Trusting my predictions: on the value of Instance-Level analysis

ANA C. LORENA, Instituto Tecnológico de Aeronáutica, Brazil
PEDRO Y. A. PAIVA, Instituto Tecnológico de Aeronáutica, Brazil
RICARDO B. C. PRUDÊNCIO, Centro de Informática, Universidade Federal de Pernambuco, Brazil

Machine Learning solutions have spread along many domains, including critical applications. The development of such models usually relies on a dataset containing labeled data. This dataset is then split into training and test sets and the accuracy of the models in replicating the test labels is assessed. This process is often iterated in a cross-validation procedure for obtaining average performance estimates. But is the average of the predictive performance on test sets enough for assessing the trustfulness of a Machine Learning model? This paper discusses the importance of knowing which individual observations of a dataset are more challenging than others and how this characteristic can be measured and used in order to improve classification performance and trustfulness. A set of strategies for measuring the hardness level of the instances of a dataset is surveyed and a Python package containing their implementation is provided.

CCS Concepts: • **Computing methodologies** → **Machine learning**.

Additional Key Words and Phrases: instance hardness, metalearning

ACM Reference Format:
Ana C. Lorena, Pedro Y. A. Paiva, and Ricardo B. C. Prudêncio. 2022. Trusting my predictions: on the value of Instance-Level analysis. *ACM Comput. Surv.* 00, 00, Article 00 (2022), 28 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Some recent pieces of work in the Machine Learning (ML) literature have identified the value of better understanding the datasets available for learning [25, 27, 39]. This is a natural follow up as ML techniques are data-driven and have their performance affected by the quality of learning data. Meanwhile, the Metalearning (MtL) literature has been studying how to characterize datasets and relate these characteristics to the predictive performance achieved by ML models since the late 1990s [40]. This allows to support ML algorithm selection for new problems, which can be tailored for their difficulty level and potentially less prone to overfitting.

The common practice to assess difficulty in ML is to adopt dataset-level methods that either: (1) directly evaluate ML algorithms on a dataset using standard evaluation procedures (e.g., cross-validation) and performance measures (e.g., accuracy); or (2) compute dataset complexity measures [21], such as feature importance measures, class overlap, noise level, among others. Such methods provide useful information to choose the most promising algorithms and to identify the limits of learning in a particular dataset. However, the dataset-level approach fails to identify that some

Authors' addresses: Ana C. Lorena, aclorena@ita.br, Instituto Tecnológico de Aeronáutica, Praça Marechal Eduardo Gomes, 50, São José dos Campos, São Paulo, Brazil, 12228-900; Pedro Y. A. Paiva, paiva@ita.br, Instituto Tecnológico de Aeronáutica, Praça Marechal Eduardo Gomes, 50, São José dos Campos, São Paulo, Brazil, 12228-900; Ricardo B. C. Prudêncio, rbcp@cin.ufpe.br, Centro de Informática, Universidade Federal de Pernambuco, Av. Jornalista Aníbal Fernandes, s/n – Cidade Universitária, Recife, Pernambuco, Brazil, 50740-560.

instances might be harder (or easier) to classify than others. In fact, particular performance patterns may be hidden when a model is evaluated using global aggregation measures like accuracy. Two learned models may have similar accuracies, but present distinct abilities when predicting the classes of different instances in a dataset. Also, a given model may have an acceptable global performance but fail miserably for certain instances. Dataset complexity measures adopted in literature also have the same limitation since they have a global perspective for evaluating difficulty, without considering that in most cases difficulty is not uniformly distributed in a classification problem. Some instances are harder to predict than others because they are close to a class boundary, lie in sparse regions or in regions with a lot of label noise, are outliers, among many different factors. Such characteristics in a dataset can result in less reliable predictions for certain instances, which consequently may harm the user acceptance of a ML solution.

The above limitations motivate a more fine-grained analysis of problem difficulty, which can be measured at the instance-level. Different from the dataset-level analysis, the instance-level approach aims to answer more specific questions, like: Which instances in a dataset are more difficult? How difficulty is distributed in a classification problem? Which features make an instance more difficult to predict? Answers to such questions are useful to understand when to trust the predictions of a ML model, what is the uncertainty level related to such predictions and eventually to suggest the use of corrective procedures like data pre-processing and augmentation.

Measure difficulty per instance can also address other dimensions that are usually required for making the ML trustworthy, particularly: (1) fairness, i.e., to ensure that a ML system is free from any bias, discriminationon or favoritism toward an individual or a group; (2) explainability, i.e., to ensure that the behaviour of a ML system can be understood by developers, stakeholders and final users; and (3) reliability, i.e., to ensure that the predictions of the ML system can be trusted and at what level. Regarding fairness, a model may be biased to favor or discriminate certain instances possibly due to a bias that is already present in the data. The instance-level approach for measuring difficulty can be useful to identify in advance the risks of an unfair predictive behavior. Regarding explainability in turn, the instance-level analysis would be useful not only to understand why a model returned a particular prediction for a given instance (as done in well-known techniques like LIME), but whether the returned prediction is reliable. This second explainability layer is not well investigated in literature. i.e., why a model failed for a particular instance? Is it due to any model limitation or is the instance intrinsically difficult? Such questions are very relevant for ML developers to detect vulnerabilities in the ML system and to decide how to correct errors. This type of analysis can also leverage the reliability the ML systems, whose outputs are inspected at a fine-grained level than the usual approach of relying on average statistics across all instances of a dataset.

This paper surveys the main recent developments on characterizing and understanding the difficulty level of the individual instances of a dataset, aka instance hardness level. This concept was introduced in the work of Smith et al. [39] and has since then been successfully used to improve the predictive performance of classifiers and to identify potential data quality issues and biases [26, 27, 41]. The formulations, strengths and limitations of each of the approaches for instance hardness measurement are presented and discussed. Finally, a Python package named PyHard[1] is provided so that interested users can extract different instance hardness measures from their datasets.

This paper is structured as follows: Section 2 proposes a taxonomy for the analysis of the hardness level of a dataset. Sections 3 and 4 review the main strategies for instance hardness measurement. Section 5 reviews the usage of instance hardness measures in the literature. Section 6 concludes the paper and discusses some future work venues foreseen.

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

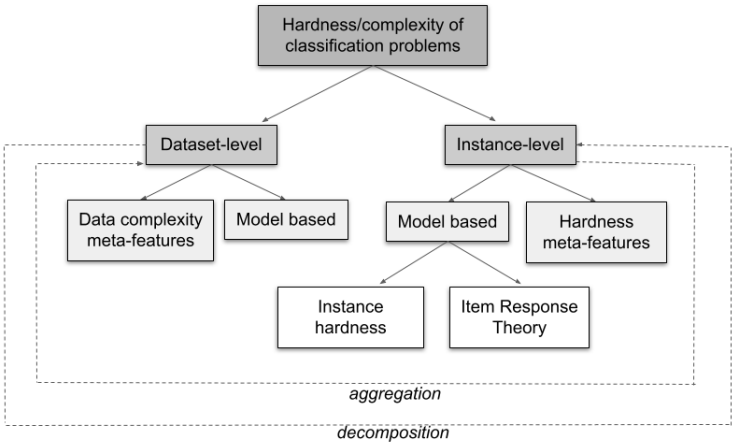


Fig. 1. A taxonomy for the ways of estimating the hardness level of a classification problem in ML.

2 A TAXONOMY FOR HARDNESS ANALYSIS

The difficulty aka hardness level of a classification problem can be estimated by the analysis of the dataset available for learning [13]. If a dataset requires a complex decision boundary for the proper separation of the classes, this is an indicative that the underlying classification problem may be complex to solve. This is true given the dataset available for learning. In this way, a simple classification problem can become very complex if its associated dataset lacks proper discriminating input features, for instance. Similarly, a naturally complex problem can look deceptively simple if the available observations are sparsely distributed in the input space. In this paper we will refer to the terms hardness level and complexity level as synonyms, as they are both used in the ML literature to refer to the difficulty level in predictive problems [13, 21, 39].

Given a dataset $\mathcal{D}$ containing n labeled observations or instances in the form $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathcal{X} \wedge y_i \in \mathcal{Y}\}$, there are multiple forms of estimating how difficult it will be to learn from such data. Theoretically, one may regard at the Kolmogorov complexity, where the complexity of a classification problem can be measured by the size of the smallest algorithm that is able to describe the relationships between $\mathcal{X}$ and $\mathcal{Y}$ using $\mathcal{D}$ [13, 20]. But, in practice, the Kolmogorov complexity is incomputable and approximations are made.

The most standard procedure for assessing the hardness level in solving a classification problem as represented by $\mathcal{D}$ is to induce different ML models using the dataset as input and measuring the predictive performance attained. This involves splitting the dataset into training and testing folds, usually in a cross-validation procedure. The ML models learn from training data and have their predictive performance assessed in the corresponding test partitions. If the predictive accuracy attained is low despite of the ML technique employed and how tuned it is, the problem may be regarded as difficult to solve (complex or hard) given $\mathcal{D}$. Strategies relying on the predictive performance of ML models for measuring classification complexity will be referred as model based.

The Metalearning (MtL) literature has also plenty of measures that can be extracted from $\mathcal{D}$ for estimating classification complexity [13, 21]. The idea in this case is to extract indicators of the geometry of the classes and their structure. For instance, if the classes overlap too much given the available input features or there are many outliers, the learning

problem will be difficult to solve using $\mathcal{D}$. As there are multiple possible sources of complexity in a classification problem, there are also multiple perspectives and measures for estimating classification complexity. These measures will be called meta-features in this paper. Some meta-features rely on running simple classifiers using $\mathcal{D}$, which allows to obtain hardness level estimates at low computational cost, approaching the landmarking and model-based categories of meta-features in MtL [45]. Others extract descriptors of the structure and organization of the classes [21].

But the focus of this section is to distinguish two levels where the complexity/hardness analysis can take place, as shown in Figure 1. The first is the dataset-level complexity analysis. Here global estimates of the hardness level of the classification problem are obtained. In model based hardness estimates, this means taking an aggregated measure of the accuracy performance (or another metric) observed for all observations in $\mathcal{D}$, usually by averaging. And when regarding at the data complexity meta-features, characteristics such as the estimated amount of overlap of the classes in the dataset or the expected shape of the decision boundary needed to separate the classes are taken [21]. Recent work has specifically focused on the ability of the complexity measures to detect class overlapping regions in datasets [8, 37] and in evaluating classification complexity when the distribution of the classes is uneven (imbalanced domains) [2, 16].

Although valuable on its own, the dataset-level analysis of classification complexity hinders important information. Even a problem considered globally simple may contain challenging instances for which any classifier will struggle to classify. That is, the classification complexity is usually non-uniform along the instances of a dataset. This has lead to the investigation of instance-level hardness [39]. Here the idea is to identify which particular instances of a dataset are hard to classify and possible reasons why they are so. This allows to perform a more fine-grained analysis of a dataset and of the reliability of the predictive performance achieved for it.

The instance-level analysis of classification complexity can also be estimated by relying on the performance of one or multiple classifiers in predicting the class of each instance or by extracting some hardness meta-features. Smith et al. [39] define instance hardness as the average of misclassification probability of a pool of representative learning models of different biases. The Item Response Theory (IRT) has also been recently framed for measuring instance hardness based on the predictive performance of multiple learning models for each instance [22], where the ML models are regarded as the respondents and the instances as items of a test from which latent characteristics such as the level of difficulty of the items can be extracted. Meanwhile, the hardness meta-features at an instance-level were introduced by Smith et al. [39] as ways of understanding why an instance is hard to classify. As an example, an instance can be surrounded by many instances from a different class, making it more difficult to classify than if it was placed in a region containing many elements sharing its class label.

Since a dataset $\mathcal{D}$ is a collection of individual instances $(\mathbf{x}_i, y_i)$, it is also often possible to decompose the model based and meta-features values at the dataset-level for obtaining hardness estimates at the instance-level. This is done in [1] for some of the data complexity measures described in [21]. On the other hand, by aggregating the values of the instance-level hardness estimates, one may obtain dataset-level estimates of classification complexity. Smith et al. [39] average their hardness meta-features for obtaining aggregate estimates at the dataset-level and show they are also good descriptors of global classification complexity. One may also aggregate the instance-level meta-features' values at the class-level, obtaining the level of difficulty of each of the classes in a dataset, as done in [17].

Next sections will present the right hand side of Figure 1 in a more detailed way.

3 MODEL BASED INSTANCE HARDNESS

In [39] the concept of *instance hardness* (IH), roughly speaking, is defined as the likelihood of an instance being misclassified despite the ML technique considered. In order to illustrate the IH concept, consider the bidimensional

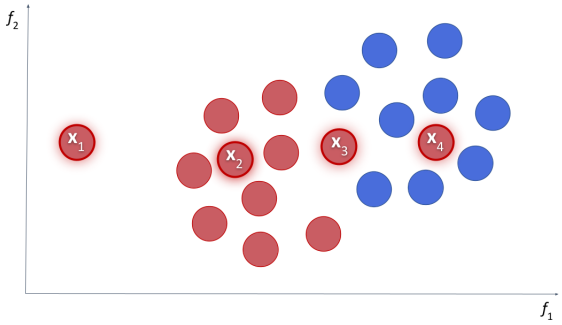


Fig. 2. Sample dataset for showing the computation of the HM: $\mathbf{x}_1$ is an outlier or anomalous instance of the red class; $\mathbf{x}_2$ is an easy instance; $\mathbf{x}_3$ is a borderline instance; and $\mathbf{x}_4$ is a noisy instance.

dataset in Figure 2. Four instances of interest are highlighted in this dataset: (i) $\mathbf{x}_1$ is an outlier from the red class, that is, a correct instance with anomalous or distinct patterns from those of the same class it belongs; (ii) $\mathbf{x}_2$ is an easy instance in a dense region surrounded by elements sharing its label; (iii) $\mathbf{x}_3$ is a borderline instance, that is, an instance near the two classes, which can also be regarded as situated in an overlapping region of the classes; and (iv) $\mathbf{x}_4$ is a noisy instance, being surrounded by many elements from the other class. Instance $\mathbf{x}_4$ is more prone to classification errors and is more difficult to classify. Instance $\mathbf{x}_3$ might also be a little difficult and some classifiers may struggle to correctly predict its label. Both $\mathbf{x}_3$ and $\mathbf{x}_4$ are in overlapping regions of the classes, although $\mathbf{x}_4$ might be wrongly labeled. In contrast, both $\mathbf{x}_1$ and $\mathbf{x}_2$ are easy to classify. $\mathbf{x}_2$ is in a dense region of instances sharing its class. Despite being distant from the other elements of its class, $\mathbf{x}_1$ will also be probably correctly classified, as it is far from the decision boundary needed to separate the classes.

Next we formalize two strategies for computing IH that consider the performance of ML models in the prediction of the labels of an instance $\mathbf{x}_i$.

3.1 Instance Hardness

Formally, the hardness of the instance $\mathbf{x}_i$ with respect to a classification hypothesis h is defined by Smith et al. [39] as:

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

where $h : \mathcal{X} \rightarrow \mathcal{Y}$ is a hypothesis or function mapping input features in $\mathcal{X}$ to output labels in $\mathcal{Y}$. In practice, h is induced by a learning algorithm l trained on a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in X \wedge y_i \in \mathcal{Y}\}$ with hyper-parameters β, that is, $h = l(D, \beta)$. To be less biased by a specific classifier, the authors in [39] derive the instance hardness for a set of representative learning algorithms $\mathcal{L}$ as:

$$IH_{\mathcal{L}}(\mathbf{x}_i, y_i) = 1 - \frac{1}{|\mathcal{L}|} \sum_{j=1}^{|\mathcal{L}|} p\big(y_i|\mathbf{x}_i, l_j(\mathcal{D}, \beta)\big). \tag{2}$$

This equation averages the probabilities of misclassification of the learning algorithms $\mathcal{L}$ when using $\mathcal{D}$ as input. It outputs values in the range $[0, 1]$, and higher values are observed for instances that are hard to classify. 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, despite their biases.

Another formulation of IH averages a 0-1 loss function, that is, if a learning algorithm l hits the prediction for a given instance the value 1 is summed up, otherwise the value 0 is summed up. But the formulation in Equation 2 is smoother and preferred. In [31], a similar model-based instance hardness approach is proposed, but also including a squared error and a ranking function to measure the model loss for each instance.

Learning algorithms $\mathcal{L}$ were chosen in [39] in such a way to guarantee the diversity of predictions and avoid model bias while measuring hardness. Initially a pool of 20 well-known algorithms were evaluated on different datasets and the Classifier Output Difference (COD) was computed to measure the diversity between algorithms. Then an unsupervised method clustered the learning algorithms based on the COD values. A representative algorithm from each cluster was finally chosen to compose the pool $\mathcal{L}$.

3.2 Item Response Theory

Another model-based approach to measure instance hardness is based on the use of Item Response Theory (IRT), adapted to evaluation in AI. IRT is a framework from Psychometrics, originally aiming to estimate the latent abilities of human respondents based on their responses to items with different levels of difficulty. IRT has been widely adopted to evaluate cognitive skill of humans and more recently for AI evaluation, where AI techniques are treated as respondents and AI tasks are modelled as items [22]. Abilities are estimated in such a way that highly skilled respondents are those ones who provide better responses to the most difficult items. Each response provided by a respondent to an item is modelled as a random variable whose distribution depends on the respondent's ability and the item's parameters.

The most common IRT models are dichotomous, in which responses are either correct or incorrect. In the 3-parameter logistic (3PL) IRT model, the probability of a correct response is defined by a logistic function of the respondent's ability and the item's difficulty, discrimination and guessing parameters. For each item i, this model results in an Item Characteristic Curve (ICC), which defines the expectation of a correct response for that item along the ability range according to Equation (3):

$$E[r_{ij} = 1|\theta_j, \delta_i, a_i, c_i] = c_i + \frac{1 - c_i}{1 + e^{-a_i(\theta_j - \delta_i)}} \tag{3}$$

in which:

- r_{ij} is the response of respondent j to item i, which is 1 of the response is correct and 0 otherwise;
- δ_i is the difficulty of item i (the location parameter of the ICC);
- a_i is the discrimination of item i (the slope of the ICC);
- c_i is the guessing parameter (the asymptotic minimum of the ICC).

Figure 3 shows an example of ICC, with difficulty $\delta_i = 3$, discrimination $a_i = 2$ and guessing $c_i = 0.1$. For estimation, responses are collected from a pool of respondents and items. Then, both respondents and items parameters are estimated to fit the observed responses, for instance via maximum likelihood estimation or other inference method. In the ML context, IRT has been adopted to evaluate instance difficulty by treating classifiers as respondents and instances as items. Regarding the difficulty parameter δ_i, easy items are solved by almost all classifiers, while difficult items are only solved by very good classifiers. In turn, the discrimination parameter a_i measures the capability of an item to differentiate between strong or weak classifiers. The guessing parameter c_i indicates how likely a very bad classifier obtains the correct answer by (random) guessing.

Similar to [39], IRT also considers a pool of learning algorithms to define instance hardness. However, instead of just averaging responses across models to estimate each instance's difficulty (like Equation 2), IRT considers the ability

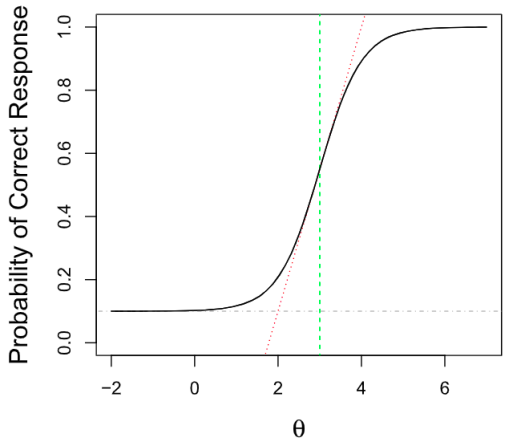


Fig. 3. Example of ICC [22]. Green line indicates the item difficulty. When ability is equal difficuly, the probability of correct response lies in the middle point between the guessing value and 1. Redline indicates the discrimination (slope) at the difficulty level. It measures how fast good responses are observed as ability is higher than the level of difficulty of an item. Guessing parameter is the asymptotic response probability for very low levels of ability.

of classifiers that correctly respond the instance. For example, an instance may be homogeneously difficult for all classifiers or may be difficult especially for some competent classifiers. In IRT, an instance is considered hard if it tends to be correctly responded only by the best classifiers. A correct and unexpected response by a bad classifier to a difficult instance can be just detected by IRT as a guess, and thus will not receive too much importance when defining instance hardness and classifier ability.

In [22], the authors applied IRT to classification, by adopting a pool of 123 classifiers for difficulty estimation. In the experiments, difficulty can be caused by several reasons depending on the classification dataset, like borderlines, instances located close to examples of a different class, outliers, among other. Also, negative discrimination is a strong indication that the instance has label noise. Other previous work extended the use of IRT to other ML contexts. In [9], the authors proposed a new continuous IRT model to deal with responses derived from the estimated correct class probabilities. In this case, responses are equivalent to Equation 1. In [24], the authors adopted IRT to model regression errors.

4 HARDNESS META-FEATURES

Apart from the definition of IH, Smith et al. [39] also propose a set of *hardness measures* (HM) for classification problems, which aim to explain *why* some instances are harder to be classified correctly than others in a dataset. These measures are presented next. All of them are computed for the instances of a dataset $\mathcal{D}$ with n data instances $\mathbf{x}_i \in \mathcal{X}$ assuming labels y_i in a set $\mathcal{Y}$. In addition, let each instance be described by m input attributes and the cardinality of $\mathcal{Y}$ be C, that is, the problem has C classes. The described measures were proposed in [1, 39]. Measures from Arruda et al. [1] are based on dataset-level complexity measures of classification datasets [21], decomposed at the instance-level. All measures are standardized in this paper so that larger values are observed for more difficult instances. There was also a general attempt to bound the values achieved by the measures in an interval, which is between $[0, 1]$. This is important so that one can compare more directly the measures values computed for different instances.

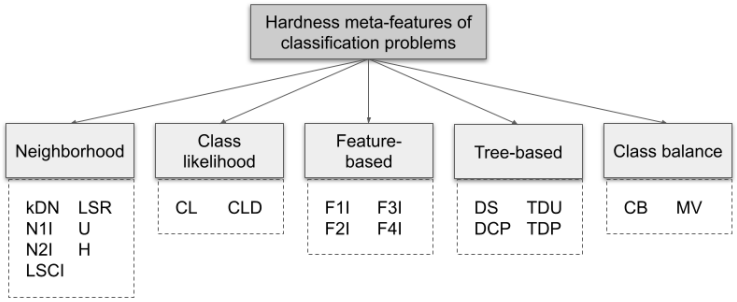


Fig. 4. An overview of the main hardness meta-features of classification problems.

The presented measures are divided here into the following categories:

- *Neighborhood-based*: these measures regard on the nearest neighbors of an instance $\mathbf{x}_i$ to compute its hardness level.
- *Class Likelihood*: these measures consider the likelihood an instance has of belonging to its class y_i.
- *Feature-based*: these measures try to verify if the instance has features values in overlapping areas of the classes.
- *Tree-based*: these measures build a decision tree model and extract some features from it, like depth in which an instance is classified.
- *Class balance*: these measures take the proportion of examples sharing the same class as $\mathbf{x}_i$.

A summary of the HM described in this section is presented in Figure 4. In order to illustrate the HM, their values are computed for the four representative instances shown in the bidimensional dataset from Figure 2.

4.1 Neighborhood-based HM

These measures regard on the neighborhood of each instance $\mathbf{x}_i$ in the dataset $\mathcal{D}$ in order to estimate its hardness level. The idea is that if an instance is surrounded by elements sharing its class label in the input space $\mathcal{X}$ it can be considered easier to classify. In contrast, an instance which is locally close to elements of another class can be considered hard to classify.

k-Disagreeing Neighbors $kDN(\mathbf{x}_i)$ [39]: outputs the percentage of the k nearest neighbors of $\mathbf{x}_i$ in $\mathcal{D}$ which do not share its label:

$$kDN(\mathbf{x}_i) = \frac{\sharp\{\mathbf{x}_j | \mathbf{x}_j \in kNN(\mathbf{x}_i) \wedge y_j \neq y_i\}}{k}, \tag{4}$$

where $kNN(\mathbf{x}_i)$ represents the set of k-nearest neighbors of the instance $\mathbf{x}_i$ in the dataset $\mathcal{D}$. k is frequently set to 5 [27, 39], although other k values could be used. Using $k = 5$ limits the values of kDN in the set $[0, 0.2, 0.4, 0.6, 0.8, 1]$. A larger k value would give smoother values, but at the cost of taking into account

the information of instances that are too dissimilar from $\mathbf{x}_i$. kDN can be computed at $O(n \cdot m)$ asymptotic computational cost for a dataset $\mathcal{D}$ with n instances and m input features.

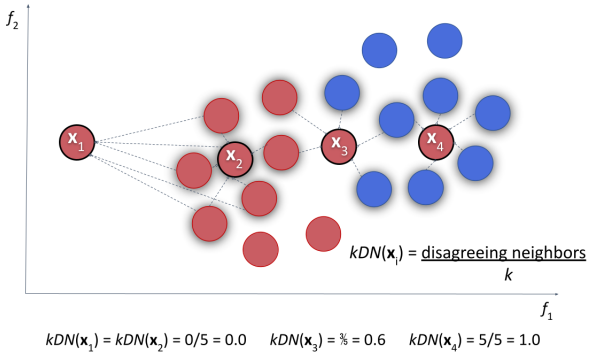


Fig. 5. Examples of kDN computation for different instances in the dataset from Figure 2. The five nearest neighbors in each case are shadowed and connected by dashed lines to the corresponding instances.

The higher the value of $kDN(\mathbf{x}_i)$, the harder $\mathbf{x}_i$'s classification shall be, since it is surrounded by examples from a different class. This will be the case of noisy instances, which will show kDN values close to 1, or borderline instances, which will tend to present intermediary kDN values. Figure 5 presents examples of the computation of kDN value for the different types of instances highlighted in Figure 2. The kDN values are minimum for $\mathbf{x}_1$ and $\mathbf{x}_2$, intermediary for $\mathbf{x}_3$ and maximum for $\mathbf{x}_4$.

Fraction of nearby instances of different classes at an instance-level $N1I(\mathbf{x}_i)$ [1]: first a minimum spanning tree MST is built from $\mathcal{D}$. In this MST, each instance of the dataset $\mathcal{D}$ corresponds to one vertex and nearby instances are connected according to their distances in order to obtain a tree of minimal cost concerning the sum of the edges' weights. Close elements in the input space will tend to be connected in this structure. $N1I$ gives the percentage of instances of different classes $\mathbf{x}_i$ is connected to in the MST:

$$N1I(\mathbf{x}_i) = \frac{\sharp\{\mathbf{x}_j | (\mathbf{x}_i, \mathbf{x}_j) \in MST(\mathcal{D}) \wedge y_i \neq y_j\}}{\sharp\{\mathbf{x}_j | (\mathbf{x}_i, \mathbf{x}_j) \in MST(\mathcal{D})\}}. \tag{5}$$

$N1I$ computation requires first assembling a distance matrix between all pairs of elements in $\mathcal{D}$, which requires $O(m \cdot n^2)$ operations and dominates the computational cost of this measure over the MST construction.

Larger values of $N1I(\mathbf{x}_i)$ indicate that $\mathbf{x}_i$ is close to examples of different classes, either because it is borderline or noisy, making it hard to classify. Examples of $N1I$ computation for the instances highlighted in Figure 2 are shown in Figure 6. Part of the pairwise proximity graph is shown and the MST edges are shown as ticker edges. The values of $N1I$ are null for instances $\mathbf{x}_1$ (anomalous) and $\mathbf{x}_2$ (easy). The value is maximum for the noisy instance $\mathbf{x}_4$. And it is intermediary for the borderline instance $\mathbf{x}_3$.

Ratio of the intra-class and extra-class distances at an instance-level $N2I(\mathbf{x}_i)$: this measure considers 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):

$$\text{IntraInter}(\mathbf{x}_i) = \frac{d(\mathbf{x}_i, NN(\mathbf{x}_i) \in y_i)}{d(\mathbf{x}_i, NE(\mathbf{x}_i))}, \tag{6}$$

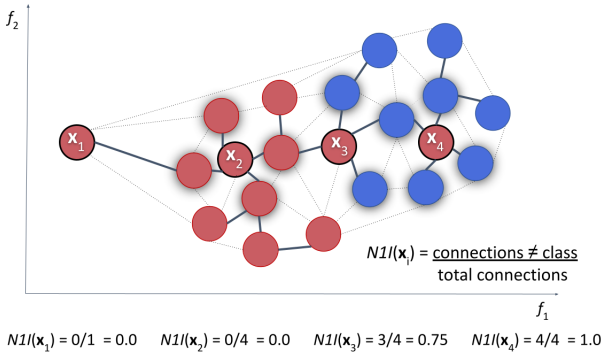


Fig. 6. Example of *N1I* computation for some instances of the dataset from Figure 2. The MST edges are highlighted as thicker edges. Connected neighbors to the labeled instances in the MST are shadowed. Not all edges between all pairs of nodes from the proximity graph are presented in the figure, for better readability.

where d is a distance measure, $NN(\mathbf{x}_i)$ represents a nearest neighbor of $\mathbf{x}_i$ and $NE(\mathbf{x}_i)$ is the nearest enemy of $\mathbf{x}_i$, that is:

$$NE(\mathbf{x}_i) = \{NN(\mathbf{x}_i) \in y_j \neq y_i\}. \tag{7}$$

N2I is then taken as:

$$N2I(\mathbf{x}_i) = 1 - \frac{1}{\text{IntraInter}(\mathbf{x}_i) + 1} \tag{8}$$

As in *N1I*, the larger computational cost involved in obtaining *N2I* is to compute a distance matrix between all pairs of elements in $\mathcal{D}$, requiring $O(m \cdot n^2)$ operations.

Larger values of $N2I(\mathbf{x}_i)$ 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. *N2I* resembles the Silhouette coefficient from the clustering literature [34], giving indicatives on whether an instance is well situated in its class or would be better placed in other class. Figure 7 presents the computation of the *N2I* value for the highlighted instances in the dataset from Figure 2. The nearest neighbor from the same class for each instance of interest is the connected red shadowed instance and the nearest enemy is the connected blue shadowed instance. Noisy instances will tend to show large *N2I* values. Correct instances of a class with atypical patterns can also have differentiated *N2I* values, as the intra-class distance will be high for them. This can be noted in our example, where the easiest instance according to *N2I* is $\mathbf{x}_2$ and the hardest is $\mathbf{x}_4$, whilst $\mathbf{x}_1$ and $\mathbf{x}_3$ present intermediary *N2I* values.

Local Set Cardinality at an instance-level $LSCI(\mathbf{x}_i)$ [1]: the Local-Set (LS) of an instance $\mathbf{x}_i$ is the set of points from $\mathcal{D}$ whose distances to $\mathbf{x}_i$ are smaller than the distance between $\mathbf{x}_i$ and $\mathbf{x}_i$'s nearest enemy [18], as defined in:

$$LS(\mathbf{x}_i) = \{\mathbf{x}_j | d(\mathbf{x}_i, \mathbf{x}_j) < d(\mathbf{x}_i, NE(\mathbf{x}_i))\}, \tag{9}$$

where $NE(\mathbf{x}_i)$ is defined by Equation 7. *LSCI* outputs the complement of the relative cardinality of such set:

$$LSCI(\mathbf{x}_i) = 1 - \frac{\sharp\{\mathbf{x}_j | d(\mathbf{x}_i, \mathbf{x}_j) < d(\mathbf{x}_i, NE(\mathbf{x}_i))\}}{\sharp\{\mathbf{x}_j | y_i = y_j\}}. \tag{10}$$

Larger local sets are obtained for easier examples, which are in dense regions surrounded by instances from their own classes. As a consequence, the complement of their local set cardinality will be low. The asymptotic cost of

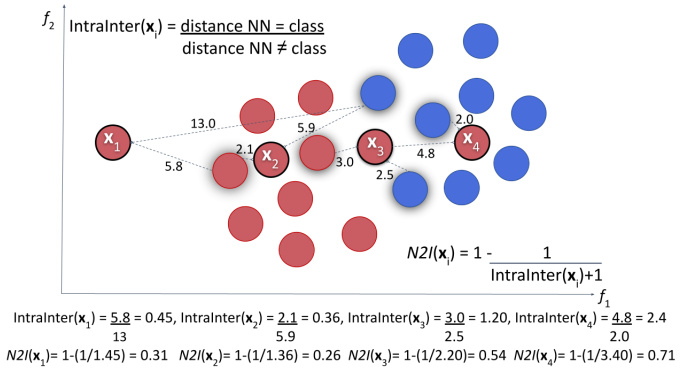


Fig. 7. Example of *N2I* computation for some instances from the dataset in Figure 2. The nearest neighbor from a same class and nearest enemy instances are shadowed and linked between each other.

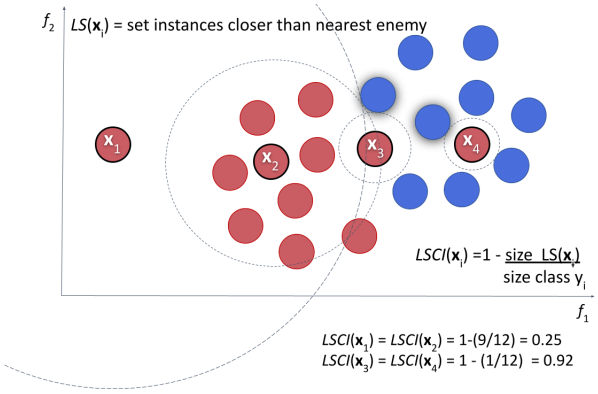


Fig. 8. Example of *LSCI* computation for some instances of the dataset from Figure 2. The nearest nearest enemy of each instance $\mathbf{x}_i$ is shadowed and the LS instances are within the dotted circle centered at $\mathbf{x}_i$.

LSCI is dominated by the computation of pairwise distances between all instances in $\mathcal{D}$, resulting in $O(m \cdot n^2)$ operations.

Figure 8 presents the computation of *LSCI* for the highlighted instances from the dataset in Figure 2. The local set of each instance $\mathbf{x}_i$ is contained in the dotted circle centered at it. A noisy instance surrounded by many instances from another class will tend have large *LSCI* values as defined by Equation 10. In contrast, easy instances close to many instances from their class will have low *LSCI* values. But the anomalous observation $\mathbf{x}_1$ will also present a low *LSCI* value, since its distance to a nearest enemy will be large and consequently its local set will encompass many other instances sharing its class. In the example, both $\mathbf{x}_1$ and $\mathbf{x}_2$ have the *LSCI* value of 0.25 and $\mathbf{x}_3$ and $\mathbf{x}_4$ achieve a *LSCI* value of 0.92.

Local Set Radius $LSR(\mathbf{x}_i)$ [1]: takes the normalized radius of $\mathbf{x}_i$'s local set:

$$LSR(\mathbf{x}_i) = 1 - \min\left\{1, \frac{d(\mathbf{x}_i, NE(\mathbf{x}_i))}{\max(d(\mathbf{x}_i, \mathbf{x}_j)|y_i = y_j)}\right\} \tag{11}$$

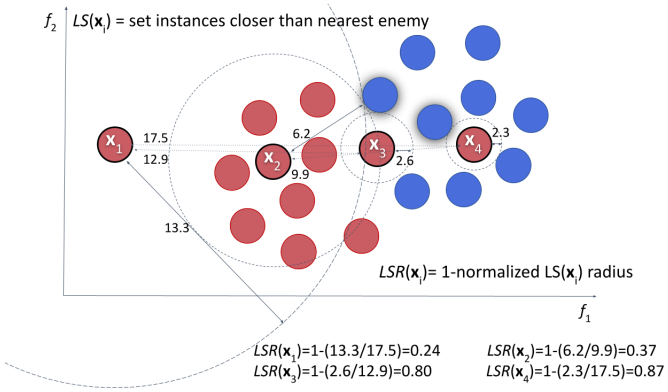


Fig. 9. Example of *LSR* computation for some instances of the dataset from Figure 2. The nearest enemy of an instance $\mathbf{x}_i$ is shadowed and the LS instances are within the dotted line centered at the instance. The radius size is presented as an arrow.

Larger radiuses are expected for easier instances, which are surrounded by many instances sharing their class, so we take the complement of such measure. This measure tends to present smoother values than *LSCI*. As in *LSCI*, the asymptotic cost of *LSR* is $O(m \cdot n^2)$.

Figure 9 presents the computation of the *LSR* measure for the highlighted instances from Figure 2. The *LSR* values are increasing for the instances in the following ascending order: $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ and $\mathbf{x}_4$. The values are clearer smoother than the *LSCI* values computed for the same instances.

Usefulness $U(\mathbf{x}_i)$ [1]: corresponds here to the complement of the fraction of instances having $\mathbf{x}_i$ in their local sets [19]:

$$U(\mathbf{x}_i) = 1 - \frac{\sharp\{\mathbf{x}_j | d(\mathbf{x}_i, \mathbf{x}_j) < d(\mathbf{x}_j, NE(\mathbf{x}_j))\}}{\sharp\{\mathbf{x}_j | y_i = y_j\}} \tag{12}$$

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 definition as output. The asymptotic computational cost of U is $O(m \cdot n^2)$, since the cost of computing the distance between all pairs of elements in the dataset is dominant.

Figure 10 presents the computation of the U measure for the highlighted instances from the dataset in Figure 2. The unique local set different from its own where the outlier instance $\mathbf{x}_1$ is contained is shown in the figure as a light red circle. This measure is able to indicate the outlier instance $\mathbf{x}_1$ is in a sparse region of the dataset, so that it has a large U value. The easiest instance in this HM is $\mathbf{x}_2$, which is in a dense region of a dataset from the same class it belongs.

Harmfulness $H(\mathbf{x}_i)$: number of instances having $\mathbf{x}_i$ as their nearest enemy [19]:

$$H(\mathbf{x}_i) = \frac{\sharp\{\mathbf{x}_j | NE(\mathbf{x}_j) = \mathbf{x}_i\}}{\sharp\{\mathbf{x}_j | y_i \neq y_j\}} \tag{13}$$

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 computing H is also $O(m \cdot n^2)$.

Figure 11 presents the computation of the H measure for the highlighted instances from Figure 2. The instances they are nearest enemy of are connected to the corresponding instances in the figure. Only $\mathbf{x}_3$ and $\mathbf{x}_4$ figure as nearest enemies of other instances. For the instances $\mathbf{x}_1$ and $\mathbf{x}_2$ the H value will be minimum (null). In fact, the H value will be null for other red instances in the example dataset, with the exception of $\mathbf{x}_3$ and $\mathbf{x}_4$.

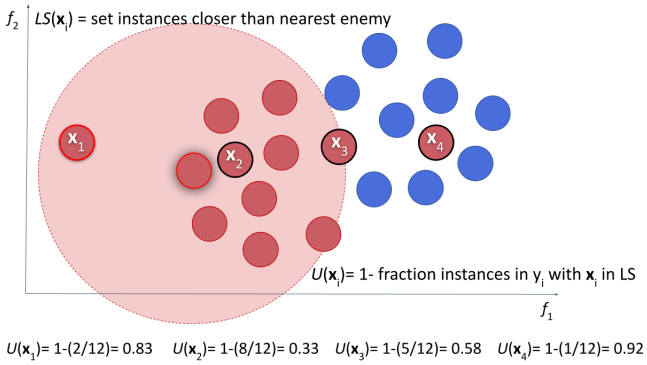


Fig. 10. Example of U computation for some instances in the dataset from Figure 2. The unique local set where $\mathbf{x}_1$ is contained (is useful) besides its own is represented as a light red circle.

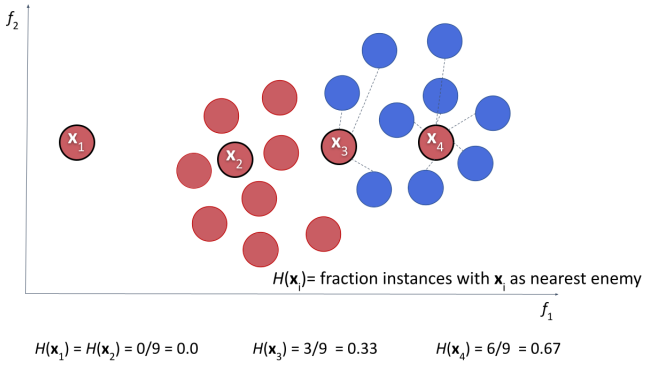


Fig. 11. Example of H computation for highlighted instances of the dataset from Figure 2. Only $\mathbf{x}_3$ and $\mathbf{x}_4$ figure as nearest enemies of other instances, to whom they are connected to.

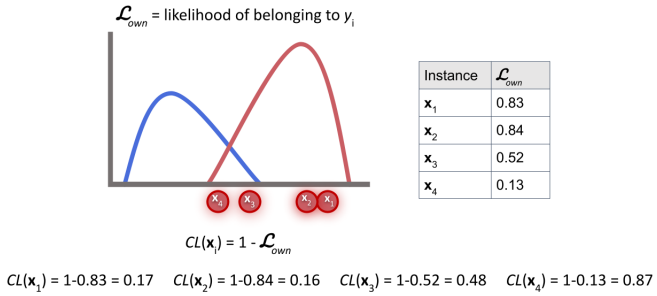
4.2 Class likelihood HM

The measures in this category try to capture whether an instance is well situated in its class regarding the general patterns of this class. For such, they estimate likelihoods which, for simplicity, consider each input feature independent from each other, as in Naive Bayes [49] classification.

Class Likelihood $CL(\mathbf{x}_i)$ [39]: measures the likelihood $\mathbf{x}_i$ belongs to y_i as:

$$CL(\mathbf{x}_i) = 1 - P(\mathbf{x}_i|y_i)P(y_i), \tag{14}$$

where $P(\mathbf{x}_i|y_i)$ represents the likelihood of $\mathbf{x}_i$ belonging to class y_i, measured in the dataset $\mathcal{D}$, and $P(y_i)$ is the prior of class y_i, which we set as $\frac{1}{n}$ for all data instances. In [39] the prior component is ignored so that the skeweness of the classes does not influence the results, but the effect is the same as using the previous prior. In turn, the conditional probability $P(\mathbf{x}_i|y_i)$ is estimated considering each of the input features independent from each other. Larger class likelihood values are expected for easier instances, so we output the complement of this value. The asymptotic computational cost to compute the probabilities from the dataset is $O(m \cdot n)$.



Instance	$\mathcal{L}_{own}$
$\mathbf{x}_1$	0.83
$\mathbf{x}_2$	0.84
$\mathbf{x}_3$	0.52
$\mathbf{x}_4$	0.13

Fig. 12. Example of CL computation for some instances of the dataset from Figure 2. The likelihoods each instance has to its class is shown in the accompanying table.

Figure 12 presents the computation of the CL values of instances $\mathbf{x}_1$ to $\mathbf{x}_4$ in Figure 2. The instance with the highest likelihood to the red class is $\mathbf{x}_2$, followed by $\mathbf{x}_1$. This is expected, since $\mathbf{x}_2$ is situated in a region with many instances of the red class, while $\mathbf{x}_1$ is in a sparse region. $\mathbf{x}_3$ and $\mathbf{x}_4$ have lowest likelihoods to the red class, as they are in overlapping regions of the classes.

Class Likelihood Difference $CLD(\mathbf{x}_i)$ [39]: takes the difference between the likelihood that $\mathbf{x}_i$ belongs to its class y_i and the maximum likelihood it has to any other class:

$$CLD(\mathbf{x}_i) = \frac{1 - \Big(P(\mathbf{x}_i|y_i)P(y_i) - \max_{y_j \neq y_i}[P(\mathbf{x}_i|y_j)P(y_j)]\Big)}{2}. \tag{15}$$

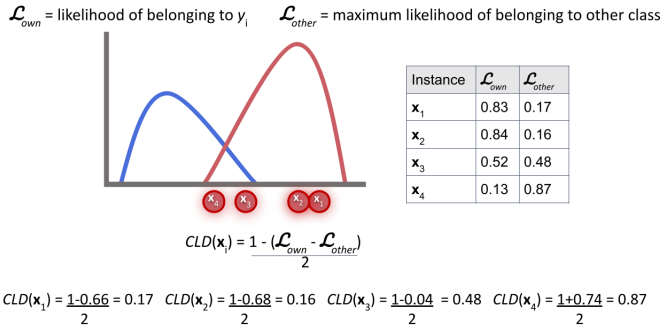
As in CL, $P(\mathbf{x}_i|y_i)$ represents the likelihood $\mathbf{x}_i$ belongs to class y_i and $P(y_i)$ is the prior of class y_i, which we set as $\frac{1}{n}$ for all data instances. Again, the conditional probability $P(\mathbf{x}_i|y_i)$ can be estimated considering each of the input features independent from each other. 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 difference as indicated in Equation 15. The probabilities can be calculated as in CL, requiring $O(m \cdot n)$ operations. One must notice that, for binary classification problems, CL and CLD as defined by Equations (14) and (15) will be the same, as $\max_{y_j \neq y_i}[P(\mathbf{x}_i|y_j)P(y_j)] = 1 - P(\mathbf{x}_i|y_i)P(y_i)$. This happens because all measures here are managed to present values bounded in the $[0, 1]$ interval. Therefore, the CLD values for the highlighted instances from the dataset in Figure 2 coincide with those of the CL measure, with $\mathbf{x}_2$ being the simplest instance, followed by $\mathbf{x}_1$. Next we have $\mathbf{x}_3$ and $\mathbf{x}_4$.

4.3 Feature-based HM

Another common source of difficulty in classification problems is the absence of informative input features [13, 21]. The measures of the feature-based category try to quantify if the instance has feature values out of overlapping regions of the classes. If so, it will be easier to classify than instances that are in overlapping regions for many of the input features.

Fraction of features in overlapping areas $F1I(\mathbf{x}_i)$ [1]: this measure outputs the percentage of input features of $\mathbf{x}_i$ whose values lie in an overlapping region of the classes, as defined in:

$$F1(\mathbf{x}_i) = \frac{\sum_{j=1}^{m} I(x_{ij} \geq \text{max\_min}(\mathbf{f}_j) \wedge x_{ij} \leq \text{min\_max}(\mathbf{f}_j))}{m}, \tag{16}$$



Instance	$\mathcal{L}_{own}$	$\mathcal{L}_{other}$
$\mathbf{x}_1$	0.83	0.17
$\mathbf{x}_2$	0.84	0.16
$\mathbf{x}_3$	0.52	0.48
$\mathbf{x}_4$	0.13	0.87

Fig. 13. Example of *CLD* computation for some instances of the dataset from Figure 2. The likelihoods of belonging to their classes and the maximum likelihood to the other class (which is unique here, as we have a binary classification problem) are illustrated in the companion table.

where m is the number of input features of $\mathcal{D}$, I is the indicator function (returns 1 if its argument is true and 0 otherwise), $\mathbf{f}_j$ is the j-th feature vector in $\mathcal{D}$ and:

$$\min\_\max(\mathbf{f}_j) = \min(\max(\mathbf{f}_j^{c_1}), \max(\mathbf{f}_j^{c_2})), \tag{17}$$

$$\max\_\min(\mathbf{f}_j) = \max(\min(\mathbf{f}_j^{c_1}), \min(\mathbf{f}_j^{c_2})). \tag{18}$$

$\max(\mathbf{f}_j^{y_i})$ and $\min(\mathbf{f}_j^{y_i})$ are the maximum and minimum values of $\mathbf{f}_j$ in a class $y_i \in \{c_1, c_2\}$ (the measure is defined for binary classification problems originally, for which $C = 2$). Multiclass classification problems are first decomposed into multiple pairwise binary classification problems (*one-versus-one* - OVO - strategy [35]), whose results are averaged. The asymptotic computational cost of this measure is $O(m \cdot n)$ for binary classification problems and $O(m \cdot n \cdot C)$ for multiclass problems, supposing that each of the classes has the same number of observations, that is, $\frac{n}{C}$.

According to Equation 16, the overlap for a feature $\mathbf{f}_j$ is measured according to the maximum and minimum values it assumes in two different classes. A feature has overlap if it is not possible to separate the classes using a threshold on that feature's values. Larger $F1I$ values are obtained for instances which lie in overlapping regions for most of the features. Figure 14 presents the computation of the $F1I$ measure for the instances in the dataset from Figure 2. The overlapping regions for each feature are highlighted in light red. The rectangular area within $\max\_\min(\mathbf{f}_1)$ and $\min\_\max(\mathbf{f}_1)$, $\max\_\min(\mathbf{f}_2)$ and $\min\_\max(\mathbf{f}_2)$ has a stronger color as it is the intersection of the overlapping areas of f_1 and f_2. Both $\mathbf{x}_1$ and $\mathbf{x}_2$ are in an overlapping area for feature f_2 only, presenting a $F1I$ value of 0.5. $\mathbf{x}_3$ and $\mathbf{x}_4$ are in the overlapping areas of both features f_1 and f_2, presenting a maximum $F1I$ value of 1.

Minimum/Mean/Maximum distance to overlapping of features $F2I(\mathbf{x}_i)/F3I(\mathbf{x}_i)/F4I(\mathbf{x}_i)$ [1]: other three feature-based measures are based on the distance of each instance to the center of the overlapping regions of the features. This distance is calculated for each feature according to Equation 19.

$$d_o(\mathbf{x}_i, \mathbf{f}_j) = \frac{\min\_\max(\mathbf{f}_j) - x_{ij}}{\min\max(\mathbf{f}_j) - \max\min(\mathbf{f}_j)}, \tag{19}$$

where the $\min\_\max$ and $\max\_\min$ values are defined in Equations 17 and 18, respectively. Next a transformation is applied so that a maximum hardness value is obtained for instances lying in the center of an overlapping

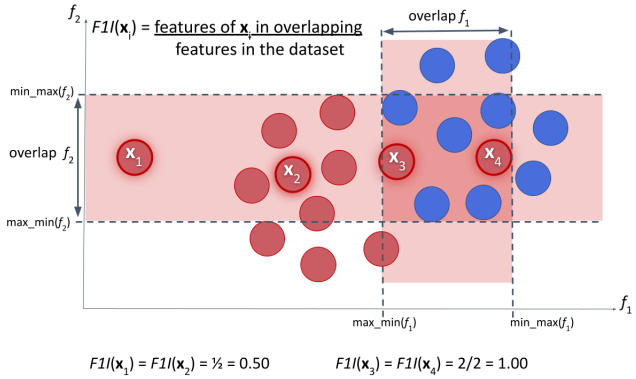


Fig. 14. Example of $F1I$ computation for some instances of the dataset from Figure 2. Overlapping areas are shadowed in light red.

region, as shown in Equation 20.

$$d_o^t(\mathbf{x}_i, \mathbf{f}_j) = \frac{1}{(1 + |0.5 - d_o(\mathbf{x}_i, \mathbf{f}_j)|)}. \tag{20}$$

Since the dataset has m features, the measures taken are the minimum ($F2I$ - Equation 21), mean ($F3I$ - Equation 22) and maximum ($F4I$ - Equation 23) of the transformed distance values registered for an instance in relation to each feature.

$$F2I(\mathbf{x}_i) = \min_{i=1}^{m} d_o^t(\mathbf{x}_i, \mathbf{f}_j). \tag{21}$$

$$F3I(\mathbf{x}_i) = \frac{1}{m} \sum_{i=1}^{m} d_o^t(\mathbf{x}_i, \mathbf{f}_j). \tag{22}$$

$$F4I(\mathbf{x}_i) = \max_{i=1}^{m} d_o^t(\mathbf{x}_i, \mathbf{f}_j). \tag{23}$$

The asymptotic cost of these measures is similar to that of $F1I$. They are all originally designed to work with binary classification problems, which can be first divided according to the OVO strategy and have the results averaged for the multiple pairwise classification problems generated.

Figure 15 presents the computation of the $F2I$, $F3I$ and $F4I$ values for the highlighted instances from Figure 2. An accompanying table shows the d_o^t values for each of the instances. All instances are close to the center of the overlapping region of feature f_2. $\mathbf{x}_1$ presents the lowest $F2I$ value, because it is far from the overlapping region of the f_1 feature. This is also the instance with the lowest $F3I$ value, since in average it is also farthest from the center of the overlapping regions of the features. But for $F4I$, the lowest value is observed for $\mathbf{x}_2$, which is farthest from the center of the overlapping region for feature f_2 compared to the other instances. $\mathbf{x}_3$ and $\mathbf{x}_4$ present identical values for the HM $F2I$, $F3I$ and $F4I$, at least using two decimal places of precision.

4.4 Tree-based HM

As stated in [39], Decision Tree (DT) models [33] can be used to estimate the minimum description length (Kolmogorov complexity) of an instance. The measures from the tree-based category use such models for estimating the hardness

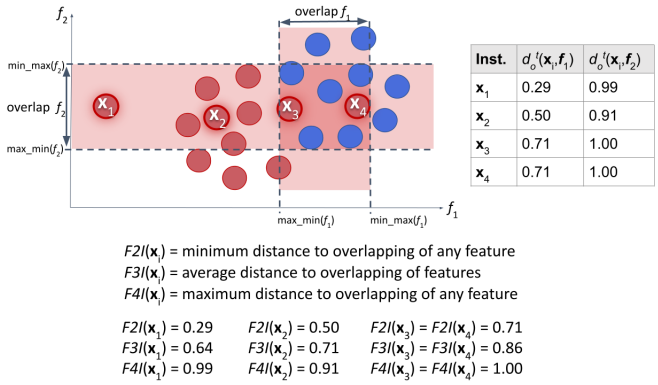


Fig. 15. Example of $F2I$, $F3I$ and $F4I$ computation for some instances of the dataset from Figure 2. Overlapping areas are shadowed in light red. The adapted distances to the center of the overlapping of the features are presented in the accompanying table.

level of each instance in a dataset. If many splits are required to classify an instance when a decision tree is built from $\mathcal{D}$, this instance is hard to classify and it will be easy otherwise.

Disjunct Size $DS(\mathbf{x}_i)$ [39]: builds an unpruned decision tree DT using $\mathcal{D}$ and verifies the relative size of the disjunct (leaf node) where $\mathbf{x}_i$ is placed:

$$DS(\mathbf{x}_i) = 1 - \frac{\sharp\{\mathbf{x}_j | \mathbf{x}_j \in Disjunct(\mathbf{x}_i)\}}{\max |Disjunct(\mathcal{D})|}, \tag{24}$$

where $Disjunct(\mathbf{x}_i)$ represents the set of instances contained in the disjunct where $\mathbf{x}_i$ is placed and the denominator is the size of the largest disjunct obtained for the dataset $\mathcal{D}$. A larger disjunct size is expected for easier instances, so we output the complement of the relative disjunct size. Hard instances, on the other hand, will require many splits to be correctly classified by the tree and will tend to be placed in small disjuncts. Building the DT model dominates the asymptotic computational cost of this measure, which can be performed at $O(m \cdot n \cdot \log_2 n)$ [36].

The computation of the DS measure for the representative instances from Figure 2 is shown in Figure 16. The leaf nodes where the instances are placed are shadowed. The distribution of examples per class can be measured by the number of instances per class reaching each of the leaf nodes. The largest disjunct is the one that contains $\mathbf{x}_1$ and $\mathbf{x}_2$, with 10 elements, so that both $\mathbf{x}_1$ and $\mathbf{x}_2$ are classified in an upper leaf containing many elements of their class. $\mathbf{x}_3$ and $\mathbf{x}_4$, in turn, require many divisions to get correctly classified and are placed in a small disjunct in one of the deepest leaf nodes of the tree.

Disjunct Class Percentage $DCP(\mathbf{x}_i)$ [39]: builds a pruned decision tree using $\mathcal{D}$ and considers the percentage of instances in the disjunct of $\mathbf{x}_i$ which share the same label as $\mathbf{x}_i$:

$$DCP(\mathbf{x}_i) = 1 - \frac{\sharp\{\mathbf{x}_j | \mathbf{x}_j \in Disjunct(\mathbf{x}_i) \wedge y_j = y_i\}}{\sharp\{\mathbf{x}_j | \mathbf{x}_j \in Disjunct(\mathbf{x}_i)\}}, \tag{25}$$

Easier instances will have a larger percentage of examples sharing the same label as them in their disjunct in the pruned DT, so we output the complement of this percentage. As in DS, the asymptotic cost of DCP is $O(m \cdot n \cdot \log_2 n)$.

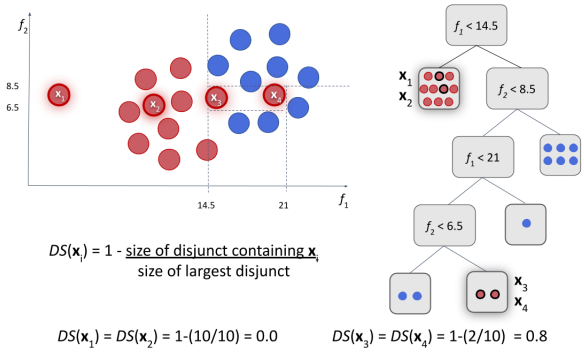


Fig. 16. Example of DS computation for highlighted instances of the dataset from Figure 2. The partitions made by the DT model are illustrated in the left, whilst the tree is in the right of the figure. The instances placed in each disjunct are shown as dots inside them.

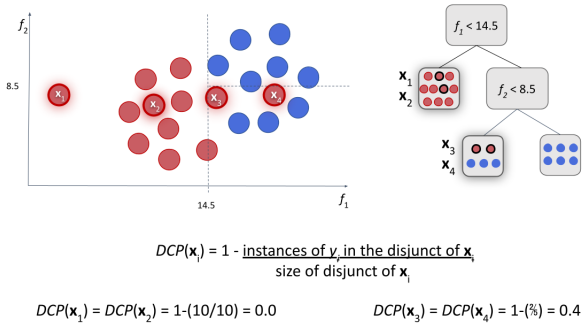


Fig. 17. Example of DCP computation for highlighted instances of the dataset from Figure 2. The partitions made by the pruned DT model are illustrated in the left, whilst the pruned tree is in the right of the figure. The instances placed in each disjunct are shown as dots inside them.

Figure 17 presents the computation of the DCP measure for the highlighted instances from Figure 2. The unique difference compared to DS is that the disjuncts in the pruned tree can contain a mixture of classes. This makes the DCP value for $\mathbf{x}_3$ and $\mathbf{x}_4$ lower than the DS values obtained for the same instances.

Tree Depth $TD(\mathbf{x}_i)$ [39]: 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 $\mathcal{D}$:

$$TD(\mathbf{x}_i) = \frac{depth_{DT}(\mathbf{x}_i)}{\max(depth_{DT}(\mathbf{x}_j \in \mathcal{D}))}, \tag{26}$$

where $depth_{DT}(\mathbf{x}_i)$ gives the depth where the instance $\mathbf{x}_i$ is placed in the DT. This measure has two versions, one in which the DT is pruned (TD_P) and another in which the DT is unpruned (TD_U). The need to build DT models makes this measure have an asymptotic cost of $O(m \cdot n \cdot \log_2 n)$.

Harder to classify instances tend to be placed at deeper levels of the tree and present higher TD values. The computation of the two versions of this measure for the dataset from Figure 2 is presented in Figure 18. The

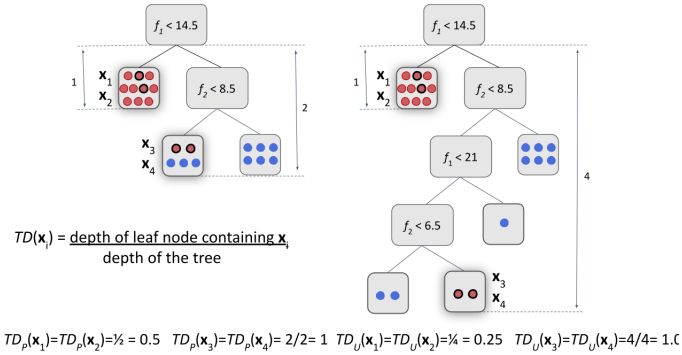


Fig. 18. Example of TD computation for highlighted instances of the dataset from Figure 2. There are two versions of this measure, one using a pruned DT (TD_P) and another using an unpruned DT (TD_U).

instances $\mathbf{x}_3$ and $\mathbf{x}_4$ are placed in the deeper level of the trees, presenting maximum TD values. In contrast, $\mathbf{x}_1$ and $\mathbf{x}_2$ are in the shallower leaf of the trees and present smaller TD values. TD_U tends to present a larger variation of possible values, as the tree will present greater depth. One must also notice that, for very simple problems where one unique split is able to separate the classes (as a decision stump model), the TD values as expressed by Equation 26 will be 1 (maximum) for all instances in the dataset, despite the simplicity of the problem. This is a side effect of our standardization procedure of dividing the depth where the instances are placed to the maximum tree depth, which will coincide for all instances in this case.

4.5 Class balance HM

Class balance is also an aspect which may influence in classification complexity [21]. If a dataset has a high skewness in the proportion of examples per class, the ML techniques will tend to fail in the minority class examples [10]. Therefore, the class balance category of HM will characterize whether an instance is in a minority class, making it more prone to classification errors and more difficult, or in a majority class, when it will tend to be easier to classify.

Class Balance $CB(\mathbf{x}_i)$ [39]: measures the skewness of $\mathbf{x}_i$'s class as:

$$CB(\mathbf{x}_i) = \left(1 - \frac{\sharp\{\mathbf{x}_j | y_j = y_i\}}{n} + \frac{1}{C}\right)\left(\frac{C}{C+1}\right), \tag{27}$$

where n is the number of instances in the dataset $\mathcal{D}$ and C is the number of classes the problem has. The measure had its definition adapted here so that it is bounded in the $[0, 1]$ interval. CB will be null for all instances if the problem is balanced, which is simpler concerning the class balance aspect. The higher the CB value, the more difficult the instance shall be to classify concerning the class balance aspect, that is, the class it belongs is rare and presents a low number of instances. The cost of this measure is $O(n)$, involving counting the number of elements of the dataset.

Figure 19 presents the computation of the CB measure for the highlighted instances from the dataset in Figure 2. All instances are from the red class, which contains the majority of the examples in the dataset and share the same CB value.

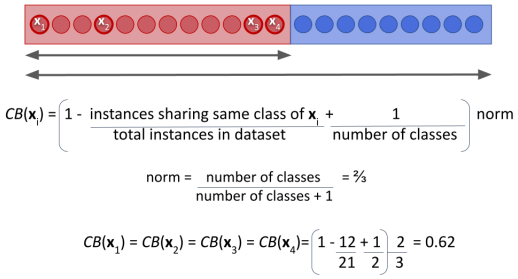


Fig. 19. Example of CB computation for some instances of the dataset from Figure 2. They all belong to the red class, which is the majority class in this particular dataset.

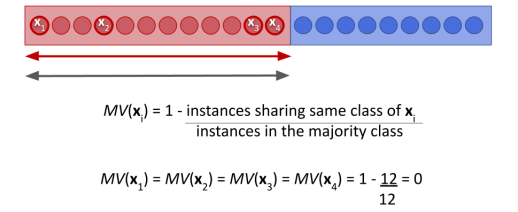


Fig. 20. Example of MV computation for some instances of the dataset from Figure 2. They all belong to the red class, which is the majority class in this particular dataset.

Majority Value $MV(\mathbf{x}_i)$ [39]: also measures the skewness of $\mathbf{x}_i$'s class, with an alternative formulation based on the imbalance ratio of the class the instance belongs to:

$$MV(\mathbf{x}_i) = 1 - \frac{\sharp\{\mathbf{x}_j | y_j = y_i\}}{\max_{y_j \in \mathcal{Y}} \left\{\sharp\{\mathbf{x}_j | \mathbf{x}_j \in y_j\}\right\}}, \tag{28}$$

The first term of Equation 28 resambles the imbalance ratio commonly used to characterize the skewness of imbalanced problems [10]. MV will be null if the instance belongs to the majority class, in which case it is simpler to classify concerning the class balance aspect. It will be close to 1 if the instance belongs to a very rare class, with a low number of instances compared to the majority class. The asymptotic cost of this measure is $O(n)$ for counting the number of elements per class in the dataset.

Figure 20 presents the computation of the MV measure for the highlighted instances from the dataset in Figure 2. All instances are from the red class, which contains the majority of the examples in the dataset and share the same minimum MV value.

4.6 PyHard package

Table 1 presents a summary of the characteristics of the HM presented previously. The table presents, for each measure, the following information: category, name, acronym ("Acron." column), minimum ("Min") and maximum ("Max") values achievable and worst case asymptotic computational cost for computing them. All of the measures were bounded in the

Category	Measure	Acron.	Min	Max	Asymptotic cost
Neighborhood	k-Disagreeing Neighbors	kDN	0	1	$O(m \cdot n)$
	Frac. nearby instances different class at instance-level	$N1I$	0	1	$O(m \cdot n^2)$
	Ratio of intra-extra class distances at instance level	$N2I$	0	≈ 1	$O(m \cdot n^2)$
	Local set cardinality at instance-level	$LSCI$	0	≈ 1	$O(m \cdot n^2)$
	Local set radius	LSR	0	1	$O(m \cdot n^2)$
	Usefulness	U	≈ 0	1	$O(m \cdot n^2)$
	Harmfulness	H	0	≈ 1	$O(m \cdot n^2)$
Likelihood	Class Likelihood	CL	0	1	$O(m \cdot n)$
	Class Likelihood Difference	CLD	0	1	$O(m \cdot n)$
Feature-based	Frac. features in overlapping areas	$F1I$	0	1	$O(m \cdot n \cdot C)$
	Min. distance to overlapping areas of features	$F2I$	0	1	$O(m \cdot n \cdot C)$
	Mean distance to overlapping areas of features	$F3I$	0	1	$O(m \cdot n \cdot C)$
	Max. distance to overlapping areas of features	$F4I$	0	1	$O(m \cdot n \cdot C)$
Tree-based	Disjunct Size	DS	0	1	$O(m \cdot n \cdot \log_2 n)$
	Disjunct Class Percentage	DCP	0	1	$O(m \cdot n \cdot \log_2 n)$
	Tree Depth (pruned)	TD_P	≈ 0	1	$O(m \cdot n \cdot \log_2 n)$
	Tree Depth (unpruned)	TD_U	≈ 0	1	$O(m \cdot n \cdot \log_2 n)$
Class Balance	Class Balance	CB	0	1	$O(n)$
	Majority Value	MV	0	1	$O(n)$

Table 1. Summary of the hardness meta-features surveyed in this work.

[0,1] interval and the higher the value, the higher the indicated difficulty level. These standardizations allow to better understand and interpret the values of the measures.

Concerning the worst case computational cost of the measures, all of them are polynomials in the number of features m and observations n (and some of them on the number of classes C too). But one must notice that many of the measures can take advantage of the information of a common data structure which can be computed only once, saving computational time. This is the case of most of the neighborhood measures, which require building a pairwise distance matrix between all observations in $\mathcal{D}$ at an asymptotic cost of $O(m \cdot n^2)$, which can be computed once and can be reused afterwards to obtain all of these measures (namely $N1I$, $N2I$, $LSCI$, 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 them. DS and TD_U can use the same unpruned DT model and DCP and TD_P can use the same pruned DT model induced from $\mathcal{D}$. For class-likelihood HM, the Naive Bayes classification rule can also be reused for CL and CLD. Finally, the min _ max and max _ min values of the features (Equations 17 and 18) can be reused for all feature-based measures.

There is also a natural correlation between the values of the measures in a same category, which are based on similar concepts. In a binary classification problem, for example, CL and CLD as defined in this paper will be the same. But even for more classes these measures can show some correlation of values. CLD can be considered more complete and informative in measuring the hardness level of an instance compared to CL, as it compares the likelihoods of different classes. The same reasoning applies to DCP and DS, in which case DCP might be a better choice by taking the mixture of classes in a tree leaf into account. Finally, TD_U will tend to be more informative than TD_P for identifying noisy instances, since many splits will be needed to separate them from the other instances and they will be placed at deeper levels of the unpruned DT. In the neighborhood-based category, measures such as $N2I$ and $LSCI$ will tend to present smoother and more varied values. H will be different from null only for instances in overlapping regions of the classes.

Category	HM	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$
Neighborhood	*kDN*	0.00 (1.5)	0.00 (1.5)	0.60 (3)	**1.00** (4)
	N1I	0.00 (1.5)	0.00 (1.5)	0.75 (3)	**1.00** (4)
	N2I	0.31 (2)	0.26 (1)	0.54 (3)	**0.71** (4)
	LSCI	0.25 (1.5)	0.25 (1.5)	**0.92** (3.5)	**0.92** (3.5)
	LSR	0.24 (1)	0.37 (2)	0.80 (3)	**0.87** (4)
	U	0.83 (3)	0.33 (1)	0.58 (2)	**0.92** (4)
	H	0.00 (1.5)	0.00 (1.5)	0.33 (3)	**0.67** (4)
Likelihood	*CL*	0.17 (2)	0.16 (1)	0.48 (3)	**0.87** (4)
	CLD	0.17 (2)	0.16 (1)	0.48 (3)	**0.87** (4)
Feature-based	*F1I*	0.50 (1.5)	0.50 (1.5)	**1.00** (3.5)	**1.00** (3.5)
	F2I	0.29 (1)	0.50 (2)	**0.71** (3.5)	**0.71** (3.5)
	F3I	0.64 (1)	0.71 (2)	**0.86** (3.5)	**0.86** (3.5)
	F4I	0.99 (2)	0.91 (1)	**1.00** (3.5)	**1.00** (3.5)
Tree-based	*DS*	0.00 (1.5)	0.00 (1.5)	**0.80** (3.5)	**0.80** (3.5)
	DCP	0.00 (1.5)	0.00 (1.5)	**0.40** (3.5)	**0.40** (3.5)
	TD_P	0.50 (1.5)	0.50 (1.5)	**1.00** (3.5)	**1.00** (3.5)
	TD_U	0.25 (1.5)	0.25 (1.5)	**1.00** (3.5)	**1.00** (3.5)
Class balance	*CB*	**0.62** (2.5)	**0.62** (2.5)	**0.62** (2.5)	**0.62** (2.5)
	MV	**0.00** (2.5)	**0.00** (2.5)	**0.00** (2.5)	**0.00** (2.5)
Average ranking		1.7	1.6	3.1	**3.6**

Table 2. Summary of the HM values computed for the instances highlighted in Figure 2. The lowest values per row are underlined and the highest values are boldfaced. In parenthesis we show the ranking of the HM values per instance. When there are ties, an average of the involved positions is shown. The last row presents the average of the ranks for each instance.

And *U* will be able to characterize if an instance is in a dense region of its class or not. In the class balance category, *MV* might be preferred to *CB* because the variation of the values achieved is less dependent of the number of classes the dataset has. Finally, in the feature-based category, *F1I* is the simplest measure and will present a low variation of possible values (for the dataset from Figure 2, for example, the possible values achievable are 0, 0.5 or 1). But it can be tricky to interpret the hardness level measured by the other measures from this category. Overall, it is advisable to extract all HM for the instances of a dataset and to perform a (meta)feature-selection for determining the HM of interest to use.

Table 2 summarizes the values of the HM computed for the four instances highlighted in the dataset from Figure 2. Lower values per HM are underlined and higher values are highlighted in boldface. In parenthesis we indicate the ranking of the instance concerning its hardness level according to the corresponding HM compared to those of the other instances. In the case of ties, an average of the involved positions is indicated. For instance, for the *kDN* HM, instances $\mathbf{x}_1$ and $\mathbf{x}_2$ tie in the first two positions of the ranking, so that an average ranking of 1.5 is shown within parenthesis for both of them. Next comes $\mathbf{x}_3$ in the third position and $\mathbf{x}_4$ in the fourth position as the hardest to classify instance. In the last row, the average of the rankings each of the instances assume is shown. $\mathbf{x}_2$ was in average the easiest instance, followed closely by $\mathbf{x}_1$. Next come, in this order, $\mathbf{x}_3$ and $\mathbf{x}_4$. This is in accordance to what was expected given the placement of the instances in the input space. Interestingly, few HM were able to pinpoint $\mathbf{x}_1$ as an anomalous instance from the red class, being *U* the most effective in doing so.

All measures presented in Table 1 have been gathered in a Python package named PyHard[2]. The package also includes a projection tool of the datasets according to their hardness level using an Instance Space Analysis framework [25], as described in [27]. One can run the tool using the option `pyhard run -no-isa` to withdraw this projection step and obtain the HM only, specifying the set of measures to be calculated in a configuration file (`config.yaml`). When computing distances in the neighborhood-based HM, the PyHard tool uses the Gower distance measure, which is heterogeneous and admits data with both qualitative and quantitative features. The DT and NB models are from the `scikit-learn` package [29].

PyHard also implements the instance hardness average of Equation 2, including as pool of classifiers $\mathcal{L}$ the following ML algorithms: Bagging (Bag) [5], Gradient Boosting (GB) [11], Support Vector Machines (SVM, with both linear and RBF kernels) [44], Multilayer Perceptron (MLP) [46], Logistic Regression (LR) [3] and Random Forest (RF) [6]. Other algorithms can be easily added if desired. The probabilities of misclassification for $\mathcal{D}$ are estimated in a stratified K-fold cross validation procedure, with $K = 5$ by default. The hyperparameters of the techniques are tuned by an internal cross-validation procedure and the probabilities of misclassification are adjusted using Platt scaling [4].

A recent work worth mentioning is [17], which introduces the notion of *hostility* for assessing the complexity/difficulty of a classification problem at a multi-level resolution. The hostility at the instance-level is defined as the difficulty of correctly classifying an instance given its neighborhood. The neighborhoods are defined by a recursive and hierarchical application of the k-means algorithm, whose results are aggregated. The hostility measure can be taken at the instance, class and dataset-levels and can be regarded as a representative of the neighborhood HM, resembling kDN, but giving smoother results[3].

5 INSTANCE HARDNESS IN USE

Instance hardness measures have been adopted in literature for different purposes, like noise filtering [39], dynamic classifier selection [7], selection of subspaces in decision trees [42], characterizing local samples in biometric systems [43], dealing with imbalanced problems [47], to name a few. Often these works aim to improve predictive performance, by building ML models that are robust to deal with instance regions with different difficulties. In this section, we are focused on discussing examples of using instance hardness measures in topics that are more closely related to trustworthy AI. Specifically we discuss the use of instance hardness in three dimensions: fairness, explainability and reliability. Dealing with such topic requires that developers, analysts and stakeholders move from a (global) dataset-level analysis of model behaviour and task characteristics to a (local) instance-wise analysis.

In literature, we identified previous works that directly adopted the term instance hardness in their proposals, as well explicitly made use of the instance hardness measures reviewed in the previous sections. Other previous works adopted ideas that are shared with instance hardness, using other terms like uncertainty, confidence and distrust. In this section, we make these connections explicit, aiming to open new ideas and leverage a more systematic use of instance hardness.

5.1 Fairness

Under unfair situations, ML model behaviour usually changes for individuals or groups with sensitive attributes. A possible unfair behaviour is a lower predictive performance for individuals in the sensitive group. This is the basis for different fairness indicators (e.g., equal opportunity, treatment equality), which consider the mistakes obtained by a model for particular instances. Unfair behaviour can be a consequence of training data bias, which can be linked to

[2] https://pypi.org/project/pyhard/
[3] The implementation is available at https://github.com/URJCDSLab/Hostility_measure/tree/main/Algorithm_code

the different concepts of instance hardness. For example, an annotator may be biased in such a way that label noise is frequent among the sensitive instances. Many instance hardness measures are designed exactly to quantify label uncertainty.

A straightforward idea to detect unfairness is to inspect the distributions of hardness measures across instances in different groups. In [48], the kDN hardness is extended to quantify data bias regarding a sensitive attribute $A \in \{0, 1\}$. A hardness bias measures $\Gamma_A(y)$ is defined as the Kullback–Leibler divergence (KL) of distributions of kDN for instances with different values of the sensitive attribute A:

$$\Gamma_A(y) = KL(f(kDN(\mathbf{x})|Y = y, A = 1) - f(kDN(\mathbf{x}, y)|Y = y, A = 0)), \tag{29}$$

where $f(kDN(\mathbf{x})|Y = y, A = a)$ is the density of the kDN distribution of all instances with $A = a$ and $Y = y$. If the hardness measure distribution changes in a given sensitive group, classification performance for that group can be affected as well.

This idea can be extended to other instance hardness measures in literature, i.e., inspect the hardness distribution across instances in a sensitive group and assume that unfair bias is more likely for groups with more difficult instances. Particularly, comparing model based hardness measures (Section 3.1) for different sensitive groups is similar to the *equal opportunity* measures derived in previous works like [12].

In [27] an approach named Instance Space Analysis (ISA) [25] is adapted to obtain a hardness embedding of a dataset in a bi-dimensional plot, which can be visually inspected. The objective is to place the instances in this space so that linear trends of classification difficulty are evidenced. The authors generated this embedding for two versions of the well known COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) dataset: with and without a sensitive attribute. The hardness embeddings are very similar and the instances falling in difficult and easy regions have similar biases towards the sensitive race attribute, showing the bias is embed within the dataset. Anomalous behaviors regarding decisions face to the juvenile offenses count were also observed in the analysis.

5.2 Explainable ML

There is a huge literature on explainability for ML, usually focused on explaining the predictions given by a model. In this paper, we focus on a different perspective, which is to explain the difficulty of instances. Obviously, such perspectives are complementary. On one hand, the user may want to understand why a particular prediction was given to a particular instance. On the other hand, the user may want to know whether (or when) such predictions are reliable. In post-hoc explanations, for example, if the user understands that an instance is extremely hard, the prediction could be rejected. In ex-ante explanations, if hardness is high for local regions in the instance space, corrective procedures (e.g., data augmentation, label correction) can be adopted at development time to enhance a dataset and train a better model.

A straightforward idea to connect xAI and instance hardness is to train a meta-model, in which the original problem features are used as predictors and instance hardness is used as the target [23]. In such case, applying xAI techniques on the top of such meta-models can be useful to understand what makes instances hard in a problem. In [23], the authors trained regression models to predict two different instance hardness measures: the average error obtained by a pool of classifiers (similar to [39]) and the IRT difficulty. Different regression models were evaluated in different tasks. Also, a variable importance method is used to inspect which attributes make an instance hard.

In [30], the authors proposed a subgroup discovery method to find Error Distribution Rules (EDRs) to uncover atypical error regions of a given prediction model. Each EDR represents a region in instance space in which model performance deviates and provides an interpretable description of these regions. A similar approach was proposed in [32], but limiting the number of attributes in the rules in order to improve interpretability. The target attribute in the EDRs is the error obtained by a given model. A possible extension would be to derive EDRs using as target attribute other instance hardness measures that are not limited to a single model. In such, case one could derive local regions of instances that are intrinsically difficult, like [23].

In [14], the authors investigated the problem of predicting surgical outcomes using ML and developed a trustworty framework based on instance hardness. The main ideia is to improve trust in the ML models by characterising difficult patients to predict and identifying the sources of difficulty in terms of instance hardness measures. The authors identified for example that model errors are usually explained due to globalised and localised class overlap, considering the high values of kDN and CLD. A framework called Artificial Intelligence with Trust by InterrogAtion (AITIA) was proposed to include a model interrogation process to check which models are more robust to deal with hard instances.

5.3 Reliability

An important dimension in trustworthy ML is to check whether the predictions provided by a ML model are reliable or not. In case the prediction is not reliable for a test instance, it can be rejected and sent to the human operator to make a final decision. This is a procedure adopted to monitor a trained model at deployment time, which can be investigated in literature using terms like classification with reject option or cautious classification.

Reliability is closely related to instance hardness, as model predictions for hard instances tend to be uncertain. In [38], for example, a rejection score is computed for each input instance by considering two criteria: (1) the density principle, which indicates if the input instance is close to training examples; and (2) the local fit principle, which indicates if the model is accurate for the training samples close to input instance. In [38], an instance score called Resampling Uncertainty Estimation is proposed to cover the two criteria, by estimating the amount that a prediction changes if the model is trained with different training data. An ensemble of models is learned from bootstrapping the training set and the variance of predictions for a test instance is used as rejection score.

The above idea can be addressed by adopting other instance hardness measures adopted in literature. Density fit principle in essence measures if the test instance is well represented in the training set or lies in dense areas in the instance space, which can be addressed by neighborhood-based measures. The local fit in turn measures the class complexity in the neighborhood of the test instance, which can be directly addressed by model-based instance hardness and also by different hardness meta-features.

Another line of research in reliability is to measure instance hardness once assuming that a model prediction is true. For example, in [15] the authors defined a trust score for each instance at deployment time as the ratio between the distance from the instance to the nearest class different from the predicted class and the distance to the predicted class. This is equivalent to use the IntraInter function in the $N2I$ hardness measure (see Equation 6). This idea can be generalized to use other instance hardness measures by the following procedure: (1) assume that a model prediction for the test instance is true; (2) measure the instance hardness; (3) use the instance hardness as distrust score (or alternatively its inverse as a trust score).

An analysis of a COVID prognosis dataset in [27] using hardness embeddings obtained by the ISA framework using HM allowed to identify problematic instances requiring further inspection. This inspection revealed the presence of label

noise in the dataset that should be corrected for building more reliable ML prognostic models. This type of data-centric analysis is important to identify issues on training data available for the ML models before they are deployed [28].

6 CONCLUSION

This paper surveyed different strategies for estimating the hardness level of each individual instance composing a labeled dataset for classification problems in ML. This type of information is important to better understand and characterize the variation of predictive performance of ML models for different regions of a dataset. Trustness in the predictions cannot be assured in average across a dataset and the instance-level analysis fulfills this perspective of a more fine-grained analysis.

Although some recent literature on how instance-level hardness analysis can contribute to increase trust of ML models, there are still many venues of research to be developed. By inspecting and characterizing the correct/incorrect predictions of a model, the ML analyst can define trust/distrust areas in the instance space, thus foreseeing whether a model is reliable or not. Moving from a single model to a pool of diverse models to measure IH, one can identify that certain instances are not only hard for a single model, but they may be intrinsically difficult to be predicted. While the model-based IH techniques can be used to define *which* instances are difficult, IH meta-features (hardness measures - HM) can be used in turn to explain *why* they are difficult. Applying xAI techniques on the top of HM meta-features is an interesting line of research to distinguish between different sources of difficulty. Explanations of hardness could be derived using both the original features of a problem (e.g., a particular disease is difficult to be predicted for older than 60 patients) and meta-features (e.g., there is a lot of class label noise among older than 60 patients). The former explanations would be focused on the stakeholders, while the second type of explanations would be focused on the ML developers and analysts.

It is always possible to define new meta-features able to characterize the hardness level of the instances. Most of the existent measures focus in the overlapping of classes as major source of classification difficulty. But it would be interesting to regard also on the density/sparsity of the classes and their structure. Identifying a core set of measures that are non-correlated and provide a better understanding of the hardness profile of a dataset is also of interest. And there are still few work trying to characterize the difficulty of regression problems, which provide a large venue for future work in the area.

Finally we highlight that IH measures have been usually adopted at training (development) time, to improve training data quality or the learning process itself. It would be very opportune too to investigate the use of IH measures at testing (deployment) time, for example to monitor the predictions of a learning system. For example, the hardness of a test instance could be predicted by inspecting the difficult of similar instances in the training set. In such case, if a test instance is expected to be hard, a rejection option could be triggered. Monitoring difficulty at testing time would be useful too to detect data shifts, which would require some intervention like model retraining or adaptation.

ACKNOWLEDGEMENTS

The authors thank the Brazilian Research Agencies FAPESP (grant 2021/06870-3) and CNPq.

REFERENCES

[1] José LM Arruda, Ricardo BC Prudêncio, and Ana C Lorena. 2020. Measuring instance hardness using data complexity measures. In *Brazilian Conference on Intelligent Systems*. Springer, 483–497.

[2] Victor H Barella, Luis PF Garcia, Marcilio CP de Souto, Ana C Lorena, and André CPLF de Carvalho. 2021. Assessing the data complexity of imbalanced datasets. *Information Sciences* 553 (2021), 83–109.

[3] Christopher M Bishop and Nasser M Nasrabadi. 2006. *Pattern recognition and machine learning*. Vol. 4. Springer.

[4] Björn Böken. 2021. On the appropriateness of Platt scaling in classifier calibration. *Information Systems* 95 (2021), 101641.

[5] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

[6] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[7] André L Brun, Alceu S Britto, Luiz S Oliveira, Fabricio Enembreck, and Robert Sabourin. 2016. Contribution of data complexity features on dynamic classifier selection. In *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 4396–4403.

[8] David Charte, Francisco Charte, and Francisco Herrera. 2021. Reducing data complexity using autoencoders with class-informed loss functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 12 (2021), 9549–9560.

[9] Yu Chen, Telmo M. Silva Filho, Ricardo B. Prudencio, Tom Diethe, and Peter Flach. 2019. β^3-IRT: A New Item Response Model and its Applications. In *Proceedings of Machine Learning Research (Proceedings of Machine Learning Research, Vol. 89)*. PMLR, 1013–1021.

[10] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. 2018. *Learning from imbalanced data sets*. Vol. 11. Springer.

[11] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[12] Moritz Hardt, Eric Price, Eric Price, and Nati Srebro. 2016. Equality of Opportunity in Supervised Learning. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf

[13] Tin Kam Ho and Mitra Basu. 2002. Complexity measures of supervised classification problems. *IEEE transactions on pattern analysis and machine intelligence* 24, 3 (2002), 289–300.

[14] Andrew Houston, Georgina Cosma, Phillipa Turner, and Alexander Bennett. 2021. Predicting surgical outcomes for chronic exertional compartment syndrome using a machine learning framework with embedded trust by interrogation strategies. *Scientific Reports* 11, 1 (2021), 24281.

[15] Heinrich Jiang, Been Kim, Melody Y. Guan, and Maya R. Gupta. 2018. To Trust Or Not To Trust A Classifier. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 5546–5557.

[16] Joanna Komorniczak, Paweł Ksieniewicz, and Michał Woźniak. 2022. Data complexity and classification accuracy correlation in oversampling algorithms. In *Fourth International Workshop on Learning with Imbalanced Domains: Theory and Applications*. PMLR, 175–186.

[17] Carmen Lancho, Isaac Martín De Diego, Marina Cuesta, Víctor Acena, and Javier M Moguerza. 2023. Hostility measure for multi-level study of data complexity. *Applied Intelligence* 53, 7 (2023), 8073–8096.

[18] E. Leyva, A. González, and R. Pérez. 2014. A Set of Complexity Measures Designed for Applying Meta-Learning to Instance Selection. *IEEE Transactions on Knowledge and Data Engineering* 27, 2 (2014), 354–367.

[19] Enrique Leyva, Antonio González, and Raúl Pérez. 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 (2015), 1523 – 1537.

[20] Ling Li and Yaser S Abu-Mostafa. 2006. Data complexity in machine learning. *Technical Report CaltechCSTR:2006.004* (2006).

[21] Ana C Lorena, Luís PF Garcia, Jens Lehmann, Marcilio CP Souto, and Tin Kam Ho. 2019. How Complex is your classification problem? A survey on measuring classification complexity. *ACM Computing Surveys (CSUR)* 52, 5 (2019), 1–34.

[22] Fernando Martínez-Plumed, Ricardo BC Prudêncio, Adolfo Martínez-Usó, and José Hernández-Orallo. 2019. Item response theory in AI: Analysing machine learning classifiers at the instance level. *Artificial intelligence* 271 (2019), 18–42.

[23] Fernando Martínez-Plumed, David Castellano, Carlos Monserrat-Aranda, and José Hernández-Orallo. 2022. When AI Difficulty Is Easy: The Explanatory Power of Predicting IRT Difficulty. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 7 (2022), 7719–7727.

[24] João V.C. Moraes, Jéssica T.S. Reinaldo, Manuel Ferreira-Junior, Telmo Silva Filho, and Ricardo B.C. Prudêncio. 2022. Evaluating regression algorithms at the instance level using item response theory. *Knowledge-Based Systems* 240 (2022), 108076. https://doi.org/10.1016/j.knosys.2021.108076

[25] Mario A Muñoz, Laura Villanova, Davaatseren Baatar, and Kate Smith-Miles. 2018. Instance spaces for machine learning classification. *Machine Learning* 107, 1 (2018), 109–147.

[26] Gustavo H Nunes, Gustavo O Martins, Carlos HQ Forster, and Ana C Lorena. 2021. Using instance hardness measures in curriculum learning. In *Anais do XVIII Encontro Nacional de Inteligência Artificial e Computacional*. SBC, 177–188.

[27] Pedro Yuri Arbs Paiva, Camila Castro Moreno, Kate Smith-Miles, Maria Gabriela Valeriano, and Ana Carolina Lorena. 2022. Relating instance hardness to classification performance in a dataset: a visual approach. *Machine Learning* 111, 8 (2022), 3085–3123.

[28] Pedro Yuri Arbs Paiva, Kate Smith-Miles, Maria Gabriela Valeriano, and Ana Carolina Lorena. 2021. PyHard: a novel tool for generating hardness embeddings to support data-centric analysis. *arXiv preprint arXiv:2109.14430* (2021).

[29] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[30] J. Pimentel, P. J. Azevedo, and L. Torgo. 2022. Subgroup mining for performance analysis of regression models (to appear). *Expert Systems* (2022).

[31] Ricardo B. C. Prudêncio. 2020. Cost Sensitive Evaluation of Instance Hardness in Machine Learning. In *Machine Learning and Knowledge Discovery in Databases*, Ulf Brefeld, Elisa Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet (Eds.). Springer International Publishing, 86–102.

[32] R. B. C. Prudêncio and T. M Silva-Filho. 2022. Explaining Learning Performance with Local Performance Regions and Maximally Relevant Meta-Rules. In *Brazilian Conference on Intelligent Systems (to appear)*.

[33] J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning* 1, 1 (1986), 81–106.

[34] HANA Řezanková. 2018. Different approaches to the silhouette coefficient calculation in cluster evaluation. In *21st International Scientific Conference AMSE Applications of Mathematics and Statistics in Economics*. 1–10.

[35] José A Sáez, Mikel Galar, Julián Luengo, and Francisco Herrera. 2014. Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. *Knowledge and information systems* 38, 1 (2014), 179–206.

[36] Habiba Muhammad Sani, Ci Lei, and Daniel Neagu. 2018. Computational complexity analysis of decision tree algorithms. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 191–197.

[37] Miriam Seoane Santos, Pedro Henriques Abreu, Nathalie Japkowicz, Alberto Fernández, and João Santos. 2023. A unifying view of class overlap and imbalance: Key concepts, multi-view panorama, and open avenues for research. *Information Fusion* 89 (2023), 228–253.

[38] Peter Schulam and Suchi Saria. 2019. Can You Trust This Prediction? Auditing Pointwise Reliability After Learning. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan (Proceedings of Machine Learning Research, Vol. 89)*, Kamalika Chaudhuri and Masashi Sugiyama (Eds.). PMLR, 1022–1031.

[39] Michael R Smith, Tony Martinez, and Christophe Giraud-Carrier. 2014. An instance level analysis of data complexity. *Machine learning* 95, 2 (2014), 225–256.

[40] Kate A Smith-Miles. 2009. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)* 41, 1 (2009), 1–25.

[41] Mariana A Souza, George DC Cavalcanti, Rafael MO Cruz, and Robert Sabourin. 2019. Online local pool generation for dynamic classifier selection. *Pattern Recognition* 85 (2019), 132–148.

[42] Mariana A Souza, Robert Sabourin, George DC Cavalcanti, and Rafael MO Cruz. 2023. OLP++: An online local classifier for high dimensional data. *Information Fusion* 90 (2023), 120–137.

[43] Victor LF Souza, Adriano LI Oliveira, Rafael MO Cruz, and Robert Sabourin. 2020. A white-box analysis on the writer-independent dichotomy transformation applied to offline handwritten signature verification. *Expert Systems with Applications* 154 (2020), 113397.

[44] Ingo Steinwart and Andreas Christmann. 2008. *Support vector machines*. Springer Science & Business Media.

[45] Joaquin Vanschoren. 2019. Meta-learning. In *Automated machine learning*. Springer, Cham, 35–61.

[46] Halbert White et al. 1992. *Artificial neural networks*. Blackwell Cambridge, Mass.

[47] Jie Xie, Mingying Zhu, Kai Hu, and Jinglan Zhang. 2023. Instance hardness and multivariate Gaussian distribution-based oversampling technique for imbalance classification. *Pattern Analysis and Applications* (2023), 1–15.

[48] Shen Yan, Hsien-Te Kao, and Emilio Ferrara. 2020. Fair Class Balancing: Enhancing Model Fairness without Observing Sensitive Attributes. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (2020).

[49] Harry Zhang. 2004. The optimality of naive Bayes. *AAAI* 1, 2 (2004), 3.