Collecting Meta-Data from the OpenML Public Repository

Nathan F. Carvalho[1], **André A. Gonçalves**[1], **Ana C. Lorena**[1]

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

{nathan.carvalho.8849,andre.goncalves.8750}@ga.ita.br, aclorena@ita.br

Abstract. *In Machine Learning (ML), selecting the most suitable algorithm for a problem is a challenge. Meta-Learning (MtL) offers an alternative approach by exploring the relationships between dataset characteristics and ML algorithmic performance. To conduct a MtL study, it is necessary to create a meta-dataset comprising datasets of varying characteristics and defying the ML algorithms at different levels. This study analyzes the information available in the OpenML public repository for building such meta-datasets, which provides a Python API for easy data importation. Assessing the content currently available in the platform, there is still no extensive meta-feature characterization for all datasets, limiting their complete characterization.*

1. Introduction

In Machine Learning (ML), it is common practice to investigate and choose which algorithm is best suited to solve a particular problem. Common approaches include seeking advice from domain specialists or experienced data scientists [Zöller and Huber 2021] or employing significant computational power to explore various possibilities. According to [Wolpert 2002], there is no universally superior ML algorithm for every type of problem. Taking this into consideration, the field of Meta-Learning (MtL) offers an alternative approach to address the algorithm selection problem [Rice 1976, Smith-Miles 2009]. MtL explores the relationships between the characteristics of the datasets and the performance of ML algorithms when run on these datasets [Aha 1992]. This approach aims to formulate a more informed response regarding the most suitable algorithm for a given dataset.

To conduct a MtL study, it is necessary to create a meta-dataset, which must be composed of measures revealing characteristics from a pool of datasets (named meta-features [Rivolli et al. 2022]), as well as measures of the performance of a pool of ML algorithms run on the datasets [Smith-Miles 2009]. This can be achieved by compiling a collection of datasets and running different ML algorithms locally. This task can be computationally demanding, and, for this reason, various companies, academic institutions, and individuals have developed solutions to facilitate dataset research, leading to the establishment of public platforms known as public dataset repositories [Noy et al. 2019], which allow for the publication and sharing of datasets.

Several public dataset repositories are accessible for research purposes, including the UC Irvine Machine Learning Repository [Newman et al. 1998], Keel [Alcalá-Fdez et al. 2011], Kaggle[1], and OpenML [Vanschoren et al. 2013] repositories. In the context of this study, particular attention is given to the OpenML public repository. The OpenML repository holds special significance as it provides a Python API

[1]https://www.kaggle.com/datasets

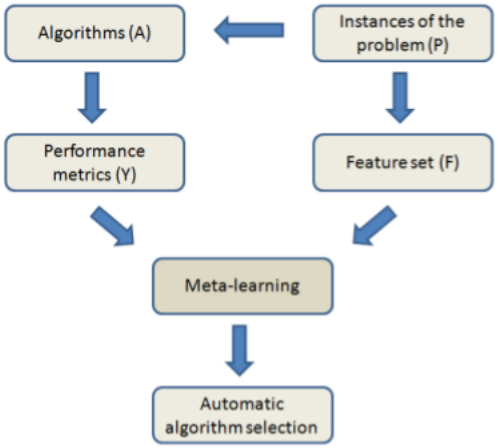

Figure 1. ASP framework from Rice [Rice 1976, Smith-Miles 2009].

[Feurer et al. 2019] that enables straightforward importation of datasets available on the platform, along with some meta-features values and performance data for some standard ML algorithms when run on each dataset.

OpenML provides a vast repository of datasets and ML algorithm performance data, containing a collection of over 5,400 datasets and 20,000,000 runs of ML algorithms, as reported on the website `https://www.openml.org/`. Within the scope of MtL, it is possible to compile a meta-dataset using information available from the platform, aligning with the specifications outlined by [Smith-Miles 2009], which are:

1. Availability of large collections of problem instances of various complexities, which in ML represents diverse datasets challenging the ML techniques at different levels;
2. Existence of a large number of diverse algorithms for tackling problem instances, meaning having a pool of ML algorithms of different biases run on the datasets;
3. Performance metrics to evaluate algorithm performance, which can be the accuracy, AUC (Area Under the ROC Curve), F1-score, among others, in the case of classification problems;
4. Existence of suitable features to characterise the properties of the instances, that is, meta-features which reveal the main characteristics of the datasets.

At present, several studies have utilized OpenML to generate meta-datasets for MtL purposes. For instance, [Bilalli et al. 2017] conducted an analysis of the meta-features available on the platform, concluding that different classification algorithms required different sets of meta-features. [Muñoz et al. 2018] focused on creating an Instance Space Analysis tool, which plots the datasets as points in a 2D space where linear relationships of hardness levels are contrasted, while Bischl et al. [Bischl et al. 2021] employed OpenML for the selection of benchmark suites. Other related studies are [Post et al. 2016], which used OpenML datasets to study the influence of feature selection on the performance of classification algorithms and [Kühn et al. 2018], that used a tool called *OpenML Random Bot* to study the influence of hyperparameters on the performance of ML algorithms.

The main contributions of this work are: (1) characterization and analysis of meta-data available on the OpenML public repository, as a widely used platform for MtL studies, since currently there is a lack of a comprehensive formal characterization of the meta-information associated with the datasets and their corresponding runs within the OpenML repository, including factors such as the overall distribution of the datasets according to their characteristics and ML algorithmmic performances; (2) determine if the platform's existing meta-features provide sufficient diversity for standard MtL practices, as highlighted by [Smith-Miles 2009] and include all meta-features used in recent studies, as pointed by [Alcobaça et al. 2020]; (3) Create a meta-dataset useful for up to date MtL research, considering that most MtL studies typically utilize a relatively small meta-dataset [Bilalli et al. 2017], this work also explores the viability of constructing a larger meta-dataset by leveraging the available information from the platform.

The remainder of the paper is organized as follows. Section 2 introduces the main concepts behind formulating a meta-dataset and navigating the OpenML platform, as well as presents the main tools utilized on assessing data on OpenML. The results of data distribution and availability in OpenML are highlighted on Section 3 and main conclusions are drawn in Section 4.

2. Background and methodology

This section presents the main materials and methods used in the paper, taking Rice's framework for the Algorithm Selection Problem (ASP) [Rice 1976, Smith-Miles 2009] as a reference, as shown in Figure 1. In this framework, given a set of instances P of the problem, characterized by a set of features F and solved by a pool of candidate algorithms A, whose performances are registered in a set Y, one is able to build a new (meta-)learning problem for automatic algorithm selection. The objective is to learn a mapping from properties of instances of the problem to the performance achieved in their solution, so that suitable solutions can be recommended for novel problems with a given profile. Here we take the set P as a pool of datasets for ML studies.

2.1. Meta-learning

Meta-learning (MtL) is the field of ML that focuses on studying how algorithms perform across a variety of training sets. The objective is to extract meta-knowledge about the relationships of the characteristics of the datasets to the algorithmic performance achieved in their solution, that is, relating the sets F and Y in Figure 1. To conduct such evaluations, a meta-dataset is required [Vanschoren 2018]. To assemble a meta-dataset, it is necessary to collect individual pieces of information about each dataset in P, known as meta-features (composing the set F), along with the predictive results (the set Y) of ML algorithms (the set A).

The current meta-features from the literature can be categorized into the following groups [Rivolli et al. 2022]:

1. **Simple**: These are features extracted directly from the data, without requiring significant computational resources to be computed. Examples include the number of instances, number of classes, and majority class percentage.
2. **Statistical**: These extract statistical properties of the dataset, such as skewness and kurtosis of the input features.

3. **Information theoretic**: These are features extracted based on principles from information theory, such as entropy.
4. **Model-based**: These are measures derived from simple models induced using training data. One example is the size of Decision Tree models induced from the dataset.
5. **Landmarking**: These are the performances of simple ML algorithms, such as K-nearest neighbours and Naïve Bayes.

Several studies have explored the inclusion of complementary meta-features to provide a more complete characterization of properties associated with the performance of specific ML models. For instance, [Lorena et al. 2019] considered classification complexity, where one estimates how difficult it is to solve the classification problem based on properties of the data, while [Song et al. 2012] examined itemset measures. However, the OpenML repository does not currently encompass these specific meta-features, which often have a high computational cost.

2.2. The OpenML public repository

The OpenML repository serves as a place for sharing and organizing datasets, as well as publishing evaluations for some standard ML algorithms [Vanschoren et al. 2013]. OpenML is structurally divided, fundamentally, into four main categories, which can be manipulated either through the website or the REST API [Feurer et al. 2019]:

1. **Datasets:** Contains the datasets, which can be candidates to compose the set P. They are accompanied by a set of qualities which describe them, which are here regarded as candidate meta-features to compose the set F;
2. **Tasks:** Define the problem to be solved involving a particular dataset, so that users can apply ML algorithms. They are:
   (a) *Supervised Classification*.
   (b) *Supervised Regression*.
   (c) *Learning Curve*.
   (d) *Supervised Datastream Classification*.
   (e) *Clustering*.
   (f) *Machine Learning Challenge*.
   (g) *Survival Analysis*.
   (h) *Subgroup Discovery*.
   (i) *Multitask Regression*.
3. **Flows:** Implementations of a ML algorithm in order to solve a specific task. From the existent flows we are able to assemble the set A.
4. **Runs:** Results of a flow applied to a specific task, including the result of the ML algorithm for a variety of evaluation metrics. These information can be used to assemble the set Y.

Through the manipulation of these categories of information, users can select which type of information they want to extract from the platform. In order to assist researches interested in using the content provided by OpenML, a client API for Python is available. It provides access to all datasets currently available in the platform, as well as all tasks, evaluations and flows that have been previously submitted [Feurer et al. 2019]. On top of that, through a scikit-learn extension, it allows users to run tasks locally and

upload the results. In this study, the Python API was extensively used in order to access the tasks related to classification problems, as well as analyse all ML models evaluations accessible in OpenML at the moment.

3. Results and Discussion

Since a meta-dataset suitable for MtL purposes must comprise meta-features and algorithmic performances [Rivolli et al. 2018], the availability of both these elements in OpenML will be evaluated separately.

3.1. Availability of meta-features

Upon user submission of a dataset, OpenML automatically generates the corresponding meta-features, called "qualities" on the platform. Using the classification proposed by [Rivolli et al. 2018], all qualities from datasets present in OpenML can be categorized in the groups from Table 1.

Currently, according to the website and data extracted from the Python API, there are about 5,300 datasets available on the platform. However, not all meta-features are available for every dataset. In fact, most datasets do not contain all possible qualities that describe them. The information from a total number of 5,259 datasets were downloaded using the Python API in order in order to formulate Figure 2, which shows boxplots of the number of qualities filled in this pool of datasets.

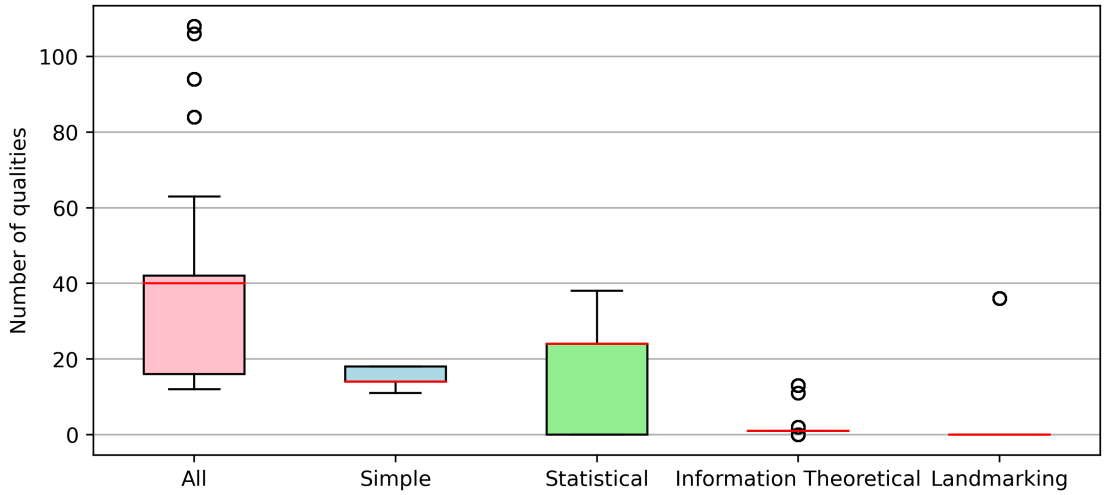

Figure 2. Distribution of the number of qualities available for 5,259 datasets of the OpenML repository.

In Figure 2, dataset qualities were divided into five groups, which consists of a group containing all of the qualities and four individual groups of meta-features. Through the 'All' boxplot, it is possible to see that the median number of qualities filled in for this pool of datasets is of 40 qualities, and most of them contain a number between 20 and 40 qualities which are filled in. Datasets that contain more than 80 qualities are regarded as outliers. Indeed, only 260 of the 5,259 datasets have all 108 quality values filled in. In Figure 2, we can also observe that most of the qualities available are from the statistical

Table 1. Distribution of qualities (108 total) based on each meta-feature group.

Measure Type	Meta-Features
Simple	NumberOfClasses, NumberOfFeatures, NumberOfInstances, [Number/Percentage]OfBinaryFeatures, [Number/Percentage]OfInstancesWithMissingValues, [Number/Percentage]OfMissingValues, [Number/Percentage]OfNumericFeatures, [Number/Percentage]OfSymbolicFeatures, MajorityClass[Percentage, Size], MinorityClass[Percentage, Size], Dimensionality
Statistical	Quartile[1,2,3]KurtosisOfNumericAtts, Quartile[1,2,3]MeansOfNumericAtts, Quartile[1,2,3]SkewnessOfNumericAtts, Quartile[1,2,3]StdDevOfNumericAtts, [Max, Min, Mean]KurtosisOfNumericAtts, [Max, Min, Mean]MeansOfNumericAtts, [Max, Min, Mean]NominalAttDistinctValues, [Max, Min, Mean]SkewnessOfNumericAtts, [Max, Min, Mean]StdDevOfNumericAtts, StdvNominalAttDistinctValues
Information Theoretic	ClassEntropy, [Max, Min, Mean]AttributeEntropy, Quartile[1,2,3]AttributeEntropy, Quartile[1,2,3]MutualInformation, [Max, Min, Mean]MutualInformation, EquivalentNumberOfAtts, MeanNoiseToSignalRatio, AutoCorrelation
Model-Based	CfsSubsetEval_DecisionStump[AUC, ErrRate, Kappa],, CfsSubsetEval_NaiveBayes[AUC, ErrRate, Kappa], CfsSubsetEval_kNN1N[AUC, ErrRate, Kappa]
Landmarking	REPTreeDepth[1,2,3][AUC, ErrRate, Kappa], RandomTreeDepth[1,2,3][AUC, ErrRate, Kappa], kNN1N[AUC, ErrRate, Kappa], NaiveBayes[AUC, ErrRate, Kappa], J48.00001[AUC, ErrRate, Kappa], J48.0001[AUC, ErrRate, Kappa], J48.001[AUC, ErrRate, Kappa], DecisionStump[AUC, ErrRate, Kappa],

category. The median values filled in per category are: 14 simple qualities, 24 statistical, 1 information theoretical and zero landmarking.

Figure 3 presents the logarithm distribution of values of one of the meta-features for the datasets, which refers to the logarithm of the number of input attributes they have. Considering this factor, the absolute dimensionality value in OpenML datasets usually ranges from about 1 to 30. High dimensional datasets are less frequent in this repository.

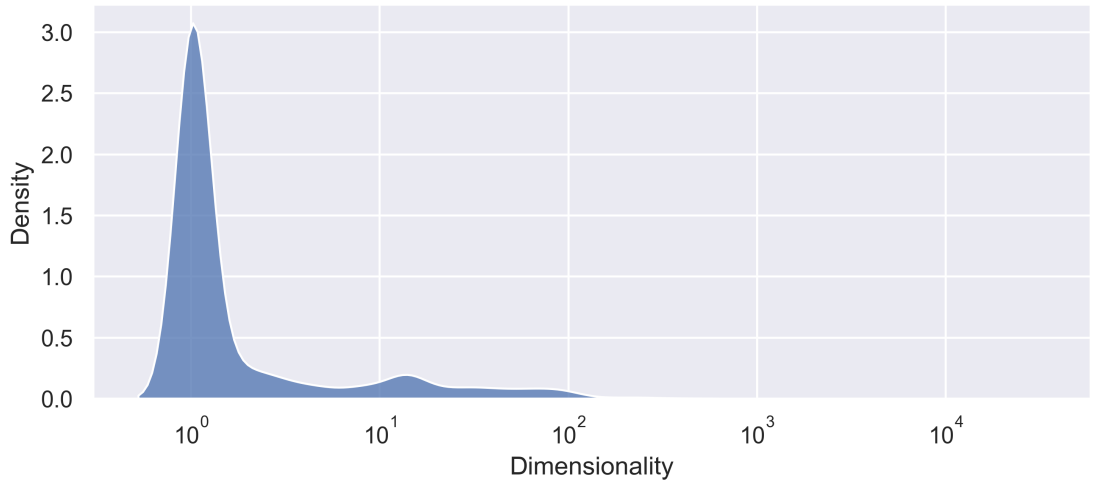

Figure 3. Distribution of dimensionality meta-feature for 5,259 datasets.

Another important aspect in dataset characterization in the case of classification problems is the majority class size, since datasets with an overwhelming imbalance in the proportions of observations per class pose a challenge in the correct classification of the minority class, which is often of most interest [Fernández et al. 2018]. Figure 4 provides a descriptive representation of the distribution of this meta-feature for a subset of the datasets where this information was available. In most of the 2,227 datasets from this pool, the majority class accounts for about 50% of the dataset. For binary classification datasets this does not impose an imbalance, but this might not be the case for multiclass classification datasets. What stands out the most is the presence of datasets where the percentage of majority class observations are close to 100%, imposing a high imbalance ratio which must be dealt with in order to obtain better ML models.

Regarding landmarking qualities, a subset of 897 datasets was analyzed, all of which contain the full set of landmarking qualities. They correspond to the Area Under the ROC Curve (AUC) metric for simple ML models. AUC ranges between 0 and 1 and higher values are indicative of a better predictive performance, while values around 0.5 (or below) are indicative of a bad predictive performance, which can be obtained at random. Figure 5 presents boxplots of the AUC values of KNN, Random Tree, Naïve Bayes, Decision Stump and REPTRee classifiers. Naïve Bayes was in general the best performing algorithm in the pool, although it also shows some outlier low AUC values. REPTree had also a higher predictive performance, but with a larger variation of values. The Decision Stump, which is a decision tree with one unique root node, was in general the worst performing classifier in this pool of algorithms. What stands out is that the

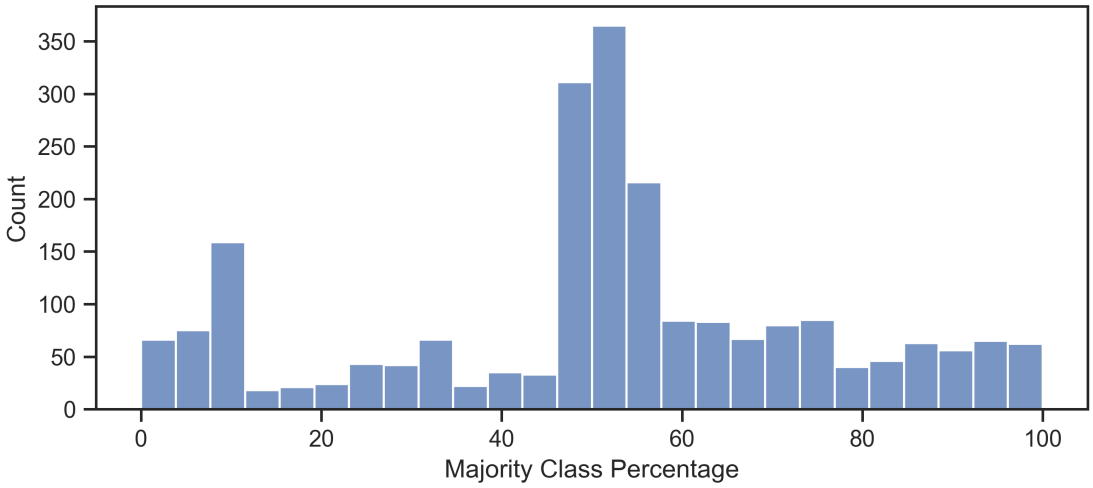

Figure 4. Distribution of the percentage of majority class observations on 2,227 datasets.

median of the AUC values are all above 0.7, which is in general a medium predictive performance.

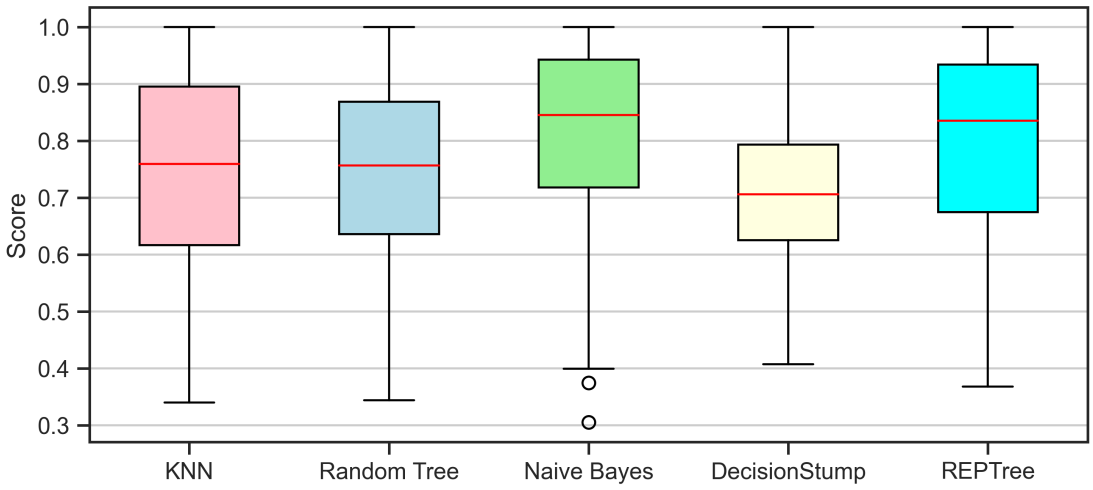

Figure 5. Distribution of the AUC score of different algorithms on 897 datasets.

3.2. Availability of algorithm performance

Since a dataset alone does not characterize a ML pipeline, the OpenML repository uses Tasks for storing the models' configurations for a problem. From Figure 6, it is clear that clustering problems are dominant in the website, followed by classification and regression tasks. On the other hand, survival analysis and ML challenge are the lest frequent tasks.

Another relevant value to consider is the estimation procedure used for each of the Tasks listed. This is shown on Figures 7 and 8 for regression and classification problems,

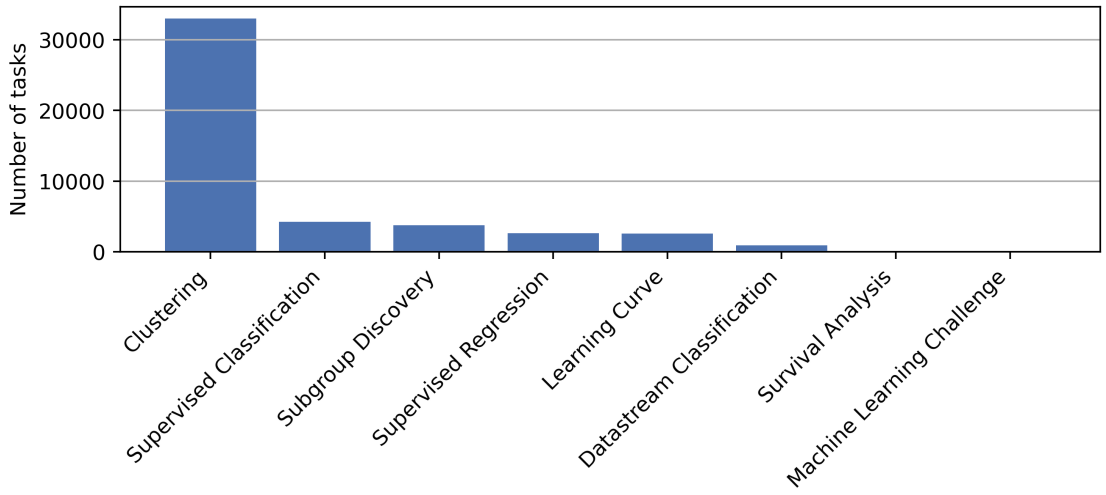

Figure 6. Distribution of tasks based on task type in the OpenML repository.

respectively. The standard 10-fold crossvalidation procedure is the most used, where the dataset is divided into ten folds and there are 10 train-test rounds. At each round, one fold is left out for testing and the remaining folds are joined for model training.

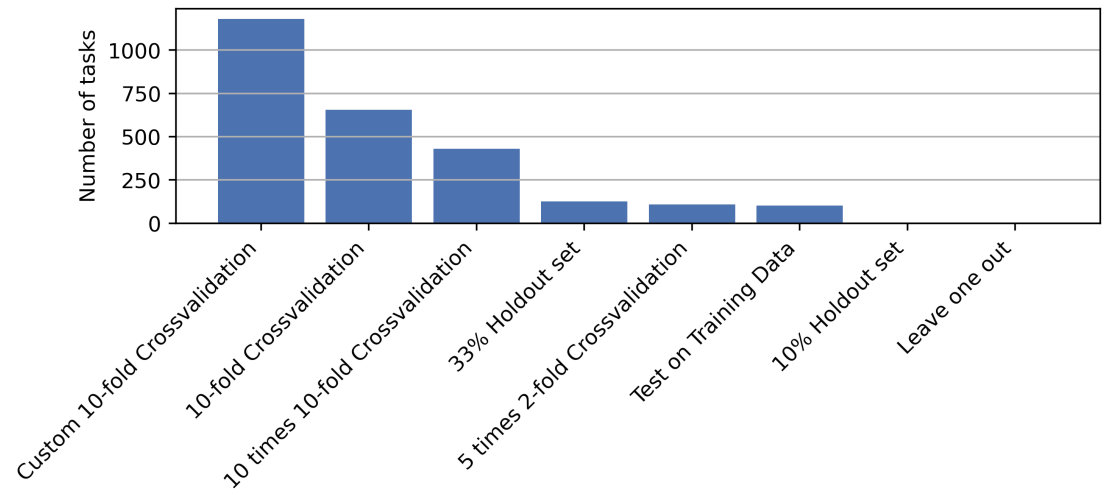

Figure 7. Distribution of estimation procedures for regression problems in OpenML.

While some algorithm performance values can already be assessed by the landmark meta-features, the performance of more complex models must be obtained from runs uploaded by users in the OpenML website. Each run uses an algorithm with specific hyper-parameter values. Data from the Python API reveals that there are 16,697 flows available on the platform, which comprises algorithms with specific versions and hyper-parameters, specially from scikit-learn and Weka, as well as simple functions for algorithmic performance evaluation. There are, also, 71 possible evaluation measures on

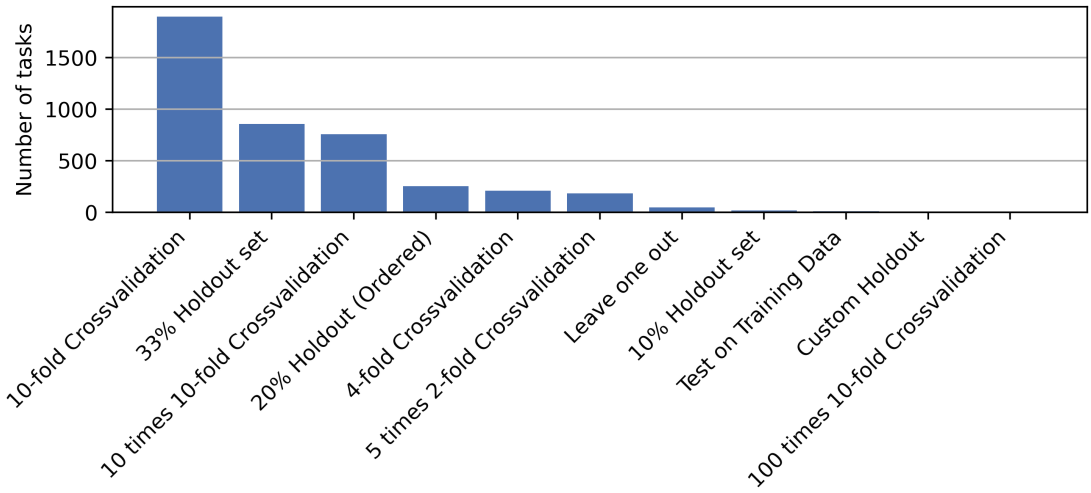

Figure 8. Distribution of estimation procedures for classification problems in OpenML.

the website.

3.3. Assembling a Meta-Dataset from OpenML

As a final endeavor in this work, a meta-dataset for classification problems was assembled. Summarizing, this meta-dataset has 459 instances composing the set P, described by 14 meta-features in the set F and evaluated by seven ML algorithms in A, whose AUC performance are recorded in Y[2].

The ML models considered are listed in Table 2, which are seven in total. These were the ML models with evaluations registered for most of the datasets when using the Python API. The estimation procedure used was 10-fold Cross validation and the algorithm performance measure was AUC. From the 528 datasets comprising the meta-dataset, 459 instances have no missing values.

Table 2 includes algorithms from the Weka Java software [Frank et al. 2005], version 3.9.2, available in the meta-dataset. As in Figure 5, the AUC metric was utilized to evaluate algorithm performance across all 528 datasets, as depicted in Figure 9. Only SVM and Hoeffding Tree had median AUC scores below 0.8. On the other hand, algorithms such as Random Subspace, KStar, and Random Forest, which were the best performing algorithms in the pool, exhibited a median AUC score close to 0.85, despite having a higher number of outliers compared to other algorithms. Overall, the average median AUC was approximately 0.8, which is a value significantly higher than that observed for the simpler landmarking algorithms presented in Figure 5. This distinction is expected for algorithms with finer hyper-parameter tuning and greater overall capacity.

With respect to the meta-features within the dataset, 14 qualities were chosen from the datasets evaluated by the algorithms in Table 2. The selection criteria was to include qualities with values registered for at least 466 datasets, in order to minimize the pres-

[2]https://github.com/NathanFCarvalho/Metadataset-OpenML

Table 2. Flows used as evaluation measures for the given classification tasks in OpenML.

Id	Flow	Description
7789	weka.kf.Bagging-NaiveBayes	Naive Bayes with Bagging method
7790	weka.kf.RandomForest	Random Forest implementation
7839	weka.kf.AttributeSelection-BestFirst-CfsSubsetEval-KStar	Instance based classifier called K Star
7844	weka.kf.Bagging-IBk5	K-nearest neighbours algorithms with K=5
7847	weka.kf.Bagging-SMO	A Suppor Vector Machine (SVM) algorithm based on Sequential Minimal Optimization (SMO)
7850	weka.kf.RandomSubspace	Random Subspace algorithm
8311	weka.kf.HoeffdingTree	Hoeffding Tree algorithm

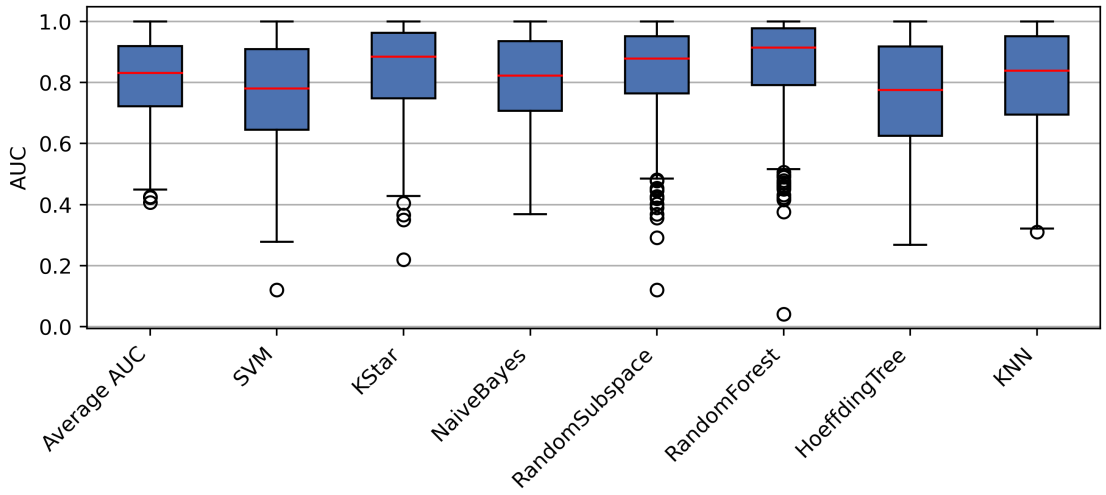

Figure 9. Distribution of the AUC score of different algorithms for the assembled meta-dataset.

ence of missing values in the meta-dataset. This limited the meta-features to simple and statistical categories, as they were more prevalent in the OpenML platform. Landmarking and model-based meta-features were relatively scarce in comparison. Furthermore, percentage-based features were preferred over those representing total numbers, as they are correlated. For example, the percentage of missing values quality was chosen instead of the number of missing values. The selected meta-features can be observed in Table 3.

The distribution of the number of instances, attributes, and classes of the datasets

Table 3. Distribution of qualities (14 total) for the assembled dataset based on each meta-feature group.

Measure Type	Meta-Features
Simple	NumberOfClasses, NumberOfFeatures, NumberOfInstances, PercentageOfBinaryFeatures, PercentageOfInstancesWithMissingValues, PercentageOfMissingValues, PercentageOfNumericFeatures, PercentageOfSymbolicFeatures, Dimensionality
Statistical	MeanKurtosisOfNumericAtts, MeanMeansOfNumericAtts, MeanSkewnessOfNumericAtts, MeanStdDevOfNumericAtts

contained in the assembled meta-dataset is depicted in Figure 10. The number of instances in these datasets predominantly falls below 10 thousand instances, while datasets larger than 20 thousand instances are nearly absent. This highlights the scarcity of datasets with a significant number of instances. In terms of classes, there is limited variation, with the majority of datasets having a range between two and 25 classes. Similarly, when considering the number of features, datasets containing more than 250 features are scarce.

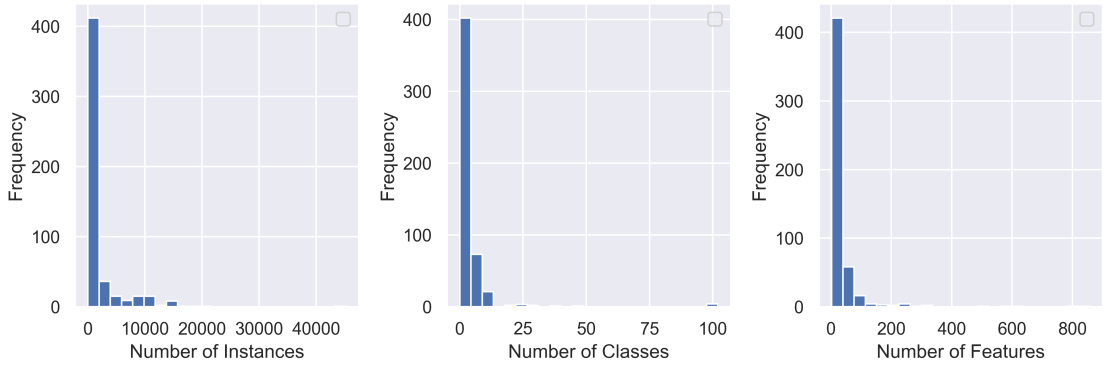

Figure 10. Distribution of the number of instances, attributes and classes.

Other meta-features that could provide helpful insights into the characteristics of the datasets present in the meta-dataset could not be included due to their limited presence across a significant number of datasets. For instance, the majority class percentage, which could offer valuable information, was only available in 307 datasets. This insight, along with the fact that OpenML meta-features are not sufficient for a full characterization of dataset complexity, highlight the current limitations of the meta-dataset and call for future efforts to complement the existing qualities. Expanding the range of meta-features would

enable a more thorough investigation of the behavior of this meta-dataset in the context of the ASP.

4. Conclusion

This work evaluated the extent of the information available on the OpenML dataset repository, which can be useful for MtL studies. We observed that a significant number of datasets within the repository lack complete characterization by meta-features. Specifically, out of the 5,250 datasets examined, only 260 datasets contained the complete set of 108 possible meta-features. This scarcity of comprehensive meta-data limits the ability of researchers to fully understand and analyze the datasets available on the platform.

Regarding the types of problems represented in the repository, the analysis revealed that clustering problems are the most prevalent. Furthermore, in the domains of regression and classification, the majority of datasets employed the 10-fold validation technique as an evaluation procedure. Next, a meta-dataset was assembled comprising 528 classification datasets for which the most complete set of algorithmic performance values and meta-features was available in the repository. Future work can explore the usefulness of such meta-dataset in MtL tasks for algorithm recommendation, as well as enhance it by considering regression datasets alongside.

In summary, the study highlights the importance of accessible and comprehensive meta-data in facilitating ML research. While the OpenML repository provides a valuable platform for sharing datasets, further enhancements could be valuable to enrich its meta-data, facilitating a deeper understanding of dataset complexity and enabling more robust MtL studies in the future.

Acknowledgements

To the FAPESP research agency (grants 21/06870-3, 23/04911-0 and 23/03958-2).

References

[Aha 1992] Aha, D. W. (1992). Generalizing from case studies: A case study. In Sleeman, D. and Edwards, P., editors, *Machine Learning Proceedings 1992*, pages 1–10. Morgan Kaufmann, San Francisco (CA).

[Alcalá-Fdez et al. 2011] Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., and García, S. (2011). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Multiple Valued Log. Soft Comput.*, 17(2-3):255–287.

[Alcobaça et al. 2020] Alcobaça, E., Siqueira, F., Rivolli, A., Garcia, L. P. F., Oliva, J. T., and de Carvalho, A. C. P. L. F. (2020). Mfe: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research*, 21(111):1–5.

[Bilalli et al. 2017] Bilalli, B., Abelló, A., and Aluja-Banet, T. (2017). On the predictive power of meta-features in openml. *International Journal of Applied Mathematics and Computer Science*, 27(4):697–712.

[Bischl et al. 2021] Bischl, B., Casalicchio, G., Feurer, M., Gijsbers, P., Hutter, F., Lang, M., Mantovani, R. G., van Rijn, J. N., and Vanschoren, J. (2021). Openml benchmarking suites.

[Fernández et al. 2018] Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., and Herrera, F. (2018). *Learning from imbalanced data sets*, volume 10. Springer.

[Feurer et al. 2019] Feurer, M., van Rijn, J. N., Kadra, A., Gijsbers, P., Mallik, N., Ravi, S., Mueller, A., Vanschoren, J., and Hutter, F. (2019). Openml-python: an extensible python api for openml. *arXiv*, 1911.02490.

[Frank et al. 2005] Frank, E., Hall, M. A., Holmes, G., Kirkby, R., Pfahringer, B., and Witten, I. H. (2005). *Weka: A machine learning workbench for data mining.*, pages 1305–1314. Springer, Berlin.

[Kühn et al. 2018] Kühn, D., Probst, P., Thomas, J., and Bischl, B. (2018). Automatic exploration of machine learning experiments on openml.

[Lorena et al. 2019] Lorena, A. C., Garcia, L. P., Lehmann, J., Souto, M. C., and Ho, T. K. (2019). How complex is your classification problem? a survey on measuring classification complexity. *ACM Computing Surveys (CSUR)*, 52(5):1–34.

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

[Newman et al. 1998] Newman, D., Hettich, S., Blake, C., and Merz, C. (1998). Uci repository of machine learning databases.

[Noy et al. 2019] Noy, N., Burgess, M., and Brickley, D. (2019). Google dataset search: Building a search engine for datasets in an open web ecosystem. In *28th Web Conference (WebConf 2019)*.

[Post et al. 2016] Post, M. J., van der Putten, P., and van Rijn, J. N. (2016). Does feature selection improve classification? a large scale experiment in openml. In Boström, H., Knobbe, A., Soares, C., and Papapetrou, P., editors, *Advances in Intelligent Data Analysis XV*, pages 158–170, Cham. Springer International Publishing.

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

[Rivolli et al. 2018] Rivolli, A., Garcia, L. P., Soares, C., Vanschoren, J., and de Carvalho, A. C. (2018). Towards reproducible empirical research in meta-learning. *arXiv preprint arXiv:1808.10406*, pages 32–52.

[Rivolli et al. 2022] Rivolli, A., Garcia, L. P., Soares, C., Vanschoren, J., and de Carvalho, A. C. (2022). Meta-features for meta-learning. *Knowledge-Based Systems*, 240:108101.

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

[Song et al. 2012] Song, Q., Wang, G., and Wang, C. (2012). Automatic recommendation of classification algorithms based on data set characteristics. *Pattern Recognition*, 45(7):2672–2689.

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

[Vanschoren et al. 2013] Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2013). Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60.

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

[Zöller and Huber 2021] Zöller, M.-A. and Huber, M. F. (2021). Benchmark and survey of automated machine learning frameworks.