Machine Learning (2022) 111:3085–3123
https://doi.org/10.1007/s10994-022-06205-9

Relating instance hardness to classification performance in a dataset: a visual approach

Pedro Yuri Arbs Paiva[1] · **Camila Castro Moreno**[1,2] · **Kate Smith-Miles**[3] · **Maria Gabriela Valeriano**[1,2] · **Ana Carolina Lorena**[1]

Received: 5 September 2021 / Revised: 3 April 2022 / Accepted: 26 May 2022 /
Published online: 22 June 2022

Abstract
Machine Learning studies often involve a series of computational experiments in which the predictive performance of multiple models are compared across one or more datasets. The results obtained are usually summarized through average statistics, either in numeric tables or simple plots. Such approaches fail to reveal interesting subtleties about algorithmic performance, including which observations an algorithm may find easy or hard to classify, and also which observations within a dataset may present unique challenges. Recently, a methodology known as Instance Space Analysis was proposed for visualizing algorithm performance across different datasets. This methodology relates predictive performance to estimated instance hardness measures extracted from the datasets. However, the analysis considered an instance as being an entire classification dataset and the algorithm performance was reported for each dataset as an average error across all observations in the dataset. In this paper, we developed a more fine-grained analysis by adapting the ISA methodology. The adapted version of ISA allows the analysis of an individual classification dataset by a 2-D hardness embedding, which provides a visualization of the data according to the difficulty level of its individual observations. This allows deeper analyses of the relationships between instance hardness and predictive performance of classifiers. We also provide an open-access Python package named PyHard, which encapsulates the adapted ISA and provides an interactive visualization interface. We illustrate through case studies how our tool can provide insights about data quality and algorithm performance in the presence of challenges such as noisy and biased data.

Keywords Instance hardness · Classification performance · Hardness embedding · Meta-learning

Editor: Salvador Garcia.

✉ Pedro Yuri Arbs Paiva
paiva@ita.br

Extended author information available on the last page of the article

1 Introduction

A well known maxim from the Machine Learning (ML) literature is that each ML algorithm has a bias that makes it more suitable for some classes of problems than others. This is stated formally by the *No-Free-Lunch theorem* (Wolpert, 2002), which asserts that any two algorithms perform equally well on average when considering all classes of problems. Learning which classification technique should be used to tackle a particular test problem or instance can be modeled by Meta-Learning (MtL) (Vilalta & Drissi, 2002), which seeks to learn how to map problem characteristics in the performance of ML algorithms (Vanschoren, 2019).

Despite discussions on the (in)feasibility of universal predictors and how MtL can help their construction (Giraud-Carrier & Provost, 2005), a common and advisable practice in ML studies is to compare multiple models in a controlled set of experiments, using the same datasets and data partitions. Usually a summary of the results are reported and compared among each other in the form of averages and standard deviation values, across all the instances in a dataset, and for all datasets in the study. While traditional, this type of coarse-grained statistical analysis hinders a more fine-grained evaluation of the strengths and weaknesses of the predictive models obtained. Despite a good overall performance on average on a dataset, a model may be inaccurate on important subsets of instances (observations within a dataset). This can lead to algorithmic biases (Hajian et al., 2016) and deceptive results when some ML models are put into production. In other words, examining performance on individual instances within a dataset can offer a better understanding of an algorithm's true effectiveness and also allows the identification of possible quality issues - inaccuracies and biases - with the dataset.

Recently, a methodology known as Instance Space Analysis (ISA) has been developed for the analysis of algorithms at the instance level, and demonstrated success when analyzing the performance of classification techniques in ML across multiple datasets (Muñoz et al., 2018). There, popular classification datasets from public repositories are described by a set of meta-features and have their predictive performance assessed for multiple ML classification techniques. A 2-D projection relating these characteristics to the performances of the algorithms is then created, presenting linear trends that reveal pockets of hard and easy datasets, that is, datasets that are harder or easier for the algorithms to classify, and how each classifier performs on them. This provides valuable knowledge towards understanding the domains of competence of each classification technique and for aiding automated algorithm selection. It also reveals the lack of diversity of these common benchmarks and the need for generating more challenging datasets. In the previous work however, the analysis was done by considering an instance as an entire classification dataset, and the algorithm performance was reported for each dataset as an average error across all observations in the dataset.

This paper considers a more fine-grained analysis by adapting the ISA framework to the analysis of a single classification dataset, with an instance defined as an observation within the dataset. The idea is to project the original data into a 2-D hardness embedding which can be scrutinized to inspect data quality and to more deeply understand classifier behaviors in a single dataset. This enables closer inspection of observation's (instance) characteristics that each classifier struggles the most with. To this end, we revisit the concept of instance hardness, introduced in the work of Smith et al. (2014) for assessing the level of difficulty or probability of misclassification of each instance in a classification dataset. By relating meta-features that describe instance hardness to the predictive performance

of multiple classifiers, the ISA projections provide valuable information on each classifier's strengths and weaknesses. Furthermore, an analysis of data quality issues in a dataset becomes possible. The main contributions of our paper can therefore be summarized as:

- We propose the analysis of a classification dataset and algorithms by a 2-D hardness embedding, which allows the visualization of the data according to the difficulty level of its individual instances;
- We adapt the ISA framework to obtain this projection, by relating instance hardness meta-features to the predictive performance of multiple classifiers;
- We present and analyze the hardness profile of a few illustrative datasets, including a real dataset of COVID-19 patients with symptoms and comorbidities;
- We analyze how the hardness profile of some datasets changes when subject to interventions such as the introduction of label noise;
- We provide an open-access Python package named "PyHard",[1] which encapsulates the adapted ISA and provides an interactive visualization interface for relating instance hardness to classification performance.

With our open source software contribution PyHard, we expect to leverage the concept of instance hardness and provide users with the possibility of inspecting their data and algorithmic performance beyond simple descriptive summaries and plots. As shown in the experiments presented in this paper, the developed tool allows for a better understanding of which characteristics pertaining to the training dataset most affect the predictive performance of different ML classification algorithms. The tool also allows the analysis of the effects of typical data quality issues faced in ML. Specifically, the experiments performed seek to ask the following questions:

1. How can we use ISA and instance hardness metrics to understand a dataset at the level of its individual observations?
2. Is it possible to identify and explain any data quality issues by visually inspecting hard instances and their feature values?
3. How robust are the instance hardness metrics and conclusions regarding algorithm strengths and weaknesses in the presence of data quality issues such as label noise?

We show that ISA can help provide evidence and understanding of common issues a data scientist and Machine Learning practitioner face when applying classification models on datasets, and how the biases of a dataset or algorithm can become apparent.

The remainder of this paper is organized as follows. Section 2 summarizes the ISA framework and its main methodological steps. Section 3 reformulates the ISA framework for its application on a single dataset. Section 4 presents how the ISA projections can be used for inspecting data and highlighting classifier strengths and weaknesses in a dataset. Real, benchmark and synthetic datasets are analyzed to this end. Section 5 concludes this work, and provides recommendations for future research.

[1] https://pypi.org/project/pyhard/.

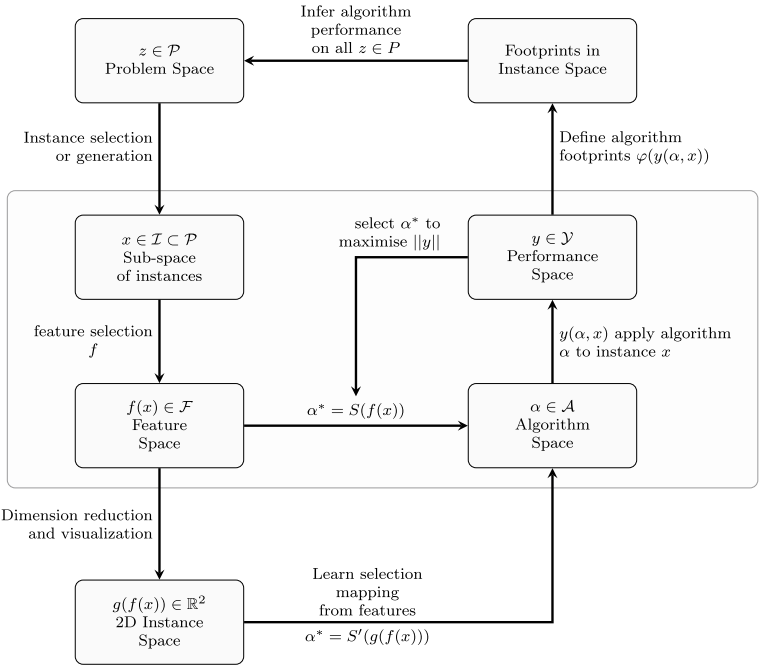


Fig. 1 Instance Space Analysis framework. Extracted from Muñoz et al. (2018)

2 Instance space analysis

The Instance Space Analysis (ISA) framework was originally proposed in (Smith-Miles & Lopes, 2011; Smith-Miles et al., 2014) for the analysis of optimization problems and algorithms, and later extended to ML and other domains (Muñoz et al., 2018; Kang et al., 2017). Since its proposal, the ISA methodology has been used and validated in the analysis of multiple problems, including: rostering (Kletzander et al., 2021), knapsack problems (Smith-Miles et al., 2021), timetabling (Smith-Miles and Lopes, 2011), traveling salesman problems (Smith-Miles and Tan, 2012), graph coloring (Smith-Miles et al., 2014), black-box optimization (Muñoz and Smith-Miles, 2017), time series forecasting (Kang et al., 2017), classification (Muñoz et al., 2018), anomaly detection (Kandanaarachchi et al., 2020) and regression (Muñoz et al., 2021).

As originally proposed and described in this section, ISA builds upon the Algorithm Selection Problem (ASP) (Rice, 1976; Smith-Miles, 2009), highlighted as the shaded blue area in Fig. 1. The objective in ASP is to automate the process of selecting good candidate algorithms and their hyperparameters for solving new problems, based on knowledge gathered from similar problems they solved in the past. The following sets from Fig. 1 compose the core of ASP:

- **Problem space** $\mathcal{P}$: all instances from the problem/domain under consideration;
- **Instance sub-space** $\mathcal{I}$: contains a sub-set of instances sampled from $\mathcal{P}$ for which the characteristics and solutions are available or can be easily computed;

- **Feature space** $\mathcal{F}$: set of descriptive characteristics, also known as meta-features in ML, extracted from the instances belonging to $\mathcal{I}$;
- **Algorithm space** $\mathcal{A}$: contains a portfolio of algorithms that can be used to solve the instances in $\mathcal{I}$;
- **Performance space** $\mathcal{Y}$: evaluating the performance of the algorithms from the set $\mathcal{A}$ on the instances from $\mathcal{I}$ yields the performance space $\mathcal{Y}$.

The combination of tuples $(x, f(x), \alpha, y(\alpha, x))$, where $x \in \mathcal{I}$ is an instance, described by meta-features $f(x) \in \mathcal{F}$, $\alpha \in \mathcal{A}$ is an algorithm and $y(\alpha, x) \in \mathcal{Y}$ gives the performance of α when applied to x, for all instances in $\mathcal{I}$ and all algorithms in $\mathcal{A}$, composes a meta-dataset $\mathcal{M}$. A meta-learner S can then be trained in order to select the best algorithm (or a ranking of algorithms) to be recommended for a new instance x based on its meta-features, that is, $\alpha^* = S(f(x)) = \arg\max_{\alpha} ||y(\alpha, x)||$. α^* is the algorithm (or a set of algorithms) with maximum predictive performance for x as measured by y.

The Instance Space Analysis (ISA) framework goes further and extends the ASP analysis to give insights into why some instances are harder to solve than others, combining the information of meta-features and algorithm performance in a new embedding that can be visually inspected. To this end, an optimization problem is solved to find the mapping $g(f(x))$ from the meta-features' multidimensional space into a 2-D space, such that the distribution of algorithm performance metrics and meta-feature values across instances in the 2-D space displays as much of a linear trend as possible to assist the interpretation of hardness directions. The 2-D Instance Space (IS) can then be inspected for regions of good and bad algorithmic performance, with ML techniques used to predict algorithms to be recommended for each instance $\alpha^* = S'(g(f(x)))$, providing an alternative approach for ASP as well as the insights permitted by the visualization.

Within the instance space, it is also possible to define areas of strength for each algorithm known as the *algorithm footprint* $\varphi(y(\alpha, x))$, that is, areas of the IS where the algorithm α performs well. A set of objective measures can be extracted from an algorithm's footprint for algorithmic power evaluation in the IS, such as the area of coverage, purity and density, which will be discussed next. Such meta-knowledge can also support the inference on algorithmic performance for other instances $z \in \mathcal{P}$ which were not in the sub-space $\mathcal{I}$ used to build the IS.

Finally, it is possible to examine the diversity of the projected instances and, when applicable, to enrich the IS with carefully designed new instances (Smith-Miles and Bowly, 2015; Muñoz et al., 2018; Smith-Miles et al., 2021). With such a procedure, one might be able to produce more challenging problem instances expanding the boundaries of the ISA.

Summarizing, the application of the ISA methodology requires (Muñoz et al., 2018):

i. Building the meta-dataset $\mathcal{M}$;
ii. Reducing the set of meta-features in $\mathcal{M}$ by keeping only those able to best discriminate the algorithms' performances;
iii. Creating a 2-D IS from the meta-dataset $\mathcal{M}$;
iv. Building the algorithms' footprints in the IS for measuring algorithmic performance across the IS.

Step (i) is dependent on the problem domain, involving the choice of the problem's instances, meta-features, algorithms and performance measures (the sets $\mathcal{I}, \mathcal{F}, \mathcal{A}$ and $\mathcal{Y}$). The choice of a subset of the meta-features in step (ii) can be done by employing any

suitable feature selection algorithm. Here we use a rank aggregation approach, described in Sect. 3. Steps (iii) and (iv) are implemented in the MATLAB language and freely available for use in MATILDA (Melbourne Algorithm Test Instance Library with Data Analytics) tool.[2] MATILDA also includes a feature selection procedure for performing step (ii), although the user is encouraged to explore independent methods to arrive at a strong feature selection. Our work has re-implemented steps (iii) and (iv) using the Python language, which are gathered in a public package named "PyISpace".[3]

2.1 Instance Space construction

We now consider the problem of finding an optimum mapping from the metadata domain to the 2-D instance space. We follow the Prediction Based Linear Dimensionality Reduction (PBLDR) method proposed in Muñoz et al. (2018). Given a meta-dataset $\mathcal{M}$ with n instances and m meta-features, let $\mathbf{F} \in \mathbb{R}^{m \times n}$ be a matrix containing the meta-features values for all instances and $\mathbf{Y} \in \mathbf{R}^{n \times a}$ be a matrix containing the performance measure of a algorithms on the same n instances. An ideal 2-D projection of the instances for this group of algorithms is achieved by finding the matrices $\mathbf{A}_r \in \mathbb{R}^{2 \times m}$, $\mathbf{B}_r \in \mathbb{R}^{m \times 2}$ and $\mathbf{C}_r \in \mathbb{R}^{a \times 2}$ which minimize the approximation error:

$$\|\mathbf{F} - \widehat{\mathbf{F}}\|_F^2 + \|\mathbf{Y} - \widehat{\mathbf{Y}}\|_F^2 \tag{1}$$

such that:

$$\mathbf{Z} = \mathbf{A}_r \mathbf{F} \tag{2}$$

$$\widehat{\mathbf{F}} = \mathbf{B}_r \mathbf{Z} \tag{3}$$

$$\widehat{\mathbf{Y}}^\top = \mathbf{C}_r \mathbf{Z} \tag{4}$$

where $\mathbf{Z} \in \mathbb{R}^{2 \times n}$ is the matrix instance coordinates in the 2-D space and $\mathbf{A}_r$ is the projection matrix. Essentially, this optimization problem seeks to find the optimal linear transformation matrix $\mathbf{A}_r$, such that the mapping of all instances from $\mathbb{R}^m$ to $\mathbb{R}^2$ results in the strongest possible linear trends across the instance space when inspecting the distribution of each algorithm's performance metric, and each feature. The maximization of linear trends for both meta-features and algorithmic performances in the new space is guaranteed by the matrices $\mathbf{B_r}$ and $\mathbf{C_r}$ in Eqs. (3) and (4).

Assuming that $m < n$ and that $\mathbf{F}$ is full row rank (or considering the problem in a subspace spanned by $\mathbf{F}$), the following alternative optimization problem is obtained:

[2] https://matilda.unimelb.edu.au/matilda/.
[3] https://pypi.org/project/pyispace/.

$$\begin{aligned} &\min \left\|\mathbf{F}-\mathbf{B}_r\mathbf{Z}\right\|_F^2+\left\|\mathbf{Y}-\mathbf{C}_r\mathbf{Z}\right\|_F^2 \\ &\text{s.t.}\mathbf{Z}=\mathbf{A}_r\mathbf{F} \\ &\quad \mathbf{A}_r \in \mathbb{R}^{2\times m} \\ &\quad \mathbf{B}_r \in \mathbb{R}^{m\times 2} \\ &\quad \mathbf{C}_r \in \mathbb{R}^{a\times 2} \end{aligned} \tag{5}$$

Muñoz et al. (2021) solve this problem numerically using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm as the numerical solver, which is also used here. From multiple runs, the solution that achieves a maximum topological preservation as proposed in (Yarrow et al., 2014) is chosen, that is, the solution with maximum Pearson Correlation between the distances in the feature space and the distances in the instance space.

2.2 Footprint analysis

A *footprint* is a region in the instance space where an algorithm is expected to perform well based on inference from empirical performance analysis (Muñoz et al., 2018). Two types of footprints are currently output in the ISA analysis. The first indicates regions of the IS where the algorithm shows a good performance according to a given threshold on algorithmic performance. The second corresponds to regions where the algorithm performs better compared to all others contained in the portfolio.

In order to construct an algorithm's footprint of good performance, first the performance measure values contained in $\mathcal{Y}$ must be binarized so that the performance label for each algorithm on an instance is either easy (also named good in ISA) or hard (or bad in the ISA terminology) based on a user-defined threshold. This is done for each algorithm in the portfolio $\mathcal{A}$, resulting in a binary matrix $\mathbf{Y}_{bin}$ with instances as rows and algorithms as columns. For each algorithm in $\mathcal{A}$, the DBSCAN algorithm (Khan et al., 2014) is then used to identify high density clusters of easy instances. Next, α-shapes are used to construct hulls which enclose all the points within the clusters (Edelsbrunner, 2010). For each cluster hull, a Delaunay triangulation creates a partition, and those triangles that do not satisfy a minimum purity (the percentage of good instances enclosed within it) requirement are removed. The union of the remaining triangles gives the footprint of the algorithm where good performance is expected based on statistical evidence.

The footprint of best performance is built similarly, but taking into account the relative performance of the algorithms in the IS. That is, a Delaunay triangulation is formed for dense regions containing instances where the algorithm performs better than the other algorithms in the pool. The best footprints of multiple algorithms are also compared in order to remove contradicting areas due to overlaps. These footprints are generally smaller and may be absent if there is not a region of the IS where the algorithm performs consistently better when compared to the others.

It is also possible to define some objective measures of algorithmic power for each algorithm across the IS by computing:

i. The area of the footprint (A), which can be normalized across multiple algorithms for ease of comparison;
ii. The density of the footprint (ρ), computed as the ratio between the number of instances enclosed by the footprint and its area;

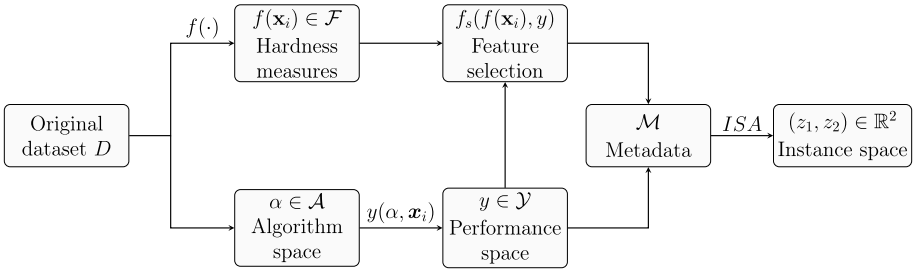


Fig. 2 ISA framework for a single dataset

iii. The purity of the footprint p, which corresponds to the percentage of good instances enclosed by the footprint.

Larger values for these measures provide evidence of a better performance of an algorithm across the IS. A large A implies the algorithm shows a good performance for a large portion of the IS. A large ρ means such area is dense and contains a large amount of instances. Finally, p is large when most of the instances enclosed in A are good and will be maximum when all instances in A are good. A strong algorithm in the IS is expected to present a large normalized footprint area, with density close to one and purity as close to 100% as the chosen feature set will permit.

3 ISA for a single dataset

ISA has been used in the analysis of public benchmark repositories in ML and popular classification algorithms in Muñoz et al. (2018) and, more recently, in the analysis of regression datasets and algorithms (Muñoz et al., 2021). In this section we present how we recast the framework for the analysis of a single classification dataset. Our main interest is on the insights which can be obtained by relating data characteristics and meta-features to algorithmic performance. Therefore, some steps of the original ISA framework are not included, such as the algorithmic recommendation module and the generation of new instances.

Given a classification dataset D containing n_D instances $\mathbf{x}_i \in X$, with m_D input features and labeled in a class $y_i \in Y$ each, we have:

- **Problem space** $\mathcal{P}$: is reduced to the dataset D;
- **Instance space** $\mathcal{I}$: contains all individual instances $\mathbf{x}_i \in D$;
- **Feature space** $\mathcal{F}$: contains a set of meta-features describing instance hardness, also known as *hardness measures* (HM) (Smith et al., 2014);
- **Algorithm space** $\mathcal{A}$: comprises a portfolio of classification algorithms of distinct biases;
- **Performance space** $\mathcal{Y}$: records the performance obtained by each algorithm in $\mathcal{A}$ for each instance $\mathbf{x}_i \in D$.

The following subsections present the components from our framework, which are summarized in Fig. 2. Accordingly, for each instance $\mathbf{x}_i$ in the dataset D, a set of

hardness measures is extracted, and each algorithm in the set $\mathcal{A}$ has its performance measured on $\mathbf{x}_i$, as represented by $y(\alpha, \mathbf{x}_i)$. This is done in a cross-validation step, where the log-loss error obtained in the prediction of the instance label is recorded for each classification technique. This cross-validated log-loss error is the performance metric stored in $\mathcal{Y}$. A feature selection step considering the power of the meta-features to describe the performances of the algorithms in $\mathcal{A}$ is performed, resulting in a reduced meta-feature subset $f_s(f(\mathbf{x}_i), y)$. Combining the sub-set of selected meta-features to the predictive performances of multiple classification algorithms allows the construction of the meta-dataset $\mathcal{M}$, from which the 2-D Instance Space projection with coordinates z_1 and z_2 is extracted. These steps will be further explained in Sects. 3.1 to 3.4.

3.1 Hardness measures

One important aspect when performing ISA is using a set of informative meta-features that are able to reveal the capabilities of the algorithms and the level of difficulty each individual instance poses. Here we revisit the definition of *Instance Hardness* (IH) proposed by Smith et al. (2014) as a property that indicates the likelihood that an instance will be misclassified. Namely, the hardness of the instance $\mathbf{x}_i$ with respect to a classification hypothesis h is

$$IH_h\big(\mathbf{x}_i, y_i\big) = 1 - p(y_i|\mathbf{x}_i, h), \tag{6}$$

where $h : X \rightarrow Y$ is a hypothesis or function mapping input features in an input space X to output labels in an output space Y. In practice, h is induced by a learning algorithm l trained on a dataset $D = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in X \wedge y_i \in Y\}$ with hyper-parameters β, that is, $h = l(D, \beta)$ (Smith et al., 2014). The authors also derive the instance hardness for a set of representative learning algorithms $\mathcal{L}$. We adopt this expression throughout the work, instantiating the set $\mathcal{L}$ to the pool of classifiers $\mathcal{A}$ in ISA.

$$IH_{\mathcal{A}}\big(\mathbf{x}_i, y_i\big) = 1 - \frac{1}{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|} p\big(y_i|\mathbf{x}_i, l_j(D, \beta)\big) \tag{7}$$

The idea is that instances that are frequently misclassified by a pool of diverse learning algorithms can be considered hard. On the other hand, easy instances are likely to be correctly classified by any of the considered algorithms.

An additional interest of this paper is assessing which characteristics of the data items make them hard to classify. Smith et al. (2014) define a set of *hardness measures* (HM) intended to explain why some instances are often misclassified. These are the measures employed as meta-features in $\mathcal{F}$. Since their objective is to characterize the level of difficulty in the classification of each instance in a dataset, they are natural candidates for describing the algorithm's performances on the same data. Table 1 summarizes the HM employed in this work, with their names, acronyms, minimum and maximum values achievable and references from where they are extracted. We introduced modifications into some of the measures in order to limit and standardize their values. Consequently, all measures are constructed so that higher values are registered for instances that are harder to classify.

For each instance $\mathbf{x}_i \in \mathcal{D}$, the hardness measures extracted are:

Table 1 Hardness measures employed as meta-features in this work

Measure	Acron.	Min	Max	References
k-Disagreeing Neighbors	*kDN*	0	1	Smith et al. (2014)
Disjunct Class Percentage	*DCP*	0	1	Smith et al. (2014)
Tree Depth (pruned)	TD_P	0	1	Smith et al. (2014)
Tree Depth (unpruned)	TD_U	0	1	Smith et al. (2014)
Class Likelihood	*CL*	0	1	Smith et al. (2014)
Class Likelihood Difference	*CLD*	0	1	Smith et al. (2014)
Frac. features in overlapping areas	*F*1	0	1	Arruda et al. (2020)
Frac. nearby instances different class	*N*1	0	1	Arruda et al. (2020)
Ratio of intra-extra class distances	*N*2	0	≈ 1	Arruda et al. (2020)
Local set cardinality	*LSC*	0	1	Arruda et al. (2020)
Local set radius	*LSR*	0	1	Arruda et al. (2020)
Usefulness	*U*	≈ 0	1	Arruda et al. (2020)
Harmfulness	*H*	0	≈ 1	Arruda et al. (2020)

k-Disagreeing Neighbors $kDN(\mathbf{x}_i)$: gives the percentage of the k nearest neighbors of $\mathbf{x}_i$ which do not share its label, as described by Eq. (8). As in (Smith et al., 2014), the value of k is set to 5.

$$kDN(\mathbf{x}_i) = \frac{\sharp\{\mathbf{x}_j | \mathbf{x}_j \in kNN(\mathbf{x}_i) \wedge y_j \neq y_i\}}{k}, \tag{8}$$

where $kNN(\mathbf{x}_i)$ represents the set of k-nearest neighbors of the instance $\mathbf{x}_i$ in the dataset D. The higher the value of $kDN(\mathbf{x}_i)$, the harder its classification tends to be, since it is surrounded by examples from a different class. This measure can be computed at a $O(n_D \cdot m_D)$ asymptotic computational cost for a dataset D with n_D instances and m_D input features.

Disjunct Class Percentage $DCP(\mathbf{x}_i)$: builds a decision tree using $\mathcal{D}$ and considers the percentage of instances in the disjunct of $\mathbf{x}_i$ which share the same label as $\mathbf{x}_i$. The disjunct of an example corresponds to the leaf node where it is classified by the decision tree.

$$DCP(\mathbf{x}_i) = 1 - \frac{\sharp\{\mathbf{x}_j | \mathbf{x}_j \in Disjunct(\mathbf{x}_i) \wedge y_j = y_i\}}{\sharp\{\mathbf{x}_j | \mathbf{x}_j \in Disjunct(\mathbf{x}_i)\}}, \tag{9}$$

where $Disjunct(\mathbf{x}_i)$ represents the set of instances contained in the disjunct where $\mathbf{x}_i$ is placed. Easier instances will have a larger percentage of examples sharing the same label as them in their disjunct. Therefore, we output the complement of this percentage. Building the DT dominates the asymptotic computational cost of this measure and can be performed at $O(m_D \cdot n_D \cdot \log_2 n_D)$ steps (Sani et al., 2018).

Tree Depth $TD(\mathbf{x}_i)$: returns the depth of the leaf node that classifies $\mathbf{x}_i$ in a decision tree DT, normalized by the maximum depth of the tree built from D:

$$TD(\mathbf{x}_i) = \frac{depth_{DT}(\mathbf{x}_i)}{\max(depth_{DT}(\mathbf{x}_j \in D))}, \tag{10}$$

where $depth_{DT}(\mathbf{x}_i)$ gives the depth where the instance $\mathbf{x}_i$ is placed in the decision tree. There are two versions of this measure, using pruned ($TD_P(\mathbf{x}_i)$) and unpruned ($TD_U(\mathbf{x}_i)$) decision trees. Harder to classify instances tend to be placed at deeper levels of the trees and present higher *TD* values. This measure also requires building a decision tree, with an asymptotic computational cost of $O(m_D \cdot n_D \cdot \log_2 n_D)$.

Class Likelihood $CL(\mathbf{x}_i)$: measures the likelihood of $\mathbf{x}_i$ belonging to its class:

$$CL(\mathbf{x}_i) = 1 - P(\mathbf{x}_i|y_i)P(y_i), \tag{11}$$

where $P(\mathbf{x}_i|y_i)$ represents the likelihood of $\mathbf{x}_i$ belonging to class y_i, measured in D, and $P(y_i)$ is the prior of class y_i, which we set as $\frac{1}{n}$ for all data instances. For ease of computation, the conditional probability $P(\mathbf{x}_i|y_i)$ can be estimated considering each of the input features independent from each other, as done in Naive Bayes classification. Larger class likelihood values are expected for easier instances, so we output the complement of this value. The asymptotic computational cost to compute the required probabilities from the dataset is $O(m_D \cdot n_D)$.

Class Likelihood Difference $CLD(\mathbf{x}_i)$: takes the difference between the likelihood of $\mathbf{x}_i$ in relation to its class and the maximum likelihood it has to any other class.

$$CLD(\mathbf{x}_i) = \frac{1 - \left(P(\mathbf{x}_i|y_i)P(y_i) - \max_{y_j \neq y_i}[P(\mathbf{x}_i|y_j)P(y_j)]\right)}{2}. \tag{12}$$

The difference in the class likelihood is larger for easier instances, because the confidence it belongs to its class is larger than that of any other class. We take the complement of the measure as indicated in Eq. (12).[4] The probabilities can be calculated as in CL, resulting in an asymptotic computational cost of $O(m_D \cdot n_D)$.

Fraction of features in overlapping areas $F1(\mathbf{x}_i)$: this measure takes the percentage of features of the instance $\mathbf{x}_i$ whose values lie in an overlapping region of the classes as:

$$F1(\mathbf{x}_i) = \frac{\sum_{j=1}^{m_D} I(x_{ij} > \text{max\_min}(\mathbf{f}_j) \wedge x_{ij} < \text{min\_max}(\mathbf{f}_j))}{m_D}, \tag{13}$$

where I is the indicator function, which returns 1 if its argument is true and 0 otherwise, $\mathbf{f}_j$ is the j-th feature vector in D and:

$$\text{min\_max}(\mathbf{f}_j) = \min(\max(\mathbf{f}_j^{c_1}), \max(\mathbf{f}_j^{c_2})),$$
$$\text{max\_min}(\mathbf{f}_j) = \max(\min(\mathbf{f}_j^{c_1}), \min(\mathbf{f}_j^{c_2})).$$

The values $\max(\mathbf{f}_j^{y_i})$ and $\min(\mathbf{f}_j^{y_i})$ are the maximum and minimum values of $\mathbf{f}_j$ in a class $y_i \in \{c_1, c_2\}$. According to the previous definition, the overlap for a feature $\mathbf{f}_j$ is measured according to the maximum and minimum values it assumes in the different classes and one may regard a feature as having overlap if it is not possible to separate the classes using a threshold on that feature's values. F1 defines instance hardness according to whether the example is in one or more of the feature overlapping regions in a dataset. Larger values of $F1$ are obtained for data instances which lie in overlapping regions for

[4] For binary classification problems, CL and CLD as defined by Equations (11) and (12) are the same, as $\max_{y_j \neq y_i}[P(\mathbf{x}_i|y_j)P(y_j)] = 1 - P(\mathbf{x}_i|y_i)P(y_i)$ in this case.

most of the features, implying they are harder according to the $F1$ interpretation. Multiclass classification problems are first decomposed into multiple pairwise binary classification problems, whose results are averaged. The asymptotic computational cost of this measure is $O(m_D n_D)$ for binary classification problems and $O(m_D \cdot n_D \cdot C)$ for multiclass problems with $C > 2$ classes, supposing that each of the classes has the same number of observations, that is, $\frac{n_D}{C}$.

Fraction of nearby instances of different classes $N1(\mathbf{x}_i)$: in this measure, first a minimum spanning tree (MST) is built from D. In this tree, each instance of the dataset D corresponds to one vertex and nearby instances are connected according to their distances in the input space in order to obtain a tree of minimal cost concerning the sum of the edges' weights. $N1$ gives the percentage of instances of different classes $\mathbf{x}_i$ is connected to in the MST.

$$N1(\mathbf{x}_i) = \frac{\sharp\{\mathbf{x}_j | (\mathbf{x}_i, \mathbf{x}_j) \in MST(D) \wedge y_i \neq y_j\}}{\sharp\{\mathbf{x}_j | (\mathbf{x}_i, \mathbf{x}_j) \in MST(D)\}} \tag{14}$$

Larger values of $N1(\mathbf{x}_i)$ indicate that $\mathbf{x}_i$ is close to examples of different classes, either because it is borderline or noisy, making it hard to classify. This measure requires first computing the distance matrix between all pairs of elements in D, which requires $O(m_D \cdot n_D^2)$ operations and dominates the computational cost of this measure.

Ratio of the intra-class and extra-class distances $N2(\mathbf{x}_i)$: first the ratio of the distance of $\mathbf{x}_i$ to the nearest example from its class to the distance it has to the nearest instance from a different class (aka nearest enemy) is computed:

$$\text{IntraInter}(\mathbf{x}_i) = \frac{d(\mathbf{x}_i, NN(\mathbf{x}_i) \in y_i)}{d(\mathbf{x}_i, \text{ne}(\mathbf{x}_i))}, \tag{15}$$

where $NN(\mathbf{x}_i)$ represents a nearest neighbor of $\mathbf{x}_i$ and $\text{ne}(\mathbf{x}_i)$ is the nearest enemy of $\mathbf{x}_i$:

$$\text{ne}(\mathbf{x}_i) = NN(\mathbf{x}_i) \in y_j \neq y_i. \tag{16}$$

Then $N2$ is taken as:

$$N2(\mathbf{x}_i) = 1 - \frac{1}{\text{IntraInter}(\mathbf{x}_i) + 1} \tag{17}$$

Larger values of $N2(\mathbf{x}_i)$ indicate that the instance $\mathbf{x}_i$ is closer to an example from another class than to an example from its own class and is, therefore, harder to classify. As in $N1$, the larger computational cost involved in obtaining $N2$ is to compute a distance matrix between all pairs of elements in D, requiring $O(m_D \cdot n_D^2)$ operations.

Local Set Cardinality $LSC(\mathbf{x}_i)$: the Local-Set (LS) of an instance $\mathbf{x}_i$ is the set of points from D whose distances to $\mathbf{x}_i$ are smaller than the distance between $\mathbf{x}_i$ and $\mathbf{x}_i$'s nearest enemy, as defined in Equation (19) (Leyva et al., 2014). LSC outputs the relative cardinality of such set:

$$LSC(\mathbf{x}_i) = 1 - \frac{|LS(\mathbf{x}_i)|}{\sharp\{\mathbf{x}_j | y_i = y_j\}}. \tag{18}$$

$$LS(\mathbf{x}_i) = \sharp\{\mathbf{x}_j | d(\mathbf{x}_i, \mathbf{x}_j) < d(\mathbf{x}_i, \text{ne}(\mathbf{x}_i))\}, \tag{19}$$

where $\text{ne}(\mathbf{x}_i)$ is the nearest enemy of $\mathbf{x}_i$ (Eq. (16)), that is, the example from another class that is closest to $\mathbf{x}_i$. Larger local sets are obtained for easier examples, which are in dense regions surrounded by instances from their own classes. Therefore, in Eq. (18) we output a complement of the relative local set cardinality. The asymptotic cost of *LSC* is dominated by the computation of pairwise distances between all instances in D, resulting in $O(m_D \cdot n_D^2)$ operations.

Local Set Radius $LSR(\mathbf{x}_i)$: takes the normalized radius of the local set of $\mathbf{x}_i$:

$$LSR(\mathbf{x}_i) = 1 - \min\left\{1, \frac{d(\mathbf{x}_i, \text{ne}(\mathbf{x}_i))}{\max(d(\mathbf{x}_i, \mathbf{x}_j)|y_i = y_j)}\right\} \tag{20}$$

Larger radiuses are expected for easier instances, so we take the complement of such measure. As in *LSC*, the asymptotic cost of *LSR* is $O(m_D \cdot n_D^2)$.

Usefulness $U(\mathbf{x}_i)$: corresponds to the fraction of instances having $\mathbf{x}_i$ in their local sets (Leyva et al., 2015).

$$U(\mathbf{x}_i) = 1 - \frac{\sharp\{\mathbf{x}_j | d(\mathbf{x}_i, \mathbf{x}_j) < d(\mathbf{x}_j, \text{ne}(\mathbf{x}_j))\}}{|D| - 1} \tag{21}$$

If $\mathbf{x}_i$ is easy to classify, it will be close to many examples from its class and therefore will be more useful. We take the complement of this measure as output. The asymptotic computational cost of U is $O(m_D \cdot n_D^2)$, since the cost of computing the distance between all pairs of elements in the dataset is dominant.

Harmfulness $H(\mathbf{x}_i)$: number of instances having $\mathbf{x}_i$ as their nearest enemy (Leyva et al., 2015).

$$H(\mathbf{x}_i) = \frac{\sharp\{\mathbf{x}_j | \text{ne}(\mathbf{x}_j) = \mathbf{x}_i\}}{|D| - 1} \tag{22}$$

If $\mathbf{x}_i$ is nearest enemy of many instances, this indicates it is harder to classify and its harmfulness will be higher. The asymptotic computational cost of H is also $O(m_D \cdot n_D^2)$.

All measures are computed using the entire dataset. Concerning the computational cost of the measures, all of them are polynomials in the number of features and observations. Although the distance-based measures are the most costly, one must observe that the matrix of pairwise distances between all elements must be computed only once and it can be reused afterwards to compute all of these measures (namely *N*1, *N*2, *LSC*, *LSR*, *U* and *H*). The same reasoning applies to the measures that require building a decision tree model, which can be induced only once and have its information extracted for computing *DCP* and *TD*, and for measures based on the Naive Bayes classification rule (*CL* and *CLD*).

3.2 Algorithms and performance assessment

The candidate classification algorithms considered in this work are: Bagging (Bag), Gradient Boosting (GB), Support Vector Machines (SVM, with both linear and RBF kernels), Multilayer Perceptron (MLP), Logistic Regression (LR) and Random Forest (RF). They are representative of different learning paradigms commonly employed in the ML classification literature. But alternative algorithms can be easily added to the pool.

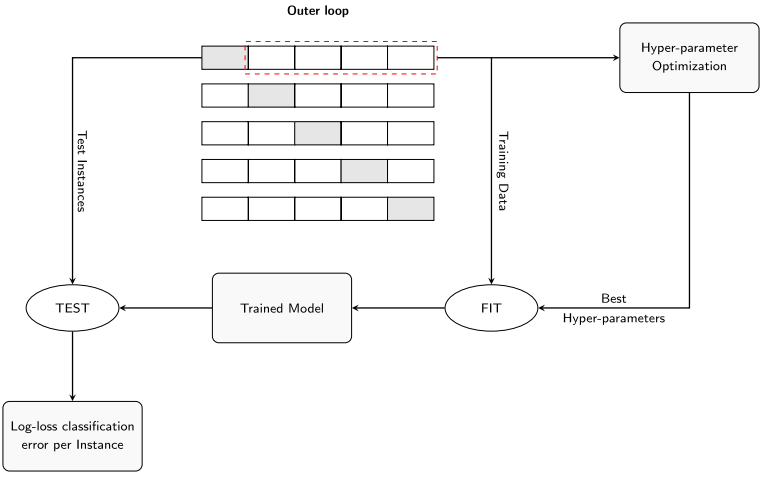


Fig. 3 Performance assessment process of the algorithms in $\mathcal{A}$

For assessing the classifiers' performances in a dataset D, we first split D into r folds ($r = 5$ by default) according to the cross-validation (CV) strategy, such that each instance belongs to only one of the r test sets. At each round, $r - 1$ folds are used for training the classifiers and the left-out fold is left for testing. Therefore, for each instance and algorithm combination we have one performance estimate. Repeating this process yields an interval estimation, which may be more reliable and can be set by the user.

At first, we could simply record whether the classification algorithms classify the instances correctly or not. But a more fine-grained evaluation can be obtained if the confidences the classifiers have in their predictions are considered. Therefore, we opted for a measure which takes into account the probabilities associated to each class, namely the log-loss or cross-entropy performance measure (Eq. (23)):

$$\text{logloss}(\mathbf{x}_i) = -\sum_{c=1}^{C} y_{i,c} \log(p_{i,c}), \tag{23}$$

where C is the number of classes the problem has, $y_{i,c}$ is a binary indicator of whether the class c is the actual label of $\mathbf{x}_i$ (1) or not (0) and $p_{i,c}$ is the calibrated probability the classifier attributes $\mathbf{x}_i$ to class c. Platt scaling is employed for calibrating the probability values (Platt, 1999; Böken, 2021).

An hyper-parameter optimization step was added in our setting, acting as an inner loop for each of the training sets of the outer CV. Within this inner loop, a candidate set of parameters is evaluated through cross-validation upon the training data from the outer loop. We employ a Bayesian optimization (Bergstra et al., 2011; Snoek et al., 2012; Bergstra et al., 2013) algorithm from a range of hyperparameter values for each classifier in the pool. The objective is to get closer to the best predictive performance achievable for the given data instances and classification algorithms. Within this inner loop, a candidate tuple of parameters is evaluated through cross-validation. Optimizing hyper-parameters is not so common in meta learning studies due to the high computational cost associated when many datasets and algorithms are used, but we consider that it can bring a significant improvement in classification performance. Nonetheless, one may opt for disabling the

hyper-parameter optimization when convenient. Figure 3 shows a schematic representation of the complete process described previously.

3.3 Feature selection

According to Muñoz et al. (2018), it is advisable to maintain just the most informative meta-features in the meta-dataset $\mathcal{M}$ before the IS projection is generated. We performed a supervised meta-feature selection in $\mathcal{M}$, based on the continuous response value $y(\alpha_j, \mathbf{x}_i)$, that is, the log-loss performance of the classifiers for the instances in $\mathcal{D}$, where α_j is the j-th classifier in the pool. Since there are seven classification algorithms in the pool, a ranking of meta-features for each one of them is obtained. Next, a rank aggregation method is employed to merge these subsets, as suggested in (Prati, 2012).

Taking the hypothesis that no previous knowledge about the data domain is available, a more general criterion for feature ranking is preferred. Information-theoretic methods offer this domain agnostic characteristic, being independent of any learning algorithm, and capable of capturing linear and non-linear relationships present in data (Li et al., 2017; Gao et al., 2015). A general formulation is presented in Eq. (24).

$$J(\mathbf{f}_k) = MI(\mathbf{f}_k;\mathbf{y}_j) + \sum_{\mathbf{f}_i \in \mathcal{S}_j} \text{eval}\big(MI(\mathbf{f}_i;\mathbf{f}_k), MI(\mathbf{f}_i;\mathbf{f}_k|\mathbf{y}_j)\big), \tag{24}$$

where $\mathbf{f}_k$ represents the k-th feature vector, $\mathbf{y}_j$ is the response variable for the j-th algorithm and $\mathcal{S}_j$ is the set of selected features for the j-th algorithm. The term $MI(\mathbf{f}_k;\mathbf{y}_j)$ is the mutual information, and $MI(\mathbf{f}_i;\mathbf{f}_k|\mathbf{y}_j)$ is the conditional mutual information. eval is an arbitrary function and different options for it will lead to different methods. The feature set $\mathcal{S}_j$ is initially empty, and the first feature chosen is the one showing maximum mutual information with the response variable $\mathbf{y}_j$. When a next feature is selected, according to its score $J(\mathbf{f}_k)$, it is added to $\mathcal{S}_j$, and the process continues until $|\mathcal{S}_j| = n_f$, a desired number of selected features is reached, in a forward feature selection process. By default, we set $n_f = 10$, which is the maximum recommended number of meta-features for the ISA projection tool.

The method employed for evaluating the meta-features is the *Minimum Redundancy Maximum Relevance* (MRMR), described in Eq. (25). It is a criterion that gradually reduces the effect of feature redundancy as more features are selected (Li et al., 2017). The rationale for this choice is that some meta-features may be redundant, since they are built on similar assumptions about the source of difficulty of the instances. There is a trade-off between minimizing the number of redundant meta-features in the selected set as an attempt to diversify it, and keeping the most relevant features. The MRMR method is an interesting choice, since it rejects redundant features at first, but tolerates the redundancy as it becomes more difficult to select informative features.

$$J_{\text{MRMR}}(\mathbf{f}_k) = MI(\mathbf{f}_k;\mathbf{y}_j) - \frac{1}{|\mathcal{S}_j|} \sum_{\mathbf{f}_i \in \mathcal{S}_j} MI(\mathbf{f}_k;\mathbf{f}_i) \tag{25}$$

Since we have seven different classification algorithms, there will be seven feature sets, $S_j (j = 1, \ldots, 7)$, ranked according to their importance as measured by Eq. (25). They are joined by a Instant Runoff Voting ranking aggregation method (Hillinger, 2004).[5] The

[5] Available in the Python rankaggregation package.

top n_f meta-features in this aggregated ranking are kept, whilst the remaining ones are discarded.

3.4 Instance space representation and footprints

Given the meta-dataset $\mathcal{M}$ with a selected subset of meta-features and the log-loss classification performances of the seven algorithms considered in this paper, further steps of the work employ the ISA functionalities of generating the 2-D IS and the footprints of the algorithms using the PyISpace implementation.[6]

We included in the former package a rotation step in order to standardize the interpretation of the IS projections. The rotation is performed so that the hard instances are always placed towards the upper left corner of the space, whilst the easier instances are placed towards the bottom right corner of the space. To achieve such a transformation in the instance space, a standard rotation as presented in Eq. (26) is applied, since the original IS is always centered at the origin. This rotation preserves the distances between instances as in the original instance space, so that there are no topological changes.

$$\mathbf{Z}' = \mathrm{R}(\theta)\,\mathbf{Z} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \mathbf{Z} \tag{26}$$

To proceed with this rotation step, we first need to find the angle of the original IS relative to the abscissa. For this, we consider the vector pointing to the centroid of the hard instances. In order to locate this centroid, we use the same binarized performance matrix $\mathbf{Y}_{bin}$ used for building the footprints of the algorithms (Sect. 2.2). This matrix indicates, for each instance and algorithm combination, whether a good or bad predictive performance was attained when compared to a threshold. Next, we calculate the mode of the categorization each instance has as either good or bad relative to the algorithms' performances. An instance for which the majority of the algorithms achieve a good predictive performance is categorized as easy. In contrast, the bad instances are those for which the majority of the algorithms do not attain a good predictive performance, corresponding to the hard instances. Once we know the location of the bad (hard) instances in the original IS, it is straightforward to find their centroid in this space. Lastly, the rotation angle is $\theta = 135° - \theta_{bad}$, where θ_{bad} is the angle of the vector pointing to the centroid of the instances for which a bad predictive performance was achieved most of the times. This angle assures that hard instances are placed towards the upper left of the IS and the easy instances are placed in the bottom right.

The definition of what constitutes a good and bad predictive performance of the classification algorithms according to the log-loss metric for each individual observation, required for building the binarized matrix $\mathbf{Y}_{bin}$, is based on the results of the following proposition, whose proof is enclosed in Appendix A:

Proposition 1 (cross-entropy bounds) *For any classification problem with C classes there is a lower bound* L_{lower} *and an upper bound* L_{upper} *for the cross-entropy loss* (*aka log-loss*) *such that*: *if* $\text{logloss}(\mathbf{x}_i) < L_{\text{lower}}$, *the prediction was correct*; *if* $\text{logloss}(\mathbf{x}_i) > L_{\text{upper}}$, *the prediction was incorrect*; *and if* $L_{\text{lower}} \leq \text{logloss}(\mathbf{x}_i) \leq L_{\text{upper}}$, *the prediction can be either*

[6] https://pypi.org/project/pyispace/.

correct or incorrect , *where* $\text{logloss}(\mathbf{x}_i)$ *is the log-loss of instance* $\mathbf{x}_i$. *Specifically*, *these bounds can be set as* $L_{\text{lower}} = -\log\left(\frac{1}{2}\right) = \log 2$ *and* $L_{\text{upper}} = -\log\left(\frac{1}{C}\right) = \log C$.

Therefore, if for an instance $\mathbf{x}_i$ the measured log-loss value is lower than $\log 2$, one can be certain this instance was correctly classified. On the other hand, a measured log-loss value larger than $\log C$ implies the instance was certainly misclassified. However, if the value of the log-loss metric falls in the interval between $\log 2$ and $\log C$, nothing can be said about the prediction, neither whether it was correct or incorrect. Based on the previous proposition, as a heuristic the log-loss performance of an algorithm for a given instance $\mathbf{x}_i$ is considered good if its value is lower than the harmonic average of $\log 2$ and $\log C$. The idea is to include as many correctly classified instances as possible, while avoiding the inclusion of too many misclassified instances, however stricter or smoother threshold values can be set if desired. Computationally, as defined in Eq. (23), the log-loss function calculation requires the actual label of the instance and the probabilities of classification assigned by a predictor to each of the classes.

Finally, we also introduce in this paper the concept of "instance easiness footprint", by taking the values from Eq. (7) as input and defining the easiness of an instance according to a threshold on IH, which is by default set as 0.4, implying that on average the probabilities assigned to the correct class of an instance by the pool of classifiers is 0.6, a value that can be made stricter or smoother if desired.[7] With such an approach, it is possible to obtain an indication of regions of the instance space in which the instances consistently receive a good classification score and are, therefore, easier to classify. As a footprint, objective measures of area, density and purity can be extracted from these regions. Therefore, larger areas are expected for datasets for which a larger portion of the instances across the instance space are considered easier.

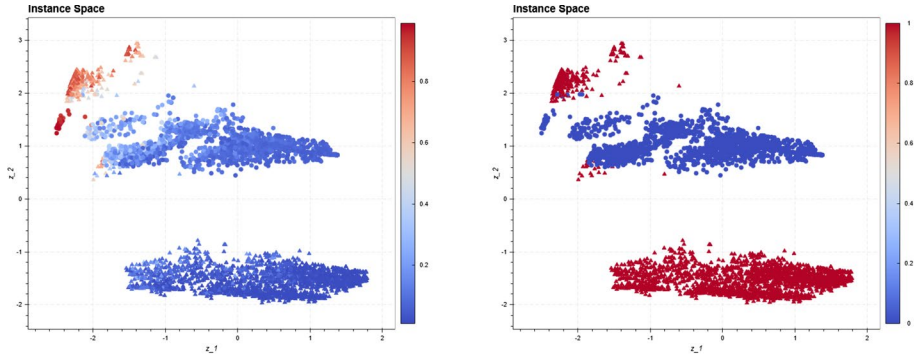
4 Experiments

One of the main contributions of our work, along with instantiating the IH-ISA framework as presented previously, is *PyHard*, a Python implementation of the framework, which is hosted in PyPI.[8] PyHard is a self-contained solution, which encapsulates the ISA framework and runs an application to explore the results visually and interactively. This visualization app allows the user to interact with the IS of a dataset and select regions to be further inspected by interactive plots or saved. Throughout the configuration file, it is possible to choose the set of meta-features, the algorithm portfolio, to enable and to configure the hyper-parameter optimization and feature selection steps. In this section we show the potential insights obtained by constructing the ISA of a dataset through case studies.[9]

[7] Recalling that IH takes the average likelihood of misclassification of an instance $\mathbf{x}_i$ by the multiple algorithms in the pool $\mathcal{A}$, the threshold on IH cannot be based on Proposition 1.

[8] https://pypi.org/project/pyhard/.

[9] The benchmark datasets and outputs of their analysis are available at the shared url https://gitlab.com/ita-ml/pyhard. The real-data COVID dataset in the repository had the features names uncharacterized for anonymization.



(a) ISA of hospitalization dataset: IH. **(b)** ISA of hospitalization dataset: classes.

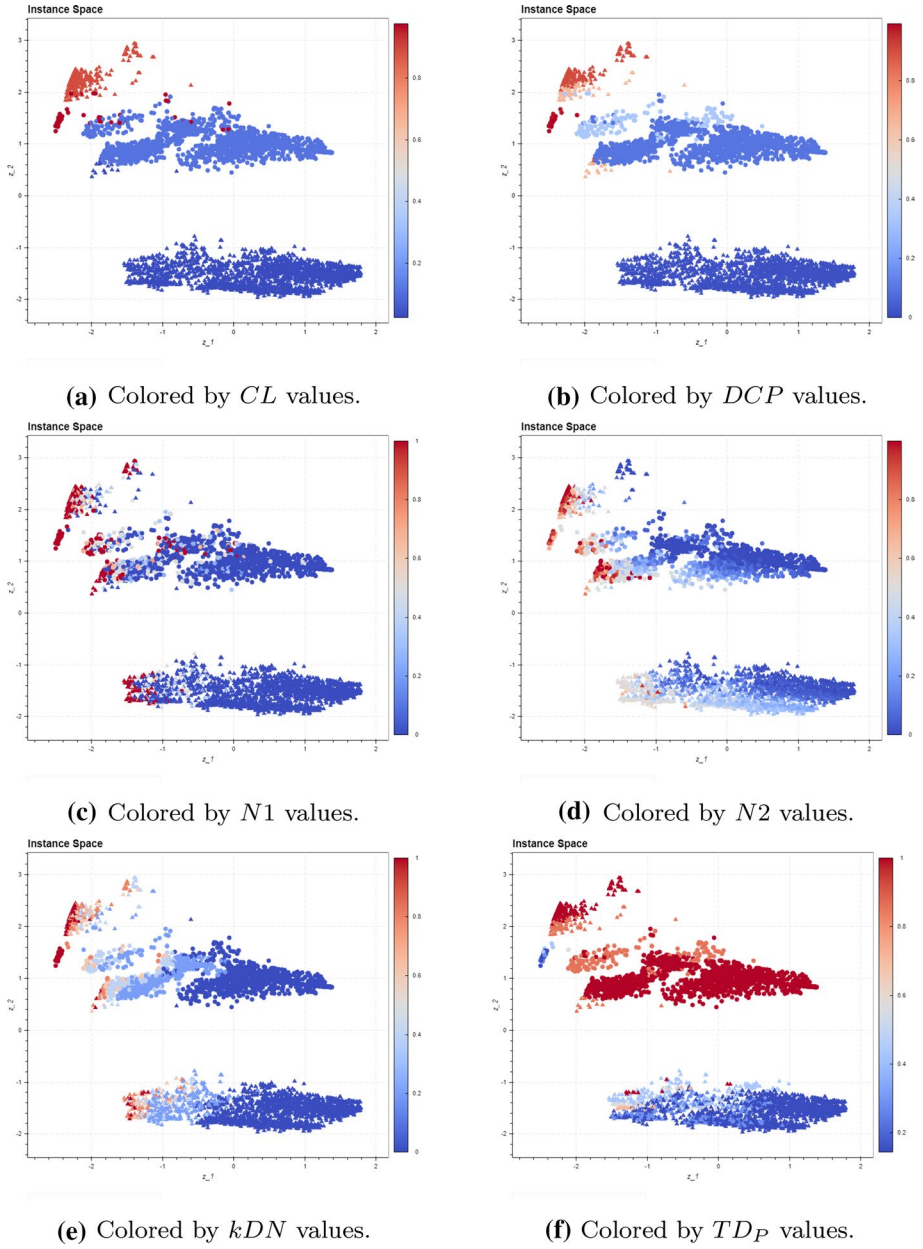
Fig. 4 ISA for the hospitalization dataset, with points colored according to (a) IH values (blue is easy, red is hard) and (b) classes (blue is non-hospitalized, red is hospitalized)

4.1 Case Study: ISA for inspecting a COVID prognosis dataset

Whenever an individual is tested for COVID-19 diagnosis in the Brazilian territory, some information must be entered into specialized government systems. They include the presence or absence of symptoms commonly associated with COVID-19 and comorbidities which can impact the severity of the cases. The São José dos Campos municipal health department gathers this information and joins outputs from multiple governmental systems in order to follow up on cases and formulate public health strategies. This is a large city from the São Paulo state with a population around 750,000 inhabitants, an industrial economy and a high human development index according to Brazilian standards. In a partnership with the health department of this city, part of this data was formatted for predictive analysis to support public health decision making.

Here we present an analysis of a dataset containing anonymized data from citizens diagnosed with COVID-19, collected from March 1st, 2020 to April 15th, 2021. The dataset involves predicting whether a citizen will require hospitalization or not taking as input the following attributes: age, sex, initial symptoms (fever, cough, sore throat, dyspnea, respiratory distress, low saturation, diarrhea, vomit and other symptoms) and comorbidities (chronic cardiovascular disease, immunodeficiency-immunodepression, chronic kidney disease, diabetes mellitus, obesity, chronic respiratory diseases and other risks). The idea is to take information routinely supplied during COVID testing for estimating the amount of resources from the city's health system that may be required, supporting public heath management policies.

The "hospitalization" dataset used here has data from 5,156 citizens, half of which were hospitalized. Our objective is to analyze the hardness profile of this dataset and to extract some insights from the visualization and interaction with its IS. Figure 4 presents the ISA of the hospitalization dataset. Each point corresponds to an observation of the dataset, that is, a confirmed COVID case. In Fig. 4a, the observations are colored according to their IH values, with harder observations colored in red and easier observations colored in blue. Using our rotation step, the hard instances are concentrated in the upper left of the plot and easier instances are placed towards the bottom right. In Fig. 4b the same observations are colored according to their original labels, where red points correspond to hospitalized



(a) Colored by CL values.

(b) Colored by DCP values.

(c) Colored by $N1$ values.

(d) Colored by $N2$ values.

(e) Colored by kDN values.

(f) Colored by TD_P values.

Fig. 5 ISA for the hospitalization dataset, colored according to some of the meta-features values

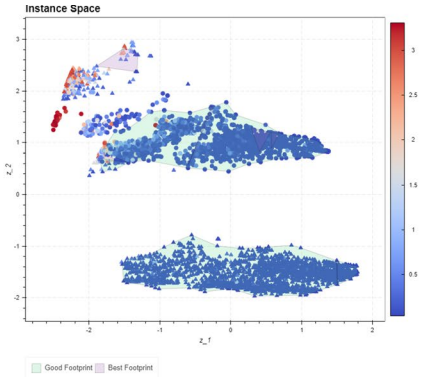
citizens and blue points are non-hospitalized citizens. It took 17 minutes and 27 seconds to build this IS on a laptop with an Intel i75500U processor with 2.40 GHz using 8 GB of RAM, running Ubuntu version 20.04.

The combined analysis of these plots already provides us with some interesting insights: most of the observations are easy to classify correctly by most of the algorithms, but the

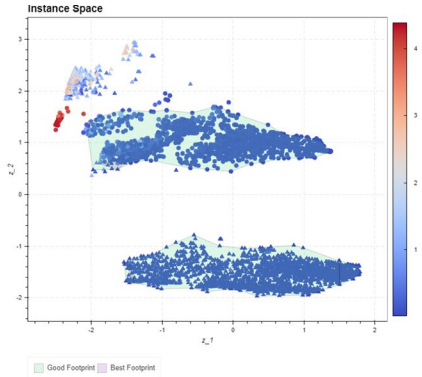
group of hospitalized citizens in most of the cases has either an easy profile, being placed in the bottom of the IS, or a hard profile, being colored in red regarding IH. In contrast, patients who were not hospitalized had mostly an intermediate hardness level, which contains observations of low and medium IH values. Nonetheless, there is an specific cluster of non-hospitalized subjects that were very hard to classify correctly (with z_1 coordinates lower than -2 and z_2 coordinates between 1 and 2), which will be referred as "anomalous non-hospitalized group" (acronym ANH) hereafter. The hospitalized individuals placed near the instances of the non-hospitalized class in the bottom left of the intermediate cluster of points in the IS (with z_1 coordinates lower than -1 and z_2 coordinates between 0 and 1) are also worth investigating, since their hardness profile is more similar to that of observations of the opposite class. They will be referred as "anomalous hospitalized group 1" (AH1) from here on. A second group of interest from the hospitalized class is composed of the hard instances in the top of the IS (with z_1 coordinates between -2.5 and -1 and z_2 coordinates between 2 and 3) and will be named "anomalous hospitalized group 2" (AH2), since most of the data from this class is regarded as easy. Note that the anomalous term refers to the expected hardness profile of the instances of both classes, as evidenced by the ISA projection.

Figure 5a to f show the IS projection of the hospitalization dataset colored according to the values of some of the meta-features used, those which were more explanatory of the hardness profile of the data. The following interesting aspects can be highlighted:

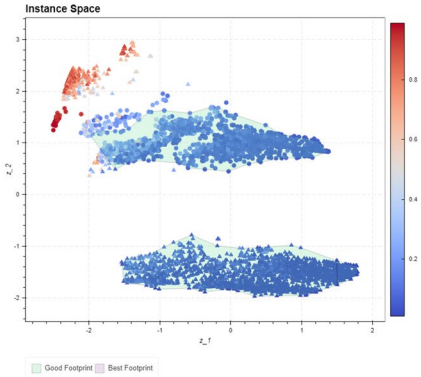
- Many of the observations with high IH values at the top of the IS have also a low likelihood of belonging to their own classes (as measured by *CL* in Fig. 5a) and are placed in disjuncts with elements which do not share their labels (measured by *DCP* in Fig. 5b). They include mostly instances from the ANH and AH2 groups, although the *DCP* measure also highlights the instances from the AH1 group as hard. The high values for these measures provide evidence that the observations from these groups have characteristics which overlap with those from the other classes;
- Most of the instances with high IH values at the top of the IS are also close to elements from the opposite class (measured by *N*1 in Fig. 5c), have lower inter class distance compared to their intra class distance (measured by *N*2 in Fig. 5d) and have a high proportion of nearest neighbors with labels which differ from their own (measured by *kDN* in Fig. 5e). But there are also other hard observations within the left borders of the different groups of instances in the ISA according to these measures. They can be borderline cases, that is, observations near the decision frontier required for separating the classes. The groups ANH and AH1 are highlighted as hard according to these measures too, but the group AH2 has mixed results. *N*2 in particular indicates that some of the observations from AH2 are closer to an instance of their class than to their nearest neighbors from another class;
- The TD_P values are high for instances from the non-hospitalized class with exception of those in the ANH group. They are also high for a subset of the instances from the hospitalized class, mostly those in the AH2 group. Interestingly, the cluster ANH has a low TD_P value. This measure involves building a pruned decision tree from the data and we can see that the observations from this group are classified at depths similar to those of the hospitalized observations. The same happens for the AH1 and AH2 instances, which are placed at depths similar to those of the observations of the non-hospitalized class. Combining these results to those of the previous meta-features, we can infer these groups probably contain noisy or outlier instances, which have input data characteristics similar to that of the opposite class.



(a) Colored by Bagging log-loss performance.



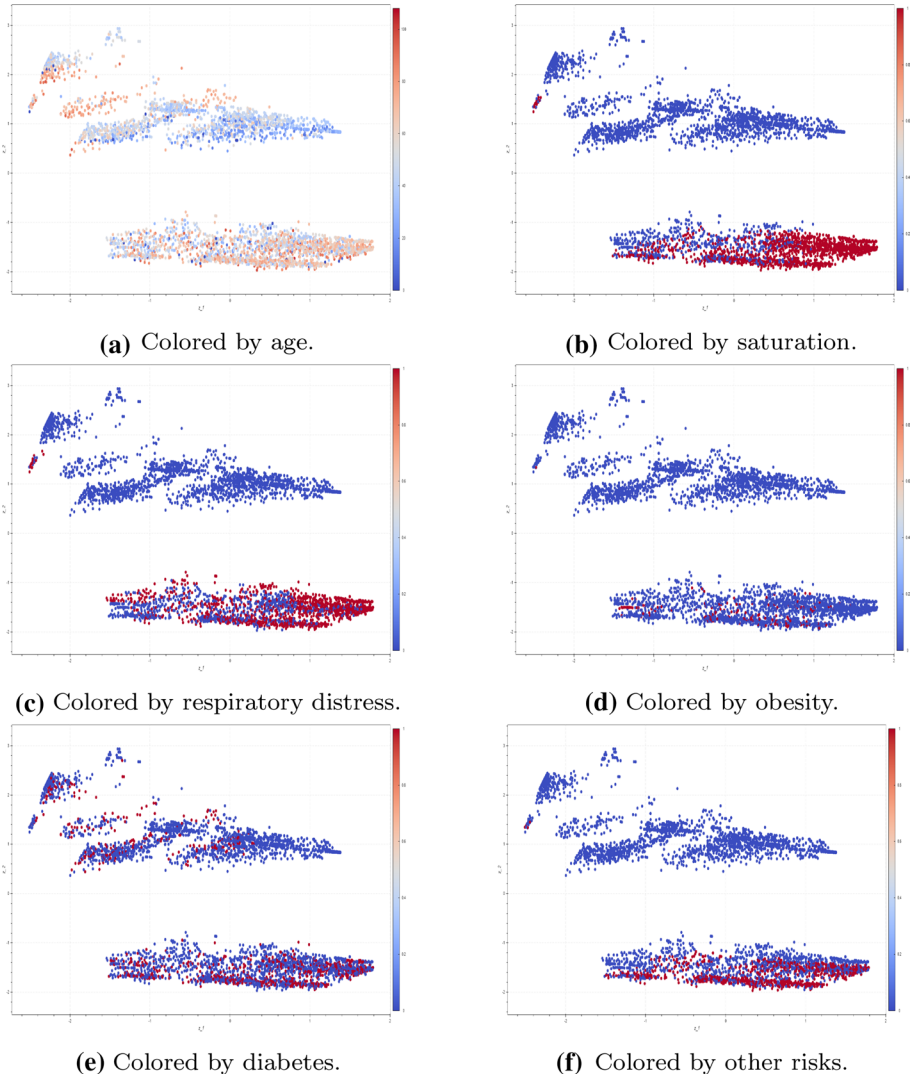
(b) Colored by MLP log-loss performance.



(c) Colored by IH values.

Fig. 6 ISA for the hospitalization dataset, colored according to the algorithmic performances of two algorithms and IH values. Footprints are also shown, in green (good) and purple (best). Observations with higher log-loss error and IH values are colored in red and observations with lower log-loss errors and IH values are colored in blue

In Fig. 6 we present the good (in green) and best (in purple) footprints of the classifiers which attained the largest and the smallest footprint areas in the IS built for the hospitalization dataset. The easiness footprint encompassing the instances which are easier to classify by most of the algorithms of the portfolio is also shown in Fig. 6c. Bagging (Fig. 6a) had a normalized footprint area of 0.856, with density of 1.055 and purity of 0.972. The MLP (Fig. 6b) had a normalized footprint area of 0.926, with density of 1.007 and purity of 0.992. Therefore, MLP showed a good predictive performance in a larger area of the IS, which was also slightly purer and encompass good instances in 99% of the cases. Indeed, the MLP showed a good predictive performance for most of the instances except from those of the ANH and AH2 groups, while Bagging was not so consistent for instances in the non-hospitalized class. But it is interesting to notice that Bagging had some areas of best performance for some hard instances in the top of the IS when compared to other algorithms. In fact, Bagging was the algorithm with largest best normalized area in our portfolio. The easiness footprint area for this dataset is 0.859, with a density of 1.06 and a purity of 0.998 and encompasses only the easiest instances from both hospitalized and non-hospitalized



(a) Colored by age.

(b) Colored by saturation.

(c) Colored by respiratory distress.

(d) Colored by obesity.

(e) Colored by diabetes.

(f) Colored by other risks.

Fig. 7 ISA for the hospitalization dataset, colored according to some of the raw attribute values

classes. This excludes all instances from groups ANH, AH1 and AH2. Therefore, most of the observations from the hospitalization dataset are easy to classify, except for those in the former groups, which are more challenging. Recommending a particular algorithm for new observations is beyond the scope of this paper, but one might expect an algorithm to perform well for instances of similar characteristics to those encompassed in its footprint, given its high purity level. The PyHard tool allows users to save and to inspect the characteristics of the instances from a selected footprint for supporting such studies.

The PyHard tool also allows to plot the values of the raw input attributes along the IS. Figure 7 shows the distributions of some of the attributes with interesting patterns in the IS which can help to understand the hardness profile of the data. Age is a well known feature with influences on COVID severity, impacting hospitalization. According to Fig. 7a, older

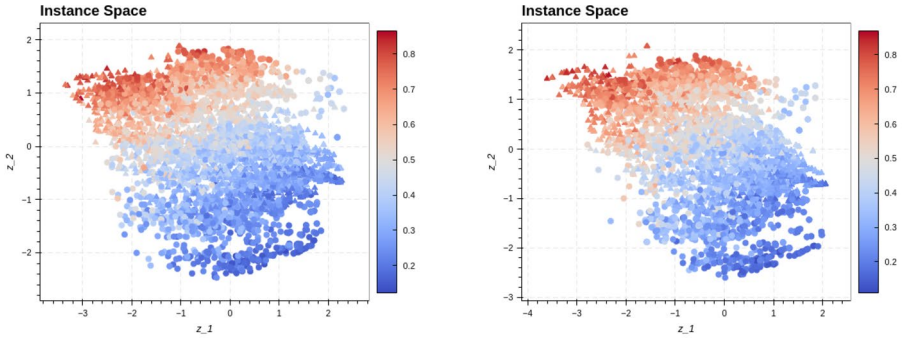
people (in warmer colors) are predominant in the easy group of hospitalized individuals, but also imposes some higher level of difficulty in classifying some of the non-hospitalized individuals. Indeed, the easiest cases of non-hospitalized citizens tend to be younger. Some hospitalized people from the AH2 group, which are hard to classify, are also younger, while from the groups AH1 and ANH are older. According to the medical literature (Barek et al., 2020), older people tend to evolve to worse cases of COVID, therefore one might expect cases requiring hospitalization to be more frequent among elderly people. This can partially explain why some instances from the ANH and AH2 groups are hard to classify, since they have conflicting age patterns to what is commonly expected.

All other raw attributes analyzed are binary, indicating that if a citizen has either reported the presence of some symptom (colored in red) or did not report such symptom (colored in blue), such as low saturation (Fig. 7b) and respiratory distress (Fig. 7c) or some comorbidity, namely obesity (Fig. 7d), diabetes (Fig. 7e) and other risks (Fig. 7f). All patients who did not require hospitalization, except from some from the ANH group, had no saturation or respiratory distress issues and did not report obesity or other risks. The ANH group had some patients with low saturation, respiratory distress and other risks reported (there is also one case of obesity in this group), which can influence why they are harder to classify, since these are patterns of symptoms and comorbidities observed more commonly in hospitalized cases. In contrast, the individuals from the AH1 and AH2 groups have a good saturation, no respiratory distress, no obesity and no other risks. Therefore, they also have contradicting patterns regarding these aspects while requiring hospitalization. Diabetes did not influence much on the difficulty level of the instances and individuals with and without diabetes are evenly distributed in the IS. This also happens for other features, which are omitted here.

Our analysis allows us to highlight some groups of observations from the dataset with conflicting raw attributes values considering their expected outcomes. These are cases worthy of closer examination by a domain expert and may either correspond to outliers or were wrongly labeled.

Based on these insights, follow-up investigation of the raw data from the observations in ANH identified three groups of individuals: people who were cured and did not require hospitalization, despite their symptoms and comorbidities; people who quickly progressed to death and did not seek hospitalization in time; and people who were actually hospitalized, but missing information on their hospitalization date has lead to an erroneous data labeling. While the first two groups were correctly labelled and can be considered outliers, the latter are incorrectly labeled and should be discarded from the dataset or corrected for building a more reliable prediction model. The group AH1 has many individuals with few and mild symptoms which were hospitalized. Whilst they are anomalous regarding the expected characteristics from the hospitalized class, it is possible that their forms have missing information on some of the symptoms, bringing noise to the input attributes values. AH2 can also present some noise in the input attributes, but they are correct yet atypical observations considering the general patterns of the input attributes from the other observations of the class they belong to.

Summarizing, the prediction of hospitalization risk from standard information collected from forms filled by the population when they are tested is practicable and can be used to support public health decision making. Nonetheless, some of the observations may have been wrongly filled or are incomplete, which impacts the hardness level achieved in their classification. There is a general need for more careful data collection in Brazil, which is often neglected but can be of great value in fighting the pandemic in one of the most affected countries worldwide. Our tool allows us to highlight such problematic



(a) COMPAS dataset ISA projections with `race` as an input attribute.

(b) COMPAS dataset ISA projections without `race` as an input attribute.

Fig. 8 COMPAS dataset ISA projections with and without `race` as an input attribute, colored according to the IH value for each instance

observations and to explore the main reasons why they are hard to classify, whether due to labelling errors in the inputs, outputs or anomalies. Such insights offer a richer analysis as the foundation for building trusted ML predictive models in practical and critical contexts.

4.2 Case Study: ISA for detecting algorithmic bias using the COMPAS dataset

This section builds an IS for the COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) dataset.[10] It presents data regarding crime recidivism and is commonly employed in the literature for analyzing sample and algorithmic biases (Khademi & Honavar, 2020). The dataset has 5,278 instances and 13 input attributes, where 47% of the instances are from the two-year recidivist class and is quite balanced. Some of the attributes are nominal, but since they are binary, we treat them as numerical flags of either 0 or 1 values.

Algorithmic bias can occur whenever sensitive input attributes from a dataset influence the predictive results when they should not. In the case of recidivism data, aspects such as race and gender are protected attributes and should not be decisive in determining if an offender is likely to commit new crimes (Corbett-Davies & Goel, 2018). In this case study we show how ISA can be used to identify potential biases in the errors of the predictors induced by the COMPAS dataset. For this, we will compare how low IH and mainly high IH instances differ when running the analysis with `race` as an input attribute, and when removing it for being a protected attribute. Our reasoning for this is that even though a protected attribute is not used to train a model, it may still influence the model indirectly due to an inherent data collection bias. If we train a model without protected attributes, but evaluate that there are differences within the error rates for different races or genders, then we can suppose that bias is present.

In one analysis, we employed all 13 attributes of the COMPAS dataset as input to the ISA, while in the other we removed `race_African-American` and `race_Caucasian` from the attributes list used as input to the classifiers. As a result, two instance

[10] https://www.openml.org/d/42192.

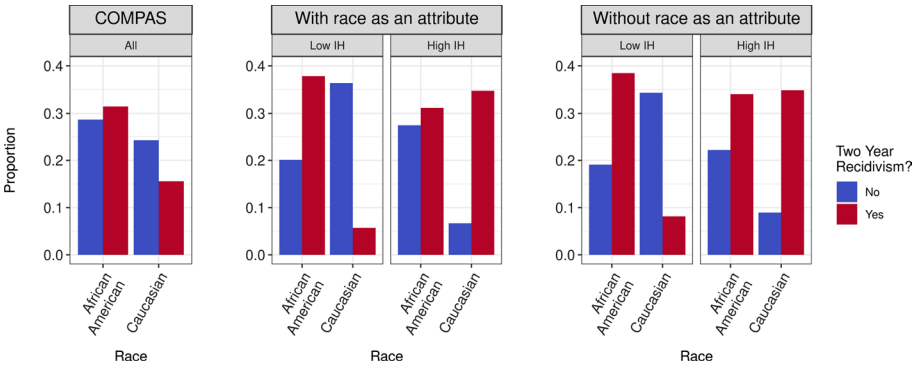


Fig. 9 Distributions of attributes `race_African-American` and `race_Caucasian` for the entire dataset (left) and in different data selections on ISA built with (center) and without (right) `race` as an attribute

spaces are produced and analyzed. Figure 8 presents the instance spaces of the COMPAS dataset with (in Fig. 8a) and without (in Fig. 8b) `race` as an attribute, where the points are colored according to their IH values. The IH values are quite linearly distributed in each IS, and they clearly delimit areas of good and poor predictive performance. The IS projection of Fig. 8a is similar to that of Fig. 8b, offering the first evidence that the hardness profile of the dataset is maintained despite the usage or not of the race attribute. It took 20 minutes and 35 seconds to generate the IS of the original COMPAS dataset and 19 minutes and 26 seconds to obtain the IS of the dataset counterpart which eliminates the race attribute. The experiments were run on a laptop with an 2.40 GHz Intel i75500U processor using 8 GB of RAM, running Ubuntu version 20.04.

Using PyHard's Lasso tool we manually selected regions of good/easy observations (blue points in the bottom of the IS) and bad/hard observations (red points in the top of the IS) from both instance spaces and compared their distributions. These selections are presented in Appendix B. The interpretation of these selections is as follows: an area with points predominantly in red represents instances that were most likely misclassified, whereas an area with points predominantly in blue represents instances which were most likely classified correctly. Therefore, in the case of binary classification, a selection of good points is a surrogate for true positives plus true negatives ($TP + TN$), and a selection of bad points is a surrogate for false positives plus false negatives ($FP + FN$). Those sums can be further broken down using class information, if necessary. The focus of our analysis will be on the most relevant attributes of the dataset considering potential algorithmic bias: `race_African-American`, `race_Caucasian`, and in addition `priors_count`, `age` and `sex`.

First, we show in Fig. 9 the distribution of race values considering the entire dataset and the selections with low IH and high IH values, respectively, with and without `race` as an attribute. Comparing the distributions, what stands out the most is the fact that the graphs present an inversion in the proportion of the classes (recidivism) when comparing low IH and high IH values. That is, observations from the no-recidivism class are more frequent for low IH values and observations of the recidivism class are more frequent in the high IH selections. This provides evidence of greater difficulty in classifying repeat offenders. The tendencies when `race` is and is not used as an attribute are also similar in the plots, with an exception seen for instances with low IH values of

Table 2 Summary statistics for high IH instances of both analyses of the COMPAS dataset, with and without `race` as an input attribute

Class	Recidivist stats	With `race`	Without `race`
Recidivist	Caucasian (%)	52.7	50.6
	Men (%)	77.3	78.8
	Avg Age	37.6	37.1
	Avg Prior Offenses	1.3	1.4
Non-recidivist	African American (%)	80.4	71.2
	Men (%)	92.4	90.2
	Avg Age	30.1	30.0
	Avg Prior Offenses	6.9	6.9

Table 3 Aggregated confusion matrix based on the binarized log-loss performance of the algorithms in the pool $\mathcal{A}$, for two scenarios: with `race` as an input attribute (left) or without `race` as as input attribute (right)

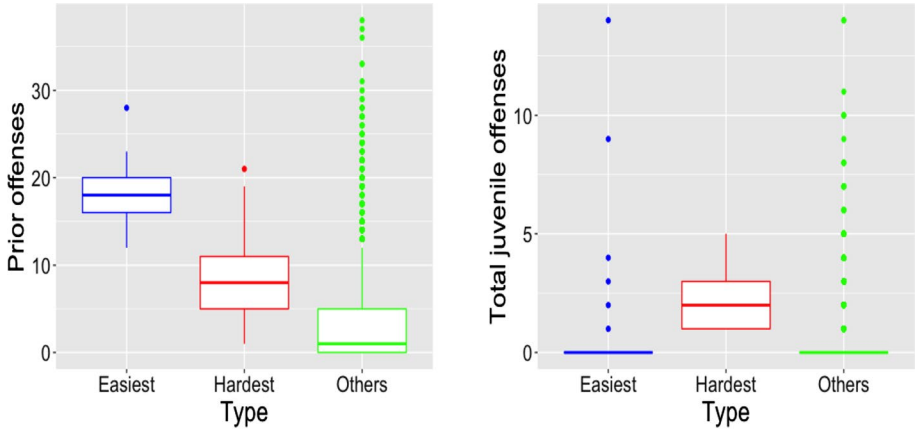
	With `race`		Without `race`	
	African American (%)	Caucasian (%)	African American (%)	Caucasian (%)
FP	16.0	8.8	14.6	9.4
FN	16.1	24.2	17.1	23.2

the `race_African-American` where the proportion of non-recidivists is lower than that of recidivists.

Interestingly, the proportions of low IH instances per class and per race in Fig. 9 are very similar despite the usage or not of `race` as an input attribute. High IH observations have some minor variations, with an increased proportion of instances from the recidivism class for African American individuals when the `race` attribute is disregarded. This is an additional indication that the hardness profile is very similar for both scenarios, that is, with and without taking the `race` attribute as input to the classification models.

In Table 2 we focus on the extract of hard to classify instances and show the summary statistics of recidivists and non-recidivists instances with high IH values, for both classification scenarios, namely with and without using `race` as an input attribute. Plots with the distributions of the same attributes per class are presented in Appendix B. Among the instances with high IH value, the actual recidivists (`two_year_recid` = *Yes*) have a lower average of prior offenses which leads them to be wrongly predicted to be non-recidivists by the classification models. However, it is actually more interesting to check non-recidivists (`two_year_recid` = *No*) with high values of IH, because they are a surrogate for FP. In practice, this can lead to the conviction of a person who is in fact innocent if the classification models are used. In this group, the instances represent, on average, younger male African-Americans with a higher average number of prior offenses than the average of the non-recidivists along the entire dataset, which also explains why they are harder to classify.

We notice that the classification algorithms continue to be biased for non-recidivist high IH instances even when the `race` attribute is omitted from the set of input attributes. The average profile of the hard instances from the recidivists and non-recidivists is maintained, despite a reduction in the percentage of instances with `race_African.American = 1`. Therefore, whilst FN is most likely for individuals of the Caucasian race, FP has occurred more frequently for the African American individuals.



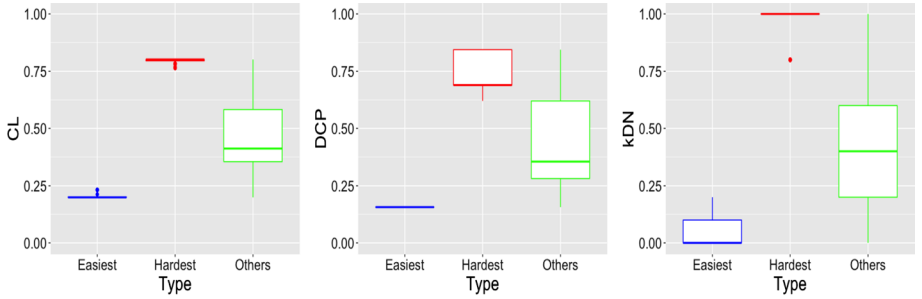
(a) Prior offenses. **(b)** Juvenile offenses.

Fig. 10 Boxplots of the raw feature values referring to prior and juvenile offenses for three sets of instances of the COMPAS dataset without `race` as an input attribute: the easiest to classify (in blue), the hardest to classify (in red) and all the other instances (in green)

Algorithmic fairness occurs when groups of instances pertaining to the same protected attribute have the same chance of being misclassified, that is, both groups have the same FP and FN rates (Corbett-Davies & Goel, 2018). Therefore, even when disregarding the race attribute in the classification models, there is no fairness concerning this sensitive attribute.

As an additional validation of our previous analysis, we have evaluated the FP and FN rates using an aggregated confusion matrix, taking into account the outputs of the multiple classifiers in our pool $\mathcal{A}$. To build this matrix, we count, for each instance, how many algorithms in our pool have achieved a good log-loss predictive performance according to the threshold specified in Sect. 3.4. This information can be obtained by summing up the rows from matrix $\mathbf{Y}_{bin}$. If the majority of the algorithms achieved a bad log-loss predictive performance for an instance, an incorrect prediction is accounted. By relating this information with the true class of the instance, we can identify whether it was a FP or a FN. Next, we have decomposed the FP and FN errors by race, as shown in Table 3. Confirming what was observed in our previous lasso selections, FP is more common among the African American individuals, even when the `race` attribute is explicitly disregarded. The FN rates are slightly superior for the Caucasian individuals, but this difference is smaller than that observed for the FP rate.

We now further refine our analysis by focusing the Lasso selection tool on two specific regions of the ISA of the COMPAS dataset without `race` as an input attribute (projection shown in Fig. 8b). The reason for analyzing this projection only is that the recommended procedure for avoiding undesirable biases is to delete protected attributes from the input set of the ML techniques. Two selections of the given ISA are examined: (i) the easiest to classify instances (with z_1 coordinates between 1 and 2 and z_2 coordinates between -3 and -2); and (ii) the hardest to classify instances (with z_1 coordinates between -4 and -3 and z_2 coordinates between 1 and 2). These areas were chosen based on the fact that the more difficult instances are placed towards the upper left corner of the IS, whilst the easier instances are in the bottom right corner.



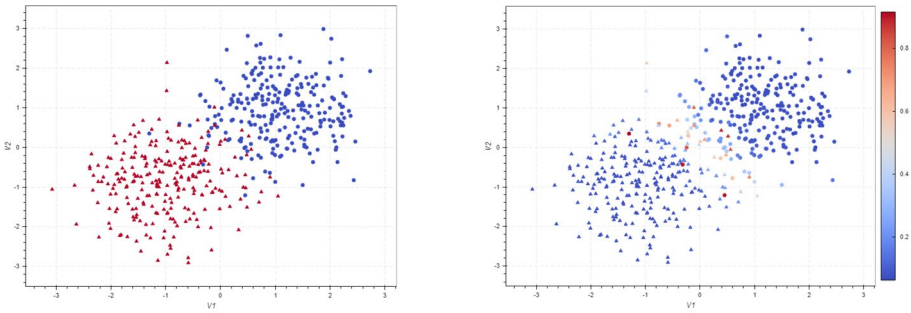
(a) CL values. **(b)** DCP values. **(c)** kDN values.

Fig. 11 Boxplots of some of the meta-features values for three sets of instances of the COMPAS dataset without `race` as an input attribute: the easiest to classify (in blue), the hardest to classify (in red) and all the other instances (in green)

The easiest data excerpt contains 27 instances, all of whom are male recidivists (`two_year_recid` = *Yes*) with less than 45 years of age. 88.9% of the individuals of this group are African Americans. They also have a quite high number of prior offences, which varies between 12 and 28 (median of 18). Therefore, the ML algorithms in our pool found it easier to classify correctly young male recidivist individuals with a high number of prior offences, the majority of whom are also African Americans.

The hardest selection has 25 male non recidivist individuals (`two_year_recid` = *No*). Most of them have less than 45 years of age too, except for one individual with 51 years of age (who has a high number of prior offenses and two juvenile convictions). The number of prior offences varies between 1 and 21 (median of 8) and 76.0% are African American individuals. All individuals in the hardest set had prior juvenile convictions registered too, while this characteristic was not observed for the majority of the easiest to classify instances (21 out of the 27 easiest instances had no juvenile offenses registered). Therefore, in general the algorithms faced difficulties in classifying young male individuals who were not labeled as recidivists but have prior offences registered in their adult or juvenile criminal records. The race attribute had a lower prominence in this group than that verified for the easiest set. What stands out the most in the hardest to classify observations are the unexpected raw features values which contradict the overall patterns of the non-recidivist class, especially regarding the `priors_count` (number of prior crimes committed), `juv_fel_count` (number of juvenile felonies), `juv_misd_count` (number of juvenile misdemeanors) and `juv_other_count` (number of other prior juvenile convictions) attributes.

Figure 10 contrasts the raw feature values referring to prior and juvenile offenses of the easiest (in blue), hardest (in red) and all the other instances (in green) of the COMPAS (without `race`) dataset. The plot from Fig. 10b sums up the counts of the `juv_fel_count`, `juv_misd_count` and `juv_other_count` attributes as a total count of juvenile offenses registered for each individual. As shown in Fig. 10a, the easiest data excerpt contains individuals with high numbers of prior crimes (all recidivists) and the others data excerpt has in general minor amounts of prior offenses (although there are many outliers, since this set contains individuals from both recidivist and non-recidivist classes). The hardest to classify data excerpt has an intermediary number of prior offenses registered, despite being originally labeled as non-recidivists. Taking the total number of juvenile offenses (Fig. 10b), the contrast is more evident. Whilst the "easiest" and "others" data excerpts have in general a



(a) Base 2normals dataset, colored by class. **(b)** Base 2normals dataset, colored by IH.

Fig. 12 ISA for the base 2normals dataset, with points colored by class (left) or IH (right)

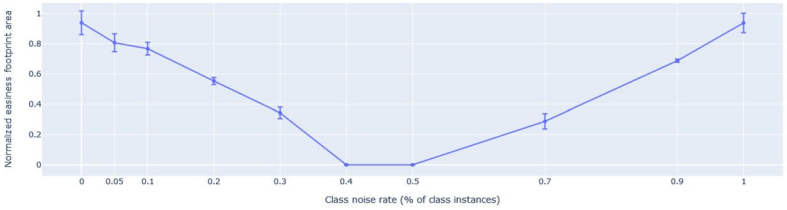
low number of juvenile offences (with first, second and third null quartiles), the hardest to classify individuals clearly have a pattern of more juvenile offences registered.

The values of some of the meta-features (boxplots from Fig. 11) in the easiest to classify (in blue), hardest to classify (in red) and others (in green) set of instances also evidence the contradicting patterns of the hardest to classify instances. The hardest set shows: high *CL* values (Fig. 11a), indicating the instances in this set have a low likelihood of pertaining to their class; high *DCP* values (Fig. 11b), indicating that such observations tend to be placed in disjunctures with a majority of examples from the other class; and very high *kDN* values (median at the maximum of 1.0, Fig. 11c), indicating that most of these observations are close and surrounded by instances from the other class. In contrast, the easiest to classify instances show low hardness measures values, whilst the other instances have intermediary hardness measures values. All of these results corroborate that the hardest to classify instances are either noisy or very atypical. As observed in our refined analysis, Rudin et al. (2020) also report data inconsistencies in COMPAS and the risk of dangerous individuals being released to society.

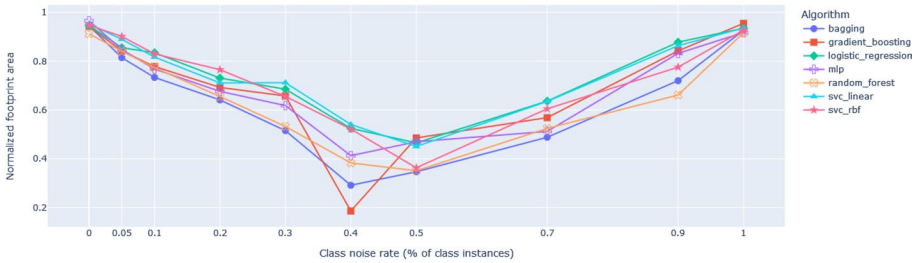
The analysis performed on the COMPAS dataset shows how ISA can be used for analyzing algorithmic bias, through the inspection of the predominant characteristics of groups of instances which are hard to classify and surrogates for FN and FP. Particularly, for the COMPAS dataset we can notice that even when the sensitive attribute is omitted from the input set, the hardness profile of the instances is similar to that observed when the whole set of attributes is used. In both cases, the FN and FP rates differ for different populations, pointing out an intrinsic bias regarding the `race` attribute. But a closer inspection of the hardest to classify instances when contrasted to the easiest to classify instances also evidence that other types of problems can be present in this dataset, such as data inconsistencies. It would be worth investigating how the usage of fairness-enhancing strategies (Friedler et al., 2019) and data cleasing strategies can change the hardness profiles identified in the COMPAS dataset when using ISA.

4.3 Case study: ISA of datasets with label noise

Real world datasets are subject to data quality issues, such as the presence of noise due to errors in data collection, storage and transmission. According to Zhu & Wu (2004), a classification dataset has two possible types of noise: in the predictive input features or in the



(a) Instance easiness footprint average area for different noise levels.



(b) Classifiers average footprint area for different noise levels.

Fig. 13 Instance easiness and classifiers' footprint areas for different noise levels

target attribute. The later type of noise, also known as label/class noise, usually implies more severe problems for the supervised ML techniques, since they rely on optimizing some loss-function based on reproducing the labels of the training data (Garcia et al., 2015). Here we study the effects of the presence of different levels of label noise in the ISA of a dataset. For such, we use a controlled synthetic dataset for which the absence of label noise can be granted and progressively introduce random label noise to it, at increasing rates.

Figure 12a shows the base dataset used in these experiments. It is generated using the mlbench package from the R language and consists of two classes described by Gaussians with spread of 0.8, with 250 data items each. The classes are fairly linearly separable, with a little degree of overlap. Figure 12b presents the same dataset with data items colored by IH values. We can notice that the hardest instances in the original space are those in the boundary and overlap of the classes.

Taking the dataset from Fig. 12a as base, we randomly flip the class labels at the following rates: 5%, 10%, 20%, 30%, 40%, 50%, 70%, 90% and 100%. Since the choice of the examples to be corrupted is random, this process is repeated 10 times for each noise rate. Although high noise rates are quite unrealistic, studies suggest that even controlled real datasets have at least 5% of noise rate (Maletic & Marcus, 2000).

The ISA is run for the original dataset and for the corrupted versions, using default hyperparameter values for the classifiers in order to reduce the computational burden. Next, we computed the average areas of the easiness footprints and of each of the seven classification techniques considered in this work. Figure 13a shows the average easiness footprint area registered in the instance spaces produced for the different noise levels. Recalling that the easiness footprint corresponds to regions of the IS containing instances that are easier to classify, as expected the easiness footprint areas decrease as more noise is introduced

until reaching the 40-50% noise levels. For 40 and 50% of noise levels, the footprint areas of instances for which a consistent good performance is reached is null. This clearly shows how the IH values and footprint areas are able to reflect the harmful effects of label noise in classification performance. From this point on, the classification problems become complementary to those of the lower noise levels. In the extreme case of 100% noise level, the labels of all data points are flipped, so that the classes are inverted and the classification problem is therefore equivalent to that of the noiseless version.

The same effect can be verified in the footprint areas of the classification techniques (Fig. 13b). Indeed, the regions of the IS where the techniques have a good classification performance is reduced for increasing noise levels until the 50% level is reached, when the areas begin to increase again. But it is also interesting to notice that some of the classification techniques are more robust than others to the presence of the different noise levels. For the original dataset, all classification techniques are quite effective, as we have a simple classification problem. This includes the linear models (Logistic Regression and linear SVM), since this is a fairly linearly separable problem. For increasing noise levels, the performance of the ensemble techniques (Bagging, Gradient Boosting and Random Forest) degrades to a large extent, whilst the linear predictors remain quite robust. Boosting algorithms focus on hard instances in their iterations, which can justify the loss of performance, since the hard instances will correspond to the noisy ones. But interestingly, Boosting was far more affected by a noise level of 40% and not so much in the case of 50% of label noise. One must observe, however, that all algorithms have a large decrease of footprint area for high noise levels, indicating that they perform well on very few instances of the IS. The linear predictors remained quite robust compared to other algorithms for most of the noise levels, despite their simplicity. Bagging, on the other hand, was in general the most affected algorithm for all noise levels as far as the footprint area is concerned.

It would be worth evaluating in the future how the employment of noise cleansing techniques affect the ISA of a noisy dataset, by monitoring the differences of footprint areas for both noisy and clean versions of a same dataset.

5 Conclusion

In this paper we have presented an approach to build and analyze an embedded space of instance hardness for a given classification dataset. We have also launched *PyHard*, a new analytical tool intended to address a gap in meta-learning studies regarding instance hardness for a single dataset. The problem was introduced recalling the ISA framework and linking its formulation to our problem setting. ISA plays a central role here, finding a transformation that reduces the dimensionality of a meta-dataset in a space with linear trends with respect to the difficulty of the individual instances and traces regions of good performance for different classification algorithms.

We have shown projections of sample datasets, including a real COVID prognosis dataset, and insights that can be obtained through their ISA visualization and inspection, such as highlighting observations with potential quality issues and making the strengths of different classification techniques more explicit. The tool also includes functionalities in order to better support the end-user in the analysis of their datasets, including visualizing features distributions for different selections of the dataset. Although a visual inspection does not offer irrefutable proof, it can give valuable insights and guide some descriptive analysis, as demonstrated.

Regarding the computational cost of our analyzes, the step which demands more time is tuning the hyperparameter values of the classification techniques. This hyperparameter tuning step can be easily withdrawn from the analysis, although it is more interesting to evaluate instance hardness considering the best performance achievable by each classification model for a particular dataset. One may also run the predictive evaluation before-hand using another set of desired classifiers and meta-features and use PyISpace to obtain the IS projections from his/her own meta-dataset, while the PyHard visualization application can be used to inspect and interact with the obtained IS afterwards.

As future work, we plan to consolidate the visualization tool and validate its usage in the analysis of datasets of increasing complexity levels and scale. We can also study how the ISA reflects the effectiveness of data pre-processing approaches for improving data quality. This is the case of data cleaning approaches for dealing with label noise, a proper missing data imputation and removing possible sample biases. Dealing with other types of problems, such as learning with imbalanced datasets, is also of interest. We also intend to assist the user in relating classification performance to the meta-features values by obtaining rules aimed to describe situations where each algorithm shows good predictive performance.

More hardness meta-features can be devised and added to the tool too. In fact, most of the meta-features (hardness measures) used in this work rely on class overlapping as a main source of difficulty for classification. But other perspectives can also be regarded, such as the density and structure of the input space. One promising approach is to model the data as a proximity graph from which centrality based features can be extracted.

Another worthwhile strategy in order to validate the hardness embedding for a given dataset is to separate a validation set and build the IS projection only on the remaining data points. By projecting the left-out validation instances in the IS built, we can assess whether hard/easy instances are placed in proper regions of the IS and how the included ML algorithms are expected to behave in their classification.

While we focused on 2-D projections for obtaining visual insights and delineating the footprints of the algorithms more easily, it is also possible to generate hardness embeddings of higher dimensions. The higher the dimension, the less information on the original meta-features values is lost, but the visualization appeal is also hindered. The usefulness of such higher-dimensional embeddings needs to be characterized and understood in future work.

Finally, we have not fully explored all the functionalities included in the MATILDA tool[11] in this work. For instance, there is a module for automatic algorithm selection for different regions of the instance space that was not included in our analyses. This information is interesting to better characterize the domains of competence of the algorithms and will be explored in the future. We can also consider ways to generate new data instances at targeted regions of the instance space, which can be useful for data augmentation, including new instances with increased measures of difficulty to drive algorithmic advances.

[11] Online tool for Instance Space Analysis, available at https://www.matilda.unimelb.edu.au.

Appendix A Proof of proposition

Proposition 1 (cross-entropy bounds). *For any classification problem with C classes there is a lower bound* L_{lower} *and an upper bound* L_{upper} *for the cross-entropy loss* (*aka log-loss*) *such that*: *if* $\text{logloss}(\mathbf{x}_i) < L_{lower}$, *the prediction was correct*; *if* $\text{logloss}(\mathbf{x}_i) > L_{upper}$, *the prediction was incorrect*; *and if* $L_{\text{lower}} \leq \text{logloss}(\mathbf{x}_i) \leq L_{\text{upper}}$, *the prediction can be either correct or incorrect* , *where* $\text{logloss}(\mathbf{x}_i)$ *is the log-loss of instance* $\mathbf{x}_i$. *Specifically*, *these bounds can be set as* $L_{\text{lower}} = -\log\left(\frac{1}{2}\right) = \log 2$ *and* $L_{\text{upper}} = -\log\left(\frac{1}{C}\right) = \log C$.

Proof Given a multiclass setting consisting of C classes, the outcome of a classifier is the predicted probability vector $[p_1, p_2, \ldots, p_C]$, and the predicted class is defined as $\arg\max_j p_j$. And $y_{j,c} := I_{j=c}$ indicates the true class c.

We first prove that if the classifier succeeds in correctly predicting the class of instance $\mathbf{x}_i$, then $\text{logloss}(\mathbf{x}_i) < L_{\text{upper}}$ and that the value of L_{upper} is $-\log(\frac{1}{C})$. Without loss of generality, suppose $j = 1$ is the correct class, since the classes can be always reordered so that each one of them becomes the first in the set. In that case, $\arg\max_j p_j = 1$, which implies that:

$$\begin{aligned} p_1 &> p_2 \\ p_1 &> p_3 \\ &\vdots \\ p_1 &> p_C \end{aligned}$$

Summing all those inequalities results in

$$(C-1)p_1 > p_2 + \cdots + p_C$$

On the other hand, $\sum_j p_j = 1$. Therefore,

$$\begin{aligned} (C-1)p_1 &> 1 - p_1 \\ p_1 &> \frac{1}{C} \\ \log p_1 &> \log \frac{1}{C} \\ -\log p_1 &< -\log \frac{1}{C} \\ -y_{1,c} \log p_1 &< -\log \frac{1}{C} \\ \therefore \text{logloss}(\mathbf{x}_i) &< -\log \frac{1}{C} = L_{\text{upper}} \end{aligned}$$

However, if $\text{logloss}(\mathbf{x}_i) < L_{\text{upper}}$ it does not necessarily imply that the instance $\mathbf{x}_i$ was correctly classified. We show this by counterexample: take the particular predicted probability vector $\left[\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon, 0, \ldots, 0\right]$, and define the right class as $c = 1$. For this vector, the classifier predicts the class 2, since it has the highest probability. The log-loss value is

$$-\sum_{j=1}^{C} y_{j,c} \log p_j = -y_{1,c} \log p_1 = -\log\left(\frac{1}{2} - \varepsilon\right)$$

If we choose $0 < \varepsilon < \frac{1}{2} - \frac{1}{C}$, which is always possible for $C \geq 3$, then

$$\begin{aligned}\text{logloss}(\mathbf{x}_i) &= -\log\left(\frac{1}{2}-\varepsilon\right)\\ &< -\log\left(\frac{1}{2}-\frac{1}{2}+\frac{1}{C}\right)\\ &< -\log\frac{1}{C} = L_{\text{upper}}\end{aligned}$$

Specifically for binary problems with only two classes, if $\text{logloss}(\mathbf{x}_i) < -\log\frac{1}{2}$, then

$$-y_{1,c}\log p_1 < -\log\frac{1}{2} \implies p_1 > \frac{1}{2} > p_2 \implies \arg\max_j p_j = 1$$

So, in the particular case of binary classification problems, $L_{\text{lower}} = L_{\text{upper}}$. Thus,

$$\arg\max_j p_j = 1 \iff \text{logloss}(\mathbf{x}_i) < -\log\frac{1}{2} \quad \text{(binary class problems)}$$

To find L_{lower}, we first show that a multiclass problem can be reduced to a binary classification problem in the sense of log loss metric. Herewith, the vector $[p_1, p_2, \ldots, p_n]$ is equivalent to $[p_1, p_2']$, with $p_2' = \sum_{j=2}^{C} p_j$. In both cases, the log loss value is the same. Thus, we assume $L_{\text{lower}} = -\log\frac{1}{2}$ and prove it by contradiction.

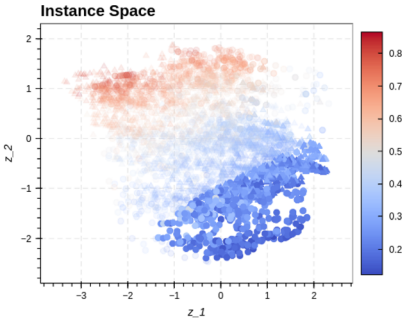
Assume that $\text{logloss}(\mathbf{x}_i) < -\log\frac{1}{2}$, that the correct class is $c = 1$ and $\exists p_k : p_k > p_1$ (classification error). Then,

$$\begin{aligned}&-\log p_1 < -\log\frac{1}{2}\\ &\implies \frac{1}{2} < p_1 < p_k\\ &\implies 1 < p_1 + p_k \quad \text{(absurd!)}\end{aligned}$$

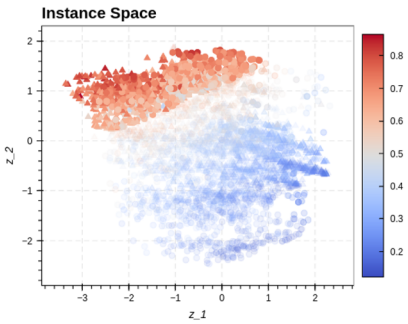
Therefore, $L_{\text{lower}} = -\log\frac{1}{2}$. □

Appendix B Additional figures

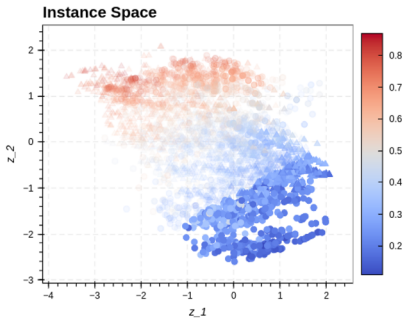
Additional figures from the analysis of the COMPAS dataset are presented here. They show the Lasso selections of easy and hard instances (Fig. 14) and distributions of some other attributes besides `race`, namely `number of priors` (Fig. 15a), `age` (Fig. 15b) and `sex` (Fig. 15c). Median values are represented by vertical dashed lines. A summary of these results is presented and discussed in Sect. 4.2 using Table 2.



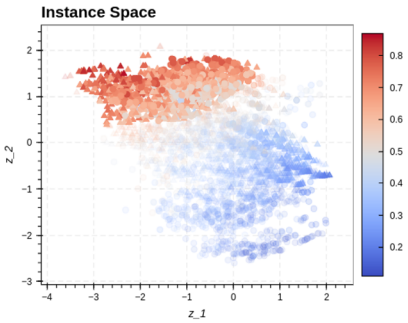
(a) Lasso selection of easy instances from the COMPAS dataset ISA projections with `race` as an attribute.



(b) Lasso selection of hard instances from the COMPAS dataset ISA projections with `race` as an attribute.



(c) Lasso selection of easy instances from the COMPAS dataset ISA projections without `race` as an attribute.



(d) Lasso selection of hard instances from the COMPAS dataset ISA projections without `race` as an attribute.

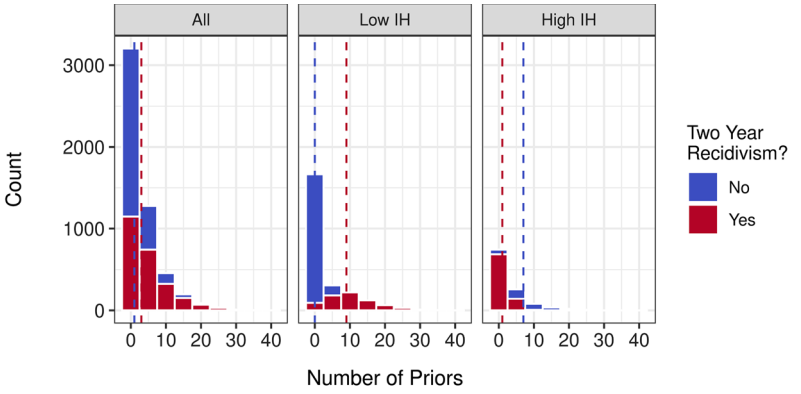
Fig. 14 Lasso selection of hard and easy instances from the COMPAS dataset ISA projections with and without `race` as an attribute

Acknowledgements The authors are thankful to the São José dos Campos municipal health secretariat and IPPLAN (Instituto de Pesquisa e Planejamento) for providing data on COVID cases.

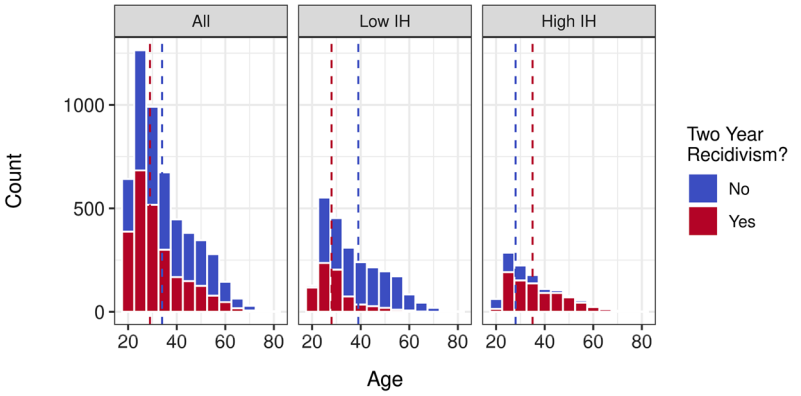
Author contributions P. Y. A. Paiva implemented PyHard and PyIspace, along with all the parts of the framework dedicated to the ISA analysis of a single dataset and has run experiments with the label noise datasets. C. C. Moreno has run and performed the analysis of the COMPAS dataset. K. Smith-Miles has contributed with paper organization, with the validation of the experiments and is the proponent of the original ISA framework. M. G. Valeriano has built, run and performed the analysis of the Covid dataset. A. C. Lorena proposed framing the ISA for the analysis of a single dataset and supervised all the work. All authors contributed with paper writing and organization.

Funding This work was partially supported by the Brazilian research agencies Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 (main grant 88887.507037/2020-00), CNPq (grant 307892/2020-4) and FAPESP (grant 2021/06870-3) and by the Australian Research Council (grant FL140100012).

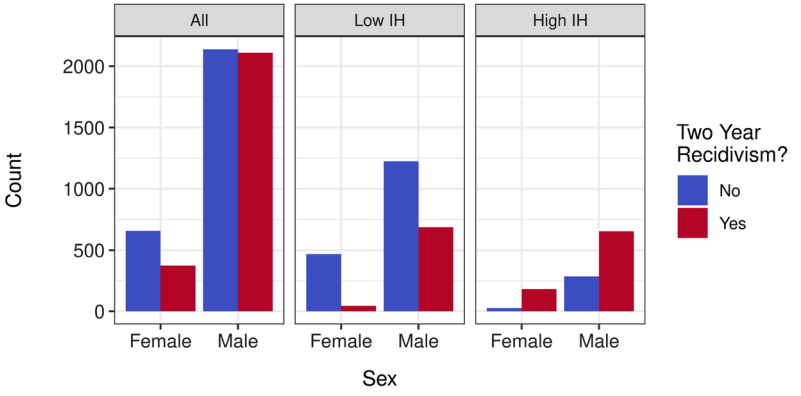
Availability of data and material The benchmark datasets and outputs of their analysis are available at our open repository under the fold 'experiments' (https://gitlab.com/ita-ml/pyhard). The real-data COVID dataset had the features names uncharacterized for anonymization.



(a) Distributions of feature `priors_count`. The blue dashed line indicates the median of the non-recidivist data while the red dashed line indicates the median of the recidivist data.



(b) Distributions of feature `age`. The blue dashed line indicates the median of the non-recidivist data while the red dashed line indicates the median of the recidivist data.



(c) Distributions of feature `sex`.

Fig. 15 Distribution of different features values for the entire COMPAS dataset, low IH and high IH data points, with `race` as an input attribute

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Code availability The source code of the cited Python packages used in the experiments section is available at the public repositories PyHard (https://gitlab.com/ita-ml/pyhard) and PyISpace (https://gitlab.com/ita-ml/pyispace).

Ethical approval The anonymized Covid dataset was obtained as part of a partnership between the project "Data science for fighting outbreaks, epidemics and pandemics in hospitals" coordinated by researcher A. C. Lorena and the São José dos Campos health department.

Consent to participate Not Applicable.

Consent for publication Not Applicable.

References

Arruda, J. L., Prudêncio, R. B., & Lorena, A. C. (2020). Measuring instance hardness using data complexity measures. In *Brazilian Conference on Intelligent Systems*, Springer, pp 483–497.

Barek, M. A., Aziz, M. A., & Islam, M. S. (2020). Impact of age, sex, comorbidities and clinical symptoms on the severity of covid-19 cases: A meta-analysis with 55 studies and 10014 cases. *Heliyon, 6*(12), e05684.

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA, NIPS'11, p 2546–2554.

Bergstra, J., Yamins, D., & Cox, D. D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proc. 30th International Conference on International Conference on Machine Learning - Volume 28*, p I–115–I–123.

Böken, B. (2021). On the appropriateness of platt scaling in classifier calibration. *Information Systems, 95*, 101641.

Corbett-Davies, S., & Goel, S. (2018). The measure and mismeasure of fairness: A critical review of fair machine learning. arXiv preprint arXiv:1808.00023

Edelsbrunner, H. (2010). Alpha shapes–a survey. *Tessellations in the Sciences, 27*, 1–25.

Friedler, S. A., Scheidegger, C., Venkatasubramanian, S., Choudhary, S., Hamilton, E. P., Roth, D. (2019). A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the conference on fairness, accountability, and transparency*, pp 329–338.

Gao, S., Ver Steeg, G., & Galstyan, A. (2015). Efficient estimation of mutual information for strongly dependent variables. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp 277–286.

Garcia, L. P., de Carvalho, A. C., & Lorena, A. C. (2015). Effect of label noise in the complexity of classification problems. *Neurocomputing, 160*, 108–119.

Giraud-Carrier, C., & Provost, F. (2005). Toward a justification of meta-learning: Is the no free lunch theorem a show-stopper. In *Proc. ICML-2005 Workshop on Meta-learning*, pp 12–19.

Hajian, S., Bonchi, F., & Castillo, C. (2016). Algorithmic bias: From discrimination discovery to fairness-aware data mining. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp 2125–2126.

Hillinger, C. (2004). Voting and the cardinal aggregation of judgments. SSRN 548662

Kandanaarachchi, S., Muñoz, M. A., Hyndman, R. J., & Smith-Miles, K. (2020). On normalization and algorithm selection for unsupervised outlier detection. *Data Mining and Knowledge Discovery, 34*(2), 309–354.

Kang, Y., Hyndman, R. J., & Smith-Miles, K. (2017). Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting, 33*(2), 345–358.

Khademi, A., & Honavar, V. (2020). Algorithmic bias in recidivism prediction: A causal perspective (student abstract). *Proceedings of the AAAI Conference on Artificial Intelligence, 34*(10), 13839–13840.

Khan, K., Rehman, S. U., Aziz, K., Fong, S., & Sarasvady, S. (2014). Dbscan: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, IEEE, pp 232–238.

Kletzander, L., Musliu, N., & Smith-Miles, K. (2021). Instance space analysis for a personnel scheduling problem. *Annals of Mathematics and Artificial Intelligence, 89*, 617–637.

Leyva, E., González, A., & Pérez, R. (2014). A set of complexity measures designed for applying meta-learning to instance selection. *IEEE Transactions on Knowledge and Data Engineering, 27*(2), 354–367.

Leyva, E., González, A., & Pérez, R. (2015). Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective. *Pattern Recognition, 48*(4), 1523–1537.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM Computing Surveys, 50*(6), 1–45.

Maletic, J. I., & Marcus, A. (2000). Data cleansing: Beyond integrity analysis. In *Iq*, pp 200–209.

Muñoz, M. A., Villanova, L., Baatar, D., & Smith-Miles, K. (2018). Instance spaces for machine learning classification. *Machine Learning, 107*(1), 109–147.

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.

Muñoz, M. A., Yan, T., Leal, M. R., Smith-Miles, K., Lorena, A. C., Pappa, G. L., & Rodrigues, R. M. (2021). An instance space analysis of regression problems. *ACM Transactions on Knowledge Discovery from Data (TKDD), 15*(2), 1–25.

Platt, J., et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers, 10*(3), 61–74.

Prati, R. C. (2012). Combining feature ranking algorithms through rank aggregation. In: *The 2012 International joint conference on neural networks (IJCNN)*, IEEE, pp 1–8.

Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers* (Vol. 15, pp. 65–118). Elsevier.

Rudin, C., Wang, C., & Coker, B. (2020). The age of secrecy and unfairness in recidivism prediction. *Harvard Data Science Review*. https://doi.org/10.1162/99608f92.6ed64b30.

Sani, H. M., Lei, C., & Neagu, D. (2018). Computational complexity analysis of decision tree algorithms. In: *International conference on innovative techniques and applications of artificial intelligence*, Springer, pp 191–197.

Smith, M. R., Martinez, T., & Giraud-Carrier, C. (2014). An instance level analysis of data complexity. *Machine Learning, 95*(2), 225–256.

Smith-Miles, K., & Bowly, S. (2015). Generating new test instances by evolving in instance space. *Computers & Operations Research, 63*, 102–113.

Smith-Miles, K., & Lopes, L. (2011). Generalising algorithm performance in instance space: A timetabling case study. In: *International conference on learning and intelligent optimization*, Springer, pp 524–538.

Smith-Miles, K., & Tan, T. T. (2012). Measuring algorithm footprints in instance space. In: *2012 IEEE congress on evolutionary computation*, IEEE, pp 1–8.

Smith-Miles, K., Baatar, D., Wreford, B., & Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers and Operations Research, 45*, 12–24.

Smith-Miles, K., Christiansen, J., & Muñoz, M. A. (2021). Revisiting where are the hard knapsack problems? via instance space analysis. *Computers & Operations Research, 128*, 105184.

Smith-Miles, K. A. (2009). Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection. *ACM Computing Surveys, 41*(1), 1–25.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In: *Proceedings. 25th international conference on neural information processing systems - Volume 2*, Curran Associates Inc., Red Hook, NY, USA, p 2951–2959.

Vanschoren, J. (2019). Meta-learning. In *Automated Machine Learning*, Springer, pp 35–61.

Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review, 18*(2), 77–95.

Wolpert, D. H. (2002). *The Supervised Learning No-Free-Lunch Theorems* (pp. 25–42). Springer London.

Yarrow, S., Razak, K. A., Seitz, A. R., & Seriès, P. (2014). Detecting and quantifying topography in neural maps. *PloS one, 9*(2), e87178.

Zhu, X., & Wu, X. (2004). Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review, 22*(3), 177–210.

Authors and Affiliations

Pedro Yuri Arbs Paiva[1] · Camila Castro Moreno[1,2] · Kate Smith-Miles[3] · Maria Gabriela Valeriano[1,2] · Ana Carolina Lorena[1]

Camila Castro Moreno
camila.moreno@ga.ita.br

Kate Smith-Miles
smith-miles@unimelb.edu.au

Maria Gabriela Valeriano
maria.valeriano@ga.ita.br

Ana Carolina Lorena
aclorena@ita.br

[1] Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos, São Paulo, Brazil

[2] Universidade Federal de São Paulo (Unifesp), São José dos Campos, São Paulo, Brazil

[3] School of Mathematics and Statistics, University of Melbourne, Melbourne, Victoria, Australia