Proceedings of Machine Learning Research 140:90–99, 2021 AAAI Workshop on Meta-Learning and MetaDL Challenge

# Exploiting Performance-based Similarity between Datasets in Metalearning

**Rui Leite** RLEITE@FEP.UP.PT
**Pavel Brazdil** PBRAZDIL@INESCTEC.PT
*LIAAD-INESC TEC, FEP, Univ. of Porto, Portugal*

**Editors:** Isabelle Guyon, Jan N. van Rijn, Sébastien Treguer, Joaquin Vanschoren

## Abstract

This paper describes an improved algorithm selection method of a previous method called *active testing* (Abdulrahman et al., 2018). This method seeks a workflow (or its particular configuration) that would lead to the highest gain in performance (e.g., accuracy).

The new version uses a particular performance-based characterization of each dataset, which is in the form of a vector of performance values of different algorithms. Dataset similarity is then assessed by comparing these performance vectors. One useful measure for this comparison is Spearman's correlation. The advantage of this measure is that it can be easily recalculated as more information is gathered. Consequently, as the tests proceed, the recommendations of the system get adjusted to the characteristics of the target dataset. We show that this new strategy leads to improved results of the active testing approach.

**Keywords:** Metalearning; Algorithm selection; Dynamic dataset similarity; Active testing

## 1. Introduction

The aim of metalearning is to use past performance results on different tasks/datasets to recommend the potentially best algorithm (or workflow) for the new (target) task/dataset (Michie et al., 1994; Brazdil and Henery, 1994). In the early days of metalearning, the "past results" involved solely the results on prior datasets (i.e., past metaknowledge). This vision has changed later. One example of this is the *Active Testing* method (Leite et al., 2012a) that used also past results on the target dataset to select the next algorithm to test.

The area of AutoML uses both the results on the target dataset and metaknowledge (performance results and other information) obtained on prior datasets in various ways. Some researchers used it to initialize the search (Reif et al., 2012; Feurer et al., 2014, 2015). Others tried to explore past results directly in the search (Wistuba, 2018).

One crucial problem in these systems is to decide which part of hypothesis space should be searched through at any given point. Metalearning/AutoML systems need to resolve the following two major issues. One is how to control the selection process of different alternatives in the configuration space. The second one is how to focus on parts of the configuration space that are actually relevant to the current task. In this paper we focus on the second issue.

Our aim is to adjust the configuration space as the search proceeds, while taking into account the current task/dataset and its measures. Obviously, static dataset measures that do not change, while the search proceeds, are not very useful here. So, it is

necessary to employ measures that are dynamic and performance-based measures are just what is needed here. They capture information about the search and thus enable to reorient the search.

In this paper we show how certain performance-based measures can be incorporated into the variant of active testing method (Abdulrahman et al., 2018). This approach relies on the concept of *vectors of performance values* (*multiple landmarkers*) obtained on different datasets that can be used to characterize datasets. Dataset similarity is then assessed by comparing these two performance vectors, by employing, for instance, Spearman's correlation. The advantage of this similarity is that it can be easily updated as the tests proceed on the target dataset. So, the similarity of the target dataset to existing datasets gets more refined as more tests are carried out. This has a positive effect on the recommendations suggested by the system. We have used this strategy on an experimental setup and show that this new strategy leads to improved results.

## 2. Related work

Performance-based measures exploited in this work are not new. The concept of *landmarkers* representing the performance of simple algorithms was introduced by Pfahringer et al. (2000). *Relative landmarkers* involve ratios (or differences) of two performance values of two algorithms. Relative landmarkers that exploit differences were exploited in the active testing method (Leite et al., 2012a). In various subsequent publications the term *performance gain* was used for the same concept (Abdulrahman et al., 2018). The concept of *subsampling landmarkers* refers to the algorithm performance on samples of data (Fürnkranz and Petrak, 2001).

Most of the work on active testing did not use static dataset measures. This is partly due to the findings of Leite et al. (2012b). They have shown that the version with static dataset measures (*ATdc*) did not surpass the baseline variant (*AT0*) that did not use any measures.

The authors also used a version of active testing (*AT1*) that incorporated a performance-based dataset similarity measure. The similarity between the target dataset $d_t$ and some other dataset $d_k$ was assessed using a method captured in Eq. 1.

$$Sim_{pol}(d_t, d_k) = \\ \Delta P(a_*, a_{*^-}, d_t) > 0 \ \& \ \Delta P(a_*, a_{*^-}, d_k) > 0 \quad (1)$$

where $\Delta P(a_*, a_{*^-}, d_t)$ represents performance gain of current best algorithm (incumbent) $a_*$ over the previous incumbent $a_{*^-}$ on dataset $d_t$. The condition states that two datasets are similar if the performance gains on datasets $d_t$ and $d_k$ are both positive. The authors reported that the version *AT1* achieved better performance than the baseline method *AT0*. We have evaluated this version on the current setup and found that its performance was inferior to the new version discussed here. This is probably due to the fact that *AT1* takes into account the performance of two algorithms only in the calculation of similarity, which is not the case with the current method. This approach can be seen as a predecessor of the improved method described here.

## 3. Enhancing the Active Testing Method

### 3.1. Overview of active testing

The active testing method $AT^*$ uses past performance results to guide the search for the potentially best algorithm for a given task/dataset (Abdulrahman et al., 2018). It is an iterative procedure that uses, at each

step, algorithm $a_*$, considered as the current best (the incumbent), to determine the best competitor ($a_c$) to test. This is done on the basis of estimates elaborated on the basis of past performance results. The aim is to consider different algorithms and select the one that maximizes the performance gains on prior datasets. This can be captured by Eq. 2,

$$a_c = \underset{a_k}{\text{argmax}} \sum_{d_i \in D} \Delta P(a_k, a_*, d_i) \quad (2)$$

where $a_c$ and $a_k$ represent algorithms from a given set of algorithms (portfolio) $A$, $D$ represents the set of datasets used in prior experiments for which we have obtained performance results for the set of algorithms $A$. The term $\Delta P(a_k, a_*, d_i)$ represents the estimate of the performance gain of algorithm $a_k$ with respect to $a_*$ on dataset $d_i$. In the previous publication by Abdulrahman et al. (2018) this was calculated using Eq. 3, where $A3R^{d_i}_{a_k,a_*}$ is computed as shown in Eq. 4,

$$\Delta P(a_k, a_*, d_i) = max(A3R^{d_i}_{a_k,a_*} - 1, 0) \quad (3)$$

$$A3R^{d_i}_{a_k,a_*} = \frac{P^{d_i}_{a_k} / P^{d_i}_{a_*}}{(T^{d_i}_{a_k} / T^{d_i}_{a_*})^q} \quad (4)$$

where $P^{d_i}_{a_j}$ represents the performance of algorithm $a_j$ (e.g., accuracy, AUC etc.) on dataset $d_i$ and $T^{d_i}_{a_j}$ is the time spent on running $a_j$. The parameter $q$ expresses the relative importance of time with respect to accuracy. The authors suggested that the value of $q = 1/32$ represents a good default. We note that smaller values would attribute less importance to time, while larger values would have the opposite effect. As can be seen, Eq. 3 uses values greater than 1 to calculate the value of $\Delta P$.

### 3.2. Introducing an improved version of $\Delta P$

In this study we have also conducted experiments with a simpler version of $\Delta P$ defined in Eq. 5:

$$\Delta P(a_k, a_*, d_i) = A3R^{d_i}_{a_*,a_k} \quad (5)$$

The corresponding version of active testing method that uses this $\Delta P$ is referred to as $AT^{*'}$. Surprisingly, we found that this version has achieved better results than the original version $AT^*$ (see Section 4.2). So, we have used the new version ($AT^*$) as the basis for further improvements.

### 3.3. Role of dataset measures in algorithm selection

A great deal of work in the area of metalearning has explored dataset measures in the design of the algorithm selection methods (Vilalta and Drissi, 2002; Brazdil et al., 2009; Muñoz et al., 2018). This is motivated by the fact that dataset measures enable to restrict search to a subset of datasets that is most similar to the target dataset $d_t$. In classification tasks many different dataset measures have been defined in the past.

The measures can be organized into two major groups depending on whether they are *static* (i.e., non-performance-based) or *performance-based*. The static ones include various types, including, for instance, simple, statistical and information-based measures (Brazdil et al., 2009). Some researchers have also considered concept characterization and complexity-based measures (Muñoz et al., 2018).

The performance-based measures include *landmarkers* representing performance of algorithms on given datasets, *sampling landmarkers* representing performance of algorithms on samples of given datasets and *partial learning curves* (series of sampling landmarkers on a particular dataset).

In this paper we focus on performance-based measures, as these have the advantage that they can be updated as the search proceeds. The following subsection describes the details.

**Performance-based similarity**

Performance-based dataset similarity uses, as the name suggests, performance values to determine similarity between datasets. Specifically, here we consider the concept of *a series of landmarkers*, representing performance values of different algorithms on a given dataset. Let us represent this measure by $P_A^{d_t}$, where $A$ represents the algorithms and $d_t$ the given target dataset. This measure can be calculated for any dataset used in the past.

The measure $P_A^{d_i}$ is particularly useful, as it can be recalculated as the search progresses and more algorithms have been tested. Consequently, it can thus be used to refine the measure of similarity between datasets, as more results have become available. More details about how this is done are given in the next subsection.

**Correlation-based similarity between datasets**

Pairs of measures $P_A^{d_t}$ and $P_A^{d_i}$ discussed in the previous subsection can be used to calculate dataset similarity. Different functions can be used to estimate this similarity. First, let us consider Spearman's correlation (Neave and Worthington, 1992). The similarity based on this can be calculated as shown in Eq. 6

$$Sim_{sp}(P_A^{d_t}, P_A^{d_i}) = r_s(P_A^{d_t}, P_A^{d_i}) \quad (6)$$

where $r_s$ represents the Spearman's correlation function. The weighted rank measure of correlation $r_w$ (da Costa and Soares, 2005; da Costa, 2015) gives different importance to items according to where they appear in the ranking. The importance decreases linearly with the position in the ranking. So, this measure can be used in the definition of similarity, as is shown in Eq. 7.

$$Sim_{sw}(P_A^{d_t}, P_A^{d_i}) = r_w(P_A^{d_t}, P_A^{d_i}) \quad (7)$$

### 3.4. Exploiting correlation-based similarity in algorithm selection

The correlation-based similarity described above can be used to upgrade the active testing method. The upgraded version is referred to as $AT^*_{DS}$. This method requires various inputs, including the target dataset $d_t$ and the set of algorithms to choose from. It requires also a good starting point, i.e., an algorithm that should be used to initiate the search. Any algorithm could be used, but here we use the topmost algorithm returned by the average ranking method $AR^*$, as it usually leads to good results. Let us refer to this ranking as $A^{AR^*}$.

This method requires also the value of parameter $q$ that controls the relative importance of accuracy and time. Here we follow the recommendation given by Abdulrahman et al. (2018) and use the default setting of $q = 1/32$.

This method is based on the algorithm in Abdulrahman et al. (2018), but differs from it in one important aspect. It includes dataset weights capturing dataset similarity of each dataset $d_i$ to the target dataset $d_t$. The identification of the best competitor takes these weights into account. Basically, the method calculates a weighted average of the individual performance gains using a vector of weights $W_{d_i}$, as shown in Eq. 8.

$$a_c = \underset{a_k \in A}{\text{argmax}} \sum_{d_i \in D} (W_{d_i} \times \Delta P(a_k, a_*, d_i, q)) \quad (8)$$

The complete algorithm is shown in Algorithm 1. The instructions used to recalculate the dataset similarity have been separated out and appear in Algorithm 2. They are recalculated using $W_{d_i} \leftarrow Sim_{sp}(P_{A_t}^{d_t}, P_{A_t}^{d_i})$ on the basis of Spearman's rank correlation applied to two vectors, $P_{A_t}^{d_t}$ and $P_{A_t}^{d_i}$, that include the performance values of different algorithms.

**Algorithm 1:** Algorithm with dynamic performance similarity

$AT^*_{DS}(d_t, D, A^{AR^*}, A, q)$
$A_t \leftarrow a_* \leftarrow A^{AR^*}[1]$
*(Initialize alg. sequence)*
$W_{d_i}$ $_{d_i \in D} \leftarrow 1/|D|$ *(Initialize dataset weights)*
$P^{d_t}_{a_*} \leftarrow CV(a_*, d_t)$
*(Use CV to evaluate $a_*$ on $d_t$)*
**while** $|A| > 0$ **do**
  $a_c = \underset{a_k \in A}{\text{argmax}} \sum_{d_i \in D} (W_{d_i} \times \Delta P(a_k, a_*, d_i, q))$
  *(Best competitor)* $P^{d_t}_{a_c} \leftarrow CV(a_c, d_t)$
  $P^{d_t}_{A_t} = \{P^{d_t}_{a_k} : a_k \in A_t\}$
  ***Recalc. dataset weights - see Algorithm 2 (or 3)***
  *Choose between $a_c$ and $a_*$ depending on performance (accuracy):*
  **if** $P^{d_t}_{a_c} > P^{d_t}_{a_*}$ **then**
    $P^{d_t}_{a_*} \leftarrow P^{d_t}_{a_c}$
    $a_* \leftarrow a_c$
  **end**
  $A_t \leftarrow A_t \cup \{a_c\}$
  $A \leftarrow A \setminus \{a_c\}$ *(Move $a_c$ from $A$ to $A_t$)*
**end**
**return** $a_*, P^{d_t}_{a_*}$

**Algorithm 2:** Recalculating dataset weights with correlation-based similarity

**for** $d_i$ *in* $D$ **do**
  $P^{d_i}_{A_t} = \{P^{d_i}_{a_k} : a_k \in A_t\}$
  *(Similar performance (accuracy) on $d_i$)*
  $W_{d_i} \leftarrow Sim_{sp}(P^{d_t}_{A_t}, P^{d_i}_{A_t})$
  *(Recalculate weights)*
**end**

### 3.5. Exploiting dynamic adjustment of similarity

The algorithm discussed in this section exploits the idea of dynamic adjustment of the dataset similarity. This is done first by calculating the difference between the value of $\Delta P(a_c, a_*, d_t, q)$ obtained on the target dataset $d_t$ and the predicted value

$$\sum_{d_i \in D} (W_{d_i} \times \Delta P(a_c, a_*, d_i, q))$$

on prior datasets. The difference of the two values is used as a *reward*. This value can be positive or negative. If it is positive (negative) it is used to increase (decrease) the weight of the datasets that contributed most to the decision. The contribution is judged by considering the value of $\Delta P(a_c, a_*, d_i, q)$. The larger this value is, the larger the adjustment. Parameter $\eta$ controls the rate of adjustment. The instructions just described are summarized in Algorithm 3.

The complete algorithm is the same as Algorithm 1 with the exception that the code of Algorithm 2 is substituted by the code of Algorithm 3. Our preliminary experiments have shown that the setting $\eta = 0.5$ leads to good results, so we have used this value in all experiments reported further on.

## 4. Experiments and Results

In this section we describe experiments that were conceived to compare the performance of different metalearning methods. We start by describing the experimental setup, which involves, in the first place, the datasets and workflows used in the process. Following this, we list the metalearning methods used in the experiments.

**Algorithm 3:** Recalculating dataset weights using reinforcement learning

$$Rewd \leftarrow \Delta P(a_c, a_*, d_t, q) - \sum_{d_i \in D} (W_{d_i} \times \Delta P(a_c, a_*, d_i, q))$$

**for** $d_i$ *in* $D$ **do**
  $\Delta_{W_{d_i}} = \eta \times Rewd \times \Delta P(a_c, a_*, d_i, q)$
  *(Calculate weight adjustment)*
  $W_{d_i} \leftarrow W_{d_i} + \Delta_{W_{d_i}}$
  *(Recalculate dataset weight)*
**end**

### 4.1. Experimental setup and methodology

#### Datasets and workflows used

The experimental setup used was similar to the one in the previous work in Cachada et al. (2017). It included 184 workflows, corresponding to two sets of 92 workflows. The first set included 62 algorithms with default configurations and 30 variants of some of the algorithms with different hyperparameter configurations (3 extra versions of MLP, 7 of SVM, 7 of RFs, 8 of J48 and 5 of k-NN). The other set was similar to the first set, but with one difference. The particular classifier was preceded by feature selection (i.e., CFS method (Hall, 1999)).

#### Baseline metalearning methods

In this study we consider the following metalearning methods as baselines:

- $AR^*$ - average ranking as in Abdulrahman et al. (2018) with parameter setting $q = 1/32$,
- $AT^*$ - active testing as in Abdulrahman et al. (2018) with parameter setting $q = 1/32$,
- $AT^{*'}$ - active testing variant described in subsection 3.1 with the same parameter setting of q as in $AT^*$,

#### Metalearning methods with dynamic dataset similarity

In this study we have considered the following metalearning methods that employ dynamic similarity:

- $AT^{*'}_{DS}$ - active testing with dynamic similarity based on Spearman's correlation,
- $AT^{*'}_{DW}$ - similar to above, but based on weighted rank correlation.
- $AT^{*'}_{D\Delta}$ - active testing with dynamic similarity based on reinforcement learning.

#### Evaluation methodology

The experiments were run on 37 datasets shown in Table 4 in the Appendix. All datasets include the OpenML *task id*, enabling anyone to download the same data.

The evaluation was done in a leave-one-out mode. In each cycle the performance metadata on all but one datasets were used by each metalearning system to generate a series of recommendations of algorithms (workflows) for the target dataset. Each recommended algorithm (workflow) was run and the corresponding performance and runtime were recorded. This in turn enables us to elaborate loss curves and rank curves. The curves obtained for individual datasets in the leave-one-out cycle are then aggregated into a single curve for each metalearning method. The details of how this is done are given in the next subsection.

### 4.2. Results

In this subsection we present the evaluation results in terms of rank and loss curves of different metalearning methods.

#### Analysis of rank curves

The rank curves show the evolution in time of the ranks of different metalearning methods considered. The calculation is done for each time point separately. For each time point, different datasets are considered one by one. Suppose we are considering dataset $d_j$. The performance (or loss) of different metalearning methods on this dataset is used to calculate their ranks. After all datasets have been processed, it is possible to calculate the mean rank of a particular method across all datasets. The points obtained this way for a particular metalearning method are used to construct a rank curve. This is then repeated for all metalearning methods.

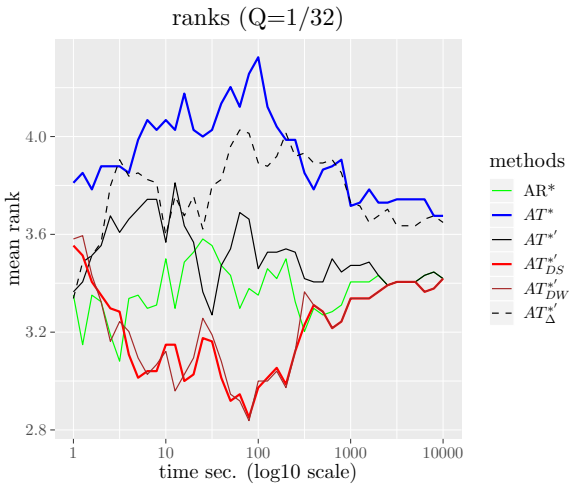


Figure 1: Rank curves of different metalearning methods for 184 workflows

The rank curves obtained by different metalearning methods are shown in Fig. 1.

Table 1: Mean interval rank ($MIR$) values for different runtime intervals, 6 metalearning methods and 184 workflows

| | 1-10 | 10-100 | 100-$10^3$ | $10^3$-$10^4$ | 1-$10^4$ |
|---|---|---|---|---|---|
| $AR^*$ | 3.29 | 3.44 | 3.35 | 3.37 | 3.36 |
| $AT^*$ | 3.92 | 4.12 | 3.95 | 3.92 | 3.98 |
| $AT^{*'}$ | 3.59 | 3.55 | 3.48 | 3.51 | 3.53 |
| $AT^{*'}_{DS}$ | **3.25** | **3.03** | **3.16** | **3.21** | **3.16** |
| $AT^{*'}_{DW}$ | 3.26 | 3.04 | 3.17 | 3.22 | 3.17 |
| $AT^{*'}_{D\Delta}$ | 3.68 | 3.81 | 3.89 | 3.77 | 3.79 |

Table 1 accompanied these curves and helps in the analysis of results. It shows, for each metalearning method, a mean rank that this method has achieved in a particular runtime interval. We refer to this measure as *mean interval rank ($MIR$)*. The first interval in our table spans from 1 to 10 seconds. We see, for instance, that the proposed method $AT^{*'}_{DS}$ has the lowest rank in this interval, namely 3.25.

The rank curves show that the proposed method $AT^{*'}_{DS}$, that exploits correlation-based similarity, is the best one in all runtime intervals. This version uses Spearman's correlation in the calculation of dataset similarity. The variant $AT^{*'}_{DW}$, that uses a weighted correlation, has a rather similar rank and behaviour.

As for the improved variant $AT^{*'}$, discussed in subsection 3.2, we see that it achieves better mean rank (3.21) than the original version used in Abdulrahman et al. (2018) (rank 3.92). The original version was even inferior to average ranking method $AR^*$ (rank 3.37), which is much simpler than $AT^*$. This is not the case with $AT^{*'}$, which has surpassed $AR^*$, as expected.

#### Analysis of loss curves

The results of different methods are presented in the form of loss curves, which have been used by various authors to compare

Table 2: $MIL$ values for different runtime intervals, 6 metalearning methods and 184 workflows

| | 1-10 | 10-100 | 100-$10^3$ | 1-$10^4$ |
|---|---|---|---|---|
| $AR^*$ | **0.9733** | 0.1003 | 0.0000 | 0.2790 |
| $AT^*$ | 1.1363 | 0.1757 | 0.0078 | 0.3437 |
| $AT^{*'}$ | 1.1597 | **0.0673** | 0.0000 | 0.3216 |
| $AT^{*'}_{DS}$ | 0.9900 | 0.0716 | 0.0000 | **0.2787** |
| $AT^{*'}_{DW}$ | 1.0282 | 0.0914 | 0.0000 | 0.2914 |
| $AT^{*'}_{D\Delta}$ | 1.0688 | 0.1361 | 0.0000 | 0.3157 |

different metalearning methods (see e.g., (van Rijn et al., 2015; Abdulrahman et al., 2018)). The loss curve used here represents a *median curve* across all datasets. The loss curves obtained by different metalearning methods are shown in Fig. 2. Each loss curve can be characterized by the mean value in a given interval ($MIL$). The results are shown in Table 2.

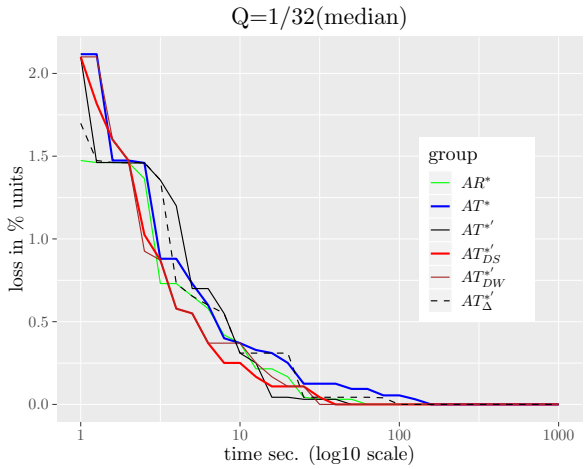


Figure 2: Loss curves of different metalearning methods for 184 workflows

We see that the results corroborate the conclusions of the previous subsection. We note again that the proposed method $AT^{*'}_{DS}$ is the best one in the interval spanning from 1 to $10^4$ seconds.

## 5. Conclusions

In this paper we have described a new version of *Active Testing* method that can be used for selecting algorithms (workflows) for a given task. This method seeks a configured algorithm (workflow) that would lead to the highest gain in performance (e.g., accuracy).

Our work on the active testing approach had also one rather positive side-effect. We have noted that the way the performance gain was calculated before (Abdulrahman et al., 2018) could be improved. Curiously enough, the improvement was obtained by simplifying the formula used before.

The new version uses a novel performance-based measure - *multiple landmarkers* - to compute dataset similarity. We have investigated several alternative ways of calculating this similarity and have shown that the variant $AT^{*'}_{DS}$, based on Spearman's correlation, achieves the best results. The advantage of this measure is that it can be recalculated and refined as more performance data is obtained through tests on the target dataset. Consequently, the recommendations of the metalearning system that are based on past metaknowledge get adjusted to the target dataset. We have shown that this new strategy leads to improved results of the active testing approach.

The concept of performance-based similarity based on multiple landmarkers reported here is, to the best of our knowledge, novel.

Although these results can be considered as preliminary – given that we have not yet conducted tests regarding the statistical significance of differences – we believe that they represent an important finding that can be reused in other settings. Performance-based dataset measures are relatively easy to define in many domains, unlike the static dataset counterparts, which may not even be effective.

Acknowledgements

This work was financed by National Funds through the Portuguese funding agency, FCT - Fundacão para a Ciência e a Tecnologia, within project: UIDB/50014/2020.

## References

S.M. Abdulrahman, P. Brazdil, Jan N. van Rijn, and J. Vanschoren. Speeding up algorithm selection using average ranking and active testing by introducing runtime. *Machine Learning*, 107(1):79–108, Jan 2018. ISSN 1573-0565.

P. Brazdil and R. J. Henery. Analysis of results. In D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors, *Machine Learning, Neural and Statistical Classification*, chapter 10 (175-212). Ellis Horwood, 1994.

P. Brazdil, C.G. Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining*. Springer, 2009.

M. Cachada, S.M. Abdulrahman, and P. Brazdil. Combining feature and algorithm hyperparameter selection using some metalearning methods. In *Proc. of Workshop AutoML 2017, CEUR Proceedings Vol-1998*, pages 75–87, 2017.

J. P. da Costa. *Rankings and Preferences: New Results in Weighted Correlation and Weighted Principal Component Analysis with Applications*. Springer, 2015.

J. P. da Costa and C. Soares. A weighted rank measure of correlation. *Aust. N.Z. J. Stat.*, 47(4):515–529, 2005.

M. Feurer, J. T. Springenberg, and F. Hutter. Using meta-learning to initialize bayesian optimization of hyperparameters. In *ECAI WS on Metalearning and Algorithm Selection (MetaSel)*, pages 3–10, 2014.

M. Feurer, A. Klein, K. Eggensperger, J.T. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In C. et al. Cortes, editor, *Advances in Neural Information Processing Systems 28*, NIPS'15, pages 2962–2970. Curran Associates, Inc., 2015.

J. Fürnkranz and J. Petrak. An evaluation of landmarking variants. In C.G. Carrier, N. Lavrač, and S. Moyle, editors, *Working Notes of the ECML/PKDD2000 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, pages 57–68, 2001.

M.A. Hall. *Correlation-based feature selection for machine learning*. PhD thesis, University of Waikato, 1999.

R. Leite, P. Brazdil, and J. Vanschoren. Selecting Classification Algorithms with Active Testing. In *Machine Learning and Data Mining in Pattern Recognition*, pages 117–131. Springer, 2012a.

R. Leite, P. Brazdil, and J. Vanschoren. Selecting Classification Algorithms with Active Testing on Similar Datasets. In J. Vanschoren, P. Brazdil, and J.-U. Kietz, editors, *PlanLearn-2012, 5th Planning to Learn Workshop at ECAI*, 2012b.

D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

M. Muñoz, L. Villanova, D. Baatar, and K. Smith-Miles. Instance Spaces for Machine Learning Classification. *Machine Learning*, 2018.

H. R. Neave and P. L. Worthington. *Distribution-Free Tests*. Routledge, 1992.

B. Pfahringer, H. Bensusan, and C.G. Carrier. Meta-learning by Landmarking Various Learning Algorithms. In P. Langley,

editor, *Proc. of 17th ICML*, pages 743–750, 2000.

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

J. N. van Rijn, S.M. Abdulrahman, P. Brazdil, and J. Vanschoren. Fast algorithm selection using learning curves. In *Int. Symp. on Intelligent Data Analysis XIV*, pages 298–309, 2015.

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

M. Wistuba. *Automated Machine Learning: Bayesian Optimization, Meta-Learning & Applications*. PhD thesis, University of Hildesheim, Germany, 2018.

## Appendix

Table 3: Alternative hyperparameter configurations.

| Algorithm | Parameter interval* | Parameter description** |
|---|---|---|
| Random Forest | P = [80, **100**] | Bag size (% of the training set size) |
| | K = [**0**, 250] | #attrs. to randomly investigate, where 0 = int(log2(#attributes)+1). |
| | M = [**1**, 10] | Minimum of instances per leaf . |
| J48 | C = [0.01, **0.25**, 0.5] | Confidence threshold for pruning (smaller values incur more pruning). |
| | M = [**2**, 10, 20] | Minimum of instances per leaf . |
| SMO PolyKernel | C = [0.1, **1**, 10] | Complexity parameter. |
| | E = [**1**, 2] | Exponent of the polynomial kernel. |
| SMO RBFKernel | C = [0.1, **1**, 10] | Complexity parameter. |
| IBK | K = [**1**, 5, 20] | The number of neighbors to use. |
| | I = [**No**, Yes] | Perform distance weighting by 1/distance? |
| Multilayer Perceptron | H = [**a**, o] | Number of hidden layers, where 'a' = (#attributes + #classes)/2 'o' = #classes. |
| | D = [**No**, Yes] | Whether the learning rate decays as training progresses. |

*Parameters in bold are the default.
**Adapted from WEKA documentation.
The parameters not mentioned were set to their default values except in MultilayerPerceptron where epochs ($N$) set to 100 instead (def. 500)

Table 4: Datasets used in the study.

| Dataset | OpenML Task_id | #Inst. | #Attrs. | #Classes |
|---|---|---|---|---|
| anneal | 2 | 898 | 39 | 6 |
| kr-vs-kp | 3 | 3 196 | 37 | 2 |
| letter | 6 | 20 000 | 17 | 26 |
| balance-scale | 11 | 625 | 5 | 3 |
| mfeat-factors | 12 | 2 000 | 217 | 10 |
| mfeat-fourier | 14 | 2 000 | 77 | 10 |
| breast-w | 15 | 699 | 10 | 2 |
| mfeat-karhunen | 16 | 2 000 | 65 | 10 |
| mfeat-morphological | 18 | 2 000 | 7 | 10 |
| mfeat-pixel | 20 | 2 000 | 241 | 10 |
| car | 21 | 1 728 | 7 | 4 |
| mfeat-zernike | 22 | 2 000 | 48 | 10 |
| cmc | 23 | 1 473 | 10 | 3 |
| mushroom | 24 | 8 124 | 23 | 2 |
| nursery | 26 | 12 960 | 9 | 5 |
| optdigits | 28 | 5 620 | 65 | 10 |
| credit-a | 29 | 690 | 16 | 2 |
| page-blocks | 30 | 5 473 | 11 | 5 |
| credit-g | 31 | 1 000 | 21 | 2 |
| pendigits | 32 | 10 992 | 17 | 10 |
| cylinder-bands | 33 | 540 | 40 | 2 |
| segment | 36 | 2 310 | 20 | 7 |
| diabetes | 37 | 768 | 9 | 2 |
| soybean | 41 | 683 | 36 | 19 |
| spambase | 43 | 4 601 | 58 | 2 |
| splice | 45 | 3 190 | 62 | 3 |
| tic-tac-toe | 49 | 958 | 10 | 2 |
| vehicle | 53 | 846 | 19 | 4 |
| waveform-5000 | 58 | 5 000 | 41 | 3 |
| electricity | 219 | 45 312 | 9 | 2 |
| solar-flare | 2068 | 1 066 | 13 | 3 |
| yeast | 2073 | 1 484 | 9 | 10 |
| satimage | 2074 | 6 430 | 37 | 6 |
| abalone | 2075 | 4 177 | 9 | 29 |
| kropt | 2076 | 28 056 | 7 | 18 |
| baseball | 2077 | 1 340 | 18 | 3 |
| eucalyptus | 2079 | 736 | 20 | 5 |

Table 5: Algorithms used in the study.

| # | Algorithm | # | Algorithm |
|---|---|---|---|
| 1 | A1DE | 32 | Kstar |
| 2 | AdaBoostM1_DecisionStump | 33 | LADTree |
| 3 | AdaBoostM1_IBk | 34 | LMT |
| 4 | AdaBoostM1_J48 | 35 | LogitBoost_DecisionStump |
| 5 | AdaBoostM1_LMT | 36 | LWL_DecisionStump |
| 6 | AdaBoostM1_NaiveBayes | 37 | LWL_J48 |
| 7 | AdaBoostM1_OneR | 38 | LWL_RandomTree |
| 8 | AdaBoostM1_RandomTree | 39 | MultiBoostAB_DecisionStump |
| 9 | AdaBoostM1_REPTree | 40 | MultiBoostAB_IBk |
| 10 | Bagging_DecisionStump | 41 | MultiBoostAB_J48 |
| 11 | Bagging_IBk | 42 | MultiBoostAB_JRip |
| 12 | Bagging_J48 | 43 | MultiBoostAB_NaiveBayes |
| 13 | Bagging_Jrip | 44 | MultiBoostAB_OneR |
| 14 | Bagging_LMT | 45 | MultiBoostAB_RandomTree |
| 15 | Bagging_LWL_DecisionStump | 46 | MultiBoostAB_REPTree |
| 16 | Bagging_NaiveBayes | 47 | MultilayerPerceptron |
| 17 | Bagging_OneR | 48 | NaiveBayes |
| 18 | Bagging_RandomTree | 49 | NaiveBayesUpdateable |
| 19 | Bagging_REPTree | 50 | NBTree |
| 20 | BayesNet | 51 | OneR |
| 21 | ClassificationViaRegression_M5P | 52 | PART |
| 22 | CVParameterSelection_ZeroR | 53 | RacedIncrementalLogitBoost_D.Stump |
| 23 | Dagging_DecisionStump | 54 | RandomCommittee_RandomTree |
| 24 | DecisionStump | 55 | RandomForest |
| 25 | DTNB | 56 | RandomSubSpace_REPTree |
| 26 | END_ND_J48 | 57 | RandomTree |
| 27 | HoeffdingTree | 58 | REPTree |
| 28 | IBK | 59 | SimpleLogistic |
| 29 | IterativeClassifierOptimizer_LogitBoost_D.Stump | 60 | SMO_PolyKernel |
| 30 | J48 | 61 | SMO_RBFKernel |
| 31 | JRip | 62 | ZeroR |