Analyzing datasets under Instance Space Analysis framework: extension and exploration of hardness embeddings

Diogo B. Rodrigues[1], **Lucas R. R. Barros**[1],
Alfredo A. A. E. de Queiroz[1], **Ana C. Lorena**[1]

[1]Divisão de Ciência da Computação - Instituto Tecnológico de Aeronáutica (ITA)
São José dos Campos – SP – Brazil

{diogo.rodrigues.8911,lucas.barros.8924}@ga.ita.br

{alfredoqueiroz,aclorena}@ita.br

Abstract. *In Machine Learning (ML), no single algorithm consistently outperforms others across all datasets. Meta-Learning aims to address the challenge of choosing suitable ML algorithms for new tasks by leveraging relationships between dataset characteristics and algorithm performance. One framework that supports this process is Instance Space Analysis (ISA), which enables the visualization of algorithm performance and instance hardness in a two-dimensional space. This study expands the ISA framework for analyzing classification datasets and ML algorithms, making it more complete and enabling meaningful insights into the relationships between dataset characteristics and ML classification performance.*

1. Introduction

Given a new classification dataset, many Machine Learning (ML) techniques exist to solve the problem. This can be a challenging task for users, who might fail to explore the full potential of these techniques. In addition, it is well known that there is no universally superior ML algorithm for every type of problem [Wolpert 2002]. Considering this, the field of Meta-Learning (MtL) offers an alternative approach to address the Algorithm Selection Problem (ASP) [Rice 1976, Smith-Miles 2009, Brazdil et al. 2022].

MtL focuses on leveraging knowledge gained from prior tasks to enhance the performance of ML models on new tasks [Vanschoren 2018]. By exploiting meta-knowledge, such as prior algorithm performance and task characteristics, MtL systems can make informed decisions about model selection and hyperparameter optimization, accelerating the learning process. MtL systems tailor the choice to the specific characteristics of each task, maximizing performance by selecting the most appropriate algorithms and configurations.

Expanding upon the concepts of MtL, Instance Space Analysis (ISA) introduces a sophisticated methodology for algorithm testing that leverages these principles to offer insights into algorithm performance in a fine-grained way [Smith-Miles and Muñoz 2023]. ISA utilizes meta-data to analyze and interpret the relationships between instance characteristics and algorithm efficacy, creating a bi-dimensional instance space where the performance of various algorithms can be visualized and scrutinized. ISA enables a more nuanced understanding of where algorithms excel or struggle, going beyond simplistic average performance metrics. This approach helps recommend the most suitable algorithms for specific regions within the instance space and supports the rigorous evaluation of algorithm strengths and weaknesses [Smith-Miles and Muñoz 2023].

Recently, the ISA framework has been cast to produce a 2-D hardness embedding of a dataset, allowing fine-grained insights on data quality and ML algorithmic performance within a classification dataset [Paiva et al. 2022]. The work has culminated in implementing a Python library named PyHard [Paiva et al. 2021][1], which embeds the ISA framework. PyHard uses several components of the original ISA framework, which were implemented in Python in the PyISpace library[2]. Nonetheless, not all modules of the original ISA proposal were included. Here, we include all modules and show how the complete ISA framework can be applied in the analysis of datasets in ML to obtain meaningful insights and interpretations concerning classification performance. To increase interpretability within the ISA framework, this work also integrates LIME[3] (Local Interpretable Model-agnostic Explanations) [Ribeiro et al. 2016] to explain individual instance projections.

This paper is organized as follows: Section 2 provides a background on ISA and how it can be framed for analyzing classification datasets in ML. Section 3 presents the usage of the extended ISA tool on datasets and the type of knowledge that can be extracted from such analysis. Section 4 concludes this paper.

2. Background and methodology

This section describes the general ISA framework, along with the adaptation for analyzing datasets in ML.

2.1. The ISA framework

The Instance Space Analysis (ISA) framework provides a methodology for objectively testing algorithms. The ISA constructs a 2-D instance space that visualizes all possible test instances, revealing insightful relationships between instance features and algorithm performance. This approach aims to offer a more nuanced understanding of an algorithm's unique strengths and weaknesses across different regions of the instance space, which might be hidden by average performance metrics. ISA also facilitates an objective assessment of any bias in chosen test instances and guides the adequacy of benchmark test suites. By integrating these advanced concepts, ISA represents a robust and sophisticated extension of MtL, capable of improving the way algorithms are selected and tested.

The core of the ISA methodology involves the following key steps [Smith-Miles and Muñoz 2023]:

- **Meta-data Collection**: Gathering experimental data for a set of instances I run on a portfolio of algorithms A, including meta-feature values F for all instances and performance metrics Y for all algorithms on all instances.
- **Instance Space Construction**: Utilizing a feature selection and reduction process on the collected meta-data to define a 2-D instance space, including its theoretical

[1]https://pypi.org/project/pyhard/
[2]https://pypi.org/project/pyispace/
[3]https://github.com/marcotcr/lime

boundary. This step involves modules named PRELIM (Preparation for Learning of Instance Meta-data), SIFTED (Selection of Instance Features to Explain Difficulty), PILOT (Projecting Instances with Linearly Observable Trends), and CLOISTER (Correlated Limits of the Instance Space's Theoretical or Experimental Regions).

- **Automated Algorithm Selection**: Generating machine learning predictions, often using the PYTHIA (Performance prediction and automated algorithm selection) module, to recommend algorithms better suited for each instance in $\mathcal{I}$.
- **Algorithm Footprint Generation**: Creating and analyzing algorithm footprints, which are the regions in the instance space where an algorithm is predicted to exhibit good performance, in a module called TRACE (Triangulation with Removal of Areas with Contradicting Evidence). These footprints are characterized by properties such as location, area (indicating robustness), density (reflecting the strength of evidence), and purity (indicating the presence or absence of conflicting evidence).
- **Instance Space Analysis**: Interpreting the visualizations and metrics to gain insights into algorithm behavior and instance characteristics.
- **Iterative Refinement**: Generating additional meta-data if required to improve insights and repeating the process.

The ISA framework is general and can be applied to any problem where one needs to stress-test different algorithms and understand their behaviors. In the literature, it has been applied to many problems and areas, such as optimization [Liu et al. 2024, Katial et al. 2025], ML [Muñoz et al. 2018, Muñoz et al. 2021], and automated software testing [Neelofar et al. 2023].

2.2. ISA for generating a hardness embedding of a dataset

The ISA framework has been used in the literature for analyzing benchmarks containing a pool of datasets [Muñoz et al. 2018, Muñoz et al. 2021]. In [Paiva et al. 2022], the ISA framework was reframed so that one dataset can be individually analyzed. This builds on the fact that classification performance is non-uniform across a dataset. Specifically, noisy and borderline instances are often harder to classify than instances well situated within their classes. Although the common practice in ML is to average the performance across all observations in a dataset, this may hinder specific subpopulations where classification performance is poor and needs to be scrutinized.

Figure 1 casts ISA for examining the individual observations of a dataset. It departs from a dataset D with n pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is an observation and y_i corresponds to its class. These observations are described by a set of instance hardness measures (meta-features) F, which indicate how difficult it is to predict y_i given $\mathbf{x}_i$ from different perspectives. Meanwhile, a pool of algorithms A (algorithm space) is run in a cross-validation setup and has the log-loss performance recorded for each observation in the set Y (performance space). A feature selection step selects the meta-features F that are more descriptive of classification performance. Given the selected set of features $\tilde{F}$ and the performance values Y, a metadataset M is composed. This metadataset is then input to the ISA framework, where the PILOT module builds the 2-D projection or Instance Space (IS). This corresponds to the hardness embedding, where each observation is placed presenting linear trends in classification difficulty. Specifically, instances that are harder to classify are placed in the upper left corner of the IS.

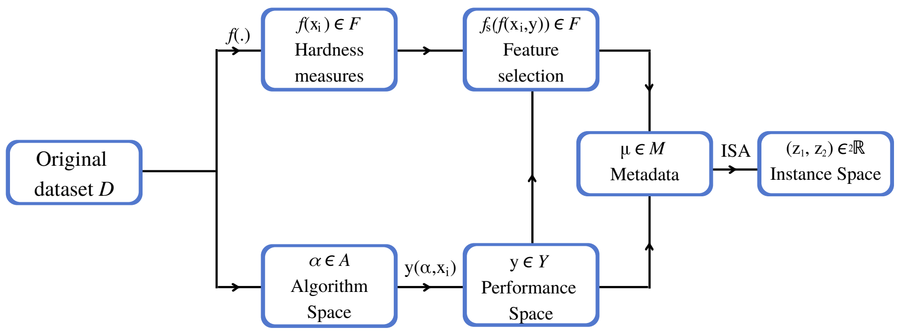


Figure 1. ISA for obtaining hardness embedding of a dataset [Paiva et al. 2022].

The PyHard tool implements these steps [Paiva et al. 2021], along with a visualization app that allows inspecting the hardness embedding and selecting specific areas to be inspected. PyHard calls another Python package named PyISpace, which implements the main components of the original ISA framework (PRELIM, SIFTED, PILOT, and TRACE). This paper adds the CLOISTER and PYTHIA modules to the packages, as described next.

2.2.1. CLOISTER Module

The CLOISTER (Correlated Limits of the Instance Space's Theoretical or Experimental Regions) module approximates the boundary of the instance space. The objective is to obtain a feasible region where instances can lie in the hardness embedding. CLOISTER achieves this by using the experimental or theoretical upper and lower bounds of the selected instance (meta-)features while accounting for correlations between these features. Visualizing this boundary is crucial for assessing the diversity of the existing instances concerning all theoretically possible instances. CLOISTER takes as input the matrix of selected meta-features ($\tilde{F}$), a minimum correlation threshold (ϵ) to consider features colinear, and a p-value (p) to determine if features are uncorrelated. Then it executes the following steps [Smith-Miles and Muñoz 2023]:

1. **Feature Bounds**: It first determines the upper (f_U) and lower (f_L) bounds for each feature in $\tilde{F}$. These bounds can be derived from the observed experimental metadata or theoretical knowledge about the features.
2. **Hyper-cube Generation**: A hyper-cube is generated, encompassing all possible vertex vectors. Each vertex is a combination of the upper and lower bounds of the features.
3. **Pruning Unlikely Vertices**: The module then prunes vertex vectors that represent unlikely combinations of feature values due to strong correlations.
4. **Projection to 2-D Space**: The remaining, more plausible vertex vectors are projected into the 2D instance space using the linear transformation matrix derived from the PILOT method.
5. **Boundary Definition**: The convex hull of these projected points forms the mathematical boundary of the instance space.

The CLOISTER-defined boundary provides a critical reference for understanding the coverage of the instances in the instance space, highlighting regions within the theoretical instance space where more data might be needed or where no instances can theoretically exist. When analyzing a dataset, one may identify regions that might be interesting to explore and eventually generate new observations to cover them.

2.3. PYTHIA Module

The PYTHIA (Performance prediction and automated algorithm selection) module solves an algorithm selection problem using MtL-based predictive modeling. It leverages historical algorithm performance data and meta-feature representations to recommend suitable algorithms for each instance. When applied in a dataset, it can help identify which algorithms perform better for specific dataset observations.

PYTHIA receives as input the metadata M and performs the following steps [Smith-Miles and Muñoz 2023]:

1. **Model Training**: For each algorithm in the portfolio A, PYTHIA trains a binary classification model to learn whether the algorithm is expected to perform satisfactorily (above a set threshold on Y) on a given instance, using cross-validation.
2. **Prediction and Ranking**: After training, each model outputs probability scores indicating the likelihood of satisfactory algorithm performance on a new instance. These probabilities are used to compute a ranking of all algorithms for each instance, based on their predicted success probabilities.

Therefore, PYTHIA allows to obtain ranked algorithm recommendations per instance in I. In this work, we have also integrated an explainability module to PYTHIA. This is done by explaining individual predictions with LIME (Local Interpretable Model-agnostic Explanations) [Ribeiro et al. 2016].

LIME is applied after the 2-D projection of meta-features produced by PILOT. It perturbs the original meta-features around a target instance to generate synthetic neighbors. The resulting 2-D coordinates are observed, and a sparse linear regression is fitted to approximate the projection behavior locally. Herewith, LIME highlights which meta-features most influenced the position of the instance in the space, explaining why it is considered easy or hard to predict.

3. Case Studies

This section presents three case studies demonstrating the potential of the extensions to the PyHard tool in analyzing datasets. For building the IS, the following sets were considered:

- A: Bagging, Gradient Boosting, Support Vector Machine (both linear and RBF kernels), Logistic Regression, Multilayer Perceptron, and Random Forest.
- F: k-disagreeing neighbors (kDN), Disjunct Class Percentage (DCP), Pruned Tree Depth (TD_P), Class Likelihood (CL), Fraction of nearby instances of different class (N1), Ratio of intra-extra class distances (N2).

Descriptions of the F measures can be found in [Lorena et al. 2024]. They all vary between 0 and 1, and higher values indicate a higher hardness level.

3.1. Case Study: IS boundaries in COVID-19 prognosis dataset

This section extends the case study on the COVID-19 prognosis dataset originally introduced in [Paiva et al. 2021], now focusing on the additional capabilities provided by the newly integrated CLOISTER module. This dataset contains anonymized data from citizens positively diagnosed for COVID-19 in São José dos Campos-SP from March 1st, 2020 to April 15th, 2021. It involves predicting whether an individual will require future hospitalization, taking as input attributes routinely supplied during COVID scanning (age, sex, symptoms, and comorbidities). The dataset has data from 5,156 citizens, half of whom were hospitalized.

Figure 2 presents the IS (hardness embedding) of the hospitalization dataset, where each point corresponds to a confirmed case of COVID-19. Blue points correspond to patients who were not hospitalized, while red points indicate those who were hospitalized. In the PyHard implementation, hard instances are placed towards the upper left corner of the IS, and easy instances are in the bottom right region. We have, therefore, a group of easy hospitalized patients at the bottom and some hard hospitalized patients at the top. The non-hospitalized patients have an intermediate hardness level.

Figure 3 also displays the Instance Space, but the points are colored according to their instance hardness (IH): blue indicates easier instances, and red indicates harder ones. Overlaid on the IH plot are two boundaries generated by the CLOISTER module. The blue boundary represents the standard Convex Hull, which includes all feature combinations, while the red boundary represents the Restricted Convex Hull, which prunes unlikely feature combinations based on a 90% correlation threshold.

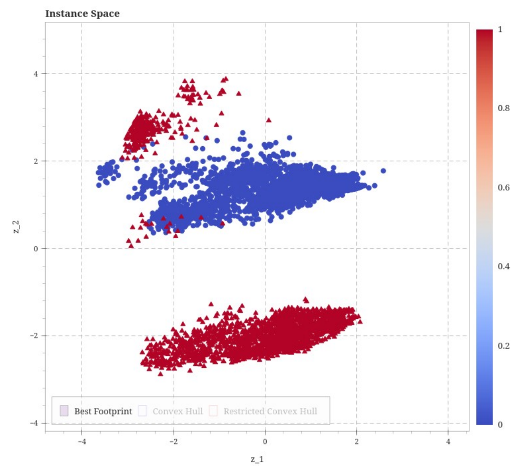


Figure 2. ISA of hospitalization dataset: classes.

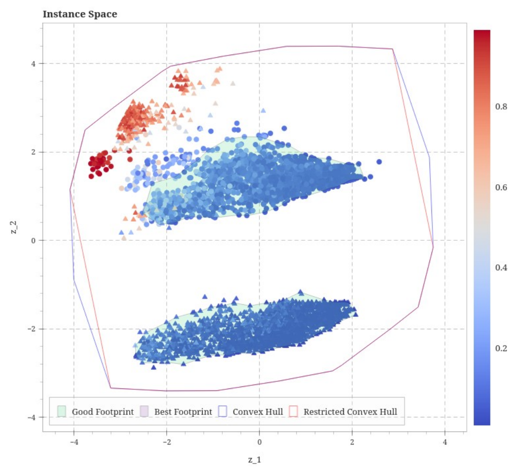


Figure 3. ISA of hospitalization dataset: IH 0.90 in CLOISTER.

The analysis of Figure 3 provides insights into the representativeness of the dataset. Notably, there is a visible absence of points in the central and top-right regions of the Instance Space. Although these regions are statistically plausible, the lack of observations may indicate under-representativeness or dataset bias. Furthermore, most of the harder instances are concentrated near the edges of the space, suggesting that these are more extreme or atypical cases. In fact, the instances in the left-most position were identified as incorrectly labeled in [Paiva et al. 2021], which can justify this behavior.

3.2. Case Study: Algorithm prediction and local interpretability

The PYTHIA module was applied to predict the most suitable algorithm for each instance in the COVID-19 prognosis dataset. For generating the meta-models for algorithm recommendation, five classification algorithms were considered: Support Vector Machine (RBF and Polynomial kernel), k-Nearest Neighbors, Random Forest, and XGBoost. All models were trained using Bayesian optimization with 5-fold cross-validation, and training was parallelized to reduce computational time. The mean accuracy obtained from the meta-training of all meta-models is presented in Table 1. The best accuracy in recommendation was attained by the Random Forest model, highlighted in bold.

Table 1. Mean accuracy of each meta-model for algorithm recommendation.

Meta-model	Random Forest	KNN	XGBoost	SVM	SVM Poly
Mean Accuracy	**0.9879**	0.9874	0.9869	0.9867	0.9865

After training, PYTHIA generated probability estimates for each instance and algorithm. The final rankings were derived based on the log loss values computed per instance and algorithm. Algorithms were ordered according to their log loss, where a lower log loss indicates a better fit for the instance. This approach emphasizes the accuracy and calibration of the probability estimates, ensuring that the selected algorithm minimizes prediction uncertainty for each instance.

The decisions of the Random Forest meta-model, which attained the highest accuracy, are shown in the plot of Figure 4. It is notable that in the regions considered easier (central and south-central parts of the instance space), the meta-model consistently predicted the Gaussian-kernel SVM and linear SVM as the best-performing algorithm. In contrast, the top-left region of the instance space, which is characterized by higher difficulty, exhibits greater variation in the algorithms predicted as optimal. This variability reflects hard instances' inherent complexity and instability, which are more susceptible to model uncertainty, outliers, and decision boundary noise.

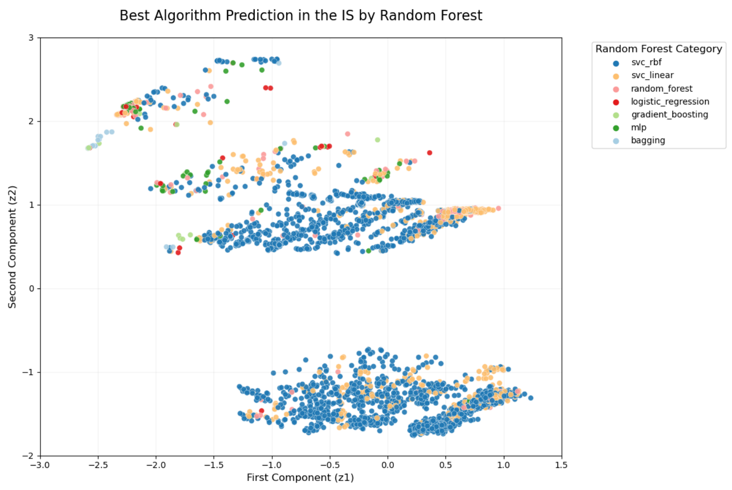
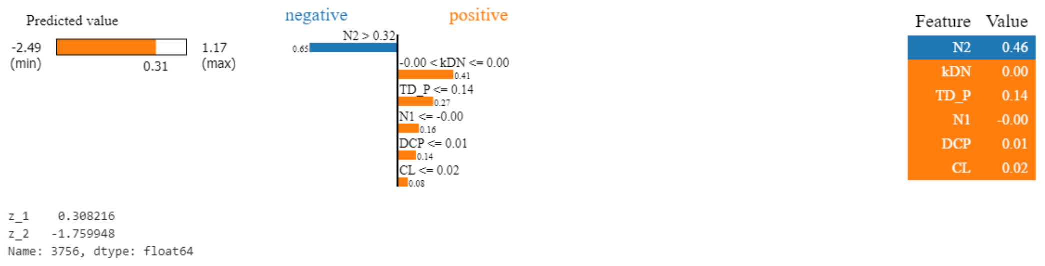


Figure 4. Best algorithm predicted for each instance by RF meta-model.

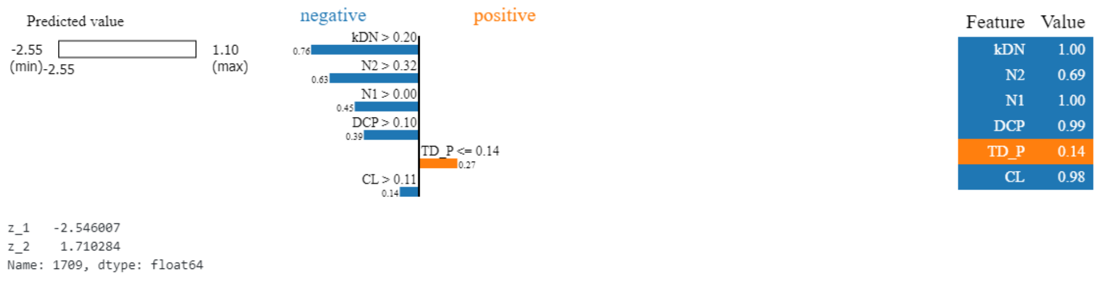
To complement the prediction, LIME was used to explain the positioning of specific instances in 2-D space. After perturbing an instance's meta-features, LIME approximates the behavior of the PILOT projection using a sparse linear regression. This allows for identifying the meta-features that most influenced the instance's coordinates and its estimated difficulty. Figures 5 and 6 illustrate local explanations for an easy and a hard instance, respectively. These results help understand which dataset characteristics contributed to their placement in the instance space and support the interpretability of algorithm recommendations made by PYTHIA.



```
z_1     0.308216
z_2    -1.759948
Name: 3756, dtype: float64
```

Feature	Value
N2	0.46
kDN	0.00
TD_P	0.14
N1	-0.00
DCP	0.01
CL	0.02

Figure 5. LIME explanation for an easy instance.



```
z_1    -2.546007
z_2     1.710284
Name: 1709, dtype: float64
```

Feature	Value
kDN	1.00
N2	0.69
N1	1.00
DCP	0.99
TD_P	0.14
CL	0.98

Figure 6. LIME explanation for a hard instance.

In Figure 5 (easy instance), the LIME explanation shows all meta-feature values at the low end: kDN (0.00), N1 (0.00), DCP (0.01), and CL (0.02), indicating it is near from instances from its class (kDN and N1), has a high likelihood of belonging to its class (CL) and is placed in a uniform disjunct (DCP). Therefore, it has feature values that align with its class. The predicted value is -2.49 with high confidence, driven primarily by N2 (0.46), which strongly supports the negative classification.

In comparison, Figure 6 (hard instance) displays high values for kDN (1.00), N1 (1.00), DCP (0.99), and CL (0.98). This means the instance is probably noisy and has features that align more with the opposite class. Multiple features influence the predicted value of -2.55, notably kDN and N2, but the contribution is more evenly distributed.

3.3. Case Study: ISA for a perturbed dataset

This section analyzes ISA results on datasets affected by label noise. Specifically, we perturb the Eye Movements dataset[4], originally developed for the "Inferring Relevance from Eye Movements" challenge [Salojärvi et al. 2005], particularly chosen for providing a compelling testbed for analyzing the robustness and sensitivity of algorithms under

[4]https://www.openml.org/search?type=data&sort=runs&id=1044&status=active

the ISA framework. This dataset contains low-level and aggregated features extracted from eye-tracking data, such as fixation durations, saccade amplitudes, and pupil dilation statistics, collected while subjects read question and answer documents. The target variable is trinary, indicating whether a document was relevant to the reader (1), not relevant to the reader (0), or correct (2).

We introduce two distinct types of noise hardness in this dataset, as defined in [Seedat et al. 2024]: uniform and asymmetric. Uniform hardness refers to randomly mislabeling the target variable, introducing noise uniformly across all samples. Asymmetric hardness, in contrast, reflects label-dependent mislabeling, where certain classes are more likely to be mislabeled with specific others. These types of errors are more realistic and commonly found in practical settings.

The original ISA dataset is illustrated in Figures 7 and 8, where the data points are colored according to the target label class and instance hardness, respectively. The dataset contains three hardness regions: easy instances at the bottom-right, which tended to concentrate elements from classes 0 (blue) and 2 (red); instances of intermediate hardness level in the middle, where most of the instances of class 1 (grey); and hard instances in the top-left, also concentrating elements from classes 0 and 2. Notably, class 2 has both the easiest and hardest instances to classify.

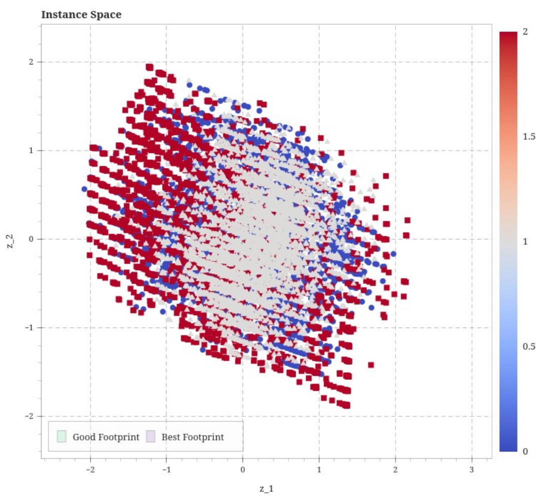


Figure 7. ISA of eye movement dataset: classes.

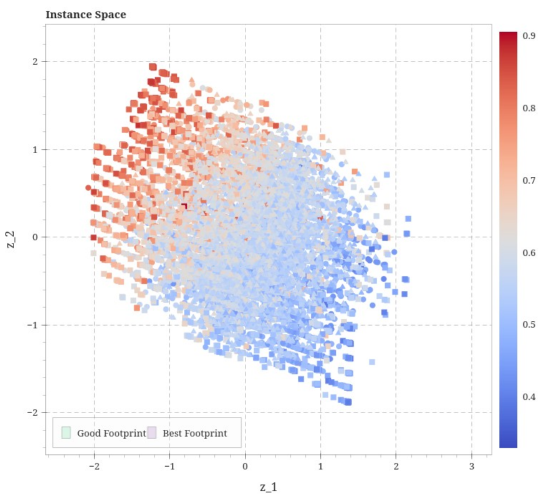


Figure 8. ISA of eye movements dataset: IH.

Starting with the original eye movement dataset, the hardness noise was sequentially applied to 10%, 15%, and 20% of the data points. For each level of perturbation, the modified dataset was analyzed using the ISA framework, and the normalized areas of the good footprint (area where the algorithm performs well) were calculated for each of the seven classification techniques implemented in PyHard. The results are presented in Figure 9, and can be compared with the original footprint values shown in each image with a dashed line, obtained for the non-perturbed dataset. The SVM classifier with the RBF Kernel was the best performing algorithm according to its area, followed by Gradient Boosting. In contrast, the MLP algorithm had the lowest footprint area, followed by Logistic Regression.

An observation from Figure 9 is that the footprint's areas for the evaluated algo-

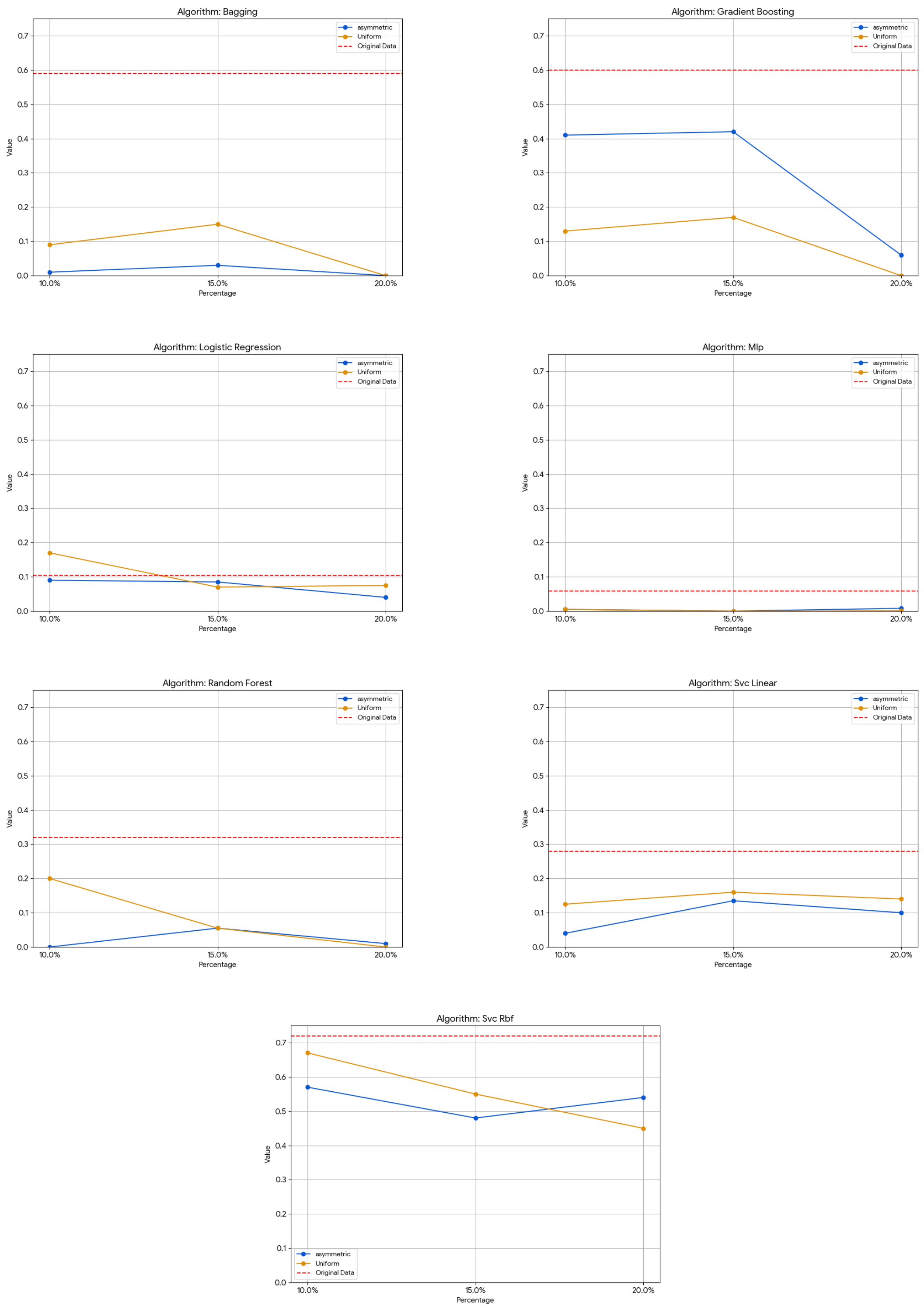


Figure 9. Classifiers footprint area for different hardness and percentages.

rithms generally decrease when embedded with uniform noise (orange line), indicating a harmful effect of that type of noise on classification performance. Additionally, the inclusion of asymmetric hardness (blue line) generally decreases the area for the evaluated algorithms more than when the data is embedded with uniform noise, suggesting that this form of noise poses a greater challenge to the classifiers. This is expected, as the asymmetric noise is more realistic in flipping the labels. But there are specificities, such as Gradient Boosting, which tend to be more robust against asymmetric noise.

The ranking of algorithms per hardness type changes slightly for intermediate performance classifiers, but the best and worst classification algorithms remain the same in the plots.

All experiments were conducted on a Dell G15 5520 equipped with an Intel Core i7 processor and 16GB of RAM. Each perturbation and corresponding footprint analysis took approximately 30 minutes to complete, resulting in a total runtime of about 3 hours. Although the Eye Movement dataset used contains around 11,000 instances and 28 features—considered relatively small in size—this runtime is expected to increase for larger or more complex datasets.

4. Conclusion

This study significantly enhances the PyHard toolkit by fully integrating the CLOISTER and PYTHIA modules into the Instance Space Analysis (ISA) framework. These additions expand PyHard's capability for analyzing classification datasets, providing a comprehensive and interpretable hardness embedding that supports both global and local interpretations.

The CLOISTER module enables the definition of realistic boundaries in the instance space by pruning implausible meta-feature combinations, improving the space's representativeness and coverage. The PYTHIA module complements this with automated algorithm recommendations tailored to each instance, backed by interpretable model predictions and performance-based rankings.

Through case studies, we demonstrated how ISA provides coarse and fine-grained insights into algorithm performance and instance difficulty. The integration of local explanations highlights how meta-features such as kDN, N1, DCP, and CL contribute to the complexity of specific instances, validating the effectiveness of ISA in pinpointing regions of algorithm strength or uncertainty.

Future work can explore the tool's functionalities in more datasets to give insights towards specific instances to be examined and the strengths and weaknesses of the ML algorithms. Other interesting approach is adding the generation of new instances in specific regions of interest in the IS.

Acknowledgements

This study was financed, in part, by the São Paulo Research Foundation (FAPESP), Brasil. Process Number #2024/07655-7, #2024/07637-9, #2024/16535-5, #2024/15817-7, #2024/09091-3 and #2021/06870-3.

References

Brazdil, P., Van Rijn, J. N., Soares, C., and Vanschoren, J. (2022). *Metalearning: applications to automated machine learning and data mining*. Springer Nature.

Katial, V., Smith-Miles, K., Hill, C., and Hollenberg, L. (2025). On the instance dependence of parameter initialization for the quantum approximate optimization algorithm: Insights via instance space analysis. *INFORMS Journal on Computing*, 37(1):146–171.

Liu, C., Smith-Miles, K., Wauters, T., and Costa, A. M. (2024). Instance space analysis for 2d bin packing mathematical models. *European Journal of Operational Research*, 315(2):484–498.

Lorena, A. C., Paiva, P. Y., and Prudêncio, R. B. (2024). Trusting my predictions: on the value of instance-level analysis. *ACM Computing Surveys*, 56(7):1–28.

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

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

Neelofar, N., Smith-Miles, K., Muñoz, M. A., and Aleti, A. (2023). Instance space analysis of search-based software testing. *IEEE Transactions on Software Engineering*, 49(4):2642–2660.

Paiva, P. Y. A., Moreno, C. C., Smith-Miles, K., Valeriano, M. G., and Lorena, A. C. (2022). Relating instance hardness to classification performance in a dataset: a visual approach. *Machine Learning*, 111(8):3085–3123.

Paiva, P. Y. A., Smith-Miles, K., Valeriano, M. G., and Lorena, A. C. (2021). Pyhard: a novel tool for generating hardness embeddings to support data-centric analysis. *arXiv preprint arXiv:2109.14430*.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144.

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

Salojärvi, J., Puolamäki, K., Simola, J., Kovanen, L., Kojo, I., and Kaski, S. (2005). Inferring relevance from eye movements: Feature extraction. Publications in Computer and Information Science, Report A82, Helsinki University of Technology, Helsinki, Finland.

Seedat, N., Imrie, F., and van der Schaar, M. (2024). Dissecting sample hardness: Fine-grained analysis of hardness characterization methods. In *The Twelfth International Conference on Learning Representations*.

Smith-Miles, K. and Muñoz, M. A. (2023). Instance space analysis for algorithmnbsp;testing: Methodology and software tools. *ACM Comput. Surv.*, 55(12).

Smith-Miles, K. A. (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.*, 41(1).

Vanschoren, J. (2018). Meta-learning: A survey. *CoRR*, abs/1810.03548.

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