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

LUCAS BENEVIDES VIANA DE AMORIM

Meta-scaler+: a meta-learning based solution for model-specific recommendations of scaling techniques

Recife

2025

LUCAS BENEVIDES VIANA DE AMORIM

Meta-scaler+: a meta-learning based solution for model-specific recommendations of scaling techniques

Thesis presented to the Post-graduation Program in Computer Science - PPGCC of the Centro de Informática of the Universidade Federal de Pernambuco, as a partial requirement for obtaining the title of Doctor in Computer Science.

Concentration Area: Computer Science

Supervisor: George Darmiton da Cunha Cavalcanti

Co-supervisor: Rafael Menelau Oliveira e Cruz

Recife

2025

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Amorim, Lucas Benevides Viana de.
Meta-scaler+: a meta-learning based solution for model-specific recommendations of scaling techniques / Lucas Benevides Viana de Amorim. - Recife, 2025.
137f.: il.

Tese (Doutorado) - - Universidade Federal de Pernambuco, Centro de Informática, Programa de Pós-Graduação em Ciências da Computação, 2025.
Orientação: George Darmiton da Cunha Cavalcanti.
Coorientação: Rafael Menelau Oliveira e Cruz.
Inclui referências e apêndice.

1. Classificação; 2. Scaling; 3. Meta-aprendizagem; 4. Normalização; 5. Preprocessamento. I. George Darmiton da Cunha Cavalcanti, Rafael Menelau Oliveira e Cruz. II. Título.

UFPE-Biblioteca Central

Lucas Benevides Viana de Amorim

“Meta-scaler+: a meta-learning based solution for model-specific recommendations of scaling techniques”

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

Aprovada em: 18/02/2025.

Orientador: Prof. Dr. George Darmiton da Cunha Cavalcanti

BANCA EXAMINADORA

Prof. Dr. Francisco de Assis Tenório de Carvalho - Centro de Informática/UFPE

Prof. Dr. Ricardo Bastos Cavalcante Prudêncio - Centro de Informática / UFPE

Prof. Dr. Luís Paulo Faina Garcia
Departamento de Ciência da Computação/UnB

Prof. Dr. Luiz Eduardo Soares de Oliveira
Departamento de Informática / UFPR

Prof. Dr. Carlos Manuel Milheiro de Oliveira Pinto Soares
Faculdade de Engenharia / Universidade do Porto

Às minhas filhas, Luana e Clara.

AGRADECIMENTOS

Agradeço a todos os mestres que me guiaram por essa jornada, desde a pré-escola até aqui, o doutorado. Todos foram importantes, todos têm uma parcela de participação neste trabalho. Mas faço agradecimento nominal aos meus orientadores, tanto aos atuais, Prof. George Darmiton e Prof. Rafael Menelau, quanto ao meu orientador de graduação e mestrado, Prof. Evandro de Barros Costa. Eu não tenho palavras para expressar o quanto foi enriquecedora essa jornada, o quanto eu pude aprender com suas orientações. Agradeço pela paciência, pela parceria, e pela compreensão em todos os momentos, mas, acima de tudo, agradeço por terem ajudado a me proporcionar essa grande oportunidade aprendizado e enriquecimento profissional e pessoal.

Agradeço ao Centro de Informática da UFPE e à CAPES que através do programa PRInt me oportunizaram o intercâmbio no Canadá, etapa importantíssima desse trabalho. Agradeço ao Instituto de Computação, à UFAL, e aos colegas de trabalho que me apoiaram, acreditaram no meu potencial e me possibilitaram esse longo período de dedicação exclusiva ao doutorado. Devolverei à altura. Por fim, agradeço aos amigos e à minha família pelo apoio e pela compreensão, pela energia e a motivação.

PREFACE

This thesis document is composed of a sequence of three research papers that were written during the period of this doctorate and present the most important results of this doctorate research. These three papers follow a logical sequence, as explained in this preface. We begin this document with a general introduction that covers the whole thesis (Chapter 1), then we present the three papers (Chapters 2, 3, and a general conclusion (Chapter 5). In a regular thesis, a "Background" chapter would be expected right after the introduction, but being this a collection of papers, we decided to present the background on demand within each paper, which also avoids redundancy.

In the following paragraphs, we will provide an informal contextualization and explanation of the decisions that made us follow this research direction. As part of a larger investment from the research group in AutoML methods, the general idea behind this thesis was to propose a meta-learning-based framework that can adequately recommend a particular preprocessing step for a given dataset. To do that, we needed to focus on a preprocessing step that was of fundamental importance to machine learning pipelines but had enough variability in terms of possible techniques to make it hard for machine learning practitioners to select it. We restricted ourselves to classification pipelines since this is one of the most commonly used machine learning tasks.

After analyzing the machine learning literature, we selected dataset scaling as our focus. It is both ubiquitous, being almost always applied in classification pipelines, and neglected, since most machine learning practitioners just select a "default" scaling technique, among several possibilities, without methodological rigor. Albeit its importance to classification performance seeming to be commonsense in the area, we needed to know if scaling was, in fact, important and if adequately choosing it could provide significant classification performance improvements. Therefore, we designed a first study to compare the effects of various scaling techniques on the performance of several classification algorithms. It culminated in the paper *The choice of scaling technique matters for classification performance*, published in January 2023 in the Applied Soft Computing journal. With an extensive empirical analysis and a good foundation and discussion on how these scaling techniques work, it was very well received in the area of machine learning with more than 100[1] citations in the last year alone. This paper is presented

[1] According to Google Scholar.

in its entirety in Chapter 2.

That first paper demonstrated that scaling is, in fact, important and that the scaling technique must be carefully selected, as inadequate scaling can be worse than not scaling the dataset at all. Furthermore, it also showed that, given a dataset, different classifiers may prefer different scaling techniques to achieve optimal performance. Therefore, a challenge was revealed: how can we effectively and efficiently select a scaling technique for a dataset while also taking the classifier into account? We came up with a meta-learning-based solution called Meta-scaler. It trains specialized meta-models, one for each different type of query classifier. With that, it recommends a scaling technique according to query dataset meta-features while also taking the query classifier into account. We published the idea in our second research paper *Meta-Scaler: a meta-learning framework for the selection of scaling techniques* in February 2024 in the IEEE Transactions on Neural Networks and Learning Systems. In this paper, Chapter 3 of this thesis, we presented the Meta-scaler framework and compared it to the state of the art. It significantly outperformed any fixed choice of scaling technique and the state-of-the-art meta-learning methods for scaling technique selection. Despite its advancements, it has one major limitation: it only works when the query classifier is known, i.e., when it has a meta-model previously trained for that specific classifier. The specialization feature had become both the Meta-scaler's strength and its crux.

Employing specialized meta-models is part of what makes this method efficient and innovative, and we could not let go of this feature. Therefore, for our third and still unpublished research paper, presented in Chapter 4, we worked on a way of giving specialized recommendations even when the query classifier is not known. For that, we had to find a way to at least extract some information, in a cost-effective way, from the query classifier to be able to specialize our recommendation to it. This led us to the idea of a Classifier Performance Space. An innovative method of classifier characterization that can obtain meta-information about the classifier regardless of its type. With the Classifier's Space, we can compare query classifiers to the ones we know and dynamically combine meta-models' decisions to provide dataset- and classifier-dependent recommendations for any given query dataset and classifier. We called this new framework the Meta-scaler+. It builds upon the previous iteration of the Meta-scaler, eliminating its main limitations and providing other features, such as the ability to recommend a ranking of the scaling techniques. It is able to provide high-quality recommendations even for unknown query classifiers, as its performance is almost as good as the one Meta-scaler obtains for known query classifiers and superior to other state-of-the-art methods.

While Meta-scaler+ works as a closure for this thesis, it is important to note that many other ideas were also explored during the development of this thesis; some were not successful **yet** and did not make it into this document, such as the idea of a neural network for better dataset characterization. In contrast, others were very successful and can spin off into full-fledged research projects in the future, such as the Classifier Performance Space. Several ideas for improving the Meta-scaler+ are also being considered for future work; some were discussed in the papers, and some are presented in the conclusion of this thesis. We see this thesis as the beginning of a promising Meta-learning and AutoML research route. The ideas and the experience with Meta-scaler and Meta-scaler+ can be later transferred to other pipeline blocks and ultimately provide the building blocks of full AutoML systems.

RESUMO

A normalização (scaling) de conjuntos de dados é uma etapa essencial de pré-processamento em um pipeline de aprendizado de máquina. Ela visa ajustar as escalas de atributos de forma que todos variem dentro do mesmo intervalo. Essa transformação é amplamente reconhecida como necessária para melhorar o desempenho dos modelos de classificação, mas muito poucos estudos verificam empiricamente essa relação. Como primeira contribuição, esta tese compara os impactos de diferentes técnicas de scaling (STs) no desempenho de vários classificadores. Seus resultados mostram que a escolha da técnica de scaling importa para o desempenho da classificação, e a diferença de desempenho entre a melhor e a pior técnica é relevante e estatisticamente significativa na maioria dos casos. No entanto, há várias STs para escolher, e o processo de encontrar manualmente, por tentativa e erro, a técnica mais adequada para um determinado conjunto de dados pode ser inviável. Como alternativa, propomos empregar meta-aprendizagem para selecionar automaticamente a melhor ST para um determinado conjunto de dados. Portanto, em nosso segundo estudo, propomos o Meta-scaler, um framework que aprende e treina um conjunto de meta-modelos para representar a relação entre meta-características extraídas dos conjuntos de dados e o desempenho de um conjunto de algoritmos de classificação nesses conjuntos de dados quando eles são normalizados com diferentes técnicas. Esses meta-modelos são capazes de recomendar uma única ST ótima para um determinado conjunto de dados de consulta, levando em consideração também o classificador de consulta. O Meta-scaler produziu melhor desempenho de classificação do que qualquer escolha de uma única ST para 10 dos 12 modelos base testados e também superou os métodos de meta-aprendizagem do estado da arte para seleção de ST. Finalmente, em nosso terceiro estudo, propomos o Meta-scaler+, onde estendemos a funcionalidade do Meta-scaler, eliminando suas limitações ao introduzir um método inovador de caracterização de classificadores, o Classifier Performance Space, que nos permite combinar dinamicamente meta-modelos para recomendações especializadas de ST para qualquer classificador e conjunto de dados. Apesar da flexibilidade adicional, o desempenho do Meta-scaler+ é competitivo com o Meta-scaler e superior a outras soluções do estado da arte. Para as próximas etapas do desenvolvimento desta pesquisa, investiremos na melhoria da representação do conjunto de dados (meta-recursos), melhorando a inicialização do Classifier Performance Space e tornando o Meta-scaler+ uma ferramenta prática e acessível, permitindo sua integração com bibliotecas populares de aprendizado de máquina.

Palavras-chaves: Classificação, Preprocessamento, Scaling, Normalização, Meta-aprendizagem, AutoML.

ABSTRACT

Dataset scaling, or normalization, is an essential preprocessing step in a machine learning pipeline. It adjusts attributes' scales in a way that they all vary within the same range. This transformation is widely assumed to improve the performance of classification models, but very few studies empirically verify this assumption. As a first contribution, this thesis compares the impacts of different scaling techniques (STs) on the performance of several classifiers. Its results show that the choice of scaling technique matters for classification performance, and the performance difference between the best and the worst scaling technique is relevant and statistically significant in most cases. However, there are several STs to choose from, and the process of manually finding, via trial and error, the most suitable technique for a certain dataset can be unfeasible. As an alternative to this, we propose employing meta-learning to select the best ST for a given dataset automatically. Therefore, in our second study, we propose the Meta-scaler, a framework that learns and trains a set of meta-models to represent the relationship between meta-features extracted from the datasets and the performance of a set of classification algorithms on these datasets when they are scaled with different techniques. These meta-models are able to recommend a single optimal ST for a given query dataset, taking into account also the query classifier. The Meta-scaler yielded better classification performance than any choice of a single ST for 10 out of the 12 base models tested and also outperformed state-of-the-art meta-learning methods for ST selection. Then, in our third study, we proposed Meta-scaler+, where we extended the functionality of Meta-scaler, eliminating its limitations by introducing an innovative classifier characterization method, the Classifier Performance Space, which allows us to dynamically combine meta-models for specialized ST recommendations for any query classifier and query dataset. Despite the additional flexibility, Meta-scaler+ performance is competitive with Meta-scaler and superior to other state-of-the-art solutions. In future work, we will invest in improving dataset representation (meta-features), improving Classifier Performance Space initialization, and making Meta-scaler+ a practical and accessible tool, enabling its integration with popular machine-learning libraries.

Keywords: Classification. Preprocessing. Normalization. Scaling. Meta-learning. AutoML.

LIST OF FIGURES

LIST OF TABLES

SUMMARY

1 INTRODUCTION

In machine learning (ML), preprocessing steps are crucial for enhancing the performance of classification models, with dataset scaling being one of the most commonly applied techniques. Scaling transforms numerical features to a similar range, preventing features with larger values from dominating the model's learning process. This step has been widely accepted as essential for improving classification accuracy, and most researchers apply this scaling by default to their analysis pipelines. Despite its ubiquity, the question of how scaling impacts model performance — and how to choose the most appropriate scaling technique — remains a challenge.

In the first of three studies in this thesis, we perform a comprehensive experiment comparing various scaling techniques (STs) in the context of both monolithic and ensemble classifiers. Our results provide strong evidence that not only does scaling improve performance, but the choice of technique can significantly alter classification results. By considering a broad set of scaling methods and classification algorithms, we reveal how performance variations are influenced by both the scaling technique and the dataset's characteristics. These findings provide a methodological framework for understanding the impact of dataset scaling, urging the research community to give more attention to this ubiquitous preprocessing step.

Despite the demonstrated importance of scaling, selecting the optimal ST for a given dataset remains an open problem. This choice is typically made through trial and error or by relying on "default" techniques, both of which can be inefficient and suboptimal, especially when we consider that making a bad choice of ST can even decrease classification performance when compared to nonscaled data (3, 4). To address this issue, the second study in this thesis introduces Meta-scaler, a meta-learning (MtL) framework designed to automate ST selection. Meta-scaler builds on insights from the first paper and uses dataset meta-features to train classifier-specific meta-models to predict the most suitable scaling technique. By analyzing a variety of dataset characteristics (e.g., size, feature distribution) and classifiers' performances on these datasets, Meta-scaler generates recommendations tailored to both the dataset and the classifier in use. Our experiments show that Meta-scaler consistently outperforms default ST selection methods and state-of-the-art meta-learning approaches, offering improved classification accuracy across a wide range of datasets and models.

However, the original Meta-scaler faces a key limitation: it is restricted to recommending scaling techniques for a fixed set of classifiers. This constraint limits its applicability, as users are

confined to classifiers included during the meta-models training. To address this limitation, in our third study, we propose Meta-scaler+, an extension of the original framework that introduces the concept of a Classifier Performance Space. This space allows Meta-scaler+ to characterize classifiers based on their performance on a small but representative set of datasets, enabling the system to recommend appropriate scaling techniques for any classifier, even those not included in the original training set. By dynamically combining meta-models trained for similar classifiers according to their distances in the Classifier Performance Space, Meta-scaler+ provides more flexible recommendations than the original Meta-scaler, offering substantial performance improvement over generalist meta-model approaches.

The main contributions of this thesis are as follows:

1. **Empirical Evidence on Scaling and Classification Performance:** The first contribution of this thesis provides strong, empirical evidence that scaling techniques significantly impact classification performance. We conduct an extensive analysis across various datasets and classification algorithms, demonstrating that the choice of ST can lead to significant changes in classification performance.

2. **Meta-scaler Framework:** We propose Meta-scaler, a specialized meta-learning framework for selecting the best scaling technique for a given dataset-classifier pair. Meta-scaler outperforms traditional static scaling methods and other state-of-the-art meta-learning methods, providing more accurate and tailored scaling recommendations that improve base-level classification performance.

3. **Meta-scaler+ and the Classifier Performance Space:** We extend Meta-scaler by introducing the Classifier Performance Space, a novel method for characterizing classifiers. Meta-scaler+ uses this space to recommend scaling techniques for any classifier, overcoming the limitation of the original framework, which was restricted to a fixed set of classifiers. The Classifier Performance Space enables dynamic combination of meta-models, improving the flexibility and generalizability of scaling technique selection.

4. **Meta-feature Interpretability:** We enhance the interpretability of Meta-scaler and Meta-scaler+ by analyzing the importance of various meta-features in determining the optimal scaling technique. This provides insights into the aspects of a dataset that most influence the choice of scaling technique depending on the classifier in use, helping practitioners understand the underlying factors that contribute to model performance.

5. **Empirical Validation:** We present extensive experiments validating the effectiveness of both Meta-scaler and Meta-scaler+ across diverse datasets, classifiers, and scaling techniques. Our results demonstrate that Meta-scaler and Meta-scaler+ provide superior scaling recommendations compared to existing methods, significantly improving classification performance in various scenarios.

This thesis also explores the broader implications of scaling techniques, particularly in the context of imbalanced datasets and ensemble models. Our work advances our understanding of how scaling affects classification performance and paves the way for more flexible and efficient automated preprocessing systems in machine learning, with potential applications in AutoML and other areas.

1.1 OBJECTIVES

The main objective of this research is to propose a framework for automatically selecting a scaling technique given a dataset and a classifier, according to meta-characteristics computed from the dataset and information extracted from the classifier.

1.1.1 Specific objectives

1. Demonstrate the impact of the choice of scaling technique in the performance of different classification algorithms;
2. Design and evaluate a framework to recommend scaling techniques for a given query dataset and a known query classifier;
3. Analyze the proposed framework, comparing it to a static choice of ST and the state of the art in meta-learning ST selection methods;
4. Understand how the importance of meta-features changes when the meta-model is trained for different base classifiers;
5. Propose a method to characterize unknown classifiers, regardless of their type;
6. Design and evaluate a framework to recommend scaling techniques for a given query dataset and an unknown query classifier.

1.2 DOCUMENT STRUCTURE

The remainder of this thesis is organized as follows: The first paper, focusing on the impact of scaling techniques in classification performance, is presented in Chapter 2. Then, in Chapter 3, the second paper is presented. It is about the first iteration of the Meta-scaler. The third paper, still unpublished, is presented in Chapter 4. This last study presents the Meta-scaler+, an extension of the Meta-scaler. Finally, Chapter 5 concludes this thesis and presents our follow-up research ideas.

2 THE CHOICE OF SCALING TECHNIQUE MATTERS FOR CLASSIFICATION PERFORMANCE

Lucas B.V. de Amorim, George D. C. Cavalcanti, Rafael M.O. Cruz

This chapter has been published as a paper in January 2023, vol. 133 of the Applied Soft Computing journal. DOI: 10.1016/j.asoc.2022.109924

Abstract

Dataset scaling, also known as normalization, is an essential preprocessing step in a machine learning pipeline. It is aimed at adjusting attributes scales in a way that they all vary within the same range. This transformation is known to improve the performance of classification models, but there are several scaling techniques to choose from, and this choice is not generally done carefully. In this paper, we execute a broad experiment comparing the impact of 5 scaling techniques on the performances of 20 classification algorithms among monolithic and ensemble models, applying them to 82 publicly available datasets with varying imbalance ratios. Results show that the choice of scaling technique matters for classification performance, and the performance difference between the best and the worst scaling technique is relevant and statistically significant in most cases. They also indicate that choosing an inadequate technique can be more detrimental to classification performance than not scaling the data at all. We also show how the performance variation of an ensemble model, considering different scaling techniques, tends to be dictated by that of its base model. Finally, we discuss the relationship between a model's sensitivity to the choice of scaling technique and its performance and provide insights into its applicability on different model deployment scenarios. Full results and source code for the experiments in this paper are available in a GitHub repository.[1]

Keywords: Classification, Normalization, Standardization, Scaling, Preprocessing, Ensemble of classifiers, Multiple Classifier System.

2.1 INTRODUCTION

In a classification task, scaling, also called normalization, is used as an essential preprocessing step to adequate data such that every feature varies within the same range. During the model learning process, this assures that features with higher or wider numerical ranges

[1] https://github.com/amorimlb/scaling_matters

do not dominate those that vary within a narrower or lower range, a phenomenon that may bias the analysis towards less important (less informative) attributes simply because they are on a larger scale. Dataset scaling is thus able to mitigate this phenomenon and consequently improve classification performance.

Most researchers apply this preprocessing step by default in their analysis pipelines, regardless of the classification algorithm or the dataset being used. It thus seems that the benefits promoted by dataset scaling to classification performance are common sense in the machine learning literature. Ratifying this common sense, Singh et al. (3) performed a brief review of papers in different application domains that compared the performance of classifiers trained with nonscaled data to that of classifiers trained with a chosen scaling technique. Their findings confirm that scaling the data previous to model training can lead to significant performance improvements when compared to nonscaled data.

In spite of that, we could estimate that only 16 out of the 50 most cited classification papers since 2018 using the KEEL(5) dataset repository[2] mentioned that this preprocessing step was applied. We believe the remaining 34 papers most likely applied dataset scaling, albeit not mentioning it. This shows how this crucial and ubiquitous preprocessing step is not being given the deserved attention in the literature.

Moreover, there are several scaling techniques that can be applied to a dataset and choosing the best one is by itself an important methodological decision. This decision must be carefully addressed, as it has been reported (3, 4) that choosing the wrong technique can be more detrimental to the classification performance than not scaling the data at all.

Notwithstanding, only a few studies investigate how these scaling techniques compare to each other in terms of the resulting classification performance when they are applied as a preprocessing step to different classification algorithms. Some of these studies focus their analysis on just two scaling techniques (6, 7), while those that experiment with a more extensive number of techniques restrict their tests to just one (3, 4) or three classification algorithms (8), as we detail in Section 2.6. The motivation of this paper, therefore, lies in a combination of all these factors: the pervasiveness of scaling techniques and, at the same time, the lack of attention to it in the research community along with the shortage of empirical studies about their impact on classification performance. This is why we focus our research at answering important questions on how the performance of different classification algorithms varies when the data is scaled with distinct scaling techniques.

[2] The full list of these papers can be found in the supplemental material in the repository.

In this paper, we performed a broader analysis, considering different types of algorithms and also a larger number of datasets to which we apply a relevant number of scaling techniques. By broadening our choices of methods and datasets, we are able to draw conclusions that can more generalized as opposed to a more limited experiment. With that, we intend to understand if scaling affects classification performance and if the choice of scaling technique significantly causes variations in the magnitude of this influence in performance. Moreover, we want to know if these variations behave differently depending on the classification model used, including both monolithic and ensemble models.

Ensemble models combine the outputs of monolithic classifiers to obtain a combined decision that is possibly better than the decisions of each individual classifier in this set (e.g. Random Forests, XGBoost). For the ensemble models, we included models that employ dynamic selection (Dynamic Classifier Selection and Dynamic Ensemble Selection), as these models have shown promising results when compared to monolithic ones (9), which compels their addition to our investigation. To simplify our analysis we included only homogeneous ensembles, in which the base models are instances of a single classifier algorithm.

The inclusion of ensemble models in our experiments also allows us to contribute with two other important and novel analysis: (i) the relation between the rank of best choices of scaling techniques for an homogeneous ensemble and that for its base model, and (ii) the relation between the sensitivity to the choice of scaling technique and the performance of the models, which enabled us to discuss how the ensemble models can be built to manipulate this relation.

Additionally, we looked at how the observed performance variations due to the choice of scaling technique behave when dealing with data presenting different imbalance ratios (IR) since, in real-world data, it is common to have a significant imbalance between the number of instances of each class. For example, in medical diagnosis or face detection problems, when there are many more examples of the negative class than those of the positive class. At the time of writing this paper, we were unable to identify other studies that have explored the effects of scaling techniques when applied to data with different IRs.

To guide this study, we declare the following research questions:

RQ1 - Does the choice of scaling technique matters for classification performance?

RQ2 - Which models present greater performance variations when datasets are scaled with different techniques?

RQ3 - Do homogeneous ensembles tend to follow the performance variation pattern presented by its base model when dealt with different scaling techniques?

In order to answer these questions and perform further analysis, we modeled an experiment using 82 datasets from the KEEL repository after preprocessing them with five different scaling techniques. We applied 20 different classification models to these datasets, including 11 monolithic and 9 ensemble models. The datasets come from different domains, with varying number of instances, features, and with IRs ranging from low to high, i.e. from virtually balanced to extremely imbalanced. We employed 5-fold cross validation and measured classification performance according to two different metrics: F1 and G-Mean. Hypothesis tests were employed to assess the significance of the results.

To summarize, these are the main contributions of this paper:

- It empirically shows that the choice of scaling technique matters for classification performance. This is based on a broad experiment considering different types and a relevant number of classification algorithms, scaling techniques and datasets.
- It is demonstrated that there is a relation between the rank of best choices of scaling techniques for an homogeneous ensemble and that for its base model. This is an important finding, since it makes it less costly to select an optimal scaling technique for an ensemble model.
- It demonstrates that the performance variation due to the choice of different scaling techniques tends to be more salient for datasets with higher imbalance ratios.
- It analyzes the relation between the sensitivity to the choice of scaling technique and the performance of the models, and discusses how the ensemble models can be built to manipulate this relation.

The rest of this paper is organized as follows: Section 2.2 discusses the different scaling techniques available. Section 2.3 presents the classification algorithms used in this paper. The experiment methodology is covered in Section 2.4 and their results and discussions are presented in Section 2.5. Section 2.6 reviews the related works on the comparison of scaling techniques. Finally, Section 2.7 summarizes the lessons learned and Section 2.8 concludes this paper.

2.2 SCALING TECHNIQUES

The raw data that generally come in real-world originated datasets often have issues that preclude them from being effectively explored in a machine learning task. One of the most prominent issues is the different scales in which the various dataset features are presented. In the case of predictive machine learning, this issue causes the algorithms to learn less accurate models. This is due to algorithms' tendency to rely on features that vary within a wider range, i.e., the dominant features, even though these are not necessarily the most informative or decisive features for the correct classification of data instances. To tackle this problem, scaling techniques are employed to adequate data such that every feature varies within the same range. Scaling is thus one of many preprocessing steps that generally have to be undertaken before applying a machine learning model to a dataset.

It is necessary to highlight that although these techniques are more commonly called normalization techniques in the machine learning literature (3, 4, 6, 7, 8) we prefer to use the term scaling because we believe it has a more general meaning. It covers both normalization or standardization methods, in the strict statistical meaning of these words, as well as simpler scaling methods, such as the Maximum Absolute Scaler.

In simple terms, the most common scaling techniques can be broken down into two components: a translational term and a scaling factor. Suppose one wants to transform a vector x, then, in Equation 2.1 each one of its components x_i is transformed into x'_i, where T represents the translational term and S the scaling factor. The translational term operates by moving data along the X-axis, while the scaling factor makes data more concentrated or spread out horizontally.

$$x'_i = \frac{x_i - T}{S} \tag{2.1}$$

Some simpler techniques may not include one of these two aforementioned operations. For example, the Mean Centering technique (Equation 2.2) simply subtracts the vector's mean from each of its components. In this case, the vector's mean is the translational term, and no actual scaling is applied.

$$x'_i = x_i - \bar{x} \tag{2.2}$$

While the Mean Centering effectively removes the offset from the data, shifting its mean to zero, it fails to equalize data variances across the different features of a dataset. More

elaborated techniques promote this equalization by multiplying the data by a scaling factor such that it varies across a defined range, e.g. [-1, 1] or [0,1].

The following subsections present each of the five scaling techniques used in this study: Standard Scaler, Min-max Scaler, Maximum Absolute Scaler, Robust Scaler and Quantile Transformer. These are five well known, diverse, and commonly used scaling techniques. We believe these techniques are good representatives of the most relevant techniques used by the machine learning community. According to the literature, these techniques are diverse and non redundant (3, 4).

Figure 1 presents examples where these techniques scale pairs of attributes, providing a visual feedback as to how these techniques work. Each column of the figure corresponds to one example. The original data distribution is always shown in the topmost graph. In Figure 1a, both variables were randomly generated to follow normal distributions with chosen means and variances. Precisely, if we represent our randomly generated normal variable as $x_i \sim \mathcal{N}(\mu, \sigma^2, n)$, where μ is its mean, σ^2 its variance, and n is the sample size, then: $x_1 \sim \mathcal{N}(10, 4, 1000)$ and $x_2 \sim \mathcal{N}(-10, 4, 1000)$. Here the intent is to see how the scaling techniques deal with variables in opposite sides of the X-axis.

In Figure 1b, the two variables were also randomly generated to follow normal distributions, but 25 outliers are added to x_2 near the 100 value. Precisely: $x_1 \sim \mathcal{N}(10, 4, 1000)$ and $x_2 \sim \mathcal{N}(50, 4, 975) + \mathcal{N}(100, 4, 25)$. This figure shows how the effects of the scaling techniques on a variable that presents outliers compares to their effects on a variable without outliers.

Finally, Figure 1c shows the effects of the techniques in attributes with different distribution shapes: While x_1 is, similar to the previous figures ($x_1 \sim \mathcal{N}(10, 4, 1000)$), x_2 is a randomly generated sample from a uniform distribution in the [-3, 5] range, with a sample size of 1000. The goal of this figure is to show how each technique manipulates the distributions' shapes.

2.2.1 Standard Scaler

The Standard Scaler technique, which implements the Z-score normalization, standardizes attributes by subtracting their mean from each value and dividing the result by the attribute's standard deviation s, resulting in a distribution with zero mean and unit variance. Let $\bar{x}$ be the mean of the x variable, a value x_i is transformed (scaled) into x'_i by means of Equation 2.3.

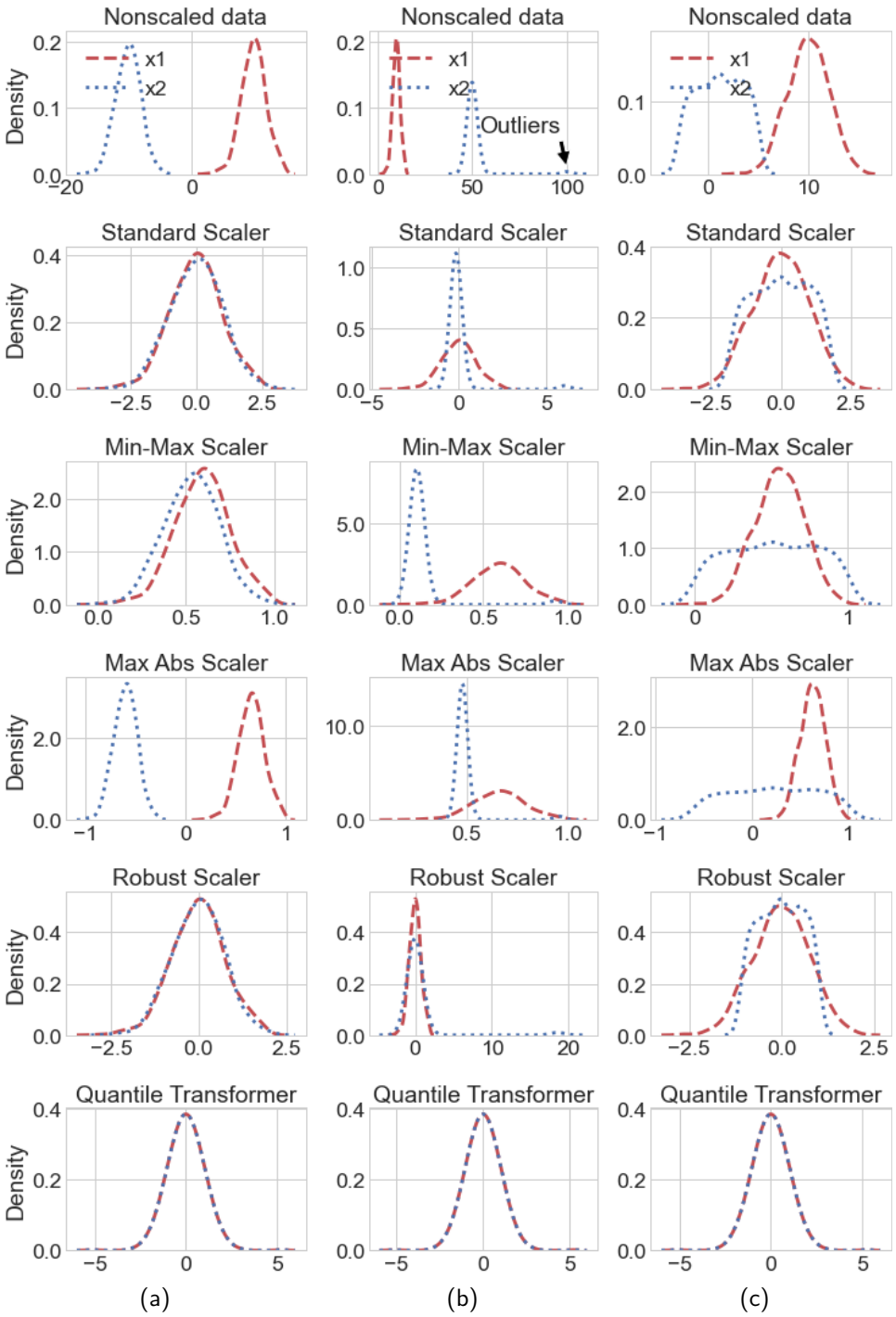


Figure 1 – Effects of the selected scaling techniques when given as input: (a) normally distributed variables with opposing means and no outliers, (b) normally distributed variables where x_2 presents outliers and (c) variables with different distribution shapes: x_1 has a normal distribution and x_2 has a uniform distribution.

$$x'_i = \frac{x_i - \bar{x}}{s} \tag{2.3}$$

In this case, the translational term is the attribute's sample mean, while the scaling factor is its standard deviation. One advantage of this technique is that it can transform both positive

and negative valued attributes into a very similar distribution, as can be seen in Figure 1a. However, in the presence of outliers, it makes the final distribution of inliers too narrow when compared to that of an attribute without outliers (Figure 1b).

Variations of this technique are the Pareto Scaling (10) (Eq. 2.4) and Variable Stability (Vast) scaling (11) (Eq. 2.5). They alter the scaling factor, making the resulting distribution wider. This way, they reduce the importance of the outliers but are not entirely immune to them either.

$$x'_i = \frac{x_i - \bar{x}}{\sqrt{s}} \tag{2.4}$$

$$x'_i = \frac{x_i - \bar{x}}{s}\frac{\bar{x}}{s} \tag{2.5}$$

2.2.2 Min-max Scaler

The Min-max Scaler alters an attribute scale and shifts its values along the X axis so that the transformed attribute ranges within the $[0, 1]$ interval, according to Equation 2.6.

$$x'_i = \frac{x_i - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{2.6}$$

In this technique, the scaling factor consists of the attribute's range, and the translational term is its minimum value. This way, this technique ensures a new minimum of zero and a new maximum of one. For attributes that do not present outliers, the Min-max Scaler has an effect similar to the Standard Scaler. Albeit, in the case of the latter, the resulting distribution will be in a less strictly defined range, as can be observed in Figure 1a. On the other hand, when data present outliers, this technique fails to equalize both the means and variances of the distributions (see Figure 1b) and is thus generally unsuitable for a machine learning pipeline.

The Min-max scaler can be generalized in Equation 2.7 in order to allow for the definition of the resulting range as [a, b] instead of [0, 1]. In fact, we notice that machine learning researchers frequently use this second form to achieve a [-1, 1] range.

$$x'_i = a + \frac{(x_i - x_{\text{min}})(b - a)}{x_{\text{max}} - x_{\text{min}}} \tag{2.7}$$

2.2.3 Maximum Absolute Scaler

The Maximum Absolute Scaler modifies the scale of an attribute by simply dividing each example by the attribute's maximum absolute value, as in Equation 2.8.

$$x'_i = \frac{x_i}{\mathsf{max}(|x|)} \tag{2.8}$$

As such, this technique is sensitive to outliers, as one can see in Figure 1b, where x_2 is transformed into a much narrower distribution than x_1. Also, we notice in Figure 1a that, since this scaling technique misses a translational term, it is unable to equalize the attribute's means.

2.2.4 Robust Scaler

The previous three scaling techniques are very sensitive to the presence of outliers since these transformations depend on the mean or on the minimum and maximum values of each variable. The Robust Scaler seeks to mitigate the effects of outliers by centering data around the median (second quartile of x, $Q_2(x)$) and by scaling it according to the interquartile range, which is the magnitude of the difference between the first quartile $Q_1(x)$ and the third quartile $Q_3(x)$ of x, as shown in Equation 2.9.

$$x'_i = \frac{x_i - Q_2(x)}{Q_3(x) - Q_1(x)} \tag{2.9}$$

Figure 1b shows that this technique is effective in equalizing the variances among the different attributes even when one of them presents outliers. This happens because when the interquartile range is used as the scaling factor, the presence of the outliers is disregarded, since they lie beyond this interval.

2.2.5 Quantile Transformer

This scaling technique is part of a family of techniques that perform a non-linear transformation on the data as opposed to the previously discussed techniques. It is able to change the shape of the original attribute distribution, which can be particularly useful as there has been reported evidence that transforming attributes so that they follow a Gaussian-like distribution

may benefit classification performance (12). The Quantile Transformer technique transforms an attribute using its quantiles information (employing a quantile function) and puts all attributes in the same desired distribution, uniform or normal. In Figure 1 we set the output distribution to 'normal'. It is also a robust technique because it is not sensitive to outliers.

This transformation is applied to each attribute independently in the following way: First, the attribute's cumulative distribution function is estimated and used to map the original values to a uniform distribution. Then, the obtained values are mapped to the desired output distribution using the associated quantile function. New values that fall outside the output range are mapped to the bounds of the output distribution (13).

Notice in Figure 1 that the Quantile Transformer is the only scaling technique that was able to provide consistently similar resulting distributions, regardless of the original attributes given as inputs. It can be seen in Figure 1c that this technique was able to adequate the x_2 attribute, changing its distribution so that it matches a standard normal distribution, with zero mean and unit variance. This promotes better comparability of attributes that previously had different scales and distribution shapes. Although, being a non-linear transformation, it may distort linear correlations between variables measured at the same scale.

2.3 CLASSIFICATION ALGORITHMS

In order to evaluate the scaling techniques over a broad choice of classification algorithms, we selected both monolithic and ensemble models spanning eight different subcategories. The complete list, along with the relevant model's parameters, is presented in Table 1. Our goal is to perform an analysis whose results may be generalized for a wide range of classification algorithms.

With that in mind, we selected monolithic models from the following subcategories: Instance-based, probabilistic, discriminant analysis, rule-based and neural networks. For the ensemble models, the algorithms are distributed among these tree subcategories: Static, DCS, DES. We chose to include ensemble models in our analysis because, (i) these methods, specially DCS and DES, have shown to be more promising than monolithic models in various scenarios (9), (ii) we assume that, due to the effect of the combination of various classifiers, these techniques may be less sensitive to changes in the data than monolithic models. Additionally, we could not find in the literature any analysis on how these algorithms behave when dealt with different scaling techniques.

Table 1 – Parameters used to build the classification models. Other unlisted parameters were left as default according to their respective libraries. DCS - Dynamic Classifier Selection, DES - Dynamic Ensemble Selection. The last five models inherited their pools from the Bagging static model listed above them.

Cat.	Subcat.	Name	Model	Parameters	Library
Monolithic	Instance-based	Support Vector Machine	SVM_lin	kernel = 'linear'	sklearn v0.23.1
		Support Vector Machine	SVM_RBF	kernel = 'rbf'	sklearn v0.23.1
		k-Nearest Neighbors	KNN	n_neighbors = 5, n_jobs=-1	sklearn v0.23.1
		Generalized Learning Vector Quantization	GLVQ	prototypes_per_class=1, max_iter=2500, gtol=1e-05, beta=2	sklearn_lvq v1.1.0
	Probabilistic	Gaussian Naive Bayes	GNB	–	sklearn v0.23.1
		Gaussian Process	GP	kernel = 1.0 RBF(1.0),	sklearn v0.23.1
	Discriminant analysis	Linear Discriminant Analysis	LDA	solver='svd', tol=0.0001	sklearn v0.23.1
		Quadratic Discriminant Analysis	QDA	tol=0.0001	sklearn v0.23.1
	Rule-based	Decision Tree	DT	criterion='gini', splitter='best'	sklearn v0.23.1
	Neural Networks	Perceptron	Percep	alpha=0.0001, max_iter=1000, tol=0.001	sklearn v0.23.1
		Multi-layer Perceptron	MLP	activation='relu', solver='adam', alpha=1e-5, hidden_layer_sizes=(5, 2)	sklearn v0.23.1
Ensemble	Static	eXtreme Gradient Boosting	XGBoost	n_estimators=100, importance_type='gain', tree_method=auto	xgboost 1.2.1
		Random Forests	RF	n_estimators=100, criterion='gini'	sklearn v0.23.1
		AdaBoost	AdaBoost	n_estimators=100, base_estimator= DecisionTreeClassifier(max_depth=1)	sklearn v0.23.1
		Bagging	Bagging	n_estimators=100, base_estimator=Perceptron	sklearn v0.23.1
	DCS	Overall Local Accuracy (OLA)	OLA	pool_classifiers=[Bagging pool]	deslib v0.3.5
		Local Class Accuracy (LCA)	LCA	pool_classifiers=[Bagging pool]	deslib v0.3.5
		Multiple Classifier Behaviour (MCB)	MCB	pool_classifiers=[Bagging pool]	deslib v0.3.5
	DES	k-Nearest Oracles Eliminate (KNORA-E)	KNORAE	pool_classifiers=[Bagging pool], k=7	deslib v0.3.5
		k-Nearest Oracles Union (KNORA-U)	KNORAU	pool_classifiers=[Bagging pool], k=7	deslib v0.3.5

2.3.1 Monolithic Models

2.3.1.1 Instance-based algorithms

Instance-based algorithms rely on a specific subset of instances, rather than a model representing the whole set, to classify each query instance. For example, in the case of Nearest Neighbors classifiers (14), the algorithm uses the training data points closest to a query instance in the feature space to determine its class.

This paper includes the following instance-based algorithms in the experiments: the k-Nearest Neighbors (KNN), GLVQ (15) and SVM (16). For all of these algorithms, we used the implementation provided by the Scikit-learn Python library (sklearn) (17).

The KNN algorithm classifies a new (query) instance by first consulting the labels of k training instances that are nearest to the query instance, and then assigning the most

common label to the query instance. The GLVQ algorithm, which is a generalization of the LVQ algorithm, first learns prototypes to represent the instances and then classify query instances based on their distances to the prototypes. The SVM algorithm tries to maximize the distances between instances of different classes by creating a hyperplane that divides the classes while being equidistant to the most "marginal" instances of both classes, the support vectors. A query instance is assigned to a class according to its position in relation to the hyperplane. Note that one can say that GLVQ and SVM are not purely instance-based, since both of them learn an abstraction of the instances.

2.3.1.2 Probabilistic algorithms

Probabilistic algorithms rely on probability models to calculate the probability that a specific instance belongs to a class. Naive Bayes classifiers well represent this category. Naive Bayes uses Bayes' theorem with the "naive" assumption of conditional independence between the pairs of attributes given a particular class. Despite this simplification, they have proven useful classifiers even when this assumption does not hold (18). Although they are not precise probability estimators in this case (19). An implementation of such classifiers is the Gaussian Naive Bayes (GNB), it assumes the likelihood of features as a Gaussian distribution. This paper includes the GNB and another type of probabilistic classifier, the Gaussian Process (GP) classifier (20).

2.3.1.3 Discriminant Analysis

Discriminant analysis is a family of methods that aim at finding decision surfaces that are most optimally able to separate the classes of the data instances in the feature space. Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) are both included in our experiments. In LDA, the algorithm seeks linear decision surfaces, while in QDA, it seeks quadratic decision surfaces. These methods are also commonly used for dimensionality reduction.

2.3.1.4 Rule-based algorithms

Rule-based classifiers are those that make use of IF-THEN rules for class prediction (21). In this way, we may consider Decision Tree as a rule-based model because it learns these rules,

with which it builds the tree. In this paper, we have included the Decision Tree (DT) classifier included in sklearn that implements an optimized version of the Classification and Regression Trees (CART) algorithm (22).

2.3.1.5 Neural Networks

Artificial Neural Networks (ANNs), which are also referred to as simply "Neural Networks", are algorithms inspired by the mechanisms present in the biological neural networks in our brains. ANNs contain computational units, called neurons, connected through weights, which mimic the role of the strengths of synaptic connections in biological neural networks (23).

The simplest ANN architecture is the Single Layer Perceptron, or simply Perceptron. It contains a single input layer and one output node. Similarly to LDA, it looks for a linear decision surface that better separates the classes of a problem. Multilayer Perceptrons are more complex models that contain multiple computational layers. Each layer feeds its outputs to the inputs of the next layer, which allows for more complex computations. The resulting decision surfaces are nonlinear and hence more flexible, to the extent that may even lead to overfitting, which may be addressed with a better tuning of the model's hyperparameters. This paper includes both a single layer (Percep) and a multilayer perceptron (MLP).

2.3.2 Ensemble Models

Ensemble models, also called Multiple Classifier Systems or Committees of Classifiers, integrate the decisions of a set (ensemble) of classifiers aiming to obtain a combined decision that is better than the decisions of each individual classifier in the ensemble. Theoretical and empirical studies have shown that an ensemble is typically more accurate than an individual classifier (24).

The classifiers inside an ensemble are called base classifiers. Most ensemble models use a single classifier algorithm to generate all the base classifiers. These ensembles are called homogeneous ensembles. On the other hand, when the base classifiers are generated by multiple methods, the ensemble is called a heterogeneous ensemble. For the sake of simplicity, we included only homogeneous ensembles in this work.

When it comes to how the multiple base classifiers' decisions are combined, there are roughly two categories: static and dynamic ensembles. Static ensembles usually generate their

decisions by combining the base classifier outputs through a majority voting rule, where the most frequent decision is chosen, or a variation of this mechanism such as the weighted majority voting (25, 26). The most widely known and employed ensemble models fall within the static ensemble category.

Dynamic ensembles select either a single classifier (DCS - Dynamic Classifier Selection), or a subset of the original ensemble (DES - Dynamic Ensemble Selection) to label a query instance. In both cases, the models consider the local region where the query instance lies, defined by its k nearest training instances, and try to select the best performing base classifier(s) in that region. This local region is called the Region of Competence (RoC) of the input query.

Dynamic ensembles have shown to be very promising for many different scenarios (9), making these models a compelling addition to our arsenal. Moreover, there are no previous studies on how their performances vary under different scaling techniques, which is an interesting investigation since, in their selection phase, these methods use this notion of a local region of competence which is prone to changes due to dataset scaling. This paper includes nine ensemble models, which we describe below. As can be seen in Table 1, for the ensemble models, we used the implementations provided by the Scikit-learn (17), XGBoost (27) and the DESlib (28) Python libraries.

2.3.2.1 Static Ensembles

Four static ensembles representing the most common approaches are considered in this paper.

Bagging Bagging, an acronym to *Bootstrap AGGregatING* was introduced by Leo Breiman, in 1996 (29). Its idea is to create an ensemble comprising diverse base classifiers by giving them bootstrap replicates of the training set (24). In other words, each base classifier is trained on a different subset of the labeled data, usually generated by random sampling with replacement. The outputs of the various base classifiers are combined with majority voting.

Random Forests Also introduced by Leo Breiman, in 2001 (30), Random Forests is an extension of Bagging, albeit specifically designed for decision tree ensembles, where the major difference is that it incorporates randomized feature selection (24). When building

each decision tree in the ensemble, besides giving them a bootstrap copy of the training set, the algorithm performs the conventional decision tree split procedure but within a random subset of the features.

AdaBoost AdaBoost, whose name derives from *ADAptive BOOSTing*, is a boosting technique introduced by Freund and Schapire in 1997 (31). This technique works by iteratively adding one classifier at a time. Each classifier is trained on a selectively sampled subset of the training instances, where the sampling distribution starts uniform but changes with each iteration by attributing higher selection probabilities to the instances that were misclassified in the previous iteration (24). This way, the algorithm gathers classifiers that have their training each time more focused on the harder instances of the training set.

XGBoost XGBoost provides optimizations over the original GBM (Gradient Boosting Machine) algorithm (32) aiming to make it more scalable and accurate. Gradient Boosting Machine, or Gradient Tree Boosting, in its turn can be understood as a generalization of AdaBoost. It also gradually adds classifiers (in this case, Decision Trees) to the ensemble. The main difference lies in how it boosts the base classifiers: instead of assigning higher weights to harder instances, GBM minimizes a differentiable loss function that can be user-defined. A tree is added to the ensemble if it reduces the loss. XGBoost adds regularization to control overfitting, parallel processing, ways to deal with sparse data, and other modifications that improve its performance over previous boosting methods (27).

2.3.2.2 Dynamic Ensembles

Five dynamic ensembles are included in our experiments: 3 DCS and 2 DES strategies.

Overall Local Accuracy (OLA) OLA is a simple DCS method that evaluates the competence (accuracy) of each base classifier in the RoC of the query instance and then selects the one that presents the highest accuracy. The output of the selected classifier determines the ensemble's decision (33).

Local Class Accuracy (LCA) LCA is a DCS method that evaluates the local competence of each base classifier with regards to a specific class. Given that a base classifier predicts

class w_l for the query instance, its competence is the percentage of training samples in the RoC labeled as w_l that it correctly predicts as w_l. The most competent classifier then determines the output of the ensemble (33).

Multiple Classifier Behavior (MCB) MCB is a DCS method where the RoC is defined both by the kNN approach and by the behavioral knowledge (BKS) method (34). First, the outputs of all base classifiers are calculated for all training instances. Then a initial RoC is defined by selecting the k nearest neighbors. This RoC is then filtered by selecting only the instances that are similar enough to the query instance based on the BKS method. The BKS method measures the similarity between the decisions of multiple classifiers when given as input a pair of instances. When this similarity is higher than a certain threshold, the instance is selected to integrate the RoC. After defining the RoC, the competence level of a classifier is calculated according to OLA. At the end, the most competent base classifier determines the ensemble output if it is better than the other classifiers by a certain threshold, else a majority voting rule is applied (35).

k-Nearest Oracles Eliminate (KNORA-E) KNORA-E is part of the KNORA family of DES methods proposed by Ko et al. (36) that is inspired by the Oracle concept (37). An Oracle is an abstract classifier selection method that always selects the classifiers that return the correct output if such classifiers exist.

For the KNORA-E method, a classifier is considered competent in a certain RoC if it achieves perfect performance for the instances within this region. Such a classifier is known as a local oracle. All local oracles are selected. In the case that no local oracles exist in the RoC, the number of neighbors that compose the RoC is iteratively reduced, by removing the farthest neighbor, until at least one classifier achieves perfect performance. When no local oracles are found, the whole pool (initial ensemble) is used for classification. In both cases, the outputs of the resulting ensemble are combined using the majority voting rule.

k-Nearest Oracles Union (KNORA-U) The KNORA-U method, in its turn, selects all classifiers that produce the correct output for at least one instance in the RoC. For each instance correctly classified in the RoC, the base classifier can submit one vote to classify the query instance. The votes of all the selected classifiers are combined to produce the output, which will be the class with more votes.

2.4 EXPERIMENT METHODOLOGY

For this experiment, we applied the five selected scaling techniques to each of the 82 original datasets selected for the study, creating 5 additional variants of each dataset. Then, we applied 20 different classification models to those datasets. In order to keep things simple, we decided not to balance the datasets, leaving them with their original class imbalance ratios. Classification performance was measured according to two different metrics: F1 and G-Mean. We chose these metrics because they consider the imbalanced nature of the datasets (38). The selection of the scaling techniques and classification algorithms was performed prioritizing the diversity and the popularity of the methods within the machine learning community. In the case of the scaling techniques, we avoided including redundant techniques, as in previous works that used a larger number of techniques it was reported that many of them yielded similar results for most datasets (3, 4).

The source code for the experiment, all the datasets used, and the full results table are available at the GitHub repository mentioned earlier.

2.4.1 Datasets selection

In order to allow the reproducibility of this study, we used multiple publicly available datasets, showing a wide range of IR values obtained from the KEEL dataset repository (5). These real world datasets were originally published in the UCI Machine Learning repository, which contains a collection of datasets that are frequently used by machine learning researchers for the empirical analysis of algorithms (39). The UCI repository also contains artificial datasets and data generators, but it was because of its wide range selection of real world datasets that it became one of the most popular sources for empirical machine learning studies.

The datasets we used were preprocessed by the KEEL team in a way that the multi-class problems were transformed into binary ones and also were split into files aiming its use with a 5-fold stratified cross-validation (which we employed). This way, since datasets are already pre-split, any researcher that uses this data will use the exact same folds. This also promotes research reproducibility.

First, we selected all the 91 datasets available at KEEL that are both binary and with varying imbalance ratios. Then, we selected those with a maximum 30% categorical attributes (such that these attributes do not exert a significant influence on the results, as we are interested

in the effects of scaling techniques and these only apply to numerical data), resulting in 82 datasets, as described in Table 2. This table shows the datasets' names alongside the numbers of numerical and categorical attributes, the class counts (i.e., how many instances pertain to each class), and, finally, the imbalance ratio (IR), which is the proportion of the number of instances of the majority class to those of the minority class. The table is sorted according to the imbalance ratio.

Table 2 – Datasets description. Note: Num. attrib. is the number of numerical attributes, while Categ. attrib. is the number of categorical attributes.

#	Dataset name	Num. attrib.	Categ. attrib.	Class counts	IR
1	glass1	9	0	(138, 76)	1.82
2	ecoli-0_vs_1	7	0	(143, 77)	1.86
3	wisconsin	9	0	(444, 239)	1.86
4	pima	8	0	(268, 500)	1.87
5	iris0	4	0	(50, 100)	2.00
6	glass0	9	0	(70, 144)	2.06
7	yeast1	8	0	(1055, 429)	2.46
8	haberman	3	0	(225, 81)	2.78
9	vehicle2	18	0	(628, 218)	2.88
10	vehicle1	18	0	(629, 217)	2.90
11	vehicle3	18	0	(634, 212)	2.99
12	glass-0-1-2-3_vs_4-5-6	9	0	(163, 51)	3.20
13	vehicle0	18	0	(199, 647)	3.25
14	ecoli1	7	0	(259, 77)	3.36
15	new-thyroid1	5	0	(180, 35)	5.14
16	ecoli2	7	0	(284, 52)	5.46
17	segment0	19	0	(1979, 329)	6.02
18	glass6	9	0	(185, 29)	6.38
19	yeast3	8	0	(1321, 163)	8.10
20	ecoli3	7	0	(301, 35)	8.60
21	page-blocks0	10	0	(4913, 559)	8.79
22	ecoli-0-3-4_vs_5	7	0	(180, 20)	9.00
23	yeast-2_vs_4	8	0	(463, 51)	9.08
24	ecoli-0-6-7_vs_3-5	7	0	(200, 22)	9.09
25	ecoli-0-2-3-4_vs_5	7	0	(182, 20)	9.10
26	glass-0-1-5_vs_2	9	0	(155, 17)	9.12
27	yeast-0-3-5-9_vs_7-8	8	0	(456, 50)	9.12
28	yeast-0-2-5-6_vs_3-7-8-9	8	0	(905, 99)	9.14
29	ecoli-0-4-6_vs_5	6	0	(183, 20)	9.15
30	ecoli-0-1_vs_2-3-5	7	0	(220, 24)	9.17
31	ecoli-0-2-6-7_vs_3-5	7	0	(202, 22)	9.18
32	glass-0-4_vs_5	9	0	(83, 9)	9.22
33	ecoli-0-3-4-6_vs_5	7	0	(185, 20)	9.25
34	ecoli-0-3-4-7_vs_5-6	7	0	(232, 25)	9.28
35	yeast-0-5-6-7-9_vs_4	8	0	(477, 51)	9.35
36	vowel0	13	0	(90, 898)	9.98
37	ecoli-0-6-7_vs_5	6	0	(200, 20)	10.00
38	glass-0-1-6_vs_2	9	0	(175, 17)	10.29
39	ecoli-0-1-4-7_vs_2-3-5-6	7	0	(307, 29)	10.59
40	led7digit-0-2-4-5-6-7-8-9_vs_1	7	0	(406, 37)	10.97
41	ecoli-0-1_vs_5	6	0	(220, 20)	11.00
42	glass-0-1-4-6_vs_2	9	0	(188, 17)	11.06
43	glass2	9	0	(197, 17)	11.59
44	ecoli-0-1-4-7_vs_5-6	6	0	(307, 25)	12.28
45	cleveland-0_vs_4	13	0	(160, 13)	12.31
46	ecoli-0-1-4-6_vs_5	6	0	(260, 20)	13.00
47	shuttle-c0-vs-c4	9	0	(1706, 123)	13.87
48	yeast-1_vs_7	7	0	(429, 30)	14.30
49	glass4	9	0	(201, 13)	15.46
50	ecoli4	7	0	(316, 20)	15.80
51	page-blocks-1-3_vs_4	10	0	(444, 28)	15.86
52	abalone9-18	7	1	(689, 42)	16.40
53	dermatology-6	34	0	(338, 20)	16.90
54	glass-0-1-6_vs_5	9	0	(175, 9)	19.44
55	shuttle-c2-vs-c4	9	0	(6, 123)	20.50
56	shuttle-6_vs_2-3	9	0	(220, 10)	22.00
57	yeast-1-4-5-8_vs_7	8	0	(663, 30)	22.10
58	glass5	9	0	(205, 9)	22.78
59	yeast-2_vs_8	8	0	(462, 20)	23.10
60	yeast4	8	0	(1433, 51)	28.10
61	winequality-red-4	11	0	(1546, 53)	29.17
62	poker-9_vs_7	10	0	(236, 8)	29.50
63	yeast-1-2-8-9_vs_7	8	0	(917, 30)	30.57
64	abalone-3_vs_11	7	1	(15, 487)	32.47
65	winequality-white-9_vs_4	11	0	(163, 5)	32.60
66	yeast5	8	0	(1440, 44)	32.73
67	winequality-red-8_vs_6	11	0	(638, 18)	35.44
68	ecoli-0-1-3-7_vs_2-6	7	0	(274, 7)	39.14
69	abalone-17_vs_7-8-9-10	7	1	(2280, 58)	39.31
70	abalone-21_vs_8	7	1	(14, 567)	40.50
71	yeast6	8	0	(1449, 35)	41.40
72	winequality-white-3_vs_7	11	0	(880, 20)	44.00
73	winequality-red-8_vs_6-7	11	0	(837, 18)	46.50
74	abalone-19_vs_10-11-12-13	7	1	(32, 1590)	49.69
75	winequality-white-3-9_vs_5	11	0	(1457, 25)	58.28
76	poker-8-9_vs_6	10	0	(1460, 25)	58.40
77	shuttle-2_vs_5	9	0	(3267, 49)	66.67
78	winequality-red-3_vs_5	11	0	(681, 10)	68.10
79	abalone-20_vs_8-9-10	7	1	(1890, 26)	72.69
80	poker-8-9_vs_5	10	0	(2050, 25)	82.00
81	poker-8_vs_6	10	0	(1460, 17)	85.88
82	abalone19	7	1	(4142, 32)	129.44

The datasets represent problems from a diverse range of domains. There are datasets on glass identification (glass), medical analyses (pima, wisconsin, haberman, cleveland, dermatology), plant identification (iris), speech recognition (vowel), molecular and cellular biology (yeast, ecoli), image recognition (vehicle, led7digit), aeronautics (shuttle) and others.

2.4.2 Datasets Scaling

Before applying the scaling techniques, two other common preprocessing tasks were performed: strings cleaning (stripping and standardization) and one-hot encoding, which were applied to the categorical attributes. In the strings cleaning step, all the class names and values were standardized within each dataset, removing extra white spaces and correcting typos. The same was applied to the values of categorical attributes, to avoid recognizing misspelled values as new categories. In the one-hot encoding step, each categorical attribute is replaced by $n-1$ binary columns where n is the number of unique values of the attribute. This way, the combination of the bits in the $n-1$ binary columns numerically encode the n possible categorical values of the attribute. This ensures that the dataset can be treated by algorithms that can not deal with nonnumerical attributes.

As for the data scaling, which is the main independent variable in this study, we implemented the following procedure: We created five copies of each of the 82 original datasets, then we applied to each of the five copies one of the selected scaling techniques: Standard Scaler, Min-max Scaler, Maximum Absolute Scaler, Robust Scaler, and Quantile Transformer. We also kept an unaltered copy of the datasets (nonscaled) to serve as a baseline.

It is important to register that, in order to avoid leaking information from the test set into the training phase, a problem known as the *look-ahead bias*, each of the five folds in a dataset were independently scaled according to the parameters obtained from the training set pertaining to that fold. For example, for the Min-Max technique, for every fold, the minimum and maximum values are estimated using the training set and are later used to transform the test set.

2.4.3 Performance metrics definitions

Classification accuracy is one of the most obvious measures of classification performance. It is calculated by the number of correctly classified examples over the total number of instances. However, this metric is not a good choice for highly imbalanced data, as it is easy to get an overly optimistic accuracy simply by classifying all instances of the test set as being of the majority class. Therefore, a different choice of metric is adamant for meaningful performance evaluation when dealing with datasets presenting different degrees of class imbalance. In this sense, we chose to evaluate models utilizing two metrics that consider the imbalanced nature

of the datasets (38): F1 and G-Mean. Both metrics can be defined based on the elements of a confusion matrix, as in Figure 2.

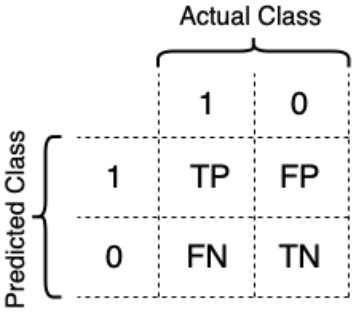


Figure 2 – Confusion matrix for a two-class problem.

In this matrix, TP (True Positives) and TN (True Negatives) represent the numbers of instances that were correctly classified as positive and negative, respectively. In contrast, FP (False Positives) and FN (False Negatives) represent the number of those incorrectly classified as positive and negative, respectively. In the following subsections, we present a definition for both F1 and G-Mean metrics.

2.4.3.1 F1 definition

F1, or F-score, is an harmonic mean between precision and recall (or sensitivity), which in turn are defined as in Equations 2.10 and 2.11.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.10}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.11}$$

Then, F1 can be defined as in Equation 2.12.

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{2.12}$$

F1 can be seen as a particular case of the F_β metric (Eq.2.13), with $\beta = 1$.

$$\text{F}_\beta = \frac{(1 + \beta^2) \cdot TP}{(1 + \beta^2) \cdot TP + \beta^2 \cdot (FP + FN)} \tag{2.13}$$

2.4.3.2 G-Mean definition

G-Mean is calculated as the square root of the product of sensitivity and specificity, as in Equation 2.16. Where sensitivity, or recall, can be seen as the True Positive Rate while specificity can be seen as the True Negative Rate.

$$\text{Sensitivity} = \text{TPR} = \frac{TP}{TP + FN} \tag{2.14}$$

$$\text{Specificity} = \text{TNR} = \frac{TN}{FP + TN} \tag{2.15}$$

$$\text{G-Mean} = \sqrt{\text{TPR} \cdot \text{TNR}} = \frac{\sqrt{TP \times TN}}{\sqrt{(TP + FN)(TN + FP)}} \tag{2.16}$$

2.4.4 Software

This experiment was executed using the Python programming language, version 3.8.3 using, mostly, the following libraries: Scikit-Learn(17) version 0.23.1, sklearn_lvq version 1.1.0, DESlib(28) version 0.3.5 and SciPy(40) version 1.5.0.

2.5 RESULTS AND DISCUSSION

This section presents graphs and tables that summarize our experimental results aiming to answer the research questions defined in Section 2.1 and also provide further analysis.

In Section 2.5.1, in order to answer RQ1, we compare the performances obtained by models considering the different scaling techniques and we answer RQ2 by summarizing our findings with respect to which models are more (or less) sensitive to the choice of the scaling technique.

Concerning RQ3, in Section 2.5.2 we look for a relation between the behavior of a monolithic model and an homogeneous ensemble built with it, regarding how the scaling techniques rank for that specific model. Finally, in Section 2.5.3 we present our findings on how the sensitivity to the choice of scaling technique relates to the model's performance, and we discuss how this analysis can be useful for model selection in different deployment scenarios.

2.5.1 Models performances

In order to answer the **research question RQ1** – Does the choice of scaling technique matters for classification performance? – for each dataset, we compared the classification performances of each model when they were trained with the five scaled copies of the dataset and the baseline, which is the original, nonscaled dataset. We then applied a Friedman hypothesis test to evaluate the statistical significance of the differences presented.

Table 3 show, for each one of the 20 classification models, the values obtained for F1 and G-Mean for each scaling technique after taking the mean across all datasets. In these tables, we abbreviated the scaling techniques names as NS - No scaling (baseline), SS - Standard Scaler, MM - Min-Max Scaler, MA - Max Absolute Scaler, RS - Robust Scaler, QT - Quantile Transformer. Full results for the experiments can be found in CSV files in the GitHub repository mentioned above.

Table 3 – Mean performances of the classification models. The table shows, for each model, the mean obtained with the pair of metric and scaling technique. Each value is a mean calculated over the 82 datasets.

Model	F1						G-Mean					
	NS	**SS**	**MM**	**MA**	**RS**	**QT**	**NS**	**SS**	**MM**	**MA**	**RS**	**QT**
SVM_lin	0.44	0.51	0.38	0.36	0.49	**0.52**	0.48	0.56	0.41	0.38	0.54	**0.57**
SVM_RBF	0.35	0.52	**0.48**	0.45	0.40	0.45	0.38	**0.56**	0.51	0.48	0.44	0.49
KNN	0.51	0.55	0.54	0.53	0.54	**0.56**	0.56	0.60	0.58	0.57	0.59	**0.60**
GNB	0.42	0.40	0.41	0.41	0.42	**0.42**	0.61	0.59	0.60	0.60	**0.62**	0.57
GLVQ	0.10	**0.11**	0.08	0.09	0.17	0.09	0.11	**0.11**	0.08	0.09	0.19	0.10
LDA	**0.54**	**0.54**	**0.54**	**0.54**	**0.54**	0.52	**0.61**	**0.61**	**0.61**	**0.61**	**0.61**	0.59
QDA	0.36	0.37	0.33	0.37	**0.38**	0.36	0.47	0.48	0.45	0.48	**0.49**	0.43
GP	0.52	0.36	**0.57**	0.56	0.44	0.32	0.56	0.38	**0.62**	0.61	0.48	0.35
DT	0.56	0.56	0.56	0.56	**0.57**	0.57	0.68	0.68	0.68	0.68	**0.69**	0.68
Percep	0.37	**0.52**	0.47	0.44	0.48	0.47	0.44	**0.63**	0.55	0.52	0.60	0.58
MLP	0.08	0.19	0.05	0.05	0.18	**0.21**	0.09	0.21	0.05	0.05	0.20	**0.23**
RF	0.56	**0.56**	0.55	0.55	0.56	0.55	0.60	0.60	0.60	0.60	**0.60**	0.60
XGBoost	0.59	0.58	0.59	**0.59**	0.59	0.58	**0.65**	0.65	0.65	0.65	0.65	0.64
AdaBoost	0.57	0.57	0.57	0.57	0.57	**0.57**	0.65	0.65	0.64	0.64	0.64	**0.65**
Bagging	0.39	**0.56**	0.50	0.48	0.52	0.53	0.44	**0.62**	0.55	0.52	0.59	0.59
OLA	0.53	**0.60**	0.60	0.58	0.58	0.60	0.60	0.70	0.67	0.65	0.68	**0.70**
LCA	0.40	**0.49**	0.46	0.42	0.47	0.48	0.45	**0.56**	0.53	0.48	0.54	0.55
MCB	0.53	0.59	**0.59**	0.58	0.59	0.59	0.60	**0.69**	0.67	0.65	0.68	0.69
KNORAE	0.58	0.61	0.61	0.60	0.61	**0.62**	0.67	0.70	0.70	0.69	0.70	**0.71**
KNORAU	0.48	**0.57**	0.53	0.50	0.55	0.55	0.53	**0.63**	0.58	0.55	0.61	0.61

It is interesting to notice that Table 3 shows that nonscaled data are not always the least performant method. This finding endorses the need to properly and wisely select the scaling

technique for a specific pair of dataset and model, as a carelessly selected technique can be worse than not scaling the data at all.

Table 3 also shows that the performance variation when we consider different scaling techniques seem to be relevant for most (14 out of 20 models), except for three monolithic models: LDA, QDA and DT, for the three ensemble models based on DTs: XGBoost, RF, and AdaBoost.

We performed Friedman tests in order to estimate the statistical significance of the observed differences in performance when using different scaling techniques. The test was performed considering these hypotheses and a 0.05 significance level:

> H_0 - There is no significant difference between the means obtained for models built with datasets processed with different scaling techniques.
>
> H_1 - There is a significant difference between the means obtained for models built with datasets processed with different scaling techniques.

For each of these tests, each sample being compared is composed of the 82 performance measurements for a certain pair of models and scaling technique, whose means were previously presented in Table 3. In order to enable a discussion on the role of class balancing in the observed performance differences, we also performed three other sets of Friedman tests considering datasets in different IR strata: low IR datasets (IR $\leq$ 3.0) i.e. virtually balanced datasets, medium IR (3 $<$ IR $<=$ 9) and high IR (IR $>$ 9). Results for all four sets of tests are presented in Table 4.

From this table, we notice that the tests for balanced datasets (i.e., low IR) and the medium IR datasets show much fewer null hypothesis rejections (15 and 11 rejections out of 40, respectively) than those performed considering all the datasets (28 rejections out of 40) or even just the high IR datasets (27 rejections out of 40). This observation is an indication that, although the scale sensitivity problem exists even for more balanced datasets, highly imbalanced datasets are more prone to significant performance variation due to the choice of different scaling techniques. Additionally, even for balanced datasets, the very low p-values observed for the SVMs, QDA, Percep, MLP and Bagging indicate that those differences in performance should not be neglected.

We now shift our focus to the last two columns in Table 4, which correspond to the tests results considering all the 82 datasets. When we take into account the monolithic models (from

SVM_lin to MLP), the results state that the null hypothesis can be rejected in most cases, except for the Decision Tree (DT) model and partially (one of two metrics) for the KNN model. Although in the case of KNN, the 0.07 p-value was close to the 0.05 threshold. This means that, for almost all monolithic models studied, there is a significant difference in performance when different scaling techniques are applied to imbalanced datasets. The insensitivity to scale observed in the DT model was already expected since this is a rule-based model, and it assigns decision thresholds for each attribute independently of the scale of the other attributes.

Although we expected that the KNN model presented more significant performance differences among the different scaling techniques, it is not as affected as several other classifiers such as SVM_lin, SVM_RBF MLP, Perceptron and GP. One must remember that this model is only sensitive to scale when the difference among the features significantly changes an instance's set of neighbors. Thus, affecting the final classification. This may not necessarily be the case with most of the studied datasets.

For the ensemble models, the results show that ensembles built with Decision Trees as their base models (RF, XGBoost e AdaBoost) are also insensitive to the choice of scaling technique. On the other hand, the remaining ensembles, built with Perceptrons as their base models, are sensitive to the scale of the data, with the KNORAE model being a notable exception. This indicates that there seems to be a relation between the sensitivity to scale that we observe in a monolithic model and that of an ensemble built based on that model. This topic is further explored in Section 2.5.2, where we answer RQ3.

As for the stability observed in the results from the KNORAE model, a possible explanation is that the algorithm works by selecting all base classifiers with a perfect performance in the region of competence (RoC) of the query instance (its k nearest neighbors), but while no classifiers are attaining perfect performance in the RoC, k keeps being decremented by 1, reducing the region of interest. This way, when the most distant neighbor(s) is(are) eliminated, the algorithm removes samples with extreme values for some features. This way, it reaches a RoC where samples present more well-behaved scales among its features, over which a different choice of scaling technique has little effect.

In order to better visualize how much difference in performance the choice of scaling technique can promote, in Figure 3 we show the bar plots representing the performance variation range for each model. The mean over all datasets is considered here. These graphs, as expected, endorses the results and conclusions obtained from the hypothesis test, as we can see greater ranges for the SVMs, GP, Percep and MLP monolithic models and very low ranges for

the DT model and the ensembles built with it as a base model.

Additionally, notice that, for the monolithic models, the mean difference in performance (mean range) is close to, or more than, 0.15 for five models (SVM_lin, SVM_RBF, GP, Percep and MLP), and this can represent, in some contexts, the difference between a useful and a useless model. For the ensemble models, the differences are less impressive but not negligible.

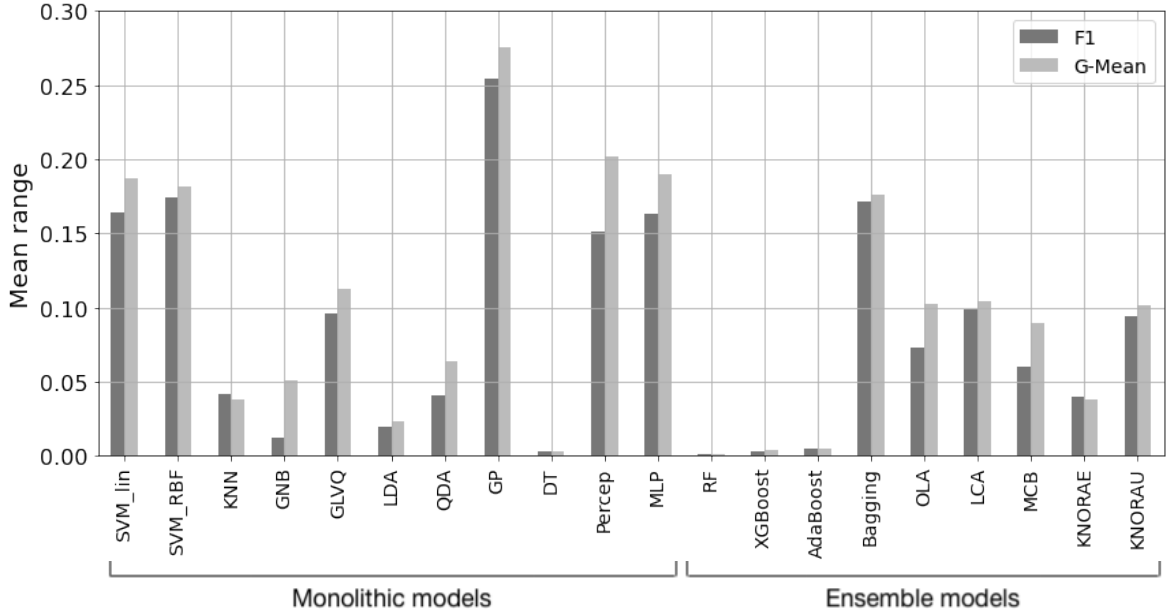


Figure 3 – Mean ranges (differences from best to worst scaling technique) for the monolithic and ensemble models.

Provided all the above evidence, we can answer the research question **RQ1: Yes, the choice of scaling technique matters for classification performance.** We could verify this for both monolithic and ensemble models. The hypothesis tests show that the performance differences are significant for most models. The varying extent to which scaling affects performance for distinct models brings us to our next research question.

With the same resources we employed to answer RQ1, we can also answer **research question RQ2** – Which models present greater performance variations when datasets are scaled with different techniques? – we begin looking back at Figure 3, which presents the mean ranges (considering all 82 datasets) in model performance (difference between the best and worst scaling technique). As previously discussed, there appears to be a great variability for some models in both monolithic and ensemble groups and almost no variation when considering a few select models.

It is important to stress that Figure 3 presents the means over all 82 datasets. Then, for the most sensitive models, the performance gain when choosing the right scaling technique

may be even higher for some datasets. If we take the SVM_lin model, for example, looking at the raw results (available in the supplemental materials), we can see that its maximum difference between the best and the worst scaling technique occurred for the "segment-0" dataset, a range of 0.991 for the F1 metric. This result for the "segment-0" dataset is not exactly an outlier, since, for the SVM_lin model, the range achieved in 26 (32%) of the 82 datasets was above 0.5, and in 6 (7%) of them, the range was higher than 0.8. These are extreme differences between the best and worst performances of the same model caused only by swapping the scaling techniques. This further emphasizes the need for careful choice of the scaling technique to be applied in a classification task.

We can see in Figure 3 that some models, such as SVM_lin, SVM_RBF, GLVQ, GP, Percep, MLP, Bagging, OLA, LCA and KNORAU present larger mean ranges while models such as DT, and the DT-based ensembles (RF, XGBoost and AdaBoost) are almost insensitive to the choice of scaling technique. This is confirmed by the results of the Friedman tests in Table 4, which leads us to some possible generalizations: Considering the monolithic models and their categories in Table 1, instance-based (KNN is an exception) and Neural Networks (MLP and Percep) models seem to be more sensitive to how data are scaled. In contrast, rule-based and discriminant analysis models are less sensitive. When we consider the ensemble models, those built using decision trees as their base models (XGBoost, RF and AdaBoost) are equally insensitive to the scaling technique chosen, while those based on Perceptrons (all the others) are more sensitive.

As an additional analysis, in Table 5 we present the number of wins of each scaling techniques (how many times each one was better than the others) for each IR strata. This allows us to conclude that considering all the IR strata, the Quantile Transformer (QT) technique is the most successful. In the low IR stratum, the best technique is the Standard Scaler (SS), while in the medium and high strata the Quantile Transformer is again the most performant technique. This indicates that the IR may be an important variable to determine which scaling technique is the best for a certain dataset.

2.5.2 Do scaling techniques rank similarly for an ensemble and its base model?

Up to this point, we can see from Figure 3 that there is an apparent relationship between the variability to scale observed in ensembles and that of their base models. This observation demands further investigation that we intend to carry out as we answer **research question**

RQ3 - Do homogeneous ensembles tend to follow the performance variation pattern presented by their base model when dealing with different scaling techniques?

By "performance variation pattern" we mean the order in which the scaling techniques rank when considering a particular model and dataset. We then aim to answer if, on average, the best ranking scaling technique is the same between a specific monolithic model and its corresponding ensemble and also if the second, third-ranking techniques and so on, are coincident.

In order to better compare these rankings for a monolithic model and an ensemble built with it, we must select a monolithic model and all its corresponding ensembles. Due to the way we designed the ensembles in our study, we have two possible monolithic models for this study: DT or Percep, because the ensembles we built have either DT or Percep as their base models. Since we have already verified that the DT model and its correspondent ensembles are practically scale insensitive, we chose to compare the Percep monolithic model and the ensembles built with it as their base model: Bagging, OLA, LCA, MCB, KNORAE, KNORAU.

Although we are limiting this analysis to perceptrons and perceptron-based ensembles, this is a specially useful model for building ensembles due to its simplicity, low computational cost and sensitivity to data sampling, which helps promote the much-desired ensemble diversity (24). Additionally, it has been shown in the DES literature that the use of weak models as a base classifier, such as decision trees and perceptrons, allows better results, outperforming stronger models like Random Forest, SVM or MLP (41, 42, 43, 44, 45).

Figure 4 shows, for the Percep and the six ensembles based on it, the performance variation pattern along with the different scaling techniques for both F1 and G-Mean metrics. Each bar represents the mean result over all the 82 datasets for the corresponding scaling technique and metric.

We note from these bar plots that, for both metrics, the best and worst scaling techniques are coincident for all models: The best performances were obtained with the Standard Scaler, and the worst results were obtained with nonscaled data (NS). That furthers our belief in the relationship between the behavior of the Percep model and its derived ensembles. However, we can see counterexamples when we look for the bar that ranks second in each model: While that is Robust Scaler for the Percep, it is Quantile Transformer for all the others, although they do not seem to differ significantly. These observations indicate that a better analysis must be made in order to answer this question. Instead of only looking at the rankings of the means in Figure 4, we calculated the average ranking (considering the 82 datasets) of each scaling technique for the Percep model versus those from the six corresponding ensemble models and

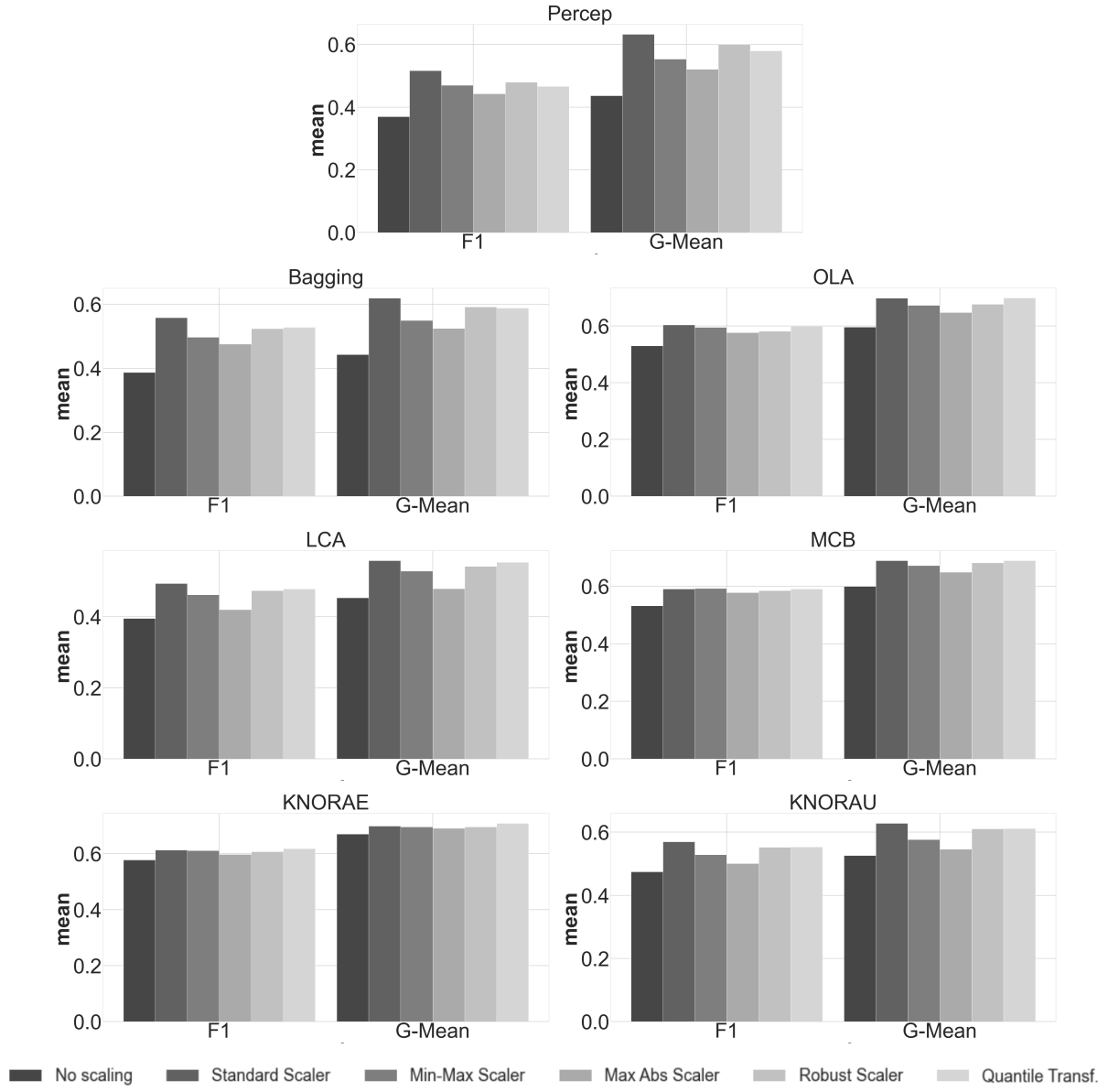


Figure 4 – Mean performances for the six scaling techniques for the Perceptron model (above) and its correspondent ensembles.

represented the results as critical difference (CD) diagrams in Figures 5 and 6 for F1 and G-Mean respectively. These CD diagrams were built using a Nemenyi post hoc test.

In Figures 5 and 6, we observe that the more scale-sensitive ensembles (i.e., those that are more prone to changes due to the choice of scaling technique) tend to follow the trend set by the Percep model in which they are based. For example, for both F1 and G-Mean, the top two and bottom two scaling techniques are coincident when we consider the Percep and the more sensitive ensemble models (Bagging, LCA and KNORAU): SS and RS appear in first and second places while MA and NS appear as fifth and sixth places. Nonetheless, this does not happen for models that are less sensitive to scale (specially KNORAE), mainly because, for these models, the differences between the rankings of the distinct scaling techniques are

less significant, as can be seen in Figures 5 and 6.

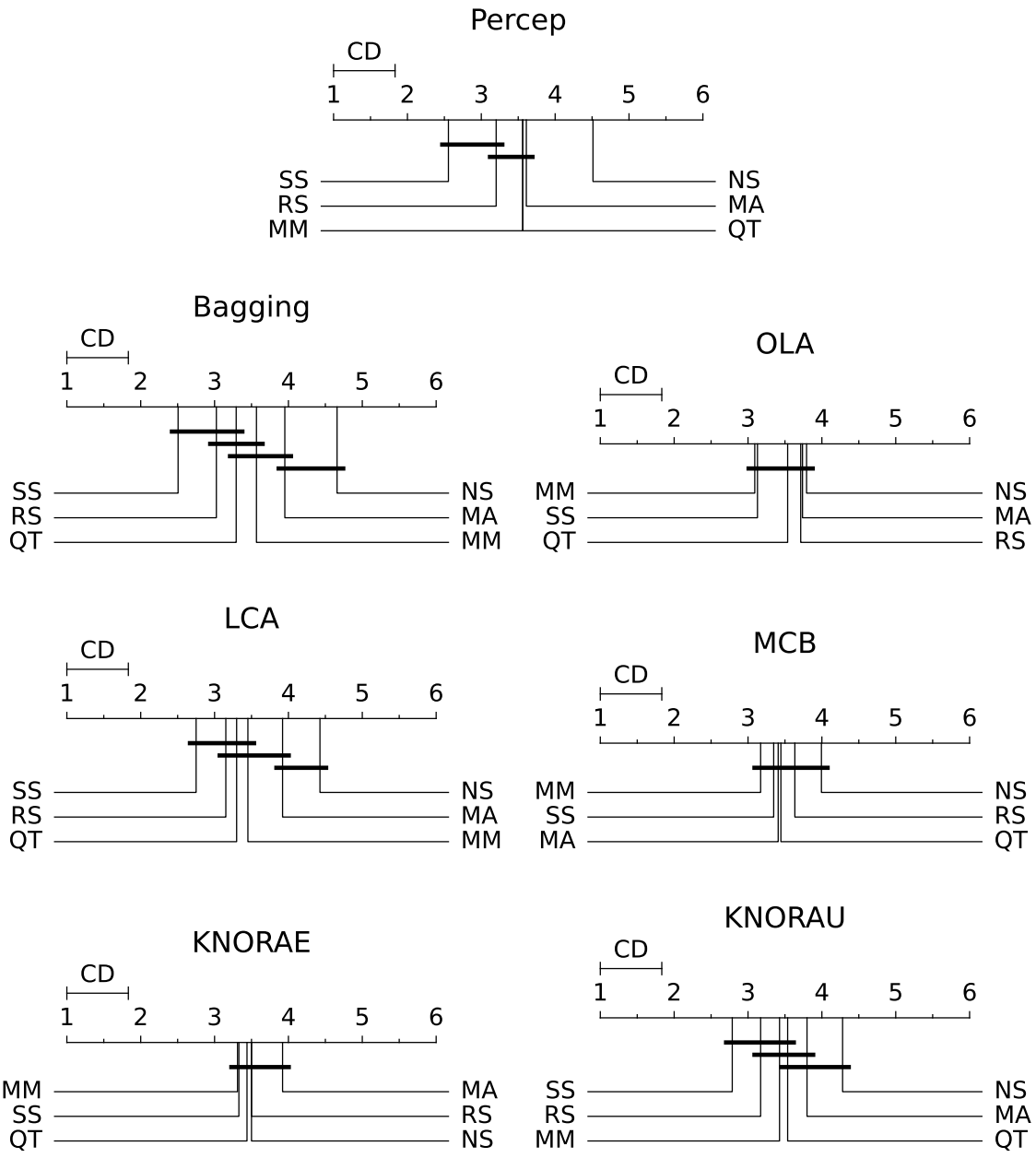


Figure 5 – CD diagrams of average rankings (considering **metric F1**) of the six scaling techniques for the Perceptron model (above) and its corresponding ensembles.

These results can be seen as an indication that when scale matters to the ensemble, monolithic models may dictate the data scale-variability behavior of the ensembles built with them as their base model. It is, therefore, an important finding since, in order to choose the optimal scaling technique for an ensemble, tests can be performed with only one instance of the base model. This incurs significantly less computational cost than running the same tests with the entire ensemble.

Another valuable take from Figures 5 and 6 is that, for all ensemble models, the perfor-

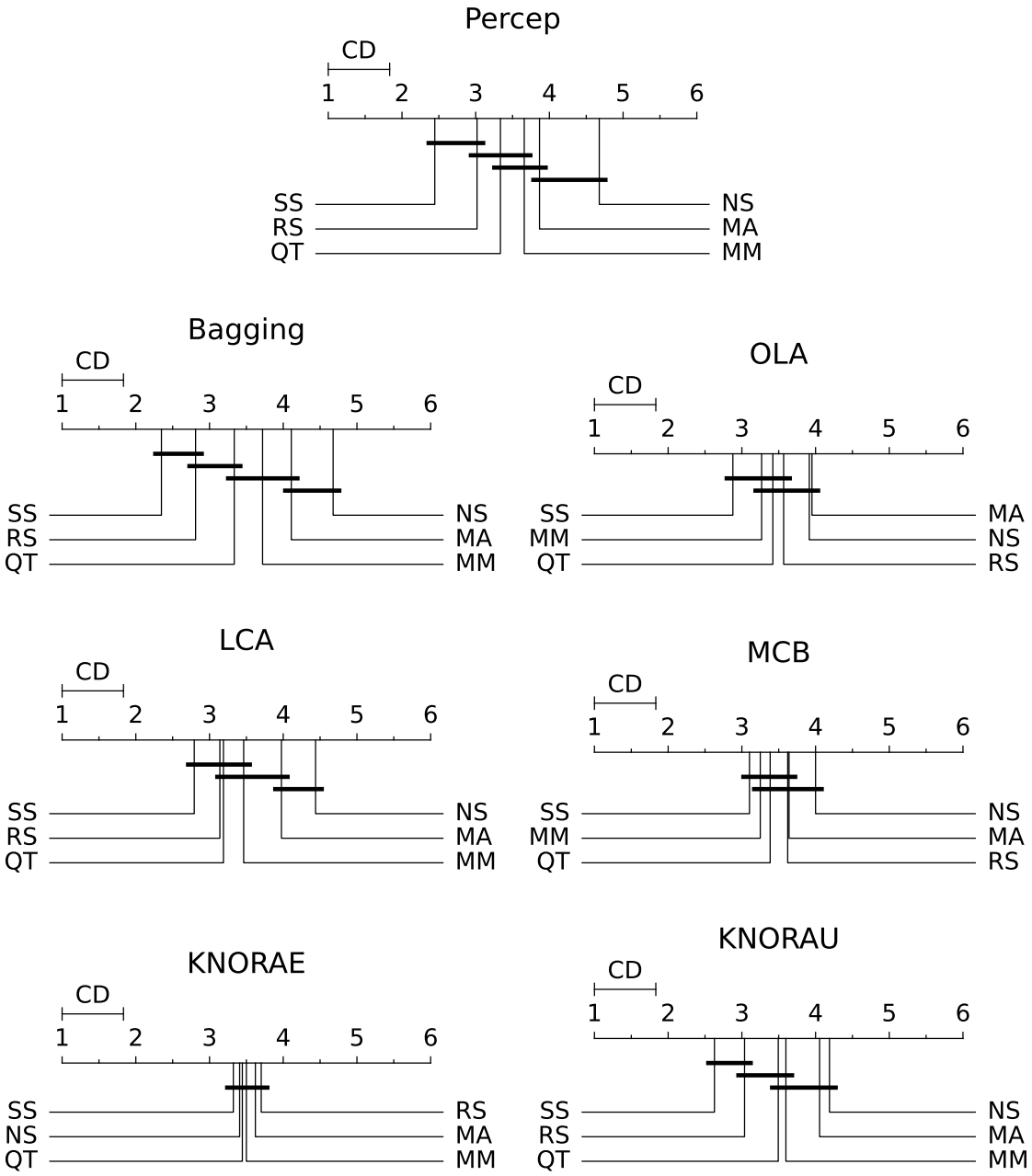


Figure 6 – CD diagrams, using Nemenyi test, of average rankings (considering **metric G-Mean**) of the six scaling techniques for the Perceptron model (above) and its corresponding ensembles.

mance with nonscaled data (NS) is statistically similar to at least one of the adjacent scaling techniques in the ranking, even for those ensembles that are more prone to variation due to scale, such as the Bagging model. Hence, choosing the wrong scaling technique can be as bad as not scaling the data at all.

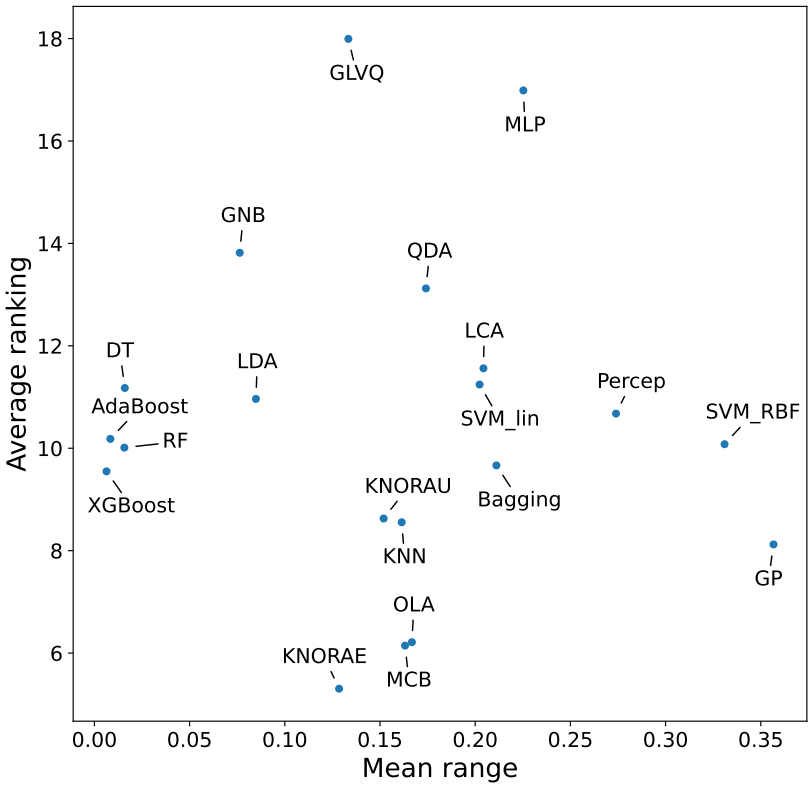
2.5.3 Scale-sensitivity vs. model performance

If we look at both Figure 3 and Table 3 we can see that there are models that yield higher performances, such as KNORAE (close to 0.6 for F1 and 0.7 for G-Mean) and other models that are very sensitive to the choice of scaling technique, such as GP, that presents a mean range close to 0.26 for both metrics. However, it is hard to see from these resources how scale-sensitivity and model performance relate to each other.

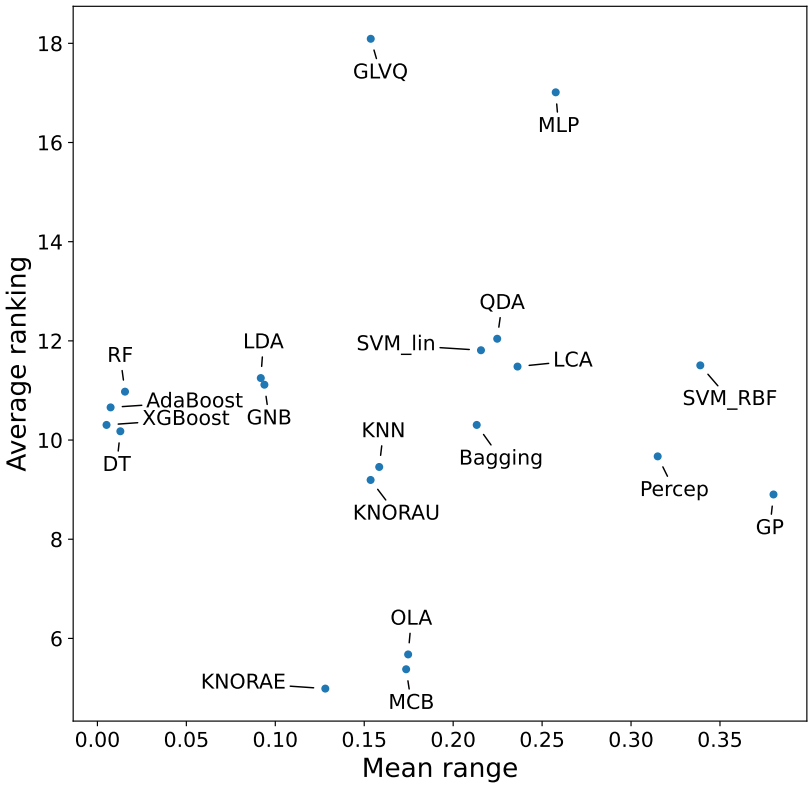
In order to provide a better way to look at this relation, in Figure 7 we present two scatter plots, one for each metric, where we compare the mean range against the average ranking of each model. For the mean range, we consider the ranges (differences between the performances of the best and the worst scaling techniques) obtained by a model over each of the 82 datasets and then calculate their mean. For the average ranking, we calculate the rank of a model compared to the others in each dataset and then we take the mean of all the 82 rankings of that model. Since each model is applied to six versions of the same dataset (one for each scaling technique), for the ranking, we take into account its best performance within the six versions. In these plots, the average ranking is better when it is lower. These scatter plots convey a way to analyze the models' scale-sensitivities and their performances, in terms of their average ranking over the 82 datasets.

This is an important analysis when one needs to select one model or a group of model for deployment in real-world applications with different requirements. For example, dataset scaling is often not an option for a real-time or online application in which model training is executed on the fly. Therefore, one would have to choose a model that performs well regardless of the data scale, such as DT or AdaBoost. In the other end of the spectrum, for a static and more conventional application, when training is done prior to deployment, it makes more sense to choose performance over scale stability, which would lead to the choice of a model such as KNORAE, MCB or OLA.

Another interesting observation from Figure 7 is that the DT model and the DT based ensembles (AdaBoost, RF and XGBoost) appear in a cluster positioned in the middle left portion of the space. This means they are virtually insensitive to scale – agreeing with previous discussions – while presenting a reasonable performance. Additionally, the fact that they are together in a cluster means that there is little difference in choosing one over the others. This is somewhat surprising since we expected that these ensembles, even though static, would present a consistently better ranking throughout the datasets when compared to its base model.



(a)



(b)

Figure 7 – Scatter plots relating model scale-sensitivity (mean range) with its performance (average ranking) considering metrics (a) F1 and (b) G-Mean.

On the other hand, the perceptron model (Percep) and the ensembles built with it as their base model (OLA, LCA, MCB, KNORAE and KNORAU) are more dispersely distributed in the space, with KNORAE being, at the same time, the best performant and the least sensitive to scale when compared to the other perceptron based ensembles. It also stands out that even though Percep and DT have very similar average ranking, their ensembles are positioned very differently in the space, which may be a consequence of their different nature, especially regarding their classifier selection method: static vs. dynamic. This could be further investigated by building dynamic ensembles of DTs as well as static ensembles of perceptrons. We expect that a DT-based KNORAE would be positioned closer to the origin in this graph.

Table 4 – Results for the Friedman hypothesis tests, each set of tests consider a stratum of datasets with different IR levels: low, medium, high and then, all the datasets. Check marks indicate p-values that reject the null hypothesis.

Model	Metric	Low IR	Medium IR	High IR	All datasets
		p-value	p-value	p-value	p-value
SVM_lin	F1	0.0040 ✓	0.0121 ✓	0.0000 ✓	0.0000 ✓
	G-Mean	0.0017 ✓	0.0004 ✓	0.0000 ✓	0.0000 ✓
SVM_RBF	F1	0.0082 ✓	0.7154	0.0001 ✓	0.0000 ✓
	G-Mean	0.0108 ✓	0.5642	0.0001 ✓	0.0000 ✓
KNN	F1	0.2732	0.8664	0.0722	0.0182 ✓
	G-Mean	0.2500	0.9975	0.2172	0.0732
GNB	F1	0.8886	0.1820	0.0001 ✓	0.0000 ✓
	G-Mean	0.8886	0.0823	0.0000 ✓	0.0000 ✓
GLVQ	F1	0.3480	0.7904	0.0000 ✓	0.0027 ✓
	G-Mean	0.7364	0.7423	0.0000 ✓	0.0039 ✓
LDA	F1	0.5364	0.5364	0.2239	0.0132 ✓
	G-Mean	0.5364	0.0005 ✓	0.2239	0.0002 ✓
QDA	F1	0.0016 ✓	0.3045	0.0512	0.0052 ✓
	G-Mean	0.0078 ✓	0.7324	0.0133 ✓	0.0006 ✓
GP	F1	0.0263 ✓	0.2627	0.0000 ✓	0.0000 ✓
	G-Mean	0.0405 ✓	0.1699	0.0000 ✓	0.0000 ✓
DT	F1	0.5288	0.0869	0.0164 ✓	0.1037
	G-Mean	0.4136	0.1139	0.0361 ✓	0.1198
Percep	F1	0.0024 ✓	0.0038 ✓	0.0000 ✓	0.0000 ✓
	G-Mean	0.0000 ✓	0.0217 ✓	0.0000 ✓	0.0000 ✓
MLP	F1	0.0000 ✓	0.0001 ✓	0.0000 ✓	0.0000 ✓
	G-Mean	0.0000 ✓	0.0000 ✓	0.0000 ✓	0.0000 ✓
RF	F1	0.2858	0.5907	0.4050	0.5169
	G-Mean	0.2229	0.5907	0.7535	0.4980
XGBoost	F1	0.0933	0.8063	0.1639	0.9574
	G-Mean	0.0933	0.8063	0.1639	0.9574
AdaBoost	F1	0.2448	0.5662	0.0137 ✓	0.0705
	G-Mean	0.2448	0.5662	0.0137 ✓	0.0705
Bagging	F1	0.0041 ✓	0.0078 ✓	0.0000 ✓	0.0000 ✓
	G-Mean	0.0014 ✓	0.0102 ✓	0.0000 ✓	0.0000 ✓
OLA	F1	0.5834	0.6362	0.0160 ✓	0.0322 ✓
	G-Mean	0.6915	0.9364	0.0002 ✓	0.0011 ✓
LCA	F1	0.0633	0.0309 ✓	0.0000 ✓	0.0000 ✓
	G-Mean	0.0226 ✓	0.0364 ✓	0.0000 ✓	0.0000 ✓
MCB	F1	0.1317	0.8412	0.3877	0.0744
	G-Mean	0.1510	0.4385	0.1434	0.0256 ✓
KNORAE	F1	0.5368	0.3985	0.6109	0.2935
	G-Mean	0.6592	0.2445	0.9499	0.7844
KNORAU	F1	0.2378	0.5939	0.0000 ✓	0.0000 ✓
	G-Mean	0.1257	0.3655	0.0000 ✓	0.0000 ✓
H_0 **Rejections**	F1	7	5	13	14
	G-Mean	8	6	14	14
	Total	**15**	**11**	**27**	**28**

Table 5 – Number of wins per scaling techniques considering all 82 models. Each IR stratum (low, medium and high) consider only the datasets within that stratum. The best result for each row is highlighted in bold.

IR Stratum	NS	SS	MM	MA	RS	QT
All	125.533	168.100	105.883	109.467	172.233	**220.783**
Low IR	12.817	**32.567**	15.733	10.983	23.450	25.450
Medium IR	15.900	19.650	11.983	19.983	22.983	**30.500**
High IR	96.817	115.883	78.167	78.500	125.800	**164.833**

2.6 RELATED WORK

This section chronologically presents a review of the few papers published so far, to the best of our knowledge, that present a comparative analysis of the effects of different scaling techniques on the classification performance of machine learning algorithms. In Table 6 we summarize the scaling techniques, the number of datasets, and the classification models that each of these papers have employed in their experiments. The last line presents the same information regarding the experiment conducted in this paper.

Table 6 – Amplitude of the related works and this paper.

Paper	Scaling techniques	Datasets	Classification Algorithms	
			Monolithic	Ensemble
Jain et al. (6) 2018	**2 techniques:** Min-max, Z-score.	**48**	**1 algorithm:** Gaussian Kernel ELM.	
Dzierżak et al. (7) 2019	**2 techniques:** Min-max, Z-score.	**1**	**4 algorithms:** Naive Bayes, SVM, MLP, Classification via regression.	**1 algorithm:** Random Forests (RF)
Raju et al. (8) 2020	**7 techniques:** Min-max, Z-score, Scale, Robust scaler, Quantile Transform, Power Transform, Max abs scaler.	**1**	**3 algorithms:** SVM (2 variants), KNN.	
Singh et al. (3) 2020	**14 techniques:** Z-score, Min-max (2 variants), Mean centering, Pareto scaling, Variable stability scaling, Power Transform, Max abs scaler, Decimal scaling, Median and Median Abs Deviation Normalization, Tanh normalization (2 variants), Logistic sigmoid, Hiperb. tangent.	**21**	**1 algorithm:** KNN	
Mishkov et al. (4) 2022	**16 techniques:** Z-score, Pos. standardization, Unitization, Min-max, Normalization (3 variants), Pos. normalization (2 variants), Quotient transform. (7 variants).	**4**	**1 algorithm:** MLP	
This paper	**5 techniques:** Min-max, Z-score, Max abs scaler, Robust Scaler, Quantile Transformer.	**82**	**11 algorithms:** SVM (2 variants), KNN, GLVQ, GNB, GP, LDA, QDA, DT, Percep, MLP.	**9 algorithms:** XGBoost, RF, AdaBoost, Bagging, OLA, LCA, MCB, KNORAE, KNORAU.

We can see, from Table 6, that the previous work lacked an analysis that is sufficiently diverse regarding all three perspectives: number of techniques, datasets and classification

algorithms.

Jain et al. (6) and Dzierżak et al. (7), for example, only considered two scaling techniques in their experiments, which were Min-max and Z-score normalization. In the case of Jain et al. (6), authors aim their study on the dynamical selection of one of these two techniques based on data complexity measures. Although this paper reports the performances obtained by the techniques, a comparison is only minimally performed since its not their main focus. Dzierżak et al. (7), assessed the influence of these two scaling techniques on the performance of a model to detect osteoporosis and osteopenia from a single dataset comprising of 290 features representing computerized tomography (CT) images of sections of patients' spine. Their results indicated a superiority of the Z-score method in improving classification performance. However they did not employ a hypothesis test to evaluate the statistical significance of the improvement over the Min-max method.

While Raju et al. (8), Singh et al. (3) and Mishkov et al. (4) all were able to compare significantly larger and more diverse sets of scaling techniques, they all failed to include a diverse set of classification algorithms. Additionally, when it comes to the generality of their findings, only in Singh et al. (3) a relevant number of datasets was considered.

Raju et al. (8) applied seven distinct scaling techniques on a Kaggle diabetes dataset and then compared the performance of three classification models: KNN and two SVM variants. The authors compared the model's results with each scaling technique to the performance attained on original (nonscaled) data. These comparisons showed that scaling allowed a performance increase in every case, ranging from 5% to 10%. Endorsing our findings on the insensitivity of the KNN model to the choice of scaling technique, in their results KNN was the least sensitive when compared to the two SVM variants. In addition to compare the performance obtained with each scaling technique versus that of nonscaled data, it was also performed a comparison of the scaling techniques to one another. However, the paper lacked a more thorough analysis of the results along with hypothesis testing.

Singh et al. (3) employed 14 scaling techniques in 21 datasets and then compared the performance of a k-Nearest Neighbors (KNN) classifier trained on the scaled and nonscaled datasets. The outcomes of the experiments when using the full feature set suggest that most of the methods help improve accuracy, nonscaled data does not always lead to the worst accuracy, which supports our conclusions in Section 2.5.1. Hypothesis tests confirmed that different techniques allow for significantly different results, but only two techniques (Pareto Scaling and Power Transformation) were significantly better than nonscaled data. However,

the authors did not perform an analysis involving classification algorithms other than the KNN.

Finally, Mishkov et al. (4) applied 16 scaling techniques to 4 different datasets in order to compare the classification accuracy of a Multi-layer perceptron (MLP). Their results showed that the choice of scaling technique influences classification accuracy and suggested that this decision must be made on a case-by-case basis, i.e. depending on the dataset being analyzed. They also observed that nonscaled data do not always lead to the worst performance, agreeing with our findings in Section 2.5.1. This paper also lacked an analysis involving other classification algorithms.

To summarize, two of the five papers focus their analysis on just two scaling techniques (6, 7), while those that experiment with a more extensive number of techniques restrict their tests to just one (3, 4), or three classification algorithms (8). Also, only in (6) and (3), authors provided results from a relevant number of datasets. Therefore, the literature lacks research works that consider various scaling techniques, a reasonable number of datasets, and classification algorithms simultaneously. Additionally, we notice that none of these works compared the effect of the scaling techniques in datasets affected by different imbalance ratios, neither measured their impact in ensemble models apart from the traditional Random Forest included in only two of the five papers.

It is also important to mention that, although in the two most recent papers (3, 4) authors employed a significantly larger set of scaling techniques than all the other works (including ours), some of those techniques are too similar or even equivalent to one another. As reported in (3), many of the techniques frequently return the same results for most datasets. Additionally, in (4), the set of techniques is increased largely to the decision of adding up to 7 variants of the same technique. In our paper, we aimed at selecting a diverse but concise set of techniques that we believe are representative enough of the scaling techniques available, as we detail in Section 2.2.

2.7 LESSONS LEARNED

- The choice of scaling technique matters for classification performance. Except for models that are, by nature, not sensitive to scale, the performance difference between the best and the worst scaling technique is relevant and statistically significant in most cases.
- Choosing the wrong scaling technique can be more detrimental to the classification

performance than not scaling the data at all. For the SVM_lin model, for example, the difference in F1 was higher than 0.5 for 32% of the datasets, and higher than 0.8 for 7% of them.

- The difference in performance considering the distinct scaling techniques can be observed across datasets presenting all levels of class imbalance ratio. However, it seems to be more salient for datasets where IR is higher.

- The Standard Scaler technique was the most successful on low IR datasets but the Quantile Transformer technique was better for all other IR strata. It indicates that IR may play an important role on the selection of the best scaling technique for a dataset.

- Models such as SVMs, GLVQ, Gaussian Process, Perceptron, MLP, Bagging, OLA, LCA and KNORA-U are very sensitive to the choice of scaling technique.

- Decision Trees (DT) and DT-based ensembles are virtually insensitive to the choice of scaling technique. On the other hand, ensembles built with Perceptrons present a wider range of sensitivity to the choice of scaling techniques and varying F1 and G-Mean performances.

- For ensembles that are more sensitive to the choice of scaling techniques, such as Bagging, LCA and KNORAU, the scaling techniques tend to rank similarly to their base models. This indicates that, when scale matters to the ensemble, base models may dictate the data scale-variability behavior of their ensembles. Therefore, testing which scaling technique is the best for an ensemble can be done by running only one instance of its base model, incurring in significantly lower computational cost.

- For real-time or online classification applications, in which model training is executed on the fly, with the data streaming into the learning process and therefore are hard to be effectively scaled, models such as DT or AdaBoost are preferable since they present the least sensitivity to scale while attaining reasonable performances.

- For static applications, when one can choose performance over scale stability, KNORAE, MCB or OLA are the most reasonable choices since they are the ones that present the best mean performances given that data is adequately scaled.

2.8 CONCLUSION

This study sought to understand whether the choice of the scaling technique significantly influences the performance of classification models and whether this influence changes according to the types of models used, including monolithic and ensemble models. We performed a broad experiment with five scaling techniques, 82 binary datasets, presenting a wide range of imbalance ratios, and 20 classification models, comprising 11 monolithic algorithms from five different subcategories and nine ensemble algorithms, including static, DCS and DES models.

Results indicated that the choice of scaling technique significantly affects classification performance and that choosing the wrong scaling technique can even be worse than not scaling the data at all. The analysis of scale sensitivity versus model performance revealed the relationship between an ensemble and its base models when we look at their performance variations. It also showed that there is no "one size fits all" when it comes to model deployment, where application constraints, such as the feasibility of scaling and other requirements may determine the more appropriate model. This endorses the importance of such an analysis as a tool for model selection.

In future works, we intend to further investigate this relation between the performance variability due to the choice of scaling technique of an ensemble and its base model, e.g. better understand how an ensemble, when compared to its base model, is able to reduce the performance range considering different scaling techniques and how different configurations and types of ensembles present themselves in the graph in Figure 7. Another direction can be to understand which data characteristics are important to the selection of scaling techniques, which can also subsidize meta-learning research aiming at the dynamic selection of scaling techniques.

3 META-SCALER: A META-LEARNING FRAMEWORK FOR THE SELECTION OF SCALING TECHNIQUES

Lucas B.V. de Amorim, George D. C. Cavalcanti, Rafael M.O. Cruz

This chapter has been published as a paper in February 2024, in the IEEE Transactions in Neural Networks and Learning Systems. DOI: 10.1109/TNNLS.2024.3366615

Abstract

Dataset scaling, a.k.a. normalization, is an essential preprocessing step in a machine learning pipeline. It aims to adjust the scale of attributes in a way that they all vary within the same range. This transformation is known to improve the performance of classification models. Still, there are several scaling techniques (STs) to choose from, and no ST is guaranteed to be the best for a dataset regardless of the classifier chosen. It is thus a problem- and classifier-dependent decision. Furthermore, there can be a huge difference in performance when selecting the wrong technique; hence, it should not be neglected. That said, the trial-and-error process of finding the most suitable technique for a particular dataset can be unfeasible. As an alternative, we propose the Meta-scaler, which employs meta-learning to build meta-models to automatically select the best ST for a given dataset and classification algorithm. The meta-models learn to represent the relationship between meta-features extracted from the datasets and the performance of specific classification algorithms on these datasets when scaled with different techniques. Our experiments using 12 base classifiers, 300 datasets, and 5 STs demonstrate the feasibility and effectiveness of the approach. When using the ST selected by the Meta-scaler for each dataset, 10 out of 12 base models tested achieved statistically significantly better classification performance than any fixed choice of a single ST. The Meta-scaler also outperforms state-of-the-art meta-learning approaches for ST selection. The source code, data, and results from the experiments in this paper are available at a GitHub repository[1].

Keywords: Meta-learning, Scaling techniques, Normalization, Classification.

3.1 INTRODUCTION

In machine learning (ML), dataset scaling is an important preprocessing step. Its goal is to transform numerical features in a way that they vary within the same range. This dataset

[1] http://github.com/amorimlb/meta_scaler

transformation can potentially improve the performance of several classification models (46, 47). This improvement happens mainly because a feature that ranges in a larger scale tends to cause greater influence in the model learning process than one that ranges in a smaller scale.

However, several scaling techniques can be applied to a dataset. Previous research shows that choosing the right one for a specific dataset may not be an easy task, especially when we consider that this choice can significantly impact classification performance and that making a bad choice can even decrease this performance compared to nonscaled data (3, 4, 48).

There is no straightforward and efficient method to select the best ST for a dataset. Some machine learning practitioners carelessly select a "default" technique. In contrast, those who select it thoughtfully must resort to an extensive trial and error endeavor, which becomes even more impractical since the best ST for a dataset may not be generalized for different classification algorithms (48). This procedure is not scalable and may require a prohibitive amount of expert time and effort, making it unfeasible.

As a guide to this study, we pose the following general research question: *Is it possible to build a recommendation system, using meta-learning, that can efficiently predict the best ST for a particular dataset and classification algorithm?* In meta-learning (MtL) approaches, a meta-model can be trained once and then efficiently predict the best algorithm for a new dataset based on the induced knowledge. The idea is to learn the relation between the characteristics extracted from a dataset, called meta-features, and the performances of models on these datasets (49). The meta-features can be simple measures, such as the number of lines or columns, or more complex ones, such as feature skewness, correlations, and the performance of simple classifiers when applied to the dataset.

Notwithstanding, most of the MtL approaches in the literature concentrate on defining a complete preprocessing pipeline (50, 51, 52), which precludes the system from learning what specific dataset meta-features are important for the selection of each single step. Besides, most of them do not take into account different classifier algorithms during the learning process, making it harder to capture the relationship between the steps and the classifier. This leads to meta-models that are general rather than specialized on each step of the pipeline and for specific classifiers, quite possibly reducing their effectiveness in finding the best algorithms for specific preprocessing steps.

As an alternative, we propose the Meta-scaler, a meta-learning framework specifically designed to select the best ST for a pair of dataset and classification algorithm. As novelties, this is one of the first frameworks that can be used to instantiate meta-models that are

trained specifically for each base classifier algorithm in a collection and the first that does this specifically for ST selection, a single step of the preprocessing pipeline. Thus, the Meta-scaler is specialized in selecting an ST for specific base classifiers. Additionally, we evaluate the framework using interpretable meta-models that allow for analyzing the importance of the meta-features. This enables an important discussion into what aspects of a dataset matter most for the selection of ST depending on the classifier.

While the single-step focus of our framework may be seen as a disadvantage against the existing multi-step AutoML systems, it allows the meta-models to be more specialized for the task, achieving better performance. In fact, it has been shown that optimizing each preprocessing step independently, rather than the entire pipeline at once, leads to better performances for pipeline definition, as there seems to be no synergic effect between the steps of the pipeline (53). This indeed matches our findings, as the Meta-scaler outperforms a state-of-the-art multi-step solution. Therefore, this work can lead to the creation of similar frameworks or the adaptation of our proposal, focused on other preprocessing steps. This way, by aggregating several modules that are all specialized and high-performing in their own tasks, one can create an AutoML system that can aggregate the recommendations of each specialized module, opening a path to another type of AutoML system.

Our empirical analysis of the proposal – using 300 diverse public datasets, 12 base classifiers, and 5 STs – shows that considering the final classification performances (attained by the base classifiers in the query datasets), the Meta-scaler yields the highest performance when compared to choosing any of the STs (or the nonscaled data) for 10 out of 12 base models. It also presents the highest mean F1 performance over all base models and the best mean ranking (1.5), visibly superior to the second-best result (3.5) by the Standard Scaler technique. Our analysis shows that the Meta-scaler significantly outperforms any ST used statically, and state-of-the-art MtL ST selection methods. Another interesting finding is that the generated meta-models rely on entirely different subsets of meta-features according to the base model they are specialized for. This indicates that specializing the meta-model according to the base classifier is a promising research direction for meta-learning applied to data preprocessing.

The main contributions of this paper are:

- It proposes a meta-learning framework focused on one step of the data preprocessing pipeline (scaling), as opposed to multiple steps, resulting in meta-models more finely tuned for the purpose.

- The framework makes recommendations that are dataset and classifier-dependent, i.e., the base classification algorithm is also taken into account instead of looking at ST selection as a problem that depends only on the dataset.
- We propose a strategy that trains and employs different meta-models for each base classifier. Therefore, the meta-models learn to recommend the best choice of ST for a pair of dataset and classifier, creating meta-models specialized for each base classifier.
- We empirically show that the proposed Meta-scaler framework is effective in selecting a scaling technique for a dataset, given a classifier, leading to better mean base-level classification performance than any static choice of ST and state-of-the-art MtL ST selection methods.
- It analyzes and discusses how meta-models trained for different base models rely on different meta-feature subsets. The analysis of meta-feature importances contributes to the interpretability of the meta-models.

The rest of this paper is organized as follows: In Section 3.2, we present a short introduction to the main concepts of meta-learning and scaling techniques. In Section 3.3, we introduce our proposed framework. Section 3.4 exhibits a review of the related work. Our proposal, Meta-scaler, is evaluated through experiments described in Section 3.5. Finally, in Sections 3.6 and 3.7, we discuss the results and conclude this research.

3.2 BACKGROUND

3.2.1 Meta-learning

Traditional algorithm selection in machine learning involves a trial-and-error approach, which tests various algorithms on a dataset to determine the best one. This approach is not efficient or scalable for large datasets, especially when the set of candidate algorithms is large (54). The challenge at hand is to discover methods for accurately forecasting the most suitable algorithm for a given dataset without requiring direct experimentation with the data itself.

In general, this is formally known as the algorithm selection problem (ASP) (1, 55, 56). Some important approaches to the ASP have been proposed in the AutoML literature (57, 58, 59, 60), but these studies rely mostly on optimization procedures that have to be executed for each new problem presented, which can make it computationally expensive.

In this scenario, meta-learning (MtL) emerges as a potential solution because it can be used to learn predictive models that can quickly recommend an algorithm for a problem based on dataset meta-features, i.e., its characteristics. To do this, MtL applies machine learning at a higher level, the meta-level. In this case, the learning problem at the meta-level (or the meta-problem) is to model the relationship between the meta-features of several datasets and the performances previously achieved by different algorithms on those datasets. This way, a meta-model learns how to achieve the best learning strategy at the base-level (61, 49), i.e., how to select the best base-level algorithm for a new given problem based on prior experience. This allows learning systems to become more efficient, flexible, and adaptable to new domains or problems (61).

MtL can be applied to various machine learning tasks, such as clustering (62, 63), time series forecasting (64, 65), hyperparameter optimization (66, 67), and classification (68, 69, 56), or even in non-ML domains, such as in tandem with optimization techniques (70, 71). In general, a meta-learner uses these meta-features to model the relationship between them and the performance of algorithms on previous tasks. The trained meta-model can then recommend the best algorithm for new tasks based solely on their meta-features.

Formally, considering a problem space $\mathbb{D}$, an algorithm space $\mathbb{A}$, given a problem (dataset) $\mathbf{D_i} \in \mathbb{D}$, and let $\mathbf{f_{D_i}} = \{f^1_{\mathbf{D_i}}, f^2_{\mathbf{D_i}}, f^3_{\mathbf{D_i}}, \cdots, f^h_{\mathbf{D_i}}\}$ be a vector of h meta-features calculated for $\mathbf{D_i}$, then a meta-learning model λ aims to select an algorithm $\alpha \in \mathbb{A}$ such that its performance p is the maximum in $\mathbb{P}$, the set of performances of all algorithms in $\mathbb{A}$ when applied to $\mathbf{D_i}$. For that, problem $\mathbf{D_i}$ meta-features' $\mathbf{f_{D_i}}$ are given as input to λ, hence $\alpha = \lambda(\mathbf{f_{D_i}})$. This accurately reflects Rice's representation of the ASP (1), depicted in Figure 8.

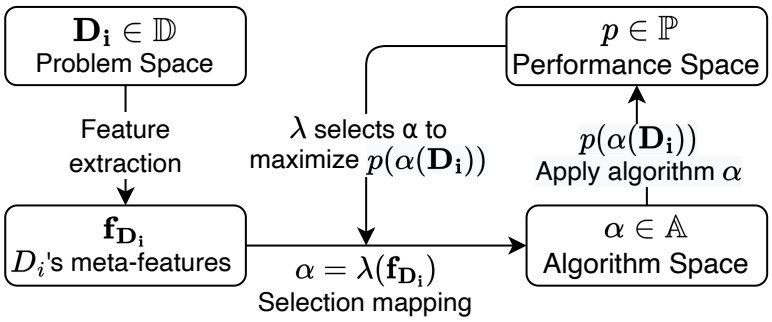


Figure 8 – Diagram of Rice's representation of the Algorithm Selection Problem (1). Adapted from (2).

Rice's representation of the ASP also highlights the three fundamental components of a meta-learning framework: The **meta-features** ($\mathbf{f_{D_i}}$), the **meta-learner** (λ) and the **meta-target**, which is how each algorithm performance $p \in \mathbb{P}$ is represented in the meta-dataset

that aggregates the training examples for the meta-learner. The following subsections describe these three components.

3.2.1.1 Meta-features

Meta-features are standardized measures that encode a dataset's characteristics numerically. They must be uniformly computable for a wide range of problems, have low computational cost, and have an intrinsic relationship with the meta-target (56). These properties allow for the construction of meta-datasets in a uniform feature space, where computing meta-features for a new instance does not cost as much as applying the candidate algorithms to the datasets.

Various meta-feature sets have been proposed for meta-learning research (72, 73, 74), and there have been efforts to standardize sets for specific tasks such as clustering (75), regression (76), time series analysis (65), and classification (77, 78, 79).

Brazdil et al. (80) organize meta-features used in the classification task into four types, to which we add a fifth to accommodate the clustering meta-features:

1. Simple, statistical, and information-theoretic;
2. Model-based;
3. Landmarking (a.k.a. performance-based);
4. Concept and complexity;
5. Clustering.

Simple, statistical, and information-theoretic meta-features: This type of meta-features is mostly composed of measures derived directly from the dependent or independent variables of the dataset. The simple ones typically include measures such as the number of instances, features, classes, and the proportion of discrete attributes. The statistical meta-features include descriptive measures such as features' skewness and kurtosis. The Information-theoretic measures are generally associated with nominal attributes. These include measures of entropy of features and classes (81, 82, 83, 84, 85). Itemset measures proposed by Song et al. (68) are also included in this type.

Model-based meta-features: These are obtained indirectly from the dataset. First, a simple model with low computational cost, often a decision tree, is induced from the dataset, and then

the meta-features are derived from its properties, such as the number of nodes, the number of leaves, branch length, nodes per feature, leaves per class, and leaves agreement (86, 87).

Performance-based (landmarkers) meta-features: Landmarkers are simplified versions of the classification algorithms used to give quick estimates of algorithm performance (88, 72). For instance, a decision stump (the root node of a decision tree) can be used to estimate the performance of decision trees. Additionally, landmarkers can be used to assess important properties from the dataset. For example, a Decision Stump can be used to characterize data separability, a 1NN algorithm to describe data sparsity, a Linear Discriminant to characterize linear separability, and a Naive Bayes to estimate feature independence.

Concept and complexity meta-features: This type of meta-feature is mainly concerned with describing the complexity of the task of classification (89, 73). They comprise (i) concept variation, (ii) overlap of individual features, and (iii) class separability.

Concept variation measures how regular is the output space (90). When neighboring examples of the dataset are labeled with different classes, it increases the irregularity of the output space. Other measures include the non-linearity of the 1NN classifier's decision boundary.

Overlap of individual features is assessed by means of measures such as Fisher's discriminant ratio and Feature efficiency (73). Fisher's discriminant ratio, $\frac{(\mu_1-\mu_2)^2}{{\sigma_1}^2-{\sigma_2}^2}$ where μ_1 and μ_2 respectively represent the mean of features associated with class 1 and class 2, while σ_1 and σ_2 their standard deviation. Feature efficiency calculates the contribution of each feature towards class separation roughly by counting instances that are separable by that feature.

Class separability mainly captures two different aspects proposed by Ho and Basu (73): (i) linear separability of the instances from different classes in the feature space and (ii) whether the two sets of instances come from two different distributions.

Clustering meta-features: This type of meta-features has been mostly proposed and used in the clustering literature. These include measures such as the Calinski-Harabasz index (*ch*) (91), and the Normalized Relative Entropy (*nre*) (75).

3.2.1.2 Meta-learner

This component is responsible for inducing knowledge, allowing the modeling of the relationship between the meta-features and the performance of a classification algorithm, or a *configuration*, when applied to a task (49). In this case, a configuration may comprehend not

only the choice of a classification algorithm but also its hyperparameters and even the data preprocessing steps, such as the scaling technique used. The classifiers most commonly used as meta-learners are instance-based (IB) (92, 93, 94, 68, 95, 96) and Decision Trees (DT) (83, 97, 98). The main advantage of the IB learners is their flexibility to extend the meta-dataset without the need for retraining. However, they tend to give equal importance to all the features and, unlike the DT learners, do not incorporate feature selection. The interpretability of the DT-based models is also an interesting point to consider.

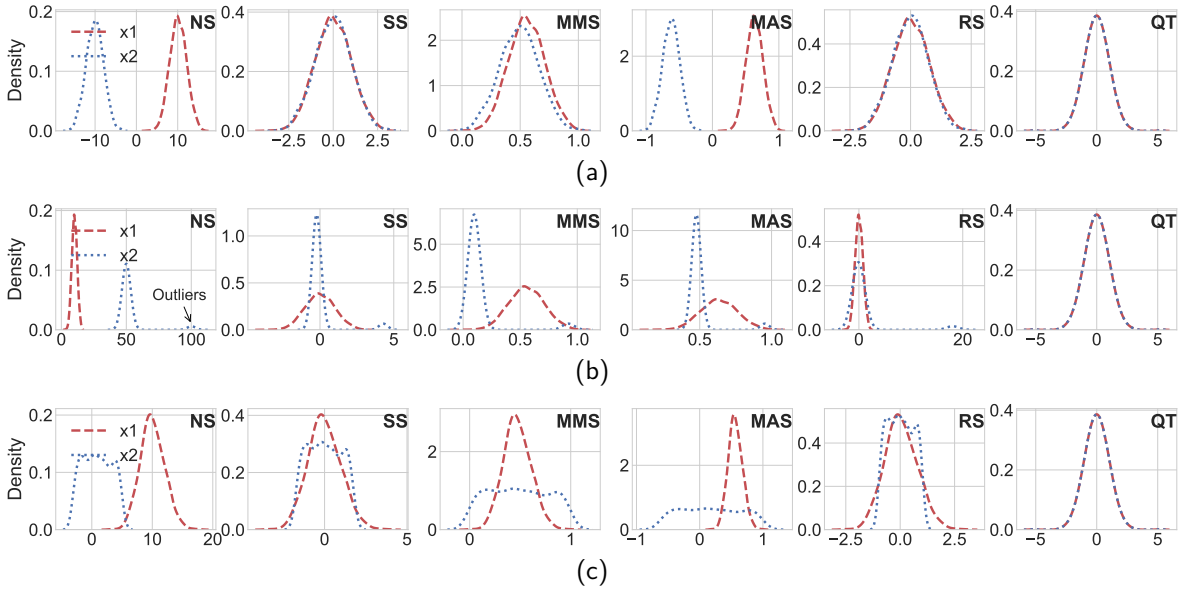


Figure 9 – Effects of the STs when given as input: (a) normally distributed variables with opposing means and no outliers, (b) normally distributed variables where x_2 presents outliers, and (c) variables with different distribution shapes: x_1 has a normal distribution and x_2 has a uniform distribution.

3.2.1.3 Meta-target

The meta-target is the third component of a meta-learning system. It refers to the predicted variable or the system's output type. According to (56), there are three types of meta-targets for algorithm selection: Best Algorithm, Ranked List, and Multiple Algorithms.

In the Best Algorithm approach, given a test instance (dataset), the meta-model must recommend a single algorithm that is expected to have the best performance on that dataset. Consequently, during the construction of the meta-dataset, only the best-performing algorithm for a training instance is assigned as its label. In this case, the learning task is a single-label, multi-class classification problem, which is commonly tackled by rule-based algorithms. One drawback of recommending a single best algorithm is that it does not let the user choose from a set of a few similarly performant and potentially appropriate algorithms for the task (56).

When the meta-target is defined as a Ranked List, the meta-model recommends a list of the algorithms ranked according to their performance. The user is then able to choose one of the top algorithms in this list. This is a particularly interesting approach when: (i) one needs to do a search constrained by a predefined computational budget, where algorithms can be tested from best to worst until the budget limit is reached(99); (ii) To warm start optimization models that tackle the ASP (60). This approach for meta-target definition is generally associated with an instance-based meta-learner (93, 68, 100, 101), where, for instance, a kNN meta-model is used to search for similar tasks in the feature space and then use the best algorithms or configurations in these similar tasks to compose a ranking of algorithms for the query task. The ranking is defined according to task similarity, i.e., tasks closer to the query task imply better ranking for their associated best algorithms. However, other meta-models based on decision trees have been demonstrated to work well for ranked list meta-targets (49), especially methods based on ensembles of decision trees (102, 103).

Finally, when the meta-target is Multiple Algorithms, the meta-model is designed to recommend a set of algorithms with similar predicted performance in the task, i.e., with no significant difference in performance. In this case, the user can select any of the recommended algorithms. In (97), the authors employed this approach using a decision tree-based algorithm, C5.0, to generate rules that describe which types of algorithms are better suited for each task.

3.2.2 Scaling techniques

Scaling techniques (STs) are commonly used in machine learning to deal with issues associated with the different scales in which the various dataset features are presented. When features vary within different ranges, algorithms can learn less accurate models. For instance, for algorithms that work based on distances within instances in the feature space, such as Support Vector Machines, one feature with a much wider scale will cause much more influence in the distance calculations than those that vary within narrower intervals. STs address this problem by standardizing the dataset features so that each varies within the same range. This is one of the preprocessing steps that need to be undertaken before applying a machine learning model to a dataset. STs are more commonly called normalization techniques in the machine learning literature (3, 4, 6, 7, 8). However, the term scaling has a more general meaning, encompassing both normalization or standardization and simpler scaling methods.

Most STs can be broken down into these two components: a translational term and a

scaling factor. Suppose one wants to transform a vector x; then, each one of its components x_i is transformed into x'_i by the formula $x'_i = (x_i - T)/S$, where T represents the translational term and S the scaling factor.

The translational term moves data along the X-axis, while the scaling factor makes data more concentrated or spread out horizontally. Some simpler techniques may not include one of these two operations. For example, the Mean Centering technique simply subtracts the vector's mean from each of its components. While it effectively removes the offset from the data, it fails to equalize data variances across the different features of a dataset. More elaborate techniques promote this equalization by multiplying the data by a scaling factor.

This study uses five well-known, diverse, and commonly used STs: Standard Scaler (SS), a.k.a. Z-score normalization, Min-max Scaler (MMS), Maximum Absolute Scaler (MAS), Robust Scaler (RS), and Quantile Transformer (QT). According to the literature, these techniques are diverse and non-redundant (3, 4). Figure 9 presents examples of how these techniques scale pairs of attributes, providing visual feedback on how they work. The original data distribution is always shown in the leftmost graph.

Figure 9a shows an example where both variables follow normal distributions with chosen means and variances. The intent is to see how the STs deal with variables on opposite sides of the X-axis. Figure 9b shows how the STs' effects on a variable with outliers compare to their effects on a variable without outliers. Here, the two variables also follow normal distributions, but some outliers are added to one of them. Figure 9c shows the effects of the techniques on attributes with different distribution shapes. In this figure, the first column represents the nonscaled data (NS) given as input, and the remaining columns represent the output of each ST.

Standard Scaler, Min-max Scaler, and Maximum Absolute Scaler are the simplest STs. Standard Scaler scales data so that it has a mean of 0 and a standard deviation of 1. Min-max Scaler scales data to a fixed range, usually between 0 and 1. Maximum Absolute Scaler scales data so that the maximum absolute value of each feature is 1. The Robust Scaler is useful when the dataset contains outliers. It scales the data according to the interquartile range and makes it centered on the median. The purpose of the Quantile Transformer, which is also robust against outliers, is to map data to a *uniform* or a *normal* distribution (in our experiments, we use the *normal* transformation), which is useful when the data is not normally distributed. However, since the Quantile Transformer is a non-linear transformation, it may corrupt linear correlations between variables measured at the same scale.

For more details on these scaling techniques and their main differences, advantages, and impact on the performance of different classification algorithms, the reader can consult our previous work (48).

3.3 PROPOSED FRAMEWORK

As previously declared, this paper proposes a meta-learning framework to automatically select the best ST for a specific pair of dataset and classification algorithm. For ease of reference, we name this system as the Meta-scaler.

Figure 10 presents a flow diagram that depicts the steps that compose both the construction and the recommendation phases. Both phases and all the steps therein are detailed below.

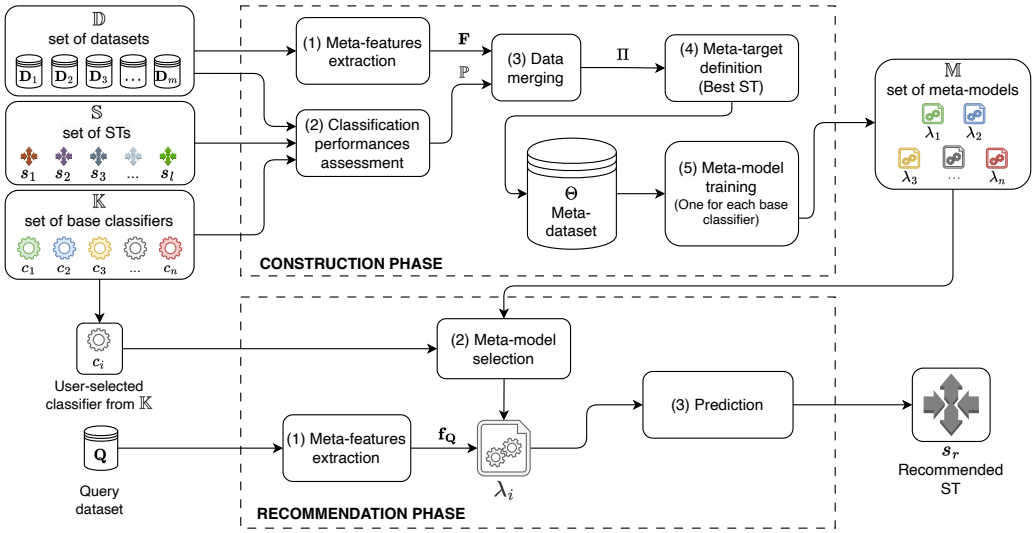


Figure 10 – Proposed Meta-scaler framework.

Based on the results of our previous work (48), where we have verified that the best ST for a dataset changes depending on the classifier used, we hypothesize that training different meta-models that are specialized for each classifier will have a positive impact in the final performance. Thus, in this framework, a different meta-model instance is trained for each classification algorithm.

Note that, in a real-life scenario, the Meta-scaler is employed as a pre-trained ST recommender, i.e., the construction phase is executed only once, prior to deployment. Therefore, given a new dataset and a chosen base classifier, only the recommendation phase needs to be executed.

3.3.1 Construction phase

The construction phase is where the meta-dataset is built and a classification algorithm is employed to build meta-models by learning the relationship between the meta-features and the meta-target in the meta-dataset. As outlined in Figure 10, this phase, which can be broken down into the five steps detailed ahead, takes the below inputs:

- $\mathbb{D} = \{\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3, \cdots, \mathbf{D}_m\}$ – The set of training datasets.
- $\mathbb{S} = \{s_1, s_2, s_3, \cdots, s_l\}$ – The set of scaling techniques.
- $\mathbb{K} = \{c_1, c_2, c_3, \cdots, c_n\}$ – The set of base classifiers.

Where $\mathbf{D_i}$ is a matrix that includes both the independent $\mathbf{X_i}$ and the dependent $\mathbf{y_i}$ variables in the i-th dataset. The elements of $\mathbb{D}, \mathbb{S}$, and $\mathbb{K}$, must be chosen with diversity in mind, as this will contribute to the generalization power of the meta-models created.

Step (1) - Meta-features extraction

A standardized set of h meta-features is extracted from every $\mathbf{D}_i \in \mathbb{D}$. Formally, we represent this as the $m \times h$ matrix $\mathbf{F} = \{\mathbf{f}_{\mathbf{D}_1}, \mathbf{f}_{\mathbf{D}_2}, \mathbf{f}_{\mathbf{D}_3}, \cdots, \mathbf{f}_{\mathbf{D}_m}\}$ where $\mathbf{f}_{\mathbf{D}_i} = \{f^1_{\mathbf{D}_i}, f^2_{\mathbf{D}_i}, f^3_{\mathbf{D}_i}, \cdots, f^h_{\mathbf{D}_i}\}$ is the meta-feature vector that represents $\mathbf{D}_i$ in the meta-problem.

Step (2) - Classification performances assessment

The classification performance of each pair of ST and classification algorithm is evaluated on every dataset. This results in the set of performances $\mathbb{P} = \{p_{\mathbf{D}_i,s_j,c_k} | \forall \mathbf{D}_i \in \mathbb{D} \wedge \forall s_j \in \mathbb{S} \wedge \forall c_k \in \mathbb{K}\}$. To simplify the notation we will refer to $p_{\mathbf{D}_i,s_j,c_k}$ as $p_{i,j,k}$ and omit the domains in $\forall \mathbf{D}_i \in \mathbb{D} \wedge \forall s_j \in \mathbb{S} \wedge \forall c_k \in \mathbb{K}$ reducing it to $\forall \mathbf{D}_i, s_j, c_k$, similar omissions will be done in other points.

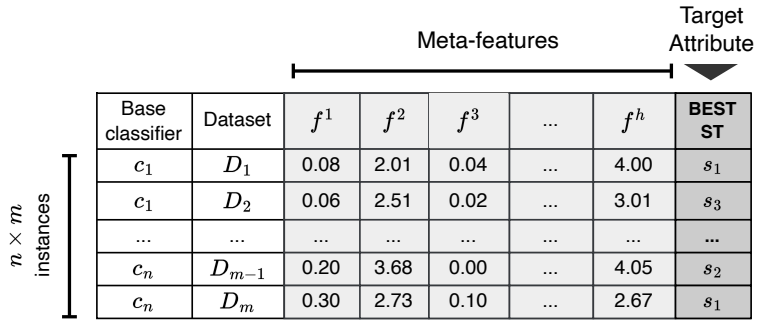
It is recommended to evaluate performances through a cross-validation procedure for stable and reliable results (99, 104).

Step (3) - Data merging

Next, the meta-features $\mathbf{f}_{\mathbf{D}_i}$ calculated for each dataset $\mathbf{D}_i \in \mathbb{D}$ and the performances $p_{i,j,k}$ of each pair of classifier and ST on those datasets are merged. The result is the set $\Pi = \{(\mathbf{f}_{\mathbf{D}_i}, p_{i,j,k}) | \forall \mathbf{D}_i, s_j, c_k\}$.

Step (4) - Meta-target definition

In the Meta-scaler framework, the meta-target is defined via the Best Algorithm approach, as the vector $\beta = \{\beta_{\mathbf{D}_i,c_k} | \forall \mathbf{D}_i, c_k\}$ (domains omitted) where $\beta_{\mathbf{D}_i,c_k} = \arg\max_j p_{i,j,k}$ which represents the best ST for the pair $(\mathbf{D}_i, c_k)$. Hence, the final meta-dataset,



Base classifier	Dataset	f^1	f^2	f^3	...	f^h	BEST ST
c_1	D_1	0.08	2.01	0.04	...	4.00	s_1
c_1	D_2	0.06	2.51	0.02	...	3.01	s_3
...	...	...	...	...	...	...	...
c_n	D_{m-1}	0.20	3.68	0.00	...	4.05	s_2
c_n	D_m	0.30	2.73	0.10	...	2.67	s_1

Figure 11 – A representation of the resulting meta-dataset Θ.

The Best Algorithm approach was chosen because it allows the creation of a system capable of recommending a single ST, which is more convenient to the user since no further experimentation and decision-making are necessary for the ST selection step. An alternative would be to recommend a ranking of two or three STs (reducing the user's ST search space up to 50%), but the experiments in Amorim et al. (2023) (48) show that the first, second, and third-ranking techniques are very often statistically equivalent.

Step (5) - Meta-model training

This step is where the meta-model is finally built. By learning from the pairs in meta-dataset Θ, it models the relationship between the meta-features $\mathbf{F}$ and the meta-targets β. As a result, a set $\mathbb{M}$ of meta-models is generated. Note that a different meta-model λ_i is

trained for the meta-dataset instances Θ_i corresponding to each base classifier c_i in $\mathbb{K}$, hence $|\mathbb{M}| = |\mathbb{K}| = n$.

3.3.2 Recommendation phase

The recommendation phase happens when the Meta-scaler is put into production, i.e., when it is requested to recommend the most suitable ST s_r, after being given as input a new query dataset $\mathbf{Q}$ and the classifier $c_i \in \mathbb{K}$.

Step (1) - Meta-features extraction

In order to recommend the ST to be used given a classifier and query dataset, the same set of standardized meta-features used in the construction phase must be extracted from this new dataset. Hence, this first step results in the vector $\mathbf{f_Q}$ that is given as input to the next step. It is important to notice here that since this meta-feature extraction will always be done when a new dataset is presented, it must be computed at a relatively low cost, such that it makes sense to employ the meta-learning system instead of exhaustively testing all the possibilities.

Step (2) - Meta-model selection

In this step, the meta-model λ_i, that was trained with the meta-dataset component Θ_i is selected for the prediction step. In order words, the meta-model selected is the one that is specialized for the base model provided as input to the recommendation phase.

Step (3) - Prediction

Finally, in the prediction step, the meta-model outputs s_r, the recommended ST.

3.4 RELATED WORK

Within the literature related to algorithm recommendation, there are two prominent approaches: optimization and meta-learning. In optimization, various iterations are executed for every new dataset for which one looks for the optimal algorithm. In MtL, a complex inductive

process is executed only once and generates a model that can then be employed to predict the optimal algorithm for any new query dataset. As opposed to the iterative process that happens in the optimization approaches for each query dataset, the predictive step in MtL approaches is computationally inexpensive. This is the main distinction between these approaches that define the scope of this section, where we focus on the recommendation of preprocessing techniques via meta-learning.

We explore the literature regarding MtL applied to data preprocessing in general because very few studies specifically apply MtL to select scaling techniques. A summary of the related work described here and their relevant characteristics are presented in Table 7. Bilalli et al. (50), Gemp et al. (51), and Zagatti et al. (52) have proposed employing meta-learning to define the whole data preprocessing pipelines. Only Jain et al.(6) present an MtL approach specifically for the selection of a scaling technique, similarly to our proposal.

Table 7 – Scope of related works. Where NOR – Normalizer, PT – Power Transform (PT) and N/I – Not informed.

Paper	**Recommends**	**Scaling Techniques**	**Num. of base models**	**Meta-model type**
Bilalli et al. (2016) (50)	Preprocessing pipeline	MMS, SS.	5	Multiple, one for each base model.
Gemp et al. (2017) (51)	Preprocessing pipeline	N/I	N/I	N/I
Jain et al. (2018) (6)	Scaling Technique	MMS, SS.	1	Single
Zagatti et al. (2021) (52)	Preprocessing pipeline	MMS, SS, NOR.	1	Multiple, depends on dataset constraints.
Meta-scaler (ours)	Scaling technique	MMS, SS, MAS, RS, QT.	12	Multiple, one for each base model.

In Bilalli et al.(50), the focus is the automatic definition of a sequence of transformations, or preprocessing steps, better suited for a pair of dataset and machine learning model given as input. The selected transformations are discretization, binarization of nominal values, scaling, missing values imputation, and principal component analysis. After applying each transformation to a dataset, the meta-features are extracted again and used to predict the model's performance after the change. Experiments on hundreds of datasets, including five different models, show that applying the recommended transformations improves, most of the time, the final performance of the models. In these experiments, similarly to our proposal, a different meta-model is trained for each base classifier. However, the focus of this work is on modeling the relationship between a pipeline of transformations and a classifier instead of the relationship between a dataset's meta-features and a pair of a specific transformation (ST)

and a classifier, as happens in our proposal. As its main limitations, this paper lacks a greater number of base classifiers and possibilities for each transformation considered within the search space. Regarding the STs, for instance, only two possibilities are included in the search space.

Gemp et al.(51) take a standard meta-learning approach, using 22 meta-features and more than 1600 datasets from different sources, to train and evaluate an automated data cleansing solution. However, the main focus of their paper is on the definition of an appropriate distance metric measured between the datasets in the meta-features space such that datasets that are nearer, w.r.t. a chosen distance metric, also rank similarly in terms of final model performance. The authors do not specify the data preprocessing tasks they included in their search space nor the nature or quantity of base and meta-models essayed.

In Zagatti et al.(52), the MetaPrep meta-learning system is proposed to recommend data preprocessing pipelines, focusing on four tasks: imputation, categorical to numerical transformation, scaling, and class balancing. It defines 180 different pipelines and uses statistical and information theory meta-features to train a Nearest Neighbor (1NN) meta-model to predict a ranking of the five best pipelines for a dataset, given a fixed classification algorithm. Different instances of the 1NN meta-model are trained for subsets of the meta-dataset that contain datasets with similar constraints, such as whether they contain features with missing values or categorical data. According to the authors, this strategy ensures that the proposed pipelines take these constraints into account. The MetaPrep system achieved accuracy rates similar to competing approaches when tested with a Random Forest base classifier. This limited choice of a single base classifier algorithm is an important limitation of this work, especially since Random Forests are known to be less sensitive to the scale of attributes (48).

Finally, Jain et al.(6) apply meta-learning to the task of specifically selecting a scaling technique for a dataset, given a fixed base classification model (Gaussian Kernel ELM). A single meta-model[2] learns from 12 data complexity meta-features extracted from 48 datasets drawn from KEEL repository (105) to determine the best of two STs (Min-max and Z-score, a.k.a. Standard scaler) for each dataset. It was reported that the meta-learning approach yielded significantly superior classification performance when compared to using either Min-max or Z-score normalization for preprocessing all the datasets. Nonetheless, the scope of the experiments is very limited, with just two STs and a single base classifier, precluding further generalization of the results.

[2] Fourteen different classification algorithms were evaluated as meta-models, but they are employed as separate systems, hence we consider this a single meta-model approach.

While the approaches just described were important for the advancement of the field of meta-learning applied to preprocessing, when it comes to the specific task of data scaling, most of them have contributed in a very limited way. For instance, the maximum number of STs in their search space was just 3. Furthermore, the analysis in (6) and (52) rely on the results of a single classification algorithm. Hence, the literature still lacks studies focused on the automatic selection, via meta-learning, of each single preprocessing step, which has the potential to allow a more effective algorithm recommendation and more thorough investigation, ultimately resulting in insights that can certainly contribute to the meta-learning and AutoML fields. In our proposal, we address the problem by designing a more flexible framework that generates specialized meta-models for recommending the ST, finely tuned for the base classifier being used. We also evaluate the proposed algorithm with a broad choice of datasets, STs, base models, and meta-models.

3.5 EXPERIMENTAL PROTOCOL

In this section, we describe the methodological procedure employed to empirically assess the effectiveness of the Meta-scaler.

3.5.1 Datasets

The collection of datasets used in this study is composed of 300 binary datasets originating from the data made available during The Landscape Contest at ICPR 2010 (106). We chose this collection because it was designed to contain datasets showing varying levels of complexity (107), which conveniently yields a higher degree of diversity among the datasets. We used the 300 binary labeled datasets from this collection (D1 to D300). By using this diverse collection of datasets, we aim to strengthen the meta-models' ability to generalize. This allows the deployment of the trained meta-models on problems from several unseen domains.

3.5.2 Evaluation procedure

For this experiment, we employed the leave-one-out cross-validation (LOOCV) procedure, where, at each iteration, one dataset ($\mathbf{Q}$) is left for testing, and the remaining ones are used to construct the meta-dataset Θ used for training the meta-models. Hence, for each iteration of the LOOCV, the procedure follows the steps of the proposed framework in Figure 10.

3.5.3 Base classifier algorithms

The set $\mathbb{K}$ of base classifiers used in this experiment is composed of 12 diverse algorithms from different families and types (ensemble and non-ensemble): Support Vector Machine (108) (SVM_RBF, using the RBF kernel and SVM_lin using the linear kernel), Bagging (29), Perceptron (Percep) (109), Gaussian Process (GP) (110), Multilayer Perceptron (MLP) (111), Generalized Learning Vector Quantization (GLVQ) (112), Local Class Accuracy (LCA) (33), Multiple Classifier Behavior (MCB)(34), Overall Local Accuracy (OLA) (33), k-Nearest Oracles - Eliminate (Knora-E), and k-Nearest Oracles - Union (Knora-U) (36). This is a diverse set of algorithms, belonging to different families and including single models, ensemble and dynamic ensemble models (113). It is a subset of the one used in (48), with the same hyperparameters as described therein. This subset was chosen because, in (48) they showed the highest differences in performances under the use of different STs, i.e., these models are more sensitive to the choice of ST.

3.5.4 Scaling Techniques

The set of STs used in this study is composed of the 5 STs described in Section 3.2 plus NS, which simply leaves the data as is, nonscaled. Hence, we can define $\mathbb{S} = \{$NS, SS, MMS, MAS, RS, QT$\}$.

3.5.5 Classification performance assessment

To obtain the base-level classification performances set $\mathbb{P}$ (Section 3.3.1 - Step 2), we evaluated the performances of the pairs of classifiers and STs (under the F1 metric) using 5-fold cross-validation (CV). We split each dataset in the collection into 5 parts to enable the use of 5-fold cross-validation (CV) in a reproducible manner. The resulting pre-split data is available in the paper's GitHub repository. To avoid the data leakage problem (114), the scaling was applied as follows: for each iteration of the 5-fold CV, the STs were fitted only for the training data and then used to transform both training and test sets.

3.5.6 Meta-features

Two open-source libraries were used for extracting the meta-features from the datasets: PyMFE (78) (111 meta-features) and ImbCoL (115) (22 meta-features). This results in 133 meta-features for each dataset. One of the original 112 PyMFE meta-features could not be used in this work because it returned invalid (undefined) results for all of the datasets: num_to_cat. It happened because this meta-feature tries to calculate the ratio of numerical over categorical attributes, but our datasets are all numeric only. Another meta-feature, sd_ratio, also returned undefined results, but only for a maximum of 84 rows out of 3600 (2.4%), where $3600 = |\mathbb{D}| \times |\mathbb{S}|$; hence we decided to keep it and use a kNN imputer (with $k = 2$) to fill the gaps. A brief description of the resulting set of meta-features used in our meta-dataset can be seen in the supplementary material. Readers interested in more information on these meta-features should refer to (78) and (115).

3.5.7 Meta-models

To build the set of meta-models $\mathbb{M}$, we began with the task of selecting an appropriate classification algorithm for the meta-models. We restricted our choices to decision trees based models as they are less sensitive to the scale of the attributes, i.e., the meta-features (48, 116, 117), and because of their interpretability, competitive performance, and low computational costs (104). Furthermore, Decision Trees implicitly perform feature selection, which is useful given that we use a large set of meta-features, and the literature has shown that these algorithms are able to perform well for the meta-learning tasks (98).

We evaluated the performances of meta-models built with these 8 ensemble algorithms: Random Forests (RF), Extra Trees (ET), XGBoost (XGB), Bagging (BAG), Knora-E (KNE), Knora-U (KNU), META-DES (MTD)(44) and DES-MI (DMI)(118). Where, for all of them, the pool of base classifiers was generated with the Bagging algorithm, from the Scikit-learn library(17) that, in its turn, was built with the following configuration: the base estimators are 100 DT classifiers; the proportion of samples and features drawn from the training set for each base estimator was respectively 50% and 100%. For details, the reader may consult the supplementary material and the source code. After comparing their results, we selected the DES-MI algorithm, which showed slightly superior meta-model performance (details in the supplementary material), and proceeded to further analysis of the Meta-scaler considering

only this meta-model. The implementation of the DES-MI used was the one available from the DESlib library (28).

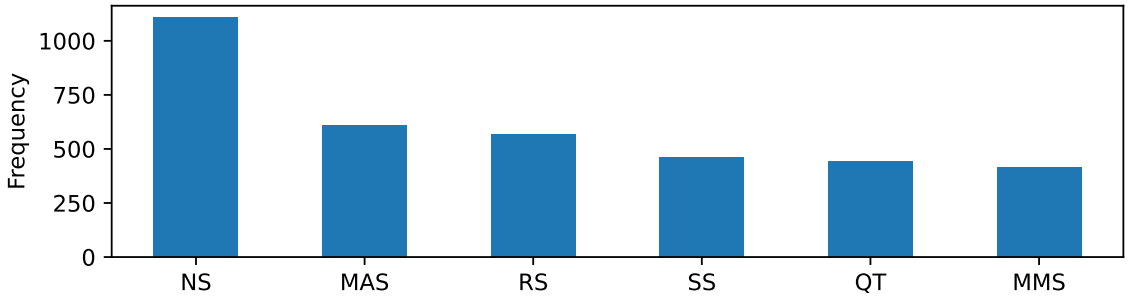


Figure 12 – Frequencies of the classes within the meta-dataset. NS: Nonscaled, MAS: Maximum Absolute Scaler, RS: Robust Scaler, SS: Standard Scaler, QT: Quantile Transformer, MMS: Min-max Scaler.

Note that, since $|\mathbb{S}| > 2$, the resulting meta-problem is a multi-class classification task, as the meta-target can be any of the six elements in $\mathbb{S}$. Analyzing the resulting class frequencies, shown in Figure 12, we can see that the meta-dataset is only slightly imbalanced, with an imbalance ratio (IR) (from most to less frequent class) of 2.67. Since the IR is relatively low, we did not employ any oversampling or undersampling techniques. Instead, we opted to tackle this problem by attributing different weights to the classes during the training process of the DT classification models within the ensemble used as the meta-model. The weights were set to be inversely proportional to the class frequencies. Additionally, the DES-MI algorithm was specifically designed to deal with multi-class imbalanced problems, which may explain why it performed well.

3.5.8 Software

This experiment was executed in the Python programming language, version 3.8.10, using the Scikit-Learn (17) library, version 1.1.2, PyMFE (78) version 0.4.1, DESlib (28) version 0.3.5 and XGBoost (27) version 1.0.2 and the ImbCoL R library (119), version 1.0.1.

3.6 RESULTS AND DISCUSSION

3.6.1 Meta-model performances

The meta-model (or meta-level) performance is the performance achieved by the meta-model in predicting the meta-target for the query datasets. The meta-level performance attained by the Meta-scaler for each base model is shown in Table 8 in both the accuracy and

F1-macro metrics.

Table 8 – Meta-scaler performance (meta-level) for each base-model (using the DMI meta-model). Values obtained w/ LOOCV.

	Bagging	GLVQ	GP	KNORAE	KNORAU	LCA	MCB	MLP	OLA	Percep	SVM_RBF	SVM_lin	Median	Mean	Std-dev
Accuracy	0.3800	0.7900	0.5133	0.3233	0.4167	0.2900	0.4100	0.5500	0.3833	0.2333	0.6233	0.5500	0.4133	0.4553	0.1567
F1-macro	0.3389	0.5235	0.4540	0.3067	0.3936	0.2873	0.3741	0.4440	0.3523	0.2048	0.5322	0.4043	0.3839	0.3846	0.0958

We decided to include the accuracy metric here because it allows us to compare the meta-model performance against a random ST selector. Since $|\mathbb{S}| = 6$ in this experiment, a random selector would have a $1/6$ chance of picking the appropriate ST for a given dataset, which would yield an accuracy of 0.1667. In Table 8, we can see that the minimum accuracy, which occurred for base model Percep, was 0.2333, which is considerably superior to that of the random selector. When we take into account the summary measures (median and mean), we can see accuracy values above 0.4, more than double the accuracy of a random choice of ST. This is already strong evidence of the effectiveness of this meta-learning approach to the selection of STs.

In Table 8, we can emphasize the higher meta-model performances for the base models GLVQ, GP, MLP, SVM_RBF, and SVM_lin. The Meta-scaler achieved accuracy above 0.5, and F1-macro of at least 0.45 for all these base models.

3.6.2 Classification performances

When considering a meta-learning approach, one important analysis is to compare the base-level performance of the models with and without the meta-model influence. This analysis allows us to assess if the meta-learning approach effectively improves performance at the base level. In our case, we compared the classification performance achieved by the base models when the Meta-scaler selects the ST to that of the classification performance achieved when a single ST is selected to be applied to all the datasets. In Table 9, we show these performances (measured by the F1 metric) and also the *Truth* performance, which is the classification performance achieved when the best ST per dataset is always selected. The *Truth* performance can be understood as the upper limit of the Meta-scaler performance. To make it possible to represent the performances over all 3600 meta-dataset instances, we grouped the performances by base model – totaling 300 values per base model – and represented them by their means in Table 9.

Note, in Table 9, that the Meta-scaler presented the highest median (0.7276) and mean

Table 9 – Mean classification performance (F1) of base models over the 300 datasets when each ST, the Meta-scaler (MS), or the *Truth* is employed. For each row, the best result is in bold. The last row shows the resulting p-values after comparing the performances of each ST versus the Meta-scaler using the Wilcoxon Signed-rank test. NS: Nonscaled, SS: Standard Scaler, MMS: Min-max Scaler, MAS: Maximum Absolute Scaler, RS: Robust Scaler, QT: Quantile Transformer.

	NS	**SS**	**MMS**	**MAS**	**RS**	**QT**	**MS**	**Truth**
Bagging	0.6736	0.7142	0.7094	0.7046	0.7143	0.7119	**0.7170**	0.7311
GLVQ	0.6329	0.6223	0.6336	0.6520	0.6167	0.6025	**0.6576**	0.6642
GP	0.7409	0.7097	0.7303	0.7360	0.6721	0.6998	**0.7522**	0.7671
KNORAE	0.7244	0.7210	0.7214	0.7293	0.7371	0.7185	**0.7434**	0.7593
KNORAU	0.7064	0.7186	0.7125	0.7169	0.7324	0.7145	**0.7382**	0.7486
LCA	0.6192	0.6873	0.6523	0.6436	**0.6905**	0.6839	0.6788	0.7207
MCB	0.7190	0.7197	0.7179	0.7267	0.7327	0.7162	**0.7431**	0.7600
MLP	0.6675	0.6886	0.6518	0.6425	0.6683	0.6833	**0.7008**	0.7358
OLA	0.7182	0.7200	0.7204	0.7273	0.7343	0.7151	**0.7437**	0.7597
Percep	0.5932	**0.6889**	0.6510	0.6420	0.6876	0.6864	0.6702	0.7249
SVM_RBF	0.6735	0.7248	0.7242	0.7236	0.7342	0.6999	**0.7399**	0.7483
SVM_lin	0.7048	0.7027	0.6953	0.6925	0.7026	0.6994	**0.7071**	0.7143
Median	0.6892	0.7120	0.7110	0.7107	0.7084	0.6999	**0.7276**	0.7421
Mean	0.6811	0.7015	0.6933	0.6947	0.7019	0.6943	**0.7160**	0.7362
Stddev	0.0464	0.0285	0.0355	0.0386	0.0369	0.0316	0.0328	0.0286
Mean rank	5.2500	3.5000	4.9167	4.5833	3.0000	5.2500	**1.5000**	N/A
p-values	1.1e-174	2.5e-62	7.4e-99	3e-75	2.4e-52	1.5e-114	N/A	N/A

(0.7160) over all base models. Additionally, the Meta-scaler was able to yield the highest performance when compared to choosing any of the STs (or the nonscaled data) for all but two base models: LCA and Percep. Since the meta-model performance, presented in Table 8, was also lower for these two base models, this is evidence that there is a relationship between the performance at the meta and base levels and that the set of meta-features chosen may be missing important measures to capture the relationship of these two base models with the choice of scaling techniques. We can also highlight that the performance of the Meta-scaler is close to the performance achieved by the *Truth* most of the time. When we look at the mean rank, i.e., the average of the rankings of the STs for each base model, the Meta-scaler has a better result (1.5) than any of the STs, being visibly superior to the second-best result (3.5), achieved by the Standard Scaler technique.

In order to evaluate the differences in performances between the STs, the Meta-scaler, and the *Truth*, and understand their statistical significance, considering the 3600 performances for each of the eight scaling strategies (8 samples), we executed hypothesis tests as follows: First, a Shapiro-Wilk normality test was employed and confirmed that the eight samples are not normally distributed. Then, a Friedman test rejected the null hypothesis that the means for the populations are equal (statistic = 5683.92, p-value = 0.0), indicating that there is at least one scaling strategy significantly different from another one.

Finally, to find out which techniques significantly differ from the Meta-scaler, we followed the recommendation of Benavoli et al. (120): We performed multiple comparisons considering the performances of each ST plus the *Truth* versus the Meta-scaler, using the Wilcoxon signed-rank test. The resulting p-values are reported in the last row of Table 9. Considering a significance level of $\alpha = 0.05$ and correcting it to control family-wise Type-I error, we obtain a Bonferroni corrected significance level of 0.000893. Since all pairwise comparisons resulted in p-values much lower than this level, we conclude that the Meta-scaler's performance is significantly greater than all the STs.

This analysis and the discussion of Table 9, further confirms the effectiveness of the Meta-scaler as an approach to efficiently select an appropriate ST for a pair of dataset and classifier.

3.6.3 Analysis of Meta-feature importances

What meta-features are more important for the Meta-scaler?

To understand which meta-features are more important for the task of ST recommendation, we calculated the average meta-feature importance for every DT within the ensemble used by the DMI meta-model when predicting the class of each test instance. The importance is calculated by the normalized total reduction of Gini impurity by meta-feature (Gini importance) (22). In Table 10, we report the mean importance (considering all base models), mean ranking, type, subtype, origin, and a short description of the 15 most important meta-features.

Table 10 – Top 15 most important meta-features according to the mean ranking across all base models.

Meta-feature	Mean ranking	Mean importance	Type	Subtype	Origin	Description
ch	14.000 (±7.874)	0.015 (±0.006)	Clustering	clustering	PyMFE	Calinski and Harabasz index.
t1.mean	14.333 (±19.246)	0.016 (±0.005)	Concept & complexity	complexity	PyMFE	Fraction of hyperspheres covering data.
pb	15.583 (±9.671)	0.015 (±0.005)	Clustering	clustering	PyMFE	Pearson corr. between class matching and instance distances.
attr_conc.mean	16.917 (±14.594)	0.014 (±0.004)	Simple, stats & info-theory	info-theory	PyMFE	Compute concentration coef. of each pair of distinct attributes.
sparsity.mean	18.333 (±13.727)	0.017 (±0.010)	Simple, stats & info-theory	statistical	PyMFE	(Possibly normalized) sparsity metric for each attribute.
skewness.mean	18.333 (±17.233)	0.014 (±0.004)	Simple, stats & info-theory	statistical	PyMFE	Skewness for each attribute.
kurtosis.mean	19.917 (±12.003)	0.013 (±0.003)	Simple, stats & info-theory	statistical	PyMFE	Kurtosis of each attribute.
sd_ratio	20.333 (±19.579)	0.013 (±0.004)	Simple, stats & info-theory	statistical	PyMFE	Statistical test for homogeneity of covariances.
N2_partial.0	24.083 (±19.528)	0.013 (±0.006)	Concept & complexity	neighborhood	ImbCoL	Average intra/inter class nearest neighbor distances (Class 0).
N3_partial.0	24.167 (±19.757)	0.012 (±0.003)	Concept & complexity	neighborhood	ImbCoL	Leave-one-out error rate of the 1-NN algorithm (Class 0).
mad.mean	26.000 (±21.667)	0.014 (±0.006)	Simple, stats & info-theory	statistical	PyMFE	Median Absolute Deviation (MAD) adjusted by a factor.
N4_partial.0	26.000 (±17.077)	0.012 (±0.004)	Concept & complexity	neighborhood	ImbCoL	Nonlinearity of the one-nearest neighbor classifier (Class 0).
wg_dist.mean	27.250 (±14.442)	0.011 (±0.002)	Concept & complexity	concept	PyMFE	Weighted dist. captures how dense/sparse is the example distrib.
F2_partial.0	29.750 (±12.337)	0.011 (±0.002)	Concept & complexity	overlapping	ImbCoL	Overlapping of the per-class bounding boxes (Class 0).
nodes_per_level.mean	30.250 (±16.012)	0.011 (±0.003)	Model-based	model-based	PyMFE	Ratio of number of nodes per tree level in DT model.

Ranking first in the list is the *ch* meta-feature, the Calinski-Harabasz index(91), a clustering-related measure that is the ratio of the sums of extra and intra-clusters dispersion. The value is higher when clusters are dense and far apart. Next, *t1* is a complexity meta-feature of the neighborhood subtype that measures the fraction of hyperspheres covering the data (73). To

calculate *t1*, hyperspheres are built centered at each instance, and their radii increase until they encounter an instance of another class. Hyperspheres enclosed within larger ones are eliminated. Finally, *t1* is the ratio between the numbers of hyperspheres and instances. It is also a measure related to clusters, as low values indicate fewer hyperspheres and simpler datasets, i.e., more densely clustered same-class examples (121).

Following in the rank, the *pb* meta-feature(122) computes the Pearson correlation between the class matching of two dataset instances and the distance between them. Class matching is a Boolean variable that is true when the instances belong to the same class and false otherwise. Therefore, this measure describes how much the instances of different classes are far apart or superimposed in the space.

We also highlight three well-ranked statistical meta-features *skewness, sparsity,* and *kurtosis* that aim at capturing the shape of the distribution of instances in the dataset. Skewness measures how asymmetric the data is. Sparsity, as the name implies, measures if the data is sparse or denser. Kurtosis tells us whether the data distribution is heavily or lightly tailed, which is related to the presence or absence of outliers.

As a summary, it is essential to highlight the meta-models' reliance on features that are related to how the data is distributed in the space (*ch, t1, skewness, sparsity, kurtosis, wg_-dist*) and how the instances of a class are clustered together and apart from the instances from the other class (*pb, N2, F2*). Note that a few of these metrics sense the presence of outliers (*kurtosis, mad*). If we recall Figure 9, we see that depending on the shape of the data and the presence of outliers, different STs can have drastically different outcomes, which entails the need to take these properties into account to choose the right ST.

Do the meta-models rely on different meta-features depending on the base model used?

To find out if meta-models trained for recommending STs for different base models rely on different meta-features in their decision processes, in Figure 13, we plot the importances (y-axis) of all 133 meta-features (x-axis) for the meta-models trained for each one of the 12 base models (labeled on the right side). The meta-features are grouped and colored according to their type. This figure shows that the meta-model relies on entirely different meta-feature subsets when trained for different base models. When we look at the top two most important meta-model meta-features for each base model (indicated in the figure), we can see that they vary all the time. There is no pair of base models for which the top two meta-model

meta-features are coincident.

Also notable in Figure 13 is the high importance of the *L1* meta-feature for the meta-models trained for the Bagging, LCA, Percep, and SVM_lin base models. *L1* is a complexity meta-feature of the linearity subtype that measures the distance of erroneous instances to a linear classifier, i.e., given that a linear binary classifier can be seen as a hyperplane that divides the feature space of a dataset into two sections, each attributed to one of two classes, *L1* measures the distance of each instance in the wrong subsections to the hyperplane. The *partial.0* suffix means that this meta-feature is the component of *L1* that takes into account only the instances of class 0. In order words, it is a way to measure the linear separability of the dataset, which explains why this meta-feature is important when selecting the ST for linear classifiers such as the LCA, the Perceptron, the SVM with the linear kernel, and also the Bagging model, which uses Perceptrons as its base models.

Although there are other ensembles of Perceptrons in the base classifier collection, the Bagging is the one to which the *L1* meta-feature is more important for ST selection. This is probably because it is the only static ensemble technique in the collection. This means that it always selects all its base classifiers to combine their decisions instead of the more complex dynamic ensembles that employ extra steps to select only the most competent base classifiers. Hence, Bagging is more dependent on the behavior of the type of classifiers in the pool it is built with, which in this case is the Perceptron, a linear model. However, when compared to the importances obtained for the meta-model specialized for the Percep, we can see in Figure 13 that the importance is much more concentrated in the *L1* meta-feature for the Bagging classifier. It seems that the linearity of the dataset is even more decisive for ST selection when using the Bagging classifier in this case. Still, the combination of multiple classifiers may be playing a role in minimizing the effects of other data characteristics that are measured by the remaining meta-features.

The radically different subsets of meta-features used for each base model confirm the need to treat ST selection as a problem that is dependent on both the dataset and the base classifier used. We can see that even within the same classifier family (SVMs), the different kernel (linear or RBF) is enough to change the importance of meta-features and, thus, the behavior of the meta-model trained. Our approach of training a different meta-model for each base classifier given as input to the construction phase was, therefore, a decision in the right direction.

Additionally, these results indicate that an investment in feature selection (123, 124) may

benefit future iterations of the Meta-scaler, especially if it is done on a per meta-model basis. That is, for each meta-model trained, a different set of meta-features is selected. This may lead to even better specialization of the meta-models, reaching higher performances, while also reducing dimensionality and the cost of meta-feature extraction in the recommendation phase.

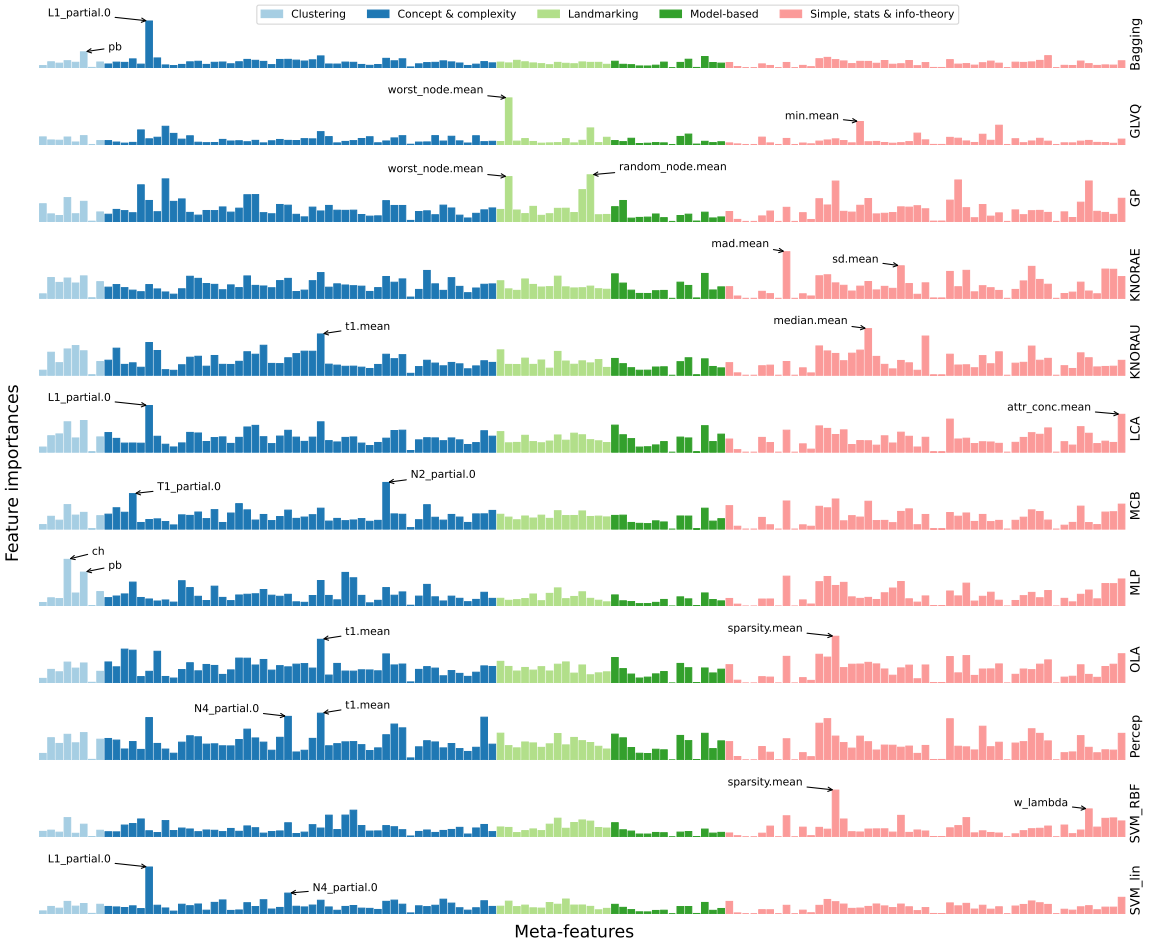


Figure 13 – Meta-features' importances when the Meta-scaler is trained for each base model.

3.6.4 Comparison with the state of the art

As can be seen in Table 7, only the work of Jain et al. (2018) (6) is concerned exclusively with the selection of a scaling technique and thus can be directly compared to our proposal. As the authors did not make the code for their solution or experiments publicly available, we reproduced their solution using the same meta-features, meta-target definition procedure, and meta-model algorithm they used. In order to compare our method to a multi-step method, the solution presented in Zagatti et al. (2021)(52), Metaprep, was also assessed by extracting only the recommended ST information from the pipelines it outputted. We used the publicly

available pre-trained Metaprep. For more details on how this comparison was performed, refer to the supplementary material and the paper's repository on GitHub.

Here we present, in Table 11, the summary of the results of the three approaches. This table presents the mean classification performances (F1), over all 300 datasets, that each base classifier was able to achieve when using the STs recommended by each of the three proposals: Meta-scaler (ours), Jain et al. (2018)(6) and Zagatti et al. (2021) (52). Note that the Meta-scaler yields the best performance in 11 out of the 12 base classifiers, resulting in the best mean and median values of the three compared methods. Similarly to Section 3.6-B, we performed a Wilcoxon Signed-rank test, with a corrected alpha of 0.0083, comparing the results of the Meta-scaler to the two other methods, concluding that they are significantly different (p-values in the last row of the table).

Table 11 – Mean classification performances (F1) achieved by the twelve base models when using different ST selection approaches, including ours (Meta-scaler) and two methods proposed in related studies. The last row shows the resulting p-values after comparing the performances of each method versus the Meta-scaler using the Wilcoxon Signed-rank test.

	Meta-scaler	**Jain et al.**	**Zagatti et al.**	**Truth**
Bagging	**0.7170**	0.7129	0.6638	0.7311
GLVQ	**0.6576**	0.6250	0.5954	0.6642
GP	**0.7522**	0.7055	0.7120	0.7671
KNORAE	**0.7434**	0.7224	0.7133	0.7593
KNORAU	**0.7382**	0.7183	0.6948	0.7486
LCA	**0.6788**	0.6766	0.6095	0.7207
MCB	**0.7431**	0.7189	0.7129	0.7600
MLP	**0.7008**	0.6876	0.5790	0.7358
OLA	**0.7437**	0.7190	0.7131	0.7597
Percep	0.6702	**0.6789**	0.5986	0.7249
SVM_RBF	**0.7399**	0.7261	0.6541	0.7483
SVM_lin	**0.7071**	0.7023	0.5585	0.7143
Median	**0.7276**	0.7092	0.6590	0.7421
Mean	**0.7160**	0.6995	0.6504	0.7362
Stddev	0.0328	0.0289	0.0593	0.0286
Mean rank	**1.0833**	2.0000	2.9167	N/A
p-values	N/A	0.0034	0.0005	N/A

The superiority of the Meta-scaler indicates that its specialized meta-models (one for each base model) approach, as opposed to the one-fits-all solution in Jain et al. (2018), allows for a better recommendation. We also highlight that the only method that is concerned with the full pipeline, Zagatti et al. (2021)(52), is exactly the one with the poorest performance for ST selection, being the third place for all except one base model (GP). This ratifies the importance of building specialized meta-models that focus on a single step of the pipeline rather than the whole sequence of transformations. It suggests that a modular approach, where several specialized meta-models are combined within a single solution, may be a better fit for the preprocessing pipeline definition problem than the single meta-model approaches

explored previously in the literature.

3.6.5 Limitations

Note that, because some STs perform the same for certain pairs of classifier and dataset, with the best algorithm approach employed in the Meta-scaler, we are transforming a problem that is essentially multi-label into a single-label one. This comes with some issues: (i) The training process can be disturbed by the arbitrary selection of one of the tied STs since it will not be able to learn what caused the selection of that technique instead of the other one with the same performance; (ii) When there is a tie for a test instance, the model predicts one of the STs in the tied group, but it will only count as a hit if it happens to be the selected technique during the meta-target definition, hence the test scores shown in Table 8 may be underestimated; (iii) The class frequencies in the meta-dataset may not represent the reality, since, for example, when the NS technique gets involved in a tie, it is always selected as the meta-target. Even so, our proposal still achieves a considerable base-level classification performance improvement over the static choices of ST and, more importantly, over other meta-learning ST selection methods. This improvement can potentially be increased by addressing these issues in future work.

3.7 CONCLUSION

In this paper, we presented the problem of scaling technique selection and stated our goal to build and evaluate a meta-learning framework to tackle this problem. We presented the framework proposal, and then empirically evaluated its effectiveness. The Meta-scaler framework can be instantiated in many different ways to accommodate different classification algorithms, scaling techniques, and meta-features.

The experiment demonstrated the effectiveness of using a meta-learning system to efficiently recommend a scaling technique for a given pair of dataset and classification algorithm, optimizing its performance. The results show that the Meta-scaler provides a significant base-level classification performance increase against the static ST choices in the mean of all 300 datasets for 10 out of 12 models, while also significantly outperforming the state-of-the-art methods for ST selection via meta-learning, where the Meta-scaler presents the best overall performance and the best performance for 11 out of 12 base models. Therefore, this work represents a step forward for tackling the problem of ST selection.

Nonetheless, the proposed framework can be improved. In future work, we intend to investigate ways to (i) enable effective ST recommendation for any classifier given as input, even if it is not in the set of base classifiers used for building the meta-dataset, (ii) tackle the current limitations related to the conversion of the multi-label problem into a single-label one. Additionally, the analysis of the meta-features used by the Meta-scaler showed that different meta-features are given more importance depending on the type of base classifier being employed on the dataset, which gives us directions on how to curate better the meta-features set seeking further improvement of performance on ST selection for the models where our solution did not work so well (LCA and Percep). Finally, the success of the single-step and classifier-dependent approach of the Meta-scaler against the state-of-the-art makes a case for the employment of multiple, per-step, specialized meta-models as a promising avenue for future AutoML research.

ACKNOWLEDGMENT

This work was partially supported by the Brazilian agencies: *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES) - Finance Code 001, *Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco* (FACEPE) and *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq).

4 META-SCALER+: IMPROVING THE META-SCALER WITH A CLASSIFIER REPRESENTATION SPACE

Lucas B.V. de Amorim, George D. C. Cavalcanti, Rafael M.O. Cruz

Abstract

Dataset scaling significantly impacts classification performance but can harm accuracy if an inappropriate scaling technique (ST) is chosen. Optimal STs vary by dataset and classifier. The Meta-scaler framework addressed this challenge by automating ST selection using meta-learning (MtL) but was limited to predefined classifiers, restricting its flexibility. To overcome this limitation, this study presents Meta-scaler+, which can recommend STs for any dataset and classifier. For this, we introduce the "Classifier Performance Space," a novel approach for characterizing unknown classifiers by positioning them in a multidimensional space that allows for comparison with known classifiers. Meta-scaler+ leverages this space to dynamically combine meta-models specialized for similar classifiers, providing precise ST recommendations tailored to the query dataset and classifier. The framework's performance was evaluated against three benchmarks: a baseline generalist meta-model, an optimistic use-case where the query classifier is known, and the theoretical upper limit, a meta-model that always selects the optimal ST. Results demonstrate that Meta-scaler+ outperforms the baseline in meta-model accuracy, achieves competitive base-level performance close to the upper limit, and provides high-quality ST rankings. Furthermore, a comparison with the state of the art positions Meta-scaler+ higher than other methods and close to Meta-scaler, albeit without its limitations. This study is open-source, and its data and code are available in this paper's repository [1].

Keywords: Meta-learning, Scaling techniques, Normalization, Classification, Classifier representation.

4.1 INTRODUCTION

Dataset scaling has been shown to be an important preprocessing step of a machine learning (ML) pipeline, often allowing the improvement of classification models (3, 48). However, the choice of scaling technique (ST) to apply to a certain dataset is also relevant, and choosing an inadequate technique can be more detrimental to performance than not scaling the data

[1] <https://github.com/amorimlb/meta_scaler_plus>

at all. Furthermore, the optimal ST for a dataset cannot be generalized for different choices of classification algorithms (48).

To tackle the challenge of ST selection as a dataset- and classifier-dependent task, we have previously proposed the Meta-scaler (125), a meta-learning (MtL) framework to automate ST selection. We have shown that the Meta-scaler is effective in selecting an ST for a given pair of dataset and classifier, significantly optimizing its performance against the static choice of any of six scaling methods studied and outperforming the state-of-the-art methods for ST selection via MtL.

Despite its significant advancements, this initial iteration of Meta-scaler faces a notable limitation: the query classifier is constrained to the set of classifiers used to create the meta-dataset. Consequently, potential users have limited options for classification algorithms. This restriction arises from the Meta-scaler's design, which trains a separate meta-model for each possible classifier. This approach addresses the ST selection problem as a classifier and dataset-dependent task, enhancing performance. However, it also limits system flexibility for end-users, as they can only obtain ST recommendations for a dataset if they also use one of the base classifiers included in the Meta-scaler's set.

Due to this limitation of the Meta-scaler, we propose Meta-scaler+, which is capable of selecting an ST not only for any dataset but also for any classifier. To do this without losing the desirable model-dependent recommendation feature available in Meta-scaler, we devise a novel method to characterize an unknown classifier and compare it to the ones known by the system. This method, called the Classifier Performance Space, positions the query classifier in a multidimensional space of classifiers. Meta-scaler+ then selects the closest known classifiers and combines their respective meta-models to predict the performances of all the STs for the pair of query dataset and query classifier. It turns out that the Classifier Performance Space has much broader applications than just as an addition to the Meta-scaler+, as it enables the construction of meta-learning systems capable of predicting classifiers' performances on unseen datasets.

To evaluate Meta-scaler+, we compared its performance to (i) a baseline generalist meta-model that is trained using information from all known base models, (ii) an optimistic use-case where the query classifier is known, and (iii) the truth, i.e., a hypothetical model that always selects the best ST for the pair of query dataset and query classifier.

The results indicated that Meta-scaler+ can perform better than the baseline method in terms of meta-model performance, achieving precise ST performance prediction and high-

quality rankings. Base-level performance was similar to the baseline but also very close to the theoretical upper limit (truth). Furthermore, a comparison with the state of the art positions Meta-scaler+ higher than other methods in the literature and close to Meta-scaler, albeit without its limitations.

The main contributions of this study are:

- It eliminates the previous Meta-scaler's limitation where the query classifier must be in the set of classifiers used for training the meta-models.
- It provides a solid alternative to a generalist meta-model approach. Employing an ensemble of meta-models that are dynamically combined to provide an adequate prediction depending on the query classifier and the query dataset.
- It introduces the concept of the Classifier Performance Space, a novel method to characterize a classifier generating meta-data that is then used for an effective combination of meta-model decisions and the prediction of models' performance on unseen datasets.

The rest of this text is organized as follows. Sections 4.2 and 4.3 present the proposed methods, the Classifier Performance Space, and the Meta-scaler+. Then, Section 4.4 details the experimental methodology used to evaluate the proposed methods. Results and their discussion are presented in Section 4.5, which is followed by a description of the main limitations of this study in Section 4.6. Finally, in Section 4.7, we present our conclusions and ideas for future work.

4.2 CLASSIFIER PERFORMANCE SPACE

Classifier Performance Space (CPS) is a simple classifier characterization technique that relies on the classifier's behavior in a preferably small set of datasets to effectively obtain meta-data about any classifier. Given a set of t datasets $\mathbb{L}$ and a set of n classifiers $\mathbb{K}$, the CPS can be defined as:

$$\text{CPS} = \{(p_{\mathbf{D'}_1,c_i}, p_{\mathbf{D'}_2,c_i}, p_{\mathbf{D'}_3,c_i}, \ldots, p_{\mathbf{D'}_t,c_i}) | \forall c_i \in \mathbb{K}\} \tag{4.1}$$

where $\mathbf{D'}_1, \mathbf{D'}_2, \mathbf{D'}_3, \ldots, \mathbf{D'}_t$ are the t datasets in the $\mathbb{L}$ set, and $p_{\mathbf{D'}_j,c_i}$ represents the performance achieved by classifier c_i on dataset $\mathbf{D'}_j$. This performance can be measured by

any classification performance metric, such as accuracy or F1-score, and cross-validation is recommended for stable and reliable results.

The tuples in CPS can be interpreted as a set of points in a t-dimensional space. Each point represents a classifier, and each dimension $p_{\mathbf{D'}_j}$ is the performance on dataset $\mathbf{D'}_j$. Figure 14 illustrates such interpretation.

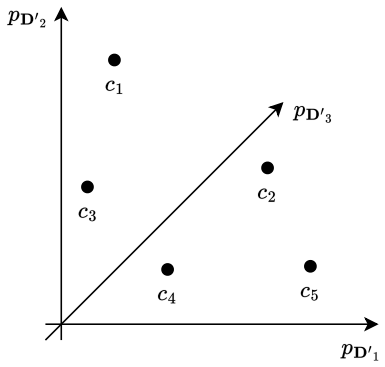


Figure 14 – An example of Classifier Performance Space represented as an Euclidean space. Here, $t = 3$ and $n = 5$.

CPS is an effective way of characterizing classifiers by means of their behaviors on different datasets. It allows us to compare classifiers and calculate their similarities based on conventional spatial distance metrics such as the Euclidean distance.

4.2.1 Classifier Performance Space application to meta-learning

Meta-learning relies on meta-information to make predictions on how models will behave on new tasks (datasets). Meta-learning systems can exploit meta-information from tasks for which the base model's behaviors are known to learn to predict how the same models are going to behave on unknown tasks. However, when the query model is also unknown, the system must employ a way of extracting meta-information about this query model to make model-dependent predictions, otherwise its predictions may be obsolete or inadequate for the new model.

While there are several ways of obtaining meta-information from datasets, classifiers are much harder to effectively characterize. Most methods of classifier characterization rely on models' hyperparameters that only apply to a specific classification algorithm or algorithm family. CPS, on the other hand, is classifier agnostic because it relies on classification performances

for characterization and, therefore, can be used to obtain homogeneous meta-information from any type of classification model.

The resulting classifiers' meta-characterization can be employed, in tandem with dataset meta-features, to build meta-learning systems that are dataset and classifier-aware. That is, systems that make different, specialized predictions for each different query classifier. One example of such a meta-learning system is the proposed Meta-scaler+ framework.

4.3 META-SCALER+

The proposed Meta-scaler+ framework builds upon the previous Meta-scaler framework (125) to create an equally robust but much more flexible meta-learning solution for scaling technique selection. Figure 15 presents a flow diagram illustrating the steps of both the construction and the recommendation phase of the framework. Each step of the process is detailed in the following subsections.

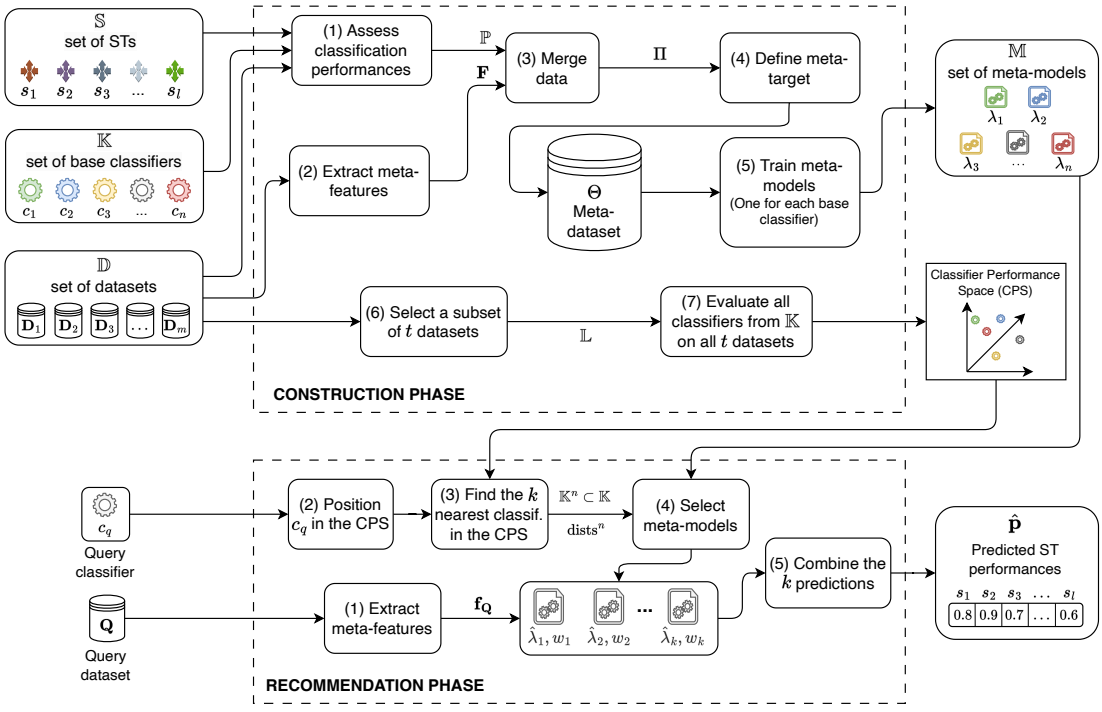


Figure 15 – Proposed Meta-scaler+ framework

Meta-scaler+ retains all the features of its predecessor, such as the ability to train one meta-model for each base classifier used in the meta-dataset construction, thus allowing for a specialized recommendation. It is important to retain this functionality of specialist meta-models because we have verified that this approach yields better recommendations than a

generalist solution (125). However, two new features were added: (1) It works with unknown query classifiers, which solves the main limitation of the previous iteration, and (2) It can predict the performances of the STs. Therefore, it can recommend a ranking of the STs instead of only the predicted best ST. To make these improvements possible, a novel concept of Classifier Performance Space is introduced in the construction phase, and the meta-models employ multi-output regression algorithms to predict the performance of each ST instead of the multi-class classification approach of the previous iteration.

The Classifier Performance Space (CPS) consists of a t-dimensional space where the classifiers are positioned according to the classification accuracy they achieve on each of the t datasets. This way, similar classifiers, with similar performances on the same datasets, are closer in the CPS than dissimilar ones. In the Meta-scaler+, this concept is used to select the most appropriate combination of specialist meta-models for an unknown query classifier based on its distance to the known classifiers in the CPS.

4.3.1 Construction phase

In the construction phase, three crucial components are built: the meta-dataset, the meta-models, and the Classifier Performance Space. As illustrated in Figure 15, this phase, which can be divided into the seven steps outlined below, requires the following inputs:

- $\mathbb{S} = \{s_1, s_2, s_3, \cdots, s_l\}$ – The set of scaling techniques.
- $\mathbb{K} = \{c_1, c_2, c_3, \cdots, c_n\}$ – The set of base classifiers.
- $\mathbb{D} = \{\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3, \cdots, \mathbf{D}_m\}$ – The set of training datasets.

Where $\mathbf{D_i}$ is a matrix that contains both the independent variables $\mathbf{X_i}$ and the dependent variables $\mathbf{y_i}$ in the i-th dataset. It's essential to select the elements of $\mathbb{D}, \mathbb{S}$, and $\mathbb{K}$ with diversity in mind, as this will enhance the generalization capability of the meta-models generated.

Step (1) - Assess classification performances

The classification performance of each pair of ST and classification algorithm is evaluated on every dataset. It is recommended to employ a cross-validation procedure for stable and reliable results (99, 104). This evaluation results in the set of performances $\mathbb{P} = \{p_{\mathbf{D}_i,s_j,c_k} | \forall \mathbf{D}_i \in \mathbb{D} \wedge \forall s_j \in \mathbb{S} \wedge \forall c_k \in \mathbb{K}\}$. To simplify the notation we will sometimes refer to $p_{\mathbf{D}_i,s_j,c_k}$ as $p_{i,j,k}$

and omit the domains in $\forall \mathbf{D}_i \in \mathbb{D} \wedge \forall s_j \in \mathbb{S} \wedge \forall c_k \in \mathbb{K}$ reducing it to $\forall \mathbf{D}_i, s_j, c_k$, similar omissions will be done elsewhere in this text.

Step (2) - Extract meta-features

A standardized set of h meta-features is extracted from every $\mathbf{D}_i \in \mathbb{D}$. Formally, we represent this as the $m \times h$ matrix $\mathbf{F} = \{\mathbf{f}_{\mathbf{D}_1}, \mathbf{f}_{\mathbf{D}_2}, \mathbf{f}_{\mathbf{D}_3}, \cdots, \mathbf{f}_{\mathbf{D}_m}\}$ where $\mathbf{f}_{\mathbf{D}_i} = [f^1_{\mathbf{D}_i}, f^2_{\mathbf{D}_i}, f^3_{\mathbf{D}_i}, \cdots, f^h_{\mathbf{D}_i}]$ is the meta-feature vector that represents $\mathbf{D}_i$ in the meta-problem.

Step (3) - Merge data

Next, the meta-features $\mathbf{f}_{\mathbf{D}_i}$ calculated for each dataset $\mathbf{D}_i \in \mathbb{D}$ and the performances $p_{i,j,k}$ of each pair of classifier and ST on those datasets are merged. The result is the set $\Pi = \{(\mathbf{f}_{\mathbf{D}_i}, p_{i,j,k}) | \forall \mathbf{D}_i, s_j, c_k\}$.

Step (4) - Define meta-target

In the previous iteration of the Meta-scaler framework, the meta-target is defined via the Best Algorithm approach. In contrast, in the Meta-scaler+ framework, the meta-targets are derived from the performances set $\mathbb{P}$, where for a given instance that represents dataset $\mathbf{D}_i$ and classifier c_k, the meta-target is the vector in Eq. 4.2.

$$\mathbf{p}_{i,k} = [p_{\mathbf{D}_i,c_k,s_1}, p_{\mathbf{D}_i,c_k,s_2}, p_{\mathbf{D}_i,c_k,s_3}, \ldots, p_{\mathbf{D}_i,c_k,s_l}] \tag{4.2}$$

Then, the final meta-dataset can be formalized as $\Theta = \{(\mathbf{f}_{\mathbf{D}_i}, \mathbf{p}_{i,k} | \forall \mathbf{D}_i, c_k\}$, and its instances originated from the performances of the base classifier c_k can be written as $\Theta_k = \{(\mathbf{f}_{\mathbf{D}_i}, \mathbf{p}_{i,k} | \forall \mathbf{D}_i\}$. Note that $\Theta_a \cap \Theta_b = \emptyset$, $\forall a \neq b$.

Each element in the target vector represents the performance of a single ST on a corresponding dataset when using the corresponding base classifier. Consequently, this meta-task is addressed by the Meta-scaler+ as a multi-output regression problem. This approach provides additional flexibility in combining the outputs of multiple meta-models and enables the prediction of the performances for all the STs in $\mathbb{S}$. With the predicted performances, one can

derive a ranking of STs rather than limiting itself to predicting the best ST.

Step (5) - Train meta-models

This step is where the meta-models are finally built. By learning from the pairs of meta-features and meta-targets in meta-dataset Θ, the regression algorithm models the relationship between them to create the predictive models. As a result, a set $\mathbb{M}$ of meta-models is generated. Note that a different meta-model λ_i is trained for the meta-dataset instances Θ_i corresponding to each base classifier c_i in $\mathbb{K}$, hence $|\mathbb{M}| = |\mathbb{K}| = n$.

Step (6) - Select a subset of t datasets

To form a Classifier Performance Space, a subset $\mathbb{L}$ of t datasets from $\mathbb{D}$ is selected. This subset must be diverse in order to appropriately characterize and differentiate classifiers. The *k-means++* algorithm (126) is employed in this step to determine a set of t datasets where their distances in the meta-features' space are maximized. The *k-means++* algorithm was designed to find a better set of initial cluster centers for the traditional *k-means* algorithm. Adapting it for our task of selecting diverse datasets is then straightforward. Given the set $\mathbb{D}$ of m datasets, each described by their meta-features vector $\mathbf{f}_{\mathbf{D}_i}$, and let $d(\mathbf{D}_i)$ represent the distance from a dataset $\mathbf{D}_i$ to the closest dataset previously selected. Here we use the Euclidean distance in the meta-features' space. Then, the *k-means++* algorithm works as follows:

1. Choose an initial dataset $\mathbf{D}_0$ uniformly at random from $\mathbb{D}$.
2. Choose the next dataset $\mathbf{D}_i \in \mathbb{D}$ with probability $\frac{d(\mathbf{D}_i)^2}{\sum_{j=1}^{m} d(\mathbf{D}_j)^2}$.
3. Repeat Step 2 until a total of t datasets are chosen.

This approach ensures that datasets farther from the already selected ones have a higher probability of being chosen. Consequently, the likelihood of selecting a subset $\mathbb{L}$ of t datasets with comprehensive coverage of the meta-feature space increases. In other words, datasets exhibiting diverse meta-features are more likely to be selected. We hypothesize that this diversity will provide an adequate characterization of the classifiers for our specific purposes.

Step (7) - Evaluate all classifiers in $\mathbb{K}$

Finally, all classifiers in $\mathbb{K}$ are evaluated in all t datasets in $\mathbb{L}$. The resulting performances of a classifier in each dataset are arranged in a tuple that represents this classifier as a point in the Classifier Performance Space (CPS) as defined in Eq. 4.1.

4.3.2 Recommendation phase

The recommendation phase happens when the Meta-scaler is put into production, i.e., when it is requested to predict the performances $\hat{\mathbf{p}}$ of the STs, given a particular query dataset $\mathbf{Q}$ and the classifier c_q. With the output $\hat{\mathbf{p}}$ one can derive Meta-scaler+'s recommendation using its method of choice, for example, (1) The ST with the maximum predicted performance (zero-shot recommendation), (2) The best of the 2 STs with the highest predictions (2-shot recommendation), or (3) The best of the 3 STs with the highest predictions (3-shot recommendation).

Step (1) - Extract meta-features

To predict the ST performances $\hat{\mathbf{p}}$ to be achieved when given a classifier and query dataset, the same set of standardized meta-features used in the construction phase must be extracted from this new dataset. Hence, this first step results in the vector $\mathbf{f_Q}$ that is given as input to the next step. It is important to notice here that since this meta-feature extraction will always be done when a new dataset is presented, it must be computed at a relatively low cost, such that it makes sense to employ the meta-learning system instead of exhaustively testing all the ST possibilities.

Step (2) - Position c_q *in the CPS*

This is the classifier characterization step. The query classifier c_q is evaluated on the t datasets from $\mathbb{L}$ to obtain its performances when applied to these datasets. This results in a tuple of performances, which is precisely the representation of c_q in the Classifier Performance Space.

Step (3) - Find the k nearest classifiers in the CPS

In this step, the k nearest classifiers from $\mathbb{K}$ are calculated according to their Euclidean distance to c_q in the CPS. Therefore, from this step, the subset $\mathbb{K}^n$ of $\mathbb{K}$, that contains the k nearest classifiers and their respective distances $\text{dists}^n = \{\text{dist}(c_i, c_q) | \forall c_i \in \mathbb{K}^n\}$ to the query classifier c_q are passed on to the next step.

Step (4) - Select meta-models

In this step, for each classifier $c_i \in \mathbb{K}^n$, the corresponding meta-model λ_i (i.e., the meta-model that was trained with the meta-dataset component Θ_i) is selected to predict the ST performances in the next step. To combine these predictions, a weight w_i, that decays with the distance $\text{dist}(c_i, c_q)$, is calculated for each selected meta-model λ_i, according to Eq. 4.3 and Eq. 4.4.

$$\mathfrak{d}_i = \begin{cases} 0, & \text{if} \quad \max(\text{dists}^n) = \min(\text{dists}^n) \\ \frac{\text{dist}(c_i, c_q) - \min(\text{dists}^n)}{\max(\text{dists}^n) - \min(\text{dists}^n)}, & \text{otherwise} \end{cases} \tag{4.3}$$

$$w_i = \frac{2 - \mathfrak{d}_i^2}{2} \tag{4.4}$$

Where, $\max(\text{dists}^n)$ and $\min(\text{dists}^n)$ represent the maximum and minimum values in dists^n, respectively. Note that w_i is in a range of 0.5 to 1. The goal is for meta-models trained for classifiers that are more similar to the query classifier to have higher weights in the prediction.

Step (5) - Combine the k predictions

Finally, the final prediction is the weighted average of the predictions output by the k meta-models, as in Eq. 4.5.

$$\hat{\mathbf{p}} = \frac{\sum_{i=1}^{k} \hat{\mathbf{p}}_i . w_i}{\sum_{i=1}^{k} w_i} \tag{4.5}$$

Where $\hat{\mathbf{p}}_i$ is the output of meta-model λ_i.

4.4 EXPERIMENTAL PROTOCOL

4.4.1 Datasets

Two publicly available collections of datasets were used in this study, one originates from contest in ICPR 2010 (106), and the other originates from a particular subset of datasets available at the KEEL repository (105). Using these public dataset collections fosters the reproducibility of this study. These collections were used in our previous studies. The ICPR 2010 collection was used in (125), and the KEEL collection was used in (48). Both collections are also available in this study's repository.

4.4.1.1 ICPR 2010 datasets

This collection comprises 300 binary datasets originating from the data made available during The Landscape Contest at ICPR 2010 (106). We chose this collection because it was designed to contain datasets showing varying levels of complexity (107), which conveniently yields a higher degree of diversity among the datasets. We used the 300 binary labeled datasets from this collection (D1 to D300). By using this diverse collection of datasets, we aim to strengthen the meta-models' ability to generalize. This allows the deployment of the trained meta-models on problems from several unseen domains.

4.4.1.2 KEEL datasets

This collection is composed of 82 real-world datasets downloaded from KEEL. They have been originally published on the UCI repository (39), but we are using the KEEL versions since the multi-class datasets were broken down to form multiple binary datasets. The exact procedure used to filter KEEL's dataset collection to obtain these 82 datasets and their description is available in (48). These datasets present a diverse range of domains such as glass identification (glass), medical analyses (pima, wisconsin, haberman, cleveland, dermatology), plant identification (iris), speech recognition (vowel), molecular and cellular biology (yeast, ecoli), image recognition (vehicle, led7digit), aeronautics (shuttle) and others. This diversity makes this an ideal testbed for our proposed Meta-scaler+ framework.

4.4.2 Meta-features

Two open-source libraries were used for extracting the meta-features from the datasets: PyMFE (78) (111 meta-features) and ImbCoL (115) (22 meta-features). This results in 133 meta-features for each dataset. This is the same meta-feature set used in our previous Meta-scaler study (125) and has been detailed in its supplementary material.

4.4.3 Scaling Techniques

As in (48) and (125), the set of scaling techniques used was: No Scaling, Standard Scaler, Min Max Scaler, Maximum Absolute Scaler, Robust Scaler, Quantile Transformer. Therefore, using their respective acronyms, we define $\mathbb{S} = \{\text{NS, SS, MMS, MAS, QT}\}$.

4.4.4 Base classifier algorithms

The set $\mathbb{K}$ of base classifiers used in this experiment is composed of $n = 12$ diverse algorithms from different families and types (ensemble and non-ensemble): Support Vector Machine (108) (SVM_RBF, using the RBF kernel and SVM_lin using the linear kernel), Bagging (29), Perceptron (Percep) (109), Gaussian Process (GP) (110), Multilayer Perceptron (MLP) (111), Generalized Learning Vector Quantization (GLVQ) (112), Local Class Accuracy (LCA) (33), Multiple Classifier Behavior (MCB)(34), Overall Local Accuracy (OLA) (33), k-Nearest Oracles - Eliminate (Knora-E), and k-Nearest Oracles - Union (Knora-U) (36). This is a diverse set of algorithms belonging to different families and including single models, ensemble, and dynamic ensemble models (113). This is the same set of classifiers used in the previous Meta-scaler paper (125). As in that paper, we chose this set because in (48), these classifiers showed the highest differences in performances under the use of different STs, i.e., these models are more sensitive to the choice of ST.

4.4.5 Compared methods

To evaluate the Meta-scaler+, we ran an experiment that compared its performances to other methods, which we will refer to via their code names specified below. For all methods below, the meta-models were trained on the 300 datasets from ICPR 2010 and then tested on

the 82 datasets from KEEL, hence, $\mathbf{Q} \notin \mathbb{D}$.

1. **MS+**: This is the proposed Meta-scaler+ method. To evaluate MS+ effectiveness when given unseen classifiers, this procedure was repeated in a Leave One Classifier Out loop, where, at each iteration, one of the 12 base classifiers was removed from $\mathbb{K}$ and assigned to c_q. Therefore, $c_q \notin \mathbb{K}$. The parameters t and k were both set to the value of 5.

2. **Baseline**: As a baseline method, we include a generalist meta-model, trained with information from all base classifiers in $\mathbb{K}$. Instead of training one specialist meta-model λ_i with each Θ_i, we take the mean of the target variables, for each instance (dataset), across all Θ_i for all $i \in \{1, 2, 3, \ldots, n\}$ and form a $\bar{\Theta}$ with which we train a single generalist meta-model. The Leave One Classifier Out procedure was also employed here.

   Note that the Baseline represents the simplest meta-learning method capable of predicting the ST performances for any given classifier (including the unknown). However, unlike MS+, it does not provide specialized, classifier-aware predictions.

3. **Optimistic**: This is as optimistic use-case of MS+ where the query classifier c_q is known, i.e., $c_q \in \mathbb{K}$, therefore there is a meta-model $\lambda_q \in \mathbb{M}$ that is specialized for c_q. In this case, $k = 1$, which causes the selection of only the λ_q meta-model to predict $\hat{\mathbf{p}}$ instead of combining multiple predictions. It can be seen as a best-case scenario for the Meta-scaler+ and is expected to perform better than the other setups. This is very similar to the previous iteration of the Meta-scaler (125), but here the meta-model is a regressor instead of a classifier. The Leave One Classifier Out procedure was not employed here, but we executed this method once for each possible $c_q \in \mathbb{K}$ and took the average performance.

Additionally, to provide a reference when analyzing base-level performances, we also compared the methods to the **Truth**, which can be understood as a hypothetical meta-model that always predicts the exact performances of all scaling techniques, given any base classifier and dataset. Therefore, the **Truth** is the maximum possible performance achievable by a meta-model.

4.4.6 Performance metrics

For the construction of the meta-dataset, the performances of the base classifiers was measured according to the accuracy metric. Although we could have used the F1 or G-mean metrics for more robust results on some datasets that present higher degrees of class imbalance, these metrics are ill-defined and can lead to undefined results in some cases, which ultimately yields a noisy meta-dataset. We also employed the accuracy metric to evaluate and compare the base-level accuracies achieved with the different setups. The accuracy metric is simply defined as the ratio of the number of correctly classified instances over the total number of instances, as in Eq. 4.6.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.6}$$

Where TP (True Positives) and TN (True Negatives) represent the numbers of instances that were correctly classified as positive and negative, respectively. In contrast, FP (False Positives) and FN (False Negatives) represent the number of those incorrectly classified as positive and negative, respectively.

4.4.6.1 Regression metrics

Since the meta-models are regression models, their performance was initially measured according to the regression metrics R^2, MSE (Mean Squared Error), and MAE (Mean Absolute Error). Consider $\mathbf{y}$ as the vector holding the true values of the target variable, $\hat{\mathbf{y}}$ the predicted target values, $\hat{y}_i$ the predicted value of the i-th instance and y_i its corresponding true value, then, for a total of n instances, R^2, MSE and MAE are defined as in Equations 4.7, 4.8, 4.9, respectively.

$$R^2(\mathbf{y}, \hat{\mathbf{y}}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.7}$$

$$\text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.8}$$

$$\text{MAE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.9}$$

4.4.6.2 Classification metrics

As an additional analysis, the regression's output (predicted vector of ST performances $\hat{\mathbf{p}}$), can also be interpreted as a classification result by taking the ST with maximum predicted performance as the predicted class. We used the Accuracy (Eq. 4.6) and the F1 metric (Eq. 4.10) for evaluating these predicted classes. Since our meta-task becomes a multi-class problem, we average the F1 performances calculated for each class individually, which is referred to as *macro* averaging, thus F1-macro.

$$\text{F1} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{4.10}$$

4.4.6.3 Ranking metrics

Additionally, since the output vector can be transformed into a predicted ranking of the STs, we also evaluated the performance in terms of the Normalized Discounted Cumulative Gain (NDCG) metric (100). NDCG is a measure of ranking quality. This metric is typically used to measure the effectiveness of search engine algorithms. We chose this metric because it is able to calculate the quality of rankings where each element has a scalar relevance (i.e., score) associated with them instead of a binary relevance, as in most ranking metrics. To understand the NDCG metric, we will first define the Cumulative Gain (CG), which is the sum of the relevance values of all results in a search result list. The CG at rank position p is defined as in Eq. 4.11.

$$\text{CG}_p = \sum_{i=1}^{p} rel_i \tag{4.11}$$

Here, rel_i is a scalar representing the relevance of the result at a position i. Note that CG is not affected by changes in the order of results in the list as long as these changes occur in a position lower than p. To overcome this, DCG (Discounted Cumulative Gains), as defined in Eq. 4.12, includes a penalization factor to account for highly relevant documents appearing lower in a search result list. From that, NDCG can be calculated as shown in Eq. 4.13.

$$\text{DCG}_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i+1)} \tag{4.12}$$

$$\text{NDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p} \tag{4.13}$$

Where IDCG_p is the ideal DCG (Eq. 4.14), which is the DCG for the best ranking possible.

$$\text{IDCG}_p = \sum_{i=2}^{REL_p} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{4.14}$$

Where REL_p is the sum of the relevances of all results that are more relevant than the result in position p. In our case, the NDCG metric is calculated through a simple analogy: what would be the search results in the typical use of the metric, for us, are the STs. Consequently, their relevances are their corresponding performances.

4.4.7 Software

The experiments were executed using the Python programming language, version 3.12.1, using the Scikit-Learn (17) library, version 1.4.2, PyMFE (78) version 0.4.1, DESlib (28) version 0.3.7 and XGBoost (27) version 2.1.1, and the ImbCoL R library (115), version 1.0.1. More details are available in the paper repository.

4.5 RESULTS AND DISCUSSION

4.5.1 Meta-model performances

In this section, we analyze the meta-model performance, that is, the performance the system achieves in predicting the STs' performances (regression performance), the optimal ST for a given input (classification performance), or the quality of the predicted ranking of STs.

4.5.1.1 Regression metrics

In Figure 16, we show the meta-model performances achieved by the Meta-scaler+ and the two other methods included for comparison. In this figure, we use the three regression metrics. Each bar is a mean of the performance achieved by the corresponding setup when predicting ST performances for each of the 12 query classifiers on the 82 KEEL datasets.

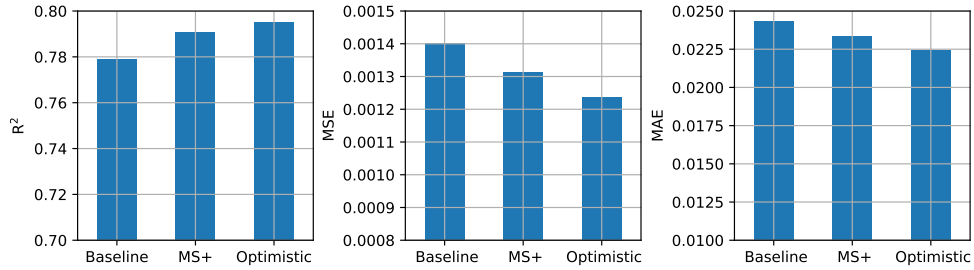


Figure 16 – Mean meta-model performances with regards to the regression metrics: R^2, MSE, MAE.

As can be seen in Figure 16, Meta-scaler+ performances are right in between the baseline generalist approach and the optimistic use case where the query classifier is always known. This means that Meta-scaler+ provides a middle-of-the-way solution between a generalist approach, where one single meta-model learns with data from all available base classifiers to provide a classifier-agnostic prediction, and a limited approach where multiple specialist meta-models are trained but can only be applied to known base classifiers.

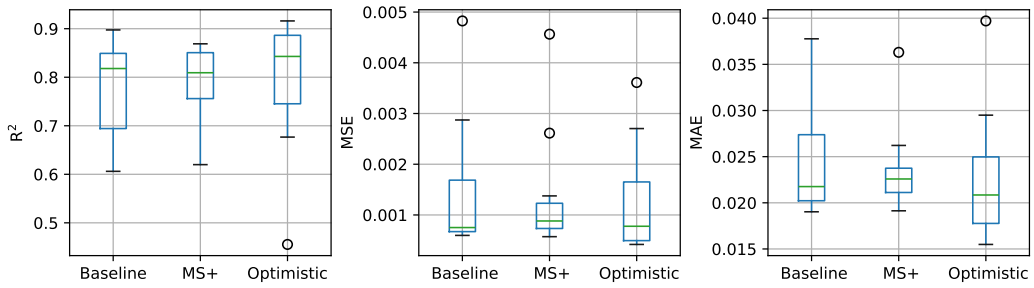


Figure 17 – Box plots of the meta-model performances with regards to the regression metrics: R^2, MSE, MAE.

While Figure 16 provides an important insight into how the different approaches behave on average, the box plots of Figure 17 detail how the performance of each method is distributed. Although these box plots tell a similar story to the bar plots in Figure 16, the wide interquartile ranges and the asymmetry of the distributions around the median show that this analysis of the overall behavior of the meta-models may be missing important information on how well these approaches work for each query classifier. Therefore, in Figure 18 we break down this analysis for each query classifier, we will focus on the R^2 metrics since the behaviors for the other two regression metrics are very similar.

Note, in Figure 18, that Meta-scaler+ shows better R^2 than the baseline approach on 8 query classifiers out of 12 (SVM_lin, GLVQ, Percep, MLP, GP, OLA, MCB, KNORAE), representing 66.6%. Another important observation is that the optimistic method, which was expected to perform better in all situations since it always employs the exact specialist meta-model trained for the query classifier, in fact did not consistently perform better, presenting

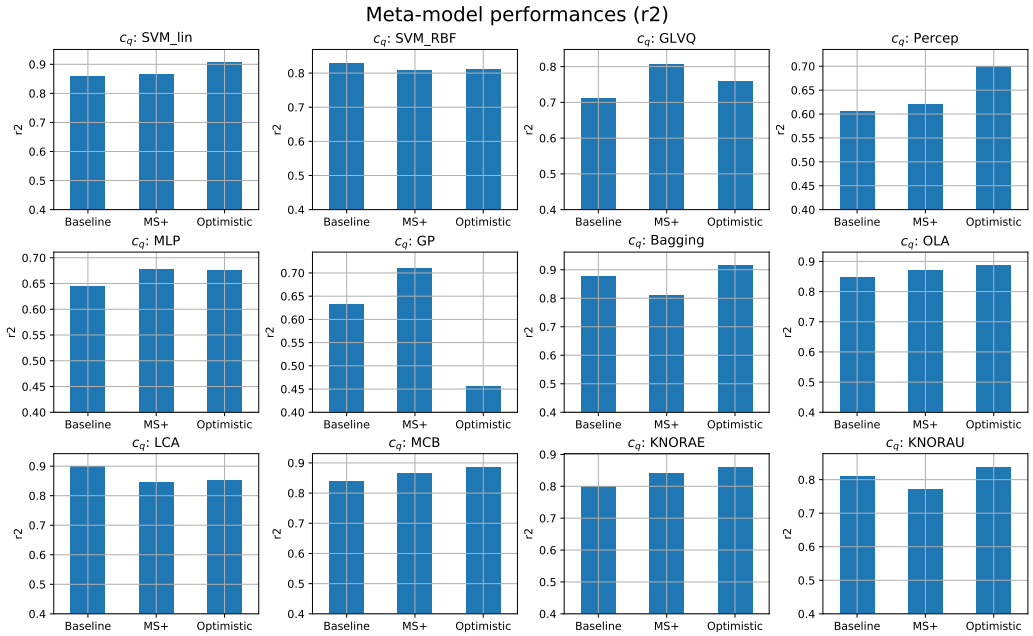


Figure 18 – Meta-model performances (R^2) when predicting for each query classifier.

lower performance than at least one of the other methods in 5 cases. This leads us to believe that for some models, the specialization learned from the training set may be promoting some degree of overfitting, precluding a better generalization on the training set. The fact that the training datasets are mostly synthetic and designed specifically for another study, focused on other matters, may be playing a role here.

This analysis indicates that there is no single better setup to tackle all possible query classifiers when seen from case to case. For four of the query classifiers, the generalist baseline meta-model approach is better suited when compared to *MS+* (SVM_RBF, Bagging, LCA, and KNORAU), possibly because for these, a meta-model trained on a more stable and less noisy meta-dataset (resulting from the means of 12 base classifiers) is more important than specialization. Another possibility is that the specialized meta-models fit well on the training set but do not generalize well to the test set. In contrast, the generalist model trains less fine-tuned models that work better on the unseen test set. On the other hand, for the remaining 8 base classifiers, where MS+ was better, the specialization seems to be more important. Nonetheless, Figures 16 and 17 show that, on average, Meta-scaler+ is a good alternative to a generalist approach.

4.5.1.2 *Classification metrics*

Although the regression metrics provide a comprehensive view of the quality of the ST performance predictions, another crucial aspect to consider is whether the meta-models accurately predict the optimal ST for a specific query classifier and query dataset. To address this, we can derive the label of the ST with maximum predicted performance from the output vector $\hat{\mathbf{p}}$. Consequently, we can view this task as a classification problem and utilize classification metrics to evaluate the meta-model's performance. Figure 19, accordingly, presents the meta-model performances in terms of the accuracy and the F1 (with macro averaging) metrics.

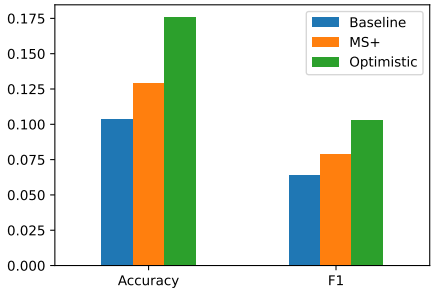


Figure 19 – Mean meta-model performances according to classification metrics: Accuracy and F1.

Similarly to the regression metrics, shown in Figure 16, we notice in Figure 19 that the classification metrics also position Meta-scaler+ in between the baseline generalist approach and the optimistic one, limited to known query classifiers. Figure 20 confirms this pattern, showing a clear advantage of the MS+ against the baseline approach.

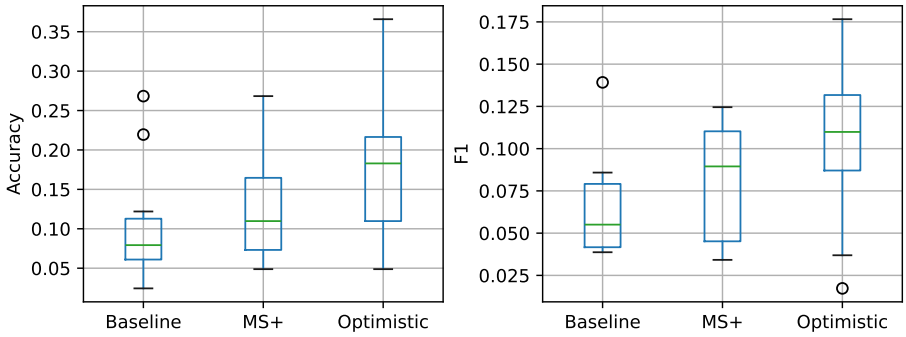


Figure 20 – Box plots of the meta-model performances according to classification metrics: Accuracy and F1.

Again, breaking down this analysis for each query classifier, Figure 21 shows that this pattern where *MS+* is superior to the baseline is not unanimous over the query classifiers, but it happens for 7 out of 12 query classifiers. This validation confirms the suitability of

the proposed Classifier Performance Space as a means of classifier characterization, enabling efficient meta-model combination based on the similarity of a classifier to known classifiers for which specialized meta-models exist. This allows *MS+* to be positioned as an alternative to the generalist approach when one looks for a scaling technique selection method that is not limited to a known set of classifiers.

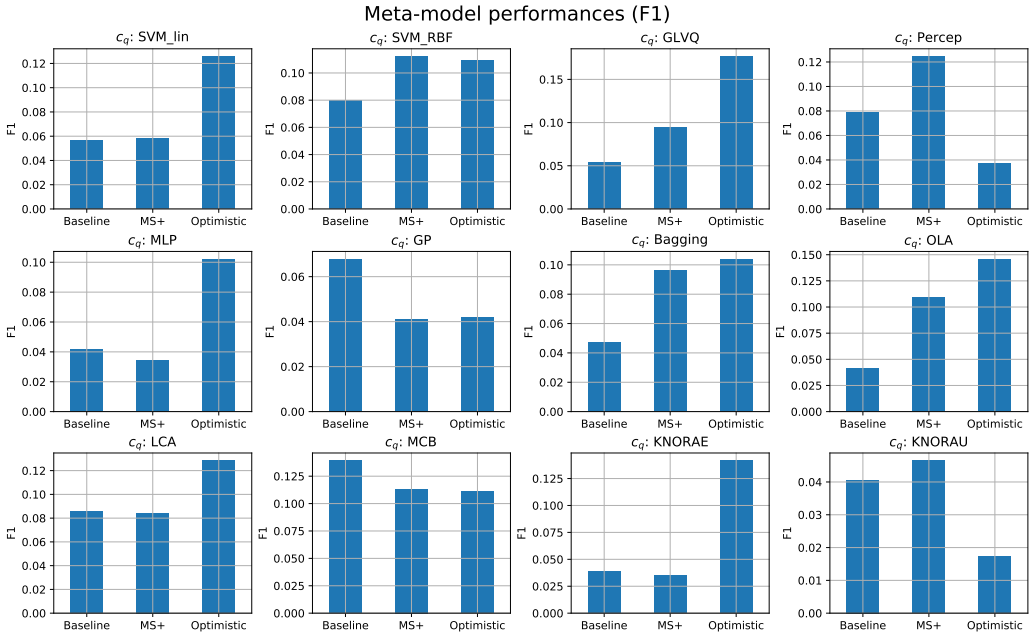


Figure 21 – Meta-model performances (F1) when predicting for each query classifier.

4.5.1.3 Ranking metrics

Another important aspect of this evaluation is to measure how effective the meta-models are for correctly predicting a ranking of the STs for a given query dataset and query classifier. Since the output $\hat{\mathbf{p}}$ of the Meta-scaler+ is a vector where each element represents the performance of each ST in $\mathbb{S}$, assembling a ranking is as simple as sorting this vector. For a real-life application of Meta-scaler+, providing a ranking instead of raw performances or a single best ST gives the users the ability to prioritize the STs to apply to their problems within the limits of their computational budgets.

In Figure 22, the distribution of the NDCG calculated for the rankings predicted by each approach is shown in the box plots. Numerically, comparing the medians, the same result observed for the other metrics can be confirmed here: in general, *MS+* is better than baseline but not as good as when the query classifier is known (*Optimistic*). Although the medians are close to the same value of 0.99, which indicates high-quality rankings. We can see how the

distribution is much more concentrated on high values for the *MS+* than it is for the baseline approach or even the known-classifier limited version (*Optimistic*).

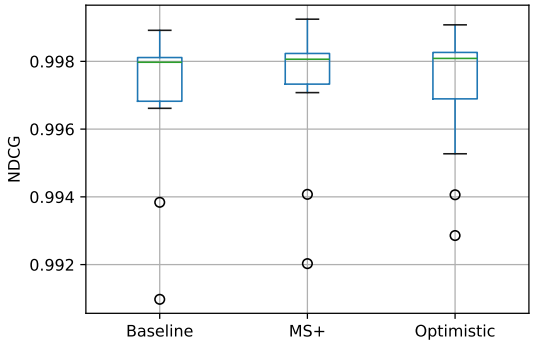


Figure 22 – Meta-model performances according to the NDCG ranking metric.

In addition to the NDCG metric, it is also useful to analyze how each method ranks compared to each others over the 12 query classifiers. Figure 23 shows the distributions of these rankings considering the F1 metric. Note how the MS+ distribution goes lower (better) than the baseline. Although presenting the same median as the baseline, MS+ achieves a better mean ranking.

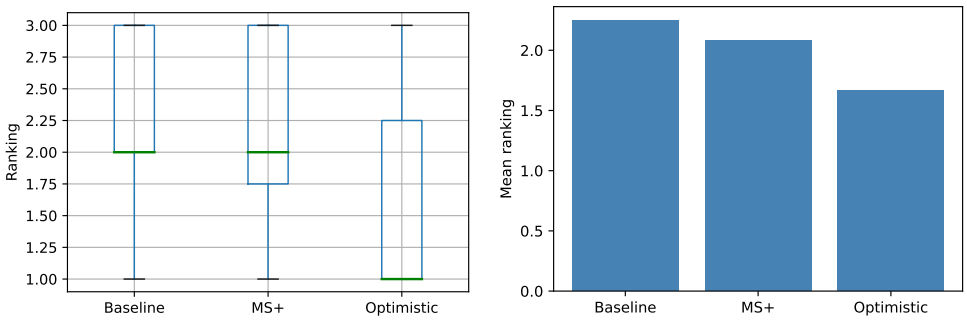


Figure 23 – Meta-model performance (F1) ranking distributions (left) and mean meta-model performance (F1) rankings (right) over the 12 query classifiers. Lower is better.

We explain this average superiority of the Meta-scaler+ versus the generalist baseline approach by its fundamental difference. The baseline method employs a single meta-model trained with averaged results from all base classifiers, which is to say that it tries to generalize the ST-preference behavior of all the classifiers as a single behavior, while they are known to be different (as shown in (48)). On the other hand, Meta-scaler+ takes these differences into account and builds multiple specialist meta-models that are dynamically combined to approximate the behavior of the unknown query classifier. This allows the latter to perform better for more query classifiers than the former.

4.5.2 Base-level performance

Base-level performance refers to the performance achieved by the query classifier when using the recommended ST, which, for this analysis, is the ST with the highest predicted performance by the meta-model. To evaluate Meta-scaler+'s base-level performance, besides comparing it to the baseline method and the optimistic case, we also compare it to the *Truth*, which refers to the performance achieved by a hypothetical meta-model with a perfect prediction. This is the maximum performance that could be obtained.

In Figure 24, the box plots show the distribution of the base-level performances, in terms of the accuracy metric, for all query classifiers. The green horizontal line coincides with the median for the baseline generalist method. If we observe the medians, while *MS+* (0.9623) is numerically superior to both the baseline (0.9608) and *Optimistic* (0.9612), we can conclude that all methods performed very similar and very close to the optimal value presented by the *Truth* (0.9687).

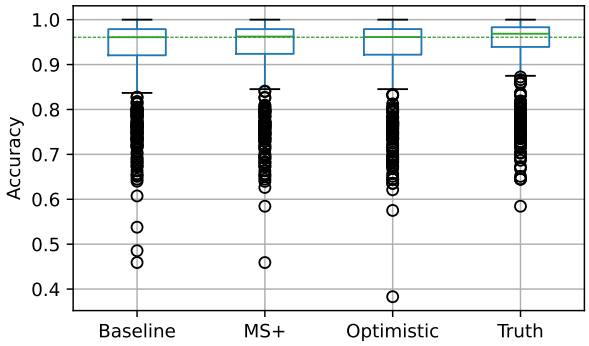


Figure 24 – Box plots of the base-level performances (accuracy) distribution for all query classifiers on the test datasets.

To analyze the performance distribution for each query classifier, we present the individual box plots in Figure 25. These plots show that the distributions are very similar for some classifiers, but some noticeable variability can be observed. The median for *MS+* is above or at least the same as the baseline (green horizontal line) for 9 out of 12 cases, but no significant improvement can be seen in this regard. However, when comparing the interquartile range and the position of the box, we can see that Meta-scaler+ provides improvements over the baseline in a few cases, such as Percep, Bagging, OLA, LCA, and KNORAE, albeit having lower medians in some cases.

To compare the number of Wins, Ties, and Losses that the Meta-scaler+ presents when compared to the generalist and the optimistic approaches, we present in Table 12 these numbers

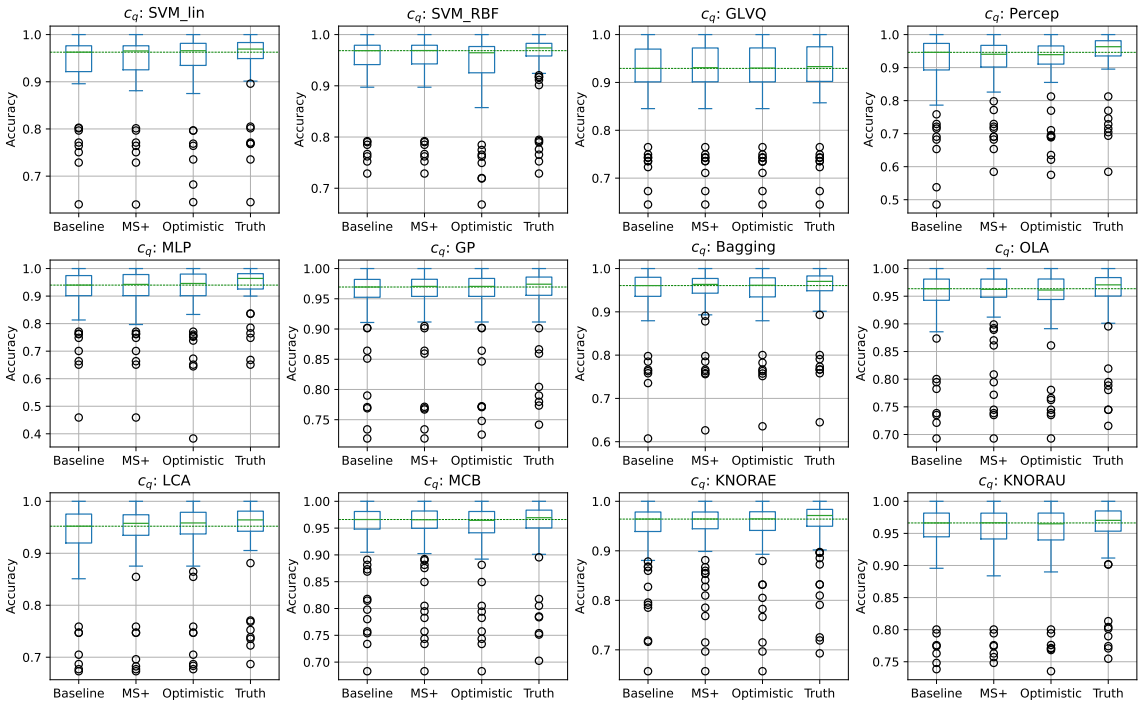


Figure 25 – Box plots of the base-level performances (accuracy) for each query classifier on the test datasets.

for each query classifier over the 82 test datasets and provide the overall numbers on the bottom of the table. We can see that *MS+* achieves more wins, overall, than the baseline and even the optimistic method. However, this superiority is not by a wide margin.

In summary, base-level performances have demonstrated remarkable similarity across the four approaches and, on average, very close to the optimal value represented by the *Truth*. When we take into account that meta-model performance differences were much more expressive, we expected higher differences in the base level. This is an issue that demands further investigation and can possibly be clarified by using different test datasets.

Using a few-shot approach to evaluate the base-level performance, we can see that there is still room for improvement, as shown by the box plots in Figure 26. This figure presents the performance distribution for *MS+* on three few-shot modes: zero-shot – the ST with the maximum predicted performance is used (same as in previous graphs); two-shot – the two STs with the top two predicted performances are selected and applied to the data, the highest performance is considered (best of two); and three-shot – similarly, the best of three. The plots show that taking a two-shot approach improves over the standard zero-shot *MS+*, and the three-shot approach furthers this improvement, getting even closer to the *Truth*. This illustrates how useful the few-shot approach can be in a deployment scenario, and this is possible due to the decision to use regression meta-models instead of classification ones.

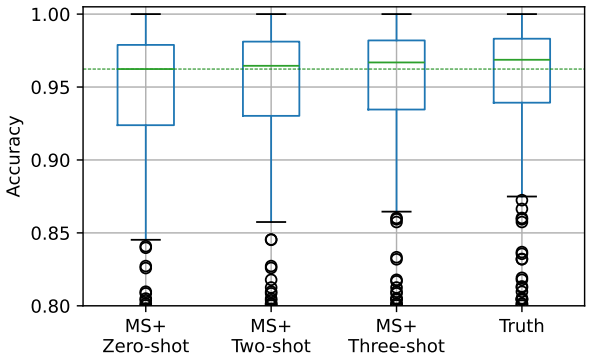


Figure 26 – Box plots of the base-level performances (accuracy) of MS+ zero-shot, two-shot and three-shot modes. For reference, the median of the zero-shot mode is represented by the green dashed line.

Table 12 – Wins (W), ties (T), and losses (L) of the proposed Meta-scaler+ compared to the other methods for each query classifier c_q and in total (bottom). Lines where the Meta-scaler+ presents an advantage are indicated with a check mark in the right-most column.

c_q	Comparing MS+ vs.	W	T	L	W ≥ L
SVM_lin	Optimistic	7	69	6	✓
	Baseline	16	35	31	
SVM_RBF	Optimistic	11	146	7	✓
	Baseline	47	71	46	✓
GLVQ	Optimistic	23	211	12	✓
	Baseline	53	145	48	✓
Percep	Optimistic	39	259	30	✓
	Baseline	93	159	76	✓
MLP	Optimistic	51	325	34	✓
	Baseline	108	219	83	✓
GP	Optimistic	57	394	41	✓
	Baseline	115	281	96	✓
Bagging	Optimistic	78	423	73	✓
	Baseline	133	310	131	✓
OLA	Optimistic	86	491	79	✓
	Baseline	148	367	141	✓
LCA	Optimistic	111	524	103	✓
	Baseline	167	397	174	✓
MCB	Optimistic	119	592	109	✓
	Baseline	179	462	179	✓
KNORAE	Optimistic	123	662	117	✓
	Baseline	194	505	203	
KNORAU	Optimistic	135	709	140	
	Baseline	209	549	226	
Totals	Optimistic	840	4805	751	✓
	Baseline	1462	3500	1434	✓

4.5.3 Comparison with the state of the art

In our previous Meta-scaler study (125), we presented a comparison with state of the art, where we contrasted the results of that version of Meta-scaler with the results of the two other methods, Jain et al. (6) and Zagatti et al. (52), on the same 300 datasets from ICPR 2010. Here, we follow the same methodology for comparing Meta-scaler+ to the previous Meta-scaler and the two other studies; therefore, we compare Meta-scaler+ to three state-of-the-art

methods. All four methods were executed within a Leave One Dataset Out cross-validation procedure. The results from the three state-of-the-art methods were obtained from the previous study's repository (125) and are detailed therein.

In Table 13, we show the mean base-level classification performances (F1) of each query classifier when the ST is selected by the Meta-scaler+, the Meta-scaler, the other two state-of-the-art methods, or the Truth. Notice that the previous study's Meta-scaler is the best performer for 11 of 12 query classifiers and presents the best median, mean, and mean ranking. Recall that Meta-scaler is an optimistic implementation that only works for known query classifiers; therefore, its evaluation only considered known query classifiers. Nonetheless, Meta-scaler+ is the second-best performer, as expressed by its median, mean, and mean ranking.

The last row of Table 13 shows the p-value resulting from a pair-wise comparison, using a Wilcoxon signed-rank hypothesis test, between Meta-scaler+ and the other methods. The null hypothesis is that the compared methods are not different. Since we are doing multiple pair-wise comparisons, we control for familywise type-I error using a Bonferroni correction, which, starting with a standard $\alpha = 0.05$, leads to a corrected significance level of 0.0042. While the Meta-scaler was shown in the previous study to be significantly different than the other two methods, here the p-values were higher, and no method has been shown to be significantly different than the Meta-scaler+.

Table 13 – Mean base-level performance (F1) achieved by the 12 query classifiers when using Meta-scaler+ and three state-of-the-art ST selection methods. Best method is bolded.

c_q	**Meta-scaler+**	**Meta-scaler**	**Jain et al.**	**Zagatti et al.**	**Truth**
Bagging	0.7129	**0.7170**	0.7129	0.6638	0.7311
GLVQ	0.6317	**0.6576**	0.6250	0.5954	0.6642
GP	0.7427	**0.7522**	0.7055	0.7120	0.7671
KNORAE	0.7409	**0.7434**	0.7224	0.7133	0.7593
KNORAU	0.7319	**0.7382**	0.7183	0.6948	0.7486
LCA	0.6768	**0.6788**	0.6766	0.6095	0.7207
MCB	0.7430	**0.7431**	0.7189	0.7129	0.7600
MLP	0.6823	**0.7008**	0.6876	0.5790	0.7358
OLA	0.7417	**0.7437**	0.7190	0.7131	0.7597
Percep	0.6679	0.6702	**0.6789**	0.5986	0.7249
SVM_RBF	0.7232	**0.7399**	0.7261	0.6541	0.7483
SVM_lin	0.6976	**0.7071**	0.7023	0.5585	0.7143
Median	0.7180	**0.7276**	0.7092	0.6589	0.7420
Mean	0.7077	**0.7160**	0.6995	0.6504	0.7362
Stddev	0.0365	0.0328	0.0289	0.0593	0.0285
Mean rank	2.4167	**1.0833**	2.5833	3.9167	N/A
p-values	N/A	1.6e-02	1.3e-01	6.3e-03	N/A

In Figure 27, we can visually inspect the distribution of the performances along the 12 query classifiers for each ST selection method. Notice how Meta-scaler+ is positioned right

behind Meta-scaler. Both Meta-scaler and Meta-scaler+ have distributions that are much more concentrated on higher values than the two other state-of-the-art methods.

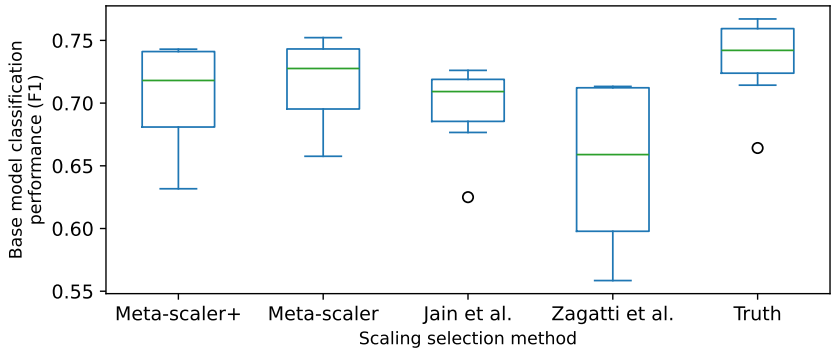


Figure 27 – Box plots of the base-level performances (F1) of Meta-scaler+, the state-of-the-art methods and the Truth.

Overall, we interpret Table 13 and Figure 27 as very positive results for Meta-scaler+, they indicate that even though Meta-scaler+ is not limited to known classifiers, it still obtains base-level performances that are on par with the optimistic Meta-scaler and visibly superior to the remaining methods. In fact, Meta-scaler, although superior to Meta-scaler+, is not significantly different from it, according to the Wilcoxon signed-rank test. This ratifies the effectiveness of the bundling of the Classifier Performance Space and the dynamic meta-regressor combination as an effective alternative to limiting the recommender to known query classifiers. It confirms Meta-scaler+ as a true advancement over Meta-scaler since it adds functionality, removes limitations and still performs equivalently.

4.5.4 Tuning the Classifier Performance Space

To adjust the Classifier Performance Space to the settings used in the experiments just presented, we ran a series of tests comparing the meta-model performance achieved with different values for the parameter t, which controls the number of datasets used for building the CPS.

Additionally, we also created a variant of the CPS, which we refer to as CPS', that, instead of having the performances on the t datasets as its dimensions, also includes the performances on the scaled versions of these datasets. Therefore, the number of dimensions of CPS' is $t \times l$, where l is the number of STs. The main reason to use CPS′ instead of CPS is that, by adding the performances of classifiers on scaled datasets, CPS′ contains information about the effects of STs on the classifiers, which could be helpful since our goal is to use the Classifier

Performance Space to select meta-models trained for classifiers that behave similarly to the query classifier in terms of ST preference.

In Figure 28, we show the distribution of meta-model performances achieved by the 12 query classifiers when using the ST recommended by the Meta-scaler+ under different CPS settings. The value for t varies from 5 to 299, and the two variants of the CPS are compared (CPS - conventional, CPS' - includes the scaled datasets).

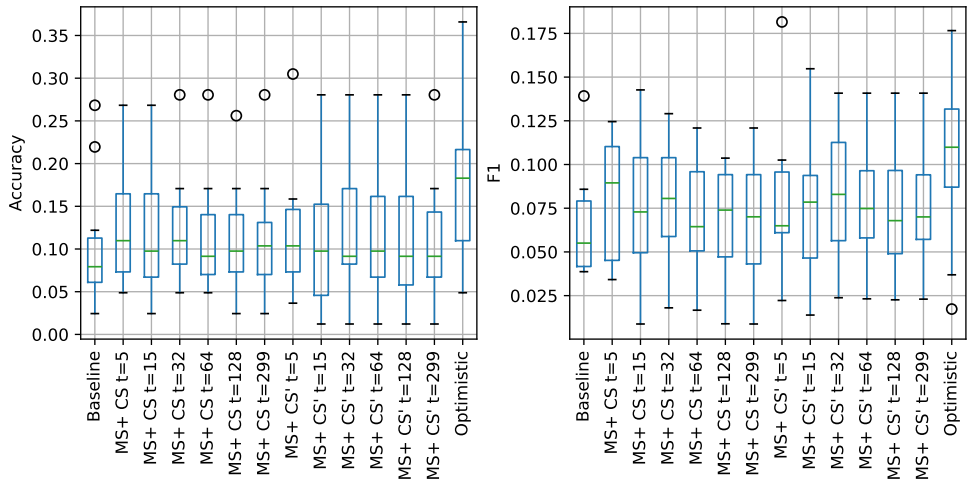


Figure 28 – Box plots of the meta-model performances (Accuracy and F1) achieved by the query classifiers using Meta-scaler+ with different CPS settings.

Note that, contrary to what could be expected, increasing t, which adds dimensions to the CPS, does not necessarily incur higher meta-model performance, and neither does replacing CPS with CPS'. The best-performing MS+ is the one that uses $t = 5$ and the simplest CPS. Although *MS+ CPS t=32* seems to perform better regarding the accuracy's median, the improvement does not justify the cost of running c_q on the additional 27 datasets to position it in the CPS. Therefore, the simplest CPS and the lowest $t = 5$, which were used throughout this study, are the optimal choices considering this comparison. This is a very positive result since it is the option with the lowest computational cost. Nonetheless, further investigation can be done by adjusting t for values closer to 5 and also by varying the values of other MS+ parameters outside of the scope of the CPS, such as the parameter k that controls the number of meta-models to be combined in the recommendation phase.

4.6 LIMITATIONS

Hyperparameter tuning: We empirically verified that Meta-scaler+ is significantly sensitive to the choice of its parameters, especially t and k. We did not invest enough time and

effort into understanding their influence and finding their optimal values.

Subset $\mathbb{L}$: In the selection of t datasets from $\mathbb{D}$ to form the Classifier Performance Space, we employed the *k-means++* algorithm to find good representatives of $\mathbb{D}$. However, albeit being a big improvement over random selection, *k-means++* is still strongly dependent on the choice of the first dataset selected, which is random. We noticed that slight changes in the random seed provided to the algorithm caused visible changes in Meta-scaler+'s performance.

Meta-dataset: The meta-dataset used to train the meta-models was selected due to its diversity. It is a collection of mostly synthetic datasets originally designed to cover a range of values for a set of complexity measures. We, therefore, acknowledge that this complexity diversity may not directly translate to diversity in terms of sensitivity to scale. More research is needed to determine if this is a suitable dataset or to define a better dataset selection method for our goal of training meta-models capable of learning the relationship between the meta-features and the performances of STs.

Dataset representation: As shown in our previous work (125), there is an important variation in how the meta-models use the different meta-features when they are trained for different base classifiers. We previously concluded that the Meta-scaler may benefit from better meta-features or even a meta-feature selection step that depends on the base classifier for which the meta-model is being trained. This idea was not investigated in the current study.

Test datasets: The similarity of the base-level performance results among the different methods and the *Truth* may indicate that we need more control and curation over the test data.

4.7 CONCLUSION

Our main goal in this study was to advance the Meta-scaler framework by removing the previous limitation on the choice of the query classifier. To solve this problem, we introduced in the Meta-scaler+, the Classifier Performance Space, a novel method to characterize a classifier by extracting performance meta-data from it. Additionally, Meta-scaler+ employs an ensemble of regressor meta-models, where the multiple decisions are combined in a weighted manner,

according to the position of the query classifier in the Classifier Performance Space. This is presented as an alternative to a generalist meta-model approach.

The proposed method, Meta-scaler+, effectively eliminates the previous limitation when selecting the query classifier. This allows for the prediction of ST performances and, consequently, ST recommendations for both known and unknown classifiers with satisfactory results. Using regression instead of classification allows more flexibility in interpreting the results, which can be delivered as a single ST or a ranking of STs along with their predicted performances.

The results show a strong tendency for Meta-scaler+ to outperform the baseline generalist method regarding meta-model performance, achieving more precise predictions and higher quality rankings of STs. It is possible that with fine-tuning of the parameters and a better dataset selection heuristic for the Classifier Performance Space, meta-model performance can be further improved. In terms of base-level performance, Meta-scaler+ only slightly outperforms the baseline method, even though it consistently obtains a higher number of wins for most query classifiers. Future work can focus on better data curation (both for training and testing collections) to better showcase the potential of the Meta-scaler+, fine-tuning the Classifier Performance Space and better dataset representation.

5 CONCLUSION AND FUTURE WORK

This thesis provides a comprehensive exploration of scaling techniques and their effects on classification performance and provides practical solutions to the problem of ST selection through three distinct but interconnected studies. These studies collectively address critical challenges in dataset scaling a classifier-specific ST recommendation providing a promising research direction for the automation of preprocessing steps, contributing to the fields of meta-learning and AutoML.

The first study establishes the foundational insight that the choice of scaling technique significantly affects classification performance and that choosing the wrong ST can even be worse than not scaling the data at all. The results underscore the need for methodical scaling technique selection. Furthermore, it reveals that different classifiers have unique preferences for scaling techniques, making the case for classifier-aware scaling recommendations.

Building on these insights, the second study introduces the Meta-scaler framework, a meta-learning-based solution that employs specialized meta-models to recommend optimal scaling techniques for known classifiers. By leveraging dataset meta-features to build classifier-specialized meta-models, Meta-scaler consistently outperforms static choices of scaling techniques and existing meta-learning methods. However, its reliance on prior knowledge of the query classifier represents a key limitation, which is addressed in the subsequent study.

The third study extends Meta-scaler's capabilities by introducing Meta-scaler+, which integrates the innovative "Classifiers' Space" concept. This allows for specialized scaling recommendations even when the query classifier is unknown. Meta-scaler+ achieves competitive performance with its predecessor for known classifiers and demonstrates superior results for unknown classifiers compared to state-of-the-art methods. Additionally, its ability to provide ranked scaling recommendations enhances its flexibility and applicability.

This thesis demonstrates the pivotal role of dataset scaling in classification pipelines and establishes meta-learning as a powerful approach for ST selection and potentially for other preprocessing steps. The contributions of Meta-scaler and Meta-scaler+ address the challenge of dataset and classifier-specific scaling recommendations while also laying the groundwork for broader applications in AutoML pipelines.

For each question answered in this thesis, luckily for us, several others were raised. It opens new avenues for our future research. Enhancements to dataset representation through

improved meta-features, better initialization of the Classifiers' Space, and refined dataset selection heuristics could further improve the performance of Meta-scaler+. Additionally, integrating these frameworks into widely used machine learning libraries would facilitate their adoption in real-world applications. Finally, extending the principles of Meta-scaler and Meta-scaler+ to other preprocessing steps and pipeline components could lead to the development of holistic AutoML systems. These ideas are detailed below.

5.1 FUTURE WORK

We list below a series of ideas we intend to invest on after this doctorate thesis. These ideas are concentrated on improving the Meta-scaler+ framework, but some may have wider implications that can lead to new research fronts:

Better understand the influences of specific meta-features: In Chapter 3 we showed how different meta-features are more or less important for meta-model inference depending on the type of query classifier for which the meta-model is trained. It seems that a more detailed analysis into this subject can lead to important insights into how to better curate a set of meta-features to improve performance. One idea is to perform such a selection of meta-features independently for each specialist meta-model trained.

Improve datasets' representation: Going further than just optimizing a subset of the existing, classical meta-features, we intend to also invest in better dataset representation. While the classical meta-features used throughout this work do provide enough predictive power to create efficient ST recommendation meta-models, we believe it can be improved if we use meta-features that are designed specifically for this task. One idea is to propose a neural network model capable of transforming a set of datasets into vectors of meta-features which create a space that we can try to approximate to the meta-target space by seeking to minimize a certain dissimilarity metric or metrics. We started some preliminary work in this direction and we have seen potential in the automatic creation of efficient task-specific meta-features.

Tune Meta-scaler+ hyperparameters: Meta-scaler+ seems to be sensitive to the choice of its hyperparameters. More effort must be directed to finding optimal values for t, as seen in Section 4.5, and k, that controls the number of meta-models to combine,

for example. Other parameters may also be explored, such as those that control how the meta-models weights decay with the distance from their associated classifier to the query classifier in the Classifier Performance Space, or even what kind of distance metric should be used.

Improve heuristic for selecting the t datasets to compose $\mathbb{L}$: As we have pointed out in Section 4.6, the *k-means++* algorithm used for selecting a subset of t datasets for the Classifier Performance Space is very dependent on its initialization, that is, on the first randomly selected dataset. We intend to seek more efficient and deterministic ways to define an optimal subset. Alternatively, we may be able to find a fixed subset of small and representative datasets for the Meta-scaler+ using optimization techniques.

Improve dataset curation: As also acknowledged in Section 4.6, the collection of datasets used for training the Meta-scaler may not be a good representation of general datasets in the real world. In their original study (106), they were generated to cover a complexity space. As we pointed out, while this yields a much-desired diversity in the collection, it may not directly translate into diversity in terms of behavior under different scaling techniques. To tackle this, we intend to generate a new collection of training datasets stemming from real-world datasets, manipulating attribute scales to generate a portion of intentionally "descaled" datasets that can help induce better learning about ST selection by the meta-models. A final collection of datasets with a balance of original and "descaled" datasets could then be used to train a new instance of the Meta-scaler+ to be evaluated on real-world datasets.

Multiple, per-step specialized meta-models for full AutoML: When we look at the AutoML literature that employs meta-learning as its main recommendation method, we notice that most of the proposed methods rely on a single meta-model to predict a whole pipeline. However, the success that Meta-scaler and Meta-scaler+ have demonstrated in using specialized meta-models within a single pipeline step (ST) recommendation may indicate that using multiple, per-step specialized meta-models for a full pipeline prediction may be a promising target to pursue. The same principles used in Meta-scaler and Meta-scaler+ may be extended to other pipeline steps, and a combination of several Meta-*step* recommenders may yield a better pipeline recommendation than a single do-it-all meta-model.

Turn Meta-scaler+ into an accessible tool: While all the code for the three papers is available in open-source repositories, we acknowledge that Meta-scaler+ is not yet in an easy-to-use form where any user can easily apply it to their dataset and classifier in a *plug-n-play* fashion. Some user-side effort is still required to adjust code to their needs. Therefore, we intend to invest in making Meta-scaler+ a practical and accessible tool, enabling its integration with popular machine-learning libraries, such as Scikit-learn.

BIBLIOGRAPHY

1 RICE, J. R. The Algorithm Selection Problem. *Adv. Comput.*, v. 15, p. 65–118, 1 1976. Disponível em: <http://docs.lib.purdue.edu/cstech/99>.

2 SMITH-MILES, K. A. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.*, v. 41, n. 1, 1 2009.

3 SINGH, D.; SINGH, B. Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, Elsevier B.V., v. 97, p. 105524, 2020. ISSN 15684946. Disponível em: <https://doi.org/10.1016/j.asoc.2019.105524>.

4 MISHKOV, O.; ZORIN, K.; KOVTONIUK, D.; DEREKO, V.; MORGUN, I. Comparative Analysis of Normalizing Techniques Based on the Use of Classification Quality Criteria. In: *Lecture Notes on Data Engineering and Communications Technologies*. [s.n.], 2022. v. 77, p. 602–612. Disponível em: <https://link.springer.com/10.1007/978-3-030-82014-5_41>.

5 ALCALÁ-FDEZ, J.; FERNÁNDEZ, A.; LUENGO, J.; DERRAC, J.; GARCÍA, S.; SÁNCHEZ, L.; HERRERA, F. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, v. 17, n. 2-3, p. 255–287, 2011. ISSN 15423980.

6 JAIN, S.; SHUKLA, S.; WADHVANI, R. Dynamic selection of normalization techniques using data complexity measures. *Expert Systems with Applications*, Elsevier Ltd, v. 106, p. 252–262, 2018. ISSN 09574174. Disponível em: <https://doi.org/10.1016/j.eswa.2018.04.008>.

7 DZIERŻAK, R. Comparison of the Influence of Standardization and Normalization of Data on the Effectiveness of Spongy Tissue Texture Classification. *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*, v. 9, n. 3, p. 66–69, 2019. ISSN 2083-0157.

8 RAJU, V. N. G.; LAKSHMI, K. P.; JAIN, V. M.; KALIDINDI, A.; PADMA, V. Study the Influence of Normalization/Transformation process on the Accuracy of Supervised Classification. In: *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2020. p. 729–735. ISBN 978-1-7281-5821-1. Disponível em: <https://ieeexplore.ieee.org/document/9214160/>.

9 CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, v. 41, p. 195–216, 5 2018. ISSN 15662535. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1566253517304074>.

10 ERIKSSON, L. *Introduction to multi-and megavariate data analysis using projection methods (PCA & PLS)*. [S.l.]: Umetrics AB, 1999. 213–225 p.

11 KEUN, H. C.; EBBELS, T. M.; ANTTI, H.; BOLLARD, M. E.; BECKONERT, O.; HOLMES, E.; LINDON, J. C.; NICHOLSON, J. K. Improved analysis of multivariate data by variable stability scaling: Application to NMR-based metabolic profiling. *Analytica Chimica Acta*, v. 490, n. 1-2, p. 265–276, 2003. ISSN 00032670.

12 HU, Y.; GRIPON, V.; PATEUX, S. Leveraging the Feature Distribution in Transfer-Based Few-Shot Learning. In: FARKAŠ, I.; MASULLI, P.; OTTE, S.; WERMTER, S. (Ed.). *Artificial Neural Networks and Machine Learning – ICANN 2021*. Cham: Springer International Publishing, 2021. p. 487–499. ISBN 978-3-030-86340-1.

13 Scikit-learn Developers. *6.3. Preprocessing data - scikit-learn 1.1.2 documentation*. 2022. Disponível em: <https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-transformer>.

14 COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, v. 13, n. 1, p. 21–27, 1 1967. ISSN 0018-9448. Disponível em: <http://ieeexplore.ieee.org/document/1053964/>.

15 SATO, A.; YAMADA, K. Generalized Learning Vector Quantization. *Proceedings of the 8th International Conference on Neural Information Processing Systems*, p. 423–429, 1996.

16 CORTES, C.; VAPNIK, V. Support-Vector Networks. *Machine Learning*, v. 20, n. 3, p. 273–297, 1995. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-34249753618&doi=10.1023%2FA%3A1022627411411&partnerID=40&md5=97a8591c7d55575e8c48344379ee2796>.

17 PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; Michel V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

18 RISH, I. An Empirical Study of the Naïve Bayes Classifier An empirical study of the naive Bayes classifier. *IJCAI 2001 workshop on empirical methods in artificial intelligence*, v. 3, n. 22, p. 41–46, 2001. Disponível em: <https://www.cc.gatech.edu/~isbell/reading/papers/Rish.pdf>.

19 ZHANG, H. The optimality of Naive Bayes. In: *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*. [s.n.], 2004. v. 2, p. 562–567. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-10044283198&partnerID=40&md5=22bb4a5d2ea3874e9f371f5f355f5472>.

20 SEEGER, M. Gaussian Processes for Machine Learning. *International Journal of Neural Systems*, v. 14, n. 02, p. 69–106, 4 2004. ISSN 0129-0657. Disponível em: <https://www.worldscientific.com/doi/abs/10.1142/S0129065704001899>.

21 TUNG, A. K. H. Rule-based Classification. In: LIU, L.; ÖZSU, M. T. (Ed.). *Encyclopedia of Database Systems*. Boston, MA: Springer US, 2009. p. 2459–2462. ISBN 978-0-387-39940-9. Disponível em: <https://doi.org/10.1007/978-0-387-39940-9_559>.

22 BREIMAN, L.; FRIEDMAN, J. H.; OLSHEN, R. A.; STONE, C. J. *Classification And Regression Trees*. Routledge, 2017. 1–358 p. ISBN 9781315139470. Disponível em: <https://www.taylorfrancis.com/books/9781351460491>.

23 AGGARWAL, C. C. *Neural Networks and Deep Learning*. 1. ed. Cham: Springer International Publishing, 2018. 497 p. ISBN 978-3-319-94462-3. Disponível em: <http://link.springer.com/10.1007/978-3-319-94463-0>.

24 KUNCHEVA, L. I. *Combining Pattern Classifiers: Methods and Algoritms*. 2nd. ed. [S.l.]: John Wiley and Sons, 2014. 351 p. ISBN 9781118315231.

25 ZHOU, Z.-H. *Ensemble Methods, Foundations and Algorithms*. [S.l.]: Chapman & Hall/CRC, 2012. 232 p. ISBN 9781439830055.

26 TULYAKOV, S.; JAEGER, S.; GOVINDARAJU, V.; DOERMANN, D. Review of Classifier Combination Methods. In: MARINAI, S.; FUJISAWA, H. (Ed.). *Machine Learning in Document Analysis and Recognition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 361–386. ISBN 978-3-540-76280-5. Disponível em: <https://doi.org/10.1007/978-3-540-76280-5_14>.

27 CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2016. v. 13-17-Augu, n. 8, p. 785–794. ISBN 9781450342322. Disponível em: <https://dl.acm.org/doi/10.1145/2939672.2939785>.

28 CRUZ, R. M. O.; HAFEMANN, L. G.; SABOURIN, R.; CAVALCANTI, G. D. C. DESlib: A Dynamic ensemble selection library in Python. *Journal of Machine Learning Research*, v. 21, n. 8, p. 1–5, 2020.

29 BREIMAN, L. Bagging predictors. *Machine Learning*, v. 24, n. 2, p. 123–140, 8 1996. ISSN 0885-6125. Disponível em: <http://link.springer.com/10.1007/BF00058655>.

30 BREIMAN, L. Random forests. *Machine Learning*, Springer, VAN GODEWIJCKSTRAAT 30, 3311 GZ DORDRECHT, NETHERLANDS, v. 45, n. 1, p. 5–32, 10 2001. ISSN 0885-6125.

31 FREUND, Y.; SCHAPIRE, R. E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, v. 55, n. 1, p. 119–139, 8 1997. ISSN 00220000. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S002200009791504X>.

32 FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, v. 29, n. 5, p. 1189–1232, 2001. ISSN 00905364.

33 WOODS, K.; KEGELMEYER, W. P.; BOWYER, K. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 19, n. 4, p. 405–410, 1997.

34 GIACINTO, G.; ROLI, F. Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*, v. 34, n. 9, p. 1879–1881, 2001. ISSN 00313203.

35 BRITTO, A. S.; SABOURIN, R.; OLIVEIRA, L. E. Dynamic selection of classifiers—A comprehensive review. *Pattern Recognition*, v. 47, n. 11, p. 3665–3680, 11 2014. ISSN 00313203. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0031320314001885>.

36 KO, A. H.; SABOURIN, R.; BRITTO JR., A. S. From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition*, v. 41, n. 5, p. 1718–1731, 5 2008. ISSN 00313203. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0031320307004499>.

37 KUNCHEVA, L. I. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 24, n. 2, p. 281–286, 2002. ISSN 01628828.

38 AKOSA, J. S. Predictive accuracy : A misleading performance measure for highly imbalanced data. *SAS Global Forum*, v. 942, p. 1–12, 2017. Disponível em: <https://support.sas.com/resources/papers/proceedings17/0942-2017.pdf>.

39 DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml>.

40 VIRTANEN, P.; GOMMERS, R.; OLIPHANT, T. E.; HABERLAND, M.; REDDY, T.; COURNAPEAU, D.; BUROVSKI, E.; PETERSON, P.; WECKESSER, W.; BRIGHT, J.; WALT, S. J. van der; BRETT, M.; WILSON, J.; MILLMAN, K. J.; MAYOROV, N.; NELSON, A. R. J.; JONES, E.; KERN, R.; LARSON, E.; CAREY, C. J.; POLAT, I.; FENG, Y.; MOORE, E. W.; VANDERPLAS, J.; LAXALDE, D.; PERKTOLD, J.; CIMRMAN, R.; HENRIKSEN, I.; QUINTERO, E. A.; HARRIS, C. R.; ARCHIBALD, A. M.; RIBEIRO, A. H.; PEDREGOSA, F.; MULBREGT, P. van; SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, v. 17, p. 261–272, 2020.

41 CAVALIN, P. R.; SABOURIN, R.; SUEN, C. Y. Dynamic Selection of Ensembles of Classifiers Using Contextual Information. In: GAYAR, N. E.; KITTLER, J.; ROLI, F. (Ed.). *Multiple Classifier Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 145–154. ISBN 978-3-642-12127-2.

42 CAVALIN, P. R.; SABOURIN, R.; SUEN, C. Y. Dynamic selection approaches for multiple classifier systems. *Neural Computing and Applications*, v. 22, n. 3-4, p. 673–688, 3 2013. ISSN 0941-0643. Disponível em: <https://doi.org/10.1007/s00521-011-0737-9>.

43 SOUZA, M. A.; CAVALCANTI, G. D.; CRUZ, R. M.; SABOURIN, R. Online local pool generation for dynamic classifier selection. *Pattern Recognition*, v. 85, n. 1, p. 132–148, 2019. ISSN 00313203.

44 CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D.; REN, T. I. META-DES: A dynamic ensemble selection framework using meta-learning. *Pattern Recognition*, v. 48, n. 5, p. 1925–1935, 5 2015. ISSN 00313203. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0031320314004919>.

45 CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. META-DES.H: A Dynamic Ensemble Selection technique using meta-learning and a dynamic weighting approach. *Proceedings of the International Joint Conference on Neural Networks*, v. 2015-Septe, n. July, 2015.

46 AKSOY, S.; HARALICK, R. M. Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern Recognition Letters*, North-Holland, v. 22, n. 5, p. 563–582, 4 2001. ISSN 0167-8655.

47 GARCÍA, S.; LUENGO, J.; HERRERA, F. *Data Preprocessing in Data Mining*. [S.l.: s.n.], 2015. v. 72. ISSN 18684408. ISBN 9783319102467.

48 AMORIM, L. B. V. de; CAVALCANTI, G. D.; CRUZ, R. M. The choice of scaling technique matters for classification performance. *Applied Soft Computing*, v. 133, p. 109924, 1 2023. ISSN 15684946. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1568494622009735>.

49 VANSCHOREN, J. Meta-Learning. In: KOTTHOFF, L. H. F.; JOAQUIN, V. (Ed.). *Automated Machine Learning: Methods, Systems, Challenges*. Cham: Springer International Publishing, 2019. p. 35–61. ISBN 978-3-030-05318-5. Disponível em: <https://doi.org/10.1007/978-3-030-05318-5_2>.

50 BILALLI, B.; ABELLÓ, A.; ALUJA-BANET, T.; WREMBEL, R. Automated data pre-processing via meta-learning. *Lecture Notes in Computer Science*, Springer Verlag, v. 9893 LNCS, p. 194–208, 2016. ISSN 16113349.

51 GEMP, I.; THEOCHAROUS, G.; GHAVAMZADEH, M. Automated Data Cleansing through Meta-Learning. *AAAI Conference on AI*, Association for the Advancement of Artificial Intelligence (AAAI), v. 31, n. 2, p. 4760–4761, 2 2017. ISSN 2374-3468. Disponível em: <https://ojs.aaai.org/index.php/AAAI/article/view/19107>.

52 ZAGATTI, F. R.; SILVA, L. C.; SILVA, L. N. D. S.; SETTE, B. S.; CASELI, H. D. M.; LUCREDIO, D.; SILVA, D. F. MetaPrep: Data preparation pipelines recommendation via meta-learning. *Proceedings - 20th IEEE International Conference on Machine Learning and Applications, ICMLA 2021*, Institute of Electrical and Electronics Engineers Inc., p. 1197–1202, 2021.

53 MOHR, F.; WEVER, M. Naive automated machine learning. *Machine Learning*, Springer, v. 112, n. 4, p. 1131–1170, 4 2023. ISSN 15730565.

54 CHAPELLE, O.; VAPNIK, V.; BOUSQUET, O.; MUKHERJEE, S. Choosing Multiple Parameters for Support Vector Machines. *Machine Learning*, v. 46, n. 1/3, p. 131–159, 2002. ISSN 08856125.

55 BISCHL, B.; KERSCHKE, P.; KOTTHOFF, L.; LINDAUER, M.; MALITSKY, Y.; FRÉCHETTE, A.; HOOS, H.; HUTTER, F.; LEYTON-BROWN, K.; TIERNEY, K.; VANSCHOREN, J. ASlib: A benchmark library for algorithm selection. *Artificial Intelligence*, Elsevier B.V., v. 237, p. 41–58, 8 2016. ISSN 00043702.

56 KHAN, I.; ZHANG, X.; REHMAN, M.; ALI, R. A Literature Survey and Empirical Study of Meta-Learning for Classifier Selection. *IEEE Access*, v. 8, p. 10262–10281, 2020. ISSN 21693536.

57 THORNTON, C.; HUTTER, F.; HOOS, H. H.; LEYTON-BROWN, K. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In: *ACM SIGKDD*. Association for Computing Machinery, 2013. Part F128815, p. 847–855. ISBN 9781450321747. Disponível em: <https://dl.acm.org/doi/10.1145/2487575.2487629>.

58 OLSON, R. S.; EDU, O.; MOORE, J. H. *TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning*. PMLR, 2016. 66–74 p. Disponível em: <https://proceedings.mlr.press/v64/olson_tpot_2016.html>.

59 DRORI, I.; KRISHNAMURTHY, Y.; RAMPIN, R.; DE, R.; LOURENCO, P.; ONO, J. P.; CHO, K.; SILVA, C.; FREIRE, J. AlphaD3M: Machine Learning Pipeline Synthesis. In: *ICML AutoML Workshop*. [S.l.: s.n.], 2018.

60 FEURER, M.; KLEIN, A.; EGGENSPERGER, K.; SPRINGENBERG, J. T.; BLUM, M.; HUTTER, F. Auto-sklearn: Efficient and Robust Automated Machine Learning. In: . Springer, Cham, 2019. p. 113–134. Disponível em: <http://link.springer.com/10.1007/978-3-030-05318-5_6>.

61 VILALTA, R.; DRISSI, Y. A Perspective View and Survey of Meta-Learning. *Artificial Intelligence Review*, v. 18, p. 77–95, 2002.

62 SOUTO, M. D.; PRUDÊNCIO, R.; SOARES, R.; ARAUJO, D. D.; COSTA, I.; LUDERMIR, T.; SCHLIEP, A. Ranking and selecting clustering algorithms using a meta-learning approach. In: *IJCNN*. [S.l.: s.n.], 2008. p. 3729–3735. ISBN 9781424418213.

63 FERRARI, D.; CASTRO, L. D. Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods. *Information Sciences*, v. 301, p. 181–194, 2015.

64 LEMKE, C.; GABRYS, B. Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, v. 73, n. 10-12, p. 2006–2016, 2010.

65 PRUDÊNCIO, R. B.; LUDERMIR, T. B. Meta-learning approaches to selecting time series models. *Neurocomputing*, Elsevier, v. 61, n. 1-4, p. 121–137, 10 2004. ISSN 0925-2312.

66 SOARES, C.; BRAZDIL, P.; KUBA, P. A meta-learning method to select the kernel width in support vector regression. *Machine Learning*, v. 54, n. 3, p. 195–209, 2004.

67 FEURER, M.; SPRINGENBERG, J.; HUTTER, F. Initializing Bayesian hyperparameter optimization via meta-learning. In: *National Conference on Artificial Intelligence*. [S.l.: s.n.], 2015. v. 2, p. 1128–1135. ISBN 9781577357001.

68 SONG, Q.; WANG, G.; WANG, C. Automatic recommendation of classification algorithms based on data set characteristics. *Pattern Recognition*, v. 45, n. 7, p. 2672–2689, 7 2012. ISSN 00313203.

69 REIF, M.; SHAFAIT, F.; GOLDSTEIN, M.; BREUEL, T.; DENGEL, A. Automatic classifier selection for non-experts. *Pattern Analysis and Applications*, Springer, v. 17, n. 1, p. 83–96, 2 2014. ISSN 14337541. Disponível em: <https://link-springer-com.ez16.periodicos.capes.gov.br/article/10.1007/s10044-012-0280-z>.

70 ZHANG, Z.; WU, Z.; ZHANG, H.; WANG, J. Meta-Learning-Based Deep Reinforcement Learning for Multiobjective Optimization Problems. *IEEE Trans. on Neural Networks and Learning Systems*, p. 1–14, 2022.

71 XIA, J.-Y.; LI, S.; HUANG, J.-J.; YANG, Z.; JAIMOUKHA, I. M.; GÜNDÜZ, D. Metalearning-Based Alternating Minimization Algorithm for Nonconvex Optimization. *IEEE Trans. on Neural Networks and Learning Systems*, p. 1–15, 2022.

72 PFAHRINGER, B.; BENSUSAN, H.; GIRAUD-CARRIER, C. G. Meta-Learning by Landmarking Various Learning Algorithms. In: *7th ICML*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. (ICML '00), p. 743–750. ISBN 1558607072.

73 HO, T. K.; BASU, M. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 3, p. 289–300, 2002. ISSN 01628828.

74 CASTIELLO, C.; CASTELLANO, G.; FANELLI, A. M. Meta-data: Characterization of input features for meta-learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, v. 3558 LNAI, p. 457–468, 2005. ISSN 16113349. Disponível em: <https://link.springer.com/chapter/10.1007/11526018_45>.

75 PIMENTEL, B. A.; CARVALHO, A. C. de. A new data characterization for selecting clustering algorithms using meta-learning. *Information Sciences*, Elsevier, v. 477, p. 203–219, 3 2019. ISSN 0020-0255.

76 LORENA, A. C.; MACIEL, A. I.; MIRANDA, P. B. de; COSTA, I. G.; PRUDÊNCIO, R. B. Data complexity meta-features for regression problems. *Machine Learning*, Springer New York LLC, v. 107, n. 1, p. 209–246, 1 2018. ISSN 15730565.

77 RIVOLLI, A.; GARCIA, L. P. F.; SOARES, C.; VANSCHOREN, J.; CARVALHO, A. C. P L F de; FRIAS, R. R. Characterizing classification datasets: a study of meta-features for meta-learning. 8 2018. Disponível em: <https://arxiv.org/abs/1808.10406v2>.

78 ALCOBAÇA, E.; SIQUEIRA, F.; RIVOLLI, A.; GARCIA, L. P. F.; OLIVA, J. T.; CARVALHO, A. C. P. L. F. D. MFE: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research*, v. 21, p. 1–5, 2020. Disponível em: <https://pypi.org/project/pymfe/>.

79 RIVOLLI, A.; GARCIA, L. P.; SOARES, C.; VANSCHOREN, J.; CARVALHO, A. C. de. Meta-features for meta-learning. *Knowledge-Based Systems*, Elsevier, v. 240, p. 108101, 3 2022. ISSN 0950-7051.

80 BRAZDIL, P.; RIJN, J. N. van; SOARES, C.; VANSCHOREN, J. Dataset Characteristics (Metafeatures). *Cognitive Technologies*, Springer Science and Business Media Deutschland GmbH, p. 53–75, 2022. ISSN 21976635. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-030-67024-5_4>.

81 RENDELL, L.; SESHU, R.; TCHENG, D. More robust concept learning using dynamically-variable bias. In: LANGLEY, P. (Ed.). *Proceedings of the Fourth International Workshop on MACHINE LEARNING*. Elsevier, 1987. p. 66–78. ISBN 978-0-934613-41-5. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/B9780934613415500118>.

82 AHA, D. W. Generalizing from Case Studies: A Case Study. In: *Machine Learning Proceedings 1992*. [S.l.]: Elsevier, 1992. p. 1–10.

83 BRAZDIL, P.; GAMA, J.; HENERY, B. *Characterizing the applicability of classification algorithms using meta-level learning*. [S.l.: s.n.], 1994. v. 784 LNCS. 83–102 p. ISBN 9783540578680.

84 MICHIE, D.; FULKERSON, B.; SPIEGELHALTER, D. J.; TAYLOR, C. C. *Machine Learning, Neural and Statistical Classification*. New York: Ellis Horwood, 1994. ISSN 00401706. ISBN 013106360X. Disponível em: <http://www1.maths.leeds.ac.uk/~charles/statlog/whole.pdf>.

85 KALOUSIS, A. *Algorithm Selection via Meta-Learning*. Tese (Doutorado) — Université de Genève, 2002.

86 BENSUSAN, H. *God doesn't always shave with occam's razor – learning when and how to prune*. [S.l.: s.n.], 1998. v. 1398. 119–124 p. ISBN 9783540644170.

87 PENG, Y.; FLACH, P.; SOARES, C.; BRAZDIL, P. *Improved dataset characterisation for meta-learning*. [S.l.: s.n.], 2002. v. 2534. 141–152 p. ISBN 9783540001881.

88 BENSUSAN, H.; GIRAUD-CARRIER, C. *Discovering task neighbourhoods through landmark learning performances*. [S.l.: s.n.], 2000. v. 1910. 325–330 p. ISBN 9783540410669.

89 RENDELL, L.; SESHU, R. Learning Hard Concepts through Constructive Induction: Framework and Rationale. *Comput. Intell.*, Blackwell Publishers, Inc., USA, v. 6, n. 4, p. 247–270, 1 1991. ISSN 0824-7935. Disponível em: <https://doi.org/10.1111/j.1467-8640.1990.tb00298.x>.

90 PÉREZ, E.; RENDELL, L. A. Learning despite Concept Variation by Finding Structure in Attribute-Based Data. In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996. (ICML'96), p. 391–399. ISBN 1558604197.

91 CALIÑSKI, T.; HARABASZ, J. A Dendrite Method For Cluster Analysis. *Communications in Statistics*, v. 3, n. 1, 1974. ISSN 00903272.

92 KALOUSIS, A.; THEOHARIS, T. NOEMON: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis*, v. 3, n. 5, p. 319–337, 1999.

93 BRAZDIL, P. B.; SOARES, C.; COSTA, J. P. da. Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning*, Springer Netherlands, v. 50, n. 3, p. 251–277, 2003. ISSN 08856125.

94 GIRAUD-CARRIER, C. The data mining advisor: Meta-learning at the service of practitioners. In: *ICMLA*. [S.l.: s.n.], 2005. v. 2005, p. 113–119. ISBN 9780769524955.

95 ZHANG, X.; LI, R.; ZHANG, B.; YANG, Y.; GUO, J.; JI, X. An instance-based learning recommendation algorithm of imbalance handling methods. *Applied Mathematics and Computation*, v. 351, p. 204–218, 2019.

96 GAROUANI, M.; AHMAD, A.; BOUNEFFA, M.; HAMLICH, M. Autoencoder-kNN meta-model based data characterization approach for an automated selection of AI algorithms. *Journal of Big Data*, v. 10, n. 1, 2023.

97 ALI, S.; SMITH, K. A. On learning algorithm selection for classification. *Applied Soft Computing*, Elsevier, v. 6, n. 2, p. 119–138, 1 2006. ISSN 1568-4946.

98 KALOUSIS, A.; HILARIO, M. Model Selection via Meta-learning: A Comparative Study. *International Journal on Artificial Intelligence Tools*, World Scientific Pub Co Pte Lt, v. 10, n. 04, p. 525–554, 12 2001. ISSN 0218-2130. Disponível em: <www.worldscientific.com>.

99 BRAZDIL, P.; CARRIER, C. G.; SOARES, C.; VILALTA, R. *Metalearning: Applications to data mining*. [S.l.]: Springer Science & Business Media, 2008.

100 WANG, G.; SONG, Q.; SUN, H.; ZHANG, X.; XU, B.; ZHOU, Y. A feature subset selection algorithm automatic recommendation method. *Journal of Artificial Intelligence Research*, AI Access Foundation, v. 47, p. 1–34, 2013. ISSN 10769757.

101 GARCIA, L. P.; LORENA, A. C.; MATWIN, S.; CARVALHO, A. C. de. Ensembles of label noise filters: a ranking approach. *Data Mining and Knowledge Discovery*, Springer New York LLC, v. 30, n. 5, p. 1192–1216, 9 2016. ISSN 1573756X. Disponível em: <https://link.springer.com/article/10.1007/s10618-016-0475-9>.

102 SUN, Q.; PFAHRINGER, B. Pairwise meta-rules for better meta-learning-based algorithm ranking. *Machine Learning*, v. 93, n. 1, p. 141–161, 10 2013. ISSN 08856125.

103 PINTO, F.; CERQUEIRA, V.; SOARES, C.; MENDES-MOREIRA, J. Autobagging: Learning to rank bagging workows with metalearning. *CEUR Workshop Proceedings*, CEUR-WS, v. 1998, 2017. ISSN 16130073.

104 HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. H.; FRIEDMAN, J. H. *The elements of statistical learning: data mining, inference, and prediction*. [S.l.]: Springer, 2009. v. 2.

105 ALCALÁ-FDEZ, J.; FERNÁNDEZ, A.; LUENGO, J.; DERRAC, J.; GARCÍA, S.; SÁNCHEZ, L.; HERRERA, F. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, v. 17, n. 2-3, p. 255–287, 2011. ISSN 15423980.

106 MACIÀ, N.; HO, T. K.; ORRIOLS-PUIG, A.; BERNADÓ-MANSILLA, E. The Landscape Contest at ICPR 2010. In: ÜNAY, D.; ÇATALTEPE, Z.; AKSOY, S. (Ed.). *Recognizing Patterns in Signals, Speech, Images and Videos*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 29–45. ISBN 978-3-642-17711-8.

107 MACIÀ, N.; ORRIOLS-PUIG, A.; BERNADÓ-MANSILLA, E. In search of targeted-complexity problems. *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10*, p. 1055–1062, 2010.

108 CORTES, C.; VAPNIK, V. Support-Vector Networks. *Machine Learning*, v. 20, n. 3, p. 273–297, 1995. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-34249753618&doi=10.1023%2FA%3A1022627411411&partnerID=40&md5=97a8591c7d55575e8c48344379ee2796>.

109 FREUND, Y.; SCHAPIRE, R. E. Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, v. 37, n. 3, p. 277–296, 1999. ISSN 08856125.

110 RASMUSSEN, C. E.; WILLIAMS, C. K. I. *Gaussian processes for machine learning*. [S.l.]: MIT Press, 2006. 248 p. ISBN 026218253X.

111 HINTON, G. E. Connectionist learning procedures. In: *Machine learning*. [S.l.]: Elsevier, 1990. p. 555–610.

112 SATO, A.; YAMADA, K. Generalized Learning Vector Quantization. *Proceedings of the 8th International Conference on Neural Information Processing Systems*, p. 423–429, 1996.

113 FERNANDEZ-DELGADO, M.; CERNADAS, E.; BARRO, S.; AMORIM, D. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research*, v. 15, p. 3133–3181, 10 2014. ISSN 1532-4435.

114 KAUFMAN, S.; ROSSET, S.; PERLICH, C. Leakage in data mining: Formulation, detection, and avoidance. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 556–563, 2011. Disponível em: <https://dl.acm.org/doi/10.1145/2020408.2020496>.

115 BARELLA, V. H.; GARCIA, L. P.; SOUTO, M. C. de; LORENA, A. C.; CARVALHO, A. C. de. Assessing the data complexity of imbalanced datasets. *Information Sciences*, Elsevier Inc., v. 553, p. 83–109, 2021. ISSN 00200255. Disponível em: <https://doi.org/10.1016/j.ins.2020.12.006>.

116 BHATT, H.; MEHTA, S.; D'MELLO, L. R. Use of ID 3 Decision Tree Algorithm for Placement Prediction. In: . [S.l.: s.n.], 2015.

117 POLYAKOVA, M. V.; KRYLOV, V. N. Data normalization methods to improve the quality of classification in the breast cancer diagnostic system. *Applied Aspects of Information Technology*, v. 5, n. 1, p. 55–63, 4 2022. ISSN 26174316.

118 GARCÍA, S.; ZHANG, Z. L.; ALTALHI, A.; ALSHOMRANI, S.; HERRERA, F. Dynamic ensemble selection for multi-class imbalanced datasets. *Information Sciences*, Elsevier, v. 445-446, p. 22–37, 6 2018. ISSN 0020-0255.

119 BARELLA, V. H.; GARCIA, L. P.; SOUTO, M. P. D.; LORENA, A. C.; CARVALHO, A. D. Data Complexity Measures for Imbalanced Classification Tasks. *Proceedings of the International Joint Conference on Neural Networks*, v. 2018-July, 2018.

120 BENAVOLI, A.; CORANI, G.; MANGILI, F. Should We Really Use Post-Hoc Tests Based on Mean-Ranks? *Journal of Machine Learning Research*, v. 17, p. 1–10, 2016.

121 LORENA, A. C.; GARCIA, L. P.; LEHMANN, J.; SOUTO, M. C.; HO, T. K. How complex is your classification problem?: A survey on measuring classification complexity. *ACM Computing Surveys*, v. 52, n. 5, 2019. ISSN 15577341.

122 LEV, J. The Point Biserial Coefficient of Correlation. *The Annals of Mathematical Statistics*, v. 20, n. 1, 1949. ISSN 0003-4851.

123 TANG, J.; ALELYANI, S.; LIU, H. Feature selection for classification: A review. *Data classification: Algorithms and applications*, CRC press, p. 37, 2014.

124 HAN, S.; ZHU, K.; ZHOU, M. C.; ALHUMADE, H.; ABUSORRAH, A. Locating Multiple Equivalent Feature Subsets in Feature Selection for Imbalanced Classification. *IEEE Transactions on Knowledge and Data Engineering*, IEEE Computer Society, v. 35, n. 9, p. 9195–9209, 9 2023. ISSN 15582191.

125 AMORIM, L. B. de; CAVALCANTI, G. D.; CRUZ, R. M. Meta-Scaler: A Meta-Learning Framework for the Selection of Scaling Techniques. *IEEE Transactions on Neural Networks and Learning Systems*, Institute of Electrical and Electronics Engineers Inc., 2024. ISSN 21622388.

126 ARTHUR, D.; VASSILVITSKII, S. k-means++: the advantages of careful seeding. In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. USA: Society for Industrial and Applied Mathematics, 2007. (SODA '07), p. 1027–1035. ISBN 9780898716245.

APPENDIX A – META-FEATURES DESCRIPTION TABLES

Table 14 – Meta-features present in the meta-dataset (part 1).

#	Meta-feature	Type	Origin	Description
1	ch	clustering	PyMFE	Calinski and Harabasz index.
2	int	clustering	PyMFE	INT index.
3	nre	clustering	PyMFE	Normalized relative entropy.
4	pb	clustering	PyMFE	Pearson correlation between class matching and instance distances.
5	sc	clustering	PyMFE	Number of clusters with size smaller than a given size.
6	sil	clustering	PyMFE	Mean silhouette value.
7	vdb	clustering	PyMFE	Davies and Bouldin Index.
8	vdu	clustering	PyMFE	Dunn Index.
9	c1	complexity	PyMFE	Entropy of class proportions.
10	c2	complexity	PyMFE	Imbalance ratio.
11	cls_coef	complexity	PyMFE	Clustering coefficient.
12	density	complexity	PyMFE	Average density of the network.
13	f1.mean	complexity	PyMFE	Maximum Fisher's discriminant ratio.
14	f1v.mean	complexity	PyMFE	Directional-vector maximum Fisher's discriminant ratio.
15	f2.mean	complexity	PyMFE	Volume of the overlapping region.
16	f3.mean	complexity	PyMFE	Compute feature maximum individual efficiency.
17	f4.mean	complexity	PyMFE	Collective feature efficiency.
18	hubs.mean	complexity	PyMFE	Hub score.
19	l1.mean	complexity	PyMFE	Sum of error distance by linear programming.
20	l2.mean	complexity	PyMFE	OVO subsets error rate of linear classifier.
21	l3.mean	complexity	PyMFE	Non-Linearity of a linear classifier.
22	lsc	complexity	PyMFE	Local set average cardinality.
23	n1	complexity	PyMFE	Fraction of borderline points.
24	n2.mean	complexity	PyMFE	Ratio of intra and extra class nearest neighbor distance.
25	n3.mean	complexity	PyMFE	Error rate of the nearest neighbor classifier.
26	n4.mean	complexity	PyMFE	Non-linearity of the k-NN Classifier.
27	t1.mean	complexity	PyMFE	Fraction of hyperspheres covering data.
28	t2	complexity	PyMFE	Average number of features per dimension.
29	t3	complexity	PyMFE	Average number of PCA dimensions per points.
30	t4	complexity	PyMFE	Ratio of the PCA dimension to the original dimension.
31	cohesiveness .mean	concept	PyMFE	Improved version of the weighted distance, that captures how dense or sparse is the example distribution.
32	conceptvar .mean	concept	PyMFE	Concept variation that estimates the variability of class labels among examples.
33	impconceptvar .mean	concept	PyMFE	Improved concept variation that estimates the variability of class labels among examples.
34	wg_dist.mean	concept	PyMFE	Weighted distance, that captures how dense or sparse is the example distribution.
35	attr_to_inst	general	PyMFE	Ratio between the number of attributes and instances.
36	cat_to_num	general	PyMFE	Ratio between the number of categoric and numeric features.
37	freq_class.mean	general	PyMFE	Relative frequency of each distinct class.
38	inst_to_attr	general	PyMFE	Ratio between the number of instances and attributes.
39	nr_attr	general	PyMFE	Total number of attributes.
40	nr_bin	general	PyMFE	Number of binary attributes.
41	nr_cat	general	PyMFE	Number of categorical attributes.
42	nr_class	general	PyMFE	Number of distinct classes.
43	nr_inst	general	PyMFE	Number of instances (rows) in the dataset.
44	nr_num	general	PyMFE	Number of numeric features.
45	attr_conc.mean	info-theory	PyMFE	Compute concentration coef. of each pair of distinct attributes.
46	attr_ent.mean	info-theory	PyMFE	Compute Shannon's entropy for each predictive attribute.
47	class_conc.mean	info-theory	PyMFE	Compute concentration coefficient between each attribute and class.
48	class_ent	info-theory	PyMFE	Compute target attribute Shannon's entropy.
49	eq_num_attr	info-theory	PyMFE	Number of attributes equivalent for a predictive task.
50	joint_ent.mean	info-theory	PyMFE	Joint entropy between each attribute and class.
51	mut_inf.mean	info-theory	PyMFE	Mutual information between each attribute and target.
52	ns_ratio	info-theory	PyMFE	Noisiness of attributes.
53	one_itemset.mean	itemset	PyMFE	One itemset meta-feature.
54	two_itemset.mean	itemset	PyMFE	Two itemset meta-feature.
55	best_node.mean	landmarking	PyMFE	Performance of a the best single decision tree node.

Table 15 – Meta-features present in the meta-dataset (part 2).

#	Meta-feature	Type	Origin	Description
56	best_node .mean.relative	landmarking	PyMFE	Performance of a the best single decision tree node.
57	elite_nn.mean	landmarking	PyMFE	Performance of Elite Nearest Neighbor.
58	elite_nn .mean.relative	landmarking	PyMFE	Performance of Elite Nearest Neighbor.
59	linear_discr.mean	landmarking	PyMFE	Performance of the Linear Discriminant classifier.
60	linear_discr .mean.relative	landmarking	PyMFE	Performance of the Linear Discriminant classifier.
61	naive_bayes.mean	landmarking	PyMFE	Performance of the Naive Bayes classifier.
62	naive_bayes .mean.relative	landmarking	PyMFE	Performance of the Naive Bayes classifier.
63	one_nn.mean	landmarking	PyMFE	Performance of the 1-Nearest Neighbor classifier.
64	one_nn.mean .relative	landmarking	PyMFE	Performance of the 1-Nearest Neighbor classifier.
65	random_node .mean	landmarking	PyMFE	Performance of the single decision tree node model induced by a random attribute.
66	random_node .mean.relative	landmarking	PyMFE	Performance of the single decision tree node model induced by a random attribute.
67	worst_node.mean	landmarking	PyMFE	Performance of the single decision tree node model induced by the worst informative attribute.
68	leaves	model-based	PyMFE	Number of leaf nodes in the DT model.
69	leaves_branch.mean	model-based	PyMFE	Size of branches in the DT model.
70	leaves_corrob.mean	model-based	PyMFE	Leaves corroboration of the DT model.
71	leaves_homo.mean	model-based	PyMFE	DT model Homogeneity for every leaf node.
72	leaves_per_class .mean	model-based	PyMFE	Proportion of leaves per class in DT model.
73	nodes	model-based	PyMFE	Number of non-leaf nodes in DT model.
74	nodes_per_attr	model-based	PyMFE	Ratio of nodes per number of attributes in DT model.
75	nodes_per_inst	model-based	PyMFE	Ratio of non-leaf nodes per number of instances in DT model.
76	nodes_per_level .mean	model-based	PyMFE	Ratio of number of nodes per tree level in DT model.
77	nodes_repeated .mean	model-based	PyMFE	Number of repeated nodes in DT model.
78	tree_depth.mean	model-based	PyMFE	Depth of every node in the DT model.
79	tree_imbalance .mean	model-based	PyMFE	Tree imbalance for each leaf node.
80	tree_shape.mean	model-based	PyMFE	Tree shape for every leaf node.
81	var_importance .mean	model-based	PyMFE	Features importance of the DT model for each attribute.
82	can_cor.mean	statistical	PyMFE	Compute canonical correlations of data.
83	cor.mean	statistical	PyMFE	Absolute value of the correlation of distinct dataset column pairs.
84	cov.mean	statistical	PyMFE	Absolute value of the covariance of distinct dataset attribute pairs.
85	eigenvalues.mean	statistical	PyMFE	Eigenvalues of covariance matrix from dataset.
86	g_mean.mean	statistical	PyMFE	Geometric mean of each attribute.
87	gravity	statistical	PyMFE	Distance between minority and majority classes center of mass.
88	h_mean.mean	statistical	PyMFE	Harmonic mean of each attribute.
89	iq_range.mean	statistical	PyMFE	Interquartile range (IQR) of each attribute.
90	kurtosis.mean	statistical	PyMFE	Kurtosis of each attribute.
91	lh_trace	statistical	PyMFE	Lawley-Hotelling trace.
92	mad.mean	statistical	PyMFE	Median Absolute Deviation (MAD) adjusted by a factor.
93	max.mean	statistical	PyMFE	Maximum value from each attribute.
94	mean.mean	statistical	PyMFE	Mean value of each attribute.
95	median.mean	statistical	PyMFE	Median value from each attribute.
96	min.mean	statistical	PyMFE	Minimum value from each attribute.
97	nr_cor_attr	statistical	PyMFE	Number of distinct highly correlated pair of attributes.
98	nr_disc	statistical	PyMFE	Number of canonical correlation between each attribute and class.
99	nr_norm	statistical	PyMFE	Number of attributes normally distributed based in a given method.

Table 16 – Meta-features present in the meta-dataset (part 3).

#	Meta-feature	Type	Origin	Description
100	nr_outliers	statistical	PyMFE	Number of attributes with at least one outlier value.
101	p_trace	statistical	PyMFE	Pillai's trace.
102	range.mean	statistical	PyMFE	Range (max - min) of each attribute.
103	roy_root	statistical	PyMFE	Roy's largest root.
104	sd.mean	statistical	PyMFE	Standard deviation of each attribute.
105	skewness.mean	statistical	PyMFE	Compute a statistical test for homogeneity of covariances.
106	sparsity.mean	statistical	PyMFE	Skewness for each attribute.
107	t_mean.mean	statistical	PyMFE	Compute (possibly normalized) sparsity metric for each attribute.
108	var.mean	statistical	PyMFE	Trimmed mean of each attribute.
109	w_lambda	statistical	PyMFE	Variance of each attribute.
110	worst_node .mean.relative	landmarking	PyMFE	Wilks' Lambda value.
111	F2_partial.0	overlapping	ImbCoL	Overlapping of the per-class bounding boxes (Class 0).
112	F2_partial.1	overlapping	ImbCoL	Overlapping of the per-class bounding boxes (Class 1).
113	F3_partial.0	overlapping	ImbCoL	Maximum individual feature efficiency (Class 0).
114	F3_partial.1	overlapping	ImbCoL	Maximum individual feature efficiency (Class 1).
115	F4_partial.0	overlapping	ImbCoL	Collective feature efficiency (Class 0).
116	F4_partial.1	overlapping	ImbCoL	Collective feature efficiency (Class 1).
117	N1_partial.0	neighborhood	ImbCoL	Fraction of points lying on the class boundary (Class 0).
118	N1_partial.1	neighborhood	ImbCoL	Fraction of points lying on the class boundary (Class 1).
119	N2_partial.0	neighborhood	ImbCoL	Average intra/inter class nearest neighbor distances (Class 0).
120	N2_partial.1	neighborhood	ImbCoL	Average intra/inter class nearest neighbor distances (Class 1).
121	N3_partial.0	neighborhood	ImbCoL	Leave-one-out error rate of the 1-NN algorithm (Class 0).
122	N3_partial.1	neighborhood	ImbCoL	Leave-one-out error rate of the 1-NN algorithm (Class 1).
123	N4_partial.0	neighborhood	ImbCoL	Nonlinearity of the 1-NN classifier (Class 0).
124	N4_partial.1	neighborhood	ImbCoL	Nonlinearity of the 1-NN classifier (Class 1).
125	T1_partial.0	neighborhood	ImbCoL	Fraction of maximum covering spheres on data (Class 0).
126	T1_partial.1	neighborhood	ImbCoL	Fraction of maximum covering spheres on data (Class 1).
127	L1_partial.0	linearity	ImbCoL	Distance of erroneous instances to a linear classifier (Class 0).
128	L1_partial.1	linearity	ImbCoL	Distance of erroneous instances to a linear classifier (Class 1).
129	L2_partial.0	linearity	ImbCoL	Training error of a linear classifier (Class 0).
130	L2_partial.1	linearity	ImbCoL	Training error of a linear classifier (Class 1).
131	L3_partial.0	linearity	ImbCoL	Nonlinearity of a linear classifier (Class 0).
132	L3_partial.1	linearity	ImbCoL	Nonlinearity of a linear classifier (Class 1).