An Instance Space Analysis of Regression Problems

MARIO ANDRÉS MUÑOZ and TAO YAN, The University of Melbourne
MATHEUS R. LEAL, Universidade Federal de Minas Gerais
KATE SMITH-MILES, The University of Melbourne
ANA CAROLINA LORENA, Instituto Tecnológico de Aeronáutica
GISELE L. PAPPA, Universidade Federal de Minas Gerais
RÔMULO MADUREIRA RODRIGUES, Instituto Tecnológico de Aeronáutica

The quest for greater insights into algorithm strengths and weaknesses, as revealed when studying algorithm performance on large collections of test problems, is supported by interactive visual analytics tools. A recent advance is Instance Space Analysis, which presents a visualization of the space occupied by the test datasets, and the performance of algorithms across the instance space. The strengths and weaknesses of algorithms can be visually assessed, and the adequacy of the test datasets can be scrutinized through visual analytics. This article presents the first Instance Space Analysis of regression problems in Machine Learning, considering the performance of 14 popular algorithms on 4,855 test datasets from a variety of sources. The two-dimensional instance space is defined by measurable characteristics of regression problems, selected from over 26 candidate features. It enables the similarities and differences between test instances to be visualized, along with the predictive performance of regression algorithms across the entire instance space. The purpose of creating this framework for visual analysis of an instance space is twofold: one may assess the capability and suitability of various regression techniques; meanwhile the bias, diversity, and level of difficulty of the regression problems popularly used by the community can be visually revealed. This article shows the applicability of the created regression instance space to provide insights into the strengths and weaknesses of regression algorithms, and the opportunities to diversify the benchmark test instances to support greater insights.

CCS Concepts: • **Mathematics of computing** → **Regression analysis**; • **Human-centered computing** → **Visual analytics**; • **Theory of computation** → **Design and analysis of algorithms**; • **Computing methodologies** → **Supervised learning by regression**; • **General and reference** → *Empirical studies*; *Experimentation*; *Performance*;

Additional Key Words and Phrases: Algorithm selection, instance spaces, machine learning, regression, visual analytics

Funding was provided by the Australian Research Council, the Conselho Nacional de Desenvolvimento Científico e Tecnológico, and the Fundação de Amparo à Pesquisa do Estado de São Paulo, under grants FL140100012, 305291/2017-3, 2012/22608-8 and 2019/20328-7.

Authors' addresses: M. A. Muñoz, T. Yan, and K. Smith-Miles, School of Mathematics and Statistics, The University of Melbourne, Parkville, Victoria 3010, Australia; emails: munoz@unimelb.edu.au, taoyanmicro@hotmail.com, smith-miles@unimelb.edu.au; M. R. Leal and G. L. Pappa, Computer Science Department, Universidade Federal de Minas Gerais, Av Antonio Carlos, 6627-Pampulha, 31270-901, Belo Horizonte, MG, Brazil; emails: matheusrleal22@gmail.com, glpappa@dcc.ufmg.br; A. C. Lorena and R. Madureira Rodrigues, Divisão de Ciência da Computação, Instituto Tecnológico de Aeronáutica, Praça Marechal , Eduardo Gomes, 50, 12228-900, São José dos Campos - SP - Brazil; emails: aclorena@ita.br, romulomadu@gmail.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.

1556-4681/2021/03-ART28 $15.00
https://doi.org/10.1145/3436893

ACM Reference format:
Mario Andrés Muñoz, Tao Yan, Matheus R. Leal, Kate Smith-Miles, Ana Carolina Lorena, Gisele L. Pappa, and Rômulo Madureira Rodrigues. 2021. An Instance Space Analysis of Regression Problems. *ACM Trans. Knowl. Discov. Data* 15, 2, Article 28 (March 2021), 25 pages.
https://doi.org/10.1145/3436893

1 INTRODUCTION

Regression problems seek to learn the mathematical relationship between dependent variable responses (outputs) and a set of independent variables (inputs). They are an important class of Machine Learning (ML) problems, and are often a critical precursor to decision-making that first requires prediction of likely responses to decisions before the most likely optimal decision can be determined. The accuracy of the algorithm used to solve the regression problem and make a prediction of responses is therefore a crucial component of the decision-making process. With increasing pressure for algorithmic transparency and accountability of decisions, it is important to develop methodologies and tools to scrutinize the performance of algorithms, and their suitability and reliability for the task at hand.

There are many available methods for solving regression problems, but which one is likely to be best suited to different datasets is poorly understood. Obviously if the mathematical relationships between independent and dependent variables are linear, then many regression methods are likely to be accurate and reliable. But for more challenging datasets, the dependence of algorithm performance on distinguishing characteristics of the data must be understood in order to justify the choice of one algorithm over another to support decision-making. Furthermore, the choice of test datasets from which such insights are sought is critical, and we must ensure that the datasets are suitable—unbiased, diverse, challenging, and discriminating—to support trust in algorithmic transparency and accountability.

In recent years, there has been a growing interest in meta-analysis of the performance achieved by diverse techniques in solving different ML problems [32]. By studying large collections of algorithm performance results on diverse test datasets, whose mathematical and statistical features can be measured, the aim is to build ML models to predict how algorithm performance depends on dataset characteristics [29]. With such a meta-learning approach, one may highlight strengths and weaknesses of each algorithm and obtain insights about situations in which they can be successfully employed [15]. This may aid new users in the choice of suitable techniques for solving their particular problems. Moreover, it also allows one to identify potential gaps that can be explored in order to develop new ML approaches and techniques [25].

Extending the ideas of meta-learning in the direction of interactive visual analytics, a recent methodology known as Instance Space Analysis (ISA) has been developed by Smith-Miles and coworkers [11, 19, 26]. ISA offers a framework to support visual insights into algorithm performance that is twofold in its aims: to provide visual insights and analytics into the strengths and weaknesses of algorithms across the broadest possible space of test instances; and to provide a visual analysis of the diversity, difficulty and discrimination capabilities of benchmark problem instances, and their adequacy for building trust in conclusions about algorithm reliability. The visualization of the instance space provides a perspective that is often hidden by tables of computational results reporting summary statistics about algorithm performance averaged across all test instances. The ISA methodology was originally proposed in [26] and demonstrated on combinatorial optimization problems, but has since then been generalized to other algorithmic science domains such as time series forecasting [11] and continuous black-box optimization [18]. In the field of ML, it has recently been employed in the analysis of classification problems [19]. Using a diverse set of classification problems from the UCI [5], OpenML [31], and KEEL [1] repositories, this work also attempted to enrich the instance space by the generation of new datasets spanning a wider range of problem complexity. Within the same framework, it is possible to reveal the unique capabilities of different classification algorithms, by visualizing and objectively measuring the area of their footprints—regions of predicted good performance—in the instance space.

An online interactive visualization tool to support ISA has been developed by Smith-Miles, Muñoz, and co-workers, and is known as Melbourne Algorithm Test Instance Library with Data Analytics (MATILDA) [27], whose MATLAB computational engine is freely available at GitHub [20]. This article presents the first ISA of regression problems using MATILDA. Despite being very related to classification problems, regression problems are far less investigated by the ML community, with far fewer meta-analyses of regression problems in the ML literature. In [30], the authors employ a meta-learning approach to select the kernel width for Support Vector Regressors (SVR). There, 14 meta-features previously used in [12] are employed to describe 16 regression problems and a nearest neighbor classifier using this dataset is able to rank the best kernel width values for new problems. This work was extended in [7] by the combined use of meta-learning and search techniques in the same problem, whereby the SVR parameter values suggested by meta-learning were used as initial seed points by a search technique. The pool of regression problems was extended to 40 datasets. In an extensive experimental study, Amasyali and Ersoy [2] investigated meta-regression using a set of 181 datasets and around 300 meta-features. As expected, the most successful algorithm varies for distinct dataset collections. Despite the large set of meta-features used by these authors, a large subset of them was very correlated to each other by definition. Lorena et al. [14] presented a set of meta-features able to characterize the complexity of regression problems, and assessed performance on three meta-learning tasks: prediction of regression function type, prediction of SVR parameter values, and prediction of the expected performance of various regression techniques. The study in [6] presents an experimental comparison of 164 algorithms in the solution of 52 non-linear regression problems. They also attempt to characterize the strength of each algorithm with respect to the others. Another recent development worth noting is the Item Response Theory (IRT) analysis of ML problems [16], inspired by educational psychology studies of student performance on test questions. IRT in ML is an explanatory model which can be used to relate the performance of a set of algorithms (like students) to the discriminatory and difficulty level of a pool of datasets (like test questions) [3]. Despite being potentially relevant, the IRT framework was not yet been applied in meta-analysis of regression problems in the literature.

In this article, we advance upon previous literature to provide a novel meta-analysis of regression problems and techniques in ML, which allows a visual analysis of the distribution of the problems according to their characteristics, and subsets of problems for which each algorithm is expected to perform better. Using the ISA framework, we will visualize for the first time the location of common benchmark test problems in a two-dimensional ($2D$) space, examining their features and identifying the regions in the space occupied by the easy and hard instances. We will overlay the performance of algorithms across the instance space to gain insights into how various methods are challenged by different properties of test instances. We will use ML methods to learn to predict which regression method is recommended for different regions of the instance space. Finally, we will analyze the sufficiency of the existing benchmark instances to support additional insights for future studies of regression instance spaces.

The remainder of this article is organized as follows: Section 2 presents the ISA framework. The methods employed in this article are described in Section 3. Section 4 presents some experiments and discusses the achieved results, while Section 5 concludes the article and discussed future avenues for research.

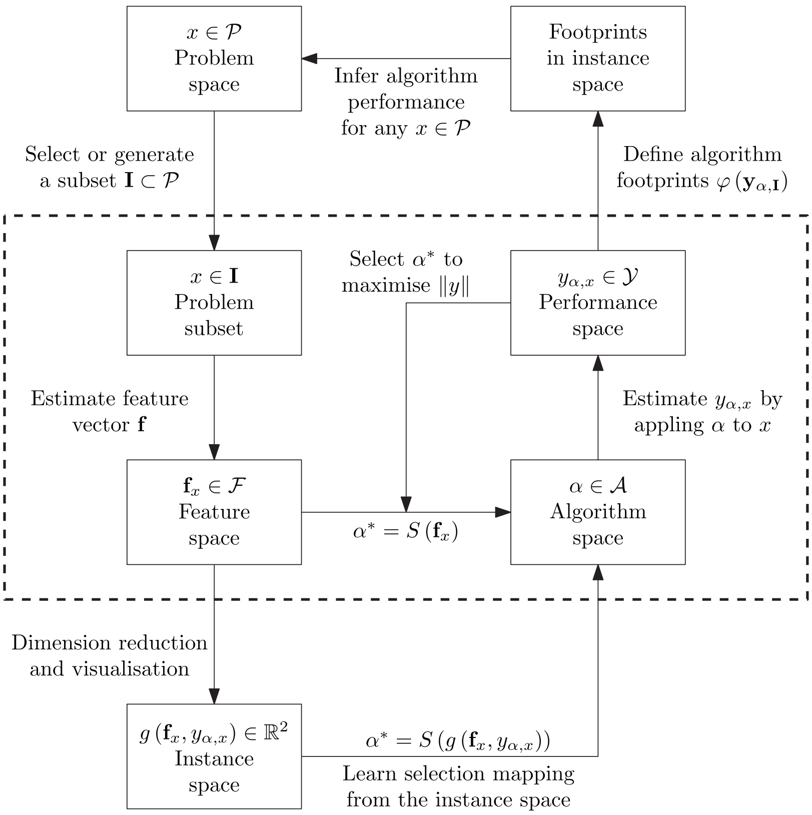

$x \in \mathcal{P}$
Problem space
Infer algorithm performance for any $x \in \mathcal{P}$
Footprints in instance space
Select or generate a subset $\mathbf{I} \subset \mathcal{P}$
Define algorithm footprints $\varphi(\mathbf{y}_{\alpha,\mathbf{I}})$
$x \in \mathbf{I}$
Problem subset
Select α^* to maximise $\|y\|$
$y_{\alpha,x} \in \mathcal{Y}$
Performance space
Estimate feature vector $\mathbf{f}$
Estimate $y_{\alpha,x}$ by appling α to x
$\mathbf{f}_x \in \mathcal{F}$
Feature space
$\alpha^* = S(\mathbf{f}_x)$
$\alpha \in \mathcal{A}$
Algorithm space
Dimension reduction and visualisation
$g(\mathbf{f}_x, y_{\alpha,x}) \in \mathbb{R}^2$
Instance space
$\alpha^* = S(g(\mathbf{f}_x, y_{\alpha,x}))$
Learn selection mapping from the instance space

Fig. 1. Summary of the ISA methodology proposed in [26], underpinned by the Algorithm Selection framework (in the dotted box) in [22].

2 THE INSTANCE SPACE ANALYSIS FRAMEWORK

The Algorithm Selection Problem (ASP) was stated by Rice [22] as finding a function able to map the characteristics of a given instance of a problem to the performance of algorithms in such a way that the algorithm that is predicted to achieve maximum performance for a given problem can be identified among a pool of candidate algorithms. For such a mapping to be learned, four spaces or sets are required [19, 29]:

— Sub-space of instances ($\mathbf{I}$): composed of instances of the problem class.
— Feature space ($\mathcal{F}$): contains characteristics used to describe properties of the instances as a summary feature vector.
— Algorithm space ($\mathcal{A}$): containing the set of algorithms which are candidate for solving the problem instances.
— Performance space ($\mathcal{Y}$): one or more metrics measuring the performance achieved by the algorithms in solving the problem instances.

These spaces are shown in the center of Figure 1, which summarizes the framework for the generation of an instance space [19]. This framework was built upon Rice's ASP original formulation and adds some other sets and steps, which provides a visualization of the meta-analysis to reveal strengths and weaknesses of the candidate algorithms, and also the representativeness of the selected subset of instances.

Given a problem space ($\mathcal{P}$) containing all possible instances of a problem class, a subset of instances is selected (or generated) to compose **I**. Here $\mathcal{P}$ is the class of regression problems, while **I** contains a selection of regression datasets from public repositories. The feature space $\mathcal{F}$ should gather a diverse set of measurements representing characteristics of the dataset, which may affect the performance of an algorithm. These features are also named meta-features in the meta-learning literature [9]. For instance, sparse datasets with a large number of attributes and a low number of data points tend to be harder to analyze due to the probable presence of underrepresented regions in the input space, and a regression problem can also be complex if it is described by non-informative attributes [14]. Developing meta-features requires significant domain knowledge. However, good meta-features should be: cheaper to compute than fitting a model or equivalent to fitting a trivial model; be uncorrelated with other features and highly correlated with performance of at least one algorithm; be agnostic of the true solution of the problem. The meta-features for regression used in this article are divided into the following categories [2, 12, 14]:

Simple measures: basic characteristics, such as the size of the datasets, their number of features, among others.
Statistical measures: measures of localization, dispersion, distribution, and correlation of variables.
Information theoretic measures: measures of the information content of the variables.
Landmarking features: considers the predictive performance of simple baseline regressors on the datasets.
Complexity measures: captures the intrinsic difficulty in solving the regression problem.

Under the ISA framework, the algorithm space $\mathcal{A}$ should also be composed of a diverse set of alternative methods which can be used to solve the given instances. There are various regression techniques, each with their own biases for solving regression problems in both the statistics and ML areas [6, 23]. They can be generically divided into parametric vs. non-parametric models, linear vs. non-linear models, among other categories. The more algorithms with different biases are included in the analysis, the greater the chances of finding the best solution to a new problem instance. It is also of interest to identify which algorithms are essentially obtaining very similar results across the instance space, and may well be mechanistically rather similar, despite being uniquely named.

The performance space $\mathcal{Y}$ records information about the performance of the algorithms in $\mathcal{A}$ when solving problem instances in **I**. In regression problems, there are various alternatives for measuring the predictive performance of the generated models. Common alternatives are the Mean Squared Error of the predictions compared to the known data outputs, the Mean Absolute Error, and also normalized versions which try to resolve scale issues.

Finding the mapping $S(\cdot)$ from a problem instance described by a set of meta-features into one or more algorithms which will have a high expected performance can be solved as a learning problem itself. In that case, $S(\cdot)$ is also called a meta-learner [9, 32]. This has been the principle of meta-learning studies [32], which started in the late 1990s. The idea is to use meta-knowledge of previous problems with known solution to solve new learning problems. For instance, by gathering the results of multiple models in several benchmark regression problem instances, one may be able to induce predictive models able to indicate the best or a set of better methods for a new regression dataset.

Beyond performance prediction, a meta-dataset relating the characteristics of multiple datasets to the performances of a pool of diverse regressors provides the opportunity to visualize these relationships via a $2D$ projection of the feature space known as an instance space [19]. The ISA methodology allows visualization of datasets as points in the instance space, and identification of pockets of hard or easy datasets, or even gaps that should ideally be fulfilled by generation or curation of new datasets. It is also possible to identify regions of the instance space where an algorithm can be considered more competent when compared to others, with this region of expected (predicted based on ML methods) good performance defined as the algorithm's footprint. Such knowledge can support the selection of the most suitable algorithms for new problem instances, after inferring performance across the broader problem space $\mathcal{P}$.

3 METHODOLOGY

To generate the regression instance space and use its visual analytics capability for insights, five key steps are recommended [19]:

(1) Building a meta-dataset containing a set of problem instances $\mathbf{I}$, described by the meta-features in $\mathcal{F}$ and labeled according to the performance of the algorithms in $\mathcal{A}$, as assessed by a chosen performance metric $\mathcal{Y}$. It is also necessary to define good performance according to this metric;
(2) Filtering a subset of meta-features which can be viewed as more relevant to describe the problem instances and their ability to explain variations in algorithm performance;
(3) Generation of the $2D$ instance space by projecting the instances from the high-dimensional feature space to a $2D$ coordinate system using a dimension reduction method;
(4) Identification and characterization of the algorithm "footprints," i.e., the areas of the instance space in which performance is expected to be good, given the empirical evidence presented to a ML algorithm; and
(5) Construct an automated algorithm selector that uses existing knowledge to predict which algorithm may be suitable and best recommended for a new instance.

Earlier work on the ISA methodology [11, 26] relied on Principal Component Analysis as the dimension reduction method for step 3, but in [19] a more powerful projection algorithm was proposed that seeks to maximize the linear trends observed when considering the distribution of meta-feature values and algorithm performance metrics across the instance space. This projection algorithm is derived by solving a high-dimensional global optimisation problem numerically to arrive at an optimal linear transformation that makes the resulting $2D$ instance space more interpretable. We adopt this projection approach in this article, which is summarized in Section 3.3, and refer the reader to [19] for details.

As a final step in the ISA methodology, the resulting $2D$ instance space can then be analyzed for insights into the relative strengths and weaknesses of algorithms; predictions can be made using ML methods about how algorithms are expected to perform in parts of the space where they have not yet been tested; and the sufficiency of the existing benchmarks can be analyzed to see if they are diverse, challenging and discriminating enough, or if there are gaps in the instance space where new test instances should be created. This article will restrict to the description of the methodology used to build the regression instance space and discussing the insights offered for the particular meta-dataset considered in the present study. As a proof of concept to show the potential of ISA for regression, we leave tasks such as the generation of new test instances to future work. In the following sections, we describe how the enumerated steps were tackled in this work for our chosen experimental setting.

3.1 Building the meta-dataset

3.1.1 Problem instances. Four sources of regression datasets were adopted:

Repositories: 246 datasets were collected from the KEEL [1], OpenML [31], and UCI Machine Learning [5] public repositories.

BlackBoxData: 2,547 datasets are randomly selected instances from the Comparing COntinuous Optimisation (COCO) benchmark set [8]. Commonly used to test numerical optimisation algorithms such as BFGS, Nelder-Mead, and CMA-ES, this benchmark set has 24 basis functions which can be scaled to arbitrary dimensions and transformed through translations and rotations, and symmetry breaking through oscillations about the identity. The COCO software uniquely identifies each instance using an index to allow replicable experiments. To generate regression datasets, we reused the sample data originally prepared for [17], which corresponds to: (1) Latin Hyper-cube samples of dimensions $D = \{2, 3, 5, 8, 10, 20, 40, 60, 100\}$ and with $10^3 \times D$ observations as the independent variables; and (2) the responses for two randomly selected instances from indexes $[1, \ldots, 30]$ for each basis functions as the dependent variables. With some datasets being quite large, we standardized their number of observations into five sizes $\{50, 100, 250, 500, 1000, 2000\}$ by randomly choosing observations. Of these 2,592, 45 were discarded due to erroneous performance data from most algorithms.

EvolvedBlackBox: 1,763 datasets correspond to instances generated for testing black-box optimisation algorithms [17]. These 2- and 10-dimensional functions were generated by targeting the instance space of black-box optimization problems, using Genetic Programming. As the COCO benchmark functions, we generated a regression dataset through a Latin Hyper-cube sample and collected the function responses.

M3C: 299 datasets correspond to time series problems from the M3-Competition [13]. To generate a regression dataset, we used the auto-regressive method, in which previous values of the series are used to predict the next. The number of the features for the data is 10% of the length of the original time series with a rounding and features are generated from the number of the lagging from time series. For example, if time series has length of 20, the features for the synthetic data should be two and those two features are lagged by one and two, respectively, from original time series. This results in instances whose observations are not independent and identically distributed (iid); hence, they may favor algorithms which do not assume iid data. Therefore, we have used a 10% subset of the available datasets.

This curation process results in a total of 4,855 regression datasets, which compose the set **I** in our study. Let n designate the number of examples or observations within a base regression dataset and m the number of input features or attributes (independent variables). The selected datasets have $n \in [13,\ 2400]$ and $m \in [1,\ 108]$. Most of the datasets (about 90%) have only quantitative attributes. The remaining have mixed types of attributes (both quantitative and qualitative). There are some regression techniques and also some meta-features that are suitable for quantitative attributes only. In such cases, two strategies were employed: disregarding the qualitative attributes; or converting them to multiple numerical values using a binarization approach. The meta-features extracted from the previous datasets are those defined and used in the articles [2, 14, 30] and are described next.

3.1.2 Algorithms. The regression algorithms considered, composing the set $\mathcal{A}$, were: Adaboost, Bagging, Bayesian ARD, Decision Tree, ϵ-SVR, linear SVR, ν-SVR, Extra tree, Gradient Boosting, Ridge Kernel regression, MLP Neural Network, Passive aggressive, Random Forest, and Stochastic Gradient Descent (SGD), making a total of 14 regression techniques. All of them are used as available in the scikit-learn package [21]. They belong to different families of algorithms and present distinct biases. According to the categorization presented in [6], we have representatives from the following families of algorithms: Neural Networks (MLP); Support Vector Machines (SVMs) (ϵ-SVR, linear SVR and ν-SVR); Regression Trees (Decision Tree and Extra Tree); Random Forests;

Table 1. Algorithms Under Study Sorted by Their Percentage of Good Instances, P_g, with the Percentage of Instances for which an Algorithm is Uniquely Good, P_{ug}, and the Recalculated Percentages of Good, P_s, and Uniquely Good, P_{us}, Instances for the Algorithms Retained for Further Analysis

		P_g	P_{ug}	P_s	P_{us}
1	Gradient Boosting	41.6%	25.4%	44.7%	28.0%
2	Bayesian ARD	29.0%	21.3%	29.7%	21.9%
3	Bagging	22.3%	1.2%	23.5%	1.3%
4	Random Forest	22.2%	1.3%	23.3%	1.3%
5	ϵ-SVR	15.6%	0.3%		
6	ν-SVR	15.1%	0.9%	16.4%	2.9%
7	Linear SVR	11.3%	0.9%	12.6%	2.2%
8	Adaboost	11.1%	0.9%	12.2%	1.0%
9	Decision Tree	7.8%	3.4%	8.8%	4.2%
10	MLP	6.9%	0.0%		
11	Kernel Ridge	6.7%	1.3%		
12	SGD	6.4%	0.1%		
13	Passive Aggressive	6.2%	0.6%		
14	Extra Tree	4.1%	1.4%		

Generalized Linear Models (Passive aggressive and SGD); Bagging; Boosting (Adaboost, Gradient Boosting); and others (Bayesian ARD and Ridge Kernel regression).

3.1.3 Performance metric. The predictive performance of the techniques was evaluated using a five-fold cross-validation (CV) strategy with the Normalized Mean Absolute Error (NMAE) metric, defined as [30]:

$$\text{NMAE} = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} |y_i - \bar{y}|}, \tag{1}$$

where $\hat{y}_i$ represents a prediction for the i-th example in the dataset and $\bar{y}$ is the mean of the target values. Being a normalized measure of performance, NMAE allows us to fairly compare the performance of any algorithm on multiple datasets. Moreover, it gives equal relevance to small and large errors, unlike the Normalized Mean Squared Error which emphasizes larger errors [10].

We use a definition of good performance that states an algorithm's performance on an instance is good if it is within 5% of the best (lowest) NMAE compared to all other algorithms in the portfolio. This definition of good performance will be needed in the construction of the instance space. Of course, we could also consider a definition of good that only considers the best NMAE, or an absolute measure of a NMAE below a given threshold, as we have done in previous studies [19, 26]. Using this definition of good, 59.0% of the instances have one good algorithm, 18.8% two, 9.5% three, 3.8% four, and 2.9% five. There are no instances for which all algorithms are good. Table 1 shows the algorithms sorted by their percentage of good instances, P_g, with the percentage of instances for which an algorithm is uniquely good, P_{ug}. From the top ten algorithms by P_g, we focus on those with a $P_{ug} > 0.5\%$. Table 1 also shows the recalculated percentages of good, P_s, and uniquely good, P_{us}, instances.

Naturally, if the definition of good is changed, then the results are also affected (different features may be selected, and so the axes of the instance space coordinate system will also be changed),

Table 2. Meta-features Employed in the Analysis

Abbreviation	Name	Category
$n1$	Number of examples in the dataset [30]	Simple
$n2$	The number of features in the dataset [30]	Simple
$n5$	Proportion of attributes with outliers [30]	Simple
$n7$	Sparsity of the target [30]	Statistical
$n8$	Outliers in the target [30]	Simple
$n9$	Stationary of the target [30]	Statistical
$c1$	Maximum feature correlation to the output [14, 30]	Statistical
$c2$	Average feature correlation to the output [14, 30]	Statistical
$c3$	Individual feature efficiency [14]	Complexity
$c4$	Collective feature efficiency [14]	Complexity
$c5$	Average correlation between features [2]	Statistical
$f2$	Average feature F-test to the output [2]	Statistical
$l1\_a$	Mean absolute error of OLS without symbolic attributes [14, 30]	Landmarking
$l1\_b$	Mean absolute error of OLS [14, 30]	Landmarking
$l2\_a$	Mean squared error of OLS without symbolic attributes [14, 30]	Landmarking
$l2\_b$	Mean squared error of OLS [14, 30]	Landmarking
$m2$	Average mutual information to the output [30]	Information
$m5$	Average mutual information among features [30]	Information
$r2\_a$	R^2 from a linear model without symbolic attributes [30]	Landmarking
$r2\_b$	R^2 from a linear model [30]	Landmarking
$s1$	Normalized output distribution [14]	Complexity
$s2$	Normalized input distribution [14]	Complexity
$s3$	Error of the nearest neighbor regressor [14]	Landmarking
$s4$	Non-linearity of nearest neighbor regressor [14]	Complexity
$t2$	Ratio between the number of examples and the number of attributes [14, 30]	Simple
$t3$	Proportion of symbolic features [2, 30]	Simple

but this article aims to show how the instance space can be constructed and analyzed for a given experimental setting that includes decisions such as definitions of good performance, choice of instances, meta-features, algorithms and their parameters. The methodology can be repeated for other experimental settings to explore the insights that are offered for a given set of experimental choices beyond the usual summary table of average results. The meta-datasets for this study are available for download from MATILDA's website[1] to support further exploration of different experimental settings.

3.1.4 *Meta-features.*

A set of 26 meta-features were employed to describe the datasets. They are presented in Table 2, separated according to the category they belong to, namely simple, statistical, information theoretic, landmarking and complexity-based. Despite the usage of over 300 meta-features in [2], a careful examination reveals that many of them capture very similar aspects and are highly correlated. Therefore, we opted to include meta-features from [2, 14, 30] that could extract more distinct aspects from the benchmarks and could reveal the difficulty level of the regression problems more clearly.

[1]Available at: https://matilda.unimelb.edu.au/matilda/problems/learning/regression.

Table 3. Top 10 Meta-features Per Algorithm Sorted by Their Absolute Correlation with Performance

	Adaboost	Bagging	Bayesian ARD	Decision Tree	Gradient Boosting	Linear SVR	ν-SVR	Random Forest
1	***n1***	***n1***	***n1***	***n1***	***n1***	***n1***	***n1***	***n1***
	0.5458	0.6783	0.4027	0.4991	0.7554	0.8054	0.9124	0.6372
2	**t2**	***t2***	**c5**	**t2**	***t2***	***c5***	***c5***	***t2***
	0.4542	0.5969	0.3246	0.4660	0.6684	0.7198	0.7390	0.5610
3	**c5**	**c5**	*t2*	**c5**	***c5***	***t2***	***t2***	**c5**
	0.4182	0.4875	0.2901	0.3722	0.5650	0.6695	0.6612	0.4625
4	**m5**	**s1**	*m5*	*s1*	**s1**	***m5***	***m5***	**s1**
	0.3142	0.3630	0.2509	0.2736	0.4057	0.5895	0.5783	0.3368
5	*s1*	**m5**	*l1_a*	*m5*	**m5**	c2	**c2**	**m5**
	0.2319	0.3512	0.1644	0.2671	0.4056	0.2875	0.3278	0.3350
6	*c4*	*n2*	*l1_b*	*n2*	*n2*	*s1*	*s1*	*n2*
	0.1328	0.1790	0.1623	0.1743	0.1947	0.2777	0.2898	0.1677
7	*c3*	*c4*	*l2_a*	*c4*	*c1*	*r2_b*	*c3*	*c4*
	0.1107	0.1488	0.1601	0.1665	0.1624	0.2674	0.2764	0.1471
8	*n2*	*s2*	*l2_b*	*s2*	*c4*	*r2_a*	*r2_b*	*c1*
	0.0929	0.1241	0.1568	0.1230	0.1399	0.2625	0.2504	0.1159
9	*c2*	*c1*	*s1*	*c3*	*s2*	*c3*	*r2_a*	*s2*
	0.0837	0.1179	0.1507	0.0976	0.1246	0.2625	0.2443	0.1140
10	*m2*	*s4*	*r2_a*	*c1*	*s4*	*c1*	*c1*	*c3*
	0.0739	0.0656	0.1275	0.0962	0.0960	0.2602	0.1988	0.0576

In boldface/italics are those meta-features with high correlation ($0.5 \leq |\rho_{x,y}|$), while in boldface are those with moderate correlation ($0.3 \leq |\rho_{x,y}| < 0.5$). By selecting the top five meta-features, all of them would have at least moderate correlation with one algorithm, with the only exception of $l1\_a$.

3.2 Filtering the meta-features

Pre-processing of the meta-data is required in the construction of an instance space, and is fully automated by the online tool MATILDA. Initially, each meta-feature is bounded between its median plus or minus five times its interquartile range to reduce the effect of outliers, as these statistics are robust estimators of the typical value and the spread, and encapsulate over 99.99% of the data. Next, the meta-features are transformed using the one parameter Box-Cox transformation to stabilise their variance and normalise the data. The transformation is defined by the equation:

$$\mathrm{f}_i^{(\lambda)} = \begin{cases} \frac{\mathrm{f}_i^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0 \\ \ln(\mathrm{f}_i) & \text{if } \lambda = 0 \end{cases} \tag{2}$$

where f_i is the value of the feature for instance i and must be a value greater than zero. As such, before the transformation, the values are shifted by adding one minus the minimum. The value of λ is estimated by maximising the Log-likelihood function. Finally, the tool applies a z-transform to standardise each feature and performance value to mean zero and standard deviation one.

Next, the complete set of meta-features was filtered since many of them may be not correlated enough with algorithmic performance to provide meaningful insights into the hardness of instances and their ability to differentiate between the levels of difficulty for different algorithms. Table 3 shows the top 10 meta-features per algorithm sorted from highest to lowest absolute value of the correlation with performance, $|\rho_{x,y}|$. In boldface/italics are those meta-features with high correlation ($0.5 \leq |\rho_{x,y}|$), while in boldface are those with moderate correlation ($0.3 \leq |\rho_{x,y}| < 0.5$). All top five meta-features for all algorithms have at least moderate correlation with one algorithm, with the only exception of $l1\_a$, which has a low but statistically significant correlation of 0.1644 (p-value of 8.9361×10^{-31}) with Bayesian ARD. We keep $l1\_a$ as it is almost uncorrelated with any other algorithm, and Bayesian ARD does not have many strongly correlated features.

Based on this filtering, the final set of seven meta-features is $\{n1, c2, c5, l1\_a, m5, s1, t2\}$. It is noteworthy the presence of measures from all categories of meta-features in this set, providing support to the idea that multiple characteristics should be considered to properly summarize the algorithms' performance in the data. Measures $n1$ and $t2$ are roughly related to how representative the datasets are when considering their sizes. While all regression algorithms can benefit from problems with a higher number of points $n1$, $t2$ relates this number to the number of attributes and therefore tries to capture the sparsity of the input space. The lower the $t2$ value, the larger tends to be data sparsity. All algorithms may struggle in face of sparse datasets, which tend to contain underrepresented regions. The meta-features $c2$, $c5$, and $m5$ quantify the representativeness of the input attributes. While $c2$ captures the average correlation of the input attributes to the output, $c5$ measures the correlation between the input attributes only, that is, how redundant they are. A high $c2$ value is obtained for datasets with many input features highly correlated to the output, being informative for predicting the output variable. On the other hand, $c5$ assumes higher values if many of the input attributes are redundant among each other, in which case many of them could be removed to make the problem simpler for the regression techniques. $m5$ is related to $c5$, measuring the redundancy of the input attributes, but using mutual information instead of correlation. $l1\_a$ measures whether a linear model might fit the data well. It will be lower for linear or quasi-linear problems, which can be considered simpler than non-linear problems. Finally, $s1$ measures the smoothness of the relationship of similar data points. It first builds a Minimum Spanning Tree (MST) from the input dataset, in which each vertex corresponds to a data point and the edges measure their similarity. Next, $s1$ averages the differences of target values of neighbor examples in the MST. For simpler regression problems, lower values of $s1$ are expected, representing a smooth variation of the target values for similar input data.

Therefore, the final meta-feature set considers different aspects which are known to affect the performance of the regression techniques: the representativeness of the data points as measured by simple measures such as $n1$ and $t2$, related to the datasets sizes; the quality of the input attributes for predicting the target attribute, as measured by $c2$; the redundancy of the input attributes, measured by $c5$ and $m5$; and the linearity and smoothness of the relation of the input attributes to the target output, measured by $l1\_a$ and $s1$. Of course some of those characteristics may affect more one technique over another due to their own biases. For instance, the linearity measure $l1\_a$ will probably be a good descriptor of the behavior of generalized linear models and the linear SVR. But the compromise set of meta-features found can be considered general enough to quantify different aspects that influence the difficulty of a regression problem.

3.3 Constructing the Instance Space

In order to visualize instances and algorithmic performance we project from the $7D$ meta-feature space to a $2D$ plane. While any dimensionality reduction method can be suitable for this task, here we use the Prediction Based Linear Dimensionality Reduction (PBLDR) method developed in our previous work [19]. For reference, we present the details of PBLDR in this section. Let $\mathbf{F} \in \mathbb{R}^{n \times m}$ be a matrix containing m features for n instances and $\mathbf{Y} \in \mathbf{R}^{n \times a}$ a matrix containing the performance measure of a algorithms on n instances. An ideal projection of the instances for this group of algorithms is achieved by finding the matrices $\mathbf{A}_r \in \mathbb{R}^{2 \times m}$, $\mathbf{B}_r \in \mathbb{R}^{m \times 2}$ and $\mathbf{C}_r \in \mathbb{R}^{a \times 2}$

which minimise the approximation error:

$$\|\mathbf{F} - \widehat{\mathbf{F}}\|_F^2 + \|\mathbf{Y} - \widehat{\mathbf{Y}}\|_F^2, \tag{3}$$

such that:

$$\mathbf{Z} = \mathbf{A}_r \mathbf{F}, \tag{4}$$

$$\widehat{\mathbf{F}} = \mathbf{B}_r \mathbf{Z}, \tag{5}$$

$$\widehat{\mathbf{Y}} = \mathbf{C}_r \mathbf{Z}, \tag{6}$$

where $\mathbf{Z} \in \mathbb{R}^{n\times 2}$ is the matrix instance coordinates in the $2D$ space. We assume that $m < n$ and $\mathbf{F}$ is full row rank, i.e., rank $(\mathbf{F}) = m$. If $\mathbf{F}$ is not full dimensional then we consider the problem in a subspace spanned by $\mathbf{F}$. Thus, we have the following optimisation problem:

$$\begin{array}{lll} & \min & \|\mathbf{F} - \mathbf{B}_r\mathbf{Z}\|_F^2 + \|\mathbf{Y} - \mathbf{C}_r\mathbf{Z}\|_F^2 \\ & \text{s.t.} & \mathbf{Z} = \mathbf{A}_r\mathbf{F} \\ (\mathcal{D}) & & \mathbf{A}_r \in \mathbb{R}^{2\times m} \\ & & \mathbf{B}_r \in \mathbb{R}^{m\times 2} \\ & & \mathbf{C}_r \in \mathbb{R}^{a\times 2}. \end{array} \tag{7}$$

PBLDR uses the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization algorithm to solve numerically $\mathcal{D}$, which is known to be convex but highly ill-conditioned with an infinite number of solutions falling within a line. BFGS always finds a global optimum for $\mathcal{D}$; therefore, the best solution is the one with the highest topological preservation, defined as the correlation between high- and low-dimensional distances, from a number of repeats N_{try}, which is set to 30. Algorithm 1 in Appendix A describes the dimensionality reduction method. The final projection matrix is defined by Equation (8) to represent each dataset as a $2D$ vector $\mathbf{z} = (z_1, z_2)$ depending on its $7D$ meta-feature vector via the following linear transformation:

$$\mathbf{z} = \begin{bmatrix} 0.5655 & 0.5303 \\ -0.1581 & 0.4419 \\ -0.3881 & -0.1159 \\ 0.3180 & -0.2530 \\ -0.4138 & -0.0987 \\ 0.2884 & -0.4519 \\ 0.0920 & 0.2102 \end{bmatrix}^\top \begin{bmatrix} n1 \\ c2 \\ c5 \\ l1\_a \\ m5 \\ s1 \\ t2 \end{bmatrix} \tag{8}$$

3.4 Footprint analysis of algorithm strengths and weaknesses

An algorithm's *footprint* is the area of the instance space where good or best performance is expected based on inference from empirical performance analysis [28] with good performance defined as being within 5% of the best (lowest) NMAE compared to all other algorithms in the portfolio, as described in Section 3.1.3. Besides its area, α, a footprint is characterized by its density, d, defined as the number of instances enclosed by the footprint divided over the area, and its purity, p defined as the percentage of good instances enclosed by the footprint. Carrying out a footprint analysis is a process which can be broadly divided in three steps: (a) estimating the area, α_S, and density, d_S, of the section of the space containing instances, which are the baselines to normalize the results; (b) building and characterizing the footprints for each algorithm; and (c) comparing the footprints to reduce or eliminate contradictions between them. The process has been refined through multiple iterations to reduce parameters, improve repeatability, and stability, with Algorithms 2–4 in Appendix A describing the details of the latest version. To construct a footprint, Algorithm 3 uses DBSCAN to identify high density clusters of good instances. As outputs, DBSCAN provides a vector $\mathbf{c} \in \{-1, 1, \ldots, N_c\}$, where "-1" marks an outlier, and values in $[1, N_c]$ range correspond to the index of the identified clusters. DBSCAN requires two parameters, $\{k, \varepsilon\}$, the former represents the minimal number of neighboring instances that would be considered a cluster, whereas the latter correspond to the neighborhood radius. DBSCAN has been shown to be robust to a variety of parameter values [24], therefore both of them are automatically chosen following the equations [4]:

$$k \leftarrow \max\left(\min\left(\lceil r/20 \rceil, 50\right), 3\right),$$
$$\varepsilon \leftarrow \frac{k\Gamma\left(2\right)}{\sqrt{r\pi}}\left(\mathrm{range}\left(\mathbf{z}_1\right) \times \mathrm{range}\left(\mathbf{z}_2\right)\right),$$

where r is the number of unique instances in the space with good performance, and $\Gamma\left(\cdot\right)$ is the Gamma function. The footprint is then constructed using an α-shape, a generalization of the concept of convex hull from computational geometry, which corresponds to a polygon that tightly encloses all the points within a cloud. An α-shape is constructed for each cluster, and all shapes are bounded together as a MATLAB polygon structure.

Once constructed, contradicting sections can appear when two different conclusions can be drawn from the same area of the instance space due to overlapping footprints, e.g., when comparing two algorithms. These sections are removed using Algorithm 4, where the contradicting section is removed from the footprint with lower purity. The process is repeated $N_{\max} = 3$ times, although more than one try is often unnecessary. If the purity is the same for both footprints, the section is kept, as there is insufficient evidence of dominance of either algorithm.

3.5 Automated algorithm selection in the instance space

We use the MATLAB implementation of SVMs with a polynomial kernel to partition the space into distinct regions where each regression method has dominant performance. Moreover, the SVMs can be used to suggest a method predicted to be good for an untested instance. For each one of the regression algorithms under study, we train an SVM using as input the coordinates Z obtained from PBLDR and as output our criterion of good, i.e., "1" if NMAE is within 5% of the best (lowest) across all algorithms, and "0" otherwise. To tune the SVM parameters $\{C, \gamma\}$, we use 30 iterations of the Bayesian Optimization algorithm bounded between $[10^{-3}, 10^{3}]$, with five-fold CV and the probability of improvement loss function, which is defined as:

$$PI\left(x\right) = \Phi\left(\frac{\mu_Q\left(x_{\mathrm{best}}\right) - m - \mu_Q\left(x\right)}{\sigma_Q\left(x\right)}\right),$$

where x is a new point, x_{best} is the location of the lowest posterior mean, $\mu_Q\left(x_{\mathrm{best}}\right)$ is the lowest value of the posterior mean, $\sigma_Q\left(x\right)$ being the posterior standard deviation at x and $\Phi\left(\cdot\right)$ being the unit normal cumulative distribution function. Once the models are trained, the CV accuracy, precision and recall are collected.

4 RESULTS

Based on the methodology described in the previous section, the experimental results are presented and discussed next. We first present the visualization of the instance space, showing the location of test instances and their source from the benchmark collections considered. We then examine the performance of algorithms across the instance space, and the distribution of meta-features to enable insights into which properties of instances create ease or difficulty for different algorithms. These insights are further supported by the footprint analysis of each algorithm. Finally, we use SVM models to learn to predict the performance of each algorithm across the instance space, and

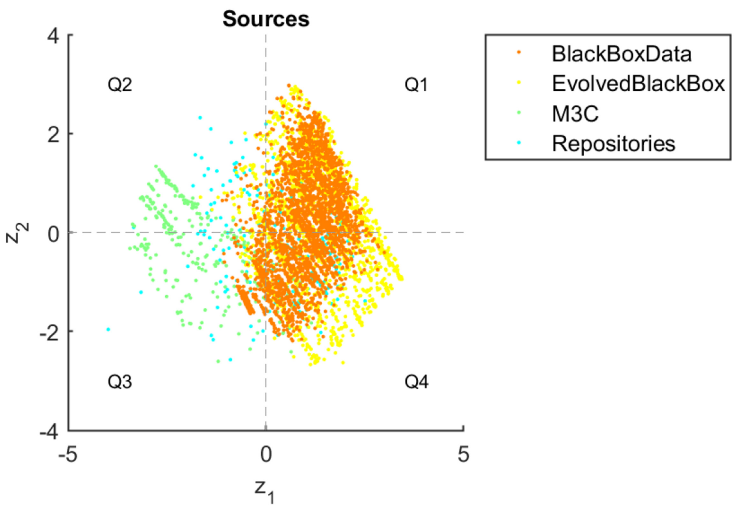


Fig. 2. Regression instance space showing sources of the benchmark problems, and the division into quadrants for discussing the obtained results.

combine them to recommend which algorithm should be used in various regions of the instance space.

4.1 Instance Space structure and location of the source groups

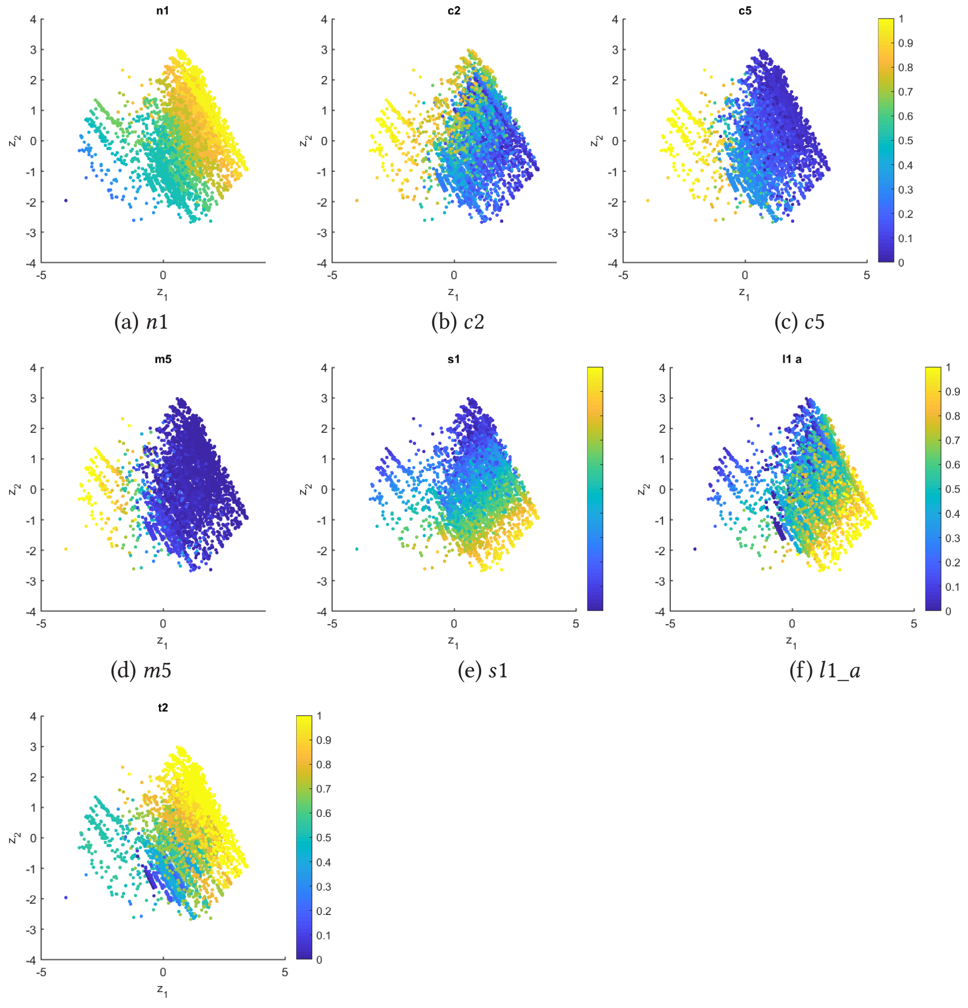
The location of the instances in the instance space is shown in Figure 2, which also indicates the sources of the test instances. We have four types of datasets by source: repositories which are included in the KEEL [1], OpenML [31], and UCI [5] repositories, BlackBoxData, EvolvedBlackBox, and M3C. OpenML and UCI are two popular public repositories used by the ML community. Moreover, Figure 2 shows the instance space divided into four quadrants, anti-clockwise from top-right. According to this division, it is possible to notice that the BlackBoxData and EvolvedBlackBox datasets are concentrated in quadrants Q1 and Q4, while M3C datasets are predominantly situated at quadrants Q2 and Q3. On the other hand, the standard ML repositories are spread in the four quadrants.

The repositories' datasets only span a small area of the instance space; thus if we were relying only on well-studied regression problems in the literature, we would be studying problems that lack diversity. The synthetic datasets not only cover the repositories' area, but also span the instance space further. Nonetheless, there are still under-represented regions of the instance space in all four corners, and opportunities for future work to generate more diverse regression datasets.

4.2 Distribution of the meta-features in the space

Figure 3 shows the distribution of the selected meta-features values across the instance space, from minimal values as blue to maximal values as yellow. Considering the interpretation of the meta-features values, we can observe the following:

- $n1$: the number of examples in the datasets increases from bottom-left (quadrant Q3) to the up-right corner of the instance space, so that larger datasets are concentrated in quadrant Q1.
- $c2$: the average correlation of the input attributes to the target increases from the bottom-right to the top-left corner of the instance space, so that datasets in quadrant Q4 have more



(a) $n1$ (b) $c2$ (c) $c5$

(d) $m5$ (e) $s1$ (f) $l1\_a$

(g) $t2$

Fig. 3. Distribution of meta-features, from minimum (blue) to maximum (yellow) values.

representative features for predicting the target and are easiest according to this meta-feature.

— $c5$: datasets in the top-right direction of the instance space (Q1) tend to present more correlated input features, whilst this correlation is lower for datasets at the left corner (Q2 and Q3); instances at the bottom of the space (quadrant Q4) are considered harder according to this measure.

— $m5$: this measure has a similar interpretation and behavior to that of $c5$, as expected, although is tends to present lower values for most of the datasets.

— $s1$: the datasets at the top of the instance space (quadrants Q1 and Q2) present smoother variations on the outputs for similar data items, which increases towards the bottom-left of the instance space (Q4), which are harder regarding this aspect.

— $l1_a$: there is some relation between the $s1$ and $l1\_a$ values and interpretations in the IS, since linear datasets will tend to present smoother variations on the outputs for similar examples.

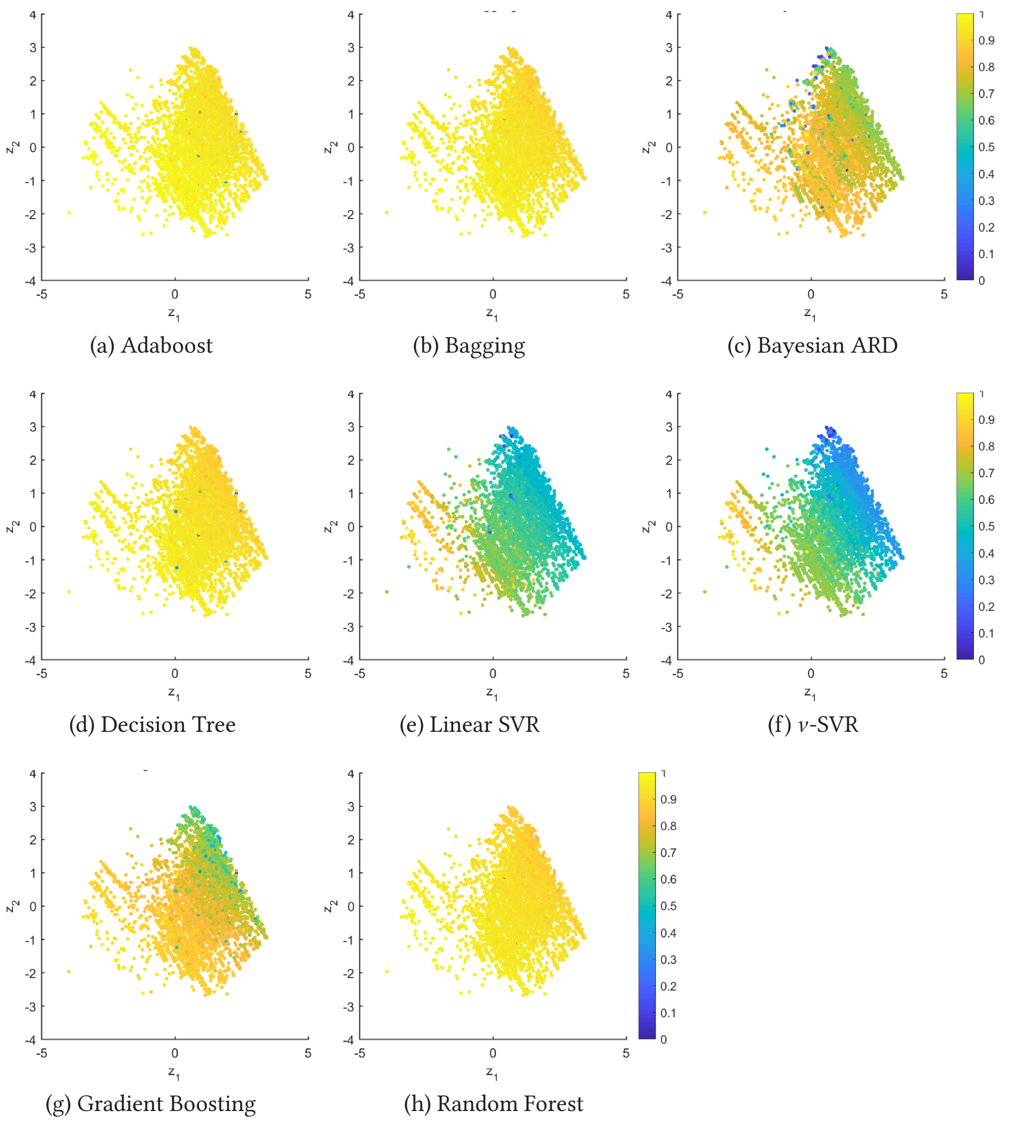


Fig. 4. Distribution of NMAE for each of the six best performing regression algorithms, from minimum (blue) to maximum (yellow) values.

Therefore, linear datasets are concentrated in Q1 and Q2, while datasets for which a non-linear regressor might be required are concentrated in Q4.

— $t2$: the sparsity of the datasets increase from the top-right (Q1) towards the bottom of the instance space (Q3 and Q4). Relating to $n1$, datasets with more examples (larger $n1$) values tend to be less sparse, as expected.

4.3 Distribution of the algorithm performances in the space

Figure 4 shows the distribution of NMAE (from minimal values as blue to maximal values as yellow) for each of the eight algorithms listed in Table 1, which had good performance in a sufficiently large percentage of instances. From this instance space perspective we can see that most of the algorithms find the instances in quadrant 1 (Q1, with positive values for both z_1 and z_2) to be easy, with lower NMAE. Examining where the algorithms are challenged though, a greater variation of performance is observed. Most algorithms consistently find the instances in quadrant 3 (Q3) to be hard. There are some easy instances spread in across the plots too, but largely they are concentrated towards the up-right corner. Combining the results of Figures 3 and 4 enables us to infer the following observations about the strengths and weaknesses of the algorithms for particular characteristics of the instances:

— Instances in Q1, with positive values on both z_1 and z_2 coordinates, are those which most algorithms find easy to solve. From the meta-feature instance space, we can observe that such instances have three obvious characteristics. First, they correspond to larger datasets, with more examples (high $n1$ values). Second, they are less sparse as captured by the higher $t2$ values, and contain more examples than input attributes. Third, the input features are also less redundant among each other, as indicated by low values of $c5$ and $m5$. Intuitively, because an instance with more examples and less correlated attributes carries more useful information, all algorithms tend to find such datasets easier.
— Quadrant Q2 contains datasets of mixed difficulties for the SVR and Bayesian ARD regressors, while they are considered quite hard for tree-based regressors such as Adaboost, Bagging, and Random Forest. The datasets in this quadrant show a smoother variation of outputs for similar instances (low $s1$), a linear trend (low $l1\_a$) and have input features more related to the target values (high $c2$ values). These characteristics may have posed challenges to the tree-based techniques in partitioning the input data space during their processing.
— In Q3 we find instances that most of the algorithms find harder to solve. In contrast to Q1, the instances in this quadrant have a lower number of examples (low $n1$ values) and show a higher sparsity (lower $t2$, the ratio of number of examples to the number of input attributes). The input attributes also tend to be more redundant, as noticed by the larger $c5$ and $m5$ values. For example, the M3C datasets are mostly located in this quadrant. These instances will be hard to solve for most of the algorithms, since they are derived from time series for which the attribute values are generated from original lagged time series (i.e., they were not typically studied as regression benchmarks, but we have added them to this study to explore greater diversity in instance difficulty).
— Finally, quadrant Q4 complements Q2 and contains datasets for which the difference of outputs of similar instances is larger (high $s1$), with a non-linear trend (high $l1\_a$ values) and input features less related to the target output values (low $c2$ values). Most regressors had mixed results for these datasets.

4.4 Footprint analysis

Table 4 presents each algorithm's footprint area (α_N), density (d_N) and purity (p_N), whenever the algorithm is good or best, identified with the subscripts G or B respectively. The results are normalized over the total area (19.7258) and density (242.0185). Moreover, Figure 5 illustrates the footprints of those algorithms with $\alpha_{N,G} \geq 1.0\%$. From both the table and figure, we observe that two algorithms dominate: (a) Bayesian ARD, which covers a large but low density section of Q2 and Q3 ($\alpha_{N,G} = 37.7\%, d_{N,G} = 24.0\%$), where most other algorithms do not perform well, resulting in a higher than average purity ($p_{N,G} = 86.8\%$); and (b) Gradient Boosting, which covers a large but high density section of Q1 and Q4 ($\alpha_{N,G} = 77.3\%, d_{N,G} = 120.9\%$),

Table 4. Footprint Analysis for the Eight Algorithms Under Study, Including Their Area (α_N), Density (d_N) and Purity (p_N), Whenever the Algorithm is Good or Best, Identified with the Subscripts G or B, Respectively

	$\alpha_{N,G}$	$d_{N,G}$	$p_{N,G}$	$\alpha_{N,B}$	$d_{N,B}$	$p_{N,B}$
Adaboost	0.5%	195.1%	62.5%	0.0%	0.0%	0.0%
Bagging	1.0%	108.0%	83.3%	6.0%	109.3%	31.5%
Bayesian ARD	37.7%	24.0%	86.8%	34.9%	26.1%	80.7%
Decision Tree	0.3%	41.5%	100.0%	6.7%	77.7%	41.6%
Gradient Boosting	77.3%	120.9%	47.6%	56.1%	145.4%	41.5%
Linear SVR	1.0%	58.5%	69.0%	1.1%	42.3%	63.6%
ν-SVR	2.0%	25.4%	66.7%	0.8%	50.2%	68.4%
Random Forest	2.7%	99.2%	71.3%	1.8%	48.4%	39.0%
Average	15.3%	84.1%	73.4%	13.4%	62.4%	45.8%

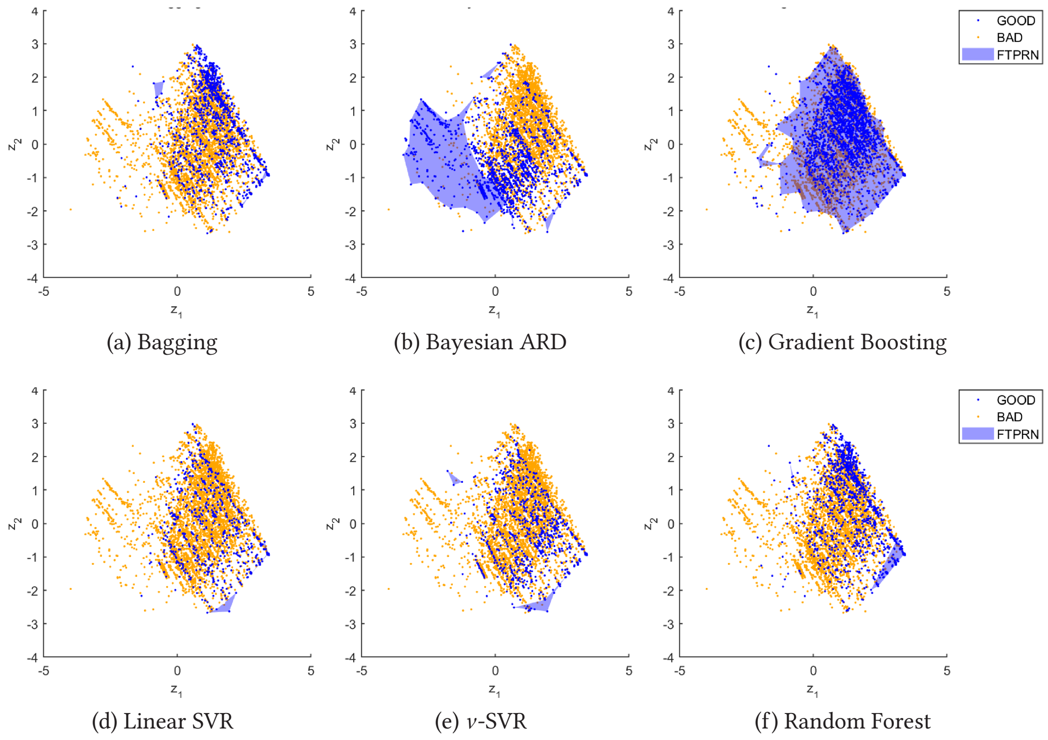


Fig. 5. Footprints of the six algorithms with area, $\alpha_{N,G} \geq 1.0\%$, noting that two algorithms dominate, i.e., (b) Bayesian ARD and (c) Gradient Booster.

where other algorithms perform well, resulting in the lowest purity ($p_{N,G} = 47.6\%$). When compared to the results from Table 1, we observe that these two algorithms have complementary strengths and weaknesses, as they are uniquely good in a significant section of the space. Moreover, while Bagging and Random Forests were the third and fourth ranked algorithms in Table 1, we observe that their good performance is limited to small and dense areas in either Q2 or Q4.

Table 5. Performance of the SVM Prediction Models

	$\overline{NMAE}\ (\sigma)$	$Pr\ (G)$	$\overline{NMAE}_S\ (\sigma_S)$	Accuracy	Precision	Recall	C	γ
Adaboost	0.211(0.481)	0.122	0.012(0.001)	88.2%	73.0%	4.6%	0.001	4.110
Bagging	0.293(2.655)	0.235	0.020(0.079)	78.8%	66.9%	19.5%	0.001	0.293
Bayesian ARD	0.358(2.839)	0.297	0.674(4.509)	76.5%	67.4%	40.9%	916.545	4.332
Decision Tree	0.289(1.539)	0.088	0.046(0.134)	92.5%	71.1%	25.3%	0.001	0.884
Gradient Boosting	0.260(1.630)	0.447	0.113(1.746)	66.1%	61.3%	65.7%	187.771	4.238
Linear SVR	1.615(8.267)	0.126		87.4%		0.0%	0.002	0.280
ν-SVR	0.323(0.977)	0.164	0.013(0.000)	83.9%	69.2%	3.4%	0.001	1.282
Random Forest	0.290(2.752)	0.233	0.037(0.098)	78.0%	66.8%	11.2%	0.002	0.357
Oracle	0.118(0.281)	1.000						
Selector	0.258(2.485)	0.562	0.243(2.699)		63.5%	39.6%		

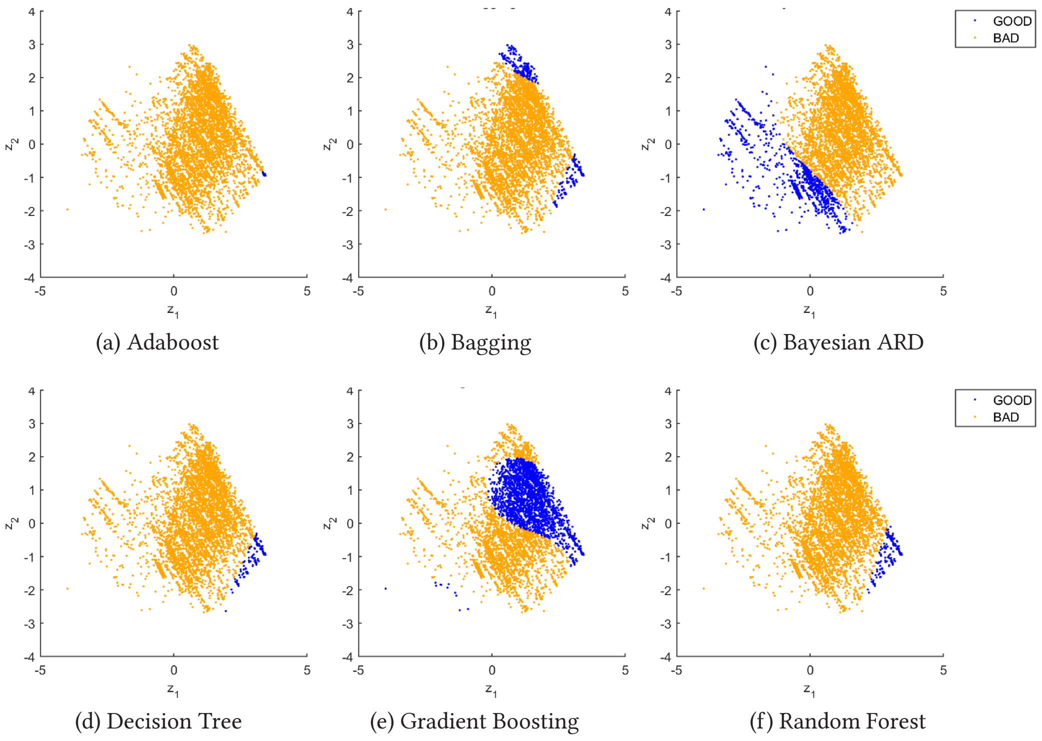
4.5 Prediction of Algorithm Strengths and Weaknesses

While a visual inspection of the strengths and weaknesses of each algorithm in regions of the instance space explained by certain meta-feature values can be conducted, it is also useful to somewhat automate the process of identifying where each algorithm's regions of strength lie. To this end, we use a series of SVMs to learn to predict if each algorithm's performance is expected to be good or bad at any point in the instance space. Table 5 shows the performance metrics for this algorithm selector. As an idealized benchmark, we also present the results of an *oracle* that always selects the right algorithm for a dataset with full knowledge of its performance. For each algorithm we report its average $NMAE$ across all instances with its standard deviation, σ, between parenthesis, its probability of being good, $Pr\ (G)$, as well as the average $NMAE$ across the selected instances $\left(\overline{NMAE}_S\right)$ and its standard deviation, σ_S, between parenthesis. We also present the values of SVMs cross-validated accuracy, precision, and recall, as well as their parameters $\{C, \gamma\}$. The results show that our selector achieves an NMAE of 0.258, better than using any one algorithm all the time. If we consider the selector's success in choosing an algorithm that is expected to have good performance however, we find that it is successful 56.2% of the time ($Pr\ (G)$ = 0.562), making it more reliable that any algorithm alone.

The resulting predictions from each individual SVM are illustrated in Figure 6. It is possible to observe that different methods are expected to perform well in distinct regions of the instance space, but no method is expected to perform well everywhere. For example, the two dominant algorithms, Bayesian ARD, and Gradient Boosting, cover complementary sections of the space confirming the observations from the footprint analysis in the previous section. Figure 7 combines the results of the multiple SVMs into a portfolio of recommendations, taking the output of the SVM predictor with the highest precision at each point. We observe that Bagging ($\alpha_{N,B} = 6.0\%$), Decision Tree ($\alpha_{N,B} = 6.7\%$) and Random Forest ($\alpha_{N,B} = 1.8\%$) have small but unique areas of good performance. On the other hand, the SVRs are not strongly preferred overall. There is a small ambiguous area in the middle, where the selector is not confident that the existing algorithms stand out. For these instances, a recommended course of action would be to use the best performing algorithm, i.e., Gradient Boosting.

5 DISCUSSION AND CONCLUSION

This article presented a first attempt to generalize the ISA framework to the analysis of regression problems in ML. Despite their utility, meta-analysis studies of regression problems are still incipient in the related literature. A large meta-dataset composed of diverse regression problems from



(a) Adaboost (b) Bagging (c) Bayesian ARD

(d) Decision Tree (e) Gradient Boosting (f) Random Forest

Fig. 6. Prediction results for six SVM models trained on each of the regression algorithms.

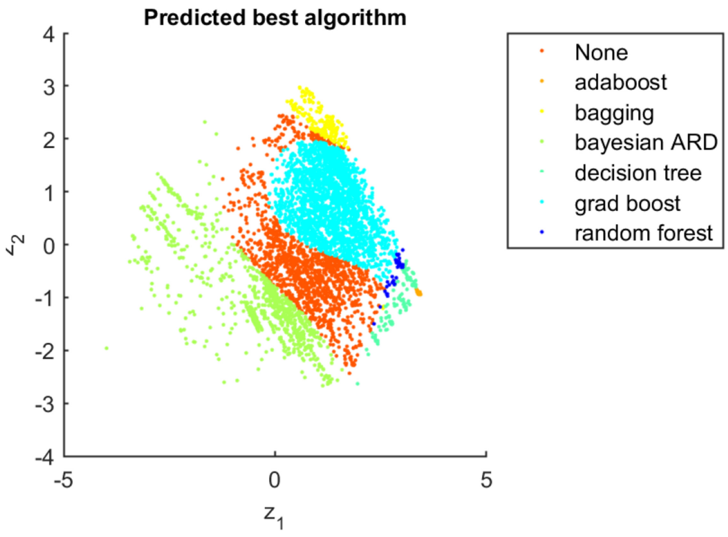


Fig. 7. Combined SVM recommendations.

benchmark repositories, described by meta-features from distinct categories and labeled according to the performance of 14 different regression techniques was built. A careful initial selection of a subset of relevant meta-features was first performed. From there, it was possible to generate a suitable 2D instance space able to offer visual insights into the ability of the different regression techniques and also the representativeness of the problem instances used. We already have strong evidence that quadrant 3 instances (at the bottom left of the instance space—Figure 2) tend to be hard for most of the algorithms, and quadrant 1 instances (at the top-right of the instance space) tend to be easy for most of the algorithms. The datasets in these regions have opposite characteristics concerning the number of examples, sparsity and attribute redundancy. All regressors performed better for datasets with more examples, less sparsity, and with less correlated attributes.

Of course, the insights offered in this article are dependent on the chosen experimental setting—the choice of meta-data and definitions of good performance and other parameter settings. The ISA merely supports a researcher to gain visual insights into the computational results generated by a given experimental setting, and a more comprehensive study would be needed to draw more definitive conclusions about regression problems and algorithms more generally. We leave this for future work, and hope that our sharing of the meta-data and the instance space tools [27] will facilitate such explorations.

Future work will consider performing other analysis within the regression instance space, such as measuring the algorithmic power of different regression techniques and generating new problem instances to expand further across the instance space. Additional meta-features can also be added to the analysis, such as the very recent developments based on IRT for measuring instance hardness in classification problems [16], which should be first generalized for regression problems, rather than classification, and for measuring the instance difficulty at a dataset level.

The regression instance space generated in this study can be viewed and explored interactively as a library problem on MATILDA's website. There, the full interactive functionality of MATILDA can be seen, including the ability to hover the mouse over each instance to view details of its source, features, and algorithm performance results. We encourage other researchers to explore the instance space to extract additional insights in this manner. The meta-data can also be downloaded, included all code for feature calculation, so that additional instances, algorithms, performance metrics and features can be considered in extensions to this work.

APPENDIX

A PSEUDO-CODE FOR THE IMPLEMENTED ALGORITHMS

This appendix presents the pseudo-code for two methods: (a) the PBLDR method; and (b) the Footprint Analysis method; where the former has one procedures described in Algorithm 1, and the latter has three procedures described in Algorithms 2–4.

ALGORITHM 1: Prediction-Based Linear Dimensionality Reduction (PBLDR) method.

```
Input: A matrix $\mathbf{F} \in \mathbb{R}^{n\times m}$ of instance features, a matrix $\mathbf{Y} \in \mathbf{R}^{n\times a}$ of performance measures, and a
       number of tries $N_{\text{try}}$.
Output: A matrix $\mathbf{Z} \in \mathbb{R}^{n\times 2}$ of coordinates in the $2D$ instance space, and a set of projection matrices
        $\{\mathbf{A}_r, \mathbf{B}_r, \mathbf{C}_r\}$.
$\mathbf{D}_H \leftarrow$ EuclideanDistance $(\mathbf{F})$ ;      // Calculate the distances between instances in the feature space
$\mathbf{D}_H \leftarrow$ MatrixAsColumnVector $(\mathbf{D}_H)$;      // Reshape as a column vector
$\rho_{\text{best}} \leftarrow -\infty$;
for $i = 1$ to $N_{try}$ do      // Repeat $N_{\text{try}}$ times
    // Initialize the projection matrices randomly between $[-1, 1]$
    $\mathbf{A}_0 \leftarrow$ UniformRandomMatrix $(2, m, [-1, 1])$;
    $\mathbf{B}_0 \leftarrow$ UniformRandomMatrix $(m, 2, [-1, 1])$;
    $\mathbf{C}_0 \leftarrow$ UniformRandomMatrix $(a, 2, [-1, 1])$;
    $\{\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i\} \leftarrow$ BFGS $(\mathcal{D}, \mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0)$;      // Use BFGS to find a solution to $\mathcal{D}$
    $\mathbf{Z}_i \leftarrow \mathbf{A}_i\mathbf{F}$;
    $\mathbf{D}_L \leftarrow$ EuclideanDistance $(\mathbf{Z}_\mathbf{i})$;
    $\mathbf{D}_L \leftarrow$ MatrixAsColumnVector $(\mathbf{D}_L)$;
    $\rho_i \leftarrow$ PearsonCorrelation $(\mathbf{D}_H, \mathbf{D}_L)$;
    if $\rho_{best} < \rho_i$ then      // If this is the best solution so far
        $\{\mathbf{A}_r, \mathbf{B}_r, \mathbf{C}_r\} \leftarrow \{\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i\}$;
        $\mathbf{Z} \leftarrow \mathbf{Z}_i$;
        $\rho_{\text{best}} \leftarrow \rho_i$;
    end
end
```

ALGORITHM 2: Footprint analysis algorithm.

```
Input: A matrix $\mathbf{Z} \in \mathbb{R}^{n\times 2}$ of coordinates in the $2D$ instance space, a matrix $\mathbf{Y} \in \{0,1\}^{n\times a}$ of binary
       performance measures, and a vector $\mathbf{p} \in \{1,\ldots,a\}^{n\times 1}$ of indexes indicating the best performing
       algorithm for an instance.
Output: Three footprint sets, $\{\boldsymbol{A},\boldsymbol{B},\boldsymbol{\Gamma}\}$, which correspond to good, bad and best performance.
Function FootprintAnalysis($\mathbf{Z},\mathbf{Y},\mathbf{p}$) is
    $S \leftarrow$ BuildFootprint ($\mathbf{Z},\mathbf{1}$);   // Calculate the area, density and purity of the space
    for $i = 1$ to $a$ do                      // Build the three footprints for all algorithms
        $\boldsymbol{A}_i \leftarrow$ BuildFootprint ($\mathbf{Z}$, $\mathbf{y}_i$);
        $\boldsymbol{B}_i \leftarrow$ BuildFootprint ($\mathbf{Z}, \neg\mathbf{y}_i$);
        $\boldsymbol{\Gamma}_i \leftarrow$ BuildFootprint ($\mathbf{Z}, \mathbf{p} = i$);
    end
    for $i = 1$ to $a$ do
        for $j = i + 1$ to $a$ do
            // Compare the best performance footprints for two different algorithms
               and remove conflicts
            $\{\boldsymbol{\Gamma}_i,\boldsymbol{\Gamma}_j\} \leftarrow$ CompareFootprints($\boldsymbol{\Gamma}_i,\boldsymbol{\Gamma}_j,\mathbf{Z},\mathbf{p} = i,\mathbf{p} = j$);
        end
        // Compare the good and bad performance footprints of an algorithm and remove
           conflicts
        $\{\boldsymbol{A}_i,\boldsymbol{B}_i\} \leftarrow$ CompareFootprints($\boldsymbol{A}_i,\boldsymbol{B}_i,\mathbf{Z},\mathbf{y}_{B,i},\neg\mathbf{y}_{B,i}$);
    end
end
```

ALGORITHM 3: Footprint construction algorithm

```
Input: A matrix $\mathbf{Z} \in \mathbb{R}^{n\times 2}$ of coordinates in the $2D$ instance space and a vector $\mathbf{y} \in \{0,1\}^{n\times 1}$ of binary
       performance measures.
Output: A footprint $\Phi$.
Function BuildFootprint($\mathbf{Z},\mathbf{y}$) is
    $\mathbf{Z}_u \leftarrow$ UniquePoints ($\{\mathbf{z}_i | y_i = \mathrm{TRUE}\}$); // Find the unique points $\mathbf{Z}_u$ that have $y_i = \mathrm{TRUE}$
    $\{r,c\} \leftarrow$ MatrixSize ($\mathbf{Z}_u$);       // Find the number of rows and columns of the matrix
    $k \leftarrow \max(\min(\lceil r/20\rceil, 50), 3)$;
    $\varepsilon \leftarrow \left(k\Gamma(2)/\sqrt{r\pi}\right)(\mathrm{range}(\mathbf{z}_1) \times \mathrm{range}(\mathbf{z}_2))$;
    $\mathbf{c} \leftarrow$ DBSCAN ($\mathbf{Z}_u, k, \varepsilon$);  // Use DBSCAN to identify outliers and clusters of dense data
    $\Phi$.polygon $\leftarrow \emptyset$
    for $i = 1$ to $\max(\mathbf{c})$ do
        // For every detected cluster, build an $\alpha$-shape
        $\Phi$.polygon $\leftarrow$ JoinPolygons ($\Phi$.polygon, BuildAlphaShape ($\{\mathbf{z}_i | c_i = i\}$));
    end
    $\Phi$.area $\leftarrow$ FindPolygonArea ($\Phi$.polygon);
    $\Phi$.density $\leftarrow$ CountElements ($\Phi$.polygon, $\mathbf{Z}$) /$\Phi$.area;
    $\Phi$.purity $\leftarrow$ CountElements ($\Phi$.polygon, $\{\mathbf{z}_i | y_i = \mathrm{TRUE}\}$) /CountElements ($\Phi$.polygon, $\mathbf{Z}$);
end
```

ALGORITHM 4: Footprint comparison algorithm

```
Input: A base and test footprints $\{\Phi_B, \Phi_T\}$, a matrix $\mathbf{Z} \in \mathbb{R}^{n\times 2}$ of coordinates in the 2D instance space
       and two vectors $\mathbf{y}_B, \mathbf{y}_T \in \{0,1\}^{n\times 1}$ of binary performance measures.
Output: A base and test footprints $\{\Phi_B, \Phi_T\}$ with removed contradictions.
Function CompareFootprints($\Phi_B, \Phi_T, \mathbf{Z}, \mathbf{y}_B, \mathbf{y}_T$) is
    $C \leftarrow$ PolygonIntersection ($\Phi_B$.polygon, $\Phi_T$.polygon);
    $N_{\text{try}} \leftarrow 0$;
    $N_{\max} \leftarrow 3$;
    while FindPolygonArea ($C$) $> 0 \wedge N_{try} < N_{\max}$ do
        $\pi_B \leftarrow$ CountElements $\left(C, \left\{\mathbf{z}_i | y_{B,i} = \text{TRUE}\right\}\right)$ /CountElements ($C, \mathbf{Z}$);
        $\pi_T \leftarrow$ CountElements $\left(C, \left\{\mathbf{z}_i | y_{T,i} = \text{TRUE}\right\}\right)$ /CountElements ($C, \mathbf{Z}$);
        if $\pi_B > \pi_T$ then
            $\Phi_T$.polygon $\leftarrow$ RemovePolygon ($\Phi_T$.polygon, $C$);
        else if $\pi_B < \pi_T$ then
            $\Phi_B$.polygon $\leftarrow$ RemovePolygon ($\Phi_B$.polygon, $C$);
        else
            break;
        end
        $C \leftarrow$ PolygonIntersection ($\Phi_B$.polygon, $\Phi_T$.polygon);
        $N_{\text{try}} \leftarrow N_{\text{try}} + 1$;
    end
    $\Phi_B$.area $\leftarrow$ FindPolygonArea ($\Phi_B$.polygon);
    $\Phi_B$.density $\leftarrow$ CountElements ($\Phi_B$.polygon, $\mathbf{Z}$) /$\Phi_B$.area;
    $\Phi_B$.purity $\leftarrow$ CountElements ($\Phi_B$.polygon, $\{\mathbf{z}_i | y_i = \text{TRUE}\}$) /CountElements ($\Phi_B$.polygon, $\mathbf{Z}$);
    $\Phi_T$.area $\leftarrow$ FindPolygonArea ($\Phi_T$.polygon);
    $\Phi_T$.density $\leftarrow$ CountElements ($\Phi_T$.polygon, $\mathbf{Z}$) /$\Phi_T$.area;
    $\Phi_T$.purity $\leftarrow$ CountElements ($\Phi_T$.polygon, $\{\mathbf{z}_i | y_i = \text{TRUE}\}$) /CountElements ($\Phi_T$.polygon, $\mathbf{Z}$);
end
```

ACKNOWLEDGMENTS

The authors are grateful to Professors Telmo Filho and Ricardo Prudencio for sharing their code for feature calculation of IRT.

REFERENCES

[1] Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, Salvador García, Luciano Sánchez, and Francisco Herrera. 2011. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing* 17, 2–3 (2011), 255–287.

[2] M. Fatih Amasyali and Okan K. Ersoy. 2009. *A Study of Meta Learning for Regression.* Technical Report. ECE Technical Reports, Purdue University.

[3] Yu Chen, Telmo Silva Filho, Ricardo B. C. Prudêncio, Tom Diethe, and Peter Flach. 2019. IRT: A new item response model and its applications. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, Vol. 89.

[4] M. Daszykowski, B. Walczak, and D. L. Massart. 2001. Looking for natural patterns in data: Part 1. Density-based approach. *Chemometrics and Intelligent Laboratory Systems* 56, 2 (2001), 83–92. DOI:https://doi.org/10.1016/S0169-7439(01)00111-3

[5] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml.

[6] María José Gacto, Jose Manuel Soto-Hidalgo, Jesús Alcalá-Fdez, and Rafael Alcalá. 2019. Experimental study on 164 algorithms available in software tools for solving standard non-linear regression problems. *IEEE Access* 7 (2019), 108916–108939.

[7] Taciana A. F. Gomes, Ricardo B. C. Prudêncio, Carlos Soares, André L. D. Rossi, and André Carvalho. 2012. Combining meta-learning and search techniques to select parameters for support vector machines. *Neurocomputing* 75, 1 (2012), 3–13.
[8] Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. 2014. *Real-Parameter Black-Box Optimization Benchmarking BBOB-2010: Experimental Setup.* Technical Report RR-7215. INRIA. Retrieved from http://coco.lri.fr/downloads/download15.02/bbobdocexperiment.pdf.
[9] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. 2019. *Automated Machine Learning.* Springer.
[10] Rob J. Hyndman and Anne B. Koehler. 2006. Another look at measures of forecast accuracy. *International Journal of Forecasting* 22, 4 (Oct. 2006), 679–688. DOI: https://doi.org/10.1016/j.ijforecast.2006.03.001
[11] Yanfei Kang, Rob J. Hyndman, and Kate Smith-Miles. 2017. Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting* 33, 2 (2017), 345–358.
[12] Petr Kuba, Pavel Brazdil, Carlos Soares, and Adam Woznica. 2002. Exploiting sampling and meta-learning for parameter setting support vector machines. In *Proceedings of the IBERAMIA.* Vol. 2002. 217–225.
[13] MultiMedia LLC. 2019. International Institution of Forecasters. Retrieved from https://forecasters.org/resources/time-series-data/m3-competition/.
[14] Ana C. Lorena, Aron I. Maciel, Péricles B. C. de Miranda, Ivan G. Costa, and Ricardo B. C. Prudêncio. 2018. Data complety meta-features for regression problems. *Machine Learning* 107, 1 (2018), 209–246.
[15] Julián Luengo and Francisco Herrera. 2015. An automatic extraction method of the domains of competence for learning classifiers using data complexity measures. *Knowledge and Information Systems* 42, 1 (2015), 147–180.
[16] Fernando Martínez-Plumed, Ricardo B. C. Prudêncio, Adolfo Martínez-Usó, and José Hernández-Orallo. 2019. Item response theory in AI: Analysing machine learning classifiers at the instance level. *Artificial Intelligence* 271, June 2019 (2019), 18–42.
[17] Mario A. Muñoz and Kate A. Smith-Miles. 2019. Generating new space-filling test instances for continuous black-box optimization. *Evolutionary Computation* 28, 3 (2019), 379–404. DOI: https://doi.org/10.1162/evco_a_00262
[18] Mario A. Muñoz and Kate Smith-Miles. 2017. Performance analysis of continuous black-box optimization algorithms via footprints in instance space. *Evolutionary Computation* 25, 4 (2017), 529–554.
[19] Mario A. Muñoz, Laura Villanova, Davaatseren Baatar, and Kate Smith-Miles. 2018. Instance spaces for machine learning classification. *Machine Learning* 107, 1 (2018), 109–147.
[20] Mario A. Muñoz and Kate Smith-Miles. 2020. Instance Space Analysis: A Toolkit for the Assessment of Algorithmic Power. Retrieved from https://github.com/andremun/InstanceSpace.
[21] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, 85 (2011), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html.
[22] John R. Rice. 1976. The algorithm selection problem. In *Advances in Computers.* Vol. 15. Elsevier, 65–118.
[23] Thomas P. Ryan. 2008. *Modern Regression Methods.* Vol. 655. John Wiley & Sons.
[24] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems* 42, 3 (July 2017), 21 pages. DOI: https://doi.org/10.1145/3068335
[25] Michael R. Smith, Tony Martinez, and Christophe Giraud-Carrier. 2014. An instance level analysis of data complexity. *Machine Learning* 95, 2 (2014), 225–256.
[26] Kate Smith-Miles, Davaatseren Baatar, Brendan Wreford, and Rhyd Lewis. 2014. Towards objective measures of algorithm performance across instance space. *Computers & Operations Research* 45, May 2014 (2014), 12–24. DOI: https://doi.org/10.1016/j.cor.2013.11.015
[27] Kate Smith-Miles, Mario A. Muñoz and Neelofar. 2019. MATILDA: Melbourne Algorithm Test Instance Library with Data Analytics. Retrieved from https://matilda.unimelb.edu.au.
[28] Kate Smith-Miles and Thomas T. Tan. 2012. Measuring algorithm footprints in instance space. In *Proceedings of the 2012 IEEE Congress on Evolutionary Computation.* IEEE, 3446–3453.
[29] Kate A. Smith-Miles. 2009. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys* 41, 1 (2009), 6.
[30] Carlos Soares, Pavel B. Brazdil, and Petr Kuba. 2004. A meta-learning method to select the kernel width in support vector regression. *Machine Learning* 54, 3 (2004), 195–209.
[31] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2013. OpenML: Networked science in machine learning. *SIGKDD Explorations* 15, 2 (2013), 49–60. DOI: https://doi.org/10.1145/2641190.2641198
[32] Ricardo Vilalta and Youssef Drissi. 2002. A perspective view and survey of meta-learning. *Artificial Intelligence Review* 18, 2 (2002), 77–95.

Received February 2020; revised July 2020; accepted November 2020