Mach Learn (2018) 107:109–147
https://doi.org/10.1007/s10994-017-5629-5

# Instance spaces for machine learning classification

**Mario A. Muñoz**[1] · **Laura Villanova**[1] · **Davaatseren Baatar**[1] · **Kate Smith-Miles**[1]

Received: 10 May 2016 / Accepted: 20 January 2017 / Published online: 28 December 2017


**Abstract** This paper tackles the issue of objective performance evaluation of machine learning classifiers, and the impact of the choice of test instances. Given that statistical properties or features of a dataset affect the difficulty of an instance for particular classification algorithms, we examine the diversity and quality of the UCI repository of test instances used by most machine learning researchers. We show how an instance space can be visualized, with each classification dataset represented as a point in the space. The instance space is constructed to reveal pockets of hard and easy instances, and enables the strengths and weaknesses of individual classifiers to be identified. Finally, we propose a methodology to generate new test instances with the aim of enriching the diversity of the instance space, enabling potentially greater insights than can be afforded by the current UCI repository.

**Keywords** Classification · Meta-learning · Test data · Instance space · Performance evaluation · Algorithm footprints · Test instance generation · Instance difficulty

## 1 Introduction

The practical importance of machine learning (ML) has resulted in a plethora of algorithms in recent decades (Carbonell et al. 1983; Flach 2012; Jordan and Mitchell 2015). Are new and improved machine learning algorithms really better than earlier versions? How do we objectively assess whether one classifier is more powerful than another? Common practice is to test a classifier on a well-studied collection of classification datasets, typically from the UCI repository (Wagstaff 2012). However, this practice is attracting increasing criticism (Salzberg 1997; Langley 2011; Wagstaff 2012; Macia and Bernadó-Mansilla 2014; Rudin

**Electronic supplementary material** The online version of this article (https://doi.org/10.1007/s10994-017-5629-5) contains supplementary material, which is available to authorized users.

✉ Kate Smith-Miles
kate.smith-miles@monash.edu

[1] School of Mathematical Sciences, Monash University, Clayton, VIC 3800, Australia

and Wagstaff 2014) due to concerns about over-tuning algorithm development to a set of test instances without enough regard to the adequacy of these instances to support further generalizations. While there is no doubt that the UCI repository has had a tremendous impact on ML studies, and has improved research practice by ensuring comparability of performance evaluations, there is concern that the repository may not be a representative sample of the larger population of classification problems (Holte 1993; Salzberg 1997). We must challenge whether chosen test instances are enabling us to evaluate algorithm performance in an unbiased manner, and we must seek new tools and methodologies that enable us to generate new test instances that drive improved understanding of the strengths and weaknesses of algorithms. The development of such methodologies to support objective assessment of ML algorithms is at the core of this study.

As stated by Salzberg (1997), "the UCI repository is a very limited sample of problems, many of which are quite easy for a classifier". Additionally, because of the intensive use of the repository, there is increasing knowledge about its problem instances. Such knowledge inevitably translates into the development of new algorithms that can be biased towards known properties of the UCI datasets. Therefore, algorithms that work well on a handful of UCI datasets might not work well on new or less popular problem instance classes. If these less-popular instances are found to be prevalent in a particular critical application area, such as medical diagnostics, the consequences for selecting an algorithm that does not generalize well to this application domain could be severe.

Indeed, based on the No-Free-Lunch (NFL) theorems (Culberson 1998; Igel and Toussaint 2005), it is unlikely that any one algorithm always outperforms other algorithms for all possible instances of a given problem. Given the large number of available algorithms, it is challenging to identify which algorithm is likely to be best for a new problem instance or class of instances. This challenge is referred to as the Algorithm Selection Problem (ASP). A powerful framework to address the ASP was proposed by Rice (1976). The framework relies on measurable features of the problem instances, correlated with instance difficulty, to predict which algorithm is likely to perform best. Rice's framework was originally developed for solvers of partial differential equations (Weerawarana et al. 1996; Ramakrishnan et al. 2002); it was then generalized to other domains such as classification, regression, time-series forecasting, sorting, constraint satisfaction, and optimization (Smith-Miles 2008). For the machine learning community, the idea of measuring statistical features of classification problems to predict classifier performance, using machine learning methods to learn the model, developed into the well-studied field of meta-learning (learning about learning algorithm performance) (Aha 1992; Brazdil et al. 2008; Ali and Smith 2006; Lee and Giraud-Carrier 2013).

Beyond the challenge of accurately predicting which algorithm is likely to perform best for a given problem instance, based on a learned model of the relationship between instance features and algorithm performance, is the challenge to explain why. Smith-Miles and co-authors have developed a methodology over recent years through a series of papers (Smith-Miles and Lopes 2012; Smith-Miles et al. 2014; Smith-Miles and Bowly 2015) that extend Rice's framework to provide greater insights into algorithm strengths and weaknesses. Focusing on combinatorial optimization problems such as graph coloring, the methodology first involves devising novel features of problem instances that correlate with difficulty or hardness (Smith-Miles and Tan 2012; Smith-Miles et al. 2013), so that existing benchmark instances can be represented as points in a high-dimensional feature space before dimension reduction techniques are employed to project to a 2-D instance space. Within this instance space (Smith-Miles et al. 2014), the performance of algorithms can be visualized and pockets of the instance space corresponding to algorithm strengths and weaknesses can be identified and analyzed to understand which instance properties are being exploited or are causing difficulties for an

algorithm. Objective measures can be calculated that summarize each algorithm's relative power across the broadest instance space, rather than on a collection of existing test instances. Finally, the location of the existing benchmark instances in the instance space reveals much about their diversity and challenge, and a methodology has been developed to evolve new test instances to fill and broaden the instance space (Smith-Miles and Bowly 2015). This methodology has so far only been applied to combinatorial optimization problems such as graph coloring, although its broader applicability makes it suitable for other problem domains including machine learning.

Of course the decades of work in meta-learning has already contributed significant knowledge about how the measurable statistical properties of classification datasets affect difficulty for accurate machine learning classification. This relates to the first stage of the aforementioned methodology. It remains to be seen how these features should be selected to create the most useful instance space for classification problems; what can be learned about machine learning classifiers in this space; and whether the existing UCI repository instances are sufficiently diverse when viewed from the instance space. Further, the question of how to evolve new classification test instances to fill this space needs to be carefully considered, since it is a more challenging task than evolving graphs in our previous work (Smith-Miles and Bowly 2015) which have a relatively simple structure of nodes and edges. In the current work we revisit the domain of ML and adapt and extend the proposed methodology to enable objective assessment of the performance of supervised learning algorithms, which are the most widely used ML methods (Hastie et al. 2005). The diversity of the UCI repository instances will be visualized, along with algorithm strengths and weaknesses, and a methodology for generation of new test classification instances will be proposed and illustrated.

The remainder of this paper is organized as follows. Section 2 summarizes the methodology that will be employed based on an extended Rice framework. Section 3 describes the building blocks of the methodology when applied to machine learning classification, namely the meta-data composed of problem instances, features, algorithms and performance metrics. In Sect. 4, we describe the statistical methodology used to identify a subset of features that capture the challenges of classification. Section 5 then demonstrates that these selected features are adequate by showing how accurately they can predict the performance of ML algorithms. In Sect. 6, details are presented of the process employed to generate a 2-dimensional *instance space* where the relative difficulty of the UCI instances and algorithm performances across the space are visualized. This includes a new dimension reduction methodology that has been developed to improve the interpretability of the visualizations. Section 7 shows how the instance space can be used for objective assessment of algorithm performance, and to gain insights into strengths and weaknesses. Section 8 then presents a proof-of-concept for a new method for generating additional test instances in the instance space, and illustrates how an augmented UCI repository could be developed. Finally, Sect. 9 presents our conclusions and outlines suggestions for further research. Supplementary material[1] that provides more detail about the developed features and all datasets and code[2] used to calculate the features are available online.

## 2 Methodological framework

The methodology used in this study extends upon the Algorithm Selection Problem framework of Rice (1976), shown in the blue box of Fig. 1. It has been extended to enable more

[1] http://users.monash.edu.au/~ksmiles/matilda/machinelearning/supplementary.pdf.

[2] http://users.monash.edu.au/~ksmiles/matilda/classification.zip.

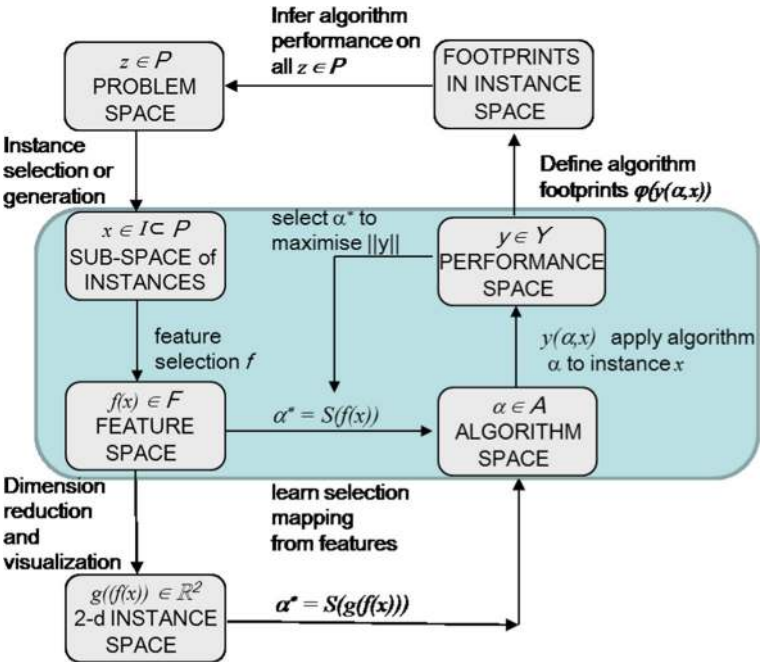


**Fig. 1** Methodological framework, extending Rice's Algorithm Selection Problem shown within the blue box

than performance prediction of algorithms given instance features: the extended framework (Smith-Miles et al. 2014) enables visualization of the instance space, instance difficulty, algorithm performance, and objective measurement of algorithmic power. The framework is composed of several spaces, which are described further below.

The *problem space*, $\mathcal{P}$, is composed of instances of a given problem for which we have computational results for a given subset $\mathcal{I}$. In this paper, $\mathcal{I}$ contains the classification datasets from the UCI repository. The *feature space*, $\mathcal{F}$, contains multiple measures characterizing properties (correlating with difficulty) of the instances in $\mathcal{I}$. The *algorithm space*, $\mathcal{A}$, contains a portfolio of selected algorithms to solve the problem, in this case, classification algorithms. The *performance space*, $\mathcal{Y}$, contains measures of performance for the algorithms in $\mathcal{A}$ evaluated on the instances in $\mathcal{I}$. For a given problem instance $x \in \mathcal{I}$ and a given algorithm $\alpha \in \mathcal{A}$, a feature vector $\mathbf{f}(x) \in \mathcal{F}$ and algorithm performance metric $y(\alpha, x) \in \mathcal{Y}$ are measured. By repeating the process for all instances in $\mathcal{I}$ and all algorithms in $\mathcal{A}$, the meta-data $\{\mathcal{I}, \mathcal{F}, \mathcal{A}, \mathcal{Y}\}$ are generated. Within the framework of Rice (1976), we can now learn, using regression or more powerful supervised learning methods, the relationship between the features and the algorithm performance metric, to enable performance prediction. Full details of this meta-data for the domain of classification, including the choice of features, will be provided in Sect. 3.

The aim of the extended methodology shown in Fig. 1 however is to gain insights into why some algorithms might be more or less suited to certain instance classes. In our extended framework, the meta-data is used to learn the mapping $g(\mathbf{f}(x), y(\alpha, x))$, which projects the instance $x$ from a high-dimensional feature space to a 2-dimensional space. The resulting 2-dimensional space, referred to as *instance space*, is generated in such a way as to result in linear trend of features and algorithm performance across different directions of the instance

space, increasing the opportunity to infer how the properties of instances affect difficulty. A new approach to achieving an optimal 2-D projection has been proposed for this paper, and the details are presented in "Appendix A". Following the optimal projection of instances to a 2-D instance space, each classification dataset is now represented as a single point in $\mathbb{R}^2$, so the distribution of existing benchmark instances can be viewed across the instance space, and their diversity assessed. Further, the distribution of features and performance metrics for each algorithm can also be easily viewed to provide a snapshot of the adequacy of instances and features to describe algorithm performance. Instances are adequate if they are diverse enough to expose areas where an algorithm performs poorly, as well as areas where an algorithm performs well. Features are adequate if they allow accurate prediction of algorithm performance while explaining critical similarities and differences between instances.

The 2-dimensional instance space, color-coded with algorithm performance, is then investigated to identify in which region each algorithm $\alpha$ is expected to perform well. Such a region is referred to as the algorithm's *footprint*. The area of the footprint can be calculated to objectively measure each algorithms' expected strength across the entire instance space, rather than on chosen test instances. The resulting measure is the *algorithmic power*.

The 2-dimensional instance space is further investigated to seek explanation as to why algorithm $\alpha$ performs well (or poorly) in different regions of the instance space. Since the projection has been achieved in a manner that creates linear trends across the space for features and algorithm performance, footprint areas can more readily be described in terms of the statistical properties of the instances found in each footprint.

The final component of the proposed methodology involves revisiting the distribution of the existing instances $\mathcal{I}$ in the instance space, and identifying target points where it would be useful to have additional instances created from the space of all possible instances $\mathcal{P}$. A methodology for evolving instances has previously been proposed for generating graphs with controllable characteristics for graph coloring problems (Smith-Miles and van Hemert 2011), but will be adapted in this paper to generate new classification datasets that lie at specific locations in the instance space.

In summary, the proposed methodology requires:

(i) Construction of the meta-data, including a set of candidate features (see Sect. 3);
(ii) Selection of a subset of candidate features (see Sect. 4);
(iii) Justification of selected features using performance prediction accuracy (see Sect. 5);
(iv) Creation of a 2-D instance space for visualization of instances and their properties (see Sect. 6);
(v) Objective measurement of algorithmic power (see Sect. 7); and
(vi) Generation of new test instances to fill the instance space (see Sect. 8).

Each of these steps in this general methodology requires considerable innovation and thought when tailored to a new application domain. The following six sections will describe these steps in more detail for our classification study.

## 3 Meta-data for supervised classification algorithms

Let $\mathbf{X}^i = [\mathbf{x}_1^i \ \mathbf{x}_2^i \ \ldots \ \mathbf{x}_p^i] \in \mathbb{R}^{q \times p}$ be the data matrix, where $p$ is the number of observations and $q$ is the number of attributes; then let $\mathbf{c}^i \in \{1, \ldots, K\}^p$ , $K \geq 2$ be the class vector taking on $K \in \mathbb{N}$ labels.

A supervised learning problem, referred to as problem instance, consists of a collection of $(\mathbf{x}_j, c_j)$ pairs. In this work, the input $\mathbf{x}_j$ is a $q$-dimensional input vector that may comprise of binary, nominal and numeric values; the output label $c_j$ (class) takes on one of $K$ labels. The focus is therefore on binary and multi-class classification. Typically, the data matrix $\mathbf{X}^i$ is divided into a training set and a test set. The learning task is to infer, from the training set, an approximating function relating the attributes to the class labels. The inferred function is then used to predict the labels in the test set, which consists of input vectors previously unseen by the learning algorithm. The performance of the learning algorithm is measured by a metric comparing true and predicted labels. The lower the degree of discrepancy between true and predicted labels, the better the algorithm performance.

The focus in meta-learning is to study how measurable features of the problem instances $(\mathbf{X}^i, \mathbf{c}^i)$ affect a given learning algorithm's performance metric. Problem instances $\mathcal{I}$, learning algorithms $\mathcal{A}$, and performance metrics $\mathcal{Y}$, are three of the four elements composing the meta-data $\{\mathcal{I}, \mathcal{F}, \mathcal{A}, \mathcal{Y}\}$. We will briefly describe these elements of the meta-data used in this study below, before presenting a more extensive discussion about the critical choice of features $\mathcal{F}$.

### 3.1 Problem instances $\mathcal{I}$

The problem instances we have used in this research consist of classification datasets comprising of one or more input variables (attributes) and one output variable (class). Datasets have been downloaded from two main sources, namely the University of California Irvine (UCI) repository (Lichman 2013) and the Knowledge Extraction Evolutionary Learning (KEEL) repository (Alcalá et al. 2010); additionally, a few datasets from the Data Complexity library (DCol—http://dcol.sourceforge.net/) have been used.

KEEL and DCol datasets rely on a convenient common format, the KEEL format. It originates from the ARFF format employed in the popular Waikato Environment for Knowledge Analysis (WEKA) suite (Holmes et al. 1994). Along with the dataset itself, KEEL and ARFF files carry information about dataset name, attributes name and type, and values taken on by both nominal and class attributes; additionally, the class attribute always occupies the last column of the data matrix. The use of this common format facilitates standardization and minimizes errors deriving from data manipulation; this is particularly true when many datasets need to be analyzed and automatic procedures are employed.

In constrast, UCI data files vary greatly in their format. Often, multiple files have to be merged to generate the final dataset; sometimes, information about the data themselves is not (clearly) available. UCI classification datasets have been extensively investigated and detailed information has been provided for the pre-processing of 166 datasets (Macia and Bernadó-Mansilla 2014) which we have adopted in this study.

Overall, we have used a total of 235 problem instances comprising 210 UCI instances, 19 KEEL instances and 6 DCol instances. A list of the problem instances and links to the files are provided in Sect. 1 of the Supplementary Material.

The selected 235 problem instances have up to 11,055 observations and up to 1558 attributes. Larger instances could have been selected, but were excluded due to the need to impose some computational constraints when deriving the features and running the algorithms described below.

Multiple datasets present missing values. For these datasets, two problem instances are derived. In the first problem instance the missing values are maintained, whereas in the second problem instance the missing values are estimated. The estimating procedure is as follow. Let $k$ be the class label of an instance with missing value(s). If the missing value pertains to a numeric attribute, the missing value is replaced with the average value of the attribute

computed over all the instances with class label $k$. For a nominal attribute, the mode is used (Orriols-Puig et al. 2010). This class-based imputation approach has been shown to efficiently achieve good accuracy and outperform other more complex methodologies (Fujikawa and Ho 2002; Young et al. 2011). For those cases where missing values are the only available data for a given class, imputation through global average/mode is used. Finally, instances with missing values in the class attribute are omitted. Note that all algorithms use the same data, hence, any unintended advantage due to the chosen imputation approach will be shared by all algorithms.

### 3.2 Algorithms $\mathcal{A}$

We consider a portfolio of ten popular supervised learners representing a comprehensive range of learning mechanisms. The algorithms are Naive Bayes (NB), Linear Discriminant (LDA), Quadratic Discriminant (QDA), Classification and Regression Trees (CART), J48 decision tree (J48), k-Nearest Neighbor (KNN), Support Vector Machines with linear, polynomial and radial basis kernels (L-SVM, poly-SVM, and RB-SVM respectively), and random forests (RF). NB, J48, CART and RF are expected to give uncorrelated errors while providing a good diversity of classification mechanisms (Lee and Giraud-Carrier 2013); LDA and QDA are expected to further extend the diversity of the algorithm portfolio, whereas KNN and SVM are considered because of their popularity. The R packages employed are `e1071` (Meyer et al. 2015) (NB, L-SVM, poly-SVM, RB-SVM, RF), `MASS` (Venables and Ripley 2002) (LDA, QDA), `rpart` (Therneau et al. 2014) (CART), `RWeka` (Holmes et al. 1994) (J48) and `kknn` (Hechenbichler 2014). For all of the packages, the default parameters value are used.

### 3.3 Performance metric $\mathcal{Y}$

There exist various measures of algorithm performance focusing on either prediction accuracy/error or computation time/cost. In this work, we consider measures of prediction accuracy/error which evaluate how well or poorly the labels are classified. The performance of a supervised learner is derived by comparing labels in the problem instance (data labels) and labels predicted by the algorithm (predicted labels).

In a binary classification, where the class labels are either positive or negative, the comparison is based on four counts. The counts are the number of (i) positive labels that are correctly classified (true positives *tp*), (ii) negative labels that are wrongly classified (false positives *fp*), (iii) negative labels that are correctly classified (true negatives *tn*), and (iv) positive labels that are wrongly classified (false negatives *fn*) (Sokolova and Lapalme 2009). The proportion of incorrectly classified labels is the *Error Rate*. The proportion of positive predicted labels that are correctly classified is the *Precision*. The proportion of positive data labels that are correctly classified is the *Recall*. The harmonic mean of precision and recall is the *F1-measure*.

In multi-class classification, problem instances with $K$ class labels are usually decomposed into $K$ binary problem instances. For each of the $K$ problem instances, counts are derived and used to calculate an overall performance measure. There exist two different strategies to derive the overall performance measure. One strategy is to calculate $K$ performance measures (one for each sub-problem) and average them out. This is referred to as macro-averaging and generates measures such as macro-Precision, macro-Recall and macro-F1. The other strategy is to obtain cumulative counts of the form $tp = \sum_{k=1}^{K} tp_k$ and use them to calculate the overall performance value. This is referred to as micro-averaging and generates measures such as

**Table 1** Overview of performance measures for both binary and multi-class classification

| Measure | Formula | |
|---|---|---|
| | Binary | Multi-class |
| Error Rate (*ER*) | $\frac{fp+fn}{n}$ | $\sum_{k=1}^{K} ER_k \cdot w_k$ |
| Precision | $\frac{tp}{tp+fp}$ | $\sum_{k=1}^{K} Precision_k \cdot w_k$ |
| Recall | $\frac{tp}{(tp+fn)}$ | $\sum_{k=1}^{K} Recall_k \cdot w_k$ |
| F1-measure (Fm) | $(\beta^2+1) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$ | $\sum_{k=1}^{K} Fm_k \cdot w_k$ |

Here, $\beta$ is a non-negative real value that we set to 1

micro-Precision, micro-Recall and micro-F1 (Tsoumakas and Vlahavas 2007; Sokolova and Lapalme 2009). While macro-averaging treats all classes equally, micro-averaging favours bigger classes (Sokolova and Lapalme 2009) biasing the overall performance toward the performance on the bigger classes. Overall, the choice of the most suitable averaging strategy depends on the purpose of the study.

In the current work, the purpose is to assess algorithm performance by adopting a broad perspective and targeting a whole range of problem instances. Therefore, we do not wish to place too much emphasis on algorithms that perform particularly well for large classes; similarly, we do not wish to disregard class size information completely. Therefore, we adopt an intermediate strategy consisting of averaging class-based performance measures (similarly to macro-averaging) but using weights that are proportional to the class size (i.e. $w_k = n_k/n$, where $n_k$ denotes the number of instances with label $k$).

In addition to the aforementioned metrics other performance measures exist (e.g. Break Even Point, Area Under the Curve—AUC); however, they are either a function of other measures or metrics that are not well developed for multi-class classification (Sokolova and Lapalme 2009). Because our problem instances embrace both binary and multi-class classification, we restrict our attention to Error Rate (ER), Precision, Recall and F1-measure using a weighted macro-average strategy, shown in Table 1.

### 3.4 Features $\mathcal{F}$

Useful features of a classification dataset are measurable properties that (i) can be computed in polynomial time and (ii) are expected to expose what makes a classification problem hard for a given algorithm.

It is well known for example that problems in high-dimensions tend to be hard for algorithms like nearest neighbor (Vanschoren 2010); indeed, the density of the data points decreases exponentially as the number of attributes increases and point density is an important requirement for nearest neighbor. Similarly, problems with highly unbalanced classes tend to be hard for algorithms like unpenalized Support Vector Machines and Discriminant Analysis (Ganganwar 2012); indeed, the algorithms' assumptions (e.g. equal distribution of data within the classes, balanced dataset) are not met. In the above mentioned cases, simple examples of relevant features are number of instances in the dataset, number of attributes, and percentage of instances in the minority class.

Features for classification problems have a relatively long history in the meta-learning field, with the first studies dating back to the early 1990s (Rendell and Cho 1990; Aha 1992; Brazdil et al. 1994; Michie et al. 1994; Gama and Brazdil 1995). Over the following years,

many authors used existing features and investigated new features based on either metrics (Perez and Rendell 1996; Vilalta 1999; Pfahringer et al. 2000a; Smith et al. 2002; Vilalta and Drissi 2002; Goethals and Zaki 2004; Ali and Smith 2006; Segrera et al. 2008; Song et al. 2012) or model fitting (Bensusan and Giraud-Carrier 2000; Peng et al. 2002; Ho and Basu 2002). Various manuscripts have provided a snapshot of the most popular features over the years (Castiello et al. 2005; Smith-Miles 2008; Vanschoren 2010; Balte et al. 2014). As the development of new features emerged, it became common practice to classify the meta-features into eight different groups: (i) simple, (ii) statistical, (iii) information theoretic, (iv) landmarking, (v) model-based, (vi) concept characterisation, (vii) complexity, and (viii) itemset-based meta-features.

An overview of these groups of features is reported below:

1. *Simple features* measure basic aspects related to dimensionality, type of attributes, missing values, outliers, and class attribute. They have been regularly adopted in meta-learning studies since the pioneering works by Rendell and Cho (1990) and Aha (1992).
2. *Statistical features* make use of metrics from descriptive statistics (e.g. mean, standard deviation, skewness, kurtosis, correlation), hypothesis testing (e.g. $p$-value, Box's $M$-statistic) and data analysis techniques (e.g. canonical correlation, Principal Component Analysis) to extract information about single attributes as well as multiple attributes simultaneously.
3. *Information theoretic features* quantify the information present in attributes that are investigated either alone (e.g. entropy) or in combination with class label information (e.g. mutual information).
4. *Landmarking features* are performance measures of simple and efficient learning algorithms (landmarkers) such as Naive Bayes, Linear Discriminant, 1-Nearest Neighbor and single-node trees (Pfahringer et al. 2000a, b). The idea behind the approach is that landmarker performance can shed light on the properties of a given problem instance (Bensusan and Giraud-Carrier 2000). For example, good performance of a linear discriminant classifier indicates that the classes are likely to be linearly separable; on the contrary, bad performance indicates probable non-linearly separable classes. In a meta-learning study, multiple and diverse landmarkers are used, so that each landmarker can contribute an area of expertise. The collection of areas of expertise to which a problem instance belongs, can then be used to characterize the problem instance itself (Bensusan and Giraud-Carrier 2000). There exist multiple variants of the landmarking approach. One such variant that is relevant for the current work and not yet herein implemented, is sampling landmarking (Fürnkranz and Petrak 2001; Soares et al. 2001; Leite and Brazdil 2008). Sampling landmarking considers computationally complex algorithms and evaluates their performance on a collection of data subsets. The use of data subsets allows saving computational time without affecting results significantly; indeed, running an algorithm on the full dataset or on a collection of data subsets is expected to give a similar profile of algorithm expertise (Fürnkranz and Petrak 2001).
5. *Model-based features* aim to characterize problem instances using the structural shape and size of decision trees fitted to the instances themselves (Peng et al. 2002). Examples are number of nodes and leaves, distribution of nodes at each level and along each branch, width and depth of the tree.
6. *Concept characterization features* measure the sparsity of the input space and the irregularity of the input-output distribution (Perez and Rendell 1996; Vilalta and Drissi 2002). Irregular input-output distributions occur when neighboring examples in the input space have different labels in the output space. Concept characterization features were shown

to provide much information about the difficulty of problem instances (Vilalta 1999; Robnik-Šikonja and Kononenko 2003). Unfortunately, they have a high computational cost because they require the calculation of the distance matrix.

7. *Complexity features* measure the geometrical characteristics of the class distribution and focus on the complexity of the boundary between classes (Ho and Basu 2002). The aim is to identify problem instances having ambiguous classes. The ambiguity of the class attribute might be an intrinsic property of the data or might derive from inadequate measurements of the attributes; class ambiguity is likely to be influenced by sparsity and high-dimensionality (Ho and Basu 2002; Macia and Bernadó-Mansilla 2014). In general, complexity features investigate (i) class overlap measured in the input space, (ii) class separability, and (iii) geometry, topology and density of manifolds (Macià et al. 2010).
8. *Itemsets- and association rules-based features* measure the distribution of values of both single attributes and pairs of attributes, as well as characteristics of the interesting variable relations (Song et al. 2012; Burton et al. 2014). In this approach, the original problem instance is transformed into a binary dataset. For nominal attributes, each distinct attribute value in the original data generates a new attribute in the binary data. For numeric attributes a discretization method is applied first. The frequency of each binary attribute is then measured, as well as the frequency of pairs of binary attributes.

Concept characterization features and, in the large majority, statistical features are suitable for numerical attributes only; information theoretic features are suitable for nominal attributes. To allow for all the features to be calculated on all the attributes, the original attributes can be pre-processed. On the one hand, we convert nominal attributes into numeric attributes by replacing labels (e.g. $\{A, B, C\}$) with numbers (e.g. $\{1, 2, 3\}$). Despite being not ideal for standard statistical analysis, this approach is of value for the purpose of the current study because it allows us to take into account potentially important relationships between numeric and nominal attributes. Note that the aforementioned transformation is not applied to the class attribute which remains unaltered throughout the analysis. On the other hand, we discretize numeric attributes using ten intervals of equal width, thus obtaining nominal attributes with ten categories. Discretization using a fixed interval width is one of the many discretization approaches existing in the literature [e.g. equal frequencies, given interval boundaries, k-means clustering, Fayyad and Irani method (Fayyad and Irani 1992)]. The motivation behind our choice lies in the simplicity and consistency of the approach throughout the problem instances. By using ten categories we discretize the original attribute without losing too much information.

When deriving features values it is possible to obtain a single number (e.g. number of instances, number of attributes), a vector (e.g. vector of attributes' entropies) or a matrix (e.g. absolute correlation matrix). When the output is a vector or matrix, further processing is required. A typical procedure is to generate a single feature value by calculating the mean of the vector or matrix; however, this can result in a considerable loss of information. To preserve a certain degree of distributional information, several authors have proposed the use of summary statistics (Michie et al. 1994; Brazdil et al. 1994; Lindner and Studer 1999; Soares and Brazdil 2000). We adopt this approach and calculate minimum, maximum, mean, standard deviation, skewness and kurtosis from the vector/matrix values. Therefore, for each vector or matrix property we obtain six new features.

Most of the statistical and information theoretic features can be calculated on the attributes either independently or in conjunction with the information in the class attribute. We name features belonging to the second case with the suffix 'by class'. For example, assume our problem instance has two numeric attributes, $X_1$ and $X_2$, and the class attribute $C$ takes on

labels $\{c_1, c_2\}$; further assume that we want to calculate the feature 'mean standard deviation of attributes'. In the first case (calculation independent of class attribute) we (i) calculate the standard deviation of attribute $X_1$ and the standard deviation of attribute $X_2$, and (ii) average the two numbers; the resulting value is our feature 'mean standard deviation of attributes'. In the second case (calculation in conjunction with class attribute), we calculate (i) the standard deviation of attribute $X_1$ computed over all the instances that predict class $c_1$, (ii) the standard deviation of attribute $X_1$ computed over all the instances that predict class $c_2$, (iii) the standard deviation of attribute $X_2$ computed over all the instances that predict class $c_1$, (iv) the standard deviation of attribute $X_2$ computed over all the instances that predict class $c_2$. The four values are then averaged and we obtain our final feature 'mean standard deviation of attributes by class'.

In this study we have generated a set of 509 features derived from the eight types of features. Not all of these will be interesting for the challenge of understanding how features affect the performance of our chosen algorithms across our selected set of test instances. Our goal is to represent the instances in a feature space, generated to maximize our chances of gaining insights via visualization. The process of selecting relevant features from this candidate set of 509 features will be discussed in the following section.

## 4 Feature selection

The candidate set of 509 features derived from the available literature contains much redundancy, with many features measuring aspects of a problem instance that are either similar or not relevant to expose the hardness of the classification task itself. Thus, a small set of relevant features must be selected. To identify relevant features, we deliberately alter the hardness of the classification task and observe how the feature values react to such alteration. Overall, the procedure we adopt to select a relevant set of features is as follows:

1. Identify broad characteristics that are either known or expected to make a classification task harder (classification challenges);
2. Alter a problem instance to deliberately vary the hardness of the classification task based on the challenges identified in the previous step (instance alteration);
3. Calculate all 509 features on both original and altered problems;
4. Use a statistical procedure to compare features values of original and altered problems (statistical test);
5. Identify the set of relevant features as those most responsive to the challenges;
6. Evaluate the adequacy of the relevant features via performance prediction.

*Classification challenges* The algorithms listed in Sect. 3.2 are known to perform better under certain circumstances. Such circumstances broadly relate to algorithm assumptions (e.g. normality, equal covariance within classes) or characteristics of the data (e.g. numeric and/or nominal, presence of missing values). Based on previous investigations of classification algorithms (Lessmann et al. 2015; Sokolova and Lapalme 2009; Kotsiantis 2007; Kotsiantis et al. 2006; Vilalta 1999; Michie et al. 1994) we identify 12 challenging circumstances. They are:

– *Non-normality within classes* instances belonging to the same class do not follow a multivariate normal distribution;
– *Unequal covariances within classes* instances belonging to the same class follow a multivariate normal distribution; however, the variance-covariance matrices of the distributions are different;

- *Redundant attributes* two or more attributes carry very similar information;
- *Type of attributes* the problem instance comprises both numeric and nominal attributes;
- *Unbalanced classes* at least one class has a considerably different number of instances;
- *Constant attribute within classes* for at least one attribute, all the instances belonging to the same class assume the same (numeric or categorical) value;
- *(Nearly) Linearly dependent attributes* at least one numeric attribute is (nearly) a linear combination of another two or more numeric attributes;
- *Non-linearly separable classes* there exists no hyperplane that well separates the classes;
- *Missing values* a considerable number of instances present missing values for one or more attribute;
- *Data scaling* the scale of one or more attributes is very different from the scale of the remainder attributes;
- *Redundant instances* there exist a considerable number of repeated instances;
- *Lack of information* a limited number of instances is available.

*Instance alteration* Based on the above challenges, we alter a problem instance to change the difficulty of the classification task. For each challenge we obtain two problem instances that we want to compare; they are the original and altered datasets. The altered dataset is either more or less challenging in terms of a specific classification challenge. As an example, consider the challenge of non-linearly separable classes. We alter each dataset to make it more linearly separable by fitting a hyperplane through the data using linear regression, and then altering the class labels so that each side of the hyperplane contains only one class. The altered dataset is therefore less challenging than the original, and we will be able to see which features are different when compared to the original dataset and therefore correlate with non-linear separability. Each challenge is treated in a similar manner. Details of the applied alterations and comparisons are reported in Sect. 2 of the Supplementary Material. Table 2 reports the identified challenges and highlights which ones are likely to be relevant for the investigated algorithms.

*Statistical test* Original and altered datasets are compared in terms of their values of the 509 candidate features. For a given feature, its value is calculated on both the original and altered problem. A statistically significant difference in values suggests that the applied alteration (cause) results in a change of the feature value (effect) and that the feature is relevant to measuring the degree of the challenge presented by an instance. Furthermore, the bigger the difference, the higher the discriminating ability of the feature. We consider only one single challenge at a time as the aim is to identify features that are in a cause-effect relationship with classification hardness.

To draw statistically sound conclusions, a distribution of feature values is required for both the original and altered problem. For the altered problem, multiple values naturally arise by running the alteration process multiple times; due to the intrinsic randomness of the alteration process, a different altered problem is obtained in each simulation run. Instead, for the original problem no intrinsic randomness exists. Therefore, we introduce a small source of variability by randomly removing one observation (i.e. dataset row) from the problem instance in each simulation run. For consistency, the same observation is removed before applying the alteration process.

The two distributions of feature values are compared through a two-sided $t$-test with unequal variances. Unequal variances are considered because the feature values derived from the original problem are usually less variable than the feature values derived from the altered problems. It is well known that two types of errors can occur when performing a statistical test. On the one hand, assume we are testing a feature that has no cause-effect

**Table 2** Classification challenges specific to the investigated algorithms

| Challenge | Challenged algorithm | | | | | | |
|---|---|---|---|---|---|---|---|
| | NB | LDA | QDA | CART-J48 | KNN | SVM | RF |
| Non-normality within classes | | ✓ | ✓ | | | | |
| Unequal covariance within classes | | ✓ | | | | | |
| Redundant attributes | | | | | ✓ | | |
| Type of attributes | | ✓ | ✓ | | ✓ | | |
| Unbalanced classes | ✓ | | | | ✓ | | ✓ |
| Constant attribute within classes | | ✓ | ✓ | | | | |
| (Nearly) Linearly dependent attributes | ✓ | ✓ | | | | | |
| Non-linearly separable classes | | ✓ | | | | | |
| Missing values | | ✓ | ✓ | | ✓ | | |
| Data scaling | | | | | ✓ | ✓ | |
| Redundant instances | | | | | ✓ | | |
| Lack of information | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Algorithms for which a specific challenge is expected to be relevant, based on their model assumptions, are highlighted with the symbol ✓

relationship with a given challenge; the error we can make is to conclude that the feature is relevant (Type I error, $\alpha$). On the other hand, assume we are testing a feature that has a cause-effect relationship with a given challenge; the error we can make is to conclude that the feature is not relevant (Type II error, $\beta$). Before implementing the test, the value of $\alpha$ is fixed and the desired value of $\beta$ is specified. Additionally, a third value needs to be specified; this is the change in the feature value ($\Delta$) that we want to detect when comparing original and altered problems. The specified values of $\alpha$, $\beta$ and $\Delta$ are used to determine a suitable sample size, namely the number of repeats or simulation runs, required to simultaneously control Type I and II errors. When fixing the value of $\alpha$ it is important to consider that we are performing a large number of tests. 509 features are tested on 12 challenges, resulting in a total of $n_{tests} = 6108$ tests. Assuming that none of the features is relevant (i.e. no cause-effect relationship with the challenge), a test with $\alpha = 0.01$ would still select 123 features as relevant. To avoid this, a smaller value of $\alpha$ must be used in the test. Such a value is typically determined through a correction. Among the available corrections, we choose the Bonferroni correction $\alpha^* = \alpha/n_{tests}$, where $\alpha^*$ is the corrected value. Such a value is typically very small and results in a restrictive test. This well serves our purpose to identify a small set of suitable features.

Overall, we set $\alpha^* = 1.64e^{-6}$, $\beta = 0.1$, $\Delta = 3$ and obtain the optimal sample size $n_{runs} = 14$ through power analysis (Cohen 1992). In this context, the sample size is the number of comparisons required and corresponds to the number of simulated altered problem instances generated. Based on these settings, the test has (i) 99% chance to correctly discard a feature that has no cause-effect relationship with the challenge, and (ii) 90% chance to correctly select a feature that has a cause-effect relationship with the challenge. Features with $|p\text{-}value| < \alpha^*$ are identified as significant. For each single challenge, significant

features are sorted in ascending order based on their $p\text{-}value$, with the most relevant features appearing at the top of the list for each challenge.

*Set of relevant features* The procedure described above applies to a single problem instance and its alterations. To ensure consistency of results, we repeat the procedure and apply it to six different problem instances selected from those described in Sect. 3.1. The selected problem instances are (1) balloons, (2) blogger, (3) breast, (4) breast with 2 attributes only, (5) iris, and (6) iris with two attributes only. All of these are relatively small problems with up to 699 instances and up to 11 attributes. The choice is motivated by both theoretical and practical aspects. From a theoretical point of view, the procedure is based on relative comparisons and it is not supposed to be influenced by problem dimensionality. From a practical point of view, the procedure can be time-prohibitive if applied to large problems.

For each single challenge, the aim is to select one single feature that has the highest chance to detect the given challenge when measured on a new problem instance. For each challenge, the output of the procedure is composed of six sorted lists (one list per tested problem instance) of significant features. We compare these six lists and select the features that most frequently appear at the top of the lists. The selected features are (i) standard deviation of class probabilities, (ii) proportion of instances with missing values, (iii) mean class standard deviation, (iv) maximum coefficient of variation within classes, (v) mean coefficient of variation of the class attribute, (vi) minimum skewness of the class attribute, (vii) mean skewness of the class attribute, (viii) minimum normalized entropy of the attributes, (ix) maximum normalized entropy of the attributes, (x) standard deviation of the joint entropy between attributes and class attribute, (xi) skewness of the joint entropy between attributes and class attribute, (xii) standard deviation of the mutual information between attributes and class attribute, (xiii) mean concept variation, (xiv) standard deviation of the concept variation, (xv) kurtosis of the concept variation, (xvi) mean weighted distance, (xvii) standard deviation of the weighted distance, (xviii) skewness of the weighted distance.

Along with these features associated with specific challenges, we have also considered features that are frequently used in meta-learning studies. Based on a literature review over the 1992–2015 period (Aha 1992; Brazdil et al. 1994; Gama and Brazdil 1995; Michie et al. 1994; Vilalta 1999; Bensusan and Giraud-Carrier 2000; Pfahringer et al. 2000a; Peng et al. 2002; Smith et al. 2002; Castiello et al. 2005; Ali and Smith 2006; Vanschoren 2010; Reif et al. 2012, 2014; Reif and Shafait 2014; Garcia et al. 2015) we identify 21 features. The details are reported in Table 3. A short explanation regarding the meaning of the symbols used in this table can be found in Table 4. Finally, we consider complexity measures (Ho and Basu 2002) because they explicitly aim to characterize the difficulty of the classification task.

All of the identified features are combined in a single list and further processed. The aim is to identify uncorrelated features that are linearly related to algorithm performance; thus we select features having feature-to-feature correlation less than |0.7| and feature-to-performance correlation greater than |0.3|. The final set of relevant features is as follows, with further details of each feature and the correlation matrix presented in Sect. 3 of the Supplementary Material:

1. $H(\mathbf{X})'_{\max}$—maximum normalized entropy of the attributes
2. $H'_c$—normalized entropy of class attribute
3. $\overline{M}_{CX}$—mean mutual information of attributes and class
4. $DN_{ER}$—error rate of the decision node
5. $SD(\nu)$—standard deviation of the weighted distance
6. $F3$—maximum feature efficiency

**Table 3** Frequent features selected from the literature over the period 1992–2015

| Feature | Aha (1992) | Brazdil et al. (1994) | Gama and Brazdil (1995) | Michie et al. (1994) | Bensusan and Giraud-Carrier (2000) | Pfahringer et al. (2000a) | Peng et al. (2002) | Smith et al. (2002) | Castiello et al. (2005) | Ali and Smith (2006) | Vanschoren (2010) | Reif et al. (2012) | Reif et al. (2014) | Reif and Shafait (2014) | Garcia et al. (2015) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p$ | ✓ | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $q$ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $q_{\text{num}}$ | ✓ | | | | | ✓ | ✓ | | | | ✓ | ✓ | ✓ | | ✓ |
| $q_{\text{nom}}$ | ✓ | | | | | ✓ | ✓ | | | | ✓ | ✓ | ✓ | | ✓ |
| $r_{\text{nom}}$ | | ✓ | | | | | | ✓ | | ✓ | ✓ | | ✓ | | ✓ |
| $K$ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | | ✓ | | ✓ | ✓ | ✓ |
| $\overline{\gamma}_1$ | | | ✓ | | | | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $\overline{\gamma}_2$ | | | ✓ | | | | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $\overline{\rho}$ | ✓ | ✓ | ✓ | | | | | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ |
| $canCor_1$ | | ✓ | ✓ | ✓ | | | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $frac_1$ | | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ |
| $\overline{H}_{\mathbf{X}}$ | | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | ✓ | | | ✓ | ✓ |
| $H_c$ | | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | ✓ | | | ✓ | ✓ |
| $\overline{H}_{CX}$ | | | | ✓ | ✓ | | ✓ | | ✓ | | | | ✓ | ✓ | ✓ |
| $\overline{M}_{CX}$ | | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ |
| $EN_{\mathbf{X}}$ | | ✓ | | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ |
| $SNR^{-1}$ | | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ |
| $DN_{ER}$ | | | | | ✓ | | ✓ | | | | ✓ | ✓ | ✓ | ✓ | ✓ |
| $WN_{ER}$ | | | | | ✓ | | ✓ | | | | ✓ | ✓ | ✓ | ✓ | |
| $NB_{ER}$ | | | | | ✓ | | ✓ | | | | ✓ | ✓ | ✓ | ✓ | ✓ |
| $LD_{ER}$ | | | | | ✓ | | ✓ | | | | ✓ | ✓ | ✓ | ✓ | |

**Table 4** Description of the attributes used in Table 3

| Attribute | Description |
|---|---|
| $p$ | number of observations |
| $q$ | number of attributes |
| $q_{\text{num}}$ | number of numeric attributes |
| $q_{\text{nom}}$ | number of nominal attributes |
| $r_{\text{nom}}$ | proportion (ratio) of nominal attributes |
| $K$ | number of classes |
| $\overline{\gamma}_1$ | mean skewness |
| $\overline{\gamma}_2$ | mean kurtosis |
| $\overline{\rho}$ | mean absolute correlation |
| $canCor_1$ | first canonical correlation |
| $frac_1$ | fraction of the variance retained by the first principal component |
| $\overline{H}_{\mathbf{X}}$ | mean entropy of attributes |
| $H_c$ | class entropy |
| $\overline{H}_{CX}$ | mean joint entropy |
| $\overline{M}_{CX}$ | mean mutual information |
| $EN_{\mathbf{X}}$ | equivalent number of attributes |
| $SNR^{-1}$ | noise-to-signal ratio |
| $DN_{ER}$ | error rate of the decision node |
| $WN_{ER}$ | error rate of the worst node |
| $NB_{ER}$ | error rate of the Naive Bayes |
| $LD_{ER}$ | error rate of the linear discriminant |

7. $F4$—collective feature efficiency
8. $L2$—training error of linear classifier
9. $N1$—fraction of points on the class boundary
10. $N4$—nonlinearity of nearest neighbor classifier

## 5 Assessing adequacy of the feature set via performance prediction

The set of ten selected features is adequate for our purposes if it exposes the strengths and weaknesses of algorithms. To achieve this, a prerequisite is that algorithm performance can be accurately predicted based on the selected set of features. We adopt an approach based on model fitting and evaluation of model accuracy.

We fit flexible models to $(\mathbf{f}_i, y_i)$ pairs where $\mathbf{f}_i \in \mathbb{R}^m$ is a input vector comprising of the selected features and $y_i \in \mathbb{R}$ measures the algorithm performance, with $i = 1, \ldots, 235$ instances and $m = 10$ features. We consider two cases. In the first case, the output is a measure of algorithm performance, namely the error rate *ER*; it varies continuously in [0, 1]

and gives rise to a regression problem. In the second case, the output is a measure of problem difficulty that we derive from *ER*:

$$h(ER) = \begin{cases} 0 & \text{if } ER \leq 0.2 \\ 1 & \text{if } ER > 0.2; \end{cases} \tag{1}$$

which takes on labels {0, 1} (corresponding to easy and hard instances respectively) and gives rise to a binary classification problem. For both regression and classification problems, we use a Support Vector Machine (SVM) model with Gaussian Radial Basis Function (RBF) kernel $k(\mathbf{f}, \mathbf{f}') = \exp(\gamma \|\mathbf{f} - \mathbf{f}'\|^2)$. The type of SVM used is $\epsilon$-regression and $C$-classification respectively. Both $\epsilon$-regression and $C$-classification present two parameters; they are the cost $C$ in the regularization term and the RBF hyper-parameter $\gamma$. An additional parameter $\epsilon$ is used in $\epsilon$-regression to tolerate small approximation errors (Vapnik 1995). We tune $C$ and $\epsilon$ through grid search in [1, 10] and [0, 1] respectively; we estimate a good value of the RBF hyper-parameter $\gamma$ based on the 0.1 and 0.9 quantile of $\|\mathbf{f} - \mathbf{f}'\|^2$ (Caputo et al. 2002). We use tenfold cross validation to train the model and assess the model generalization ability. The cross-validated Mean Squared Error (cv-MSE) and Error Rate (cv-ER) are used as estimates of the model generalization ability in regression and classification, respectively. The described procedure relies on the R packages `e1071` (Meyer et al. 2015) and `kernlab` (Karatzoglou et al. 2004).

Table 5 reports values of SVM parameters and cross-validated error for both regression and classification studies; for the regression problem the table reports also the coefficient of determination $R^2$ as a measure of goodness-of-fit; $R^2 = (\text{cor}(\mathbf{y}, \widehat{\mathbf{y}}))^2$, where $\mathbf{y}$ and $\widehat{\mathbf{y}}$ are the observed and estimated algorithm performance. SVM models for LDA and QDA tend to present the largest errors indicating that additional features might be required to capture the challenges that instances provide for those algorithms. Overall, the small values of the cross-validated errors and the large $R^2$ values indicate that the selected features are adequate to accurately predict algorithm performance and problem difficulty, although there is always room for improvement through additional feature creation.

## 6 Creating an instance space

The final aim of the current research is to expose strengths and weaknesses of classification algorithms and provide an explanation for the good or bad performance based on features of the problem instances. The quality of the problem instances to support these insights must be evaluated. A critical step is the visualization of the instances, their features and algorithm performance in a common space, the *instance space*.

Both problem instances and features play a critical role in determining a suitable instance space. Instances must be diverse and dense enough to uniformly cover a wide degree of problem difficulty; for all algorithms there must exist both easy and hard instances, and the transition from easy to hard should be densely covered. On the other hand, features must correlate to algorithm performance, measure diverse aspects of the problem instances, and be uncorrelated with one another. The feature set should be small in size, yet it should comprehensively measure aspects of the problem instances that either challenge algorithms or make their task easy.

How we choose to project the instances from a high-dimensional feature vector to a 2-D instance space is also a critical decision that affects the usefulness of the instance space for further analysis. The ideal instance space maps the available problem instances to a 2-

**Table 5** Parameters values and performance of the SVM models approximating the functional relationship between selected features and (i) algorithm performance (regression case), (ii) problem difficulty (classification case)

| Algorithm | $\epsilon$-regression | | | | | $C$-classification | | |
|---|---|---|---|---|---|---|---|---|
| | $\gamma$ | $C$ | $\epsilon$ | cv-MSE | $R^2$ | $\gamma$ | $C$ | cv-ER |
| NB | 0.104 | 9 | 0.14 | 0.006 | 0.91 | 0.102 | 2 | 0.157 |
| LDA | 0.081 | 4 | 0.04 | 0.029 | 0.72 | 0.102 | 1 | 0.161 |
| QDA | 0.117 | 8 | 0.21 | 0.023 | 0.93 | 0.093 | 3 | 0.145 |
| CART | 0.095 | 2 | 0.17 | 0.004 | 0.91 | 0.105 | 1 | 0.099 |
| J48 | 0.090 | 4 | 0.05 | 0.003 | 0.94 | 0.099 | 2 | 0.106 |
| KNN | 0.098 | 3 | 0.00 | 0.002 | 0.97 | 0.092 | 6 | 0.073 |
| L-SVM | 0.106 | 2 | 0.07 | 0.006 | 0.85 | 0.095 | 1 | 0.086 |
| Poly-SVM | 0.098 | 3 | 0.02 | 0.006 | 0.89 | 0.089 | 5 | 0.127 |
| RBF-SVM | 0.129 | 2 | 0.13 | 0.005 | 0.90 | 0.082 | 7 | 0.085 |
| RF | 0.099 | 3 | 0.37 | 0.027 | 0.63 | 0.092 | 1 | 0.132 |

dimensional representation in such a way that both features and algorithm performance vary smoothly and predictably across the space. This exposes trends in features and algorithm performance, and helps to partition the instance space into easier and harder instances, and show how the features support those partitions. The comparison can give an instant perception of why a given algorithm performs well or poorly in a given area of the instance space. We will focus here on finding projections that result in linear trends, but the general approach can be extended to encompass more complex interplays including pair-wise interactions and non-linear relationships.

Since existing problem instances are limited in number and are not necessarily representative of a broader population of classification problems, the first instance space we generate may not be capable of the insights we seek. If we generated additional instances, our choice of features might also need to be updated to explain their performance. Clearly, the instance space generation is an iterative process that might require multiple adjustments to identify an optimal subset of features and a suitable set of instances. The steps that we have implemented to generate the instance space are:

1. Select a set of relevant features and evaluate their adequacy;
2. Generate an instance space and evaluate the adequacy of the instances;
3. If the instance space is inadequate, artificially generate new instances and return to Step 1.

With strong evidence that the relevant features accurately predict algorithm performance and problem difficulty, we now build an instance space to inspect the relationships between problem instances—their features and their difficulty for the chosen algorithms—and objectively assess algorithm performance across the broadest space of instances $\mathcal{P}$ rather than just $\mathcal{I}$. In previous work in graph coloring (Smith-Miles et al. 2014), we used Principal Component Analysis to project graph features into a 2-dimensional space. However, the PCA model was somewhat unsatisfactory to predict performance, since PCA is only concerned with maximizing variance explained in the features with no regard for projecting to show trends

in difficulty. Therefore, we reformulate the dimensionality reduction problem to consider an optimal projection for our purpose below.

### 6.1 A new interpretable projection approach

Given the feature data matrix $\mathbf{F} = [\mathbf{f}_1\ \mathbf{f}_2\ \ldots\ \mathbf{f}_n] \in \mathbb{R}^{m\times n}$ and algorithm performance vector $\mathbf{y} \in \mathbb{R}^n$, where $m$ is the number of features and $n$ is the number of problem instances, we achieve an ideal projection of the instances if we can find $\mathbf{A}_r \in \mathbb{R}^{d\times m}$, $\mathbf{B}_r \in \mathbb{R}^{m\times d}$ and $\mathbf{c}_r \in \mathbb{R}^d$ which minimizes the approximation error

$$\|\mathbf{F} - \widehat{\mathbf{F}}\|_F^2 + \|\mathbf{y}^\top - \widehat{\mathbf{y}}^\top\|_F^2 \tag{2}$$

such that

$$\mathbf{Z} = \mathbf{A}_r\mathbf{F} \tag{3}$$
$$\widehat{\mathbf{F}} = \mathbf{B}_r\mathbf{Z} \tag{4}$$
$$\widehat{\mathbf{y}}^\top = \mathbf{c}_r^\top\mathbf{Z}. \tag{5}$$

with $d = 2$ being the target dimension. Without loss of generality we assume that the feature data $\mathbf{F}$ and $\mathbf{y}$ are centered, $m < n$ 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 optimization problem

$$(\mathcal{D})\quad \begin{aligned} \min\ & \|\mathbf{F} - \mathbf{B}_r\mathbf{Z}\|_F^2 + \left\|\mathbf{y}^\top - \mathbf{c}_r^\top\mathbf{Z}\right\|_F^2 \\ \text{s.t.}\ & \mathbf{Z} = \mathbf{A}_r\mathbf{F} \\ & \mathbf{A}_r \in \mathbb{R}^{d\times m} \\ & \mathbf{B}_r \in \mathbb{R}^{m\times d} \\ & \mathbf{c}_r \in \mathbb{R}^d \end{aligned} \tag{6}$$

In Appendix A we prove the existence of a global optimum for $\mathcal{D}$, and that such optimum has infinitely many solutions. A lower bound for $\mathcal{D}$ is given by the two largest principal components of the matrix $\bar{\mathbf{F}} = \left(\mathbf{F}^\top\mathbf{y}\right)^\top$, which would correspond to the solution if $\mathbf{F}\mathbf{F}^\top$ is invertible, otherwise the solution is numerically unstable and the method provides an approximation. The results from Appendix A also hold for a matrix of performances $\mathbf{Y}$, meaning that a joint instance space can be calculated for a group of algorithms. Performance is now estimated as $\widehat{\mathbf{Y}} = \mathbf{C}_r\mathbf{Z}$, $\mathbf{C}_r \in \mathbb{R}^{\cdot\times d}$, where $\cdot$ represents the number of algorithms.

Let $\mathbf{F} \in \mathbb{R}^{10\times 235}$ be a matrix whose rows correspond to the ten relevant features from Sect. 4, and its columns correspond to the 235 UCI/KEEL/DCol instances. Each feature was transformed as follows: $F4$ was scaled to $[-0.99999, 0.99999]$ and $\tanh^{-1}$-transformed, $\left\{H_C^{'}, \overline{M}_{CX}, DN_{er}, SD(\nu), F3, L2, N1\right\}$ were root-squared. Let $\mathbf{Y} \in \mathbb{R}^{10\times 235}$ be a matrix whose rows correspond to root-squared error rate of the ten algorithms in Table 5. Both features and error rates were normalized to $\mathcal{N}(0, 1)$. Using Corollary 1 from Appendix A, we found that the lower bound is $1.7216 \times 10^3$. Furthermore, $\mathbf{F}\mathbf{F}^\top$ was found to be right-side invertible only; hence, the error when using Eq. (13) is equal to $1.8749 \times 10^3$. Error values are large, as they correspond to the sum of the error per instance, feature and algorithm. The mean error rate per instance, feature and algorithm is 0.36.

We solve numerically $(\mathcal{D})$ using BIPOP-CMA-ES, a stochastic, iterative, variable metric method with demonstrated effectiveness in middle sized optimization problems (Hansen 2009). To use this method, we represent $\{\mathbf{A}_r, \mathbf{B}_r, \mathbf{C}_r\}$ as a column vector by concatenating

the matrix columns. We run 30 times BIPOP-CMA-ES starting from random positions, using the default parameters and a maximum of $10^5$ evaluations of ($\mathcal{D}$). The average error from the runs was of $1.8658 \times 10^3$ with a standard deviation of $2.0758 \times 10^{-12}$, which can be considered within numerical precision, meaning that all runs converged to the same error. On the other hand, if $\mathbf{A}_r$ is set to be the two largest principal components of $\mathbf{F}$ as in Smith-Miles et al. (2014), the average error from 30 runs of BIPOP-CMA-ES is $2.0403 \times 10^3$ with an standard deviation of $8.0996 \times 10^{-13}$, meaning that PCA is a suboptimal solution of ($\mathcal{D}$). Given the Theorem 2 from "Appendix A", we can conclude that BIPOP-CMA-ES converged to a global optimum. The ratio between the lower bound and the numerical solution was 0.9227. To select the best solution from these thirty runs, we define a measure of topological preservation as the Pearson Correlation between the distances in the feature space, $\|\mathbf{f}_i - \mathbf{f}_j\|$, and the distances in the instance space, $\|\mathbf{z}_i - \mathbf{z}_j\|$ (Yarrow et al. 2014). The chosen transformation of instances from the 10-D feature space to the 2-D instance space, with the highest topological preservation of 0.8026, is:

$$\mathbf{Z} = \begin{bmatrix} 0.070 & 0.180 \\ 0.094 & 0.618 \\ -0.277 & -0.052 \\ 0.114 & 0.192 \\ 0.045 & -0.100 \\ -0.128 & 0.151 \\ -0.045 & 0.077 \\ 0.184 & 0.017 \\ 0.449 & 0.223 \\ 0.132 & -0.112 \end{bmatrix}^{\top} \begin{bmatrix} H(\mathbf{X})'_{\max} \\ H'_c \\ \overline{M}_{CX} \\ DN_{ER} \\ SD(\nu) \\ F3 \\ F4 \\ L2 \\ N1 \\ N4 \end{bmatrix} \tag{7}$$

### 6.2 Instance space results

Figures 2, 3 and 4 show the instance space resulting from Eq. (7). Figure 2 enables us to visualize the instances described by their features selected in Sect. 4, while Fig. 3 shows the Error Rate ($ER$) of each algorithm in Sect. 3.2 distributed across the instance space. Both the features and the error rate were scaled to [0, 1] using their maximum and minimum. The fit of the projection model given by $\{\mathbf{A}_r, \mathbf{B}_r, \mathbf{C}_r\}$ is evaluated using the coefficient of determination $R^2$, defined as in Sect. 4. Recall that our objective for projection was to achieve linearity across the instance space for each feature as well as performance of each algorithm, as much as possible simultaneously. For the features, the best fit is obtained for $N1$ $\left(R^2 = 0.910\right)$, and the worst fit for $H(\mathbf{X})'_{\max}$ $\left(R^2 = 0.116\right)$. For the error rate, the best fit is obtained for KNN $\left(R^2 = 0.805\right)$, and the worst fit for QDA $\left(R^2 = 0.367\right)$. The median $R^2$ is equal to 0.650, meaning that the projection model describes a linear trend between most features and algorithms.

If the linear fit across the instance space of a feature is poor, it is simply visualizing that the feature plays no significant part in explaining the instance space (it has a low coefficient in linear combinations that create the principal component axes). So we cannot expect to describe the location of instances in terms of such features. If there is a poor linear fit for an algorithm's performance however, this tells us that the features do not suggest a good linear relationship with algorithm performance. For some algorithms, the choice of features may be better than for others. We have chosen a common feature set that performs well on average across all algorithms, but some algorithms may benefit from their own set of features

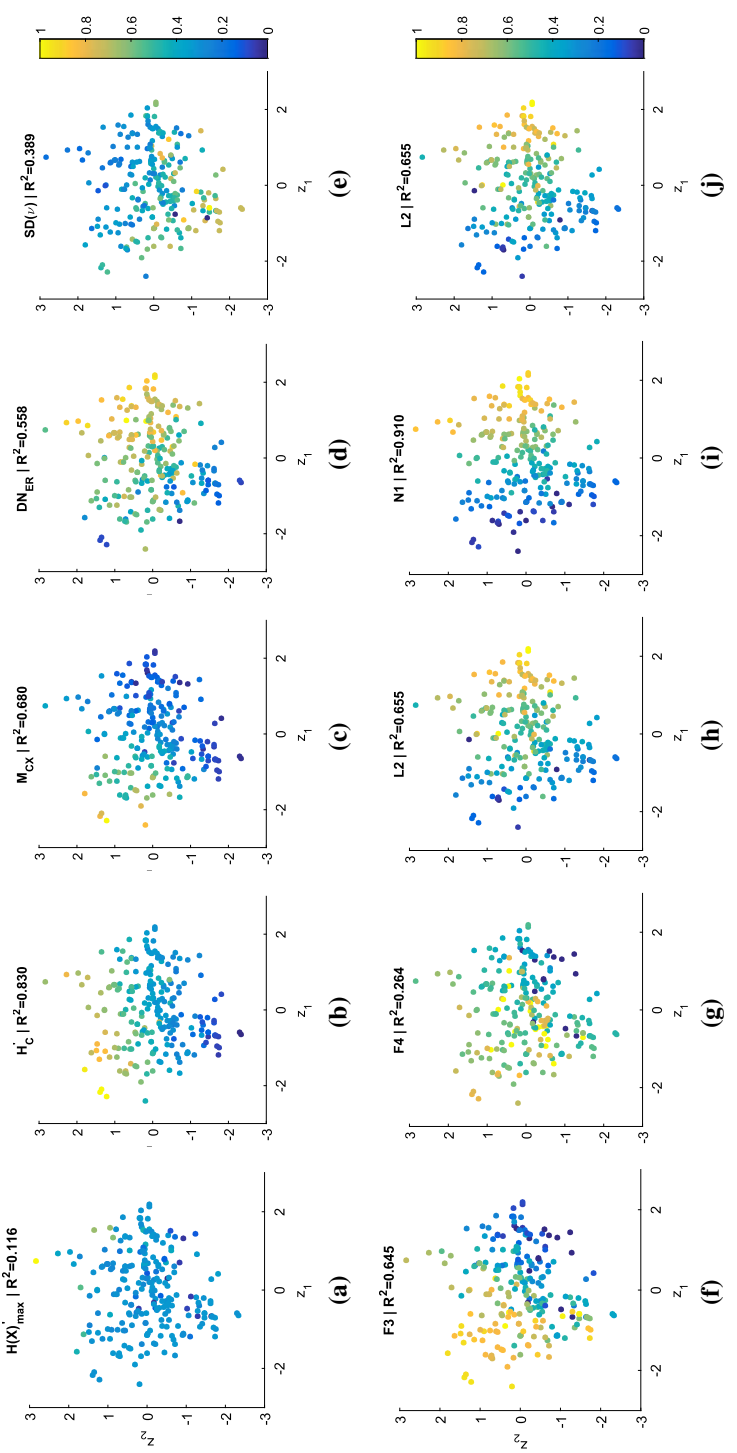


**Fig. 2** Distribution of normalized features on the projected instance space

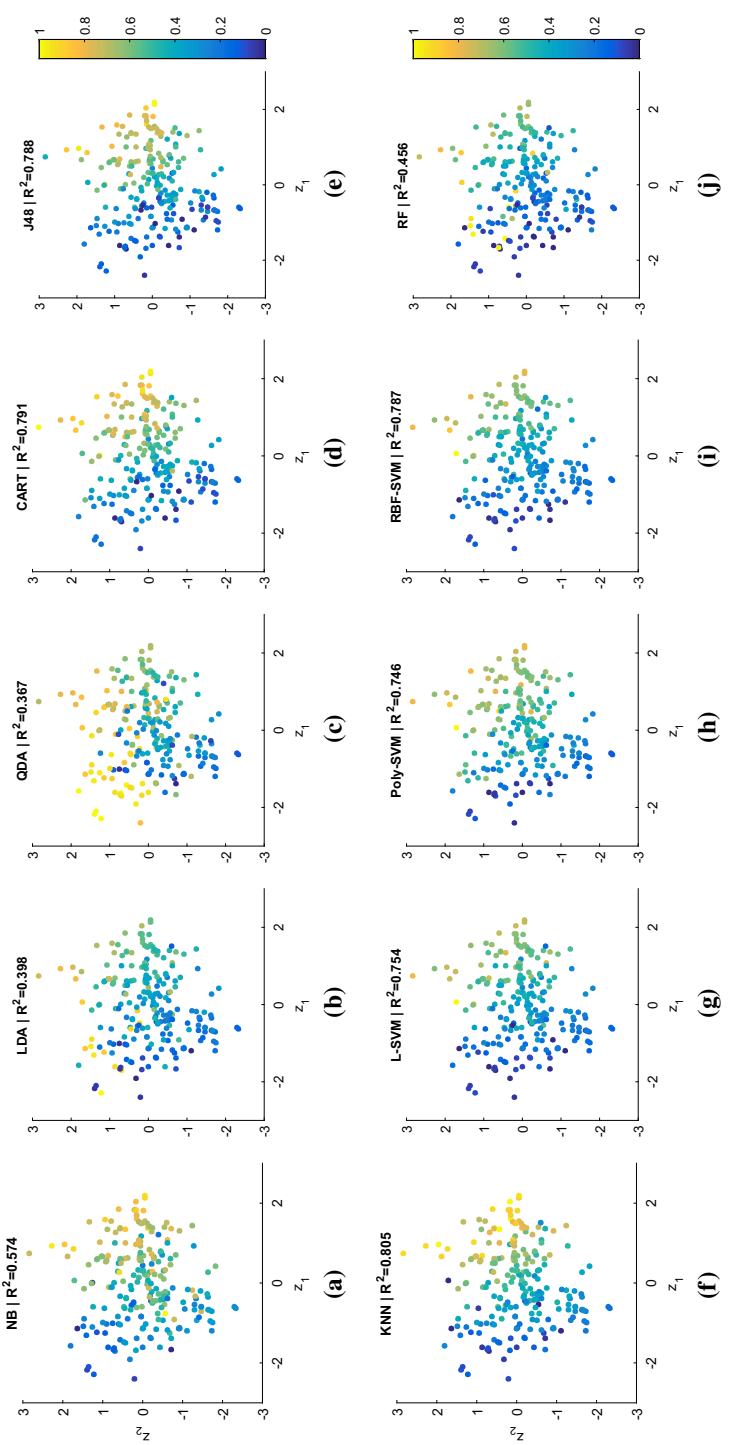


**Fig. 3** Normalized error rate of each classification algorithm on the projected instance space

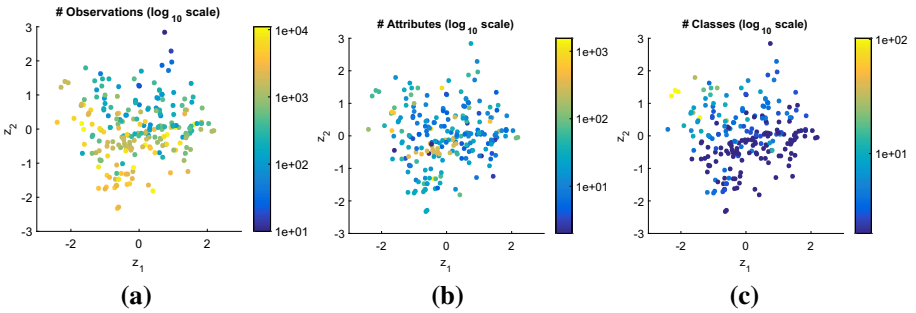


**Fig. 4** Sizes of the instances in terms of the number of observations ($p$), attributes ($q$), and classes ($K$). All have been $\log_{10}$-scaled

that explain performance. We see this mirrored also in the performance prediction results in Table 5.

Figure 4 illustrates the size of the instances by the number of (a) observations, (b) attributes, and (c) classes. We report that 2.5% of the instances have less than 50 observations, 5.9% having less than 100, and only 1.7% have more than 10,000. The majority of instances (66.1%) have between 100 and 2000 observations. In terms of attributes, 33.5% of the instances have less than ten attributes, 76.7% have less than 50, and only 2.5% have more than 500. In terms of classes, 53.8% of the instances have two classes, 14.8% have three classes. Only 5.9% of the instances have more than ten classes, with the largest being 102. In general, the selected UCI/KEEL/DCol set is skewed towards binary problems with a middle sized number of observations and attributes. It should be noted that we omitted very large datasets due to computational constraints when evaluating 509 features for the statistical study, but we don't believe this limits our conclusions except for the absence of huge big-data problems. We should still be able to understand how the features of a dataset combine to create complexity even for moderate-sized datasets.

From these figures we can conclude that, for our selected 235 instances, the number of observations per instance increases from top to bottom, while the number of classes from right to left. There is no trend emerging from the number of attributes; hence, it does not influence the performance of the algorithms as much as the number of observations or classes. Those algorithms whose $R^2$ is above 0.500 tend to find easier the instances on the bottom left side of the space, whereas the remaining algorithms tend to find easier those in the bottom center of the space. This means that most of the instances with a high number of observations and classes are relatively easier for most algorithms, with the exception of LDA and QDA. In general, high values of $H(\mathbf{X})'_{\max}$, $DN_{ER}$ and $N1$ tend to produce harder instances for most algorithms.

## 7 Objective assessment of algorithmic power

Our method for objective assessment of algorithmic power is based on the accurate estimation and characterization of each algorithm's footprint—a region in the space of all possible instances of the problem where an algorithm is expected to perform well based on inference from empirical performance analysis.

We have previously reported methods for calculation and analysis of algorithm footprints as a generalized boundary around known easy instances. For example, in Smith-Miles and

**Table 6** Footprint analysis of the algorithms using Eq. (1) and $\beta = 0.5$

| | $ER \leq 0.2$ | | | Best algorithm | | |
|---|---|---|---|---|---|---|
| | $\alpha_N$ (%) | $d_N$ (%) | $p$ (%) | $\alpha_N$ (%) | $d_N$ (%) | $p$ (%) |
| NB | 43.5 | 115.0 | 97.4 | 0.3 | 516.7 | 100.0 |
| LDA | 40.2 | 131.8 | 98.4 | 0.0 | 0.0 | 0.0 |
| QDA | 8.5 | 238.3 | 97.9 | 0.0 | 0.0 | 0.0 |
| CART | 57.6 | 114.5 | 98.7 | 5.2 | 74.9 | 77.8 |
| J48 | 63.7 | 108.9 | 98.1 | 6.5 | 178.9 | 81.5 |
| KNN | 62.4 | 109.2 | 98.1 | 0.6 | 292.6 | 100.0 |
| L-SVM | 53.1 | 125.1 | 98.7 | 5.4 | 112.4 | 85.7 |
| poly-SVM | 37.1 | 126.6 | 98.2 | 0.0 | 0.0 | 0.0 |
| RBF-SVM | 55.5 | 126.6 | 96.9 | 0.0 | 0.0 | 0.0 |
| RF | 50.4 | 129.0 | 95.4 | 15.9 | 197.3 | 75.3 |
| $\beta$-easy | | | | 52.3 | 128.5 | 98.7 |
| $\beta$-hard | | | | 19.7 | 140.2 | 90.6 |

$\alpha_N$ is the area, $d_N$ the density and $p$ the purity. The footprint areas (and their density and purity) are shown where algorithm performance is good ($ER \leq 0.2$) and best

Tan (2012) we used the area of the convex hull created by triangulating the instances where good performance was observed. The convex hull was then pruned by removing those triangles whose edges exceeded a user-defined threshold. However, there may be insufficient evidence that the remaining triangles actually represent areas of good performance. In Smith-Miles et al. (2014), unconnected triangles were generated by finding the nearest neighbors and maintaining a taboo list. The triangles were merged together if the resulting region fulfilled some density and purity requirements. However, randomization steps in the triangle construction process lead to some triangles becoming exceedingly large. In this paper, we use an improved approach (Muñoz and Smith-Miles 2017) described by the two algorithms presented in Appendix B.

Algorithm 1 constructs the footprints following these steps: (i) measuring the distances between all instances, while marking for elimination those instances with a distance lower than a threshold, $\delta$; (ii) calculating a Delaunay triangulation with the remaining instances; (iii) finding the concave hull, by removing any triangle with edges larger than another threshold, $\Delta$; (iv) calculating the density and purity of each triangle in the concave hull; and, (v) removing any triangle that does not fulfill the density and purity limits.

The parameters for Algorithm 1 are: the lower and upper distance thresholds, $\{\delta, \Delta\}$, set to 1 and 25% of the maximum distance respectively. The density threshold, $\rho$, is set to 10, and the purity threshold, $\pi$, is set to 75%. Algorithm 2 removes the contradictions that could appear when two conclusions could be drawn from the same section of $\mathcal{I}$ due to overlapping footprints, e.g., when comparing two algorithms. This is achieved by comparing the area lost by the overlapping footprints when the contradicting sections are removed, while maintaining the density and purity thresholds.

Table 6 presents the results from the analysis using Eq. (1) to define the instance hardness. The best algorithm is the one such that *ER* is the lowest for the given instance. In addition,

an instance is also defined as $\beta$-hard with $\beta = 0.5$ if 50% of the algorithms have an error rate higher than 20%. The results have been normalized over the area (11.6685) and density (19.8827) of the convex hull that encloses all instances. Further results are also illustrated in Fig. 5, which shows the instances whose error rate is below 20% as blue marks and above 20% as red marks. The footprint for QDA has a normalized area of 8.5%, making QDA the weakest algorithm in the portfolio, while J48 with an area of 63.7% could be considered the strongest. However, given the lack of diversity on the UCI/KEEL/DCol set, most algorithms fail in similar regions of the space, and we lack instances that reveal more subtle differences in performance. In fact, over half of the instance space is considered to have $\beta$-easy instances, while $\beta$-hard instances occupy only 20% approximately, for $\beta = 0.5$. Besides, large empty areas are visible. For example, a single instance is located at $\mathbf{z} = [0.744, 2.833]$, with the next closest located at $\mathbf{z} = [0.938, 2.281]$. This single instance has $ER > 20\%$ for all algorithms except J48, whereas the nearby instances have $ER > 20\%$ for all algorithms. Therefore, more instances are needed to conclude whether this region represents a strength for J48.

Figure 6 illustrates in the instance space for each instance (a) their best algorithm and (b) their $\beta$-hardness. Inspection of the best algorithm explains the results observed in Table 6 for the best algorithm, in which case LDA, QDA, poly-SVM and RBF-SVM footprints cover 0% of the instance space. This means there is no dense concentration of instances for which these algorithms are the best, although they are still competitive across a broad part of the instance space. Instead, the location of instances for which these algorithms are best are scattered throughout the space limiting our ability to construct a footprint of confidence. Of course, default parameters have been used for all algorithms, and the footprint calculation could be different with parameter tuning to allow each algorithm to maximize its footprint. We also observe less than ideal purity is present for most algorithms, due to significant overlap between footprints. Overall, the selection of the best algorithm seems to be more related to the number of attributes than any other feature. Furthermore, $\beta$-hardness of the UCI/KEEL/DCol set of instances is consistent with the conclusions drawn from Figs. 3 and 5, which show that most of these algorithms have similar performances on the problem set $\mathcal{I}$. Despite our extensive efforts to generate an instance space to provide visual insights into algorithm strengths and weaknesses—using a rigorous feature selection process and optimized projection to 2-D—we believe that the instances in UCI/KEEL/DCol are not sufficiently diverse to reveal the kinds of insights we seek.

## 8 Generation of artificial problem instances

While most of the UCI/KEEL/DCol instances are based on real-world data, the results from Sects. 6 and 7 demonstrate the limitations of this set for refined algorithm evaluation. Most instances elicit similar performance from fundamentally different algorithms, such as KNN, RBF-SVM and RF. Furthermore, there are a few areas in the instance space in which the number of instances is scarce. For example, the single instance at $\mathbf{z} = [0.744, 2.833]$, for which only J48 achieved $ER \leq 20\%$. These limitations provide an opportunity to generate new instances that (i) may produce different performance from existing algorithms, such that their strengths and weaknesses can be better understood; (ii) have features that will place them in empty areas within the space, or that help push the boundaries currently known; and (iii) represent modern challenges in machine learning classification.

Perhaps the most common way to artificially generate test instances is to select and sample an arbitrary probability distribution. However, this approach lacks control, as there is no

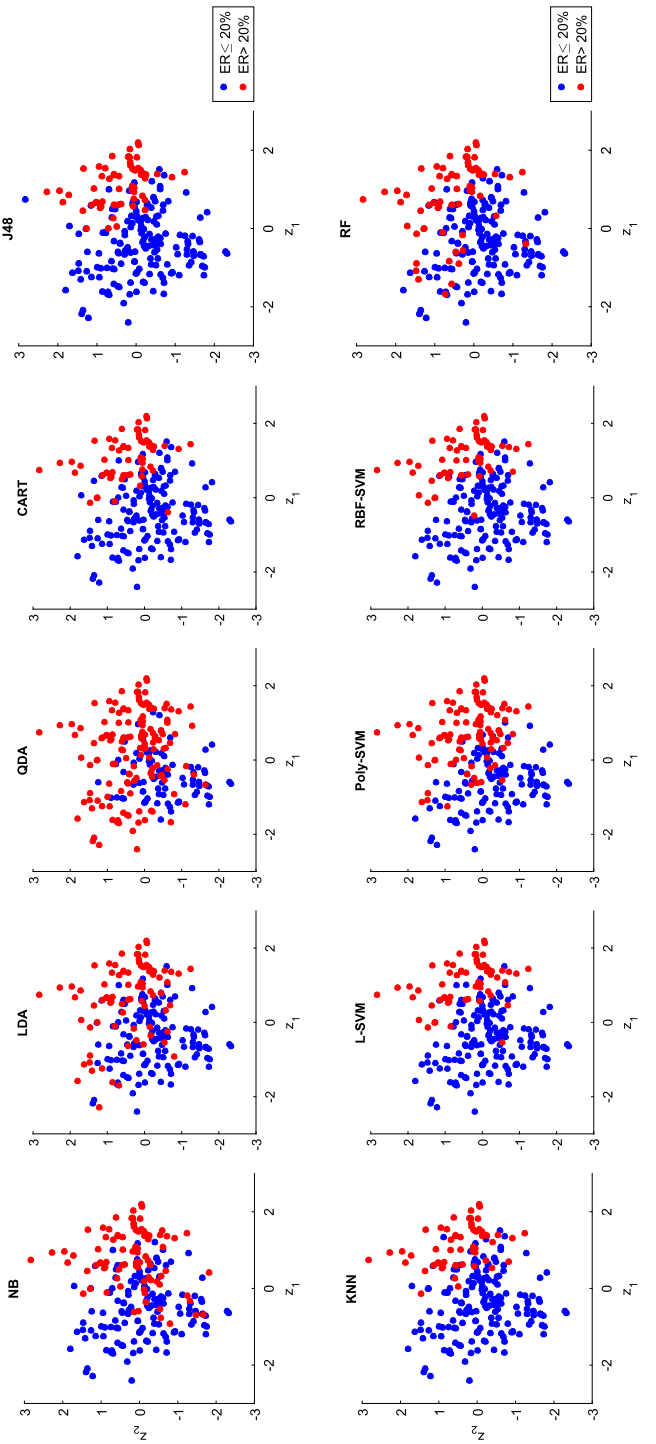


**Fig. 5** Footprints of the algorithms in the instance space

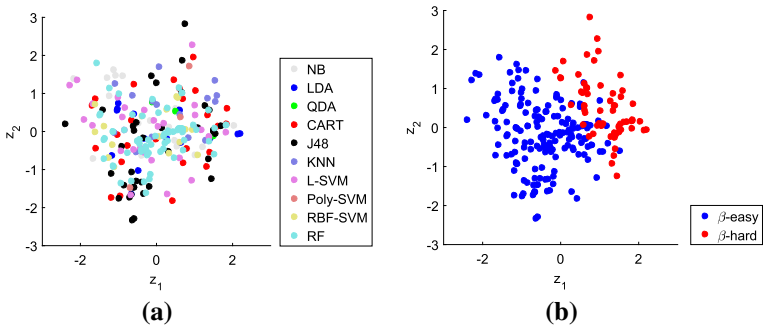


**Fig. 6** Overall performance of the algorithm portfolio, with the best algorithm for each instance shown in (**a**), while **b** shows blue marks representing $\beta$-easy instances, and red marks representing $\beta$-hard instances

guarantee that the resulting dataset will have specific features. In Macia and Bernadó-Mansilla (2014), an alternative method is proposed, in which a "seed" dataset is adjusted by evolving each observation. However, this approach resulted in very little change in the features of the dataset (merely a small magnitude perturbation), which makes it unsuitable for our task of exploring empty areas or pushing the boundaries of the instance space. Furthermore, as the number of observations increases, the evolution process becomes quickly intractable. An alternative is provided in Soares (2009), where new datasets are obtained by switching an independent attribute with the class vector. Assuming $q$ categorical attributes, it is possible to obtain $q$ new derived datasets. However, this approach is susceptible to missing target values, skewed class distributions, or difficulties when the new class is completely uncorrelated to the independent variables.

So we present in this paper a proof-of-concept for a new method to generate test instances by evolving datasets to lie at target locations in the instance space. Before describing this method though, we first consider whether selecting other datasets beyond the UCI repository, KEEL and DCoL would have given a more diverse instance space. A recent popular addition to classification dataset repositories comes from the OpenML project (Vanschoren et al. 2013). Figure 7 shows the results of projecting a set of OpenML datasets into our instance space. The blue marks represent the original UCI/KEEL/DCoL set, while the red marks are a selection of OpenML datasets of similar size to those in the UCI/KEEL/DCoL set, that is, those with less than 50 classes, 100 features and $10^4$ observations, with no missing values.[3] This resulted in 247 datasets, 49 of which are also in the UCI/KEEL/DCoL set. The figure shows that a large portion of the OpenML datasets fall within the areas currently covered by the UCI/KEEL/DCoL set. However, a number of datasets cover previously empty areas in the upper left corner of the space. This suggests that there is some additional diversity created by considering OpenML datasets within the problem sizes considered in this study. Although relaxing the size restrictions used in this example may improve the diversity, complexity should be increased without resorting to expanding the dataset size. Therefore, there is substantial scope to generate more complex datasets of similar sizes.

[3] To extract the relevant datasets, we follow the example in https://mlr-org.github.io/Benchmarking-mlr-learners-on-OpenML/, which are listed in Sect. 1 of the supplementary material.

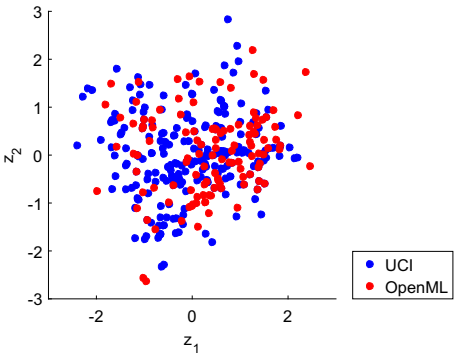


**Fig. 7** Location of 247 instances from OpenML in the constructed instance space. The blue marks represent the original UCI/KEEL/DCoL set, while the red marks are those OpenML problems with less than 50 classes, 100 features and $10^4$ observations, without missing values

### 8.1 Generating datasets by fitting Gaussian mixture models

To generate instances with a desired target vector of features, $\mathbf{f}_T$, we tune a Gaussian mixture model (GMM) until the mean squared error (MSE) between $\mathbf{f}_T$ and the feature vector of a sample from the GMM, $\mathbf{f}_S$, is zero, assuming that the GMM is sampled using a fixed seed to guarantee some level of repeatability. Let us define our GMM as having $\kappa$ components on $q$ attributes; therefore, the probability of an observation $\mathbf{x}$, being sampled from the GMM is defined as:

$$\text{pr}(\mathbf{x}) = \sum_{k=1}^{\kappa} \phi_k \mathcal{N}(\boldsymbol{\mu_k}, \Sigma_k)$$

where $\left\{\phi_k \in \mathbb{R}, \boldsymbol{\mu}_k \in \mathbb{R}^q, \Sigma_k \in \mathbb{R}^{q \times q}\right\}$ are the weight, mean vector, and covariance matrix of a $q$-variate normal distribution respectively. For simplicity, we set $\kappa$ to be a multiple of the number of labels, $K$, and $\phi_k = \kappa^{-1}$ for all components. Tuning the GMM can be thought of as a continuous black-box minimization problem; therefore, we use BIPOP-CMA-ES (Hansen 2009) as in Sect. 6.1. To use this method, we must represent the GMM parameters, $\left\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_\kappa, \Sigma_1, \ldots, \Sigma_\kappa\right\}$, as a vector $\boldsymbol{\theta}$. Since $\Sigma_k$ must be a positive semi-definite matrix, we can assume the existence of its Cholesky decomposition, i.e., an upper triangular matrix $\mathbf{A}_k$ such that $\Sigma_k = \mathbf{A}_k^\top \mathbf{A}_k$. Therefore, $\boldsymbol{\theta}$ is defined as $\left[\mu_{1,1}, \ldots, \mu_{q,1}, \ldots, \mu_{1,\kappa}, \ldots, \mu_{q,\kappa}, a_{1,1,1}, \ldots, a_{1,q,1}, a_{2,2,1}, \ldots, a_{2,q,1}, \ldots, a_{q,q,1}, \ldots, a_{1,1,\kappa}, \ldots a_{1,q,\kappa}, a_{2,2,\kappa}, \ldots, a_{2,q,\kappa}, \ldots, a_{q,q,\kappa}\right]^\top$, where $\mu_{\cdot,k}$ are the elements of the vector $\boldsymbol{\mu}_k$, and $a_{\cdot,\cdot,k}$ are the non-zero elements of $\mathbf{A}_k$. A dataset is completely defined by setting the number of observations, $p$. This approach has a number of advantages: (i) it is scalable by increasing the number of attributes, observations and classes; (ii) it allows additional flexibility, by setting the values of $\kappa$ and $\phi_k$; (iii) it enables control over the covariance between attributes, as $\sigma^2_{\cdot,\cdot,k}$ can be set permanently to a constant value, even zero, which also has the advantage of reducing the length of $\boldsymbol{\theta}$; (iv) it produces immediately a model of the data distribution, which is a solution to the classification and clustering problem; and (v) the optimization problem is unconstrained. Nevertheless, this approach does have some limitations: (i) a GMM produces datasets whose attributes are Gaussian distributed real values, eliminating the possibility of more complex variable types, such as categorical; (ii) the fitting problem is known to have local optima; and (iii) it can be can be computationally expensive for very large datasets, or inaccurate for very small ones.

Nevertheless, to test this proposed generation approach, we define two experiment types. The first one is aimed at validation, where we create datasets whose features mimic those of the

well known Iris dataset. Given that the instances can be described in the high dimensional feature space or in its 2-dimensional projection, two experiments of this type in total are carried out. The purpose of this experiment is to test whether the generation mechanism can converge to a set of target features. Furthermore, this experiment provides additional evidence of the instance space being a good representation, by confirming that a dataset with similar features produces similar response from the algorithms. For simplicity, we fix the dataset size to $p = 150$, $q = 4$, $K = 3$ and $\kappa = 3K$, and carry out ten repetitions with a soft bound of $10^4$ function evaluations, i.e., the number of times a GMM is tested. Under these conditions, $\boldsymbol{\theta}$ has a length of 84. The values of $\boldsymbol{\theta}$ are sampled at random from a uniform distribution between $[-10, 10]$.

For the second experiment, we aim to generate instances located elsewhere in the instance space, with target feature vectors determined by a latin hyper-cube sample (LHS) in the 2-D instance space, with bounds determined by the largest and smallest values for $\mathbf{Z}$. Again we use Iris as a template problem, i.e., $p = 150$, $q = 4$, $K = 3$, but we try to evolve the dataset so that its features match a different target vector. We should note that fixing the size limits our ability to achieve MSE=0, due to the relationships observed in Fig. 4 between $\{p, q, K\}$ and the instance location. However, this experiment will give us an indication of the location bounds of Iris-sized problems in the space and their complexity. We set the value of $\kappa = 3K$, and select the values of $\boldsymbol{\theta}$ at random from a uniform distribution between $[-10, 10]$. Under these conditions, $\boldsymbol{\theta}$ has a length of 126. We carry out ten repetitions with a soft bound of $10^4$ function evaluations.

### 8.2 Results

The results of the first experiment are presented in Table 7, as the *ER* of the test algorithms, with the target defined in either the high-dimensional feature space (H) or the 2-D instance space (L), $e_t$ denotes the MSE to target per dimension, and $\rho_{e,p}$ is the Pearson correlation between $e_t$ and the error rate of an algorithm. The generated instances were sorted from the lowest to the highest $e_t$. In boldface are those instances whose difference in *ER* to Iris is higher than the average difference in *ER*, which is presented in parenthesis below AVG. The table shows that as $e_t$ increases, the difference in *ER* to Iris increases, as demonstrated by $\rho_{e,p}$, with the exception of KNN, and to a lesser extent to CART. On average, the performance of the generated instances differs by 3.0% compared to Iris. The location of the Iris dataset and the newly generated Iris-like datasets in the instance space are shown in Fig. 8a and confirm that the new instances indeed have similar features to Iris. These results demonstrate that our generation approach can produce instances with controlled feature values—like Iris features in this case—and those new instances elicit similar performance from the algorithms.

Figure 8b shows the results for the second experiment. Grey marks represent the UCI/KEEL/DCol problems, blue marks represent the LHS targets, green marks represent the starting location, red marks represent the stopping location, and the black mark represents the Iris problem. The black lines represent the trajectory of the evolution process, while the red line represents the boundaries of the instance space considering the largest and smallest observed values of $\mathbf{X}$, and the correlation between features.[4] These results demonstrate that by randomly initializing an Iris-sized dataset and setting targets at different locations in the instance space we can generate datasets that are located away from Iris, and have different features despite having the same number of observations, attributes and classes. The evolution process converges towards distant targets, although not all targets were reachable within

[4] Available in the supplementary material.

**Table 7** Error rate of the test algorithms over the Iris-matching instances, with the target defined in the feature space (H) or its 2-D projection (L)

| | $e_t$ | NB | LDA | QDA | CART | J48 | KNN | L-SVM | Poly-SVM | RBF-SVM | RandF |
|---|---|---|---|---|---|---|---|---|---|---|---|
| H | 0.015 | 2.5 | 4.0 | 2.6 | **8.6** | 5.1 | 3.0 | 3.0 | 6.1 | 2.5 | **1.6** |
| | 0.017 | 4.1 | **9.1** | 3.9 | 5.8 | 6.8 | 2.8 | 5.3 | 6.3 | 3.1 | **4.9** |
| | 0.021 | 3.5 | 3.9 | 3.9 | **9.8** | **8.8** | 3.2 | 3.9 | 8.4 | 2.2 | 4.1 |
| | 0.029 | 4.0 | 6.5 | **4.5** | 6.3 | 6.1 | **1.9** | 6.5 | 12.0 | 3.3 | **4.4** |
| | 0.032 | 5.2 | 4.9 | 3.7 | 2.1 | 1.9 | 2.7 | 3.5 | 6.6 | **3.7** | 2.0 |
| | 0.033 | 5.2 | 4.7 | 4.2 | **9.3** | 6.3 | 3.6 | 5.2 | 8.7 | 3.1 | 3.6 |
| | 0.034 | 5.0 | 5.3 | 3.8 | 7.5 | **7.9** | 2.5 | 1.2 | 9.6 | 1.3 | **5.3** |
| | 0.047 | 5.6 | 7.8 | 3.0 | **9.3** | **7.9** | **1.5** | **7.6** | 12.0 | **4.4** | **5.2** |
| | 0.067 | 5.1 | 7.8 | **5.6** | 8.0 | **7.8** | 4.0 | **6.9** | 11.9 | 3.0 | **4.7** |
| | 0.139 | 6.2 | **13.5** | 4.2 | **8.9** | 5.5 | 4.8 | **13.5** | **17.3** | **3.8** | 4.2 |
| L | 0.000 | 3.1 | **10.8** | 0.5 | 4.8 | 4.3 | 3.2 | 6.3 | 10.2 | **3.7** | 2.2 |
| | 0.000 | 1.6 | 4.5 | 2.2 | 3.6 | 3.2 | **1.8** | 1.3 | 5.5 | 1.2 | 2.1 |
| | 0.000 | 1.2 | 3.0 | 1.2 | 0.9 | 0.9 | **2.0** | 0.7 | 3.9 | 1.5 | 0.9 |
| | 0.000 | 5.5 | 4.8 | 2.0 | 13.3 | 6.0 | **1.5** | 2.0 | 11.2 | 1.5 | **5.0** |
| | 0.030 | 3.9 | 5.6 | 3.7 | 6.7 | 6.6 | **2.1** | 3.7 | 11.6 | **0.8** | **2.5** |
| | 0.040 | **6.9** | 7.1 | **6.7** | 7.3 | **7.2** | **0.7** | 6.5 | **15.8** | 2.9 | 6.0 |
| | 0.100 | **12.5** | **14.6** | 3.7 | **10.4** | **9.6** | **2.2** | **14.6** | **18.3** | 2.7 | **4.2** |
| | 0.100 | **11.9** | **18.2** | **5.3** | **12.3** | 5.1 | **0.9** | **10.7** | **19.7** | **3.5** | 2.9 |
| | 0.160 | **11.3** | **15.3** | **9.2** | **10.4** | **12.8** | 4.4 | **12.2** | **25.9** | **5.7** | **8.6** |
| | 0.230 | **12.1** | **8.3** | **6.7** | **10.2** | **10.6** | 3.3 | **9.5** | **16.2** | **5.3** | **9.2** |
| AVG | 0.031 | 4.6 | 6.7 | 3.9 | 7.6 | 6.4 | 3.0 | 5.7 | 9.9 | 3.1 | 4.0 |
| | | (3.1) | (6.6) | (2.4) | (4.3) | (3.1) | (1.5) | (4.1) | (6.3) | (1.2) | (1.8) |
| Target | | 3.1 | 1.3 | 1.8 | 4.0 | 4.0 | 4.0 | 2.7 | 5.8 | 2.2 | 3.1 |
| $\rho_{e,p}$ | | 0.821 | 0.609 | 0.718 | 0.461 | 0.624 | 0.401 | 0.762 | 0.765 | 0.708 | 0.708 |

The symbol $e_t$ denotes the average squared error per dimension, AVG denotes the average per column or row, and $\rho_{e,p}$ is the Pearson correlation between $e_t$ and the error rate
In boldface are those instances whose difference in $ER$ to Iris is higher than the average difference in $ER$, which is presented in parenthesis below AVG
Instances are sorted from the lowest to the highest $e_t$

a reasonable computational time through our generation mechanism for Iris-sized problems. At this point, we do not know whether this is due to the natural boundary that Iris-sized problems can feasibly occupy in the instance space given their range of features and correlations, or if they could be forced to continue with a larger computational budget. More theoretical

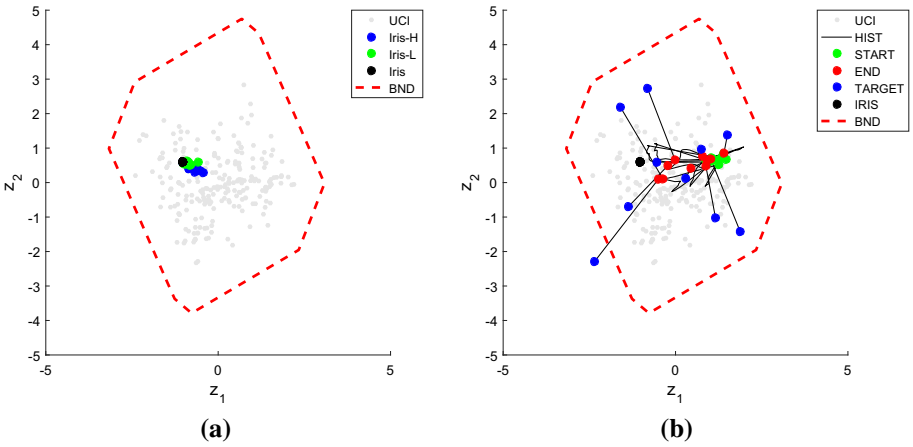


**Fig. 8** Instance space showing Iris dataset (black) and attempts to generate new datasets that are **a** similar to Iris, **b** Iris-sized instances located elsewhere (red) based on target features (blue)

work is needed to establish the instance space boundary for different sized problems, but the potential of the method for generating new test instances has been demonstrated.

## 9 Conclusions

This paper addresses the issue of objective performance evaluation of machine learning classifiers, and examines the criticality of test instances to support conclusions. Where we find that well-studied test instances are inadequate to evaluate the strengths and weaknesses of algorithms, we must seek methods to generate new instances that will provide the necessary insights. A comprehensive methodology has been developed to enable the quality of the UCI repository and other test instances used by most machine learning researchers to be assessed, and for new classification datasets to be generated with controllable features. The creation of a classification instance space is central to this methodology, and enables us to visualize classification datasets as points in a two-dimensional space, after suitable projection from a higher-dimensional feature space. In this paper we have proposed a new dimension reduction technique ideally suited to our visualization objective, one that maximizes our ability to visualize trends and correlations in instance features and algorithm performances across the instance space. The visualization reveals pockets of hard and easy instances, shows the (lack of) diversity of the set of instances, and enables an objective measure of the relative performance of each algorithm—its footprint in the instance space where the algorithm is expected to do well. Quantitative metrics, such as the area of the footprint, provide objective measures of the relative power and robustness of an algorithm across the broadest range of test instances.

The results presented in this paper demonstrate the lack of diversity of the benchmark instances, as most algorithms had similar footprints, suggesting that either the algorithms are all essentially the same (at least with default parameter settings), that the instances are not revealing the unique strengths and weaknesses of each algorithm as much as is desired, or that the features may not be discriminant enough. For this last case, it is also possible that totally new features are required in order to describe the performance of some algorithms

more effectively. Furthermore, there is significant bias on the sizes and types of problems in the repository. Therefore, we proposed a method to generate new test instances, aiming to enrich the repository's diversity. Our method modifies a template probability distribution until the features of a sample match a target, which can represent either an existing dataset, or an arbitrary point in the space.

In addition to the contributions made in this paper to developing new methodologies—instance generation and dimensional reduction—to support our broader objectives, this paper has also contributed to the meta-learning literature through its comprehensive examination of a collection of 509 features, to determine which ones can identify the presence of conditions that challenge classification algorithms. The feature set was reduced to the ten most statistically significant features after analyzing the correlation between features and a measure of algorithm performance. However, it should be noted that our final feature set is based on our current data, the selected UCI/KEEL/DCol datasets; hence, the selected features may change with a larger repository.

### 9.1 Future research

While there are theoretical and computational issues that limit our ability to extensively explore and fill the gaps in the instance space at this time—e.g., the lack of precise theoretical bounds of the instance space—our method shows significant promise. Further research on this topic will develop theoretical upper and lower bounds on the features and their dependencies to enable the theoretical boundary of the instance space to be defined more tightly than the one shown in Fig. 8. We will also continue to examine the most efficient representation of a dataset to ensure scalability and enable the instance space to be filled with new instances of arbitrary size. Of course, once we have succeeded in generating a large number of new test instances, we will need to verify that they are more useful for meta-learning, not just that they have different features and live in unique parts of the instance space.

The methodology developed in this paper is an iterative one and will need to be repeated as we accumulate more datasets with a diversity of features that really challenge algorithms. In fact, the OpenML repository (Vanschoren et al. 2013) provides an excellent collection of meta-data and additional features and algorithms from OpenML can be considered. Sampling landmarking provides relevant meta-features to further extend our current feature set, whereas meta-learning techniques such as those proposed by Lee and Giraud-Carrier (2013) provide valuable resources to select a more comprehensive set of algorithms. New features may need to be selected from the extended set of meta-features to explain the challenges of new instances, and the statistical methodology we have presented can be applied again, perhaps with even more altered datasets than used in this study. Conquering the computational challenges exposed by this proof-of-concept study, and repeating the methodology on the broadest set of instances—to determine the features that best explain the performance of different portfolios of algorithms, and creating the definitive instance space—will enable insights into the strengths and weaknesses of machine learning classifiers to be revealed.

**Acknowledgements** This work is funded by the Australian Research Council through Australian Laureate Fellowship FL140100012. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

## Appendix A: Projection methodology

We consider the optimal solution to the projection problem:

$$(\mathcal{D}) \quad \begin{aligned} \min \quad & \|\mathbf{F} - \mathbf{B}_r\mathbf{Z}\|_F^2 + \left\|\mathbf{y}^\top - \mathbf{c}_r^\top\mathbf{Z}\right\|_F^2 \\ \text{s.t.} \quad & \mathbf{Z} = \mathbf{A}_r\mathbf{F} \\ & \mathbf{A}_r \in \mathbb{R}^{d\times m} \\ & \mathbf{B}_r \in \mathbb{R}^{m\times d} \\ & \mathbf{c}_r \in \mathbb{R}^d \end{aligned} \tag{8}$$

**Theorem 1** ($\mathcal{D}$) *has at least one global minimum.*

*Proof* The problem can be presented as minimization of coercive function

$$f(\mathbf{A}_r, \mathbf{B}_r, \mathbf{c}_r) = \|\mathbf{F} - \mathbf{B}_r\mathbf{A}_r\mathbf{F}\|_F^2 + \|\mathbf{y}^\top - \mathbf{c}_r^\top\mathbf{A}_r\mathbf{F}\|_F^2.$$

Thus, ($\mathcal{D}$) must have at least one global minimum. □

**Theorem 2** ($\mathcal{D}$) *has infinitely many optimal solutions.*

*Proof* If we neglect the constraint (8) and treat $\mathbf{Z}$ as an independent variable then we get the following relaxation of the problem ($\mathcal{D}$)

$$(\mathcal{R}) \quad \begin{aligned} \min \quad & \|\mathbf{F} - \mathbf{B}_r\mathbf{Z}\|_F^2 + \left\|\mathbf{y}^\top - \mathbf{c}_r^\top\mathbf{Z}\right\|_F^2 \\ \text{s.t.} \quad & \mathbf{B}_r \in \mathbb{R}^{m\times d} \\ & \mathbf{c}_r \in \mathbb{R}^d \\ & \mathbf{Z} \in \mathbb{R}^{d\times n} \end{aligned}$$

which can be rewritten as

$$(\mathcal{R}') \quad \begin{aligned} \min \quad & \|\bar{\mathbf{F}} - \mathbf{V}\mathbf{Z}\|_F^2 \\ \text{s.t.} \quad & \mathbf{Z} \in \mathbb{R}^{d\times n} \\ & \mathbf{V} \in \mathbb{R}^{(m+1)\times d} \end{aligned}$$

where

$$\bar{\mathbf{F}} = \begin{pmatrix} \mathbf{F} \\ \mathbf{y}^\top \end{pmatrix} \quad \text{and} \quad \mathbf{V} = \begin{pmatrix} \mathbf{B}_r \\ \mathbf{c}_r^\top \end{pmatrix} \in \mathbb{R}^{(m+1)\times d} \tag{9}$$

From any feasible solution $(\bar{\mathbf{V}}, \bar{\mathbf{Z}})$ of ($\mathcal{R}'$) we can construct a feasible solution of ($\mathcal{D}$)

$$\begin{pmatrix} \bar{\mathbf{B}}_r \\ \bar{\mathbf{c}}_r^\top \end{pmatrix} = \bar{\mathbf{V}} \tag{10}$$

$$\bar{\mathbf{A}}_r = \bar{\mathbf{Z}}\mathbf{F}^\top(\mathbf{F}\mathbf{F}^\top)^{-1} \tag{11}$$

with the same objective value. In other words, the relaxed problem ($\mathcal{R}'$) provides an exact lower bound to ($\mathcal{D}$). Moreover, from any optimal solution to ($\mathcal{R}'$) we can construct an optimal solution to ($\mathcal{D}$)—see Corollaries 1 and 2 below.

From PCA, we know that ($\mathcal{R}'$) has infinitely many alternative solutions. Consequently, ($\mathcal{D}$) has infinitely many alternative solutions. □

**Corollary 1** *An optimal solution to* $(\mathcal{D})$ *can be constructed from eigenvectors of* $\bar{\mathbf{F}}\bar{\mathbf{F}}^\top$. *Precisely,*

$$\begin{pmatrix} \tilde{\mathbf{B}}_r \\ \tilde{\mathbf{c}}_r^\top \end{pmatrix} = \tilde{\mathbf{V}} \tag{12}$$

$$\tilde{\mathbf{A}}_r = \tilde{\mathbf{V}}^\top \begin{pmatrix} \mathbf{F} \\ \mathbf{y}^\top \end{pmatrix} \mathbf{F}^\top (\mathbf{F}\mathbf{F}^\top)^{-1} \tag{13}$$

*where columns of* $\tilde{\mathbf{V}}$ *are the d eigenvectors of* $\bar{\mathbf{F}}\bar{\mathbf{F}}^\top$ *corresponding to the d largest eigenvalues.*

*Proof* Immediate from PCA that eigenvectors of $\bar{\mathbf{F}}\bar{\mathbf{F}}^\top$ provide an optimal solution to $(\mathcal{R}')$ where columns of $\tilde{\mathbf{V}}$ are the $d$ eigenvectors of $\bar{\mathbf{F}}\bar{\mathbf{F}}^\top$ corresponding to the $d$ largest eigenvalues and

$$\tilde{\mathbf{Z}} = \tilde{\mathbf{V}}^\top \bar{\mathbf{F}}. \tag{14}$$

□

Note that the matrix $\tilde{\mathbf{V}}$ obtained from eigenvectors has orthonormal columns.

Due to the rank factorization, the problem $(\mathcal{D})$ is equivalent to the following problem

$$(\mathcal{D}') \qquad \begin{aligned} \min \quad & \|\mathbf{F} - \mathbf{R}\mathbf{F}\|_F^2 + \|\mathbf{y}^\top - \mathbf{s}^\top \mathbf{F}\|_F^2 \\ \text{s.t.} \quad & rank \begin{pmatrix} \mathbf{R} \\ \mathbf{s}^\top \end{pmatrix} = d \\ & \mathbf{R} \in \mathbb{R}^{m \times m} \\ & \mathbf{s} \in \mathbb{R}^m \end{aligned} \tag{15}$$

**Corollary 2** *Let* $(\tilde{\mathbf{R}}, \tilde{\mathbf{s}})$ *be an optimal solution to* $(\mathcal{D}')$. *Then the system*

$$\begin{cases} \tilde{\mathbf{R}} = \mathbf{B}_r \mathbf{A}_r \\ \tilde{\mathbf{s}}^\top = \mathbf{c}_r^\top \mathbf{A}_r \end{cases} \tag{16}$$

*is feasible and any solution of the system is an optimal solution to* $(\mathcal{D})$.

*Proof* Immediate from rank factorization. □

Corollary 1 provides a lower bound to the dimensionality reduction problem, which numerically holds iif $\mathbf{F}\mathbf{F}^\top$ is invertible, otherwise the solution is numerically unstable. Otherwise, $\tilde{\mathbf{V}}^\top \bar{\mathbf{F}} \neq \tilde{\mathbf{A}}\mathbf{F}$ meaning that the solution is unstable; hence, this analytic method is potentially an approximation in the presence of numerical instability issues.

## Appendix B: Algorithms for footprint analysis

**Input**: A set of instances in the instance space, $\mathbf{X}$, with their performance labels, $\mathbf{y}$, and the purity, $\pi$, density, $\rho$, and distance, $\{\delta, \Delta\}$, thresholds.
**Output**: An algorithm footprint composed of a triangulation, $\mathbf{T}$, and a set of areas for each triangle $\mathbf{t} \in \mathbf{T}$, $\mathbf{a}$.

```
// Generate a candidate list for triangulation
```
$\mathbf{X}_{\text{GOOD}} = \{\mathbf{x}_i : \mathbf{x}_i \in \mathbf{X}, y_i = \text{GOOD}\}$, $N = |\mathbf{X}_{\text{GOOD}}|$, $\mathbf{e} = \{\text{FALSE}\}^N$, $\mathbf{a} = \{\emptyset\}$;
**for** $i \leftarrow 1$ **to** $N$ **do** // Remove a candidate if $\|\mathbf{x}_i - \mathbf{x}_j\| < \delta$
  **for** $j \leftarrow 1$ **to** $N$ **do**
    **if** $\|\mathbf{x}_i - \mathbf{x}_j\| < \delta$ **then**
      **if** $e_i = \text{FALSE}$ **then** $e_j = \text{TRUE}$ **else** $e_j = \text{FALSE}$;
    **end**
  **end**
**end**
$\mathbf{X}_{\text{TRI}} \leftarrow \{\mathbf{x}_i : \mathbf{x}_i \in \mathbf{X}_{\text{GOOD}}, e_i = \text{FALSE}\}$;
```
// Calculate the concave hull
```
$\mathbf{T} \leftarrow$ `Delaunay` $(\mathbf{X}_{\text{TRI}})$; // Find the Delaunay triangulation for $\mathbf{X}_{\text{GOOD}}$
**for** $i \leftarrow 1$ **to** $|\mathbf{T}|$ **do** // Remove any triangle with a side greater than $\Delta$
  **if** $\|\mathbf{x}_a - \mathbf{x}_b\| > \Delta \vee \|\mathbf{x}_b - \mathbf{x}_c\| > \Delta \vee \|\mathbf{x}_c - \mathbf{x}_a\| > \Delta$ **then** $\mathbf{T} \leftarrow \mathbf{T} - \{\mathbf{t}_i\}$
**end**
**for** $i \leftarrow 1$ **to** $|\mathbf{T}|$ **do** // Calculate the purity, density and area of each triangle
  $a_i =$ `AreaOfTriangle` $(\mathbf{t}_i)$;
  $d_i =$ `InstancesInTriangle` $(\mathbf{t}_i, \mathbf{X})/a_i$;
  $p_i =$ `InstancesInTriangle` $(\mathbf{t}_i, \mathbf{X}_{\text{GOOD}})$/`InstancesInTriangle` $(\mathbf{t}_i, \mathbf{X})$;
  ```
  // Remove the triangle if it does not meet the density and purity
     requirements
  ```
  **if** $d_i < \rho \vee p_i < \pi$ **then** $\mathbf{T} \leftarrow \mathbf{T} - \{\mathbf{t}_i\}$ **else** $\mathbf{a} \leftarrow \mathbf{a} + a_i$;
**end**

**Algorithm 1:** Calculation of an algorithm footprint using concave hulls with minimum density and purity requirements. A triangle $\mathbf{t}$ is defined by a set of vertices $\{\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c\} \in \mathbf{X}_{\text{TRI}}$.

**Input**: A base and test footprints, $\{\mathbf{T}_{\text{base}}, \mathbf{a}_{\text{base}}\}$ and $\{\mathbf{T}_{\text{test}}, \mathbf{a}_{\text{test}}\}$.
**Output**: A recalculated base footprint.
$\boldsymbol{\alpha}_{|\mathbf{T}_{\text{base}}| \times 1} = \mathbf{0}$; // Area of contradiction
```
// Determine if a base triangle is intercepted and measure its area of
   contradiction
```
**for** $i \leftarrow 1$ **to** $|\mathbf{T}_{base}|$ **do**
  **for** $j \leftarrow 1$ **to** $|\mathbf{T}_{test}|$ **do**
    **if** `PolygonIntersection` $(\mathbf{t}_{\text{base},i}, \mathbf{t}_{\text{test},j})$ **is** TRUE **then** $\alpha_i = \alpha_i + a_{\text{test},j}$ ;
  **end**
**end**
```
// Remove a base triangle if its area of contradiction is larger than its area
```
**for** $i \leftarrow 1$ **to** $|\mathbf{T}_{base}|$ **do**
  **if** $\alpha_i > a_{base,i}$ **then** $\mathbf{T}_{\text{base}} \leftarrow \mathbf{T}_{\text{base}} - \{\mathbf{t}_{\text{base},i}\}$, $\mathbf{a}_{\text{base}} \leftarrow \mathbf{a}_{\text{base}} - \{a_{\text{base},i}\}$
**end**

**Algorithm 2:** Footprint contradiction detection using polygon intersection. The algorithm tests whether a base and a test footprint contradict each other. Then, it removes the contradicting sections on the base footprint depending on the size of the area of contradiction, i.e., the area lost by the test footprint when the contradicting triangles are removed.

## References


Aha, D. W. (1992). Generalizing from case studies: A case study. In *Proceedings of the 9th international conference on machine learning* (pp. 1–10).

Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., et al. (2010). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, *17*(2–3), 255–287.

Ali, S., & Smith, K. A. (2006). On learning algorithm selection for classification. *Applied Soft Computing*, *6*(2), 119–138.

Balte, A., Pise, N., & Kulkarni, P. (2014). Meta-learning with landmarking: A survey. *International Journal of Computer Applications*, *105*(8), 47–51.

Bensusan, H., & Giraud-Carrier, C. (2000). Discovering task neighbourhoods through landmark learning performances. In D. A. Zighed, J. Komorowski, & J. Żytkow (Eds.), *Principles of data mining and knowledge discovery: 4th European conference*, PKDD 2000 Lyon, France, September 13–16, 2000 Proceedings (pp. 325–330). Berlin, Heidelberg: Springer.

Brazdil, P., Carrier, C. G., Soares, C., & Vilalta, R. (2008). *Metalearning: Applications to data mining*. Berlin: Springer Science & Business Media.

Brazdil, P., Gama, J., & Henery, B. (1994). Characterizing the applicability of classification algorithms using meta-level learning. In *Machine learning: ECML-94* (pp. 83–102). Springer.

Burton, S. H., Morris, R. G., Giraud-Carrier, C. G., West, J. H., & Thackeray, R. (2014). Mining useful association rules from questionnaire data. *Intelligent Data Analysis*, *18*(3), 479–494.

Caputo, B., Sim, K., Furesjo, F., & Smola, A. (2002). Appearance-based object recognition using SVMS: Which kernel should I use? In: *Proceedings of NIPS workshop on statistical methods for computational experiments in visual processing and computer vision*, Whistler (Vol. 2002).

Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (1983). An overview of machine learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (pp. 3–23). Berlin, Heidelberg: Springer.

Castiello, C., Castellano, G., & Fanelli, A. M. (2005). Meta-data: Characterization of input features for meta-learning. In V. Torra, Y. Narukawa, & S. Miyamoto (Eds.), *Modeling decisions for artificial intelligence: Second international conference*, MDAI 2005, Tsukuba, Japan, July 25–27, 2005 Proceedings (pp. 457–468). Berlin, Heidelberg: Springer.

Cohen, J. (1992). Statistical power analysis. *Current Directions in Psychological Science*, *1*(3), 98–101.

Culberson, J. C. (1998). On the futility of blind search: An algorithmic view of "no free lunch". *Evolutionary Computation*, *6*(2), 109–127.

Fayyad, U. M., & Irani, K. B. (1992). On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, *8*(1), 87–102.

Flach, P. (2012). *Machine learning: The art and science of algorithms that make sense of data*. Cambridge: Cambridge University Press.

Fujikawa, Y., & Ho, T. (2002). Cluster-based algorithms for dealing with missing values. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 549–554). Springer

Fürnkranz, J., & Petrak, J. (2001). An evaluation of landmarking variants. In *Working notes of the ECML/PKDD 2000 workshop on integrating aspects of data mining, decision support and meta-learning* (pp. 57–68).

Gama, J., & Brazdil, P. (1995). Characterization of classification algorithms. In C. Pinto-Ferreira & N. J. Mamede (Eds.), *Progress in artificial intelligence: 7th Portuguese conference on artificial intelligence*, EPIA '95 Funchal, Madeira Island, Portugal, October 3–6, 1995 Proceedings (pp. 189–200). Berlin, Heidelberg: Springer.

Ganganwar, V. (2012). An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, *2*(4), 42–47.

Garcia, L. P., de Carvalho, A. C., & Lorena, A. C. (2015). Noise detection in the meta-learning level. *Neurocomputing*, *176*, 14–25.

Goethals, B., & Zaki, M. J. (2004). Advances in frequent itemset mining implementations: Report on FIMI'03. *ACM SIGKDD Explorations Newsletter*, *6*(1), 109–117.

Hansen, N. (2009). Benchmarking a bi-population CMA-ES on the BBOB-2009 function testbed. In *GECCO '09* (pp. 2389–2396). ACM. https://doi.org/10.1145/1570256.1570333

Hastie, T., Tibshirani, R., Friedman, J., & Franklin, J. (2005). The elements of statistical learning: Data mining, inference and prediction. *The Mathematical Intelligencer*, *27*(2), 83–85.

Hechenbichler, K. S. K. (2014). kknn: Weighted k-nearest neighbors. http://CRAN.R-project.org/package=kknn. R package version 1.2-5.

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

Holmes, G., Donkin, A., & Witten, I. H. (1994). Weka: A machine learning workbench. In *Proceedings of the 1994 second Australian and New Zealand conference on intelligent information systems*, 1994 (pp. 357–361). IEEE.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, *11*(1), 63–90.

Igel, C., & Toussaint, M. (2005). A no-free-lunch theorem for non-uniform distributions of target functions. *Journal of Mathematical Modelling and Algorithms*, *3*(4), 313–322.

Jordan, M., & Mitchell, T. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Karatzoglou, A., Smola, A., Hornik, K., & Zeileis, A. (2004). kernlab—An S4 package for kernel methods in R. *Journal of Statistical Software*, *11*(9), 1–20.

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. *Informatica*, *31*, 249–268.

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: A review of classification and combining techniques. *Artificial Intelligence Review*, *26*(3), 159–190.

Langley, P. (2011). The changing science of machine learning. *Machine Learning*, *82*(3), 275–279.

Lee, J. W., & Giraud-Carrier, C. (2013). Automatic selection of classification learning algorithms for data mining practitioners. *Intelligent Data Analysis*, *17*(4), 665–678.

Leite, R., & Brazdil, P. (2008). Selecting classifiers using metalearning with sampling landmarks and data characterization. In *Proceedings of the planning to learn workshop (PlanLearn 2008), held at ICML/COLT/UAI* (pp. 35–41).

Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, *247*(1), 124–136.

Lichman, M. (2013). *UCI machine learning repository*. http://archive.ics.uci.edu/ml

Lindner, G., & Studer, R. (1999). AST: Support for algorithm selection with a CBR approach. In J. M. Żytkow & J. Rauch (Eds.), *Principles of data mining and knowledge discovery: Third European conference*, PKDD'99, Prague, Czech Republic, September 15–18, 1999 Proceedings (pp. 418–423). Berlin, Heidelberg: Springer.

Macia, N., & Bernadó-Mansilla, E. (2014). Towards UCI+: A mindful repository design. *Information Sciences*, *261*, 237–262.

Macià, N., Orriols-Puig, A., Bernadó-Mansilla, E. (2010). In search of targeted-complexity problems. In *Proceedings of the 12th annual conference on genetic and evolutionary computation* (pp. 1055–1062). ACM.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2015). *e1071: Misc functions of the Department of Statistics, Probability Theory Group (Formerly: E1071)*, TU Wien (2015). http://CRAN.R-project.org/package=e1071. R package version 1.6-7.

Michie, D., Spiegelhalter, D. J., Taylor, C. C., & Campbell, J. (Eds.). (1994). *Machine learning, neural and statistical classification*. Upper Saddle River, NJ: Ellis Horwood.

Muñoz, M. A., & Smith-Miles, K. A. (2017). Performance analysis of continuous black-box optimization algorithms via footprints in instance space. *Evolutionary Computation, 25*(4), 529–554.

Orriols-Puig, A., Macia, N., & Ho, T. K. (2010). *Documentation for the data complexity library in c++* (Vol. 196). La Salle: Universitat Ramon Llull.

Peng, Y., Flach, P. A., Soares, C., & Brazdil, P. (2002). Improved dataset characterisation for meta-learning. In S. Lange, K. Satoh, & C. H. Smith (Eds.), *Discovery science: 5th international conference*, DS 2002 Lübeck, Germany, November 24–26, 2002 Proceedings (pp. 141–152). Berlin, Heidelberg: Springer.

Perez, E., & Rendell, L. A. (1996). Learning despite concept variation by finding structure in attribute-based data. In *Proceedings of the thirteenth international conference on machine learning*. Citeseer.

Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. (2000a). Meta-learning by landmarking various learning algorithms. In *Proceedings of the seventeenth international conference on machine learning* (pp. 743–750). San Francisco, CA: Morgan Kaufmann Publishers Inc.

Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. (2000b). Tell me who can learn you and I can tell you who you are: Landmarking various learning algorithms. In *Proceedings of the 17th international conference on machine learning* (pp. 743–750).

Ramakrishnan, N., Rice, J. R., & Houstis, E. N. (2002). Gauss: An online algorithm selection system for numerical quadrature. *Advances in Engineering Software*, *33*(1), 27–36.

Reif, M., & Shafait, F. (2014). Efficient feature size reduction via predictive forward selection. *Pattern Recognition*, *47*(4), 1664–1673.

Reif, M., Shafait, F., & Dengel, A. (2012). Meta-learning for evolutionary parameter optimization of classifiers. *Machine Learning*, *87*(3), 357–380.

Reif, M., Shafait, F., Goldstein, M., Breuel, T., & Dengel, A. (2014). Automatic classifier selection for non-experts. *Pattern Analysis and Applications*, *17*(1), 83–96.
Rendell, L., & Cho, H. (1990). Empirical learning as a function of concept character. *Machine Learning*, *5*(3), 267–298.
Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, *15*, 65–118.
Robnik-Šikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning*, *53*(1–2), 23–69.
Rudin, C., & Wagstaff, K. L. (2014). Machine learning for science and society. *Machine Learning*, *95*(1), 1–9.
Salzberg, S. L. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, *1*(3), 317–328.
Segrera, S., Pinho, J., & Moreno, M. N. (2008). Information-theoretic measures for meta-learning. In E. Corchado, A. Abraham, & W. Pedrycz (Eds.), *Hybrid artificial intelligence systems: Third international workshop*, HAIS 2008, Burgos, Spain, September 24–26, 2008 Proceedings (pp. 458–465). Berlin, Heidelberg: Springer.
Smith, K. A., Woo, F., Ciesielski, V., & Ibrahim, R. (2002). Matching data mining algorithm suitability to data characteristics using a self-organizing map. In A. Abraham & M. Köppen (Eds.), *Hybrid information systems* (pp. 169–179). Heidelberg: Physica-Verlag.
Smith-Miles, K., Baatar, D., Wreford, B., & Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, *45*, 12–24.
Smith-Miles, K., & Bowly, S. (2015). Generating new test instances by evolving in instance space. *Computers & Operations Research*, *63*, 102–113.
Smith-Miles, K., & van Hemert, J. (2011). Discovering the suitability of optimisation algorithms by learning from evolved instances. *Annals of Mathematics and Artificial Intelligence*, *61*(2), 87–104.
Smith-Miles, K., & Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, *39*(5), 875–889.
Smith-Miles, K., & Tan, T. (2012). Measuring algorithm footprints in instance space. In *IEEE CEC '12* (pp. 3446–3453).
Smith-Miles, K., & Tan, T. T. (2012) Measuring algorithm footprints in instance space. In *2012 IEEE congress on evolutionary computation (CEC)* (pp. 1–8). IEEE.
Smith-Miles, K., Wreford, B., Lopes, L., & Insani, N. (2013). Predicting metaheuristic performance on graph coloring problems using data mining. In E. Talbi (Ed.), *Hybrid metaheuristics* (pp. 417–432). Berlin, Heidelberg: Springer.
Smith-Miles, K. A. (2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, *41*(1), 6.
Soares, C. (2009). UCI++: Improved support for algorithm selection using datasetoids. In *Advances in knowledge discovery and data mining: 13th Pacific-Asia conference*, PAKDD 2009 Bangkok, Thailand, April 27–30, 2009 Proceedings (pp. 499–506). https://doi.org/10.1007/978-3-642-01307-2_46.
Soares, C., & Brazdil, P. B. (2000). Zoomed ranking: Selection of classification algorithms based on relevant performance information. In D. A. Zighed, J. Komorowski, & J. Żytkow (Eds.), *Principles of data mining and knowledge discovery: 4th European Conference*, PKDD 2000 Lyon, France, September 13–16, 2000 Proceedings (pp. 126–135). Berlin, Heidelberg: Springer.
Soares, C., Petrak, J., & Brazdil, P. (2001). Sampling-based relative landmarks: Systematically test-driving algorithms before choosing. In *Portuguese conference on artificial intelligence* (pp. 88–95). Springer.
Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427–437.
Song, Q., Wang, G., & Wang, C. (2012). Automatic recommendation of classification algorithms based on data set characteristics. *Pattern Recognition*, *45*(7), 2672–2689.
Therneau, T., Atkinson, B., & Ripley, B. (2014). *rpart: Recursive partitioning and regression trees*. http://CRAN.R-project.org/package=rpart. R package version 4.1-8.
Tsoumakas, G., Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multilabel classification. In *European conference on machine learning* (pp. 406–417). Springer.
Vanschoren, J. (2010). *Understanding machine learning performance with experiment databases*. PhD thesis, Katholieke Universiteit Leuven – Faculty of Engineering.
Vanschoren, J., van Rijn, J. N., Bischl, B., & Torgo, L. (2013). Openml: Networked science in machine learning. *SIGKDD Explorations*, *15*(2), 49–60. https://doi.org/10.1145/2641190.2641198.
Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York, NY: Springer-Verlag.
Venables, W. N., & Ripley, B. D. (2002). *Modern applied statistics with S* (4th ed.). Springer, New York. http://www.stats.ox.ac.uk/pub/MASS4. ISBN 0-387-95457-0

Vilalta, R. (1999). Understanding accuracy performance through concept characterization and algorithm analysis. In *Proceedings of the ICML-99 workshop on recent advances in meta-learning and future work* (pp. 3–9).

Vilalta, R., & Drissi, Y. (2002). A characterization of difficult problems in classification. In M. A. Wani, H. R. Arabnia, K. J. Cios, K. Hafeez, & G. Kendall (Eds.), *Proceedings of the 2002 international conference on machine learning and applications - ICMLA 2002*, June 24–27, 2002, Las Vegas, Nevada (pp. 133–138).

Wagstaff, K. (2012). *Machine learning that matters*. arXiv preprint arXiv:1206.4656

Weerawarana, S., Houstis, E. N., Rice, J. R., Joshi, A., & Houstis, C. E. (1996). Pythia: A knowledge-based system to select scientific algorithms. *ACM Transactions on Mathematical Software (TOMS)*, *22*(4), 447–468.

Yarrow, S., Razak, K. A., Seitz, A. R., & Seriès, P. (2014). Detecting and quantifying topography in neural maps. *PLoS ONE*, *9*(2), 1–14. https://doi.org/10.1371/journal.pone.0087178.

Young, W., Weckman, G., & Holland, W. (2011). A survey of methodologies for the treatment of missing values within datasets: Limitations and benefits. *Theoretical Issues in Ergonomics Science*, *12*(1), 15–43.