A meta-learning recommender system for hyperparameter tuning: predicting when tuning improves SVM classifiers

Rafael G. Mantovani[a,d,*], André L. D. Rossi[c], Edesio Alcobaça[a],
Joaquin Vanschoren[b], André C. P. L. F. de Carvalho[a]

[a]*Institute of Mathematics and Computer Sciences, University of São Paulo, São Carlos - SP, Brazil*
[b]*Eindhoven University of Technology, Eindhoven, Netherlands*
[c]*Universidade Estadual Paulista, Campus de Itapeva, São Paulo, Brazil*
[d]*Federal Technology University, Campus of Apucarana - PR, Brazil*

Abstract

For many machine learning algorithms, predictive performance is critically affected by the hyperparameter values used to train them. However, tuning these hyperparameters can come at a high computational cost, especially on larger datasets, while the tuned settings do not always significantly outperform the default values. This paper proposes a recommender system based on meta-learning to identify exactly when it is better to use default values and when to tune hyperparameters for each new dataset. Besides, an in-depth analysis is performed to understand what they take into account for their decisions, providing useful insights. An extensive analysis of different categories of meta-features, meta-learners, and setups across 156 datasets is performed. Results show that it is possible to accurately predict when tuning will significantly improve the performance of the induced models. The proposed system reduces the time spent on optimization processes, without reducing the predictive performance of the induced models (when compared with the ones obtained using tuned hyperparameters). We also explain the decision-making process of the meta-learners in terms of linear separability-based hypotheses. Although this analysis is focused on the tuning of Support Vector Machines, it can also be applied to other algorithms, as shown in experiments performed with decision trees.

Keywords: Meta-learning, Recommender system, Tuning recommendation, Hyperparameter tuning, Support vector machines

1. Introduction

Many Machine Learning (ML) algorithms, among them Support Vector Machines (SVMs) [48], have been successfully used in a wide variety of problems. SVMs are kernel-based algorithms that perform non-

---

*Rafael Gomes Mantovani
Computer Engineering Department, Federal Technology University, Campus of Apucarana
R. Marcílio Dias, 635 - Jardim Paraíso, Apucarana - PR, Brazil, Postal Code 86812-460
    Email addresses: `rafaelmantovani@utfpr.edu.br` (Rafael G. Mantovani), `alrossi@itapeva.unesp.br` (André L. D. Rossi), `edesio@usp.br` (Edesio Alcobaça), `j.vanschoren@tue.nl` (Joaquin Vanschoren), `andre@icmc.usp.br` (André C. P. L. F. de Carvalho)

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. As most ML algorithms, SVMs are sensitive to their Hyperparameter (HP) values, which directly affect their predictive performance and depend on the data under analysis. The predictive performance of SVMs is mostly affected by the values of four HPs: the kernel function (k), its width (γ) or polynomial degree (d), and the regularized constant (C). Hence, finding suitable SVM HPs is a frequently studied problem [18, 34]. SVM HP tuning is commonly modeled as a black-box optimization problem whose objective function is associated with the predictive performance of the SVM induced model. Many optimization techniques have been proposed in the literature for this problem, varying from a simple Grid Search (GS) to the state of the art Sequential Model-based Optimization (SMBO) technique [43]. In [3], Bergstra & Bengio showed theoretically and empirically that Random Search (RS) is a better alternative than GS and is able to find good HP settings when performing HP tuning. Mantovani et. al. [27] also compared RS with meta-heuristics to tune SVM HPs. A large amount of empirical experiments showed that RS generates models with predictive performance as effective as those obtained by meta-heuristics.

However, regardless the optimization technique, hyperparameter tuning usually has a high computational cost, particularly for large datasets, with no guarantee that a model with high predictive performance will be obtained. During the tuning, a large number of HP settings usually need to be assessed before a "good" solution is found, requiring the induction of several models, multiplying the learning cost by the number of settings evaluated. Besides, several aspects, such as the complexity of a dataset, can influence the tuning cost.

When computational resources are limited, a commonly adopted alternative is to use the default HP values suggested by ML tools. Previous works have pointed out that for some datasets, HP tuning of SVMs is not necessary [41]. Using default values largely reduce the overall computational cost, but, depending on the dataset, 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.

In this paper, we propose a recommender system to predict, when applying SVMs to a new dataset, whether it is better to perform HP tuning or it is sufficient to use default HP values. This system, based on Meta-learning (MtL) [5], is able to reduce the overall cost of tuning without significant loss in predictive performance. Another important novelty in this study is a descriptive analysis of how the recommendation occurs. Although the recommender system is proposed for the HP tuning of SVMs, it can also be used for other ML algorithms. To illustrate this aspect, we present an example where the recommender system is used for HP tuning of a Decision Tree (DT) induction algorithm.

The proposed recommender system can also be categorized as an Automated Machine Learning (AutoML) solution [14], since it aims to relieve the user from the repetitive and time-consuming tuning task,

automating the process through MtL. The AutoML area is relatively new, and there still many questions to be addressed. This fact, and the emerging attention it has attracted from important research groups [14, 21] and large companies [1] [2], highlights the importance of new studies in this area. An essential aspect for the success of AutoML systems is to provide an automatic and robust tuning system, which also emphasizes the relevance of the problem investigated in this paper.

In summary, the main contributions of this study are:

- the development of a modular and extensible MtL framework to predict when default HP values provide accurate models, saving computational time that would be wasted on optimization with no significant improvement;

- a comparison of the effectiveness of different sets of meta-features and preprocessing methods for meta-learning, not previously investigated;

- reproducibility of the experiments and analyses: all the code and experimental results are available to reproduce experiments, analyses and allow further investigations[3].

It is important to mention that we considered the proposed framework for predictive tasks, in particular, supervised classification tasks using SVMs. However, the issues investigated in this paper can be easily extended to other tasks (such as regression) and other ML algorithms [4].

This paper is structured as follows: Section 2 presents the basic MtL concepts used in our approach. Section 3 defines the HP tuning problem and presents a concise survey of prior work combining SVMs with MtL in some way. The complete experimental methodology covered to obtain the results is presented in Section 4. Results are discussed in Section 5 while final considerations and conclusions are presented in Section 6.

2. Background on Meta-Learning

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 data set. When applying a ML algorithm to a dataset, a higher predictive performance can be obtained if an algorithm whose bias is more adequate to the dataset 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) [5].

---

[1] Google Cloud AutoML - `https://cloud.google.com/automl/`

[2] Microsoft Custom Vision - `https://www.customvision.ai`

[3] The code is available in `Github` repositories, while experimental results are available on OpenML [47] study pages. These links are provided in Table 8 at Subsection 4.7.

[4] A note on the generalization of the proposal is presented in Section 5.8

MtL has been largely used for algorithm selection [1], and for ranking [44] and prediction [40] of predictive performance of ML algorithms. It investigates how to learn from previous ML experiments. According to Brazdil et. al. [5], meta-learning can be used to improve the learning mechanism itself after each training process. In MtL, the process of using a learning algorithm to induce a model for a data set is called base-learning. At the meta-level, likely useful information extracted from this process (meta-features) are used to induce a meta-model. This meta-model can recommend the most promising learning algorithm, a set of the N best learning algorithms or a ranking of learning algorithms according to their estimate predictive performance for a new dataset. The knowledge extracted during this process is called meta-knowledge. The meta-features extracted from each dataset is a critical aspect. They must be sufficient to describe the main aspects necessary to distinguish the predictive performance obtained by different learning algorithms when applied to this dataset. As a result, it should allow the induction of a meta-model with good predictive performance. According to [49] three different sets of measures can be applied to extract meta-features:

 (i) *Simple*, *Statistical* and *Information-theoretic* meta-features [6]: these 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 [14, 16, 32, 39, 40, 45];

 (ii) *Model-based* meta-features [2]: these 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 [39, 40];

(iii) *Landmarking* [35]: the predictive performance obtained by models induced by simple learning algorithms, called landmarkers, are used to characterize a dataset. These measures were explored in studies such as [14, 40].

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

(iv) *Data complexity* [17]: this 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 [15]; and

 (v) *Complex networks* [33]: 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 [15].

3. Meta-learning for Hyperparameter 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 for data analysis tasks. These studies can be roughly

grouped into the following approaches, according to what MtL does:

- it recommends HP settings;

- it predicts training runtime;

- it recommends initial values for HP optimization;

- it estimates predictive performance for an HP setting;

- it predicts HP tuning improvement/necessity.

Table 1 summarizes a comprehensive list of studies that either embedded or used Meta-learning (MtL) to cope with the SVM HP tuning problem. Next, these works are described in more detail.

3.1. 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 [45]. In Soares et. al. [45] and Soares & Brazdil [44], 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 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 & Smith-Miles [1] presented a similar study but selected 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 & Prudêncio [30] proposed another MtL approach, called Active Testing (AT) [23], to select the HPs γ and the soft margin (C). Experiments performed on 60 classification datasets assessed the settings using a single 10-CV and the Acc measure.

Lorena et. al. [25] proposed a set of 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, considering 39 regression problems. The recommendation of HPs for new unseen datasets was performed by a kNN distance-based meta-learner.

3.2. Prediction of Training Runtime

Other works investigated the use of MtL to estimate the training time of classification algorithms when induced by different HP settings. In Reif et. al. [38], the authors predicted the training time for several classifiers, including SVMs. They defined a discrete grid of $\gamma \times C$ HP settings, assessing these settings on

123 classification datasets considering the Pearson Product-Moment Correlation Coefficient (PMCC) and the Normalized Absolute Error (NAE) performance measures. In Priya et. al [36], the authors conducted a similar study but used a Genetic Algorithm (GA) to optimize parameters and perform meta-feature selection of six meta-learners. Experiments were carried out over 78 classification datasets assessing HP settings using a 5-fold CV and the Mean Absolute Deviation (MAD) evaluation measure.

3.3. Recommendation of initial values for HP optimization

MtL has also been used to speed up the optimization of HP values for classification algorithms [14, 16, 32, 39]. In Gomes et. al. [16] MtL is used to recommend HP settings as initial search values by the Particle Swarm Optimization (PSO) and Tabu Search (TS) optimization techniques. Experiments were conducted in 40 regression datasets adjusting the C and γ HPs to reduce the NMSE value. A kNN meta-learner was used to recommend the initial search values.

Reif et. al. [39] and Miranda et. al. [32] investigated, respectively, the use of Genetic Algorithms (GAs) and different versions of PSOs for the same task. In Miranda et.al. [31], the authors used multi-objective optimization to optimize the HPs to increase predictive the performance and the number of support vectors. 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 Machine Learning (ML) algorithms, the `Auto-skLearn` [14]. In this tool, MtL is used to recommend HP settings for the initial population of the SMBO optimization technique. The authors explored all the available SVM HPs in 140 OpenML classification datasets. It is the first and perhaps the only work that uses nested-CVs to assess HP settings. Each setting was assessed in terms of the simple Acc measure.

3.4. Estimation of predictive performance for an HP setting

A more recent approach uses MtL to estimate ML algorithms' performance considering their HPs. In Reif et. al. [40], the authors evaluated different ML algorithms in 54 datasets, including SVMs, and used the performance predictions to develop a MtL system for automatic algorithm selection.

Wistuba et. al. [50] adapted the acquisition function of surrogate models by one optimized meta-model. They evaluated several SVM 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 authors also proposed a new Transfer Acquisition Function (TAF) that extended the original proposal by predicting the predictive performance of HP settings for surrogate models.

Eggensperger et. al. [13] proposed a benchmarking approach of "surrogate scenarios", which extracts meta-knowledge from HP optimization and algorithm configuration problems, and approximates the performance surface by regression models. One of the 11 meta-datasets explored in the experimental setup has a set of SVMs HP settings assessed for the MNIST dataset. These settings were obtained executing a

6

simple RS method and three optimizers: Random Online Adaptive Racing (ROAR) [20], Iterated F-race (Irace) [24], and Iterated Local Search in Parameter Configuration Space (ParamILS).

3.5. Prediction of HP tuning improvement/necessity

Although the studies mentioned in this section are the most related to our current work regarding the proposed modeling, they have different goals. While Ridd & Giraud-Carrier [41] and Sanders & Giraud-Carrier [42] are concerned with predicting tuning improvement, Mantovani et al. [28] and the present study aimed to predict when HP tuning is necessary.

Ridd & Giraud-Carrier [41] investigated a Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem. They carried out experiments using PSO technique to search the hyperspace of this CASH problem in 326 binary classification datasets. Their MtL-based method predicts whether HP tuning would lead to a considerable increase in accuracy considering a pool of algorithms, including SVM.Even though this is one of the first studies in this direction, we could point out some drawbacks:

- the proposed method does not identify which algorithm and correspondent HP values the user should run to achieve an improved performance;

- there is no guarantee that training and testing data are not mutually exclusive;

- the rule to label the meta-examples is defined empirically, based on thresholds of the difference of the accuracy between default and tuned HP values;

- all the datasets are binary classification problems; and

- it is not possible to reproduce the experiments, specially base-level tuning since most of the details are not explained, and the code is not available.

Sanders & Giraud-Carrier [42] used a GA technique for HP tuning of three different ML algorithms, including SVMs. Their experimental results with 229 OpenML classification datasets showed that tuning almost always yielded significant improvements compared to default HP values. Thus, they focused on the regression task of predicting how much improvement can be expected by tuning HP compared to default values. They also addressed this task using MtL. However, their study presents some limitations, such as:

- the optimization process of SVM hyperparameters were computationally costly and did not finish for most of the datasets;

- the meta-learner was not able to predict hyperparameter tuning improvements for SVM in those datasets whose tuning process finished;

Table 1: Summary of related studies applying to MtL to SVMs. Fields without information in the related study are marked with a hyphen.

Reference	Year	Meta-learning	k	γ	C	d	Tuning Techniques	Meta Learner	Number of Datasets	Datasets' Source	Evaluation Procedure	Evaluation Measure
Soares et al. [45]	2004	Recommends HP settings	•				GS	kNN	42	UCI, METAL	10-CV	NMSE
Soares & Brazdil [44]	2006	Recommends HP settings	•				GS	kNN	42	UCI, METAL	10-CV	NMSE
Ali & Smith-Miles [1]	2006	Recommends HP settings			•		GS	C5.0	112	UCI, KDC	10-CV	Acc
Miranda & Prudêncio [30]	2013	Recommends HP settings			•		GS	AT	60	UCI	10-CV	Acc
Lorena & et al. [25]	2018	Recommends HP settings			•		GS	kNN	39	UCI	10-CV	NMSE
Reif et al. [38]	2011	Predicts training runtime			•		GS	SVM	123	UCI	-	PMCC NAE
Miranda et al. [31]	2014	Recommends initial values for HP optimization			•		GS,PSO	kNN	100	UCI	10-CV	Acc
Auto-sKLearn [14]	2015	Recommends initial values for HP optimization			•		GS, SMBO	kNN	140	OpenML	Nested-CV	Acc
Reif et al. [40]	2014	Estimates predictive performance for a HP setting			•		GS	SVM	54	UCI, Statlib	10-CV	RMSE PMCC
Miranda et al. [32]	2012	Recommends initial values for HP optimization			•		GS, PSO	kNN	40	UCI, WEKA	10-CV	Acc
Gomes et al. [16]	2012	Recommends initial values for HP optimization			•		GS PSO,TS	kNN	40	WEKA	10-CV	NMSE
Priya et al. [36]	2012	Predicts training runtime	•	•			GA	J48, SVM Bagging kNN, NB Jrip	78	UCI	5-CV	MAD
Reif et al. [39]	2012	Recommends initial values for HP optimization			•		GS,GA	kNN	102	UCI, Statlib	10-CV	Acc
Wistuba et al. [50]	2018	Estimates predictive performance for a HP setting		•	•		GS	GP	109	UCI, WEKA	Holdout	-
Eggensperger et al. [13]	2018	Estimates predictive performance for a HP setting		•	•		ROAR, Irace RS, ParamILS	RF	11	AClib	Holdout	RMSE SpCorr
Ridd & Giraud-Carrier [41]	2014	Predicts HP tuning improvement			•		PSO	J48, RF SVM	326	UCI, WEKA	-	AUC
Mantovani et al. [28]	2015	Predicts HP tuning necessity		•	•		GS, RS, EDA GA,PSO	J48, SVM, LR NB, kNN, RF	143	UCI	Nested-CV	BAC
Sanders & Giraud-Carrier [42]	2017	Predicts HP tuning improvement	•	•	•		GA	MLP	229	OpenML	10-CV	AUC

- there is no guarantee that the generated meta-examples are different from each other (intersection between training and test data), since OpenML stores different versions of the same dataset. This could lead to biased results; and

- experiments are not reproducible since most of the details are not explained, and the code is not available.

Mantovani et al. [28] proposed a MtL recommender system to predict when SVM HP tuning is necessary, i.e., when tuning is likely to improve the generalization power of the models. The meta-dataset was created by extracting characteristics based on simple and data complexity measures from 143 classification datasets. In the base-level, different meta-heuristics (PSO, GA and EDA) were used to tune the SVM HPs using a nested-CV resampling strategy. An ensemble of meta-models achieved the best predictive performance assessed by the F-Score using simple meta-features. Besides these promising results, this study presents some shortcomings, such as:

- the best predictive performance at the meta-level is moderately low;

- when the method recommends tuning, the meta-heuristic which would lead to the best performance is not recommended;

- only two default HP settings were investigated. In general, users try more than two settings before tuning;

- there is no evidence that this method and the results can be generalized to other ML algorithms.

The main differences between the proposed approach and the most related work are shown in Table 2. It is important to note that although these are the most similar studies we have found in the literature, they addressed different problems. Furthermore, the meta-datasets generated by each study were also different, since they were generated using different datasets, target algorithms, meta-features, and labeling rules. Because of these particularities, the straightforward comparison of these studies is unfeasible. The only free choice we could explore is the same meta-features adopted by them. In fact, Ridd & Giraud-Carrier [41] and Sanders & Giraud-Carrier [42] used a total of 68 meta-features which are included in our experimental setup (Section 4.4).

Based on the literature, we realized that there is room for improvement in terms of predicting HP tuning necessity for ML algorithms and to better understand this meta-learning process. Our present work attempts to fill this gap yielding meta-models with high predictive performance and reasons why their decisions were made. To do this, we have comprehensive and systematically evaluated different categories of meta-features and preprocessing tasks, such as meta-feature selection and data balancing, and different default HP values.

Table 2: The most related studies to our current approach. In the Goal prediction column, "improv." means improvement prediction. In the Task column, "class" denotes classification, while "regr" denotes regression.

Study	Goal prediction	Task	Tuning setup (base-level)	Labeling rule	Target algorithm	Meta features
Ridd & Giraud-Carrier [41]	improv.	class	Not detailed	Accuracy threshold	CASH (20 algs.)	Simple Statistical Landmarkers Model-based
Mantovani et al. [28]	tuning necessity	class	Nested-CVs holdout (inner) 10-CV (outer) BAC (fitness)	Confidence interval	SVM	Simple Data complexity
Sanders & Giraud-Carrier [42]	improv.	regr	10-CV (single) AUC (fitness)	Confidence interval	CART MLP SVM	Simple Statistical Landmarkers Model-based
Current study	tuning necessity	class	Nested-CVs 3-fold (inner) 10-fold (outer) BAC (fitness)	Wilcoxon test	SVM J48	Many (see Section 4.4)

3.6. Summary of Literature Overview

The literature review carried out by the authors found a large increase in the use of MtL for tasks related to SVM HP tuning. The authors found 18 related works, but only three of them investigated specifically when HP tuning is necessary or its improvement (see Section 3.5). Overall, the following aspects were observed:

- fourteen of the studies created the meta-knowledge using GS to tune the $\gamma \times C$ HP;

- most of the studies also evaluated the resultant models with a single CV procedure and the simple Acc evaluation measure;

- half of the studies used in most 100 datasets. In [41], the authors used more than 300 datasets, but all of them for binary classification;

- all investigated a small number of categories to generate meta-features;

- nine of the studies used only kNN as meta-learner;

- three of the studies applied meta-feature selection techniques to the meta-features;

- two of the studies provided the complete resources necessary for the reproducibility of experiments;

- None of the studies combined all these six previous issues.

In order to provide new insights in the investigation of how the use of MtL in the SVM HP tuning process can affect its predictive performance, this paper extends previous works by exploring:

- Meta-features produced by measures from different categories;

- Use of different learning algorithms as meta-learners;

- Adoption of a reproducible and rigorous experimental methodology at base and meta-learning levels; and

- Assessment of the use of meta-feature selection techniques to evaluate and select meta-features.

One of the main contributions of this paper is the analysis of the meta-model predictions to identify when it is better to use default or tuned HP values for the SVMs, and which meta-features have a major role in this identification.

4. Experimental methodology

In this paper, experiments were carried out using MtL ideas to predict whether hyperparameter tuning can significantly improve SVM induced models, when compared with performance provided by their default hyperparameter values[5]. The framework treats the recommendation problem as a binary classification task and is formally defined as follows:

Let $\mathcal{D}$ be the dataset collection. Each dataset $d_j \in \mathcal{D}$ is described by a vector $mf(d_j) = (mf_{j,1}, ..., mf_{j,K})$ of K meta-features, with $mf_{j,k} \in \mathcal{M}$, the set of all known meta-features. Additionally, let Ω be a statistical labeling rule based on the prior evaluations from tuned and default hyperparameter settings ($\mathcal{P}$). Given a significance level α, Ω maps prior performances to a binary classification task: $\Omega : P \times \alpha \to C \mid C = \{tune, not\_tune\}$. Thus, we can train a *meta-learner* $\mathcal{L}$ to predict whether optimization will lead to significant improvement on new datasets $d_i \notin \mathcal{D}$, i.e.:

$$\mathcal{L} : \mathcal{M} \times \Omega \to C \tag{1}$$

---

[5]The `e1071` package was used to implement SVMs. It is the LibSVM [10] interface to the R environment.

Figure 1 shows graphically the general framework, linking two-level learning steps: the *base level*, where the hyperparameter tuning process is performed for different datasets ($\mathcal{D}$); and the *meta level*, where the meta-features ($\mathcal{M}$) from these datasets are extracted, the meta-examples are labeled according to tuning experiments (Ω, $\mathcal{P}$) and the recommendation to a new unseen dataset occurs ($d_i \notin \mathcal{D}$). Further subsections will describe in detail each one of its components.
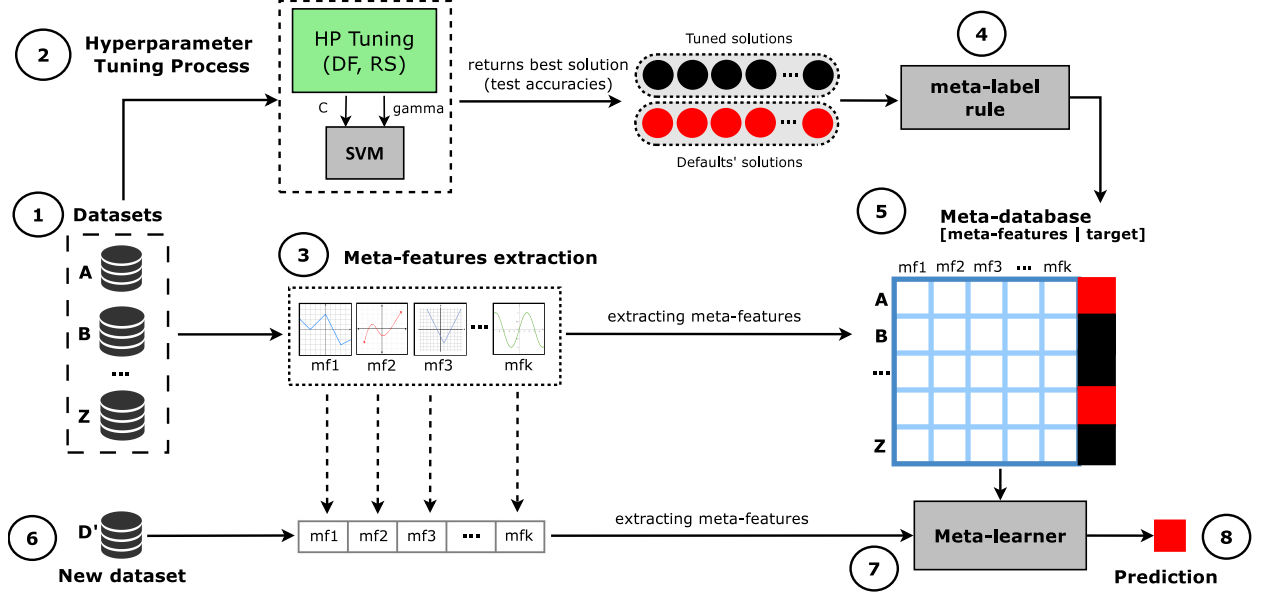
Figure 1: Meta-learning system to predict whether hyperparameter tuning is required (Adapted from [28]). At the figure, "mf" means meta-feature.

4.1. OpenML classification datasets

The experiments used datasets from OpenML [47], a free scientific platform for standardization of experiments and sharing empirical results. OpenML supports reproducibility since any researcher can have access and use the same data for benchmark purposes. A total of 156 binary and multiclass classification datasets ($\mathcal{D}$) from different application domains were selected for the experiments (Item 01 in Figure 1). From all the available and active datasets, those meeting the following criteria were selected:

(a) number of features does not exceed $1,500$;

(b) number of instances between 100 and $50,000$;

(c) must not be a reduced, modified or binarized version of the original classification problem[6];

---

[6]More details about dataset versions can be found in the OpenML paper [47] and documentation page: `https://docs. openml.org/#data`.

(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 resampling.

These criteria are meant to ensure a proper evaluation (a-b), e.g. datasets should not be so small or so large that they cause memory problems; they should not be too similar (c-d) (to avoid data leakage in our evaluation); and allow the use of 10-fold CV stratified resampling, given the high probability of dealing with imbalanced datasets (e). We also excluded datasets already used in our related work on defining optimized defaults, resulting in 156 datasets to be used in our meta-dataset. All datasets meeting these criteria and their main characteristics are presented on the study page at OpenML[7].

In order to be suitable for SVMs, datasets were preprocessed: any constant or identifier attributes were removed; the logical attributes were converted into values $\in \{0, 1\}$; missing values were imputed by the median for numerical attributes, and a new category for categorical ones; all categorical attributes were converted into the 1-N encoding; all attributes were normalized with $\mu = 0$ and $\sigma = 1$. The `OpenML` [9][8] package was used to obtain and select datasets from the `OpenML` website, while functions from the `mlr` [4][9] package were used to preprocess them.

4.2. SVM hyperparameter space

The SVM hyperparameter space used in the experiments is presented in Table 3. For each hyperparameter, the table shows its symbol, name, type, range/options, scale transformation applied, default values provided by LibSVM [10] and whether it was tuned. 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) [19]. For C and γ, the selected range covers the hyperspace investigated in [41]. LibSVM default values are $C = 1$, and $\gamma = 1/N$, where N is the number of features of the dataset under analysis[10].

4.3. Hyperparameter tuning process

The hyperparameter tuning process is depicted in Figure 1 (Item 2). Based on the defined hyperspace, SVMs hyperparameters were adjusted through a Random Search (RS) technique for all datasets selected. The tuning process was carried out using nested CV resamplings [22], an "*unbiased performance evaluation methodology*" that correctly accounts for any overfitting that may occur in the model selection (considering the hyperparameter tuning). In fact, most of the important/current state of the art studies, including the `Auto-WEKA`[11] [21, 46] and `Auto-skLearn`[12] [14] tools, have been using the nested CV methodology for

---

[7]`https://www.openml.org/s/52/data`

[8]`https://github.com/openml/openml-r`

[9]`https://github.com/mlr-org/mlr`

[10]LibSVM default values can be consulted at `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`

[11]`http://www.cs.ubc.ca/labs/beta/Projects/autoweka/`

[12]`https://github.com/automl/auto-sklearn`

Table 3: SVM hyperparameter space used in experiments. The following was shown for each hyperparameter: its symbol, name, type, range/options, scale transformation applied, default values and whether it was tuned.

Symbol	Hyperparameter	Type	Range/Options	Scale	Default	Tuned
k	kernel	categorical	{RBF}	-	RBF	**x**
C	cost	real	$[2^{-15}, 2^{15}]$	log	1	✓
γ	width of the kernel	real	$[2^{-15}, 2^{15}]$	log	$1/N$	✓

hyperparameter selection and assessment. Thus, nested-CVs were also adopted in this current study. The number of outer folds was defined as $M = 10$ such as in [22]. Due to runtime constraints, the number of inner folds was set to $N = 3$.

A budget with a maximum of 300 evaluations per (inner) fold was considered. A comparative experiment using different budget sizes for SVMs was presented in [27]. Results suggested that only a few iterations are required to reach good solutions in the optima hyperspace region. Indeed, in most of the cases, tuning has reached good performance values after 250-300 steps. Among techniques used by the authors, the Random Search (RS) was able to find near-optimum hyperparameter settings like the most complex tuning techniques did. Overall, they did not show statistical differences regarding performance and presented a runtime lower than population-based techniques[13].

Hence, the tuning setup detailed in Table 4 generates a total of $90,000 = 10$ (outer folds) $\times 3$ (inner folds) $\times 300$ (budget) $\times 10$ (seeds) HP settings during the search process for a single dataset. Tuning jobs were parallelized in a cluster facility provided by our university[14] and took four months to be completed.

4.4. Meta-features

The meta-datasets used in the experiments were generated out of '*meta-features*' ($\mathcal{M}$) describing each dataset (Figure 1 - Item 3). These meta-features were extracted by applying a set of measures mf_i to the original datasets which obtain likely relevant characteristics from these datasets. A tool was developed to extract the meta-features and can be found on `GitHub`[15], as presented in Table 8. We extracted a set of 80 meta-features from different categories, as described in Section 2. The set includes all the meta-features explored by the studies described in Subsection 3.5. The exact number of meta-features used from each category can be seen in Table 5. A complete description of them may be found in Tables A.10 and A.11 (Appendix A).

---

[13]These findings go towards what was previously described in [3].

[14]http://www.cemeai.icmc.usp.br/Euler/index.html

[15]https://github.com/rgmantovani/MfeatExtractor

Table 4: Hyperparameter base level learning experimental setup.

Element	Method	R package
Tuning techniques	Random Search	`mlr`
Base Algorithm	Support Vector Machines	`e1071`
Outer resampling	10-fold cross-validation	`mlr`
Inner resampling	3-fold cross-validation	`mlr`
Optimized measure	{Balanced per class accuracy}	`mlr`
Evaluation measure	{Balanced per class accuracy, Optimization paths }	`mlr`
Budget	300 iterations	
Repetitions	10 times with different seeds	-
	seeds = $\{1, \ldots, 10\}$	-
Baselines	LibSVM defaults	`e1071`
	optimized defaults	

Table 5: Meta-feature category used in experiments.

Acronym	Category	#N	Description
SM	Simple	17	Simple measures
ST	Statistical	7	Statistical measures
IN	Information-theoretic	8	Information theory measures
MB	Model-based (trees)	17	Features extracted from decision tree models
LM	Landmarking	8	The performance of some ML algorithms
DC	Data Complexity	14	Measures that analyze the complexity of a problem
CN	Complex Networks	9	Measures based on complex networks
	Total	80	

4.5. Meta-targets

The last meta-feature is the meta-target, whose value indicates whether the HP tuning significantly improved the predictive performance of the SVM model, compared with the use of default values. Since the HP tuning experiments contain several and diverse datasets, many of them may be imbalanced. Hence, the Balanced per class Accuracy (BAC) measure [8] was used as the fitness value during tuning, as well as for the final model assessment at the base-level learning[16].

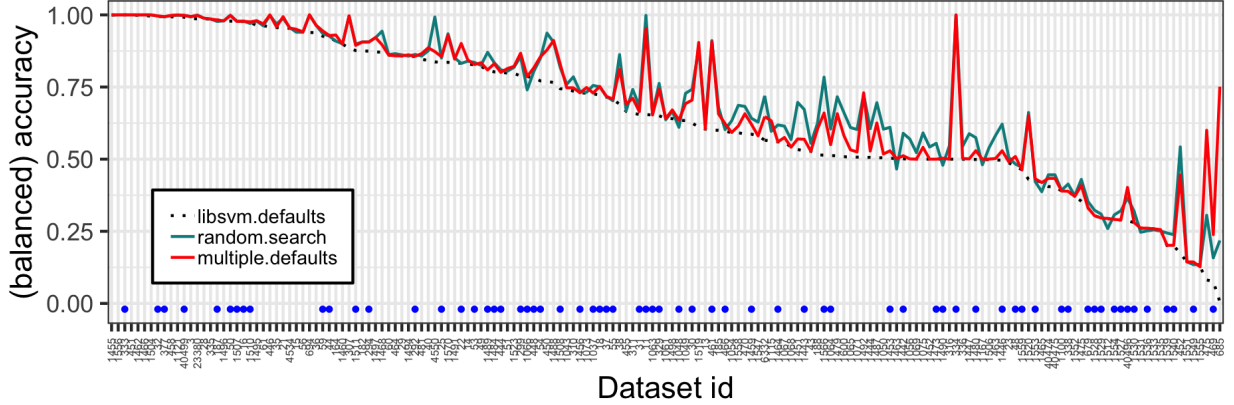[16]These performance values are assessed by BAC using a nested-CV resampling method.

Figure 2: Average balanced per class accuracies comparing LibSVM default (libsvm.defaults), Multiple optimized default hyperparameter settings (multiple.defaults) and Random Search tuning technique (random.search) when defining the meta-target of each meta-example.

The so-called "*meta-label rule*" Ω (Item 4 - Figure 1) applies the Wilcoxon paired-test to compare the solutions achieved by the RS technique ($\mathcal{P}_{tun}$) and the default HP settings ($\mathcal{P}_{def}$). Given a dataset $d_i \in \mathcal{D}$ and a significance level (α), if the HP tuned solutions were significantly better than those provided by defaults, its corresponding meta-example is labeled as 'Tuning' (C_{tun}); otherwise, it receives the 'Default' label (C_{def}).

When performing the Wilcoxon test, three different values of $\alpha = \{0.10,\ 0.05,\ 0.01\}$ were considered, resulting in three meta-datasets with different class distributions (Item 5 - Figure 1). The different significance levels (α) influence how strict the recommending system is when evaluating if tuning improved models' performance compared to the use of default HP values. The smaller the significance stricter it is, i.e., there must be greater confidence than the tuned hyper-parameter values obtained by improving the performance of the induced models. It may also imply in different labels for the same meta-example when evaluating different α values. The initial experimental designs only compared LibSVM suggested default values with the HP tuned solutions. The resulting meta-datasets presented a high imbalance rate, prevailing the "Tuning" class. It was difficult to induce a meta-model with high predictive performance using this highly imbalanced data. An alternative to deal with this problem was to consider the optimized default HP values proposed in [29]. The optimized default values were obtained optimizing a common set of HP values, able to induce models with high predictive performance, for a group of datasets.

Figure 2 illustrates the benefits of using multiple default settings: LibSVM and optimized default values. In this figure, the x-axis identifies datasets by their OpenML ids, listing them decreasingly by the balanced per class accuracy performances (y-axis) obtained using LibSVM defaults hyperparameter values. This

figure shows three different curves:

- `libsvm.defaults`: a black dotted line representing the averaged performance values obtained using LibSVM default hyperparameter values. It represents the choice of a user using LibSVM defaults;

- `random.search`: a green line representing the averaged performance values obtained using the Random Search (RS) technique for tuning. It represents the choice of always tuning SVMs hyperparameters; and

- `multiple.defaults`: a red line representing the best choice considering the LibSVM and optimized defaults hyperparameter values. It represents our approach, exploring multiple default values.

By looking at the difference between the black and green lines, it is possible to observe that tuned models using RS outperformed models using default settings (provided by LibSVM) for around 2/3 of the datasets. However, when we consider multiple default settings (the best setting between LibSVM and optimized values), identified by the red curve, their performance values were close to the performance with tuned values. Thus, the meta-target labelling rule considered the difference between the predictive performances with tuned hyperparameters and the best predictive performances with multiple default HP values. A side effect of using multiple default HP values is a more class-balanced meta-dataset, increasing the proportion of meta-examples labeled with "default" use. As a result, the imbalance rate[17] in the meta-datasets was reduced from ≈ 2.6 to 1.7.

Table 6 presents for each resultant meta-dataset: the α value used to generate the labels; the number of meta-examples, the number of meta-features and the class distribution. It is important to observe that none of these 156 datasets were used in a related previous study that produced optimized default HP setting [29].

In our experimental setup, the null hypothesis of the statistical meta-label rule states that there is no significant difference between tuned and default SVM HP settings. Since we are concerned about preventing tuning HPs when it is not necessary, a type I error is defined as labeling a meta-example as "`Tuning`" when its label is, in fact, "`Defaults`'. Therefore, the lower the α, the higher the probability that the improvement achieved by *tuned* values is not due to chance. On the other hand, the higher the alpha, the lower the requirement that the performance gain by the tuning process is significant compared to default values.

Since we are controlling the error of labeling a meta-example as "`Tuning`', smaller α values will lead to a greater number of "*default* meta-examples. On the contrary, the greater the alpha value, the greater the number of meta-examples labeled as"*tuning*". As can be seen in Table 6, a value of $\alpha = 0.10$ implies more instances with the meta-target "`Tuning`" than when using $\alpha = 0.01$. In summary, if predictive performance is more critical, the user should set the significance level as high as possible (e.g., $\alpha = 0.10$). On the other

---

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

hand, if the user is concerned about computational cost, the significance level should be set to smaller values (e.g., $\alpha = 0.01$). An example of this effect can be seen in Figure 2, where the blue dots represent all the datasets where defaults should be used, i.e., tuning is not statistically significant better (for $\alpha = 0.05$).

Table 6: Meta-datasets generated from experiments with SVMs.

| Meta-dataset | α | Meta examples | Meta features | Class Distribution | |
				Tuning	Default
SVM_90	0.10	156	80	102	54
SVM_95	0.05	156	80	98	58
SVM_99	0.01	156	80	94	62

4.6. Experimental Setup

Seven classification algorithms were used as meta-learners (Item 7 - Figure 1): Support Vector Machines (SVMs), Classification and Regression Tree (CART), Random Forest (RF), k-Nearest Neighbors (kNN), Naïve-Bayes (NB) Logistic Regression (LR) and Gaussian Processs (GPs). These algorithms were chosen because they follow different learning paradigms with different learning biases. All seven algorithms were applied to the meta-datasets using a 10-fold CV resampling strategy and repeated 20 times with different seeds (for reproducibility). All the meta-datasets presented in Table 6 are binary classification problems. Thus, meta-learners' predictions were assessed using the Area Under the ROC curve (AUC) performance measure, a more robust metric than BAC for binary problems. Moreover, AUC also enable us to evaluate the influence of different threshold values on predictions. Three options were also investigated at the meta-level:

(i) *Meta-feature Selection*: as each meta-example is described by many meta-features, it may be the case that just a small subset of them is necessary to induce meta-models with high accuracy. Thus, a Sequential Forward Selection (SFS) feature selection option was added to the meta-learning experimental setup. The SFS method starts from an empty set of meta-features, and in each step, the meta-feature increasing the performance measure the most is added to the model. It stops when a minimum required value of improvement (*alpha*=0.01) is not satisfied. Internally, it also performs a stratified 3-fold CV assessing the resultant models also according to the AUC measure;

(ii) *Tuning*: since the hyperparameter values of the meta-learners may also affect their performance, tuning of the meta-learners was also considered in the experimental setup. A simple RS technique was performed with a budget of 300 evaluations and resultant models assessed through an inner stratified 3-fold CV and AUC measure. Table B.12 (Appendix Appendix B) shows the hyperspace considered for tuning the meta-learners.

(iii) *Data balancing*: even using the optimized default HP values, the classes in the meta-datasets were imbalanced. Thus, to reduce this imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) [11] technique was used in the experiments.

Some of the algorithms' implementations selected as meta-learners use a data scaling process by default. This is the case of the SVM, kNN and GP meta-learners. A preliminary experiment showed that removing this option decreases their predictive performance considerably, while it does not affect the other algorithms. When data scaling is considered for all algorithms, the performance values of RF, CART, NB and LR meta-learners were decreased. Thus, data scaling was not considered as an option, and the algorithms used their default procedures, with which they obtained their best performance values. Two baselines were also adopted for comparisons: a meta-dataset composed only by simple meta-features and another with data complexity ones. Both categories of meta-features were investigated before by related studies listed in Section 3.5.

4.7. Repositories for the coding used in this study

Details of the base-level tuning and meta-learning experiments are publicly available in the OpenML Studies (ids 52 and 58, respectively). In the corresponding pages, all datasets, classification tasks, algorithms/flows and results are listed and available for reproducibility. The code used for the HP tuning process (`HpTuning`), extracting meta-features (`MfeatExtractor`), running meta-learning (`mtlSuite`), and performing the graphical analyses (`MtlAnalysis`) are hosted at `GitHub`. All of these repositories are also listed in Table 8.

5. Results and Discussion

The main experimental results are described in the next subsections. First, an overview of the predictive performance of the meta-models for the predicting task when it is worth performing SVM HP tuning. Next, different experimental setups and preprocessing techniques, such as dimensionality reduction, are evaluated. Finally, the predictions and meta-knowledge produced by the meta-models are analyzed.

5.1. Average performance

Figure 3 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 Figure 3a, the x-axis shows the meta-learners while the y-axis shows their predictive performance assessed by the AUC averaged over 30 repetitions. In addition, 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

Table 7: Meta-learning experimental setup.

| Element	Method	R package
Meta-learner	Support Vector Machines (SVMs)	e1071
	Classification and Regression Tree (CART)	rpart
	Random Forest (RF)	randomForest
	k-Nearest Neighbors (kNN)	kknn
	Naïve-Bayes (NB)	e1071
	Logistic Regression (LR)	gbm
	Gaussian Processs (GPs)	kernlab
Resampling	10-fold CV	mlr
Meta-feature Selection	Sequential Forward Search - $alpha = 0.01$ inner 3-CV - measure AUC	mlr
Tuning	Random Search (RS) budget $= 300$ inner 3-CV - measure AUC	mlr
Data Balancing	SMOTE oversampling rate $= 2$	mlr
Repetitions	30 times with different seeds	-
	from the interval $\{1, \ldots, 30\}$	-
Evaluation measures	AUC	mlr
	predictions (prob)	
Baselines	Simple meta-features	-
	Data complexity meta-features	-

$\alpha = 0.05$ was applied to assess the statistical significance of the predictive performance differences obtained by the meta-models with *all* meta-features, when compared to the second best approach.

An upward green triangle (▲) at the x-axis identifies situations where using all the meta-features were statistically better. On the other hand, red downward triangles (▼) show results where one of the alternative approaches 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 (*complex*) meta-features, achieving AUC values nearly 0.80 for all α levels. These meta-models were also statistically better than

Table 8: Repositories with tools developed by the authors and results generated by experiments.

Task/Experiment	Website/Repository
Hyperparameter tuning code	`https://github.com/rgmantovani/HpTuning`
Hyperparameter tuning results	`https://www.openml.org/s/52`
Meta-feature extraction	`https://github.com/rgmantovani/MfeatExtractor`
Meta-learning code	`https://github.com/rgmantovani/mtlSuite`
Meta-learning results	`https://www.openml.org/s/58`
Graphical Analysis	`https://github.com/rgmantovani/MtlAnalysis`
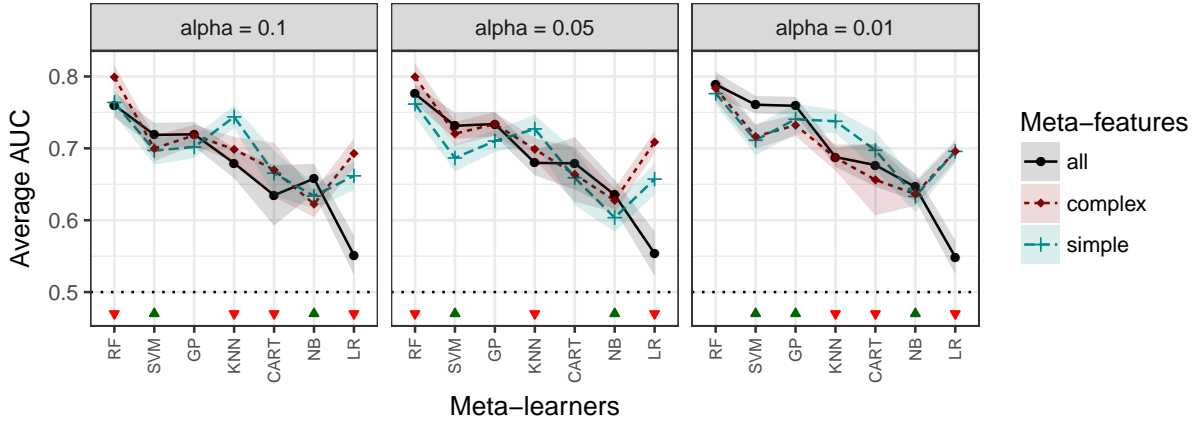
(a) Meta-learners average AUC performance on SVMs meta-datasets. The black dotted line at $AUC = 0.5$ represents the predictive performance of `ZeroR` and `Random` meta-models.
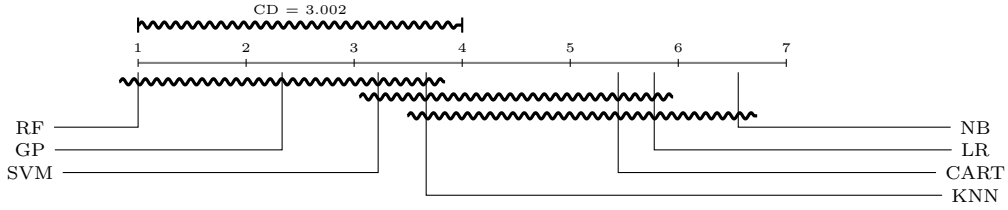
(b) Comparison of the AUC values of the induced meta-model according to the Friedman-Nemenyi test ($\alpha = 0.05$). Groups of algorithms that are not significantly different are connected.

Figure 3: AUC performance values obtained by all meta-learners considering different meta-features' categories. Results are averaged considering 30 repetitions.

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 ≈ 0.8.

When the value of α in the meta-label rule is reduced, predictive performances using data complexity and all the available meta-features tend to show similar distributions. The meta-learners obtained their best

AUC values with the highest assumption ($\alpha = 0.99$). Overall, varying the α value did not substantially change the predictive performance of the evaluated algorithms. In fact, few meta-examples had their meta-targets modified by the meta-rule with different values of α. Thus, the predictions in the different scenarios are mostly the same and the performances remained similar.

Regarding predictive performance, RF, SVM, GP and kNN induced accurate meta-models for the three meta-dataset variations. The AUC value varied in the interval $\{0.70, 0.80\}$. 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`[18] baselines obtained AUC of 0.5 in all these meta-datasets[19].

The Friedman test [12], with a significance level of $\alpha = 0.05$, was used to assess the statistical significance of the meta-learners. In the comparisons, we considered the algorithms' performance across the combination of the meta-datasets and the categories of meta-features. The null hypothesis states that all the meta-learners are equivalent regarding the Area Under the ROC curve (AUC) 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 3b presents the resultant Critical Difference (CD) diagram. Algorithms are connected when there are *no* significant differences between them. The top-ranked meta-learner was the RF with an average rank of 1.0, followed by GP (2.3), SVM (3.2) and kNN (3.6). They did not present statistically differences among them, but mostly did when compared with simpler algorithms: CART (5.4), LR (5.7) and NB(6.5). Even not being statistically better than all the other choices, the RF was always ranked at the top regardless of the meta-dataset and meta-features.

Although the best result was obtained using Data Complexity (DC) meta-features ("`complex`"), most of the meta-learners achieved their highest AUC performance values exploring all the available meta-features. Thus, since we want to analyze the influence of different categories of meta-features when inducing meta-models, and given the possibility of selecting different subsets from all the categories, we decided to explore all of them in the next analysis.

5.2. Evaluating different setups

Due to the large difference among meta-learners results, three different setups were also evaluated to improve their predictive performances and enable a comprehensive analysis of the investigated alternatives:

(i) `featsel` - meta-feature selection via Sequential Forward Selection (SFS) [4];

(ii) `tuned` - HP tuning of the meta-learners using a simple Random Search (RS) technique;

---

[18]This classifier simply predicts the majority class.

[19]The AUC performance values were assessed using the implementations provided by the `mlr` R package.

(iii) `smote`: dataset balancing with SMOTE [11].

They were compared with the original meta-data with no additional process (`none`), which is the baseline for these analyses. These setups were not performed at the same time to avoid overfitting, since the meta-datasets have 156 meta-examples and, depending on their combinations, three levels of CVs would be used to assess models. For example: if feature meta-selection and HP tuning were enabled at the meta-level, one CV would be used for meta-feature selection, one for tuning and another to assess the resulting models.

Figure 3 summarizes the main aspects of these experimental results. The top figure 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). Similarly to Figure 3a, the statistical analysis is also presented. Every time an upward green triangle is placed at the x-axis, the 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, meta-feature selection or SMOTE could statistically improve the predictive performance of the meta-models. In the remaining cases, the meta-models were equivalent.

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. Depending on the experimental setup, kNN and LR also presented good predictive performances. Regarding the HP tuning (`tuned`) of the meta-learners, only for kNN the performances slightly improved for all the alpha values. Using just SMOTE resulted in improved results for SVM, GP and CART meta-learners. In general, it produced small improvements, but most of them were statistically significant. When used with tuning or meta-feature selection, it affected the algorithms in different ways: for SVM and GP, the performance improved; for LR, NB and kNN there was no benefit; the other algorithms were not affected by its use. The low gain obtained using SMOTE may be due to the fact that data imbalance was already reduced using the optimized defaults when defining meta-targets.

Using meta-feature selection (`featsel`) deteriorated the performance of the SVM, RF, GPs and CART meta-learners. On the other hand, it clearly improved the kNN, LR and NB performances for most cases. kNN benefited from using a subset of meta-features to maximize the importance of more relevant meta-features. For NB and LR, selecting a subset of the attributes reduced the presence of noise and irrelevant attributes. Furthermore, it is important to observe that the meta-models induced with the selected features presented the highest standard deviation between the setups (light area along the curve). A possible reason is the different subsets selected every time meta-feature selection is performed for the 30 repetitions.

Additionally, Figure 4b presents a ranking with all the combinations of meta-learners and experimental setups. At the x-axis, they are presented in ascending order according to their average ranking for the three scenarios (α values), shown at the y-axis. The more red the squares, the lower the ranking, i.e., the better the results.
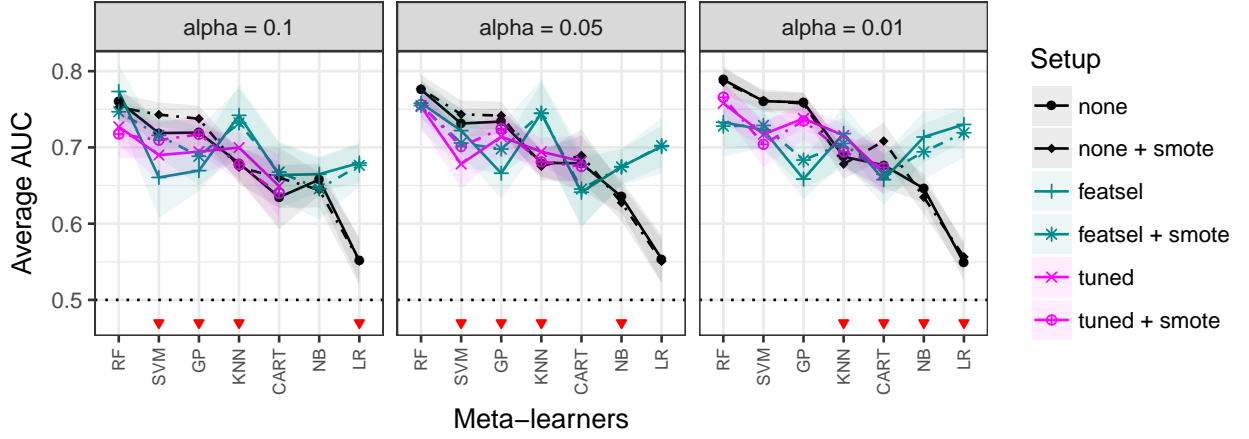
(a) Average AUC performance values. The black dotted line at $AUC = 0.5$ represents the predictive performance of `ZeroR` and `Random` meta-models.
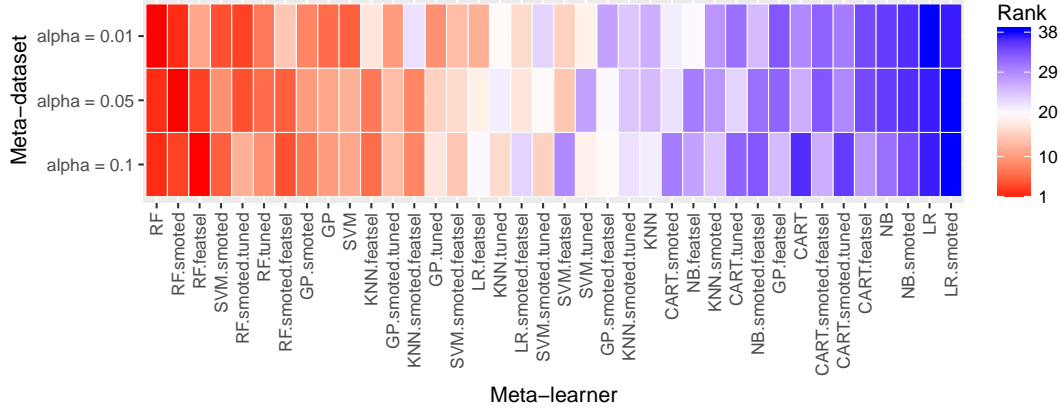
(b) Average AUC ranking values.
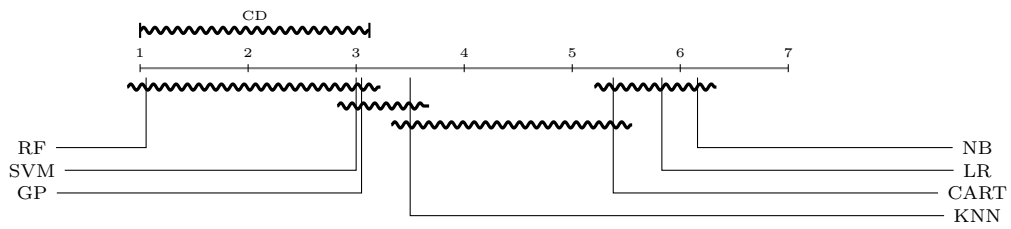
(c) Comparison of the AUC values of the induced meta-model according to the Friedman-Nemenyi test ($\alpha = 0.05$). Groups of algorithms that are not significantly different are connected.

Figure 4: AUC performance values obtained by all meta-learners considering different experimental setups. Results are averaged over 30 repetitions.

As previously reported, RF with no additional option was the best-ranked method, followed by its smoted versions. The SVM, GP and kNN versions are in the next positions. The Friedman test with a significance level of $\alpha = 0.05$ was also used to assess the statistical significance of the meta-learners when using different

experimental setups in different meta-datasets. Figure 4c shows the resultant CD diagram. Results are quite similar to those reported in Figure 3b: the RF algorithm was the best algorithm with an average ranking of 1.05, and is statistically better than most of the meta-learners, except for SVM and GP.

Since there were no improvements considering the maximum AUC values achieved so far, and the results from the RF meta-learners were still the best-ranked, the next subsections will analyze the relative importance of the meta-features according to the final RF induced models.

5.3. Importance of meta-features

From the induced RF meta-models, the relative importance of the meta-features based on the Gini impurity index used to calculate the node splits [7]. Figure 5 shows the average relative importance of the meta-features obtained from the RF meta-models. The relative importance is shown for the experiments considering all meta-features and $\alpha = 0.05$ (middle case). At the x-axis, meta-features are presented in decreasing order according to their average relative importance values. From this point, anytime a specific meta-feature is mentioned we present its name with a prefix indicating its category (according to Table 5).

Figure 5: Average meta-features relative importance obtained from RF meta-models. The names of the meta-features in the x-axis follow the acronyms presented in Tables A.10 and A.11 in Appendix A.

Since no negative value (negative relative importance) was obtained, no meta-feature was discharged to build meta-models. It also shows that a large number of meta-features were relevant to the induction of the meta-models, a possible reason for why meta-feature selection produced worse results for most of the meta-learners.

The most important meta-feature was a landmarking meta-feature: "LM.stump_sd", which describes the standard deviation of the number of examples correctly classified by a decision stump. It measures the

25

complexity of the problem considering its simplicity. The second most important was a simple meta-feature: "SM.classes_min", which measures the minority class size. The third was also a simple meta-feature: "SM.classes_sd", which describes the standard deviation of the number of examples per class. These meta-features together strongly indicate that for RF, the most important meta-features are related with class imbalance. A rule extracted from a model induced by RF states that if the dataset is imbalanced, it is better to use default HP for SVMs. The other important meta-features were:

- "IN.nClEnt" and "IN.mutInf": these are information-theoretical meta-features. While the first describes the class entropy for a normalized base level dataset, the second measures the mutual information, a reduction of uncertainty about one random feature given the knowledge of another;

- "CN.betweenness": betweenness centrality is a meta-feature derived from complex networks that measures, for a set of vertex and edges, the average number of shortest paths that traverses them. The value will be small for simple datasets, and high for complex datasets;

- "DC.l1" and "DC.t2": these are data complexity meta-features. While the first measures the minimum of an error function for a linear classifier, the second measures the average number of points per dimension. These features are related with the class separability (l1), and the geometry of the problem's dimension (t2);

- "SM.dimension": this is a simple meta-feature that measures the relation between the number of examples and attributes in a dataset;

- "CN.maxComp": this is another complex-network meta-feature. It measures the maximum number of connected components in a 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 the classes are between their levels. Data complexity meta-features try to extract information related to the class separability. The stump meta-feature works along the same lines, trying to identify the complexity of the problem by simple landmarking. The information-theoretical meta-features indirectly checks how powerful the dataset attributes are to solve the classification problem.

Although summarized rules cannot be obtained from RF meta-models, the analysis of meta-features importance provides some useful information. For instance, dataset characteristics such as the data balancing, class sizes, complexity and linearity were considered relevant to recommend when HP tuning is required.

5.4. Linearity Hypothesis

The previous sections, in particular the RF meta-analysis, suggest that linearity is a key aspect to decide between the recommendation of default or tuned HPs values. Experimental results indicate that default HP values might be good for classification tasks with high linear separability. As a consequence, tuning would be required for tasks with complex decision surfaces, where SVMs would need to find irregular decision boundaries.
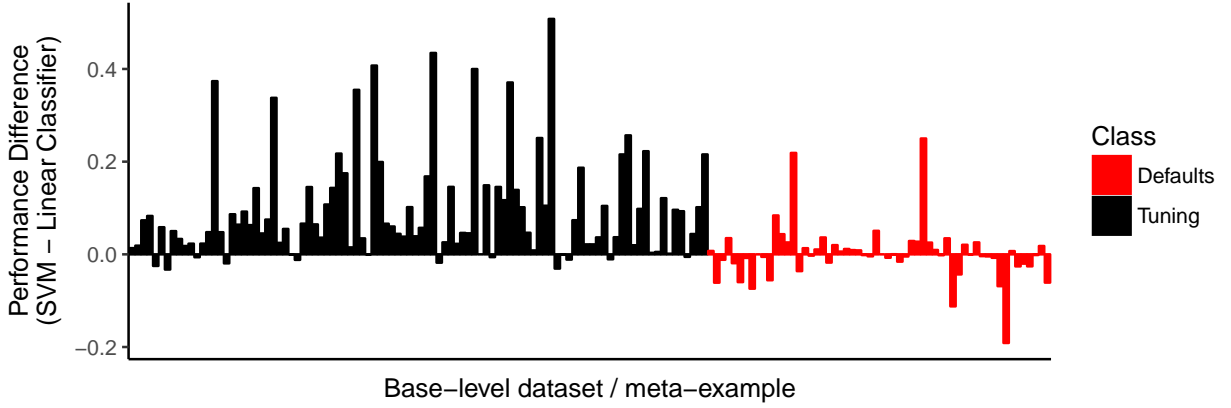
(a) Performance differences between SVM and a linear classifier in all the base-level datasets.
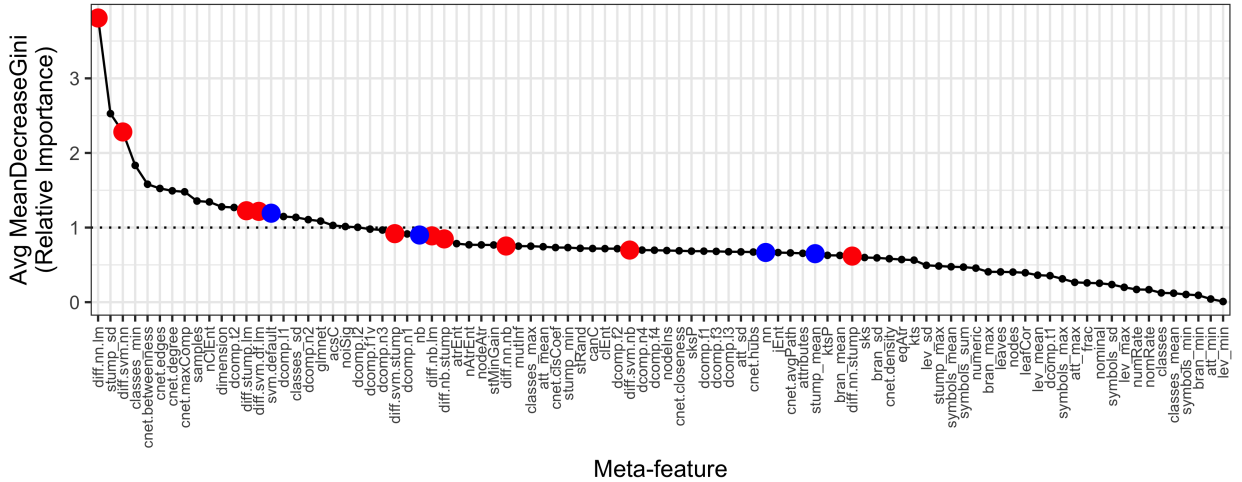
(b) Average relative importance of the meta-features obtained from RF meta-models. The names of the meta-features in the x-axis follow the acronyms presented in Tables A.10 and A.11 in Appendix A.

Figure 6: Linearity hypothesis results considering relative landmarking meta-features.

In order to investigate this hypothesis, a linear classifier was also evaluated in all the available 156 datasets using the same base-level experimental setup described in Table 4. 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 or equal to the meta-examples labeled as "Tuning".

27

Figure 6a shows the performance differences obtained in all the datasets at the base-level. Datasets at the x-axis are split based on their meta-target labels, "`Tuning`', left side, in black, and "`Defaults`", right side, in red. Despite some outliers, the performance differences for "`Tuning`" meta-examples are in general much higher than those for the "`Defaults`" meta-examples. Thus, the observed patterns support the linearity hypothesis.

In [23], the authors proposed a set of "Relative Landmarking (RL)" meta-features based on the pairwise performance difference of simple landmarking algorithms. This new data characterizations schema is used to train meta-learning based on the Active Testing (AT) algorithm. The patterns observed in Figure 6a follow the same principle, presenting a new alternative to characterize base-level datasets. Following this proposal, 10 new relative landmarking meta-features were generated based on five landmarking algorithms: kNN, NB, LR, SVM and Decision Stump (DS). These new meta-features are described in Table A.11 in Appendix Appendix A.

The same RF meta-analysis described in Section 5.3 was performed, adding the relative landmarking meta-features to the meta-datasets. These experiments pointed out how useful the new meta-features are for the recommendation problem. Figure 6 shows the relative importance values of the meta-features averaged in 30 executions. The relative importance of these new meta-features are highlighted in red, while the simple landmarking is shown in blue.

Two of the relative landmarking meta-features are placed in the top-10 most important meta-features: `RL.diff.nn.lm` ($1st$), and `RL.diff.svm.lm` ($3rd$); another two measures are in the top-20 - `RL.diff.stump.lm` and `RL.diff.stump.lm`; and all of them depend directly on the linear classifier performance. It is also important to mention that simple landmarking meta-features performed, in general, worse than relative landmarking. All these relative importance plots show evidence that the linearity hypothesis is true, and at least one characteristic that defines the need of HP tuning for SVMs is the linearity of the base-level classification task.

5.5. Overall comparison

Given the potential shown by the relative landmarking meta-features, they were experimentally evaluated in combination with the meta-features previously evaluated as most important. Complex network (`cnet`) meta-features were included because they were ranked between the most important descriptors (as shown in Subsection 5.3). Simple and data complexity (`complex`) meta-features were the other two approaches evaluated in the related studies listed in Section 3.5.

Figure 7 presents a comparison between the main experimental setups considering the addition of the relative landmarking (`relativeLand`) meta-features. The left chart of figure 7a shows AUC performance values obtained for each of the original setups, while the chart on the right presents setup performances when relative landmarking meta-features were included. This figure shows that the use of relative landmarking
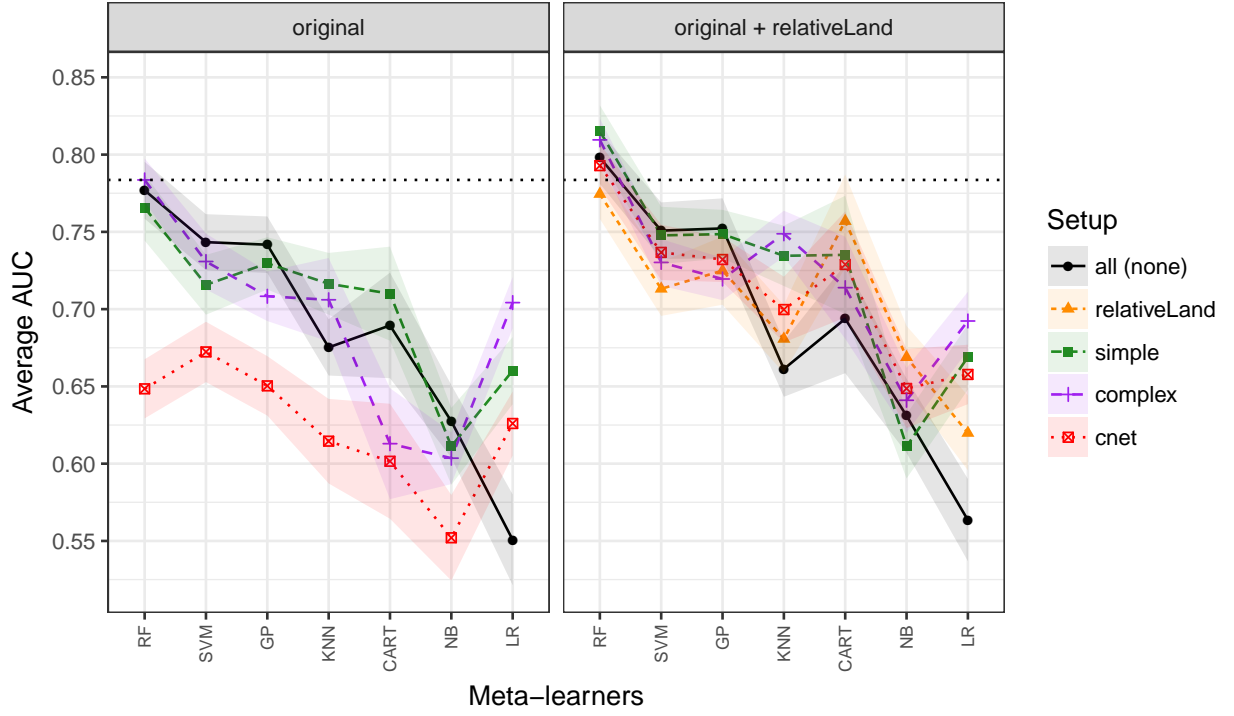
(a) Average AUC values when considering relative landmarking meta-features with all the other meta-features' categories.
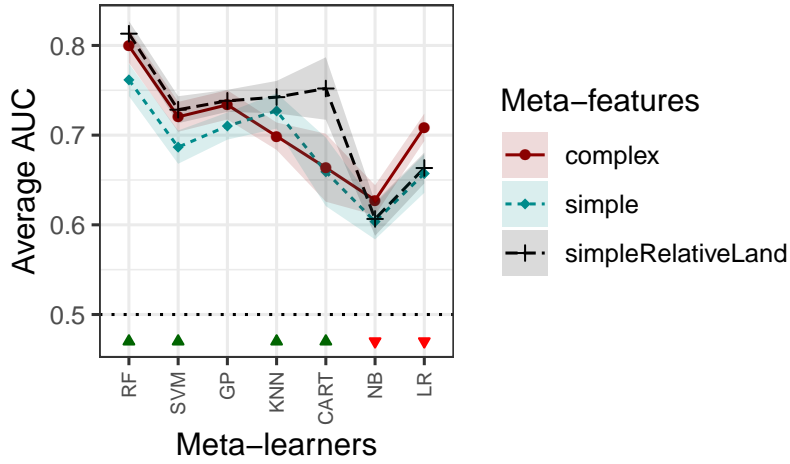
(b) Average AUC values for the best overall experimental setup with the simple and
the data complexity baselines presented in Section 5.1.

Figure 7: Evaluating the previous experimental setups adding relative landmarking meta-features. The results are the average of 30 runs.

meta-features improved all the setups where they were included. At least three different setups used by RF were higher than the AUC performance value obtained in the initial experiments. The setup considering simple and relative landmarking meta-features induced the best meta-models for RF, SVM and GPs. The

kNN and LR meta-learners obtained the best predictive performance using data complexity and relative landmarking meta-features and the same occurred for CART and NB for "relativeLand" set.

Figure 7b compares the best setup from Figure 7a: "`simpleRelativeLand`", which uses both simple and relative landmarking meta-features, with the the baselines from Figure 3a, using"`simple`" and data complexity ("`complex`") meta-features, often explored in related studies (see Table 1).

In this figure, the x-axis shows the different meta-learners, while the y-axis shows their predictive performance assessed by the AUC averaged over 30 repetitions. The Wilcoxon paired-test with $\alpha = 0.05$ was applied to assess the statistical significance of these results. An upward green triangle (▲) at the x-axis identifies situations where the use of "`simpleRelativeLand`" was statistically better than using the baselines. In the same figure, the red downward triangles (▼) indicate when the use of baselines was significantly better. In the remaining cases, the predictive performance of the meta-models were equivalents.

Overall, the meta-models induced with "`simpleRelativeLand`" meta-features were significantly better than those induced with baseline meta-features for most of the meta-learners: RF, SVM, kNN and CART obtained superior AUC values. Furthermore, the best meta-learner (RF) also significantly outperformed our previous results. The baselines produced the best meta-models for only two algorithms: NB and LR. For the GP algorithm, the different setups did not present any statistically significant difference.

5.6. Analysis of the predictions

A more in-depth analysis of the meta-learner predictions can help to understand their behavior. Figure 8 shows the misclassifications of the meta-learners considering their best experimental setups. The top chart (Fig. 8a) shows all the individual predictions, with the x-axis listing all the meta-examples and y-axis the meta-learners. In this figure, "`Defaults`" labels are shown in black and "`Tuning`" labels in gray. The top line in the 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 the SVM HP tuning recommendation task, "`Defaults`" is defined as the positive class and "`Tuning`" as the negative class. Therefore, a FN is a wrong recommendation to perform HP tuning on SVMs, and False Positive (FP) is a wrong recommendation to use default HP values. While a reduction in FN can decrease the computational cost, a reduction in FP can improve predictive performance.

Algorithms following different learning biases present different prediction patterns and this can be observed in Figure 8. Usually, most meta-examples are correctly classified (a better performance than the baselines). Besides, the following patterns can be observed:

- kNN and GP minimize the FN rate, correctly classifying most of the meta-examples as "`Defaults`". However, they misclassified many examples from the "`Tuning`" class, penalizing the overall performance of the recommender system;

30
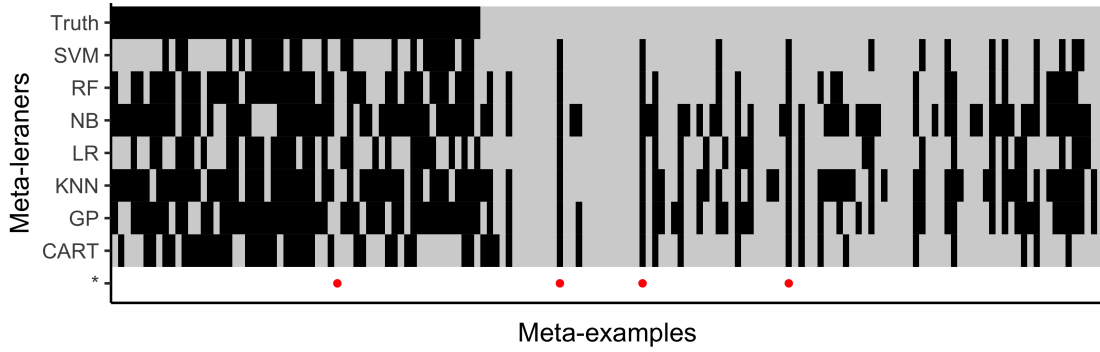
(a) Meta-learners' individual predictions.
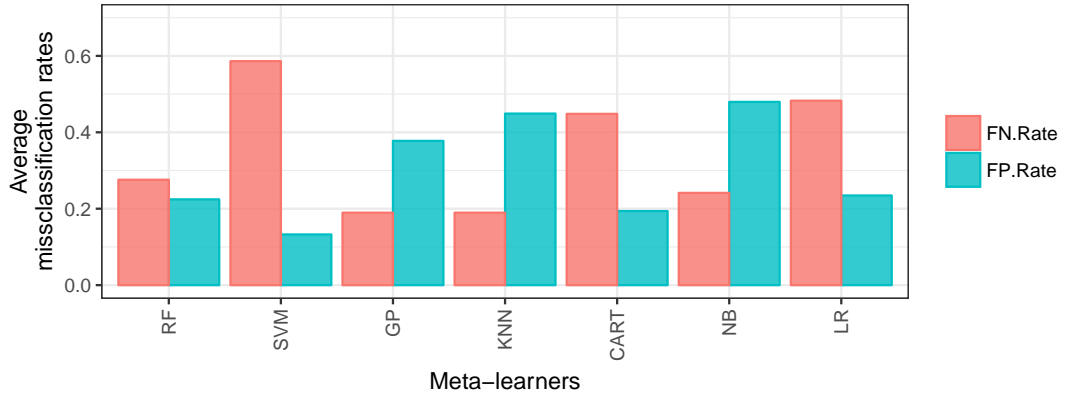
(b) Meta-learners' misclassification rates.

Figure 8: Meta-learners' predictions considering the experimental setups which obtained the best AUC values.

- SVM, CART and LR minimized the FP rate, correctly classifying most of the meta-examples requiring tuning. However, they tended to classify the meta-examples in the majority class;

A more balanced scenario is provided by the RF meta-models, which presented the best predictive performance. Although it was not the best algorithm for each class individually, it was the best when the two classes were considered.

Table 9: Misclassified datasets by all the meta-learners. 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 9 lists all the datasets misclassified by all the meta-learners as indicated in Figure 8a. Meta-examples with ids 17 ("Defaults") and 78 ("Tuning") were corrected labeled by the statistical meta-rule, and therefore 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 may indicate a limitation of the current meta-target rule criteria.

5.7. Projecting performances at base-level

This section assesses the impact of the choices made by the meta-learners at the base-level. It also analyses and discusses the reduction of runtime when using the proposed meta-learning recommender system. Figure 9 shows the predictive performance of SVMs at the base-level using the method ("Tuning' or "Defaults') selected by the meta-learners to define their HP values. The best meta-learners identified in the previous sections were compared with three simple baselines: a model that always recommends HP tuning (Tuning), a model that always recommends the use of defaults (Defaults), and a model that provides random recommendations (Random).

Figure 9a shows a scatter plot with the projected BAC performance and runtime value averaged in all the base-level datasets. Performing always the HP tuning had the highest average BAC value but was also the most expensive approach. On the other hand, always using default HP values is the fastest approach, but with the lowest average BAC value. The proposed meta-models are above Random and Defaults baselines, performing close to the Tuning baseline but with lower average runtime costs.

The Friedman test, with a significance level of $\alpha = 0.05$, was also used to assess the statistical significance of the base-level predictions. The null hypothesis is 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 applied to indicate when two different techniques are significantly different. Figure 9b presents the Critical Difference (CD) diagram. Techniques are connected when there is *no* significant differences between them.

Overall, the approach always using 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 HP tuning. In this figure, although the RF meta-model is considered the best, it was not ranked first. The first was the SVM meta-model. This occurred because it most often recommended the use of tuned settings, which was reflected in its performance at the base-level. With many datasets at the base-level, it can be pointed out that the overall gain is diluted between them. Even so, the meta-learners could considerably reduce the computational costs related to tuning, maintaining a high predictive performance.

Besides, it can be observed in Figure 9a that tuning SVM hyperparameters for just one dataset will take on average two days. The most costly datasets (with a high number of features or examples) took almost 10
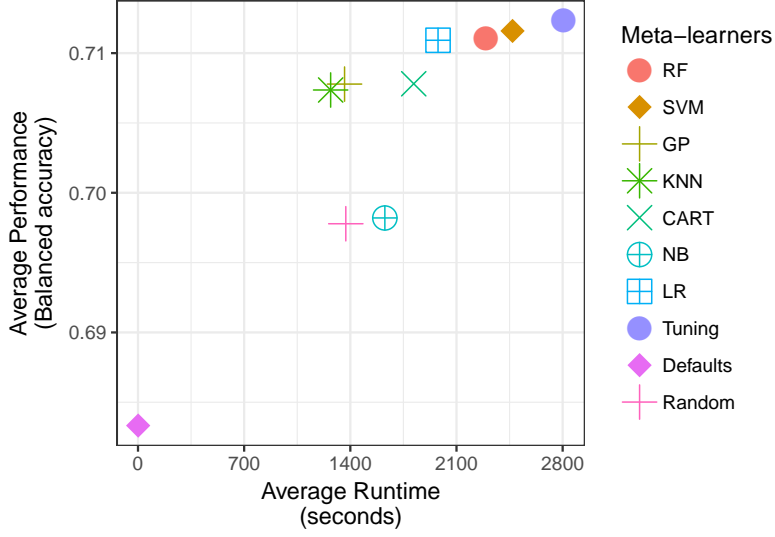
(a) Average BAC and runtime for the SVM base-level data.
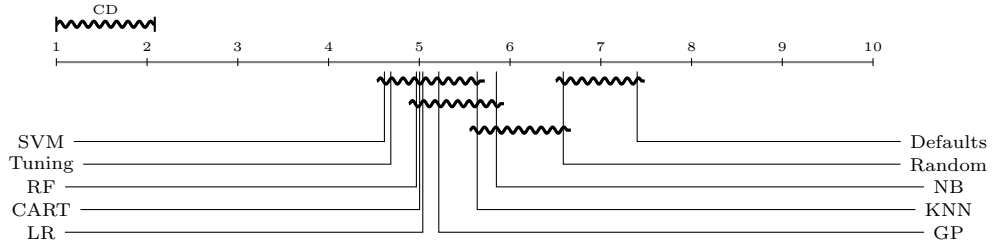
(b) CD diagram comparing the BAC values of the meta-learners at the base-level according to the Friedman-Nemenyi test ($\alpha = 0.05$).

Figure 9: Performance of the meta-learners projected into the SVM hyperparameter tuning problem (base-level).

days to finish all the 10 tuning repetitions (seeds) even paralleling the jobs in a high performance cluster. Regarding the meta-feature extraction, the same datasets will take at most 10 minutes, specially because of some mathematical operations used by Data Complexity (DC) meta-features. For most meta-features, the time taken to characterize a dataset is less than 30 seconds. Thus, during the prediction phase with the induced meta-model, the computational cost of extracting the characteristics of a new dataset is irrelevant compared to the computational cost of the tuning process. We think this is an important argument in favor of using our system, which is used in practical scenarios.

5.8. A note on the generalization

Although the main focus of the paper is to investigate the SVM hyperparameter tuning problem, we also conducted experiments for predicting the need for tuning Decision Trees (DTs). These experiments aim to provide more evidence that the proposed method can be generalized. Once the meta-knowledge is extracted, the system is able to induce meta-models to any supervised learning algorithm. In particular,

33

we investigated the HP tuning of the J48 algorithm, a WEKA[20] implementation for the Quinlan's C4.5 DT induction algorithm [37], and one of the most popular ML algorithms. The tree models were induced using the RWeka[21]. We also expanded the meta-knowledge by performing additional experiments to cover the same datasets explored with SVMs. In total, we obtained results performing the HP tuning of the J48 algorithm in 165 OpenML datasets. The tuning processes followed the experimental setup described in Table 4 with some differences:

- the J48 hyperparameter space has nine hyperparameters[22];

- a budget size of 900 evaluations was adopted in the experiments with trees. It is greater than for SVMs because of the greater search space of J48;

- the tuned hyperparameter results were compared with those obtained from the J48 RWeka/WEKA default settings.

Table C.14 in Appendix C presents the main characteristics of the meta-datasets generated with J48 tuning experiments. Class distribution columns (*Tuning* and *Default*) show values which indicate a different *hyperparameter profile*[23] compared to that observed in experiments with SVMs: most of the meta-examples are labelled as "Default", i.e., tuning did not statistically improve the algorithm performance in two thirds of the datasets. Here, we present results just on the meta-dataset using $\alpha = 0.05$ for the statistical labeling rule. However, results obtained with different α values were quite similar.

The predictive performance of the meta-learners considering different categories of meta-features are summarized in Figure 10a. The best results were obtained using the RF, SVM and GP meta-learners. They achieved their best predictive performances using preferably "all" the available meta-features, with AUC values between $AUC = (0.67, 0.72)$. It may be due to the fact that predictions tend toward a specific class (Defaults) since these meta-datasets are imbalanced. Overall, the best meta-model was obtained by the SVM algorithm. However, when considering all the possible scenarios (different α values), there was no statistical difference among the top ranked meta-learners: the RF, kNN, GP, SVM and LR algorithms (see Figure 10c). Furthermore, in general, the complete set of meta-features provided the best results for most algorithms.

Figure 10b shows the average AUC values considering different experimental setups. The NB and LR algorithms do not have any tunable hyperparameters. Consequently, their results for "tuned" setups are

---

[20]http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html

[21]https://cran.r-project.org/web/packages/RWeka/index.html package.

The meta-datasets for the J48 algorithm were generated based on HP tuning results obtained from 102 datasets reported in [26]

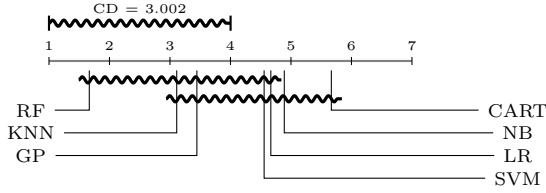[22]The J48 hyperparameter space is also presented in Appendix C.

[23]In this paper, the "*hyperparameter profile*" term refers to how sensitive an algorithm may be to the HP tuning task.
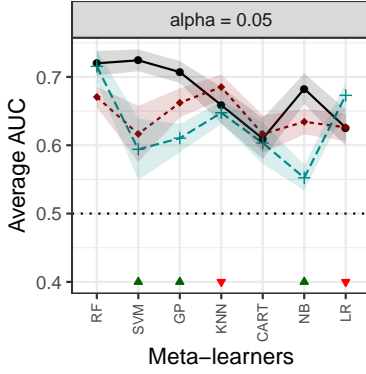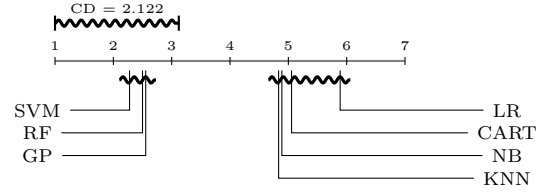
(a) AUC values considering different categories of meta-features.
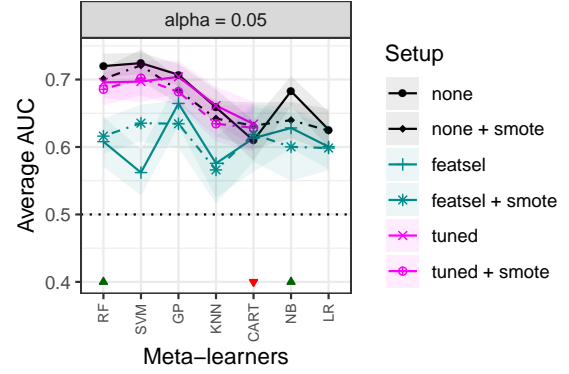
(b) AUC values considering different experimental setups.

(c) CD diagram comparing different meta-learners in different categories of meta-features.

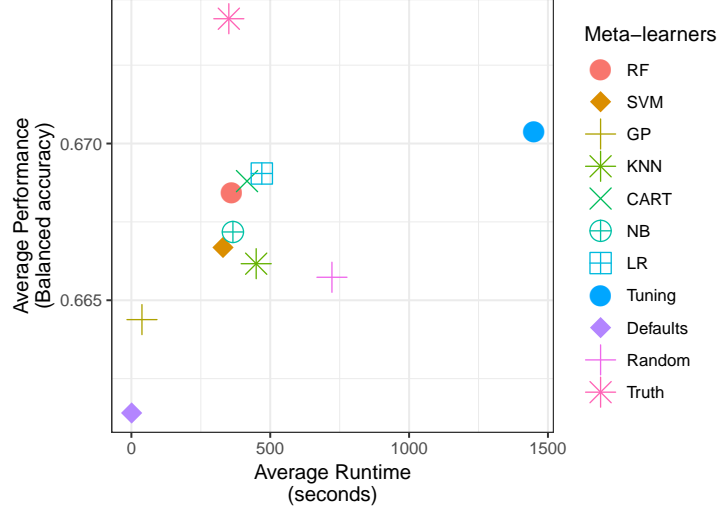(d) CD diagram comparing different meta-learners in different experimental setups).

Figure 10: Meta-learners average AUC results in the J48 meta-dataset labelled with $alpha = 0.05$, and CD diagrams comparing meta-learners according to the Friedman-Nemenyi test ($\alpha = 0.05$). Results are averaged in 30 runs.

missing in the chart. In general, when considering AUC values obtained in the original meta-dataset, improvements were obtained only for CART and kNN applying SMOTE. For the other algorithms, best meta-learners were still induced without any additional process. Statistical comparisons highlight the performance of the SVM, RF and GP algorithms (see Figure 10d).

We also evaluated the potentiality of the Relative Landmarking (RL) meta-features in the J48 tuning recommendation problem. However, differently than reported in Section 5.4, they worsened most of the meta-models. Linearity is not a key aspect in the J48 tuning problem, which further reinforces the results obtained from SVMs (see Section 5.4).

Reproducing the RF analysis in the J48 tuning problem (see Section 5.3), the most important meta-feature was the data complexity measure "`DC.f4`". This meta-feature describes the collective attribute efficiency in a dataset. The second was a simple meta-feature: "`SM.abs_cor`", a metric that measures the linear relationship between two attributes. This value is averaged in all pairs of attributes in the dataset. The top-3 is completed with another data complexity meta-feature: "`DC.f3`", which describes the maximum individual attribute efficiency. The two DC meta-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 meta-features suggest that if a dataset has representative attributes, default HP values are robust to solve it. Otherwise, the J48 tuning is recommended.

(a) Average BAC and runtime for the J48 base-level data.
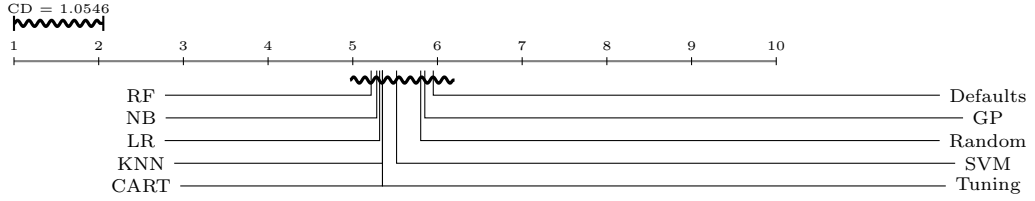
(b) CD diagram comparing the BAC values of the meta-learners at the base-level according to the Friedman-Nemenyi test ($\alpha = 0.05$).

Figure 11: Performance of the Meta-learners projected in the J48 hyperparameter tuning problem (base-level).

Figure 11 shows J48 meta-level predictions projected overall base-level datasets. Mostly, but not all of the meta-examples are labeled with "`Defaults`". The induced meta-models depicted in Figure 11a are above the "`Defaults`" baseline, but relatively close to the `Random` and `Tuning` ones. The average BAC values of all the approaches are very close, even considering all the baselines. This can be noted by the scale on the y-axis. It is explained by the small improvements obtained at the base-level tuning processes; they were relatively small if compared with those obtained with SVMs.

However, all the meta-models have a lower average runtime compared with `Random` and `Tuning` baselines. Most of the meta-models are better ranked than baselines, but, overall, there are no statistical differences among the evaluated approaches (Figure 11b). Even so, it is important to highlight that meta-learners could also considerably reduce the computational cost related to tuning, keeping the predictive performance in the dataset collection.

36

6. Conclusions

This paper proposed and experimentally investigated a MtL framework to predict when to perform SVM HP tuning. To do this, 156 different datasets publicly available at OpenML were used. The predictive performances of SVM induced with HP tuning (by a simple RS) and with default HP values were compared and used to design a recommender system. The default values were provided by the `e1071` R package), and by optimized common settings from [29]. Different experimental setups were analyzed with different sets of meta-features. The main findings are summarized next.

6.1. Tuning prediction

The main issue investigated in this paper was whether it is possible to accurately predict when HP tuning can improve the predictive performance of SVMs, when compared to the use of default HP values. If so, this can reduce the processing cost when applying SVMs to a new dataset. According to experimental results, using RF and SVM as meta-learners, this prediction is possible with a predictive accuracy of AUC=0.798.

Three significance levels ($\alpha = \{0.01, 0.05, 0.10\}$) were used with the Wilcoxon test to define the meta-target, which indicates whether it is better to use HP tuning or default HP values. Different sets of meta-features were evaluated, and RF meta-models using all the available meta-features obtained the best results, regardless the α value considered. However, the complex set of meta-features resulted in high predictive performance for most of the investigated meta-learners. Different experimental setups were also evaluated at the meta-level, but improvements were observed, and in a few cases, only when SMOTE or meta-feature selection were used. Thus, the best setup was to use raw meta-data and meta-learners with default HP values.

An analysis of the RF meta-models show that most meta-features actively contributed to predictions. This explains the decrease in performance when using meta-feature selection in most of the algorithms. Among the 10 meta-features ranked as most important, there are meta-features from different sets. Each ranked meta-feature describes different characteristics of the problem, such as data imbalance, linearity, and complexity.

This paper also investigated the hypothesis that the linear separability level could be an important meta-feature to be used by the recommender system. Meta-features based on relative landmarking were used to measure the linear separability degree. In the experiments performed, this meta-feature was shown to play an important role in the recommender system prediction. Three meta-learners had their best AUC performance values using a combination of simple and relative landmarking meta-features.

Using two different default HP settings maximized the number of default wins, reducing the imbalance rate at the meta-datasets. In addition to presenting the best predictive performance, RFs, by the frequency of meta-features in their trees, provided useful information regarding when default settings are suitable.

We also performed experiments for the J48 tuning recommendation problem aiming to show the ability of generalization of the MtL recommender system. Results showed that different to SVMs, where the linearity was important to recommend the use of default settings, the most important meta-features suggest that if a dataset has representative attributes, default settings are robust to solve it.

In fact, our extensive experiments suggest that the guideline depends on the algorithm used to induce the meta-model. If we use a white box algorithm, such as RF, we can use the meta-features in the root of the trees (and nodes close to the root) to explain when to tune. The high predictive performance of the RF algorithm indicates that the induced models were able to find a good hypothesis for situations where tuning is necessary, in both cases (for SVMs and J48).

6.2. Linking findings with the literature

Two of the related studies in the literature [28, 41] used meta-models based on decision trees to interpret their predictions. However, in the current results, CART trees were among the worst meta-learners considering all the experimental setups analyzed. Thus, in this study, the meta-analysis performed was based on the RF meta-learner, extracting the average of the Gini index from the meta-features provided by the inner RF meta-models.

Meta-feature selection was also evaluated in [41]. The authors explored a Correlation-based feature selection (CFS) method and reported the "nn" meta-feature[24] as the most important. However, the results reported here were not improved by meta-feature selection. In fact, it decreased the performance of most meta-models, as shown in Figure 3(a). Meta-feature selection was also tried with filter methods, but the results were even worse than using SFS. For this reason, it was not reported in this study. In addition, "nn" meta-features did not appear among the top-20 most important meta-features computed by RF.

Our experimental results show that using meta-features from different categories have improved the predictive performance of the meta-learners for different setups. The most important meta-features were "SM.classes_min" and "LM.stump_sd"[25]. In [28], which only used meta-features from simple and data complexity sets, "SM.classes_max" and "SM.attributes" were reported as the most important meta-features. The first describes the percentage of examples in the majority class. The second is the number of predictive attributes in a dataset. Although these studies disagree about the relative importance order of these meta-features, both extract information related to the same characteristics: data complexity and dimensionality.

The HP tuning investigated in [42] *"assumed that tuning is always necessary"*, and therefore focused on the improvement prediction as a regression task. They obtained HP tuning results for less than half of the

---

[24]The performance of 1-NN algorithm. See tables A.10 and A.11 in Appendix Appendix A.

[25]This is shown in Figure 5 in Subsection 5.3.

datasets (111/229) in the base-level when dealing with SVMs. In addition, their meta-models were not able to predict the HP improvement for SVMs, not providing any valid conclusion about the problem.

6.3. Main difficulties

During the experiments, there were several difficulties to generate the meta-knowledge. The process itself is computationally expensive, since a lot of tuning tasks must be run and evaluated in a wide range of classification tasks. Initially, a larger number of datasets were selected, but some of them were extremely expensive computationally speaking for either HP tuning or the extraction of meta-features. A *walltime* of 100 hours was considered to remove high-cost datasets.

The class imbalance at the meta-level was another problem faced in the experiments. To deal with this problem, the optimized default HP settings [29] were added to the meta-dataset, increasing the number of meta-examples in the default meta-target. Besides the addition of relative landmarking set, some meta-examples were never correctly classified. This points out the need to define specific meta-features for some MtL problems.

6.4. Future work

Some findings from this study also open up future research directions. The proposed MtL recommender system could be extended to different ML algorithms, such as neural networks, another decision tree induction algorithms and ensemble-based techniques. It could also be used to support HP tuning decision in different tasks, such as pre-processing, regression and clustering. It could even be used to define whether to tune HP for more than one task, in a pipeline or simultaneously.

It would also be a promising direction to investigate the need of new meta-features to characterize data when dealing with data quality problems, for instance imbalancing measures, due to their influence in the quality of the induced meta-models. Besides, a multicriteria objective function could replace the current meta-label rule, weighing predictive performance, memory, and runtime. Another possibility would be to explore the use of ensembles as meta-models, given the complementary behavior of some of the algorithms studied here as meta-learners.

The code used in this study is publicly available, easily extendable and may be adapted to cover several other ML algorithms. The same may be said of the analysis, also available for reproducibility. All the experimental results generated are also available at OpenML correspondent studies web pages, from which they can be integrated and reused in different MtL systems. This framework is expected to be integrated to OpenML, so that the scientific community can use it.

References

References

[1] Ali, S., Smith-Miles, K.A.. A meta-learning approach to automatic kernel selection for support vector machines. Neurocomputing 2006;70(13):173–186.

[2] 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. 2000. p. 109–118.

[3] Bergstra, J., Bengio, Y.. Random search for hyper-parameter optimization. J Mach Learn Res 2012;13:281–305.

[4] 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 2016;17(170):1–5.

[5] Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.. Metalearning: Applications to Data Mining. 2nd ed. Springer Verlag, 2009.

[6] Brazdil, P.B., Henery, R.J.. Analysis of results. In: Michie, D., Spiegelhalter, D.J., Taylor, C.C., Campbell, J., editors. Machine learning, neural and statistical classification. Ellis Horwood; 1994. p. 175–212.

[7] Breiman, L.. Random forests. Machine Learning 2001;45(1):5–32.

[8] 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. IEEE Computer Society; 2010. p. 3121–3124.

[9] 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 2017;:1–15.

[10] Chang, C.C., Lin, C.J.. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2011;2:27:1–27:27.

[11] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.. SMOTE: synthetic minority over-sampling technique. J Artif Int Res 2002;16(1):321–357.

[12] Demšar, J.. Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research 2006;7:1–30.

[13] Eggensperger, K., Lindauer, M., Hoos, H.H., Hutter, F., Leyton-Brown, K.. Efficient benchmarking of algorithm configurators via model-based surrogates. Machine Learning 2018;107(1):15–41.

[14] 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., editors. Advances in Neural Information Processing Systems 28. Curran Associates, Inc.; 2015. p. 2944–2952.

[15] Garcia, L.P.F., de Carvalho, A.C.P.L.F., Lorena, A.C.. Noise detection in the meta-learning level. Neurocomputing 2016;176:14–25.

[16] Gomes, T.A.F., Prudêncio, R.B.C., Soares, C., Rossi, A.L.D., nd André C. P. L. F. Carvalho, . Combining meta-learning and search techniques to select parameters for support vector machines. Neurocomputing 2012;75(1):3–13.

[17] Ho, T.K., Basu, M.. Complexity measures of supervised classification problems. Pattern Analysis and Machine Intelligence, IEEE Transactions on 2002;24(3):289–300.

[18] 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 2016;:1–17.

[19] Hsu, C.W., Chang, C.C., Lin, C.J.. A Practical Guide to Support Vector Classification. Department of Computer Science - National Taiwan University; Taipei, Taiwan; 2007. .

[20] Hutter, F., Hoos, H., Leyton-Brown, K., Stützle, T.. Paramils: an automatic algorithm configuration framework. Journal of Artificial Intelligence Research 2009;(36):267–306.

[21] 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 2016;17:1–5.

[22] Krstajic, D., Buturovic, L.J., Leahy, D.E., Thomas, S.. Cross-validation pitfalls when selecting and assessing regression and classification models. Journal of cheminformatics 2014;6(1):10+.

[23] 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). 2012. p. 117–131.

[24] 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 2016;3:43 – 58.

[25] Lorena, A.C., Maciel, A.I., de Miranda, P.B.C., Costa, I.G., Prudêncio, R.B.C.. Data complexity meta-features for regression problems. Machine Learning 2018;107(1):209–246.

[26] Mantovani, R.G., Horváth, T., Cerri, R., Vanschoren, J., de Carvalho, A.C.P.L.F.. 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. URL: http://dx.doi.org/10.1109/BRACIS.2016.018. doi:10.1109/BRACIS.2016.018.

[27] Mantovani, R.G., Rossi, A.L.D., Vanschoren, J., Bischl, B., de Carvalho, A.C.P.L.F.. 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.

[28] 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.

[29] 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., editors. 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; volume 1455 of *CEUR Workshop Proceedings*; 2015. p. 80–92.

[30] Miranda, P., Prudêncio, R.. Active testing for SVM parameter selection. In: Neural Networks (IJCNN), The 2013 International Joint Conference on. 2013. p. 1–8.

[31] 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. 2014. p. 325–330.

[32] 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 2012;7335 LNCS(PART 3):562–575.

[33] 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.

[34] Padierna, L.C., Carpio, M., Rojas, A., Puga, H., Baltazar, R., Fraire, H.. Hyper-Parameter Tuning for Support Vector Machines by Estimation of Distribution Algorithms; Cham: Springer International Publishing. p. 787–800.

[35] 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.

[36] 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). volume 7208 LNAI; 2012. p. 196–207.

[37] Quinlan, J.R.. C4.5: Programs for Machine Learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[38] Reif, M., Shafait, F., Dengel, A.. Prediction of classifier training time including parameter optimization. In: Proceedings of the 34th Annual German conference on Advances in artificial intelligence. Springer-Verlag; KI'11; 2011. p. 260–271.

[39] Reif, M., Shafait, F., Dengel, A.. Meta-learning for evolutionary parameter optimization of classifiers. Machine Learning 2012;87:357–380.

[40] Reif, M., Shafait, F., Goldstein, M., Breuel, T., Dengel, A.. Automatic classifier selection for non-experts. Pattern Analysis and Applications 2014;17(1):83–96.

[41] 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., editors. Meta-learning and Algorithm Selection Workshop at ECAI 2014. 2014. p. 18–23.

[42] Sanders, S., Giraud-Carrier, C.G.. Informing the use of hyperparameter optimization through metalearning. In: Raghavan, V., Aluru, S., Karypis, G., Miele, L., Wu, X., editors. 2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017. IEEE Computer Society; 2017. p. 1051–1056.

[43] Snoek, J., Larochelle, H., Adams, R.P.. Practical bayesian optimization of machine learning algorithms. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K., editors. Advances in Neural Information Processing Systems 25. Curran Associates, Inc.; 2012. p. 2951–2959.

[44] 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. ACM Press; SAC'06; 2006. p. 564–568.

[45] Soares, C., Brazdil, P.B., Kuba, P.. A meta-learning method to select the kernel width in support vector regression. Machine Learning 2004;54(3):195–209.

[46] 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. 2013. p. 847–855.

[47] Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.. Openml: Networked science in machine learning. SIGKDD Explor Newsl 2014;15(2):49–60.

[48] Vapnik, V.. The Nature of Statistical Learning Theory. Springer-Verlag, 1995.

[49] Vilalta, R., Giraud-Carrier, C.G., Brazdil, P., Soares, C.. Using meta-learning to support data mining. International Journal of Computer Science & Applications 2004;1(1):31–45.

[50] Wistuba, M., Schilling, N., Schmidt-Thieme, L.. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. Machine Learning 2018;107(1):43–78.

Appendix A. List of Meta-features used in experiments

Table A.10: Meta-features used in experiments - part 1. For each meta-features it is shown: its type, acronym and description. Extended from [15].

Type	Acronym	Description
Simple (SM)	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 (ST)	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 (IN)	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 (MB) (Trees)	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 (LM)	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 A.11: Meta-features used in experiments - part 2. For each meta-features it is shown: its type, acronym and description. Extended from [15].

Type	Acronym	Description
Data Complexity (DC)	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
Complex Network (CN)	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 (RL)	diff.svm.lm	performance(SVM) - performance(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 B. Hyperparameter space of the meta-learners used in experiments

Table B.12: Meta-learner's hyperparameter 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
	cp	complexity parameter	$(0.0001, 0.1)$	real	0.01	
	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	rpart
	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
	k	kernel	Gaussian	-	-	
	C	regularized constant	$[2^{-15}, 2^{15}]$	real	1	
SVM	γ	width of the Gaussian kernel	$[2^{-15}, 2^{15}]$	real	$^1/_N$	e1071
RF	ntree	number of trees	$[2^0, 2^{10}]$	integer	500	
	nodesize	minimum node size of the decision trees	$\{1, 20\}$	integer	1	randomForest
KNN	k	number of nearest neighbors	$\{1, 50\}$	integer	7	kknn
NB	-	-	-	-	-	e1071
LR	-	-	-	-	-	gbm

Appendix C. J48 hyperparameter space and meta-datasets used in experiments from Section 5.8

Table C.13: J48 hyperparameter space explored in experiments. The nomenclature is based on the `RWeka` package. Table adapted from [26].

Symbol	Hyperparameter	Range	Type	Default	Conditions
C	pruning confidence	$(0.001, 0.5)$	real	0.25	R = False
M	minimum number of instances in a leaf	$[1, 50]$	integer	2	-
N	number of folds for reduced error pruning	$[2, 10]$	integer	3	R = True
O	do not collapse the tree	{False, True}	logical	False	-
R	use reduced error pruning	{False, True}	logical	False	-
B	use binary splits only	{False, True}	logical	False	-
S	do not perform subtree raising	{False, True}	logical	False	-
A	Laplace smoothing for predicted probabilities	{False, True}	logical	False	-
J	do not use MDL correction for info gain on numeric attributes	{False, True}	logical	False	-

Table C.14: Meta-datasets generated for J48 experiments.

Meta-dataset	α	Meta examples	Meta features	Class Distribution Tuning	Class Distribution Default
J48_90	0.10	165	80	63	102
J48_95	0.05	165	80	57	108
J48_99	0.01	165	80	52	113

Appendix D. List of abbreviations used in the paper

Acc Accuracy.

AT Active Testing.

AUC Area Under the ROC curve.

AutoML Automated Machine Learning.

BAC Balanced per class Accuracy.

CART Classification and Regression Tree.

CASH Combined Algorithm Selection and Hyperparameter Optimization.

CD Critical Difference.

CFS Correlation-based feature selection.

CN Complex Network.

CV Cross-validation.

DC Data Complexity.

DS Decision Stump.

DT Decision Tree.

EDA Estimation of Distribution Algorithm.

FN False Negative.

FP False Positive.

GA Genetic Algorithm.

GP Gaussian Process.

GS Grid Search.

HP Hyperparameter.

IN Information-theoretic.

Irace Iterated F-race.

kNN k-Nearest Neighbors.

LM Landmarking.

LR Logistic Regression.

MAD Mean Absolute Deviation.

MB Model-based.

ML Machine Learning.

MLP Multilayer Perceptron.

MtL Meta-learning.

NAE Normalized Absolute Error.

NB Naïve-Bayes.

NMSE Normalized Mean Squared Error.

OpenML Open Machine Learning.

ParamILS Iterated Local Search in Parameter Configuration Space.

PMCC Pearson Product-Moment Correlation Coefficient.

PSO Particle Swarm Optimization.

RBF Radial Basis Function.

RF Random Forest.

RL Relative Landmarking.

ROAR Random Online Adaptive Racing.

RS Random Search.

SFS Sequential Forward Selection.

SM Simple.

SMBO Sequential Model-based Optimization.

SMOTE Synthetic Minority Over-sampling Technique.

SpCorr Spearman Correlation.

ST Statistical.

SVM Support Vector Machine.

TAF Transfer Acquisition Function.

TS Tabu Search.