied to the solutions obtained from the 30 repetitions (with $\alpha = 0.05$). An upper green triangle (▲) at x-axis identifies datasets where statistically significant improvements were detected after applying the HP tuning technique. On the other hand, every time a red down triangle (▼) is presented, the use of defaults was statistically better than the use of tuning techniques.

J48 tuning results show that all tuning techniques have similar performances, with few exceptions, since most of the curves overlap. In general, there is a small difference in predictive performance regarding the default HP values. Higher improvements may be seen only in a small subset of datasets. Tuned trees were better than those with defaults with statistical significance in $36/94$ ($\approx 38\%$) of the datasets. In most of these situations, the Irace, PSO or SMBO techniques produced the best results. Default HP values were significantly better in $15/94$ ($\approx 16\%$) of the cases, and in the remaining situations ($43/94$ - $\approx 46\%$) the approaches "tied".

Figure 18b shows the average tree size of the final J48 induced models. The tree size measures the number of nodes in the induced model. The interpretability of a tree is mostly dependent on its size. Consequently, larger trees are usually more difficult to understand than smaller trees. Therefore, regarding DT sizes, in most of the cases, the default HP values (dotted black line) induced trees larger than those obtained by tuning techniques. This fact was true whenever default HP values were the best option with statistical significance. For most of the

multiclass tasks with many classes (datasets to the right at the charts), the tuned trees were also smaller than those induced using default HP values. Although small improvements were obtained in terms performance, they were still significant.

Looking at the peaks of improvements due to the use of HP tuning, they were reached when the DTs induced using default HP values were much smaller than those using tuning techniques. This occurred for the datasets with the ids = {51 333, 334, 1568}. These datasets are the only ones in which the techniques presented significant performance differences. The soft computing techniques tend to produce smaller trees than the SMBO and RS techniques.



(a) C (b) M (c) N

(d) R (e) O (f) B

(g) A (h) S (i) J

Figure 19 – Distribution of the J48 HPs found by the tuning techniques. Figure from Mantovani *et al.* (2018a).

To compare default HP setting with the solutions found during the tuning process, and also get useful insights regarding the defaults effectiveness, the distributions of the J48 HPs found during optimization are presented in Figure 19. The numerical default HP values are represented by vertical dashed lines. In the J48 tuning scenario, the largest contrast may be noticed in the ‘`R`’ sub-plot: most of the obtained solutions presented ‘`R=FALSE`’, which disables the use of the “*reduce error pruning*” option and the HP ‘`N`’ (like default HP setting does). The values of ‘`M`’ obtained also tend to the default value in most of the cases (default is $m = 2$). The

other Boolean HPs seem not to influence the induced models since they present a very uniform distribution. Overall, only the "*confidence pruning*"(C) HP seems to influence the search for different solutions, as indicated by Figure 19a.

6.4.2 CART improvements

Figure 20 presents graphical results for the CART tuning. Different from J48, CART was more affected by HP tuning, presenting a different *HP profile*. The use of tuned values improved the predictive performance with statistical significance when compared to the use of default HP values in $62/94$ ($\approx 66\%$) of the datasets. Regarding just the predictive performance of the induced models, the Irace and SMBO were the best optimization techniques. On the other hand, default HP values were the best setup in $14/94$ ($\approx 15\%$) of the cases. In the remaining situations ($18/94 \approx 19\%$) there was no significant statistical improvement using optimized values.

Regarding trees' size, whenever default HP values were statistically better, the trees induced by them have similar or lower sizes than those obtained by the optimization techniques. However, in most cases, tuned HP settings induced trees statistically better and much larger than those created using defaults. Although 'default' trees are simpler, they were incapable of classifying most of the problems properly.

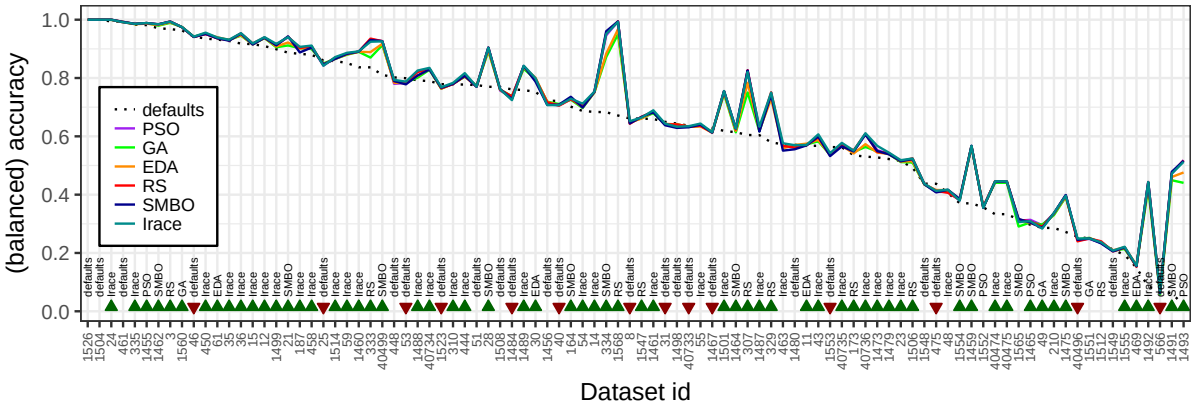
The comparison of the tuning techniques showed results different from those obtained for the J48 algorithm. All the tuning techniques induced trees with similar sizes. However, the DTs induced when Irace was used were slightly larger, and with better predictive performance than those obtained by the other techniques.

The CART HP distributions shown in Figures 20c to 20h present a different behavior than J48 scenario. The CART tuned trees were obtained from values substantially different from the default HP values. This is more evident for the numerical HPs '`cp`', '`minbucket`' and '`minsplit`'. Their values tend to be smaller than defaults. For '`maxdepth`', a wide range of values is tried, indicating a possible dependence on the input problem (dataset). However, the categorical HPs are very uniform, indicating that their choices may not influence the final predictive performance.

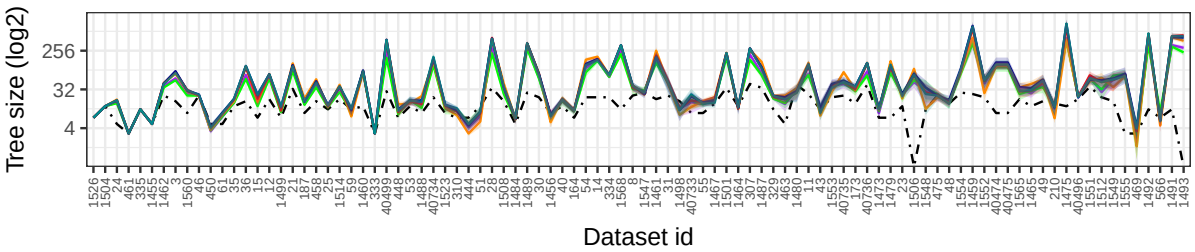
6.4.3 CTree improvements

The CTree HP tuning results are illustrated in Figure 21. Most of the tuning techniques presented similar results, except GA (the green line), which was worse than all the other techniques regarding the predictive performance. Unlike the two previous case studies, the CTree *HP profile* describes an algorithm less influenced by the HP tuning.

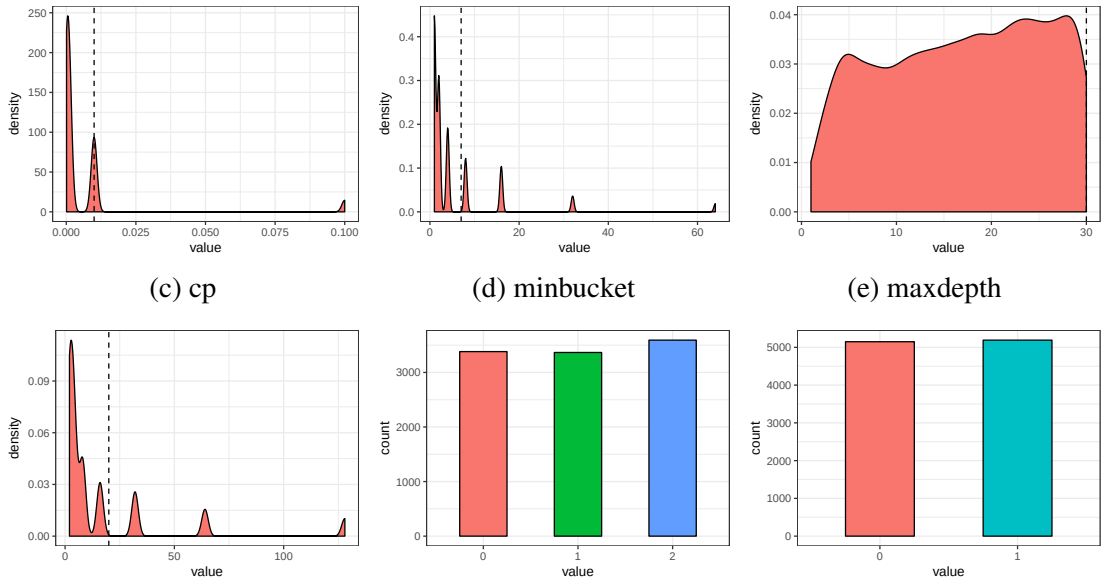
Default HP values induced the best models in $38/94$ ($\approx 40\%$) of the datasets. Tuned values improved the predictive performance of the induced trees in $23/94$ ($\approx 25\%$) of the cases. For the remaining problems $33/94$ ($\approx 35\%$) there was no statistical difference between the use



(a) Average BAC performance.



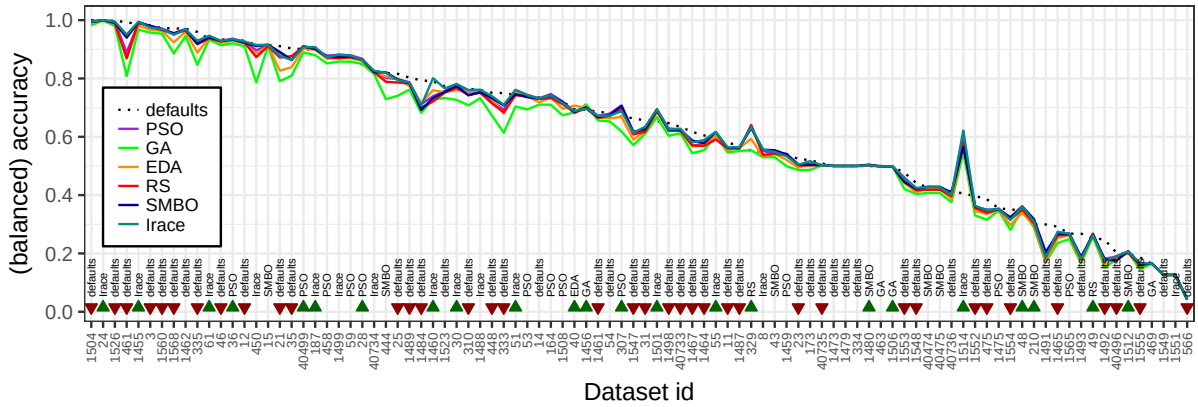
(b) Average tree size. X-axis values are in log_2 scale.



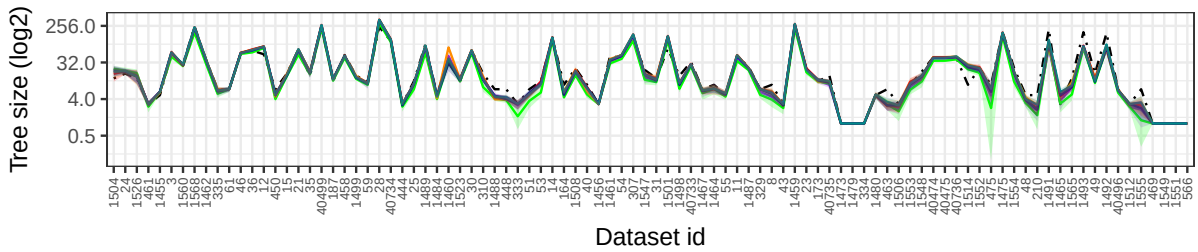
(c) cp (d) minbucket (e) maxdepth

(f) minsplit (g) usesurrogate (h) surrogatestyle

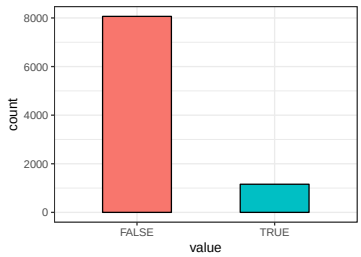
Figure 20 – HP tuning results and distributions for the CART algorithm found by the tuning techniques. Figure adapted from Mantovani *et al.* (2018a).



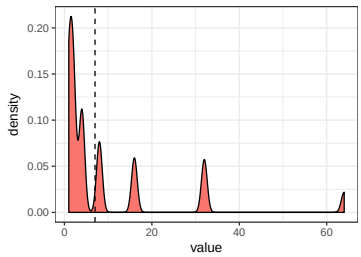
(a) Average BAC performance.



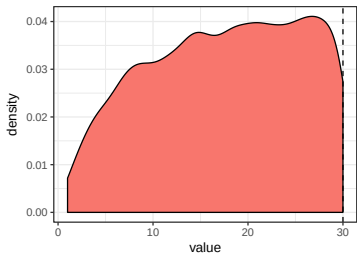
(b) Average tree size. Results are presented in log_2 scale.



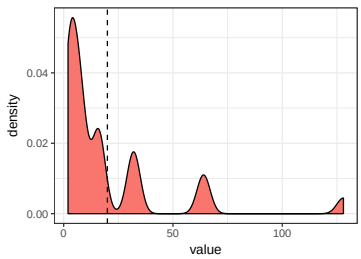
(c) stump



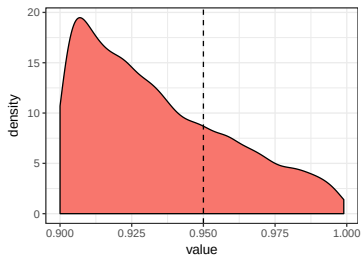
(d) minbucket



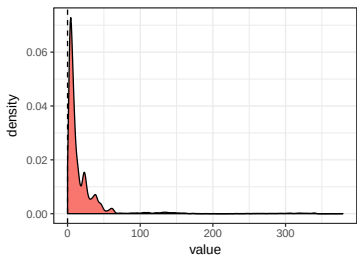
(e) maxdepth



(f) minsplit



(g) mincriterion



(h) mtry

Figure 21 – HP tuning results and distributions for the CTree algorithm found by the tuning techniques. Figure adapted from Mantovani *et al.* (2018a).

of default HP values and settings returned by tuning techniques. Considering the size of the induced trees, tuning techniques did not generate larger or smaller trees than those induced by using default values. There are just a few exceptions, for datasets ids = {79, 57}, where tuned trees improved the predictive performance and are visually larger then defaults. Regarding just the predictive performance, Irace and PSO were the best techniques, followed by the SMBO.

The CTree HPs' distributions are shown in Figures 21c to 21h. Similarly to the CART scenario, all the numerical HPs presented values different from the default values: some of them produced values smaller than default choices ('`minbucket`', '`minsplit`'); another was very close to the default value ('`mtry`'); and all the others varied in a wide range of values ('`maxdepth`', '`mincriterion`'). The categorical HP '`stump`', which enables the induction of a Decision Stump (DS) (tree with just a single level) is mostly set as `stump = FALSE`, like the default setting, having no real impact on the performance differences.

6.5 Comparing techniques

The Win-Tie-Loss statistics of the tuning techniques for the three algorithms are illustrated in Figure 22. In all the scenarios, Irace won the most considering just the tuning techniques. SMBO and PSO were the best technique for a set of problems, with RS being comparable depending on the algorithm under analysis. GA and EDA did not perform well for most datasets, showing some limitations to perform like the other techniques. On the other hand, default HP values are suitable for a high number of datasets, when analyzing CTree and J48 algorithms. It can be seen with the number of wins obtained by defaults when compared to the tuning techniques. The CART algorithm, using the "`rpart`" implementation, is the most sensitive algorithm for tuning.

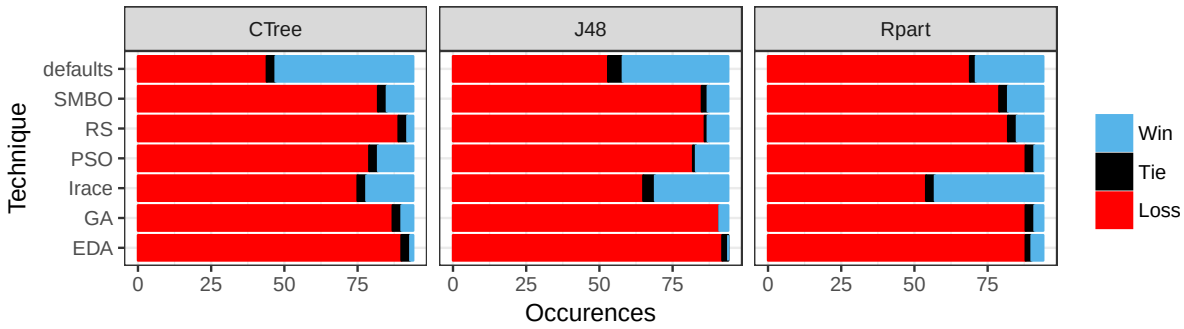
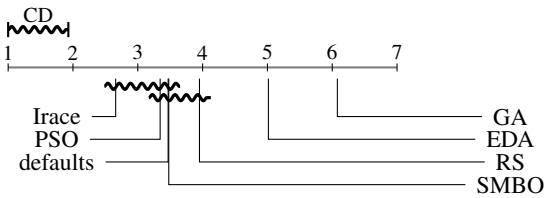


Figure 22 – Win-Tie-Loss of the tuning techniques for the DT induction algorithms

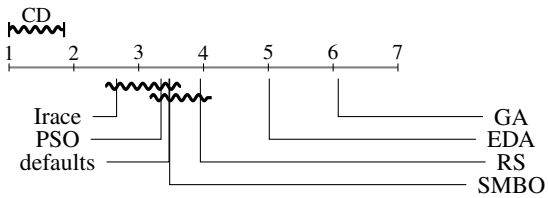
6.5.1 Statistical comparison

The Friedman test with significance levels at $\alpha = 0.05$ and $\alpha = 0.1$, was also used to compare techniques when performing the tuning of DTs. Figure 23 presents the CD diagram for the three DT algorithms. Considering $\alpha = 0.05$, Figure 23a depicts the comparison in J48

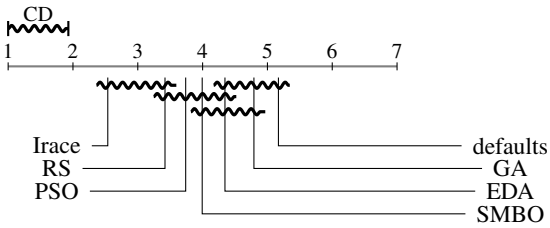
scenario. The first look indicates that there are no statistical differences between the top two best techniques: Irace and PSO. Also, the models induced with default HP values were no statistically better results than Irace, PSO, SMBO, and RS. EDA and GA techniques were statistically inferior to all the others.



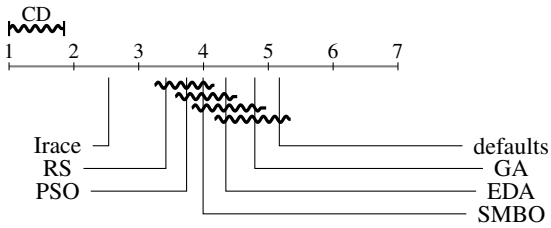
(a) J48 CD diagram with $\alpha = 0.05$.



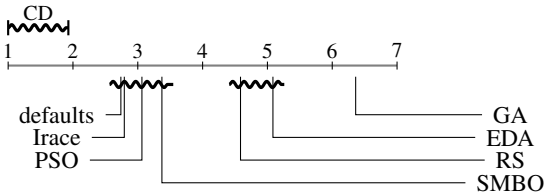
(b) J48 CD diagram with $\alpha = 0.1$.



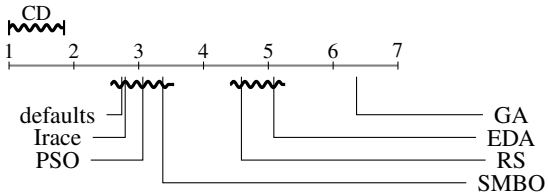
(c) CART CD diagram with $\alpha = 0.05$.



(d) CART CD diagram with $\alpha = 0.1$.



(e) CTree CD diagram with $\alpha = 0.05$.



(f) CTree CD diagram with $\alpha = 0.1$.

Figure 23 – Comparison of the BAC values of the HP tuning techniques for DTs according to the Nemenyi test. Groups of techniques that are not significantly different are connected. Left charts show results with $\alpha = 0.05$, while right charts show comparisons with $\alpha = 0.1$. Figure from Mantovani *et al.* (2018a).

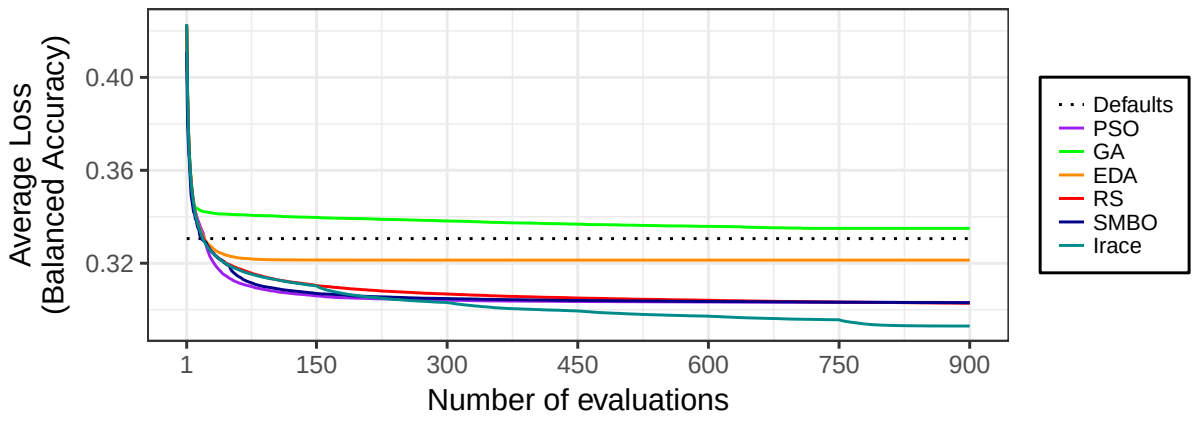
For the CART algorithm (Figure 23c), the best-ranked technique over all datasets was Irace, followed by RS, though with no statistical difference between their results. DTs induced with default HP values obtained the worst performance, being statistically comparable only with GA and EDA.

CTree's CD-diagram for $\alpha = 0.05$ is shown in Figure 23e. The defaults HP values were ranked first, followed by Irace, PSO and SMBO techniques. However, there is no statistical difference between them. The RS and EDA compose the second group of techniques. They do not present statistical differences between them, but they present difference concerning the first group of techniques. Finally, the GA technique was statistically worst than all the other techniques.

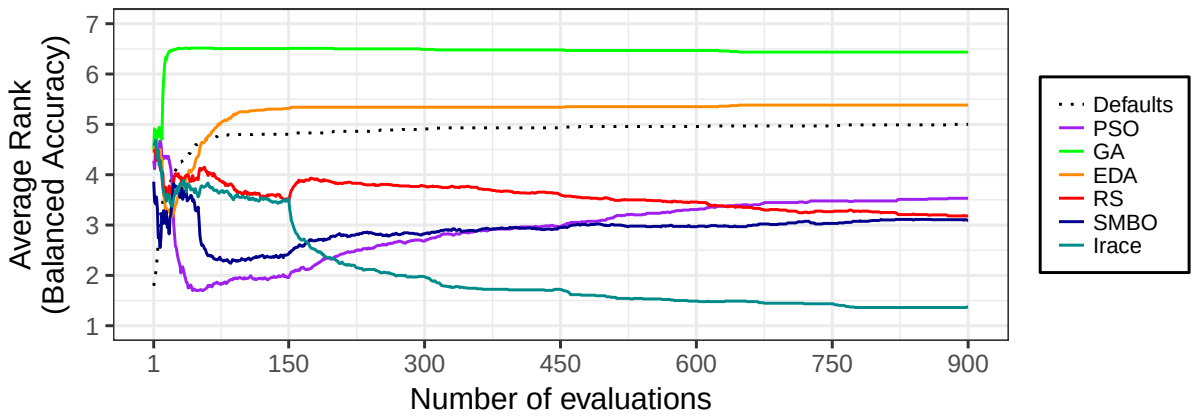
It is worth mentioning that Irace was the best-ranked tuning technique for all the algorithms even though the statistical tests did not show significant differences between Irace

and PSO (J48, CTree), and between Irace and RS (CART). When a larger $\alpha = 0.1$ value was considered (with CD= 0.848), there were no changes in J48 and CTree scenarios. However, regarding CART performances, Irace statistically outperformed all the other techniques, as it can be seen in Figure 23d.

6.5.2 Loss curves comparison



(a) Average loss curve.

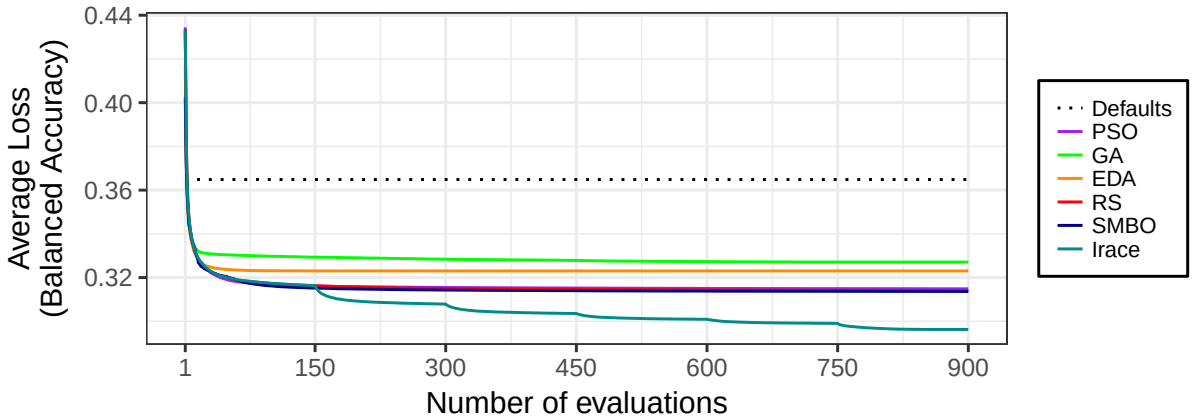


(b) Average ranking curve.

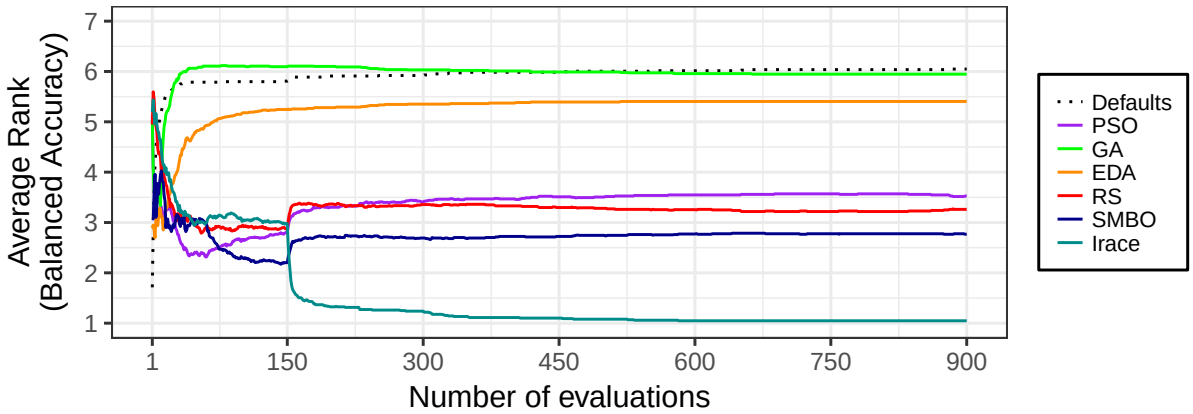
Figure 24 – Loss curves for the J48 algorithm across datasets.

Similarly to SVMs, the J48 average loss curves are shown in Figure 24. The top figure shows the average loss regarding the BAC predictive performance, while the bottom chart shows the average rank of the techniques during the budget consumption. Results are aggregated over all the 94 datasets.

Referring to the average loss, Irace presented the lowest aggregated value among techniques. It is followed by SMBO, PSO and RS, which performed similarly. EDA and GA behaved closely to the default HP settings, but just GA was worst than the default approach. Further, techniques required few iterations to obtain suitable HP settings and converge (around $300-450$). By means of the average ranking curve (Figure 24b) an user may choose different techniques



(a) Average loss curve.



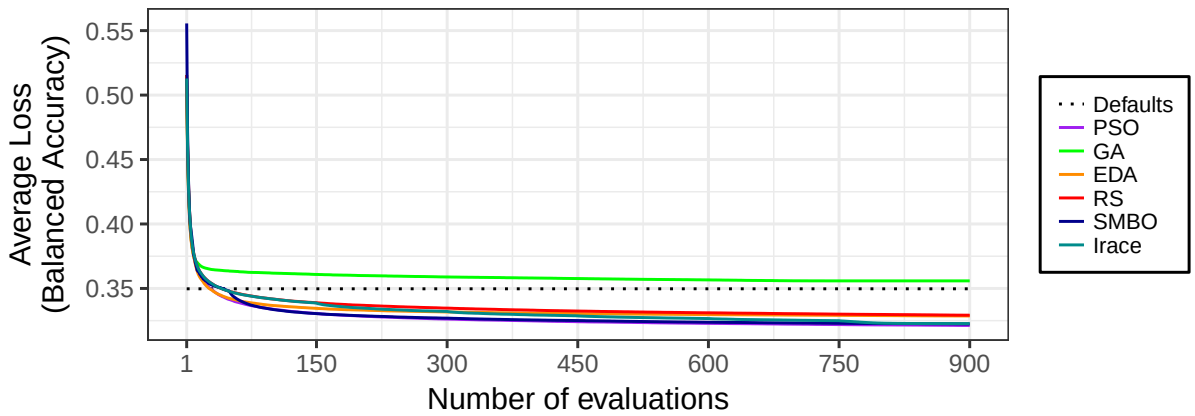
(b) Average ranking curve.

Figure 25 – Loss curves for the CART algorithm across datasets.

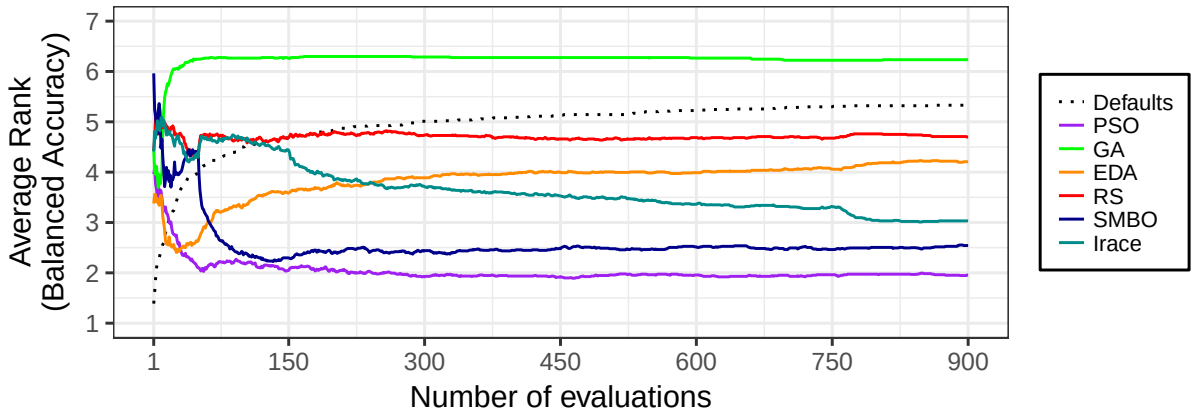
according to the available budget size. For example, if tuning were performed with at most $b = 200$ evaluations, PSO and SMBO would be the best choices. From that point and ahead, Irace surpasses all the techniques.

Figure 25 presents loss curves for the CART algorithm. Except for Irace, all the techniques converged at $b = 150$ evaluations. However, despite requiring more evaluations, Irace was the technique which most reduced the loss of the datasets. Contrarily than the previous scenario, all the tuning techniques presented average loss values lower than the value obtained by the default HP values (Figure 25a). When analyzing the average rankings, different techniques are most suitable according to the budget size. If a tiny budget is provided ($b \leq 50$), PSO is slightly better than the other techniques. From $50 < b \leq 150$, the SMBO would be the best choice. On the other hand, with budgets ($b > 150$) Irace is the technique that best recommends values for CART HPs.

CTree loss curves (Figure 26) describe a third tuning profile. Different than the two algorithms described above, CTree is the less sensitive algorithm to tuning. The average loss reduction with tuning is very small when compared with the results obtained with default



(a) Average loss curve.



(b) Average ranking curve.

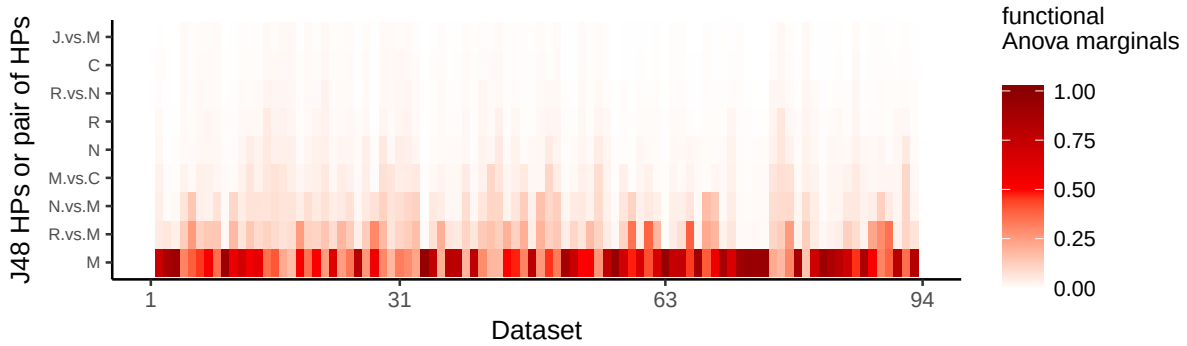
Figure 26 – Loss curves for the CTree algorithm across datasets.

HP values (Figure 26). GA was the only technique to perform worst than default HP settings. Nevertheless, the absolute differences between techniques are very small, with PSO and SMBO being slightly better than the others. Based on the average ranking values, the PSO technique would be the best one for any budget size (Figure 26b), considering the budget evaluate in this thesis.

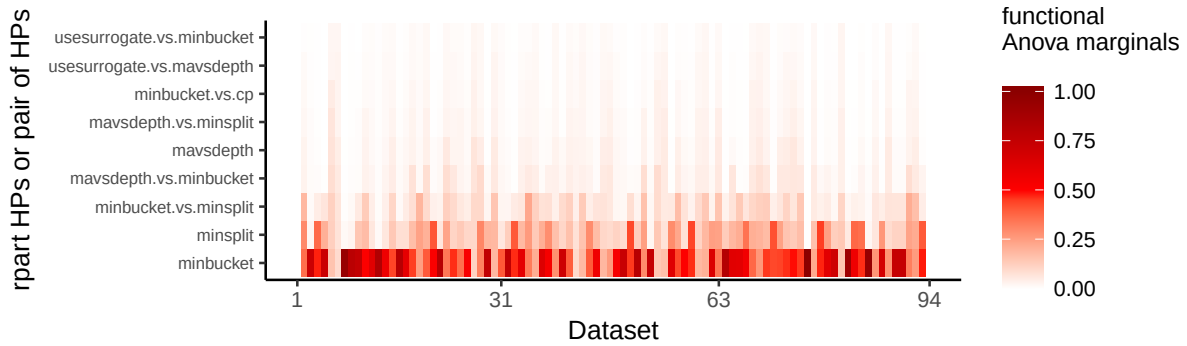
6.6 Relative importance of the HPs

Statistical analysis was also used to understand how different HPs affect each other and the DTs performances. A recent approach to evaluate HPs relative importance is the Functional ANOVA (fANOVA) framework[7], introduced in Hutter, Hoos and Leyton-Brown (2014). In that study, the authors present an algorithm for computing marginal predictions and quantify the importance of single HPs and interactions between them. The fundamental idea is to generate regression forests that predict the performance of HP settings applying a variance decomposition directly to these trees.

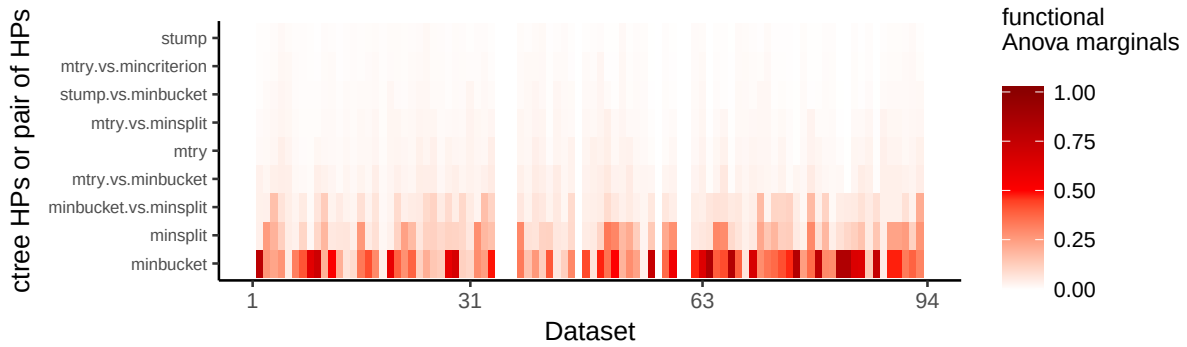
[7] <https://github.com/automl/fanova>



(a) Functional ANOVA values for J48 hyperparameters.



(b) Functional ANOVA values for CART hyperparameters.



(c) Functional ANOVA values for CTree hyperparameters.

Figure 27 – Functional ANOVA HPs marginal predictions for DTs. Marginal predictions are scaled between zero and one. Figure adapted from Mantovani *et al.* (2018a).

In the source article, the authors ran fANOVA with HP settings obtained from SMBO executions over some scenarios but never considering more than 13.000 settings. Here, a single execution of any tuning technique generates $30 \times 10 \times 900 = 270.000$ evaluations. Thus, experiments using all techniques would have a high computational cost. Since Irace was the top-ranked technique for all the DT algorithms (Figure 23), it was used to provide the HP settings for this analysis. In the experiments, 27.000 HP settings for 3 repetitions were used when evaluating the relative importance of DT HPs. When evaluating CTree models with fANOVA, seven jobs produced errors. In these situations, a white column is presented at the heatmap.

Figure 27 shows the results obtained from the fANOVA analysis. In the figure, datasets are arbitrarily listed at the x-axis while the y-axis presents HPs' relative importance regarding

fANOVA. The darker the square, more important is the HP (or a pair of them) for inducing trees in the datasets (scaled between zero and one). Single HPs, or combinations of HPs, whose contribution to the predictive performance of the induced models was lower than 0.005 were filtered from the figure. Even removing most of these unnecessary HPs, most of the heatmap's rows are quite white. It is substantial evidence that most of the HP combinations have little contribution to the predictive performance of the induced DTs.

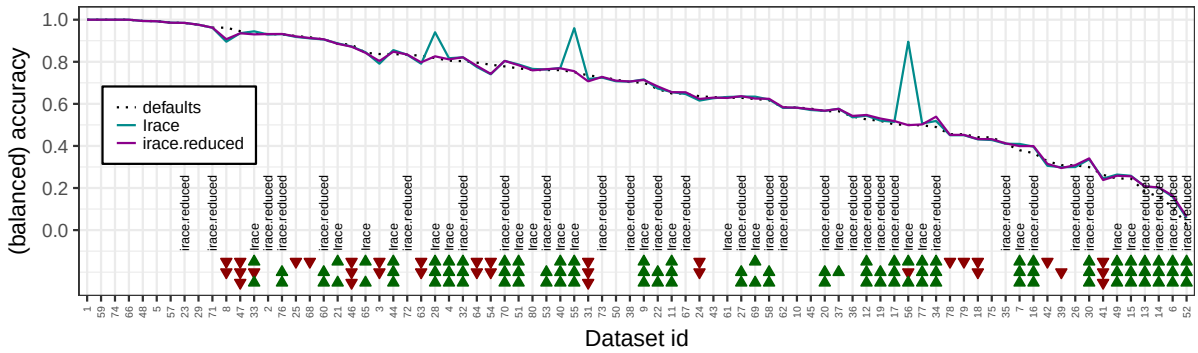
In Figure 27a, fANOVA indicates that most of the J48 performances were influenced by '`M`' values: when not alone, in combination with another HP (`R`, `N`, `C`). For CART, the '`minbucket`' and '`minsplit`' HPs are the main responsible for the performance of the induced DTs, as may be seen in Figure 27b. CTree HPs behave like CART ones: '`minbucket`' and '`minsplit`' are also the most important HPs (Figure 27c). On the other hand, they have less strength to predict marginal distributions. It reinforces previous findings describing that CTree has the less sensitive *HP profile*, because even the most important HPs have a small contribution to the final models.

6.7 Tuning with reduced HP spaces

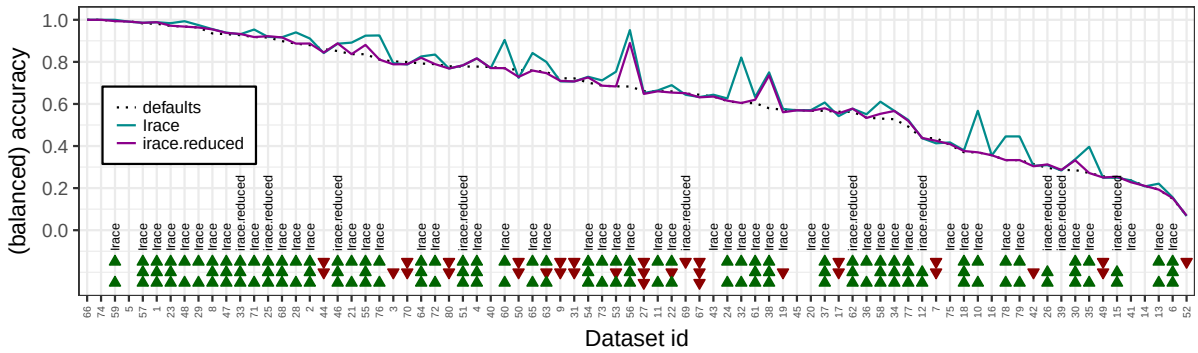
The fANOVA results indicate that although tuning was performed with several HPs, just a subset of them seems to influence the final performance of the induced DTs. Thus, it opens an interesting research question: does the tuning of DT algorithms with a reduced HP space affect the predictive performance of the induced models? So, additional experiments were performed.

Except for the techniques selected, the tuning of DTs with their reduced HP spaces followed the same experimental methodology described in Table 10. Based on the loss curves and average ranking values, the best tuning techniques per algorithm were selected for the optimization: Irace for J48 and CART, and PSO for CTree. For each algorithm, only the most important HPs were selected according to the fANOVA analysis. For CTree and CART, $H = \{\texttt{minbucket}, \texttt{minsplit}\}$ HPs were selected, while for the J48 algorithm the subset $H = \{\texttt{M}, \texttt{R}, \texttt{C}, \texttt{N}\}$ was chosen. The remaining HPs used their default values provided by their correspondent implementations. In 14 of the 94 datasets, machine errors occurred when performing the tuning task. Hence, their results were removed from the analysis and experiments were performed with 80 datasets.

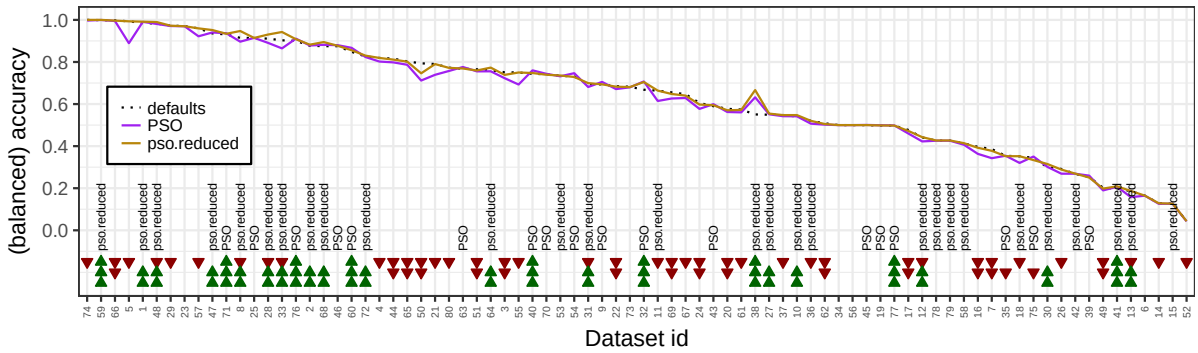
Figure 28 summarizes tuning results with reduced HP spaces for all algorithms. The top figure shows the HP tuning results of the J48 algorithm using Irace with the complete and reduced HP space (`irace.reduced`). Results are also compared with the BAC values obtained with default HP settings. The Wilcoxon paired-test with $\alpha = 0.05$ was also applied to assess the statistical significance of the results. Whenever HP tuning provided statistically significant improvements, an upper green triangle appears. On the other hand, every time a red down triangle is presented, default HP settings induced models statistically better than tuned trees. The first line of triangles at each figure shows statistical comparisons between tuning with the complete



(a) HP tuning results for J48.



(b) HP tuning results for CART.



(c) HP tuning results for CTree.

Figure 28 – HP tuning results for DTs with reduced hyperspace..

Table 11 – Statistical improvements with complete and reduced HP spaces.

Algorithm	Complete Space (▲- -▼)	Reduced Space (▲- -▼)	Both Spaces (▲- -▼)
J48	27-**37**-**16**	32-**34**-**14**	37-**39**-**4**
CART	**49**-18-13	29-**34**-**17**	**52**-26-2
CTree	9-**32**-**39**	25-**38**-**17**	25-**55**-**0**

HP space and default HP settings. The second line compare tuning with reduced HP space and default HP settings. In the end, the last lines compares the best of both spaces (complete or reduced) against the default HP values. On the remaining cases, approaches performed similarly.

Experimental results and statistical tests show different results for the different algorithms. Although BAC curves seem similar, they indicate interesting behaviors. Tuning J48 and CTree with the reduced HP space increased the number of cases where tuning is significantly better. However, the default HP values would still be preferred for most of the problems. Moreover, when CART HPs are tuned, the opposite occurs: using reduced HP space makes tuning less effective than before.

Table 11 summarizes these statistics for each algorithm. There, it is possible to see that removing unnecessary HPs benefits J48 and CTree tuning. The removed HPs could have added noise during the optimization with the complete space, which is not considered now. On the other hand, reducing the CART space worsened the tuning in general. It indicates that the removed HPs are still important for the tuning of several datasets. Even though, in the three cases, when the best choice of HP space is considered, all the algorithms had their number of statistically significant improvements increased.

6.8 Chapter Remarks

This chapter investigated the *HP profile* of three DT induction algorithms. There are some studies in the literature that perform HP tuning for DTs (BERMÚDEZ-CHACÓN; GONNET; SMITH, 2015; WAINBERG; ALIPANAHI; FREY, 2016; SARDá-ESPINOSA; SUBBIAH; BARTZ-BEIELSTEIN, 2017), but none of them used an unbiased experimental methodology or complete HP space. Thus, to the best of our knowledge, this is the first investigative study regarding the *HP profile* of DT algorithms.

As expected, different algorithms presented different *HP profiles*. Regarding the HP tuning predictive performances, CART and CTree are complementary opposites. The first was extremely sensitive while the second was unaffected by the HP tuning for most of the problems. J48 stays in the middle: tuning obtained statistical improvements for almost half of the datasets. This might be because default values adopted by `RWeka` were chosen to obtain the best overall performance on UCI ML repository (BACHE; LICHMAN, 2013) datasets.

Depending on the dataset, the predictive performance obtained by tuning techniques can be very small in relation to those obtained by default HP settings. Hence, the results indicate that, for some optimization problems, it is better to just use the default HP settings. When comparing tuning techniques, different techniques are more suitable for different budget sizes. If the user has a large enough budget, Irace is a good choice. On the other hand, PSO and SMBO are the recommended techniques with a faster convergence.

The fANOVA analysis also indicated that few of the HPs are effectively responsible for the predictive performance of the final trees. Similar results with different ML algorithms were reported in Rijn and Hutter (2017). In this sense, the fANOVA framework is a powerful tool to reduce the search HP space and time spent with optimization. The last experiments described in this chapter showed that a higher number of statistically significant improvements were obtained when a reduced HP space was used to tune J48 and CTree algorithms. All the data generated from tuning tasks is meta-knowledge that will be used by the MtL recommender system proposed in the next Chapter.

CHAPTER

7

TO TUNE OR NOT TO TUNE?

The SVM and DTs' *HP profiles* show that tuning is not always necessary. In many situations, default HP settings are already enough to induce accurate models. Therefore, when computational resources are limited, knowing beforehand which approach should be used (tuning or defaults) can reduce the computation time and efforts.

In this chapter, *HP profiles* are explored to propose a new MtL system to support the identification of situations where HP tuning can improve classification algorithms' performance. As such, this system takes advantages of previous classification experiments tuning SVMs and DTs. Given a new unseen dataset, the system recommends the most suitable approach: to tune or not to tune HPs. The experiments described in this chapter have been published in Mantovani *et al.* (2015c) and documented in Mantovani *et al.* (2018b).

The next sections are organized as follows: Section 7.1 presents the general MtL framework and its sub-tasks; Section 7.2 presents results for SVMs while Section 7.3 analyzes results for DTs; Section 7.4 projects system predictions in the base level; and Section 7.5 presents the final remarks of the chapter.

7.1 Recommender system framework

The general overview of the proposed MtL system is illustrated in Figure 29. The recommendation task contains two learning steps: the *base level*, where the HP tuning process is performed for different datasets; and the *meta level*, where meta-learning is applied to output the recommendation for a new dataset: to tune or not to tune." Further subsections will describe each component of the framework.

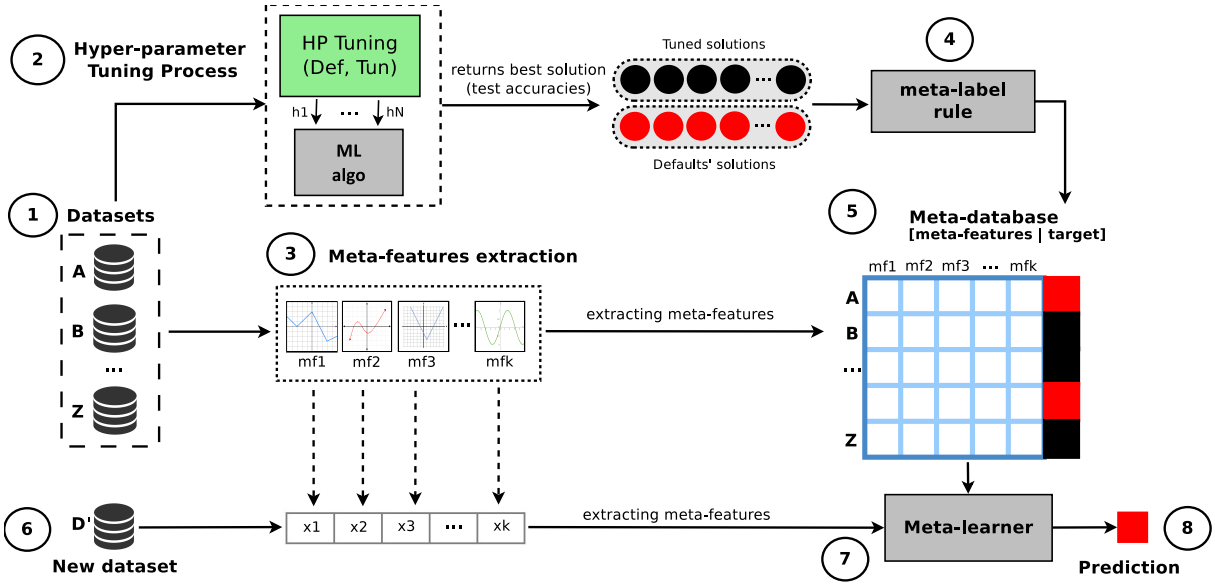


Figure 29 – MtL recommender system to predict whether HP tuning is required or not. Adapted from (MANTOVANI *et al.*, 2018b).

7.1.1 Base-level tuning

The HP tuning process illustrated in Figure 29 (Item 2) was performed over 165 datasets (Item 1) for all the target algorithms (SVM, J48, CART, CTree). However, the tuning setup was modified exploring the *HP profiles* identified in Chapters 5 (SVMs) and 6 (DTs). Thus:

- tuning of SVMs was performed with a budget size of $b = 200$ evaluations and using only the RS technique ;
- tuning of J48 and CART was performed with a budget size of $b = 900$ evaluations and using only the Irace technique; and
- tuning of CTree was performed with a budget size of $b = 900$ evaluations and using only the PSO technique.

The tuning task was performed with the experimental methodology depicted in Figure 6[1]. Techniques were executed 30 times with different data seeds. It is important to mention that tuning jobs were required only for the datasets not included in previous experiments. Datasets whose results were already available did not need to be tuned again. At the end of the HP tuning

[1] For more details, please see Chapter 4.

processes, each dataset has 30 solutions recommended by the tuning techniques (black dots at the figure) and 30 obtained by default HP settings (red dots).

7.1.2 Meta-features

The meta-datasets used in the experiments were generated out of '*meta-features*' describing each base level dataset (Figure 29 - Item 3). These meta-features were obtained by applying measures that extract likely relevant characteristics from the data. A total of 80 different meta-features from the different subsets described in Section 3.1.2 were used in this thesis. They correspond to all the categories of descriptors explored in literature, and are described in detail in Tables 22 and 23 at Appendix B. The exact number of meta-features used from each category is described in Table 12.

Table 12 – Groups of meta-features used in experiments.

Group	N	Description
Simple	17	Simple measures
Statistical	7	Statistical measures
Information-theoretic	8	Information theory measures
Model-based (trees)	17	Features extracted from decision tree models
Landmarking	8	Measures based on the predictive performance of ML algorithms
Data Complexity	14	Measures that analyze the complexity of a problem
Complex Networks	9	Measures based on complex networks
Total	80	

The Pearson correlation coefficient (HALL, 1998) values between each pair of meta-features used to build meta-datasets are shown in Figure 30. High positive correlation values are shown in red, while high negative values are in blue. Meta-features in the both axes are presented in the same order as in Tables 22 and 23 of the Appendix B, according to their correspondent groups. Most of the correlation matrix is composed by blank or light squares.

7.1.3 Meta-targets

Each object in a meta-dataset is labeled with a meta-target, whose value indicates whether the HP tuning significantly improved the predictive performance of the ML model when compared with the use of default HP settings[2]. The so-called "*meta-label rule*" (Item 4 - Figure 29) applies the Wilcoxon paired-test to compare the solutions obtained using tuned and default HP settings.

Thus, given a dataset, if the HP tuned solution was significantly better than the default solution, its corresponding meta-example is labeled as '`Tuning`'; otherwise, as '`Default`'. When

[2] These performance values are assessed by BAC using a nested-CV resampling method.

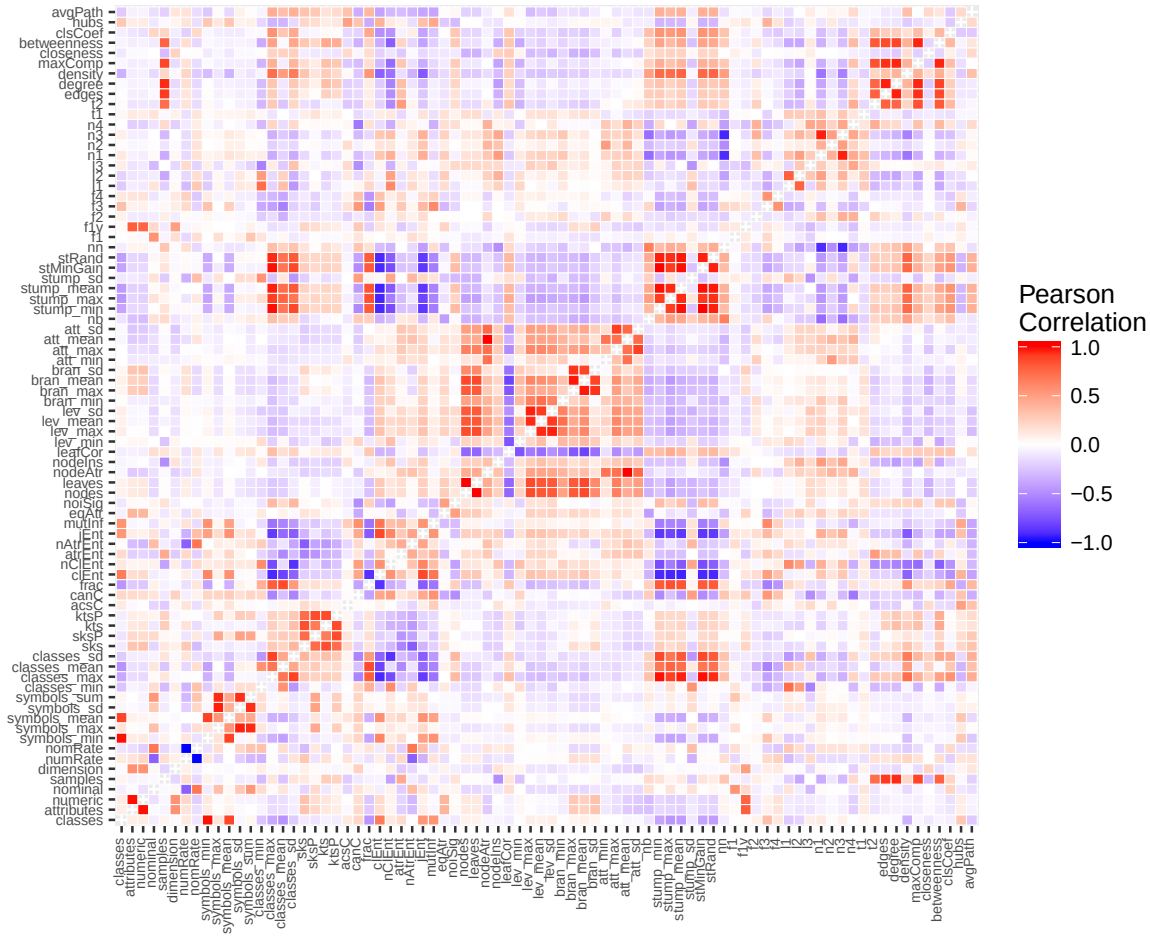


Figure 30 – Pearson correlation coefficient among the meta-features used to build meta-datasets.

performing the Wilcoxon test, three different values of $\alpha = \{0.1, 0.05, 0.01\}$ were considered, resulting in meta-datasets with different class distributions (Item 5 - Figure 29).

For the SVM meta-datasets, initial designs were compared only the `LibSVM` default HP settings with the tuned solutions. The resultant meta-datasets presented a high imbalance rate, prevailing the "`Tuning`" class. However, this became an interesting problem when SVMs "optimized default HP values", presented in Section 5.6, are also considered. These values were obtained optimizing common HP settings simultaneously for a group of datasets. Hence, when labeling SVM's meta-examples, the meta-rule first identifies the best default approach (LibSVM or optimized values) for each dataset and then compares the results with tuned HP settings. Through this approach, the imbalance rate[3] in original SVMs' meta-datasets was reduced from ≈ 2.6 to ≈ 1.7, increasing the number of meta-examples where the use of defaults is recommended.

Table 13 presents the meta-datasets generated from the tuning of the target algorithms. For each resultant meta-dataset, the following information is provided: the α value used to

[3] imbalance rate = (majority class size/minority class size)

generate the labels; the number of meta-examples and meta-features; and the class distribution. It is important to mention that SVM's meta-datasets contain fewer meta-examples because the datasets used to produce optimized default HP settings were removed.

Table 13 – Meta-datasets generated for MtL experiments.

Meta-dataset	α	**Meta examples**	**Meta features**	**Class Distribution** *Tuning*	*Default*
Svm_90	0.1	156	80	102	54
Svm_95	0.05	156	80	98	58
Svm_99	0.01	156	80	94	62
J48_90	0.1	165	80	63	102
J48_95	0.05	165	80	57	108
J48_99	0.01	165	80	52	113
Rpart_90	0.1	165	80	113	52
Rpart_95	0.05	165	80	111	54
Rpart_99	0.01	165	80	104	61
Ctree_90	0.1	165	80	42	123
Ctree_95	0.05	165	80	36	129
Ctree_99	0.01	165	80	26	139

Meta-datasets' labels projected in the PCA space with only the first two components are shown in Figure 31. The labels presented in the figure were defined by the meta-rule with $\alpha = 0.05$. Meta-examples belonging to the "Tuning" class are represented by black triangles, while red squares denote those where default HP settings are suitable. This figure shows different problems for different target algorithms.

CART and SVM problems contain more meta-examples that require tuning. It may be observed by black triangle regions in the border of the space. At the same time, they seem to be *easier* problems, since the overlapping region between classes is smaller than in the other problems. J48 and CTree problems present a higher number of meta-examples where default HP settings are better than tuned solutions. While in the J48 problem most of the tuning meta-examples can be isolated, this is not possible for CTree. In the CTree problem, almost all meta-examples from the minority class (Tuning) stay very close to a meta-example from the majority class (Default). It suggests a very hard decision boundary.

7.1.4 Meta-learning setup

Seven classification algorithms were used as meta-learners (Item 7 - Figure 29): Support Vector Machine (SVM); Classification and Regression Tree (CART); Random Forest (RF); k-Nearest Neighbors (kNN); Naïve-Bayes (NB); Logistic Regression (LR); and Gaussian Processes (GPs). These algorithms were chosen because they follow different learning paradigms with different learning biases. All of them were applied to the meta-datasets using a 10-fold CV resampling strategy and repeated 20 times with different seeds. Their predictions were assessed using the AUC performance measure, a more robust measure than BAC for binary classification

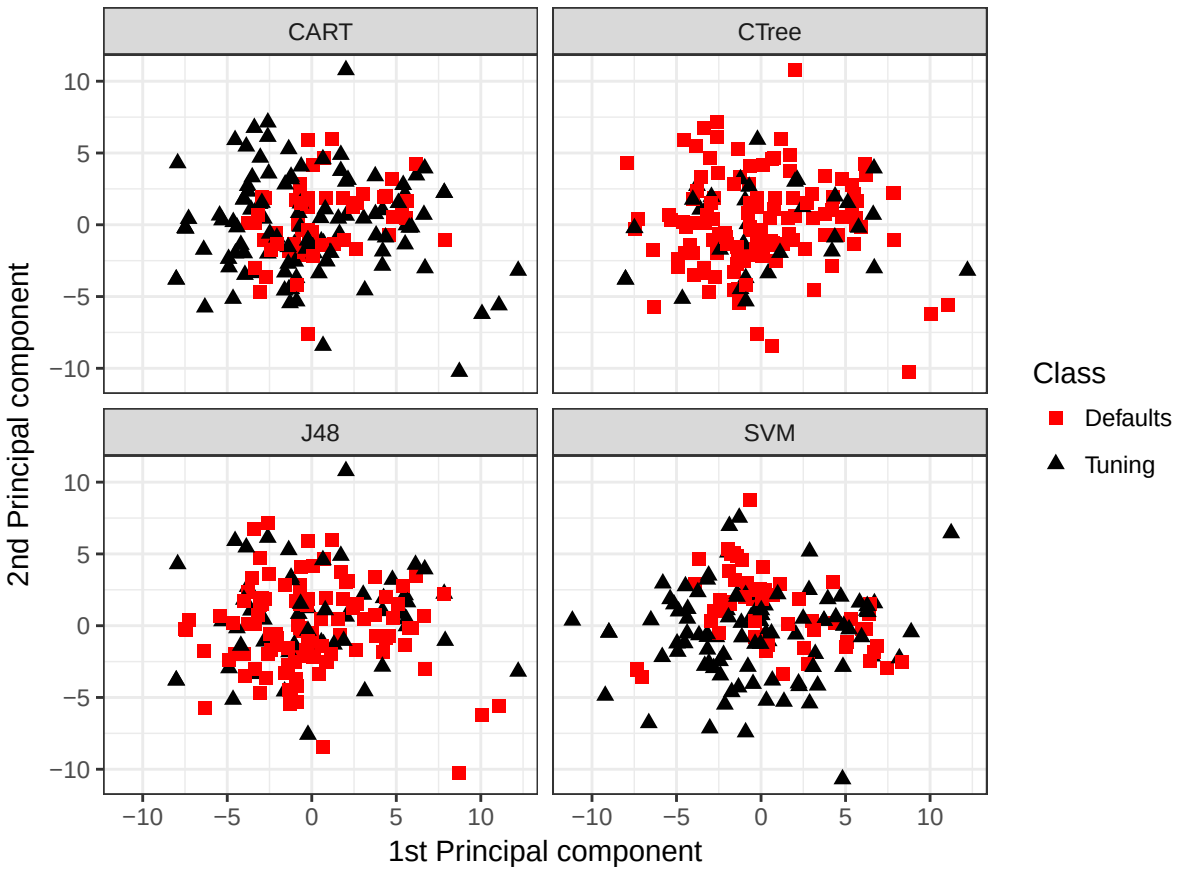


Figure 31 – Labels of the meta-datasets projected in the 2d PCA space. Meta-target labels were defined with an alpha value of $\alpha = 0.05$.

problems[4]. At the meta-level, it was possible to perform:

(i) *Feature Selection*: As each meta-example is described by many meta-features, a subset of them could be enough to induce meta-models with high predictive performance. Thus, a Sequential Forward Selection (SFS) method was added to the MtL experimental setup to perform the meta-feature selection. The SFS method starts from an empty set of meta-features, increasing the set iteratively with the feature that most improves predictive performance. This process stops when a minimum required value of performance improvement ($alpha$=0.01) is not satisfied. Internally, it uses a stratified 3-fold CV to assess the predictive performance (AUC) of the induced models.

(ii) *Tuning*: Since the HP values of the meta-learners may also affect their performance, the tuning of the meta-learners was also considered in the experimental setup. A simple RS technique was performed with a budget of 300 evaluations. Resultant models were assessed through an inner stratified 3-fold CV and the AUC measure. Table 24 in Appendix C shows the HP space considered for the tuning of meta-learners.

[4] The BAC measure was preferred at the base level because data collection contains binary and multiclass classification problems.

(iii) *Data balancing*: Even using the optimized default HP values for SVMs, the classes presented in the meta-datasets still have a disproportional rate. Consequently, the application of the Synthetic Minority Over-sampling Technique (SMOTE) technique (CHAWLA *et al.*, 2002), which can deal with data imbalance, was also investigated.

Some of the algorithms' implementations selected as meta-learners use a data scaling process by default. This is the case of SVMs, kNN, and GPs. These algorithms try to reduce distances between examples or spaces. In this way, scaling all the attributes reduces the *dominance* an attribute may have over others. Conversely, algorithms like DTs and RF follow a different learning bias, requiring little data preparation (JAMES *et al.*, 2014). In fact, a preliminary experiment showed that algorithms presented their best predictive performances using their default setups. Due to that, data scaling was not considered as an option, and algorithms used their default procedures.

Two versions of the meta-datasets were also adopted for the experimental comparisons: a meta-dataset only composed by simple meta-features and another just with data complexity meta-features. These versions were initially investigated in the first experiments reported in Mantovani *et al.* (2015c). Experimental results were also compared with two traditional baselines:

- `ZeroR`: a meta-model that always recommends the majority class label;
- (`Random`): a meta-model that provides random recommendations.

7.2 When to tune SVMs?

Figure 32 summarizes the predictive performance of different meta-learners for three different sets of meta-features, namely: *all*, *complex* and *simple*. The former has all 80 available meta-features, the *complex* set contains only 14 data complexity measures as meta-features and the latter consists of 17 simple and general meta-features. In this figure, the x-axis shows the meta-learners while the y-axis shows their predictive performance assessed by the AUC averaged over 20 repetitions. Besides, it shows the impact of different *alpha* (α) levels for the Wilcoxon test for the definition of the meta-target labels.

The Wilcoxon paired-test with $\alpha = 0.05$ was also applied to assess the statistical significance of the predictive performance differences obtained by the meta-models with *all* meta-features, when compared to the best baseline. An upper green triangle (▲) at the x-axis identifies situations where using all the meta-features were statistically better. On the other hand, red down triangles (▼) show results where one of the baselines was significantly better. In the remaining cases, the predictive performance of the meta-models were equivalents.

The best results were obtained by the RF meta-learner using data complexity meta-features (*complex*), achieving AUC values nearly 0.80 for all α levels. These meta-models were

Table 14 – Meta-learning experimental setup.

Element	Method	R package

Meta-learner	Support Vector Machine (SVM)
	Classification and Regression Tree (CART)
	Random Forest (RF)
	k-Nearest Neighbors (kNN)
	Naïve-Bayes (NB)
	Logistic Regression (LR)
	Gaussian Processes (GPs)
		`e1071`
		`rpart`
		`randomForest`
		`kknn`
		`e1071`
		`gbm`
		`kernlab`

Resampling	10-fold CV	`mlr`
Feature Selection	Sequential Forward Selection (SFS) - $alpha = 0.01$ inner 3-CV - measure AUC	`mlr`
Tuning	Random Search (RS) budget = 300 inner 3-CV - measure AUC	`mlr`
Data Balancing	SMOTE oversampling rate = $\frac{nMaj(meta-dataset)}{nMin(meta-dataset)}$	`mlr`
Repetitions	20 times with different seeds seeds = $\{1,\ldots,20\}$	- -
Evaluation measures	AUC predictions (prob)	`mlr`
Baselines	Simple meta-features Data complexity meta-features Random meta-model ZeroR meta-model	- - `mlr` `mlr`

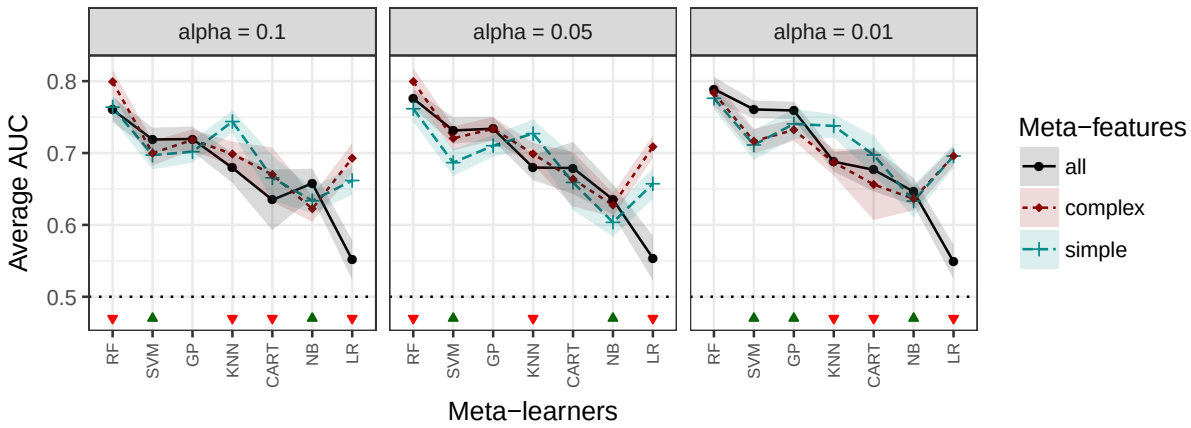


Figure 32 – Meta-learners average AUC results on SMV's meta-datasets. The black dotted line at $AUC = 0.5$ represents the predictive performance of ZeroR and Random meta-models. Figure from Mantovani *et al.* (2018b).

also statistically better than those obtained by other approaches at $\alpha = \{0.90, 0.95\}$. When $\alpha = 0.99$, the RF meta-learner using all the meta-features also generated a model with AUC near 0.8.

When a lower value of α is considered by the meta-label rule, performances using data complexity and all the available meta-features tend to show similar distributions. The meta-learners obtained their best AUC values following the highest assumption ($\alpha = 0.99$).

Overall, varying α values did not change the predictive performance of the evaluated algorithms substantially. In fact, few meta-examples have their meta-targets modified by the meta-rule with different values of alpha. Hence, the predictions in the different scenarios mostly are the same and performances remain similar.

Regarding predictive performance, the algorithms = {RF, SVM, GP, kNN} induced accurate meta-models for the three meta-dataset variations. The predictive performance obtained varied between AUC = $\{0.70, 0.80\}$. Even the LR, depending on the meta-features used to represent the recommendation problem, achieved reasonable AUC values. For comparison purposes, it is important to mention that both random and ZeroR[5] obtained AUC of 0.5 in all these meta-datasets[6]. Since best results for most of the meta-learners were obtained using all meta-features, and the possibility to select different sub-sets from this dataset, the next analysis was performed using all meta-features.

7.2.1 Evaluating different setups

Given the great difference among meta-learners' results, three different setups were also evaluated, aiming to improve their predictive performances and enabling a deeper analysis:

(i) `featsel` - meta-feature selection via SFS method (BISCHL *et al.*, 2016);

(ii) `tuned` - HP tuning of the meta-learners via a simple RS technique; and

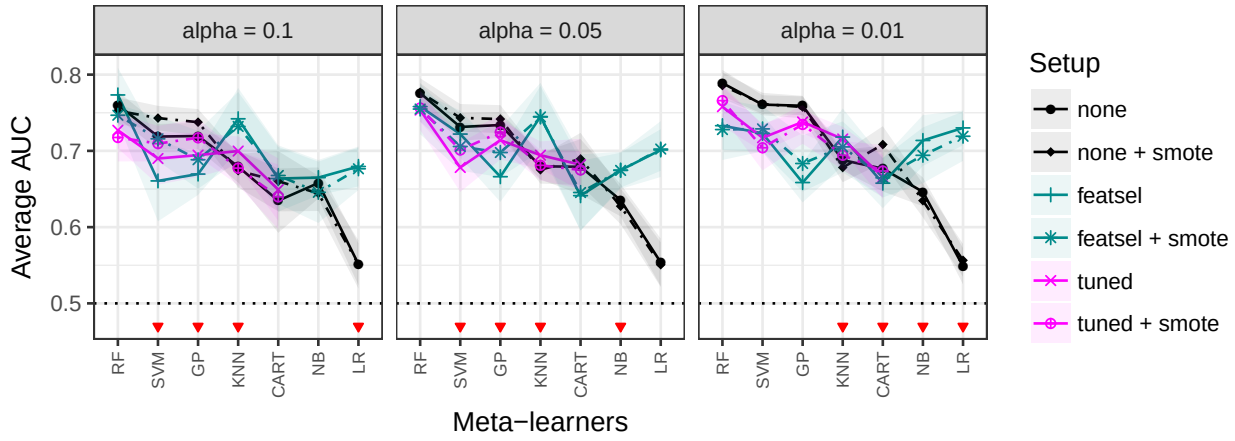
(iii) `smote`: data balancing with SMOTE (CHAWLA *et al.*, 2002).

They were compared with the original meta-data with no additional process, namely `none`, which is the baseline. Setups were not performed together to avoid overfitting, since meta-datasets have not so many meta-examples. Depending on their combinations, three levels of CV would be used to assess models. For example: if feature selection and HP tuning were enabled at the meta-level, one CV would be used for feature selection, one for tuning and another one to assess resultant models.

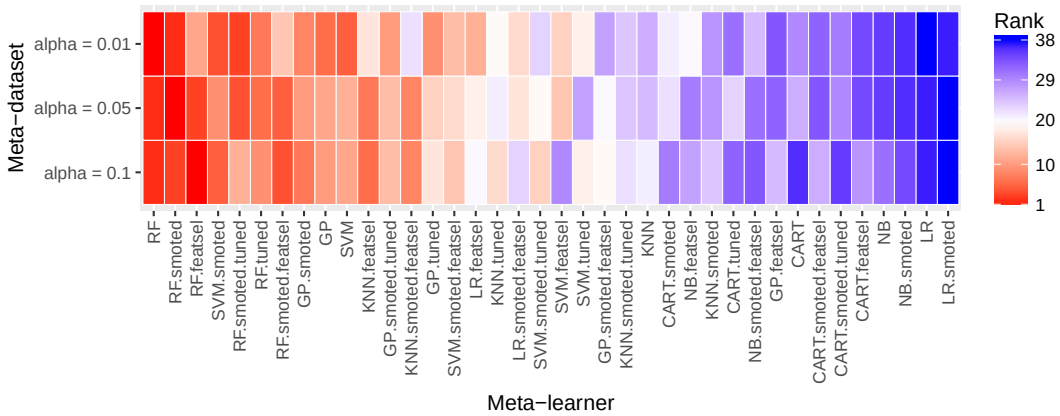
Figure 33 summarizes the main aspects of these experimental results. Figure 33a shows the average AUC values for each experimental setup considering all the meta-learners and the α levels. The NB and LR meta-learners do not have any tunable HP. Thus, their results in this figure are missing for the `tuned` setups (with and without SMOTE).

As in the case of previous experiments, statistical analysis is presented. Every time an upper green triangle is placed at the x-axis, raw meta-data (*none*) generated results statistically better than using the best of the experimental setups evaluated. On the other hand, red triangles indicate when tuning, feature selection or SMOTE could statistically improve meta-models. In

[5] This classifier simply predict the majority class.
[6] The AUC performance values were assessed using the implementations provided by `mlr` R package.



(a) Average AUC performance values.



(b) Average AUC ranking values.

Figure 33 – AUC performance values obtained by all meta-learners considering different experimental setups. The black dotted line at $AUC = 0.5$ represents the predictive performance of ZeroR and Random meta-models. Figure from Mantovani *et al.* (2018b).

the remaining cases, meta-models were equivalents, and, in theory, no additional process would be required.

Despite the different setups evaluated, RF is still the best meta-learner for all α scenarios. It is followed by SVM and GP versions using SMOTE, and depending on the experimental setup, kNN and LR also yield good predictive performances. Regarding the HP tuning (`tuned`) of the meta-learners, kNN was the single one with performances slightly improved for all the alpha values.

Using just smote improved results for SVM, GP and CART meta-learners. In general, it produces small improvements, but most of them are statistically significant. When used with tuning or feature selection, it generates different patterns: for SVM and GP it improves setups' performances; for LR, NB and kNN it does not bring any benefit; and for the other algorithms its use is indifferent. The SMOTE's inefficiency may be due to the use of default HP values, which was already maximized using the optimized values when creating the meta-datasets.

The use of feature selection (`featsel`) deteriorates the performance of {SVM, RF, GP, CART} meta-learners. On the other hand, it quite improved {kNN, LR, NB} performances for most cases. kNN benefits from a subset of meta-features to maximize the importance of more relevant meta-features. For NB and LR, selecting a subset of the attributes may reduce data noise and less-informative attribute. Furthermore, it is important to observe that the meta-models with feature selection presented the highest standard deviation between the setups (light area along the curve). It may be related to the different subsets selected every time feature selection is performed for the 30 repetitions.

Additionally, Figure 33b presents a ranking of all the combinations of meta-learners and experimental setups. In the x-axis, they are presented in ascending order according to their average ranking for the three scenarios (α values), showed on the y-axis. More red the squares, lower the ranking, i.e., better the results. As reported previously, RF with no additional option is the best-ranked method. It is also clear the some of its versions are among the best meta-models, strengthening its choice as a meta-learner. SVM, GP and kNN setups appear in the next positions.

7.2.2 Meta-features importance

From the induced RF meta-models, it is possible to estimate the relative importance of the meta-features based on the Gini impurity index. This metric is internally used for the calculation of node splits (BREIMAN, 2001). Figure 34 depicts the average relative importance of the meta-features obtained from the RF meta-models. The importance is shown for the experiments considering all meta-features and $\alpha = 0.05$ (middle case). In the x-axis, meta-features are presented in decreasing order according to their average relative importance values.

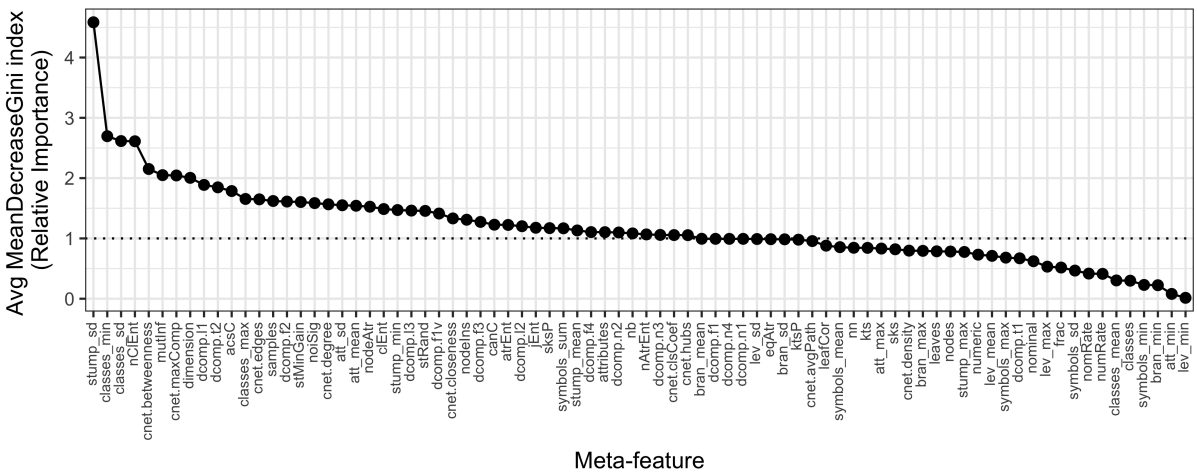


Figure 34 – Average meta-features relative importance' obtained from RF meta-models. Figure from Mantovani *et al.* (2018b).

In the face of no negative importance value obtained, no meta-feature disrupted was selected to build meta-models. It also shows that a lot of meta-features were relevant for the induction of the meta-models, what explains somewhat why feature selection at the meta-level

produced worse results for most of the meta-learners. The most important meta-feature was a landmarking measure: "`stump_sd`", which describes the standard deviation of the number of examples correctly classified by a decision stump. It measures at a low level the complexity of the problem considering its simplicity. The second most important was a simple meta-feature: "`classes_min`", a meta-feature which measures the minority class size. The third one was also a simple meta-feature: "*classes_sd*", which describes the standard deviation of the number of examples in the classes.

These meta-features together strongly indicate that RF has its recommendations based on the data imbalance: it suggests that high imbalance datasets might not provide better results with tuning, then defaults are a good estimate for SVM's HPs. The next seven most important meta-features were:

- "*nClEnt*" and "*mutInf*": are information-theoretic meta-features. The first measure describes the class entropy for a normalized base level dataset, while the second measures the mutual information, a reduction of uncertainty about one random feature given the knowledge of another;
- "*betweenness*": betweenness centrality is a meta-feature derived from complex networks that measures for vertex and edges the average number of shortest paths that traverse them. The value will be small for simple datasets, and high for complex datasets;
- "*l1*" and "*t2*": are data complexity meta-features. The first measures the minimum of an error function for a linear classifier, while the second measures the average number of points per dimension. These features try to map the class separability (l1), and the geometry of the problem's dimension (t2);
- "`dimension`": is a simple meta-feature which measures the relation between the number of samples and attributes in the dataset;
- "`maxComp`": it is also a complex-network meta-feature which measures the maximum number of connected components in the graph. If a dataset presents a high overlapping of its classes, the graph will present a large number of disconnected components, since connections between different classes are pruned.

Among the most important, there are meta-features from different categories (simple, data complexity, complex-networks and from information-theory). Complex-network measures describe data complexity regarding graphs and indicate how sparse are the classes between their levels. Data complexity meta-features try to extract information, directly and indirectly, related to the class separability. The stump meta-feature comes in the same direction, trying to identify the complexity of the problem by a simple landmarker. The information-theoretic meta-features check indirectly how powerful are the datasets attributes to solve the classification problem.

Although summarized rules cannot be obtained from RF meta-models, the meta-features importance analysis suggests some general aspects considered by the algorithm to provide accurate recommendations. For instance, datasets qualities like data balancing, class sizes, complexity, and linearity are important characteristics to recommend when a HP tuning is required for SVMs.

7.2.3 Linearity Hypothesis

Previous section analysis suggests that linearity is a key aspect to decide between the recommendation of default or tuned HPs for SVMs. Results indicate that default values might be good for classification problems close to being linearly separable. As a consequence, tuning would be required for "harder" problems, where SVMs would need to find irregular decision boundaries.

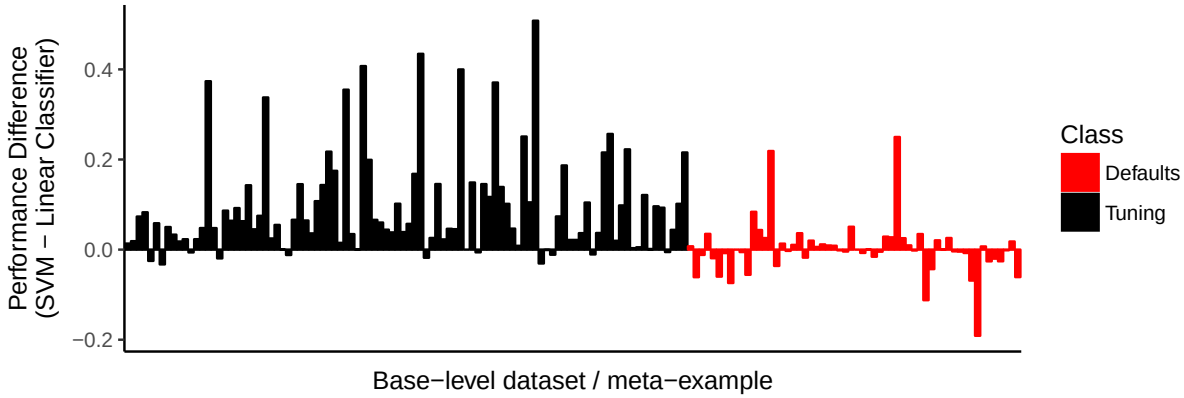


Figure 35 – Performance differences between SVM and a linear classifier in all the base-level datasets. Figure from Mantovani *et al.* (2018b).

To investigate this hypothesis, a linear classifier was also evaluated on all the 156 base-level datasets where SVMs were evaluated. If the linearity hypothesis is true, the performance difference between SVMs and the linear classifier in meta-examples labeled as "`Defaults`" would be smaller than in meta-examples labeled as "`Tuning`". Figure 35 depicts these performance differences yielded for all base-level datasets. Datasets at the x-axis are split based on their meta-target labels, namely "`Tuning`", most left in black, and "`Defaults`", most right in red. Although there are some outliers, the performance differences for "`Tuning`" meta-examples are in general much higher than for the "`Defaults`" ones. In fact, the observed patterns go towards the linearity hypothesis.

In Leite, Brazdil and Vanschoren (2012), the authors proposed a set of "relative landmarks" meta-features based on the pairwise performance difference of simple landmark algorithms. Based on this idea, a set of 10 new relative landmark meta-features was generated from the pair-wise performance comparison of five algorithms: kNN, LR, SVM, NB, and DS. All of these new meta-features are described in Table 23 at Appendix B.

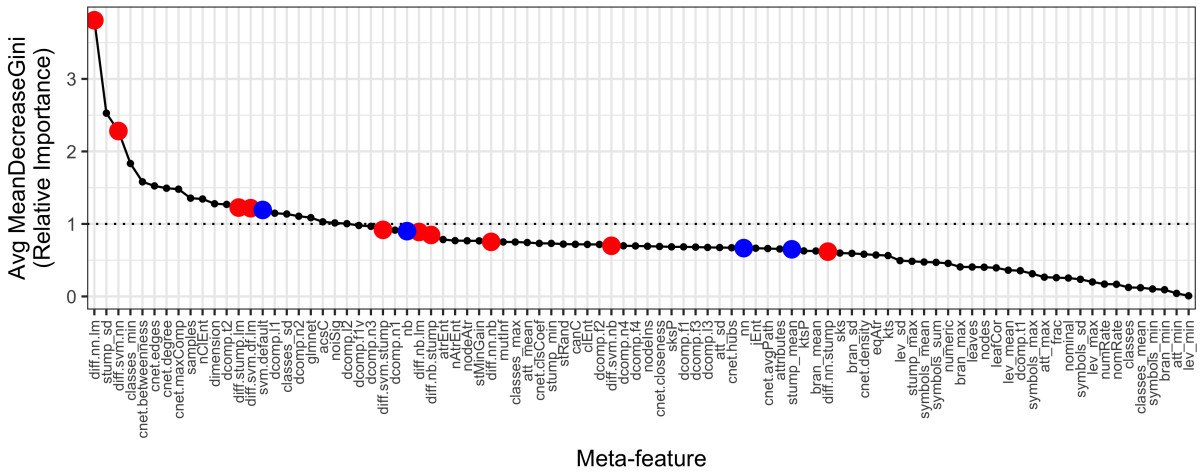


Figure 36 – Average relative importance of the meta-features obtained from RF meta-models when including new meta-features to the meta-dataset. Figure from Mantovani *et al.* (2018b).

RF meta-models were performed again with the expanded meta-datasets to analyze how useful are the new meta-features. Figure 36 depicts the relative importance values of the meta-features averaged over 30 executions with the expanded meta-dataset. The new meta-features are highlighted in red, while the simple landmarks performances are shown in blue (for comparison purposes).

Two of the relative landmark meta-features are placed in the top-10 most important meta-features:

- 1st - `diff.nn.lm`: the performance difference between 1-NN and LR; and
- 3rd - `diff.svm.nn`: the performance difference between SVM and 1-NN.

Other two measures are at top-20: the performance difference between DS and LR (`diff.stump.lr`), and the performance difference between SVM and LR (`diff.svm.df.lr`). It is interesting that three of them are directly dependent of the LR performance. The simple landmark meta-features performed in general worse than relative landmarks. All these relative importance plots bring evidence that the linearity hypothesis is true, and at least one characteristic that defines the need of tuning for SVMs is the linearity of the base-level classification problem.

Given the potential presented by the relative landmark meta-features, they were experimentally evaluated in combination with the most important sets of meta-features previously evaluated. Complex network (`cnet`) meta-features were included since they are ranked between the most important descriptors (as described in Subsection 7.2.2). Simple and data complexity (`complex`) meta-features are the first approaches adopted in the experiments.

Figure 37 presents a final comparison between the main experimental setups considering the addition of the relative landmarking (`relativeLand`) meta-features. The left chart shows AUC performance values obtained each of the original setups, while right picture presents setups

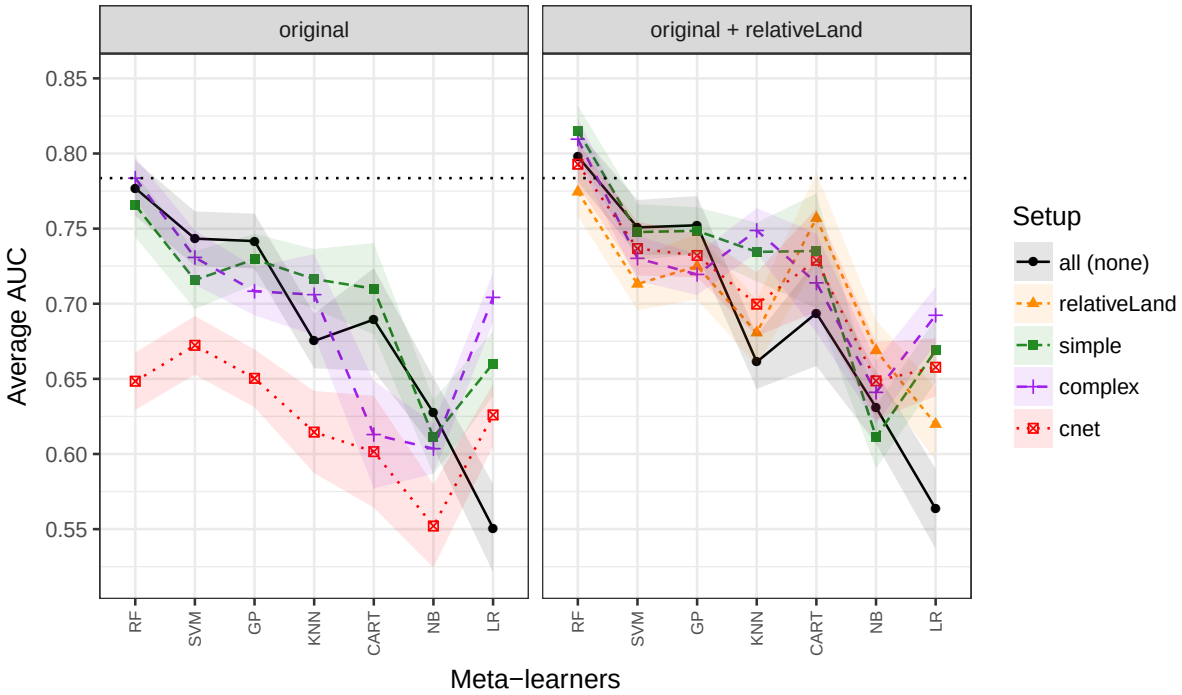


Figure 37 – Evaluating experimental setups adding relative landmarking meta-features. Results are averaged over 30 different executions. Figure from Mantovani *et al.* (2018b).

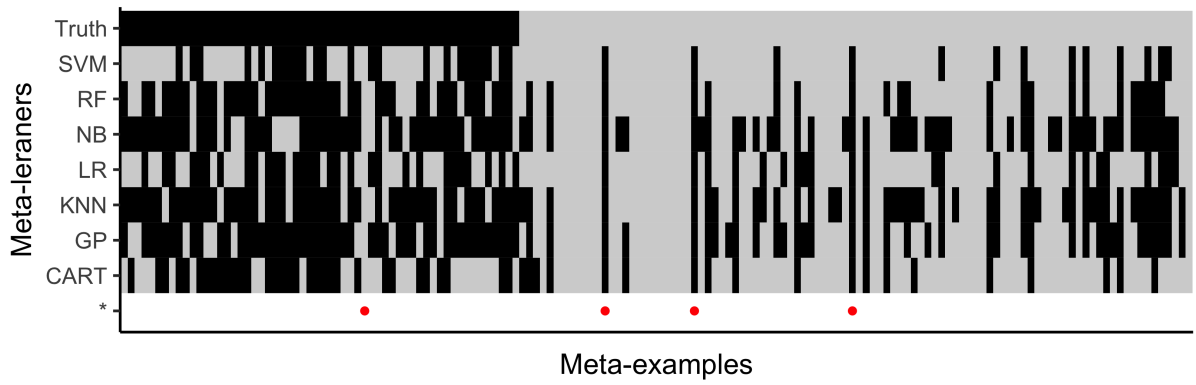
performances including the relative landmarking meta-features. The black dashed line highlights the maximum AUC value obtained so far in experiments without considering the new set of meta-features.

The figure shows that relative landmarking meta-features improve all the setups tried with them. At least three different setups used by RF could surpass the high AUC performance value obtained in the experiments. The setup considering simple and relative landmarking meta-features induced the best meta-models for {RF, SVM, GP} algorithms. kNN and LR reached maximum results using a combination of data complexity and relative landmarking meta-features, while for CART and NB just the "relativeLand" set was the best choice.

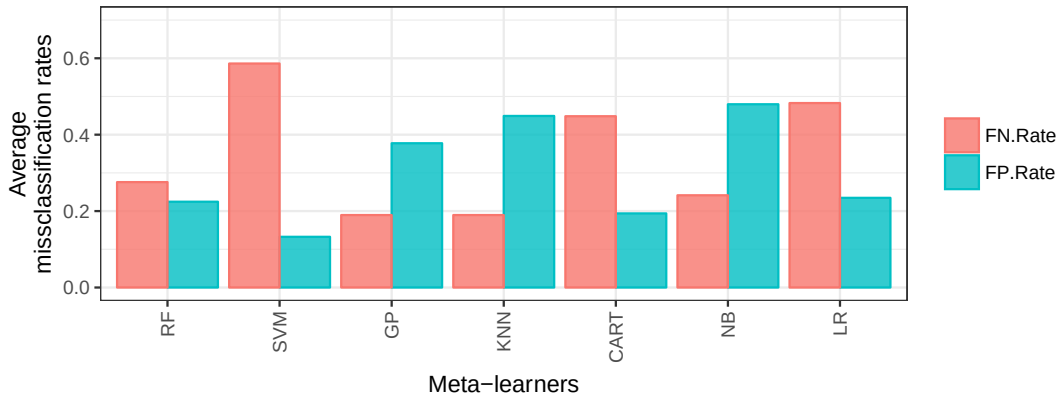
7.2.4 Checking predictions

Analyzing the predictions performed by the meta-learners at a finer granularity can be useful to better understand their behavior. In this sense, Figure 38 depicts the misclassifications of the meta-learners considering their best experimental setups. The top chart (Fig. 38a) shows all the individual predictions, with the x-axis listing all the meta-examples and y-axis the meta-models. "`Defaults`" labels are indicated in black, while "`Tuning`" ones are in gray. The top line at y-axis, "`Truth`", shows the truth labels of the meta-examples, which are ordered according to their truth labels. The bottom line ("*") shows red points for meta-examples misclassified by all meta-learners.

In SVM tuning recommendation problem, "`Defaults`" is defined as the positive class



(a) Meta-learners individual predictions.



(b) Meta-learners' misclassification rates.

Figure 38 – SVM's meta-learners predictions considering their experimental setups which obtained the best AUC values. Figure from Mantovani *et al.* (2018b).

and "`Tuning`" is the negative class. Therefore, a False Negative (FN) is a wrong recommendation to perform HP tuning, while a False Positive (FP) is a wrong recommendation to use the default HP values. It is desirable to reduce as most as possible the number of FN, but balancing the FP occurrences to not lose performance.

Algorithms following different learning biases present different prediction patterns and this may be observed in Figure 38. Few meta-examples are classified correctly by all the meta-models. Looking at the patterns presented:

- kNN and GP minimize the FN rate (as may also be observed at Sub-figure 38b), classifying correctly most of the situations where defaults are recommended. However, they misclassified a lot of examples of "`Tuning`" class, penalizing the overall performance of the recommender system;

- SVM, CART and LR minimized the FP rate, classifying mostly corrected the situations where tuning improved statistically induced models. But, it suggests a behavior predicting mostly the majority class (`Tuning`), what is also not desirable;

A more balanced scenario is indeed provided by RF meta-models, reflecting their best performance values in experiments. The algorithm is not the best predicting none of the classes individually but reduced the error rates dealing with both classes.

Table 15 – Misclassified datasets by all the meta-learners in SVM meta-dataset.. For each dataset it is shown: the meta-example number (Nro); the OpenML dataset name (Name) and id (id); the number of attributes (D), examples (N) and classes (C); the proportion between the number of examples from minority and majority classes (P); the performance values obtained by defaults (Def) and tuned (Tun) HP settings assessed by BAC; and the truth label (Label).

Nro	Name	id	D	N	C	P	Def (sd)	Tun (sd)	Label
17	jEdit_4.0_4.2	1073	8	274	2	0.96	0.73 (0.01)	0.73 (0.01)	Defaults
36	banknote-authentication	1462	4	1372	2	0.80	0.99 (0.01)	0.99 (0.01)	Tuning
78	autoUniv-au7-500	1554	12	500	5	0.22	0.29 (0.01)	0.31 (0.01)	Tuning
97	optdigits	28	62	5620	10	0.97	0.99 (0.01)	0.99 (0.01)	Tuning

Table 15 list all the datasets misclassified by all the meta-learners as indicated in Figure 38a. Meta-examples with ids 17 ("`Defaults`") and 78 ("`Tuning`") were corrected labeled by the statistical meta-rule, so the misclassification may have occurred due to the lack of the descriptive ability of meta-features or noise in the meta-dataset. The other two meta-examples (36, 97) were both labeled as "`Tuning`" but the statistical difference indicated is very small in terms of performance, and which may indicate a limitation of the current meta-target rule criterion.

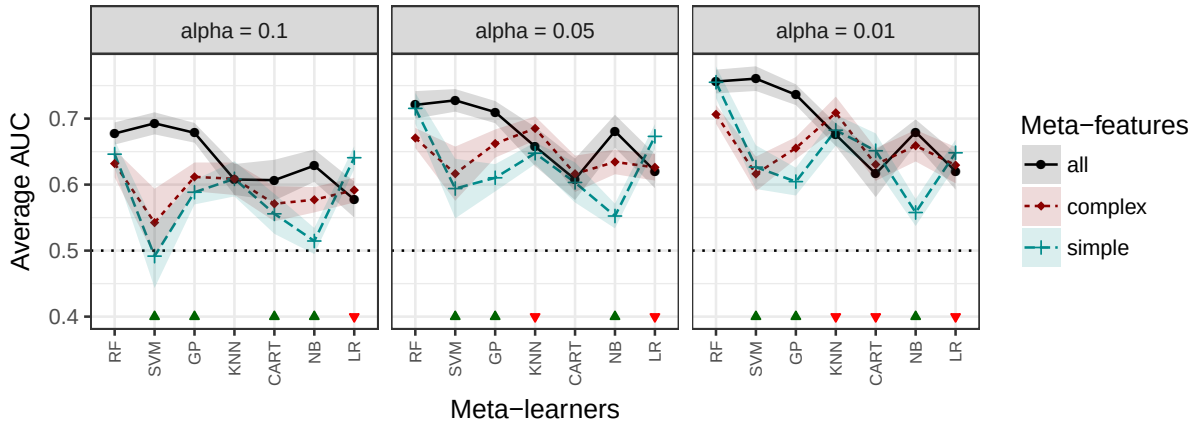
7.3 When to tune DTs?

The predictive performance of the meta-learners when applied to the DTs meta-datasets are summarized in Figure 39. The same approaches previously investigated with SVMs (`all`, `simple`, `complex`) were also applied in the DT target algorithms. The statistical results from pair-wise comparisons are also indicated at the x-axis. Predictive performance assessed in y-axis were measured with the AUC averaged over 20 repetitions. Plots also show the impact of different *alpha* (α) levels at the meta-target rule. For comparisons, it is important to mention that both Random and ZeroR[7] meta-models obtained AUC of 0.5 in all these meta-datasets[8]

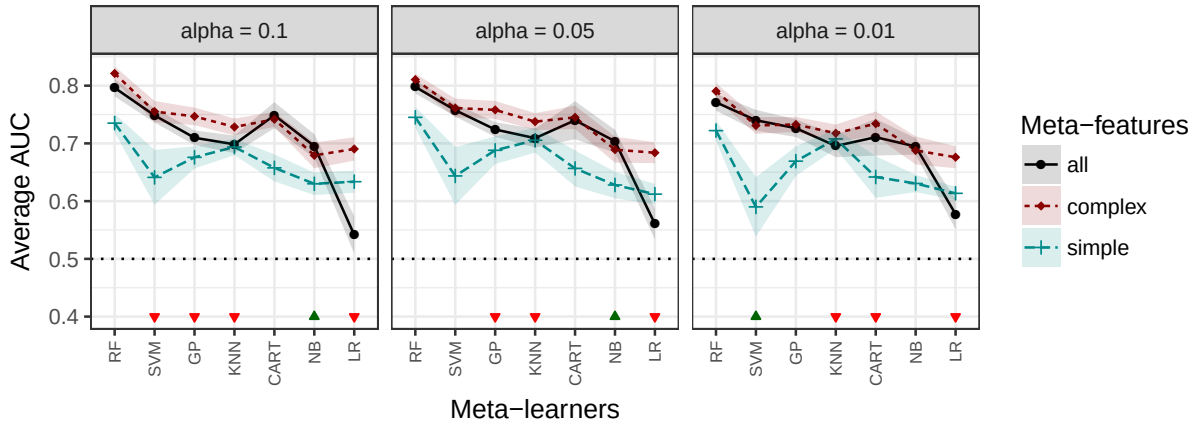
The best results considering J48 meta-datasets (Figure 39a) were obtained by the {RF, SVM, GP} meta-learners. They obtained their best predictive performances using preferably all the available meta-features. These meta-models achieved AUC values between $auc = (0.67, 0.74)$ for all the α values. It is interesting to note that a more rigid assumption at the meta-rule ($\alpha = 0.01$) provided the best results for some algorithms. It may be the case that predictions are tending to a specific class since this meta-dataset is imbalanced. When $\alpha = \{0.95, 0.99\}$, the RF meta-learner using only the simple meta-features also generated accurate meta-models

[7] This classifier simply predict the majority class.
[8] The AUC performance values were assessed using the implementations provided by `mlr` R package.

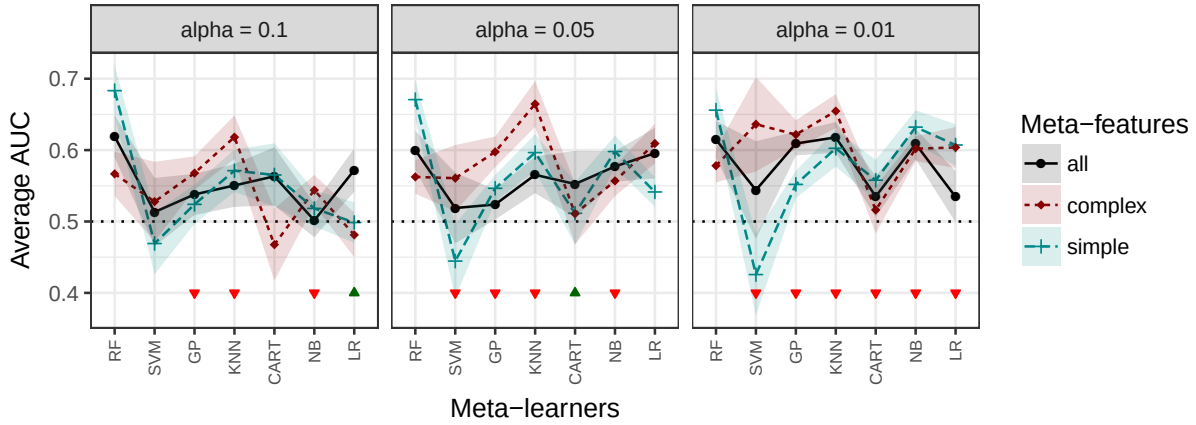
with AUC > 0.7. However, in general, the complete set of meta-features provided best results for most algorithms.



(a) AUC values for J48's meta-datasets.



(b) AUC values for CART's meta-datasets.



(c) AUC values for CTree's meta-datasets.

Figure 39 – Meta-learners average AUC results on DTs meta-datasets.

Figure 39b depicts the experimental results for CART meta-datasets. Best results were obtained by RF meta-models using data complexity meta-features, but with no significance in relation to the complete set of meta-features. These meta-models achieved AUC values near 0.8

for all α values. In fact, these two approaches ("complex" and "all") performed quite similar when explored by the meta-learners. The α value used by the meta-label rule seems no to influence meta-learners' performance. In addition to RF, all the algorithms, with the exception of NB and LR, induced meta-models with predictive performance $auc > 0.75$.

The results obtained by the meta-learners in CTree's meta-datasets are summarized in Figure 39c. Contrarily to the other two target algorithms, CTree is a highly imbalanced problem. Predictive performance values obtained in this scenario are the lowest regarding AUC. The best results were obtained by the kNN meta-models using data complexity meta-features, especially when $\alpha = \{0.95, 0.99\}$. RF could also induce accurate models, but only using the set of simple meta-features. As shown in Figure 31 from Section 7.1.3, the decision boundaries in the CTree problem are very fuzzy. This fact may be the reason why meta-learners did provide accurate models in this scenario.

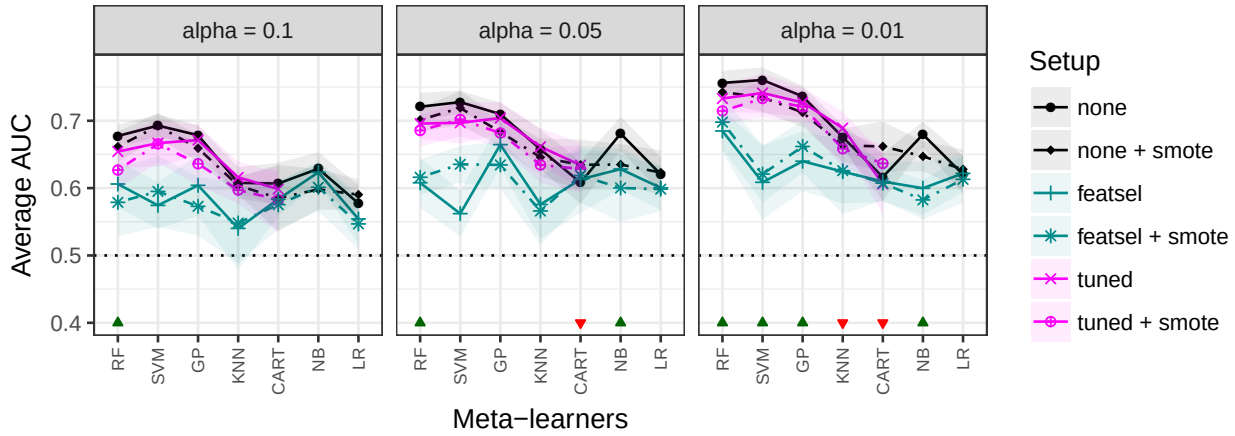
The best results for CART and J48 were obtained when inducing meta-models with all the available meta-features. Since it is possible to select different subsets from them, the next analyzes for these meta-datasets were performed using the complete set of meta-features. On the other hand, for CTree the maximum AUC value was obtained with RF and the simple descriptors. Thus, the "`simple`" meta-features were selected for the next analysis regarding CTree HP recommendation.

7.3.1 Evaluating different setups

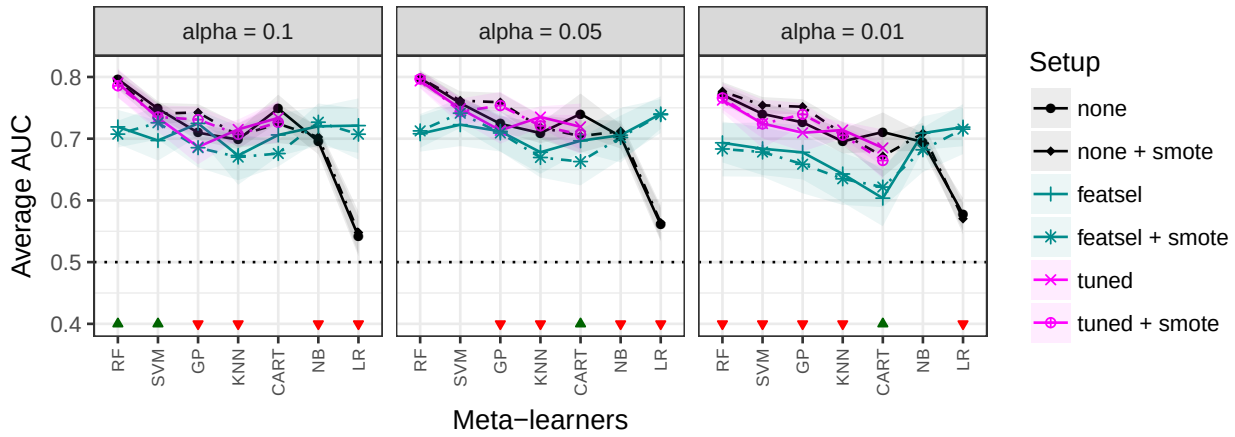
The three different setups were also evaluated when performing experiments with DT meta-datasets: meta-feature selection via SFS method (`featsel`); data imbalance with SMOTE (smote); and HP of the meta-learners via RS technique (`tuned`). The original meta-datasets, without any auxiliary process and namely `none`, are the baselines.

Figure 40 summarizes the main results of the experiments with the DT meta-datasets. The NB and LR algorithms do not have any tunable HP. Consequently, their results for `tuned` setups are missing. The top chart (Figure 40a) shows the average AUC values for the J48 meta-datasets considering all the meta-learners and α levels. In general, when considering AUC values obtained in original meta-datasets, improvements were obtained only for CART and KNN applying SMOTE at $\alpha = \{0.95, 0.99\}$. For the other algorithms, best meta-learners were still induced without any additional process.

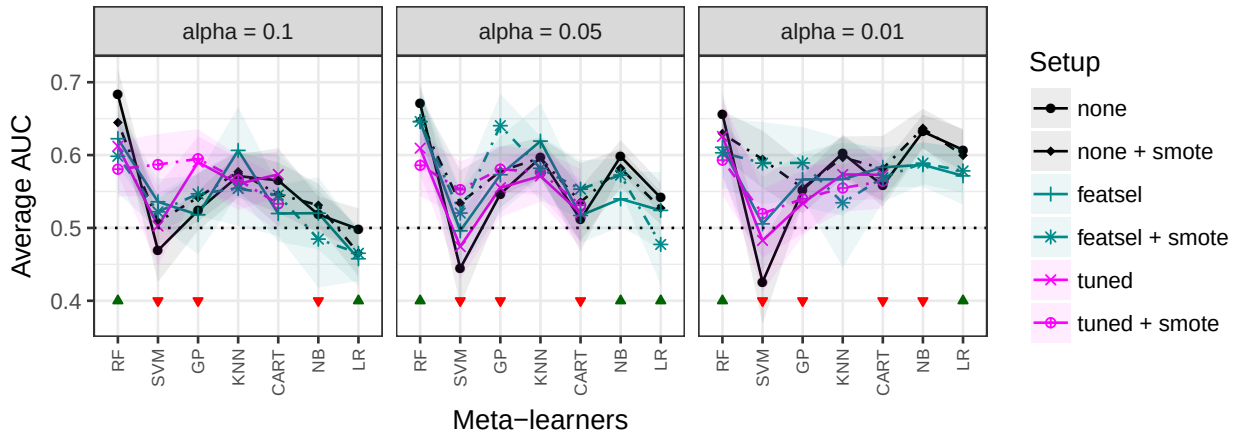
The middle figure (Figure 40b) presents results obtained in CART meta-datasets. Unlike the results achieved with the J48 target algorithm, the use of SMOTE improved the AUC values for most algorithms. Although improvements are small, they were statistically significant for {GP, KNN, NB} with any value of alpha (α). Regarding the HP tuning of the meta-learners, KNN and CART were the only algorithms with performances slightly improved by its use. However, CART improvements did not surpass AUC performances obtained using only SMOTE.



(a) J48 meta-datasets' average AUC values.



(b) CART meta-datasets' average AUC values.



(c) CTree meta-datasets' average AUC values.

Figure 40 – AUC performance values obtained by all meta-learners in DTs meta-datasets considering different experimental setups. Results are averaged over 20 repetitions.

The use of feature selection deteriorates all the algorithms' performances, except by the LR meta-learner. A linear classifier with just a subset of features could improve AUC values from 0.55 to 0.75. Results suggest that the hyperplane efficiency is maximized when a subset of features is considered, with classes becoming more separated.

The bottom figure (Figure 40c) presents results for CTree meta-datasets. Among the three target DT algorithms, this is the most irregular scenario mainly due to the high imbalance between classes. For this, the SMOTE technique could slightly improve meta-learners when $\alpha = \{0.90, 0.95\}$. The of use of HP tuning at the meta-level improved specifically SVM and GP meta-learners. However, these improvements may be related to an overfitting behavior, classifying all the meta-examples with the majority class label "(`Defaults`)". When feature selection is considered, {GP, KNN, CART} meta-learners had their performances improved, but statistically significant only for KNN.

The general picture shows that different setups could improve different algorithms. J48 had no improvement with additional learning processes. On the other hand, CART and CTree results were improved with the use of SMOTE. Overall, even presenting some statistically significant improvements, none of the target problems had their maximum performances improved by additional processes. Thus, it indicates a feature engineering is more important than using additional learning processes.

7.3.2 Meta-features importance

As explored with SVM's MtL results, RF meta-models were used to estimate the relative importance of the meta-features (based on the Gini impurity index). Figure 41 shows the average relative importance values for each meta-features in DT meta-datasets. The importance is shown for the experiments considering all meta-features and $\alpha = 0.05$ (middle case). RF meta-models obtained from J48 and CART meta-datasets were induced with all the available meta-features. For CTree, the RF meta-models used only the subset with the `simple` meta-features[9]. In all of the analyzes, results were achieved in meta-datasets considering $\alpha = 0.05$ (middle case). Relative importance values were also normalized between $[0, 1]$. Thus, it is possible to measure how the same meta-feature can influence different problems. In the x-axis meta-features are organized by their correspondent categories[10].

As expected, the figure shows that different meta-features are important for different problems. Disregarding CTree curve, which has one value for the simple meta-features, measures from different categories are important for the same problem. For example, the absolute correlation of the attributes (`abs_cor`) meta-features is important for solving J48 and CART problems. Another example is the number of `samples` which is important in CTree and CART tuning recommendations. However, in most cases, an important meta-feature is *activated* only

[9] Meta-features were selected according to the results shown in Figure 39.
[10] Meta-features' categories are listed in Tables 22 and 23 at Appendix B.

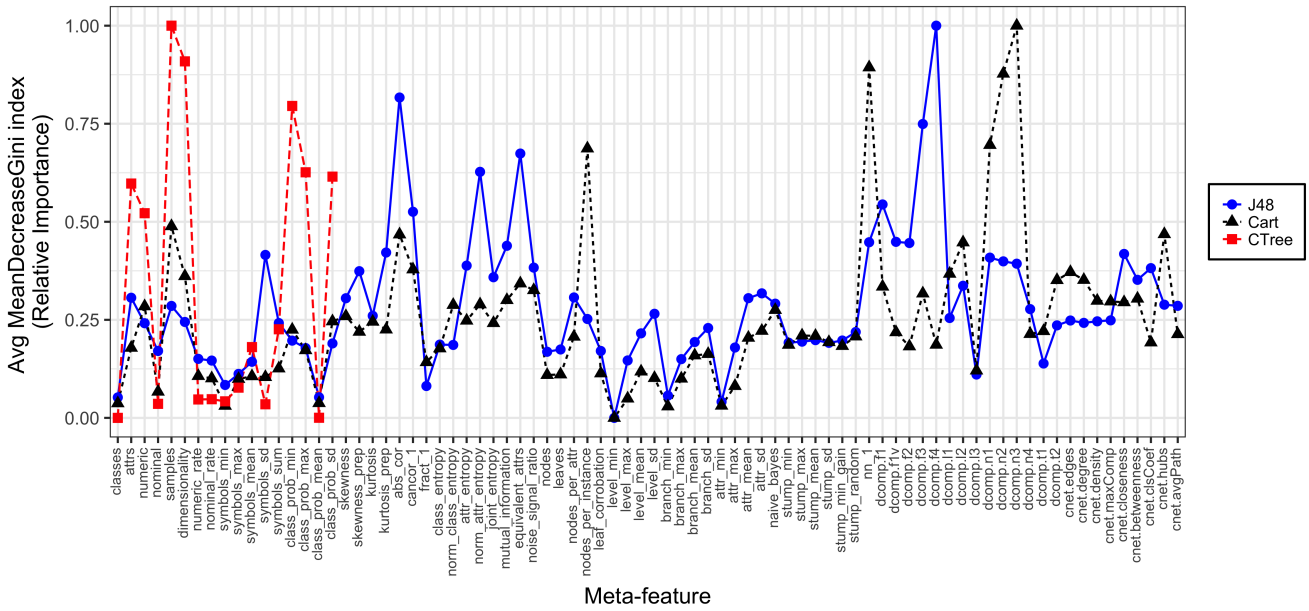


Figure 41 – Average meta-features' relative importance obtained from RF meta-models on DTs meta-datasets. Values are normalized in the interval $[0, 1]$ for each one of the meta-datasets.

in a single problem. The most important meta-feature for the J48 problem was the data complexity measure "`f4`". This feature describes the collective attribute efficiency in a dataset. The second most important one was a simple meta-feature: "`abs_cor`", a metric that measures the linear relationship between two attributes. This value is averaged over all pairs of attributes in the dataset. The top-3 is completed with another data complexity meta-feature: "`f3`", which describes the maximum individual attribute efficiency. The two data complexity features measure the discriminative power of the dataset's attributes, while the absolute correlation verifies if the information provided by attributes is not redundant. These most important metrics suggest that if a dataset has representative attributes, default HP values are robust to solve it. Otherwise, HP tuning is recommended.

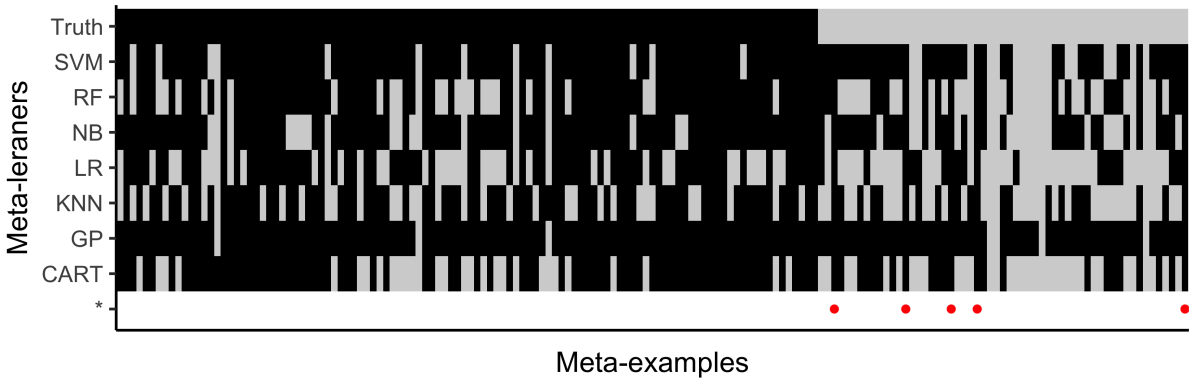
For the CART meta-dataset, the two most important meta-features are: "`n3`" "`nn`". The first one is a data complexity meta-feature, and the second one a traditional landmarking. Both measures estimate the error rate of the 1-NN classifier, by leave-one-out or cross-validation. They measure how close the examples of different classes are. Low values mean that there is a large gap in the class boundary. The third most important is also a data complexity meta-feature: "`n2`", that is the average intra-class and inter-class distances ratio used by a kNN algorithm to classy data examples. Low values indicate that examples from the same class lay closely in the feature space. In this way, it suggests that if the dataset's examples from the same class are close to each other, the 1-NN will find them closely in the feature space, and CART would also be capable solving it without tuning. On the other hand, examples would be more overlapped, and the HP tuning would fit better the learning bias of the algorithm.

The most important meta-features for the CTree meta-dataset are: the number of "`samples`"

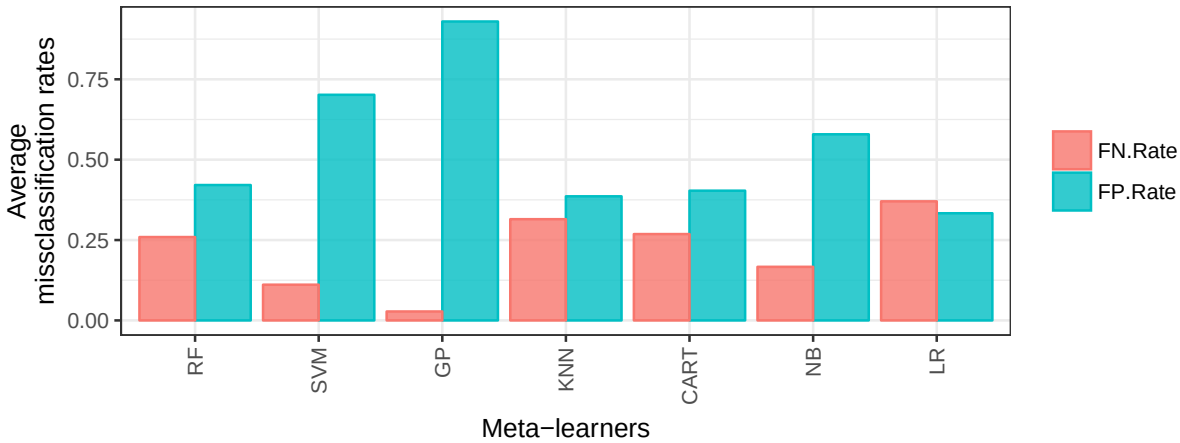
of the dataset; its "`dimensionality`"; and the proportion of examples belonging to the minority class ("`class_min`"). In CTree base-level results most of the datasets had better performances using default HP settings than tuned trees. These meta-features may suggest in which cases tuning is recommended. At a first look, tuning is most suitable for datasets with high dimensionality and imbalance rates. In all other cases, defaults are already robust.

In general, different aspects are being considered when choosing between tuned of default HP settings for the target algorithms: the discriminative power of the attributes is important to recommend the J48 tuning; the dispersion of datasets' examples from different classes to recommend CART tuning; and the dimensionality of a problem to recommend tuning for CTrees. At least for J48 and CART, it is important to verify the importance of the dataset's attributes before do the HP tuning, which makes sense for a DT induction algorithm.

7.3.3 Checking predictions



(a) Meta-learners individual predictions.



(b) Meta-learners' misclassification rates.

Figure 42 – Meta-learners' misclassifications in the J48 meta-dataset. Results consider the experimental setups which obtained best AUC values when at $\alpha = 0.05$.

The predictions obtained by the meta-learners in the DT meta-datasets can be useful to better understand results. In this sense, Figure 42 depicts the misclassifications predictions of all

the meta-learners considering their best experimental setups for the J48 meta-dataset. The figure shows all the individual predictions, with the x-axis listing all the meta-examples and the y-axis the meta-models. "`Defaults`" meta-examples are firstly shown in black, followed by '`Tuning`" ones in gray. The top line at y-axis, "`Truth`", shows the truth labels of the meta-examples, while the bottom line ("*") shows red points for meta-examples misclassified by all meta-learners.

In the experiments with DTs, the positive and negative classes are the same as defined for SVMs. The positive class recommends the use of `Defaults` HP settings, and the negative class recommends the HP "`Tuning`". Thus, FP is a wrong recommendation to use default HP values, while FN is a wrong recommendation to perform tuning. It is desirable for a meta-model to minimize FN and FP rates, predicting as most as possible both classes. By this convention, results from different target algorithms can be compared to each other.

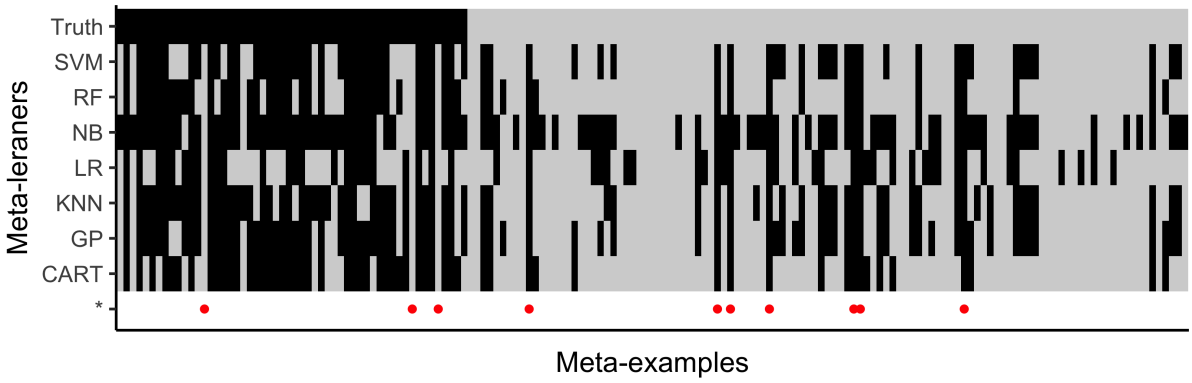
The top chart (Figure 42a) shows all the individual predictions for the J48 meta-dataset. Most of the "`Defaults`" labels are correctly predicted by GP and SVM meta-learners. However, they overfitted predicting wrongly the minority "`Tuning`" class. Conversely, LR and KNN were accurate to indicate when tuning is required, but also recommended the HP tuning for several datasets where it is not needed. More balanced predictions are provided by the RF meta-model (Figure 42b), reflecting a better performance compared with other algorithms.

All the meta-learners misclassified just five meta-examples, mostly multiclass problems and all labeled with the `Tuning` class. The main characteristics of these base-level datasets are listed in Table 16. Their meta-target values are correctly defined by the meta-target rule and, therefore, the misclassification may have occurred due to the lack of the descriptive ability of the meta-features or due to noise in the meta-dataset.

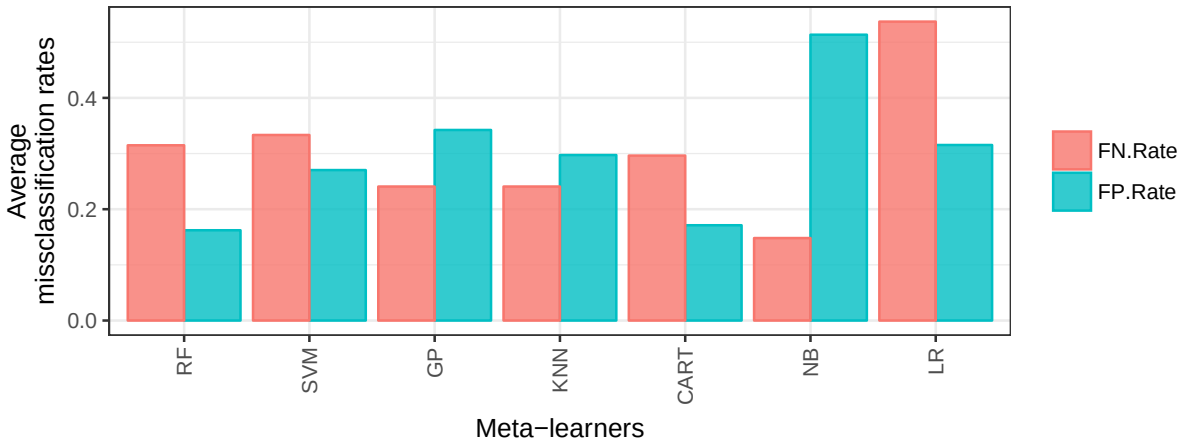
Table 16 – Misclassified datasets by all the meta-learners in J48 meta-dataset. For each dataset it is shown: the meta-example number (Nro); the OpenML dataset name (Name) and id (id); the number of attributes (D), examples (N) and classes (C); the proportion between the number of examples from minority and majority classes (P); the performance values obtained by defaults (Def) and tuned (Tun) HP settings assessed by BAC; and the truth label (Label).

Nro	Name	id	D	N	C	P	Def (sd)	Tun (sd)	Label
5	ar4	1061	29	107	2	0.23	0.65 (0.01)	0.70 (0.01)	Tuning
37	volcanoes-a1	1527	3	3252	5	0.02	0.58 (0.01)	0.71 (0.01)	Tuning
54	libras_move	299	90	360	15	0.46	0.68 (0.01)	0.70 (0.01)	Tuning
72	diggle_table_a2	694	8	310	9	0.44	0.96 (0.01)	0.97 (0.01)	Tuning
165	yeast_v7	40733	8	1269	4	0.01	0.62 (0.01)	0.64 (0.01)	Tuning

Figure 43a shows the individual predictions obtained in the CART meta-dataset. The use of defaults has the best predictions provided by {NB, KNN, GP} meta-models. However, the same meta-models also presented FP rates > 0.3. The NB, in particular, presented a FP rate > 0.5, wrongly predicting half of the meta-examples where HP is required. The misclassification rates of these meta-models can be observed in Figure 43b. Unlike the J48 problem, most of the meta-examples belong to the `Tuning` class. In this context, the tuning recommendation is most



(a) Meta-learners individual predictions.



(b) Meta-learners' misclassification rates.

Figure 43 – Meta-learners' misclassifications in the CART meta-dataset. Results consider the experimental setups which obtained best AUC values when at $\alpha = 0.05$.

correctly classified by RF and CART meta-learners. These two algorithms also provided the most balanced relation between the misclassification rates.

Table 17 list all the datasets misclassified by all the meta-learner in the CART meta-dataset. Meta-examples with ids ={40,147,156} were labeled to use default HP settings, while the other obtained best results with tuned trees. Looking to the values provided to the meta-target rule, all the meta-examples are correctly labeled, so the misclassification may have occurred due to a lack of descriptive information from the meta-features.

The individual predictions obtained in the CTree meta-dataset are shown in Figure 44a. Similarly to the J48 problem, most of the meta-examples belong to the `Default` class, but CTree meta-dataset presents the highest imbalanced rate among target algorithms. Even exploring different setups at the meta-level, in particular, the SMOTE technique, all the meta-models overfitted. Virtually, all the meta-examples belonging to the minority class are misclassified, and meta-models obtained FP rates > 0.6. Among the meta-models, the KNN obtain the most reduced error rates.

Table 17 – Misclassified datasets by all the meta-learners in CART meta-dataset. For each dataset it is shown: the meta-example number (Nro); the OpenML dataset name (Name) and id (id); the number of attributes (D), examples (N) and classes (C); the proportion between the number of examples from minority and majority classes (P); the performance values obtained by defaults (Def) and tuned (Tun) HP settings assessed by BAC; and the truth label (Label).

Nro	Name	id	D	N	C	P	Def (sd)	Tun (sd)	Label
40	volcanoes-a4	1530	3	1515	5	0.02	0.28 (0.01)	0.28 (0.01)	Defaults
147	wall-robot-navigation-v3	1526	4	5456	4	0.15	1.00 (0.00)	1.00 (0.00)	Defaults
156	steel-plates-fault	1504	33	1941	2	0.53	1.00 (0.00)	1.00 (0.00)	Defaults
17	pc1_req	1167	8	320	2	0.50	0.56 (0.01)	0.58 (0.01)	Tuning
55	credit-a	29	15	690	2	0.80	0.84 (0.01)	0.85 (0.01)	Tuning
58	grub_damage	338	8	155	4	0.39	0.39 (0.01)	0.42 (0.01)	Tuning
65	schizo	466	14	340	2	0.92	0.80 (0.01)	0.82 (0.01)	Tuning
85	autoUniv-au6-1000	1555	40	1000	8	0.37	0.19 (0.01)	0.22 (0.01)	Tuning
90	autoUniv-au7-500	1554	12	500	5	0.22	0.37 (0.01)	0.38 (0.01)	Tuning
112	hayes-roth	329	4	160	3	0.01	0.58 (0.01)	0.75 (0.01)	Tuning

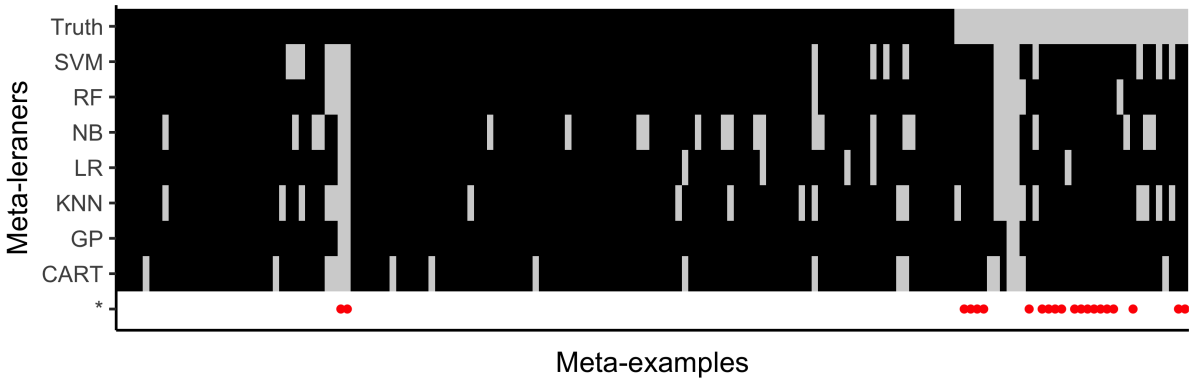
The relative landmarking meta-features, previously explored, were used only by SVMs due to the linearity hypothesis demonstrated by results. It may be the case that they work for DTs as well, but the experimental results suggest that other meta-features are important for these algorithms.

7.4 Projecting meta-models at the base-level

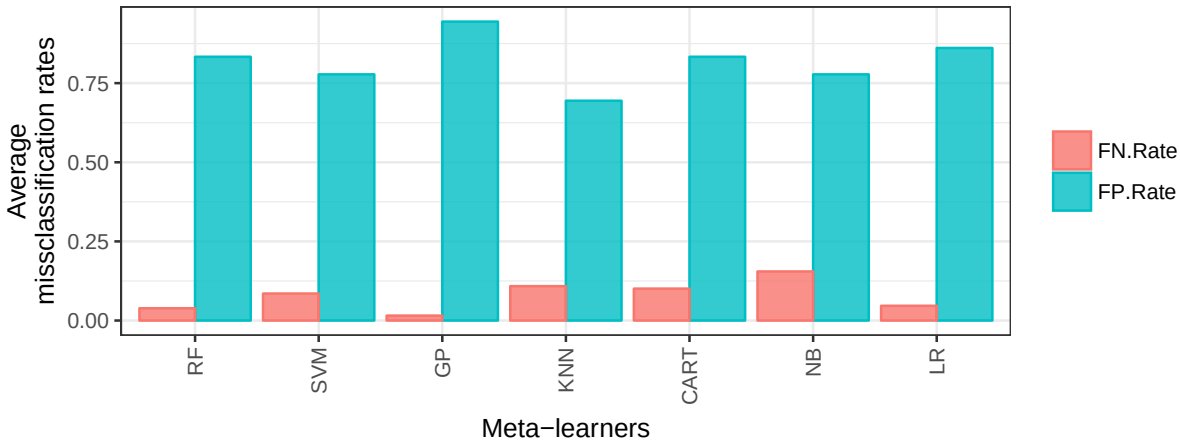
Even doing extensive analysis at the meta-level, it is important to evaluate the predictive performance of the MtL systems for the target algorithms at the base-level. In this line, MtL recommendations to use tuned or default HP settings were projected into the base-level datasets for each target algorithm. The best meta-learners identified in the experiments were compared with three simple baselines:

- `Tuning`: a model that always recommended the HP tuning;
- `Defaults`: a model that always recommended the use of defaults; and
- (`Random`): a model that provided random recommendations.

Figure 45 depicts a scatter plot with the projected BAC performance and runtime values averaged over all the base-level datasets. Results are presented separately for each algorithm. Overall, performing the HP tuning always has the highest average BAC value, except for CTree, which is not so sensitive to HP tuning. However, it is the most expensive approach for all algorithms. Using default HP values is trivially the fastest approach. It presented the lower average performance value for SVM, J48 and CART target algorithms. For CTree, it performed quite good, since in most of the base-level datasets induced models were not improved by tuning. In all the scenarios, the `Random` meta-learner stays in the middle case between always tuning or using default HP settings.



(a) Meta-learners individual predictions.



(b) Meta-learners' misclassification rates.

Figure 44 – Meta-learners' misclassifications in the CTree meta-dataset. Results consider the experimental setups which obtained best AUC values when at $\alpha = 0.05$.

Through the figure it is possible to identify three different situations when the MtL system is used:

- **CART** and **SVM**: in these meta-datasets, "`Tuning`" is mostly, but not always recommended. The proposed meta-models are above `Random` and `Defaults` baselines, performing closely to the `Tuning` baseline but with lower average runtime costs. They are also considerably near the ground truth (`Truth`), specially for CART;

- **J48**: mostly, but not all of the meta-examples are labeled with `Defaults`. The proposed meta-models are above `Defaults`, but relatively close to the `Random` and `Tuning` baselines. The average BAC values of all the approaches are very close, even considering all the baselines. This can be noted by the scale at the y-axis. However, all the meta-models have lower average runtime when compared with `Random` and `Tuning` baselines;

- **CTree**: this meta-dataset is highly imbalanced, with almost all of the meta-examples labeled to use default HP settings. Besides that, the CTree *HP profile* shows lower perfor-

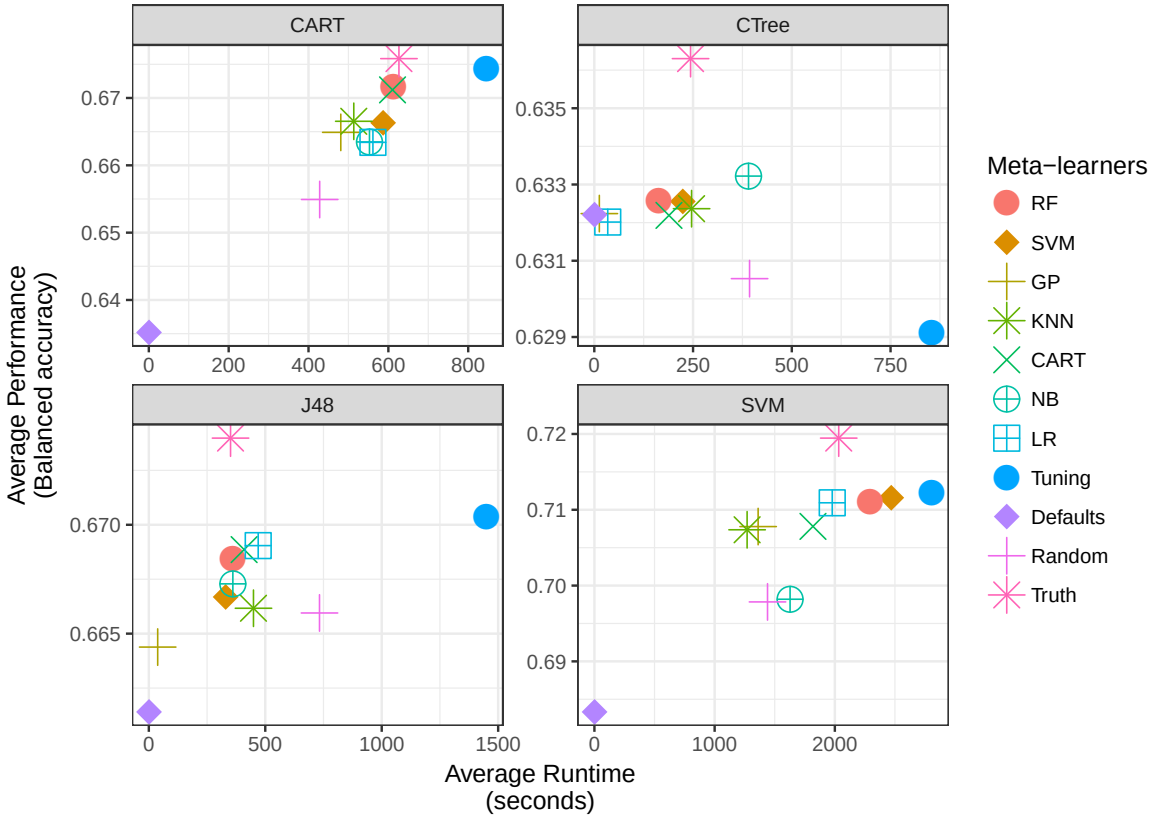
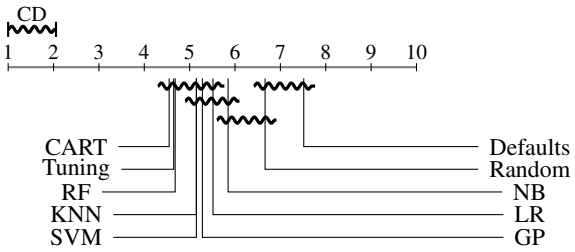
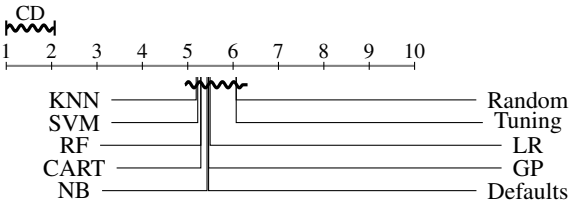


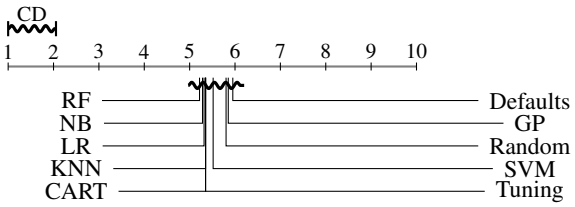
Figure 45 – Performance of the meta-learners projected into the base-level datasets. Figure adapted from Mantovani *et al.* (2018b).



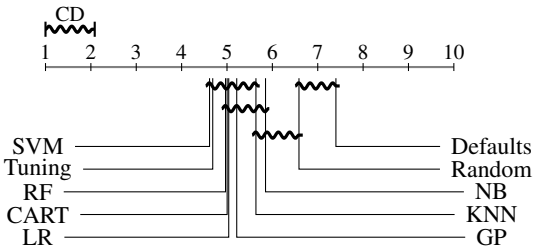
(a) CART CD diagram with $\alpha = 0.05$.



(b) CTree CD diagram with $\alpha = 0.05$.



(c) J48 CD diagram with $\alpha = 0.05$.



(d) SVM CD diagram with $\alpha = 0.05$.

Figure 46 – Comparison of the BAC values of the meta-model approaches according to the Nemenyi test. Groups of techniques that are not significantly different are connected.

mances when tuning is applied[11]. Thus, the proposed meta-models are above `Random` and `Tuning` (the worst) baselines. They also perform closely to the `Defaults` baseline (the best), but not to the ground `Truth`. These results reflect somewhat the overfitting observed in the meta-level learning task.

The Friedman test (DEMŠAR, 2006), with a significance level of $\alpha = 0.05$, was also used to assess the base-level predictions. The null hypothesis states that all the meta-learners and baselines are equivalent regarding the average predictive BAC performance. When the null hypothesis is rejected, the Nemenyi post-hoc test is also applied to indicate when two different techniques are significantly different. Figure 46 presents the CD diagrams for the target algorithms.

CD diagram at Figure 46a shows the statistical results for CART. The approach using always default HPs (`Defaults`) is ranked last, followed by the `Random` baseline. Almost all meta-learners are significantly better than both and are equivalent to `Tuning`, which always performs the HP tuning. The same can be said for SVM's results in Figure 46d. For J48 and CTree (Figures 46c and 46b), most of the meta-models are better ranked than baselines, but, overall, there are no statistical differences among the evaluated approaches.

With many datasets at the base-level, it is also correct to emphasize that the overall gain is diluted between them. Even so, the meta-learners could considerably reduce the computational costs related to tuning, keeping an accurate performance on recommendations. Finally, the proposed MtL recommender system proves to be suitable for the target algorithms most sensitive to the HP tuning (CART, SVM). Statistically significant results were obtained with a lower runtime associated. On the other hand, when a *HP profile* describes an algorithm more likely to use default HP settings, the MtL system does not provide statistically significant results.

7.5 Chapter remarks

This chapter proposed a new MtL recommender system to support the identification of cases where HP tuning can improve the predictive performance of classification algorithms. For such, it takes advantage of large-scale experiments tuning SVMs and DT induction algorithms over a comprehensive data collection.

Unlike the two related studies in literature (MOLINA *et al.*, 2012; RIDD; GIRAUD-CARRIER, 2014), the meta-level experiments described in this Chapter explored a large number of heterogeneous datasets, an unbiased tuning methodology, and induced meta-models for each of the four target algorithms: SVM, J48, CART, and CTree. The experimental setup also included meta-features from several categories and different learning processes.

[11] See Figure 21 at Section 6.4.3.

The obtained results presented accurate meta-models ($0.7 < auc < 0.8$) for most of the target algorithms. Further, these meta-models could be improved when SMOTE was used to reduce the data imbalance of the meta-datasets. For the CTree algorithm, in particular, meta-models predictive performances could not surpass $auc < 0.7$. It is clear that data imbalance is one of the main problems, but it may be the case the meta-datasets are not properly described by the available meta-features.

A descriptive analysis of the meta-models indicated which characteristics may be responsible for the HP tuning necessity for the target algorithms. The tuning of SVMs is strongly related to the non-linearity of the input problem. J48 requires tuning when the datasets' attributes are not descriptive enough, while CART tuning is recommended when dataset's examples are overlapped into the feature space. Even CTree not being sensitive to the HP tuning, when it is required the datasets are high dimensional with many meta-examples.

The last experiments described in this chapter showed that the proposed MtL recommender system is an interesting tool to reduce time spent with optimization processes, keeping the predictive performances of the induced models. Depending on the target algorithm, the effectiveness of the MtL recommender system can be statistically significant when projecting results to the base-level learning. It was observed especially for the target algorithms CART and SVM, which present the most sensitive *HP profiles*. The other two DT algorithms, J48 and CTree, had their default HP settings robust for most of the datasets. However, from a user's point of view, it could be interesting to follow MtL recommendations to perform tuning, although the results were not statistically significant.

CHAPTER

8

CONCLUSIONS

The increasing availability of data has increased the popularity of ML solutions able to relieve humans from many risky, repetitive or tedious activities. In many cases, this requires that ML algorithms are used in new and innovative ways. This development process is heavily based on human experts to perform manual tasks such as data preprocessing, feature engineering and evaluation of several possible ML algorithms. Hence, as the complexity of such solutions increases, so does the demand for automated solutions that can be used easily and without high human intervention. In this sense, the so-called Automated ML solutions (AutoML) can increase the widespread use of efficient ML solutions and free data scientists and practitioners from repetitive and time-consuming tasks. Thus, more applications can benefit from the use of ML and data scientists can allocate their time on more creative and important tasks than fine tuning algorithms (FEURER *et al.*, 2015).

Most of the algorithms employed in AutoML systems have HPs, which usually affect their predictive performance. Therefore, a recurrent task in AutoML design is tuning these HPs. Several authors explored optimization techniques to automatically search for suitable HP settings (BERGSTRA; BENGIO, 2012; KOTTHOFF *et al.*, 2016; LÓPEZ-IBÁÑEZ *et al.*, 2016). Although HP tuning may lead to more accurate models, the optimization process for finding these HP settings is still very time-consuming. Recently, MtL has been efficiently applied in the HP tuning task to overcome these limitations, suggesting initial points to an optimization technique (GOMES *et al.*, 2012; MIRANDA *et al.*, 2012; FEURER *et al.*, 2015) or estimating the predictive performance of a model given a HP setting (EGGENSPERGER *et al.*, 2018; WISTUBA; SCHILLING; SCHMIDT-THIEME, 2018).

Nevertheless, there is no guarantee that tuning will generate better results than just using the default HP settings provided by ML packages and tools. Therefore, when computational resources are limited, knowing beforehand which approach is more adequate (tuning or using defaults) can reduce the computational cost of the ML task. Also, it is also important to know which technique to use when tuning is required.

This thesis proposed a new MtL recommender system to indicate in which situations HP tuning should be applied. This objective is accomplished by proposing a framework able to predict when HP tuning is expected to lead to models with statistically significant improvements in predictive performance after identifying the *HP profile* of ML algorithms.

The experiments presented in this thesis improve the state of the art of MtL and of HP tuning in the following ways:

- providing an enhanced experimental methodology for HP tuning and MtL tasks (code available);
- presenting a large set of experiments with SVMs to identify their *HP profile* and the effectiveness of a simple RS technique;
- presenting a large set of experiments with DTs to find their *HP profiles*, improving the identification of when it is better to use tuned or default HP values;
- proposing a MtL recommender system capable of generalizing several learning processes into a single modular framework, along with the possibility of assigning different ML algorithms and HP optimization techniques.

8.1 Main contributions and results

The first contribution from this Ph.D. thesis is a novel evaluation methodology adopted to perform base-level HP tuning and meta-level experiments, which is unusual in the related literature. HP tuning results were obtained by an enhanced and reproducible methodology, using a large number of heterogeneous datasets, tuning techniques with different biases, and evaluating results with a measure that considers the imbalance between classes in real datasets, the BAC[1] measure.

There is no consensus in the literature regarding the utility of MtL for HP tuning, and each work generates meta-knowledge following different assumptions. However, most of the studies have base-level tuning using simple GPs or even simply doing HP sweeps over the hyperspace.

At the meta-level, the results reported in this thesis were obtained considering a wide set of options. Our experimental setup also included different ML algorithms as meta-learners, and not only an instance-based algorithm (kNN), as used in most studies. The complete pipeline also considered data preprocessing techniques to deal with tasks like data imbalance treatment (SMOTE), feature selection (SFS) and the tuning of these meta-learners (RS). As described in Section 3.4, none of the related studies had applied these options together as the MtL system proposed in this thesis.

[1] Balanced per class accuracy.

The third important aspect regarding the experimental methodology is the reproducibility of the experiments, which is also usually not guaranteed by related work. Although some studies provide their source code, few offer the complete experimental results. Thus, the automated and reproducible methodology (BISCHL *et al.*, 2016; VANSCHOREN *et al.*, 2013; CASALICCHIO *et al.*, 2017) adopted throughout this thesis generated a valuable contribution to the MtL community as meta-knowledge for further research[2].

8.1.1 SVM HP profile

SVMs are highly sensitive to their HP values and, therefore, widely explored by recent studies (PADIERNA *et al.*, 2017; LORENA *et al.*, 2018). However, there is a lack of studies benchmarking different techniques when performing their tuning. In fact, there is another study benchmarking different SVM implementations (HORN *et al.*, 2016), but the authors compared models tuned by the same technique (SMBO). The experiments carried out with simple and population-based techniques showed that only few HP evaluations are required to reach good solutions in the SVM hyperspace. Further, all the techniques performed statistically better than models induced by default HP values. However, when compared to each other, there were no statistical differences. Thus, to the best of our knowledge, the experiments described in Chapter 5 were the first showing the effectiveness of a simple RS technique for SVM HP tuning (MANTOVANI *et al.*, 2015a).

Further, experimental results also showed that default HP values proposed by SVM tools [3] are not suitable for a large number of classification tasks. Thus, the experiments described in section 5.6 present a new approach, namely "*Optimized defaults*", which addresses these limitations by optimizing new SVM HPs based on a group of heterogeneous datasets. The use of this new set of HP values produced significantly better models than the default values suggested by ML tools (MANTOVANI *et al.*, 2015b). Experimental results showed that it is a promising approach which could be extended to different ML algorithms.

8.1.2 DT HP profiles

Although high predictive performance is the most adopted measure to evaluate ML algorithms, in many applications, easy interpretability of the results is also an important requirement. This characteristic is found in DT induction algorithms (ROKACH; MAIMON, 2014). Although there are several studies investigating HP tuning for SVMs, the same is not true for DTs. There are some prior studies investigating the evolutionary design of new DT induction algorithms (BARROS *et al.*, 2012; BARROS; CARVALHO; FREITAS, 2015), but only a few on HP tuning for DT (REIF; SHAFAIT; DENGEL, 2011; MOLINA *et al.*, 2012; REIF *et al.*, 2014).

[2] Code and results are publicly available. A list of the correspondent repositories can be found at Table 5.
[3] At this point, they come from LibSVM and WEKA implementations.

Hence, the set of experiments described in Chapter 6 is the first large study investigating the *HP profile* of DT algorithms (MANTOVANI *et al.*, 2016; MANTOVANI *et al.*, 2018a).

Experiments were carried out with three popular DT algorithms in more than 94 classification datasets. The obtained results showed that each algorithm has its *HP profile*, which impacts the results obtained by the tuning techniques. HP tuning on DTs presented different scenarios, with some contradictory results for different DTs. The CART was the only algorithm to present a high sensitivity to HP tuning. For J48 and CTree, the improvements obtained by tuning techniques can be very small. Thus, for some optimization problems, it is better just to use the default HP settings. When comparing tuning techniques, different approaches are more suitable for different budget sizes. If the user has a large budget, Irace is indicated. On the other hand, PSO and SMBO are the most suitable techniques with a faster convergence when tuning DTs.

The fANOVA analysis (HUTTER; HOOS; LEYTON-BROWN, 2014) reported in Section 6.6 also indicated that few of the HPs are effectively responsible for the predictive performance of the induced trees. Similar results but with different ML algorithms were reported in ML (RIJN; HUTTER, 2017). In this sense, fANOVA may be used to reduce the search HP space and time spent with the optimization. The last experiments described in the chapter showed that a higher number of statistical improvements were obtained when a reduced HP space was used to tune J48 and CTree algorithms. Although this principled tuning of models was demonstrated with DTs, it may be easily explored for any ML algorithm with HPs.

8.1.3 MtL recommender system

All HP tuning results with SVMs and DTs showed that tuning is not always necessary. In many situations, default HP settings are enough to induce accurate models. Chapter 7 presented a new MtL recommender system to support the identification of cases where HP tuning can improve the predictive performance of classification algorithms. Similar studies were also reported in the literature (MOLINA *et al.*, 2012; RIDD; GIRAUD-CARRIER, 2014), but, unlike them, the system proposed in this thesis was the first to predict the necessity of HP tuning inducing meta-models for individual target algorithms (MANTOVANI *et al.*, 2015c) and exploring a wider experimental setup with heterogeneous datasets (MANTOVANI *et al.*, 2018b).

Experiments were performed with four different target algorithms, evaluating different experimental setups at the meta-level. The obtained results resulted in accurate meta-models for most the target algorithms, with AUC values in the interval $auc \in (0.68, 0.83)$, especially when SMOTE was used to reduce the data imbalance of the meta-datasets. For the CTree algorithm in particular, the HP tuning recommendation problem was highly imbalanced in favor of the use of default values.

The descriptive analyses in Sections 7.2.2 and 7.3.2 indicate which data characteristics

may be indicate the HP tuning necessity for the target algorithms. The tuning of SVMs is strongly related to the non-linearity of the input problem. J48 requires tuning when the predictive attributes of a dataset are not descriptive enough, while CART tuning is recommended when the examples in a dataset are overlapped into the feature space. Even CTree, which does not require tuning for most datasets, it is recommended for high-dimensional datasets with many examples.

The last experiments described in this chapter showed that the proposed MtL recommender system reduced the optimization time without reducing the predictive performance of the induced models. Depending on the target algorithm, the effectiveness of the MtL recommender system was statistically significant when projecting results to the base-level datasets. This behavior was observed especially for the target algorithms CART and SVM, which were the most sensitive to tuning according to their *HP profiles*. For the other two DT algorithms, J48 and CTree, their default HP settings were robust for most of the datasets. However, from a user's point of view, it could be interesting to follow MtL recommendations to perform tuning, although the results were not statistically significant.

8.2 Publications

Several papers have been published during the development of the research for this thesis. Hence, part of the results reported throughout this thesis can be found in these publications, as presented next:

8.2.1 Papers originated from thesis

- R. G. Mantovani, A. L. D. Rossi, J. Vanschoren, B. Bischl, A. C. P. L. F. de Carvalho. **Effectiveness of Random Search in SVM hyper-parameter tuning**. IJCNN 2015: 1-8;

- R. G. Mantovani, A. L. D. Rossi, J. Vanschoren, B. Bischl, A. C. P. L. F. de Carvalho. T**o tune or not to tune: Recommending when to adjust SVM hyper-parameters via meta-learning**. IJCNN 2015: 1-8;

- R. G. Mantovani, A. L. D. Rossi, J. Vanschoren, A. C. P. L. F. de Carvalho. **Meta-learning Recommendation of Default Hyper-parameter Values for SVMs in Classification Tasks**. MetaSel@PKDD/ECML 2015: 80-92;

- R. G. Mantovani, T. Horváth, R. Cerri, J. Vanschoren, A. C. P. L. F. de Carvalho. **Hyper-Parameter Tuning of a Decision Tree Induction Algorithm**. BRACIS 2016: 37-42;

- R. G. Mantovani, T. Horvath., R. Cerri, J. Vanschoren, A. C. P. L. F. de Carvalho. **Tuning Trees: on hyperparameter optimization for decision trees**. 2018, pg 1-54 (under review - Soft Computing journal);

- R. G. Mantovani, A. L. D. Rossi, E. Alcobaça, J. Vanschoren, A. C. P. L. F. de Carvalho. **A meta-learning recommender system for hyperparameter tuning: Suggesting when tuning improves SVM classifiers**. 2018, pg 1-38 (under review - Data Mining and Knowledge Discovery journal).

8.2.2 Collaborations in the same research topic

- T. Horváth, R. G. Mantovani, A. C. P. L. F. de Carvalho. **Effects of Random Sampling on SVM Hyper-parameter Tuning**. ISDA 2016: 268-278;

- B. Bischl, G. Casalicchio, M. Feurer, F. Hutter, M. Lang, R. G. Mantovani, J. N. van Rijn, J. Vanschoren. **OpenML Benchmarking Suites and the OpenML100**. 2017, pgs 1-6 (under review - Journal of Machine Learning Open Source Software);

- M. P. Basgalupp, R. C. Barros, A. G. C. de Sá, G. L. Pappa, R. G. Mantovani, A. C. P. L. F. de Carvalho, A. A. Freitas. **An Extensive Experimental Evaluation of Automated Machine Learning Methods for Recommendation Classification Algorithms**. 2018, pg 1-25 (under review - Evolutionary Computing journal);

- T. Horvath, R. G. Mantovani, A. C. P. L. F. de Carvalho. **Fast Data Characterization for Meta-learning: Untapped Potentialities of PCA Meta-Features**, (in preparation), pgs 1-30, 2018.

8.3 Limitations

During the base-level experiments, there were several difficulties to generate the meta-knowledge. The process itself is computationally expensive, since a large number of tuning tasks must be run and evaluated on a wide range of datasets. Initially, a larger number of datasets were selected, but some of them were extremely expensive for HP tuning and meta-feature extraction. A *walltime* of 100 hours was then defined and the *problematic* datasets were filtered out from the dataset collection used in the experiments.

The thesis did not discuss questions regarding the"*tuning of the tuning techniques*". In the tuning experiments carried out, all the default settings provided by the implementations of the optimization techniques were used. In fact, most of these default values have been evaluated in benchmarking studies and reported to provide good predictive performance (BISCHL *et al.*, 2017; CÁCERES; LÓPEZ-IBÁÑEZ; STÜTZLE, 2014), while PSO's showed to be robust in a high number of datasets. EDAs are known to be sensitive concerning their population sizes, while there is no standard choice for the GA's parameter values (MILLS; FILLIBEN; HAINES, 2015). Even adapting both to handle mixed HP spaces they performed poorly when tuning DTs. These results suggest that fine-tuning of the GA HPs would be needed. Since this would

considerably increase the cost of experiments by adding a new tuning level, and most of the techniques performed well with default values, this additional tuning step was not assessed in this thesis.

The class imbalance at the meta-level was another problem that needed to be looked at. At least for SVMs, the optimized default HP settings (MANTOVANI *et al.*, 2015b) were added to the experimental setup, maximizing the number of meta-examples labeled with the class "'`Default`'.

Moreover, in all the target meta-datasets, some of the meta-examples were never correctly predicted. These results suggest that the current meta-feature sets do not extract enough information to describe characteristics relevant to the recommendation process. In fact, the results suggested that a meta-feature engineering step is more important than adding learning processes to the MtL pipeline. Thus, further investigations are necessary to discover new alternatives for data characterization that could improve the predictive performance of the meta-models.

8.4 Future Work

The main findings also point out possible future research directions. First, the proposed MtL recommender system could be extended to different ML algorithms, supporting the tuning decision in different domains. Therefore, it would be possible to provide more accurate suggestions exploring different data transformation methods at the meta-learning level, such as data dimensionality techniques.

As suggested by the experiments, a feature engineering study could improve current results. Thus, it would be interesting to investigate different meta-features to characterize datasets. Also, a multicriteria objective function could replace the current meta-label rule, weighing predictive performance, runtime and predictions. Another possibility would be to explore the use of ensembles, given the complementary behavior of some of the algorithms studied here as meta-learners.

Experiments described in Section 6.7 also open new possibilities for automated HP tuning systems. Given the results reported there, a MtL system could be developed to recommend in which situations it is interesting to perform HP tuning with the complete or reduced HP set. Thus, a more focused tuning could be performed using a smaller number of evaluations and only the most important HPs.

Another possibility of research is to expand the idea of AutoML system covered in this thesis. In this case, the system would not only recommend when HP tuning is necessary but also preprocessing methods, ML algorithms, and post-processing analysis able to explain the experimental results. Thus, given a new dataset, the AutoML system would recommend the most suitable components to solve the input problem. In fact, our research groups have already begun

work in this direction.

BIBLIOGRAPHY

ABDULRAHMAN, S. M.; BRAZDIL, P.; RIJN, J. N. van; VANSCHOREN, J. Speeding up algorithm selection using average ranking and active testing by introducing runtime. **Machine Learning**, v. 107, n. 1, p. 79–108, 2018. Available: <https://doi.org/10.1007/s10994-017-5687-8>. Citation on page 42.

ALI, S.; SMITH-MILES, K. A. A meta-learning approach to automatic kernel selection for support vector machines. **Neurocomputing**, v. 70, n. 13, p. 173–186, 2006. Citations on pages 26, 39, 43, and 44.

ANDRADOTTIR, S. A review of random search methods. In: FU, M. C. (Ed.). **Handbook of Simulation Optimization**. [S.l.]: Springer New York, 2015, (International Series in Operations Research & Management Science, v. 216). p. 277–292. Citations on pages 33 and 67.

BACHE, K.; LICHMAN, M. **UCI Machine Learning Repository**. 2013. Available: <http://archive.ics.uci.edu/ml>. Citation on page 93.

BARDENET, R.; BRENDEL, M.; KÉGL, B.; SEBAG, M. Collaborative hyperparameter tuning. In: DASGUPTA, S.; MCALLESTER, D. (Ed.). **Proceedings of the 30th International Conference on Machine Learning (ICML-13)**. [S.l.]: JMLR Workshop and Conference Proceedings, 2013. v. 28, n. 2, p. 199–207. Citations on pages 26, 32, 33, and 34.

BARROS, R.; BASGALUPP, M.; CARVALHO, A. de; FREITAS, A. A survey of evolutionary algorithms for decision-tree induction. **Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on**, v. 42, n. 3, p. 291–312, May 2012. Citations on pages 75, 76, and 127.

BARROS, R. C.; CARVALHO, A. C. P. L. F. de; FREITAS, A. A. **Automatic Design of Decision-Tree Induction Algorithms**. Springer, 2015. (Springer Briefs in Computer Science). ISBN 978-3-319-14230-2. Available: <http://dx.doi.org/10.1007/978-3-319-14231-9>. Citations on pages 75 and 127.

BEN-HUR, A.; WESTON, J. A user's guide to support vector machines. In: **Data Mining Techniques for the Life Sciences**. [S.l.]: Humana Press, 2010, (Methods in Molecular Biology, v. 609). p. 223–239. Citations on pages 32 and 56.

BENDTSEN., C. **pso: Particle Swarm Optimization**. [S.l.], 2012. R package version 1.0.3. Available: <https://CRAN.R-project.org/package=pso>. Citation on page 61.

BENSUSAN, H.; GIRAUD-CARRIER, C.; KENNEDY, C. A higher-order approach to meta-learning. In: **Proceedings of the ECML - Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination**. [S.l.: s.n.], 2000. p. 109–118. Citation on page 41.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. **J. Mach. Learn. Res.**, v. 13, p. 281–305, Mar. 2012. Citations on pages 26, 32, 33, 34, 67, 78, and 125.

BERGSTRA, J.; YAMINS, D.; COX, D. D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: **Proc. 30th Intern. Conf. on Machine Learning**. [S.l.: s.n.], 2013. p. 1–9. Citation on page 34.

BERGSTRA, J. S.; BARDENET, R.; BENGIO, Y.; KéGL, B. Algorithms for hyper-parameter optimization. In: SHAWE-TAYLOR, J.; ZEMEL, R. S.; BARTLETT, P. L.; PEREIRA, F.; WEINBERGER, K. Q. (Ed.). **Advances in Neural Information Processing Systems 24**. [S.l.]: Curran Associates, Inc., 2011. p. 2546–2554. Citations on pages 26, 33, and 34.

BERMÚDEZ-CHACÓN, R.; GONNET, G. H.; SMITH, K. **Automatic problem-specific hyper-parameter optimization and model selection for supervised machine learning: Technical Report**. Zürich, 2015. Citation on page 93.

BIRATTARI, M.; YUAN, Z.; BALAPRAKASH, P.; STÜTZLE, T. F-race and iterated f-race: An overview. In: ____. **Experimental Methods for the Analysis of Optimization Algorithms**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 311–336. ISBN 978-3-642-02538-9. Citations on pages 36 and 78.

BISCHL, B.; LANG, M.; KOTTHOFF, L.; SCHIFFNER, J.; RICHTER, J.; STUDERUS, E.; CASALICCHIO, G.; JONES, Z. M. mlr: Machine learning in r. **Journal of Machine Learning Research**, v. 17, n. 170, p. 1–5, 2016. Available: <http://jmlr.org/papers/v17/15-066.html>. Citations on pages 58, 60, 103, and 127.

BISCHL, B.; RICHTER, J.; BOSSEK, J.; HORN, D.; THOMAS, J.; LANG, M. mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions. 2017. Available: <http://arxiv.org/abs/1703.03373>. Citations on pages 60, 61, and 130.

BRAGA, I.; CARMO, L. P. do; BENATTI, C. C.; MONARD, M. C. A note on parameter selection for support vector machines. In: CASTRO, F.; GELBUKH, A.; GONZÁLEZ, M. (Ed.). **Advances in Soft Computing and Its Applications**. [S.l.]: Springer Berlin Heidelberg, 2013, (Lecture Notes in Computer Science, v. 8266). p. 233–244. Citation on page 63.

BRAZDIL, P.; GAMA, J.; HENERY, B. Characterizing the applicability of classification algorithms using meta-level learning. In: **Proceedings of the European conference on machine learning on Machine Learning**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1994. p. 83–102. ISBN 3-540-57868-4. Citation on page 41.

BRAZDIL, P.; GIRAUD-CARRIER, C.; SOARES, C.; VILALTA, R. **Metalearning: Applications to Data Mining**. 2. ed. [S.l.]: Springer Verlag, 2009. Citations on pages 25, 39, 40, and 51.

BRAZDIL, P. B.; HENERY, R. J. Analysis of results. In: MICHIE, D.; SPIEGELHALTER, D. J.; TAYLOR, C. C.; CAMPBELL, J. (Ed.). **Machine learning, neural and statistical classification**. [S.l.]: Ellis Horwood, 1994. chap. 10, p. 175–212. ISBN 0-13-106360-X. Citation on page 41.

BREIMAN, L. Random forests. **Machine Learning**, Kluwer Academic Publishers, Hingham, MA, USA, v. 45, n. 1, p. 5–32, Oct. 2001. Citation on page 105.

BREIMAN, L.; FRIEDMAN, J.; OLSHEN, R.; STONE, C. **Classification and Regression Trees**. [S.l.]: Chapman & Hall (Wadsworth, Inc.), 1984. Citations on pages 56 and 76.

BROCHU, E.; CORA, V. M.; FREITAS, N. de. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. **CoRR**, abs/1012.2599, 2010. Citation on page 34.

BRODERSEN, K. H.; ONG, C. S.; STEPHAN, K. E.; BUHMANN, J. M. The balanced accuracy and its posterior distribution. In: **Proceedings of the 2010 20th International Conference on Pattern Recognition**. [S.l.]: IEEE Computer Society, 2010. p. 3121–3124. ISBN 978-0-7695-4109-9. Citation on page 59.

CÁCERES, L. P.; LÓPEZ-IBÁÑEZ, M.; STÜTZLE, T. An analysis of parameters of irace. In: ____. **Evolutionary Computation in Combinatorial Optimisation: 14th European Conference, EvoCOP 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. p. 37–48. ISBN 978-3-662-44320-0. Citation on page 130.

CASALICCHIO, G.; BOSSEK, J.; LANG, M.; KIRCHHOFF, D.; KERSCHKE, P.; HOFNER, B.; SEIBOLD, H.; VANSCHOREN, J.; BISCHL, B. OpenML: An R package to connect to the machine learning platform OpenML. **Computational Statistics**, p. 1–15, 2017. Available: <http://dx.doi.org/10.1007/s00180-017-0742-2>. Citations on pages 58, 60, and 127.

CAWLEY, G. C.; TALBOT, N. L. C. On over-fitting in model selection and subsequent selection bias in performance evaluation. **The Journal of Machine Learning Research**, v. 11, p. 2079–2107, 2010. Available: <http://www.jmlr.org/papers/v11/cawley10a.html>. Citations on pages 57 and 62.

CHANG, C.-C.; LIN, C.-J. LIBSVM: A library for support vector machines. **ACM Transactions on Intelligent Systems and Technology**, v. 2, p. 27:1–27:27, 2011. Citation on page 63.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. SMOTE: synthetic minority over-sampling technique. **J. Artif. Int. Res.**, AI Access Foundation, v. 16, n. 1, p. 321–357, 2002. ISSN 1076-9757. Citations on pages 101 and 103.

CLERC, M. Standard partcile swarm optimization. 15 pages. 2012. Citation on page 61.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **The Journal of Machine Learning Research**, v. 7, p. 1–30, 2006. Citations on pages 59 and 123.

DUARTE, E.; WAINER, J. Empirical comparison of cross-validation and internal metrics for tuning svm hyperparameters. **Pattern Recogn. Lett.**, Elsevier Science Inc., New York, NY, USA, v. 88, n. C, p. 6–11, Mar. 2017. ISSN 0167-8655. Available: <https://doi.org/10.1016/j.patrec.2017.01.007>. Citation on page 63.

EGGENSPERGER, K.; LINDAUER, M.; HOOS, H. H.; HUTTER, F.; LEYTON-BROWN, K. Efficient benchmarking of algorithm configurators via model-based surrogates. **Machine Learning**, v. 107, n. 1, p. 15–41, 2018. Citations on pages 45, 48, and 125.

FEURER, M.; KLEIN, A.; EGGENSPERGER, K.; SPRINGENBERG, J.; BLUM, M.; HUTTER, F. Efficient and robust automated machine learning. In: CORTES, C.; LAWRENCE, N. D.; LEE, D. D.; SUGIYAMA, M.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 28**. [S.l.]: Curran Associates, Inc., 2015. p. 2944–2952. Citations on pages 25, 26, 42, 44, 46, 47, 51, 52, 57, and 125.

FEURER, M.; SPRINGENBERG, J. T.; HUTTER, F. Initializing bayesian hyperparameter optimization via meta-learning. In: **Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence**. AAAI Press, 2015. (AAAI'15), p. 1128–1135. ISBN 0-262-51129-0. Available: <http://dl.acm.org/citation.cfm?id=2887007.2887164>. Citations on pages 26, 41, 42, 44, 47, 51, 52, and 57.

FRIEDRICHS, F.; IGEL, C. Evolutionary tuning of multiple svm parameters. **Neurocomputing**, Elsevier Science Publishers B. V., v. 64, p. 107–117, 2005. Citation on page 35.

GARCIA, L. P.; de Carvalho, A. C.; LORENA, A. C. Effect of label noise in the complexity of classification problems. **Neurocomputing**, v. 160, p. 108–119, 2015. ISSN 0925-2312. Citations on pages 42, 152, and 153.

GARCIA, L. P. F.; CARVALHO, A. C. P. L. F. de; LORENA, A. C. Noise detection in the meta-learning level. **Neurocomputing**, v. 176, p. 14–25, 2016. Available: <https://doi.org/10.1016/j.neucom.2014.12.100>. Citation on page 42.

GASCóN-MORENO, J.; SALCEDO-SANZ, S.; ORTIZ-GARCíA, E. G.; CARRO-CALVO, L.; SAAVEDRA-MORENO, B.; PORTILLA-FIGUERAS, J. A. A binary-encoded tabu-list genetic algorithm for fast support vector regression hyper-parameters tuning. In: **International Conference on Intelligent Systems Design and Applications**. [S.l.: s.n.], 2011. p. 1253–1257. Citation on page 67.

GOMES, T. A. F.; PRUDÊNCIO, R. B. C.; SOARES, C.; ROSSI, A. L. D.; CARVALHO nd A. C. P. L. F. Combining meta-learning and search techniques to select parameters for support vector machines. **Neurocomputing**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 75, n. 1, p. 3–13, Jan. 2012. Citations on pages 26, 35, 41, 44, 46, 51, 63, and 125.

GONZALEZ-FERNANDEZ, Y.; SOTO, M. copulaedas: An R package for estimation of distribution algorithms based on copulas. **Journal of Statistical Software**, v. 58, n. 9, p. 1–34, 2014. Available: <http://www.jstatsoft.org/v58/i09/>. Citation on page 61.

GOOGLE. **Google Cloud AutoML**. 2018. <https://cloud.google.com/automl/>. Accessed: 2018-01-20. Citation on page 26.

GUNASEKARA, N.; PANG, S.; KASABOV, N. Tuning N-gram string kernel SVMs via meta learning. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [S.l.: s.n.], 2010. v. 6444 LNCS, p. 91–98. Citations on pages 45, 48, and 51.

HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I. H. The weka data mining software: An update. **SIGKDD Explor. Newsl.**, ACM, New York, NY, USA, v. 11, n. 1, p. 10–18, Nov. 2009. Citation on page 60.

HALL, M. A. **Correlation-based Feature Subset Selection for Machine Learning**. Phd Thesis (PhD Thesis) — University of Waikato, Hamilton, New Zealand, 1998. Citation on page 97.

HAUSCHILD, M.; PELIKAN, M. An introduction and survey of estimation of distribution algorithms. **Swarm and Evolutionary Computation**, v. 1, n. 3, p. 111 – 128, 2011. Citation on page 35.

HO, T. K.; BASU, M. Complexity measures of supervised classification problems. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, v. 24, n. 3, p. 289–300, 2002. Citation on page 42.

HORN, D.; DEMIRCIOĞLU, A.; BISCHL, B.; GLASMACHERS, T.; WEIHS, C. A comparative study on large scale kernelized support vector machines. **Advances in Data Analysis and Classification**, p. 1–17, 2016. ISSN 1862-5355. Available: <http://dx.doi.org/10.1007/s11634-016-0265-7>. Citations on pages 63, 73, and 127.

HORNIK, K.; BUCHTA, C.; ZEILEIS, A. Open-source machine learning: R meets Weka. **Computational Statistics**, v. 24, n. 2, p. 225–232, 2009. Citation on page 61.

HOTHORN, T.; HORNIK, K.; ZEILEIS, A. Unbiased recursive partitioning: A conditional inference framework. **Journal of Computational and Graphical Statistics**, v. 15, n. 3, p. 651–674, 2006. Citations on pages 56, 61, and 76.

HSU, C.-W.; CHANG, C.-C.; LIN, C.-J. **A Practical Guide to Support Vector Classification**. Taipei, Taiwan, 2007. Citation on page 63.

HUTTER, F.; HOOS, H.; LEYTON-BROWN, K. An efficient approach for assessing hyperparameter importance. In: **Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014**. [s.n.], 2014. p. 754–762. Available: <http://jmlr.org/proceedings/papers/v32/hutter14.html>. Citations on pages 89 and 128.

HUTTER, F.; HOOS, H.; LEYTON-BROWN, K.; STÜTZLE, T. Paramils: an automatic algorithm configuration framework. **Journal of Artificial Intelligence Research**, n. 36, p. 267–306, 2009. Citation on page 48.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. **An Introduction to Statistical Learning: With Applications in R**. [S.l.]: Springer Publishing Company, Incorporated, 2014. ISBN 1461471370, 9781461471370. Citation on page 101.

JANKOWSKI, D.; JACKOWSKI, K. Evolutionary algorithm for decision tree induction. In: SAEED, K.; SNášEL, V. (Ed.). **Computer Information Systems and Industrial Management**. [S.l.]: Springer Berlin Heidelberg, 2014, (Lecture Notes in Computer Science, v. 8838). p. 23–32. Citation on page 76.

KALOS, A. Automated neural network structure determination via discrete particle swarm optimization (for non-linear time series models). In: **Proceedings of the 5th WSEAS International Conference on Simulation, Modelling and Optimization**. [S.l.]: World Scientific and Engineering Academy and Society (WSEAS), 2005. (SMO'05), p. 325–331. Citation on page 35.

KANDA, J.; CARVALHO, A. C. P. L. F.; HRUSCHKA, E.; SOARES, C. Selection of algorithms to solve traveling salesman problems using meta-learning. **Int. J. Hybrid Intell. Syst.**, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 8, n. 3, Aug. 2011. Citation on page 39.

KASS, G. V. An exploratory technique for investigating large quantities of categorical data applied statistics. **Applied Statistics**, v. 30, n. 2, p. 119–127, 1980. Citations on pages 56 and 76.

KOCH, P.; BISCHL, B.; FLASCH, O.; BARTZ-BEIELSTEIN, T.; WEIHS, C.; KONEN, W. Tuning and evolution of support vector kernels. **Evolutionary Intelligence**, Springer-Verlag, v. 5, n. 3, p. 153–170, 2012. Citation on page 63.

KOHAVI, R. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In: **Second International Conference on Knowledge Discovery and Data Mining**. [S.l.: s.n.], 1996. p. 202–207. Citation on page 56.

KOTTHOFF, L.; THORNTON, C.; HOOS, H. H.; HUTTER, F.; LEYTON-BROWN, K. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. **Journal of Machine Learning Research**, v. 17, p. 1–5, 2016. Citations on pages 26, 57, and 125.

KRSTAJIC, D.; BUTUROVIC, L. J.; LEAHY, D. E.; THOMAS, S. Cross-validation pitfalls when selecting and assessing regression and classification models. **Journal of cheminformatics**, v. 6, n. 1, p. 10+, 2014. Available: <http://dx.doi.org/10.1186/1758-2946-6-10>. Citations on pages 57, 58, and 62.

LANDWEHR, N.; HALL, M.; FRANK, E. Logistic model trees. **Machine Learning**, v. 95, n. 1-2, p. 161–205, 2005. Citation on page 56.

LANG, M.; KOTTHAUS, H.; MARWEDEL, P.; WEIHS, C.; RAHNENFÜHRER, J.; BISCHL, B. Automatic model selection for high-dimensional survival analysis. **Journal of Statistical Computation and Simulation**, v. 85, n. 1, p. 62–76, 2015. Citation on page 36.

LEITE, R.; BRAZDIL, P.; VANSCHOREN, J. Selecting classification algorithms with active testing. In: **Proceedings of the 2012 Conference on Machine Learning and Data Mining (MLDM 2012)**. [S.l.: s.n.], 2012. p. 117–131. Citations on pages 42, 43, and 107.

LEMKE, C.; BUDKA, M.; GABRYS, B. Metalearning: a survey of trends and technologies. **Artificial Intelligence Review**, v. 44, n. 1, p. 117–130, Jun 2015. ISSN 1573-7462. Citation on page 39.

LIAW, A.; WIENER, M. Classification and regression by randomforest. **R News**, v. 2, n. 3, p. 18–22, 2002. Citation on page 61.

LIN, S.-W.; YING, K.-C.; CHEN, S.-C.; LEE, Z.-J. Particle swarm optimization for parameter determination and feature selection of support vector machines. **Expert Systems with Applications**, v. 35, n. 4, p. 1817 – 1824, 2008. Citation on page 67.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; CÁCERES, L. P.; BIRATTARI, M.; STÜETZLE, T. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, v. 3, p. 43 – 58, 2016. ISSN 2214-7160. Available: <http://www.sciencedirect.com/science/article/pii/S2214716015300270>. Citations on pages 60, 61, and 125.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; STÜTZLE, T.; BIRATTARI, M. **The irace package, Iterated Race for Automatic Algorithm Configuration**. [S.l.], 2011. Available: <http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf>. Citation on page 33.

LORENA, A. C.; MACIEL, A. I.; MIRANDA, P. B. C. de; COSTA, I. G.; PRUDÊNCIO, R. B. C. Data complexity meta-features for regression problems. **Machine Learning**, v. 107, n. 1, p. 209–246, 2018. Citations on pages 44, 46, 51, 73, and 127.

LóPEZ-IBáñEZ, M.; DUBOIS-LACOSTE, J.; CáCERES, L. P.; BIRATTARI, M.; STüTZLE, T. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, v. 3, p. 43 – 58, 2016. ISSN 2214-7160. Citation on page 48.

MANTOVANI, R. G.; HORVÁTH, T.; CERRI, R.; VANSCHOREN, J.; CARVALHO, A. C. P. L. F. de. Hyper-parameter tuning of a decision tree induction algorithm. In: **5th Brazilian Conference on Intelligent Systems, BRACIS 2016, Recife, Brazil, October 9-12, 2016**. IEEE Computer Society, 2016. p. 37–42. ISBN 978-1-5090-3566-3. Available: <http://dx.doi.org/10.1109/BRACIS.2016.018>. Citations on pages 75 and 128.

____. Tuning trees: on hyperparameter optimization for decision trees. **Soft Computing**, p. 1–54, 2018. Under review. Citations on pages 13, 57, 75, 80, 81, 83, 84, 86, 90, and 128.

MANTOVANI, R. G.; ROSSI, A. L. D.; ALCOBACA, E.; VANSCHOREN, J.; CARVALHO, A. C. P. L. F. A meta-learning recommender system for hyperparameter tuning: Suggesting when tuning improves svm classifiers. **Data Mining and Knowledge Discovery**, p. 1–35, 2018. Under review. Citations on pages 14, 95, 96, 102, 104, 105, 107, 108, 109, 110, 122, and 128.

MANTOVANI, R. G.; ROSSI, A. L. D.; VANSCHOREN, J.; BISCHL, B.; CARVALHO, A. C. P. L. F. de. Effectiveness of random search in SVM hyper-parameter tuning. In: **2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015**. IEEE, 2015. p. 1–8. ISBN 978-1-4799-1960-4. Available: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7256526>. Citations on pages 13, 63, 65, 66, 73, and 127.

MANTOVANI, R. G.; ROSSI, A. L. D.; VANSCHOREN, J.; CARVALHO, A. C. P. L. F. Meta-learning recommendation of default hyper-parameter values for svms in classification tasks. In: VANSCHOREN, J.; BRAZDIL, P.; GIRAUD-CARRIER, C. G.; KOTTHOFF, L. (Ed.). **Proceedings of the 2015 International Workshop on Meta-Learning and Algorithm Selection co-located with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2015 (ECMLPKDD 2015), Porto, Portugal, September 7th, 2015.** CEUR-WS.org, 2015. (CEUR Workshop Proceedings, v. 1455), p. 80–92. Available: <http://ceur-ws.org/Vol-1455/paper-09.pdf>. Citations on pages 63, 71, 73, 127, and 131.

MANTOVANI, R. G.; ROSSI, A. L. D.; VANSCHOREN, J.; BISCHL, B.; CARVALHO, A. C. P. L. F. To tune or not to tune: Recommending when to adjust SVM hyper-parameters via meta-learning. In: **2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015**. IEEE, 2015. p. 1–8. ISBN 978-1-4799-1960-4. Available: <http://dx.doi.org/10.1109/IJCNN.2015.7280644>. Citations on pages 41, 42, 95, 101, and 128.

MILLS, K. L.; FILLIBEN, J. J.; HAINES, A. L. Determining relative importance and effective settings for genetic algorithm control parameters. **Evol. Comput.**, MIT Press, Cambridge, MA, USA, v. 23, n. 2, p. 309–342, Jun. 2015. ISSN 1063-6560. Citations on pages 60 and 130.

MIRANDA, P.; PRUDÊNCIO, R. Active testing for SVM parameter selection. In: **Neural Networks (IJCNN), The 2013 International Joint Conference on**. [S.l.: s.n.], 2013. p. 1–8. Citations on pages 43, 44, and 51.

MIRANDA, P.; SILVA, R.; PRUDÊNCIO, R. Fine-tuning of support vector machine parameters using racing algorithms. In: **Proceedings of the 22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2014**. [S.l.: s.n.], 2014. p. 325–330. Citations on pages 36, 44, and 47.

MIRANDA, P. B. C.; PRUDÊNCIO, R. B. C.; CARVALHO, A. C. P. L. F.; SOARES, C. An experimental study of the combination of meta-learning with particle swarm algorithms for svm parameter selection. **Lecture Notes in Computer Science**, Springer Verlag, v. 7335 LNCS, n. PART 3, p. 562–575, 2012. Citations on pages 41, 42, 44, 46, 47, and 125.

MITCHELL, T. M. **Machine Learning**. New York: McGraw Hill, 1997. Citation on page 55.

MOLINA, M. M.; LUNA, J. M.; ROMERO, C.; VENTURA, S. Meta-learning approach for automatic parameter tuning: A case study with educational datasets. In: **Proceedings of the 5th International Conference on Educational Data Mining, EDM 2012**. [S.l.: s.n.], 2012. p. 180–183. Citations on pages 45, 47, 75, 123, 127, and 128.

MORAIS, G.; PRATI, R. C. Complex network measures for data set characterization. In: **Brazilian Conference on Intelligent Systems, BRACIS 2013, Fortaleza, CE, Brazil, 19-24 October, 2013**. IEEE Computer Society, 2013. p. 12–18. ISBN 978-0-7695-5092-3. Available: <https://doi.org/10.1109/BRACIS.2013.11>. Citation on page 42.

NOJIMA, Y.; NISHIKAWA, S.; ISHIBUCHI, H. A meta-fuzzy classifier for specifying appropriate fuzzy partitions by genetic fuzzy rule selection with data complexity measures. In: **2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)**. [S.l.]: IEEE, 2011. p. 264–271. Citations on pages 42, 43, 44, and 51.

ORRIOLS-PUIG, A.; MACIA, N.; HO, T. K. **Documentation for the data complexity library in C++**. Barcelona, Spain, 2010. Citation on page 42.

PADIERNA, L. C.; CARPIO, M.; ROJAS, A.; PUGA, H.; BALTAZAR, R.; FRAIRE, H. Hyperparameter tuning for support vector machines by estimation of distribution algorithms. In: ____. **Nature-Inspired Design of Hybrid Intelligent Systems**. Cham: Springer International Publishing, 2017. p. 787–800. ISBN 978-3-319-47054-2. Citations on pages 63, 67, 73, and 127.

PENG, Y.; FLACH, P.; BRAZDIL, P.; SOARES, C. Decision tree-based characterization for meta-learning. In: **Proceedings of the ECML - Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning**. [S.l.: s.n.], 2002. p. 111–122. Citation on page 41.

PFAHRINGER, B.; BENSUSAN, H.; GIRAUD-CARRIER, C. G. Meta-learning by landmarking various learning algorithms. In: **Proceedings of the Seventeenth International Conference on Machine Learning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. p. 743–750. ISBN 1-55860-707-2. Citation on page 41.

PINTO, F.; CERQUEIRA, V.; SOARES, C.; MENDES-MOREIRA, J. autobagging: Learning to rank bagging workflows with metalearning. **CoRR**, abs/1706.09367, 2017. Available: <http://arxiv.org/abs/1706.09367>. Citations on pages 44 and 46.

PINTO, F.; SOARES, C.; MENDES-MOREIRA, J. Towards automatic generation of metafeatures. In: BAILEY, J.; KHAN, L.; WASHIO, T.; DOBBIE, G.; HUANG, J. Z.; WANG, R. (Ed.). **Advances in Knowledge Discovery and Data Mining - 20th Pacific-Asia Conference, PAKDD 2016, Auckland, New Zealand, April 19-22, 2016, Proceedings, Part I**. Springer, 2016. (Lecture Notes in Computer Science, v. 9651), p. 215–226. ISBN 978-3-319-31752-6. Available: <https://doi.org/10.1007/978-3-319-31753-3_18>. Citations on pages 42 and 46.

PRIYA, R.; De Souza, B. F.; ROSSI, A. L. D.; CARVALHO, A. C. P. L. F. Using genetic algorithms to improve prediction of execution times of ML tasks. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [S.l.: s.n.], 2012. v. 7208 LNAI, n. PART 1, p. 196–207. Citations on pages 44, 46, 51, and 52.

QUINLAN, J. R. **C4.5: Programs for Machine Learning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN 1558602402. Citation on page 56.

R Core Team. **R: A Language and Environment for Statistical Computing**. Vienna, Austria, 2016. Available: <https://www.R-project.org/>. Citation on page 60.

RAPIDMINER. **RapidMiner Auto Model**. 2018. <https://rapidminer.com/>. Accessed: 2018-02-07. Citation on page 26.

REIF, M.; SHAFAIT, F.; DENGEL, A. Prediction of classifier training time including parameter optimization. In: BACH, J.; EDELKAMP, S. (Ed.). **KI 2011: Advances in Artificial Intelligence**. [S.l.]: Springer Berlin Heidelberg, 2011, (Lecture Notes in Computer Science, v. 7006). p. 260–271. Citations on pages 44, 46, 75, 76, and 127.

____. Meta-learning for evolutionary parameter optimization of classifiers. **Machine Learning**, Springer US, v. 87, p. 357–380, 2012. Citations on pages 41, 44, 46, 47, 51, 52, and 63.

REIF, M.; SHAFAIT, F.; GOLDSTEIN, M.; BREUEL, T.; DENGEL, A. Automatic classifier selection for non-experts. **Pattern Analysis and Applications**, v. 17, n. 1, p. 83–96, 2014. ISSN 1433-755X. Available: <http://dx.doi.org/10.1007/s10044-012-0280-z>. Citations on pages 26, 39, 41, 42, 45, 48, 51, 52, 75, 76, and 127.

RIDD, P.; GIRAUD-CARRIER, C. Using metalearning to predict when parameter optimization is likely to improve classification accuracy. In: VANSCHOREN, J.; BRAZDIL, P.; SOARES, C.; KOTTHOFF, L. (Ed.). **Meta-learning and Algorithm Selection Workshop at ECAI 2014**. [S.l.: s.n.], 2014. p. 18–23. Citations on pages 27, 45, 47, 50, 51, 52, 63, 123, and 128.

RIJN, J. N. van; HUTTER, F. An empirical study of hyperparameter importance across datasets. In: BRAZDIL, P.; VANSCHOREN, J.; HUTTER, F.; HOOS, H. (Ed.). **Proceedings of the International Workshop on Automatic Selection, Configuration and Composition of Machine Learning Algorithms co-located with the European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases, AutoML@PKDD/ECML 2017, Skopje, Macedonia, September 22, 2017.** CEUR-WS.org, 2017. (CEUR Workshop Proceedings, v. 1998), p. 91–98. Available: <http://ceur-ws.org/Vol-1998/paper_09.pdf>. Citations on pages 94 and 128.

ROKACH, L.; MAIMON, O. **Data Mining With Decision Trees: Theory and Applications**. 2nd. ed. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2014. ISBN 9789814590075, 981459007X. Citations on pages 56, 75, 76, and 127.

ROSSUM, G. **Python Reference Manual**. Amsterdam, The Netherlands, The Netherlands, 1995. Citation on page 60.

SANTAFE, G.; INZA, I.; LOZANO, J. A. Dealing with the evaluation of supervised classification algorithms. **Artificial Intelligence Review**, v. 44, n. 4, p. 467–508, Dec 2015. ISSN 1573-7462. Citation on page 59.

SARDá-ESPINOSA, A.; SUBBIAH, S.; BARTZ-BEIELSTEIN, T. Conditional inference trees for knowledge extraction from motor health condition data. **Engineering Applications of Artificial Intelligence**, v. 62, p. 26 – 37, 2017. ISSN 0952-1976. Citation on page 93.

SCRUCCA, L. Ga: A package for genetic algorithms in r. **Journal of Statistical Software**, v. 53, n. 1, p. 1–37, 2013. ISSN 1548-7660. Available: <https://www.jstatsoft.org/index.php/jss/article/view/v053i04>. Citation on page 60.

SIMON, D. **Evolutionary Optimization Algorithms**. first. [S.l.]: Wiley, 2013. Citation on page 35.

SMITH-MILES, K. A. Cross-disciplinary perspectives on meta-learning for algorithm selection. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 41, n. 1, p. 6:1–6:25, Jan. 2009. ISSN 0360-0300. Citation on page 39.

SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. In: PEREIRA, F.; BURGES, C.; BOTTOU, L.; WEINBERGER, K. (Ed.). **Advances in Neural Information Processing Systems 25**. [S.l.]: Curran Associates, Inc., 2012. p. 2951–2959. Citations on pages 25, 33, 34, and 78.

SOARES, C.; BRAZDIL, P. B. Selecting parameters of svm using meta-learning and kernel matrix-based meta-features. In: **Proceedings of the 2006 ACM symposium on Applied computing**. [S.l.]: ACM Press, 2006. (SAC'06), p. 564–568. Citations on pages 43, 44, and 51.

SOARES, C.; BRAZDIL, P. B.; KUBA, P. A meta-learning method to select the kernel width in support vector regression. **Machine Learning**, v. 54, n. 3, p. 195–209, 2004. Citations on pages 40, 41, 43, and 44.

SUN, Q.; PFAHRINGER, B. Pairwise meta-rules for better meta-learning-based algorithm ranking. **Machine Learning**, v. 93, n. 1, p. 141–161, 2013. ISSN 1573-0565. Available: <http://dx.doi.org/10.1007/s10994-013-5387-y>. Citations on pages 26, 45, 47, 50, 51, and 52.

THERNEAU, T.; ATKINSON, B.; RIPLEY, B. **rpart: Recursive Partitioning and Regression Trees**. [S.l.], 2015. R package version 4.1-10. Available: <https://CRAN.R-project.org/package=rpart>. Citation on page 61.

THORNTON, C.; HUTTER, F.; HOOS, H. H.; LEYTON-BROWN, K. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In: **Proc. of KDD-2013**. [S.l.: s.n.], 2013. p. 847–855. Citations on pages 26 and 57.

VANSCHOREN, J. **Understanding Machine Learning Performance with Experiment Databases (Het verwerven van inzichten in leerperformantie met experiment databanken)**. Phd Thesis (PhD Thesis) — Katholieke Universiteit Leuven, Belgium, 2010. Available: <https://lirias.kuleuven.be/handle/123456789/266060>. Citation on page 41.

VANSCHOREN, J.; RIJN, J. N. van; BISCHL, B.; TORGO, L. OpenML: Networked science in machine learning. **SIGKDD Explorations**, ACM, New York, NY, USA, v. 15, n. 2, p. 49–60, 2013. Citation on page 127.

____. Openml: Networked science in machine learning. **SIGKDD Explor. Newsl.**, ACM, v. 15, n. 2, p. 49–60, 2014. ISSN 1931-0145. Citations on pages 28, 29, 58, 59, and 60.

VAPNIK, V. **The Nature of Statistical Learning Theory**. [S.l.]: Springer-Verlag, 1995. Citation on page 55.

VILALTA, R.; DRISSI, Y. A perspective view and survey of meta-learning. **Artif. Intell. Rev.**, Kluwer Academic Publishers, Norwell, MA, USA, v. 18, n. 2, p. 77–95, Oct. 2002. ISSN 0269-2821. Citation on page 39.

VILALTA, R.; GIRAUD-CARRIER, C. G.; BRAZDIL, P.; SOARES, C. Using meta-learning to support data mining. **International Journal of Computer Science & Applications**, v. 1, n. 1, p. 31–45, 2004. Citation on page 41.

WAINBERG, M.; ALIPANAHI, B.; FREY, B. J. Are random forests truly the best classifiers? **Journal of Machine Learning Research**, v. 17, n. 110, p. 1–5, 2016. Available: <http://jmlr.org/papers/v17/15-374.html>. Citations on pages 56 and 93.

WISTUBA, M.; SCHILLING, N.; SCHMIDT-THIEME, L. Sequential model-free hyperparameter tuning. In: AGGARWAL, C.; ZHOU, Z.; TUZHILIN, A.; XIONG, H.; WU, X. (Ed.). **2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015**. IEEE Computer Society, 2015. p. 1033–1038. ISBN 978-1-4673-9504-5. Available: <https://doi.org/10.1109/ICDM.2015.20>. Citations on pages 45 and 48.

____. Two-stage transfer surrogate model for automatic hyperparameter optimization. In: ____. **Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I**. [S.l.]: Springer International Publishing, 2016. p. 199–214. ISBN 978-3-319-46128-1. Citations on pages 45 and 48.

____. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. **Machine Learning**, v. 107, n. 1, p. 43–78, 2018. Citations on pages 45, 48, and 125.

WITTEN, I. H.; FRANK, E. **Data Mining: Practical Machine Learning Tools and Techniques**. 2nd. ed. San Francisco: Morgan Kaufmann, 2005. Citations on pages 56 and 76.

WU, X.; KUMAR, V. **The Top Ten Algorithms in Data Mining**. 1st. ed. [S.l.]: Chapman & Hall/CRC, 2009. ISBN 1420089641, 9781420089646. Citations on pages 56 and 76.

ZAMBRANO-BIGIARINI, M.; CLERC, M.; ROJAS, R. Standard particle swarm optimisation 2011 at CEC-2013: A baseline for future PSO improvements. In: **Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013**. IEEE, 2013. p. 2337–2344. ISBN 978-1-4799-0452-5. Available: <https://doi.org/10.1109/CEC.2013.6557848>. Citation on page 60.

ZHONGGUO, Y.; HONGQI, L.; ALI, S.; YILE, A. Choosing classification algorithms and its optimum parameters based on data set characteristics. **Journal of Computers**, v. 28, n. 5, p. 26–38, 2017. Citations on pages 44, 46, and 51.

APPENDIX

A

LIST OF DATASETS

This appendix presents the full table of datasets used in experiments performed for this thesis: Tables 18 to 20. For each dataset it is shown: the OpenML dataset name and id, the number of attributes (D), the number of examples (N), the number of classes (C), the number of examples belonging to the majority and minority classes (nMaj, nMin), and the proportion between them (P).

Table 18 – OpenML datasets (1 to 45) used in experiments. For each dataset it is shown: the OpenML dataset name and id, the number of attributes (D), the number of examples (N), the number of classes (C), the number of examples belonging to the majority and minority classes (nMaj, nMin), and the proportion between them (P).

Nro	OpenML name	OpenML did	D	N	C	nMaj	nMin	P
1	kr-vs-kp	3	37	3196	2	1669	1527	0.91
2	balance-scale	11	5	625	3	288	49	0.17
3	breast-cancer	13	10	286	2	201	85	0.42
4	mfeat-fourier	14	77	2000	10	200	200	1.00
5	breast-w	15	10	699	2	458	241	0.53
6	mfeat-karhunen	16	65	2000	10	200	200	1.00
7	mfeat-morphological	18	7	2000	10	200	200	1.00
8	car	21	7	1728	4	1210	65	0.05
9	mfeat-zernike	22	48	2000	10	200	200	1.00
10	colic	25	28	368	2	232	136	0.59
11	optdigits	28	65	5620	10	572	554	0.97
12	credit-g	31	21	1000	2	700	300	0.43
13	pendigits	32	17	10992	10	1144	1055	0.92
14	dermatology	35	35	366	6	112	20	0.18
15	segment	36	20	2310	7	330	330	1.00
16	diabetes	37	9	768	2	500	268	0.54
17	sick	38	30	3772	2	3541	231	0.07
18	sonar	40	61	208	2	111	97	0.87
19	haberman	43	4	306	2	225	81	0.36
20	spambase	44	58	4601	2	2788	1813	0.65
21	tae	48	6	151	3	52	49	0.94
22	heart-c	49	14	303	5	165	0	0.00
23	tic-tac-toe	50	10	958	2	626	332	0.53
24	heart-statlog	53	14	270	2	150	120	0.80
25	vehicle	54	19	846	4	218	199	0.91
26	hepatitis	55	20	155	2	123	32	0.26
27	vote	56	17	435	2	267	168	0.63
28	ionosphere	59	35	351	2	225	126	0.56
29	waveform-5000	60	41	5000	3	1692	1653	0.98
30	iris	61	5	150	3	50	50	1.00
31	molecular-biology_promoters	164	59	106	2	53	53	1.00
32	satimage	182	37	6430	6	1531	625	0.41
33	baseball	185	18	1340	3	1215	57	0.05
34	wine	187	14	178	3	71	48	0.68
35	eucalyptus	188	20	736	5	214	105	0.49
36	Australian	292	15	690	2	383	307	0.80
37	satellite_image	294	37	6435	6	871	275	0.32
38	libras_move	299	91	360	11	24	11	0.46
39	vowel	307	13	990	11	90	90	1.00
40	mammography	310	7	11183	2	10923	260	0.02
41	oil_spill	311	50	937	2	896	41	0.05
42	yeast_ml8	316	117	2417	2	2383	34	0.01
43	hayes-roth	329	5	160	4	65	0	0.01
44	monks-problems-1	333	7	556	2	278	278	1.00
45	monks-problems-2	334	7	601	2	395	206	0.52

Table 19 – OpenML datasets (46 to 95) used in experiments. For each dataset it is shown: the OpenML dataset name and id, the number of attributes (D), the number of examples (N), the number of classes (C), the number of examples belonging to the majority and minority classes (nMaj, nMin), and the proportion between them (P).

Nro	OpenML name	OpenML did	D	N	C	nMaj	nMin	P
46	monks-problems-3	335	7	554	2	288	266	0.92
47	SPECT	336	23	267	2	212	55	0.26
48	grub-damage	338	9	155	4	49	19	0.39
49	synthetic_control	377	62	600	6	100	100	1.00
50	analcatdata_boxing2	444	4	132	2	71	61	0.86
51	analcatdata_boxing1	448	4	120	2	78	42	0.54
52	analcatdata_lawsuit	450	5	264	2	245	19	0.08
53	irish	451	6	500	2	278	222	0.80
54	analcatdata_broadwaymult	452	8	285	7	118	21	0.18
55	cars	455	9	406	3	254	73	0.29
56	analcatdata_authorship	458	71	841	4	317	55	0.17
57	analcatdata_creditscore	461	7	100	2	73	27	0.37
58	backache	463	33	180	2	155	25	0.16
59	prnn_synth	464	3	250	2	125	125	1.00
60	schizo	466	15	340	2	177	163	0.92
61	analcatdata_dmft	469	5	797	6	155	123	0.79
62	profb	470	10	672	2	448	224	0.50
63	analcatdata_germangss	475	6	400	4	100	100	1.00
64	biomed	481	9	209	2	134	75	0.56
65	rmftsa_sleepdata	679	3	1024	4	404	94	0.23
66	visualizing_livestock	685	3	130	5	26	26	1.00
67	diggle_table_a2	694	9	310	9	41	18	0.44
68	ada_prior	1037	15	4562	2	3430	1132	0.33
69	ada_agnostic	1043	49	4562	2	3430	1132	0.33
70	jEdit_4.2_4.3	1048	9	369	2	204	165	0.81
71	pc4	1049	38	1458	2	1280	178	0.14
72	pc3	1050	38	1563	2	1403	160	0.11
73	mc2	1054	40	161	2	109	52	0.48
74	mc1	1056	39	9466	2	9398	68	0.01
75	ar4	1061	30	107	2	87	20	0.23
76	kc2	1063	22	522	2	415	107	0.26
77	ar6	1064	30	101	2	86	15	0.17
78	kc3	1065	40	458	2	415	43	0.10
79	kc1-binary	1066	95	145	2	85	60	0.71
80	kc1	1067	22	2109	2	1783	326	0.18
81	pc1	1068	22	1109	2	1032	77	0.07
82	pc2	1069	37	5589	2	5566	23	0.00
83	mw1	1071	38	403	2	372	31	0.08
84	jEdit_4.0_4.2	1073	9	274	2	140	134	0.96
85	datatrieve	1075	9	130	2	119	11	0.09
86	PopularKids	1100	11	478	3	247	90	0.36
87	teachingAssistant	1115	7	151	3	52	49	0.94
88	musk	1116	170	6598	2	5581	1017	0.18
89	badges2	1121	12	294	2	210	84	0.40
90	pc1_req	1167	9	320	2	213	107	0.50
91	MegaWatt1	1442	38	253	2	226	27	0.12
92	PizzaCutter1	1443	38	661	2	609	52	0.09
93	PizzaCutter3	1444	38	1043	2	916	127	0.14
94	CostaMadre1	1446	38	296	2	258	38	0.15
95	CastMetal1	1447	38	327	2	285	42	0.15

Table 20 – OpenML datasets (96 to 140) used in experiments. For each dataset it is shown: the OpenML dataset name and id, the number of attributes (D), the number of examples (N), the number of classes (C), the number of examples belonging to the majority and minority classes (nMaj, nMin), and the proportion between them (P).

Nro	OpenML name	OpenML did	D	N	C	nMaj	nMin	P
96	PieChart1	1451	38	705	2	644	61	0.09
97	PieChart2	1452	37	745	2	729	16	0.02
98	PieChart3	1453	38	1077	2	943	134	0.14
99	acute-inflammations	1455	7	120	2	70	50	0.71
100	appendicitis	1456	8	106	2	85	21	0.25
101	artificial-characters	1459	8	10218	10	1416	600	0.42
102	banknote-authentication	1462	5	1372	2	762	610	0.80
103	blogger	1463	6	100	2	68	32	0.47
104	blood-transfusion-service-center	1464	5	748	2	570	178	0.31
105	breast-tissue	1465	10	106	6	22	14	0.64
106	cardiotography	1466	35	2126	10	579	53	0.091
107	cnae-9	1468	857	1080	9	120	120	1.00
108	fertility	1473	10	100	2	88	12	0.14
109	first-order-theorem-proving	1475	52	6118	6	2554	486	0.19
110	hill-valley	1479	101	1212	2	606	606	1.00
111	ilpd	1480	11	583	2	416	167	0.40
112	lsvt	1484	311	126	2	84	42	0.50
113	ozone-level-8hr	1487	73	2534	2	2374	160	0.07
114	parkinsons	1488	23	195	2	147	48	0.33
115	phoneme	1489	6	5404	2	3818	1586	0.42
116	one-hundred-plants-shape	1492	65	1600	100	16	16	1.00
117	qsar-biodeg	1494	42	1055	2	699	356	0.51
118	qualitative-bankruptcy	1495	7	250	2	143	107	0.75
119	ringnorm	1496	21	7400	2	3736	3664	0.98
120	wall-robot-navigation	1497	25	5456	4	2205	328	0.15
121	sa-heart	1498	10	462	2	302	160	0.53
122	steel-plates-fault	1504	34	1941	2	1268	673	0.53
123	thoracic-surgery	1506	17	470	2	400	70	0.18
124	twonorm	1507	21	7400	2	3703	3697	1.00
125	wdbc	1510	31	569	2	357	212	0.59
126	wholesale-customers	1511	9	440	2	298	142	0.48
127	heart-long-beach	1512	14	200	5	56	10	0.18
128	robot-failures-lp4	1519	91	117	3	72	21	0.29
129	robot-failures-lp5	1520	91	164	5	47	21	0.45
130	vertebra-column	1523	7	310	3	150	60	0.40
131	volcanoes-a2	1528	4	1623	5	1471	29	0.02
132	volcanoes-a3	1529	4	1521	5	1369	29	0.02
133	volcanoes-b1	1531	4	10176	5	9791	26	0.00
134	volcanoes-b3	1533	4	10386	5	10006	25	0.00
135	volcanoes-b5	1535	4	9989	5	9599	26	0.00
136	volcanoes-b6	1536	4	10130	5	9746	26	0.00
137	volcanoes-d2	1539	4	9172	5	8670	56	0.01
138	volcanoes-d3	1540	4	9285	5	8771	58	0.01
139	autoUniv-au1-1000	1547	21	1000	2	741	259	0.35
140	autoUniv-au6-750	1549	41	750	8	165	57	0.35

Table 21 – OpenML datasets (141 to 168) used in experiments. For each dataset it is shown: the OpenML dataset name and id, the number of attributes (D), the number of examples (N), the number of classes (C), the number of examples belonging to the majority and minority classes (nMaj, nMin), and the proportion between them (P).

Nro	OpenML name	OpenML did	D	N	C	nMaj	nMin	P
141	autoUniv-au6-400	1551	41	400	8	111	25	0.23
142	autoUniv-au7-1100	1552	13	1100	5	305	153	0.50
143	autoUniv-au7-500	1554	13	500	5	192	43	0.22
144	acute-inflammations	1556	7	120	2	61	59	0.97
145	bank-marketing	1558	17	4521	2	4000	521	0.13
146	breast-tissue	1559	10	106	4	49	14	0.29
147	hill-valley	1566	101	1212	2	612	600	0.98
148	wilt	1570	6	4839	2	4578	261	0.06
149	SPECTF	1600	45	267	2	212	55	0.26
150	PhishingWebsites	4534	31	11055	2	6157	4898	0.80
151	MiceProtein	4550	82	1080	8	150	105	0.70
152	cylinder-bands	6332	40	540	2	312	228	0.73
153	thyroid-allbp	40474	27	2800	5	1632	31	0.02
154	thyroid-allhyper	40475	27	2800	5	1632	31	0.02
155	LED-display-domain-7digit	40496	8	500	10	57	37	0.65
156	texture	40499	41	5500	11	500	500	1.00
157	cmc	23	10	1473	3	629	333	0.53
158	credit-a	29	16	690	2	383	307	0.80
159	page-blocks	30	11	5473	5	4913	28	0.01
160	heart-h	51	14	294	5	188	0	0.00
161	banana	1460	3	5300	2	2924	2376	0.81
162	planning-relax	1490	13	182	2	130	52	0.40
163	one-hundred-plants-margin	1491	65	1600	100	16	16	1.00
164	user-knowledge	1508	6	403	5	129	24	0.19
165	volcanoes-a1	1527	4	3252	5	2952	58	0.02
166	volcanoes-a4	1530	4	1515	5	1365	29	0.02
167	autoUniv-au7-700	1553	13	700	3	245	214	0.87
168	autoUniv-au6-1000	1555	41	1000	8	240	89	0.37

APPENDIX

B

SETS OF META-FEATURES

This appendix presents the full table of meta-features used in experiments performed for this thesis: Tables 22 and 23. For each meta-feature it is shown: the category it belongs, its acronym and description.

Table 22 – Meta-features used in experiments - part 1. For each meta-features it is shown: its type, acronym and description. Extended from Garcia, de Carvalho and Lorena (2015).

Type	Acronym	Description
Simple	classes	Number of classes
	attributes	Number of attributes
	numeric	Number of numerical attributes
	nominal	Number of nominal attributes
	samples	Number of examples
	dimension	samples/attributes
	numRate	numeric/attributes
	nomRate	nominal/attributes
	symbols (min, max, mean, sd, sum)	Distributions of categories in attributes
	classes (min, max, mean, sd)	Classes distributions
Statistical	sks	Skewness
	sksP	Skewness for normalized dataset
	kts	Kurtosis
	ktsP	Kurtosis for normalized datasets
	absC	Correlation between attributes
	canC	Canonical correlation between matrices
	frac	Fraction of canonical correlation
Information-theoretic	clEnt	Class entropy
	nClEnt	Class entropy for normalized dataset
	atrEnt	Mean entropy of attributes
	nAtrEnt	Mean entropy of attributes for normalized dataset
	jEnt	Joint entropy
	mutInf	Mutual information
	eqAtr	clEnt/mutInf
	noiSig	(atrEnt - mutInf)/MutInf
Model-based (Tree)	nodes	Number of nodes
	leaves	Number of leaves
	nodeAtr	Number of nodes per attribute
	nodeIns	Number of nodes per instance
	leafCor	leave/samples
	lev (min, max, mean, sd)	Distributions of levels of depth
	bran (min, max, mean, sd)	Distributions of levels of branches
	att (min, max, mean, sd)	Distributions of attributes used
Landmarking	nb	Naive Bayes accuracy
	stump (min, max, mean, sd)	Distribution of decision stumps
	stMinGain	Minimum gain ratio of decision stumps
	stRand	Random gain ratio of decision stumps
	nn	1-Nearest Neighbor accuracy

Table 23 – Meta-features used in experiments - part 2. For each meta-features it is shown: its type, acronym and description. Extended from Garcia, de Carvalho and Lorena (2015).

Type	Acronym	Description
Data Complexity	f1	Maximum Fisher's discriminant ratio
	f1v	Directional-vector maximum Fisher's discriminant ratio
	f2	Overlap of the per-class bounding boxes
	f3	Maximum feature efficiency
	f4	Collective feature efficiency
	l1	Minimized sum of the error distance of a linear classifier
	l2	Training error of a linear classifier
	l3	Nonlinearity of a linear classifier
	n1	Fraction of points on the class boundary
	n2	Ratio of average intra/inter-class NN distance
	n3	leave-one-out error rate of the 1-NN classifier
	n4	Nonlinearity of the 1-NN classifier
	t1	Fraction of maximum covering spheres
	t2	Average number of points per dimension
Complexity Network	edges	Number of edges
	degree	Average degree of the network
	density	Average density of the network
	maxComp	Maximum number of components
	closeness	Closeness centrality
	betweenness	Betwenness centrality
	clsCoef	Clustering Coefficient
	hubs	Hub score
	avgPath	Average path length
Relative Landmarking	diff.svm.lm	perf(SVM) - perf(Linear)
	diff.svm.nb	performance(SVM) - performance(NB)
	diff.svm.stump	performance(SVM) - performance(Decision Stump)
	diff.svm.nn	performance(SVM) - performance(1-NN)
	diff.nn.lm	performance(1-NN) - performance(Linear)
	diff.nn.stump	performance(1-NN) - performance(Decision Stump)
	diff.nn.nb	performance(1-NN) - performance(NB)
	diff.nb.stump	performance(NB) - performance(Decision Stump)
	diff.nb.lm	performance(NB) - performance(Linear)
	diff.stump.lm	performance(Decision Stump) - performance(Linear)

APPENDIX

C

META-LEARNERS' HP SPACE

This appendix presents the full table of the meta-learners' HPs tuned in the experiments performed for this thesis. For each algorithm used as meta-learner, the table shows: the selected HPs, their types and range of values, default settings and the R package used to implement the algorithm.

Table 24 – Meta-learners' HP spaces explored in the experiments. The nomenclature follows their respective R packages. The NB and LR classifiers do not have any hyperparameter for tuning.

Algo	Symbol	Hyperparameter	Range	Type	Default	Package
CART	cp	complexity parameter	$(0.0001, 0.1)$	real	0.01	`rpart`
CART	minsplit	minimum number of instances in a node for a split to be attempted	$[1, 50]$	integer	20	
CART	minbucket	minimum number of instances in a leaf	$[1, 50]$	integer	7	
CART	maxdepth	maximum depth of any node of the final tree	$[1, 30]$	integer	30	
GP	sigma	width of the Gaussian kernel	$[2^{-15}, 2^{15}]$	real	-	`kernlab`
SVM	k	kernel	Gaussian	-	-	`e1071`
SVM	C	regularized constant	$[2^{-15}, 2^{15}]$	real	1	
SVM	γ	width of the Gaussian kernel	$[2^{-15}, 2^{15}]$	real	$1/N$	
RF	ntree	number of trees	$[2^{0}, 2^{10}]$	integer	500	`randomForest`
RF	nodesize	minimum node size of the decision trees	$\{1, 20\}$	integer	1	
KNN	k	number of nearest neighbors	$\{1, 50\}$	integer	7	`kknn`
NB	-	-	-	-	-	`e1071`
LR	-	-	-	-	-	`gbm`

ICMC USP
SÃO CARLOS