Algorithm Selection via Meta-learning and Sample-based Active Testing

Salisu Mamman Abdulrahman[1], Pavel Brazdil[1,2], Jan N. van Rijn[3], and Joaquin Vanschoren[4]

[1] LIAAD-INESC Tec, Porto, Portugal
salisu.abdul@gmail.com
[2] FEP, Univiversity of Porto, Porto, Portugal
pbrazdil@inesctec.pt
[3] Leiden University, Leiden, Netherlands
j.n.van.rijn@liacs.leidenuniv.nl
[4] Eindhoven University of Technology, Eindhoven, Netherlands
j.vanschoren@tue.nl

Abstract. Identifying the best machine learning algorithm for a given problem continues to be an active area of research. In this paper we present a new method which exploits both meta-level information acquired in past experiments and active testing, an algorithm selection strategy. Active testing attempts to iteratively identify an algorithm whose performance will most likely exceed the performance of previously tried algorithms. The novel method described in this paper uses tests on smaller data sample to rank the most promising candidates, thus optimizing the schedule of experiments to be carried out. The experimental results show that this approach leads to considerably faster algorithm selection.

Keywords: Algorithm selection, Meta-learning, Active testing, Algorithm Ranking

1 Introduction

A large number of data mining algorithms exist, rooted in the fields of machine learning, statistics, pattern recognition, artificial intelligence, and database systems, which are used to perform different data analysis tasks on large volumes of data. The task to recommend the most suitable algorithms has thus become rather challenging. Moreover, the problem is exacerbated by the fact that it is necessary to consider different combinations of parameter settings, or the constituents of composite methods such as ensembles.

The algorithm selection problem, originally described by Rice [17], has attracted a great deal of attention, as it endeavours to select and apply the best or near best algorithm(s) for a given task [4, 19]. The algorithm selection problem can be cast as a *learning* problem: the aim is to learn a model that captures the

relationship between the properties of the datasets, or meta-data, and the algorithms, in particular their performance. This model can then be used to predict the most suitable algorithm for a given new dataset.

This paper presents a new method, which builds on ranking approaches for algorithm selection [2, 3] in that it exploits meta-level information acquired in past experiments. Moreover, the proposed method combines this information with an algorithm selection strategy known as *active testing* [10, 11]. The aim of active testing is to iteratively select and evaluate a candidate algorithm whose performance will most likely exceed the performance of previously tested algorithms.

The method described in this paper differs from earlier approaches in various aspects. First, two methods presented earlier [10, 11] were combined, and adapted to optimize a multi-objective measure, called A3R, that combines predictive accuracy and time. The first method is known as an average ranking method, as it calculates an average ranking for all algorithms over all datasets. The upgrade here consists of using A3R as the measure to optimize, rather than simply accuracy. The second method uses fast sample-based tests instead of full cross-validation (CV) tests to identify the most promising candidates by evaluating them on small data samples. Again, we will use A3R, instead of accuracy, in the process of identifying the best competitor of the current alternative.

Fast sample-based tests allow selecting a good algorithm in less time, but are less reliable. This needs to be taken into account in the design of the method. One further contribution of this work is to show how the sample-based tests should be integrated with the elaboration of the ranking.

Finally, the experimental results are presented in the form of loss-time curves, where time is represented on a log scale. This representation is very useful in the evaluation of rankings representing schedules, as was shown in recent findings [18].

The remainder of this paper is organized as follows. In the next section we present an overview of work in related areas. Section 3 is dedicated to the description of our new method of active testing. We explain how it relates to earlier proposals. Section 4 presents the empirical evaluation of the newly proposed method. The final section presents conclusions and future work.

2 Related Work

In this paper we are addressing a particular case of the algorithm selection problem [17], oriented towards the selection of classification algorithms. Various researchers addressed this problem in the course of the last 25 years. One very common approach that could now be considered as "*the classical approach*" uses a set of measures to characterize datasets and establish their relationship to algorithm performance. This information is often referred to as *meta-data*.

The meta-data typically includes a set of simple measures, statistical measures, information-theoretic measures and/or the performance of simple algorithms referred to as landmarkers [4, 14, 19]. The aim is to obtain a model that

characterizes the relationship between the given meta-data and the performance of algorithms evaluated on these datasets. This model can then be used to predict the most suitable algorithm for a given new dataset, or alternatively, provide a ranking of algorithms, ordered by their suitability for the task at hand. Many studies conclude that ranking is in fact better, as it enables the user to iterative test the top candidates to identify the algorithms most suitable in practice. This strategy is sometimes referred to as the Top-N strategy [4].

The Top-N strategy has the disadvantage that it is unable to leverage what is learned from previous evaluations. For instance, if the top algorithm performs worse than expected, this may tell us something about the given dataset that can be used to update the ranking. Indeed, very similar algorithms are now also likely to perform worse than expected. This led researchers to investigate an alternative testing strategy, known as active testing [11]. This strategy intelligently selects the most useful cross-validation tests using the concept of relative landmarkers [7]. These landmarkers estimate the *relative probability* that a particular algorithm will outperform the current best candidate.

The method presented in [10] exploits partial learning curves created on small samples of the data, as suggested by the authors of [15]. It makes pairwise algorithm comparisons and represents the results in the form of a partially ordered ranking. The method can be evaluated by comparing the predicted partial order of algorithms to the actual partial order, representing the golden standard obtained by exhaustive testing. An extension to this method was presented in [11]. It relies on earlier performed cross-validation tests to calculate relative landmarkers. The authors showed that this led to better results than traditional top-N ranking strategies.

The novel aspect of the method described in this paper is the use of relatively fast sample-based tests to reorder the algorithms in the top-N ranking. Using tests on small data samples represents a trade-off in that they lead to less accurate performance estimation. Indeed, the tests carried out on small data samples are less reliable and thus the best algorithms may sometimes not be identified. Hence, it is difficult to say a priori which variant would be better. This is one of the motivations to investigate this issue.

Active testing is somewhat related to experiment design [6] and also to active learning. However, there has been relatively little work on active learning for algorithm selection. One notable exception is [12], who use the notion of *Expected Loss Optimization* (ELO). Another application in this area is [8], whose aim was to identify the most interesting substances for drug screening using a minimum number of tests. In these experiments, the authors have focused on the top-10 substances. Several different strategies were considered and evaluated.

Another notable active learning approach to meta-learning was presented in [16], where the authors use active learning to support the selection on informative examples. A prototype was implemented using the k-NN algorithm as a meta-learner and a certainty-based method was used for active learning. The prototype was evaluated in two different case studies, and the results obtained

by the active learning method were in general better than a random method for selecting meta-examples.

In this paper, we attribute particular importance to the tests on the new dataset. Our aim is to propose a way that minimizes the time before the best (or near best) algorithm is identified.

3 Active Sample-based Testing (ASbT) Method

We propose a new method, called *Active Sample-based Testing* (ASbT), which is detailed in Algorithm 1. It exploits meta-level information acquired from past experiments. This information includes the performance evaluations of various machine learning algorithms, on prior datasets, over multiple performance measures (e.g., accuracy, AUC, time). Moreover, we also use *data samples* or various sizes to evaluate algorithms. Further details concerning the meta-level information are provided below.

This information enables us to construct an average ranking of algorithms (line 4). The term *average ranking* is used here to indicate that it is averaged over all previously seen datasets. The average ranking can be followed by the user to identify the best performing algorithm, i.e., by performing a cross-validation test on all algorithms in the order of the ranking. This strategy was referred to as the Top-N strategy [2] and constitutes a baseline for other more competitive approaches.

Our method also exploits the active testing strategy [11]. However, in the ASbT approach, we use faster sample-based tests to identify competitive algorithms (line 5). In our experiments, these algorithms are also evaluated on the full dataset in order to construct a loss curve. To evaluate this method, experiments are carried out in a leave-one-out fashion. One dataset at a time is left out to evaluate our approach. The complete method, including evaluation, is summarized in Algorithm 1. The following sections discuss key parts of the method in more detail.

Algorithm 1 Active sample-based testing (ASbT)

1: Identify datasets D_s and algorithms
2: **for all** D_i in D_s **do**
3: {Leave-one-out cycle; D_i represents D_{new}}
4: Construct the average ranking
5: Carry-out sample-based active testing and evaluate recommended algorithms
6: Return a loss curve
7: **end for**
8: Aggregate the loss curves for all D_i in D_s
Return: Mean loss curve

3.1 Constructing the average ranking

The challenge here is to order the algorithms in the form of a top-N ranking. The underlying assumption is that the rankings obtained on past problems will transfer to new problems. Among many popular ranking criteria we find, for instance, *average ranks*, *success rates* and *significant wins* [3, 5, 10].

In this work we use average ranks, inspired by Friedman's *M statistic* [13]. For each dataset, the algorithms are ordered according to the performance measure chosen (e.g., predictive accuracy) and assigned ranks accordingly. The best algorithm is assigned rank 1, the runner-up is assigned rank 2, and so on. Let r_i^j be the rank of algorithm i on dataset j. The average rank for each algorithm is obtained using

$$r_i = \left(\sum_{j=1}^{D} r_i^j \right) \div D \tag{1}$$

where D is the number of datasets. The final ranking is obtained by ordering the average ranks and assigning ranks to the individual algorithms accordingly.

The average ranking is used here both as a baseline against which we can compare other methods, and as an input to the proposed algorithm (line 5), as discussed below.

3.2 Active Testing on Data Samples

Do we really need a full cross-validation test to establish a good sequence of algorithms to test? In this section we discuss a novel method that also follows an active testing strategy, but uses sample-based tests instead of full cross-validation tests. Hence, instead of deciding whether a candidate algorithm a_c is better than the currently best algorithm a_{best} using a full cross-validation test, we perform a test on a smaller sample of the new dataset. This is motivated by the fact that a sample-based test is much faster than a full cross-validation test.

However, as the sample-based test only yields a proxy for the actual performance, it may identify an apparent winner that differs from the one selected by the full cross-validation test. Hence, if a candidate algorithm beats the currently best algorithm on the sample-based test, we additionally carry out an evaluation using a full cross-validation test. This approach differs from [11] in three key aspects, i.e., the use of small data samples as proxies, the use of the A3R criterium that combines accuracy and run time (see below), and the strategy that we occasionally run a full cross-validation test. Algorithm 2 shows this method in detail.

3.3 Evaluating the returned ranking

To evaluate the quality of the returned ranking, the whole process is repeated in a leave-one-out fashion for all datasets D_i belonging to the set D_s (line 2 of Algorithm 1). For each generated algorithm ranking we generate a loss curve that

Algorithm 2 Active testing with sample-based tests

Require: Datasets D_i (representing D_{new}), D_s; Average ranking for D_i

1: Use the average ranking of D_i to identify the topmost algorithm and initialize a_{best};
Obtain the performance of a_{best} on dataset D_i using a full CV test;
2: Find the most promising competitor a_c of a_{best} using relative landmarkers
(previous test results on datasets)
3: Obtain the performance of a_c on new dataset using a sample-based test;
Record the accuracy and time of the test to compute A3R;
4: Compare the performance of a_c with a_{best};
use the winner as the current best algorithm a_{best};
If the current best algorithm has changed in the previous step, do:
Carry out evaluation of the new a_{best} using a full CV test
5: Repeat the whole process starting with step 2 until reaching a stopping criterion

Return: Loss curve

plots the loss in accuracy (see below) versus the time spent on the evaluation of the algorithms using full cross-validation tests. Finally, all individual loss curves are aggregated into a *mean loss curve*.

Loss-time curves In a typical loss curve, the x-axis represents the number of cross-validation tests and the y-axis shows the *loss* in accuracy with respect to the ideal ranking. Loss is defined to be the difference in accuracy between the current best evaluated classifier and the actual best classifier [11]. As tests proceed following the Top-N strategy, the loss either maintains its value, or decreases when the newly selected algorithm improved upon the previously selected algorithms. Each test represents the result of a full cross-validation evaluation on one dataset. However, some algorithms are much slower learners than others (sometimes by orders of magnitude), and these simple loss curves do not capture this.

This is why, in this article, we take into account the actual time required to evaluate each algorithm and update the loss curve accordingly. We will refer to this type of curve as a *loss versus time curve*, or *loss-time curve* for short.

As train/test times include both very small and very large numbers, it is natural to use the logarithm of the time, instead of the actual time. This has the effect that the same time intervals appear to be shorter as we shift further on along the time axis. This graphical representation is advantageous if we wish to give more importance to the initial items in the ranking.

To evaluate our algorithm (ASbT), we need to carry out tests on different datasets in a leave-one-out fashion and construct the mean loss-time curve by aggregating the individual loss-time curves.

3.4 A3R: combining Accuracy and Run time

In many situations, we have a preference for algorithms that are fast and also achieve high accuracy. However, the question is whether such a preference would

lead to better loss-time curves. To investigate this, we have adopted a multi-objective evaluation measure, A3R, described in [1], that combines both accuracy and time. The measure A3R is defined as:

$$A3R^{d_i}_{a_{ref},a_j} = \frac{\frac{SR^{d_i}_{a_j}}{SR^{d_i}_{a_{ref}}}}{\sqrt[N]{T^{d_i}_{a_j}/T^{d_i}_{a_{ref}}}} \quad (2)$$

Here, $SR^{d_i}_{a_j}$ and $SR^{d_i}_{a_{ref}}$ represent the *success rates* (accuracies) of algorithms a_j and a_{ref} on dataset d_i, where a_{ref} represents a given *reference algorithm*. Similarly, $T^{d_i}_{a_j}$ and $T^{d_i}_{a_{ref}}$ represent the run times of the algorithms, in seconds. To trade-off the importance of time, A3R includes the N^{th} root parameter. This is motivated by the observation that run times vary much more than accuracies. It is not uncommon that one particular algorithm is three orders of magnitude slower (or faster) than another one. Obviously, we do not want the time ratios to dominate the equation. If we take the N^{th} root of the run time, we will get a number that goes to 1 in the limit.

For instance, if we used $N = 256$, an algorithm that is 1,000 times slower would yield a denominator of 1.027. It would thus be equivalent to the faster reference algorithm only if its accuracy was 2.7% higher than the reference algorithm. Table 1 shows how a ratio of 1,000 (one algorithm is 1,000 times slower than the reference algorithm) is reduced for increasing values of N. As N gets higher, the time is given less and less importance.

Table 1. Effect of varying N on time ratio of 1000

C	$N = 2^C$	$1{,}000^{(1/N)}$	C	$N = 2^C$	$1{,}000^{(1/N)}$
0	1	1000.000	6	64	1.114
1	2	31.623	7	128	1.055
2	4	5.623	8	256	1.027
3	8	2.371	9	512	1.013
4	16	1.539	10	1,024	1.006
5	32	1.241	20	1,048,576	1.000

The performance measure A3R can be used to rank a given set of algorithms on a particular dataset in a similar way than using simply accuracy. Hence, the average rank method described earlier can easily be upgraded to generate a time-aware average ranking: the *A3R-based average ranking*. The method of active testing with sample-based tests can also be easily updated. We note that the algorithm shown in Algorithm 2 mentions performance on lines 3 and 4. If we use A3R instead of accuracy, we obtain a multi-objective variant of the active testing method.

Obviously, we can expect somewhat different results for each particular choice of N. Which value of N will lead to the best results in loss-time space? Another important question is whether the use of A3R is beneficial when compared to the approach that only uses accuracy. The answers to these questions will be answered empirically in the next section.

4 Experiments

This section describes the empirical evaluation of the proposed method. We have constructed a dataset from evaluation results retrieved from OpenML [20], a collaborative science platform for machine learning. This dataset contains the results of 53 parameterized learning algorithms from the Weka workbench [9] on 39 datasets[1]. Section 4.1 evaluates our proposed method independently of the A3R criterion, thus solely using accuracy. Section 4.2 explores the further impact of the A3R criterion, combining accuracy and run time.

4.1 Active Sample-based Testing using Accuracy

Figure 1 presents our results in the form of loss-time curves, with time represented on a log scale. First, it shows the loss curve of the average ranking method (AvgRank-ACC) which uses the Top-N strategy when carrying out the tests.

The figure also includes the loss-time curve of one predecessor method that uses active testing with full cross-validation tests (ATCV-ACC). This is ASbT in the extreme, using the full dataset instead of a smaller sample. In the earlier paper [11] this method was referred as AT0.

Finally, the figure shows the loss-time curves of three variants of the active sample-based testing method presented here. These variants use a different sample size for the sample-based testing. Here we use the notation ASbT-4-ACC to refer to a variant of active sample-based testing that uses accuracy on sample number 4 when conducting tests. Similarly, ASbT-5-ACC uses sample number 5 in testing. The sample sizes grow geometrically following the equation $2^{(5.5+0.5*n)}$, where n represents the sample number. So, the sizes of the first few samples, after rounding, are 64, 91, 128, 181, 256, 362 examples. Hence, sample number 4 includes 181 data points.

In Figure 1, $\log_{10}$ time is used on the x-axis. The values on the x-axis represent the time that has elapsed as we repeatedly conduct cross-validation tests.

What can we conclude from this figure? The first observation we can make is that all the three variants of the sample-based active testing methods compete quite well with the (more complex) predecessor method (ATCV-ACC) that uses full CV test to identify the best competitor. These two approaches beat the simple average ranking method (AvgRank-ACC). In Section 4.2, we investigate the effect of adopting the A3R measure instead of accuracy in the selection algorithms problem.

[1] Full details: `http://www.openml.org/project/tag/ActiveTestingSamples/u/1`

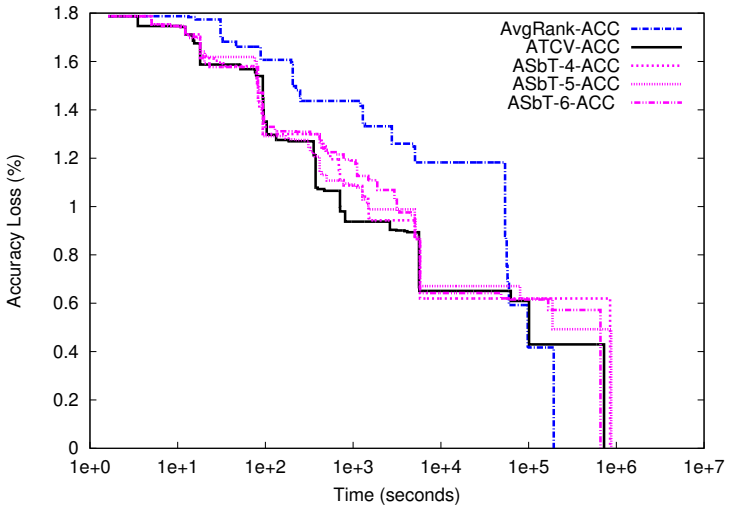

Fig. 1. Mean loss-time curves for 3 variants of ASbT (using sample number 4, 5 and 6), ACTV-ACC and the average ranking method (AvgRank-ACC).

Early stopping criteria The main active testing algorithm (ATCV) described in Algorithm 2 includes a stopping criterion: when the probability that a new candidate algorithm a_c will win over the currently best algorithm a_{best} becomes too small, it will not be considered. Since ASbT derives from ATCV, we can use the same criterion, and we have empirically evaluated the effect using ASbT-4. To do this, we need to define a minimal improvement in the sum of the relative landmarkers, which is a parameter of the method. This was set to 0.02 (2%) and the effect was that the algorithm stopped after about half of the algorithms were tested. The result of this study shows that it is only slightly better, but overall not much different. It does manage to skip less promising algorithms early on.

4.2 Active Sample-based Testing using Accuracy and Time

Our next aim in this paper is to analyze the effects of adopting A3R as a performance measure within the methods presented earlier. The first objective is to analyze the effects on the average ranking method. The second objective is to examine the effects of adopting A3R within the active testing strategy that uses sample-based tests. In each case we seek the best solution by tuning the value of parameter N of A3R. The third objective is to compare the proposed method with one predecessor method that generates a revised ranking first before conducting evaluation.

Figure 2 shows the first set of results. Note that the upgraded average ranking method (AvgRank-A3R) has an excellent performance when compared to the average ranking method that uses only accuracy as a measure (AvgRank-ACC).

Hence, the new average ranking method represents a new useful algorithm that can be exploited in practice for the selection of algorithms.

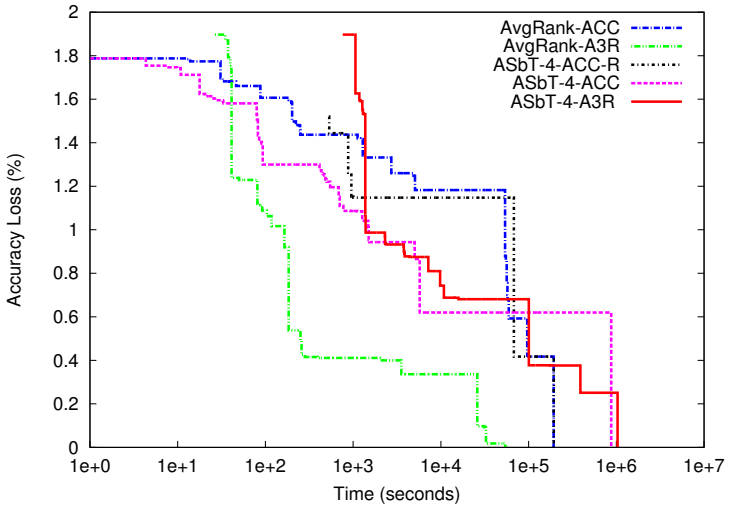

Fig. 2. Loss-time curves for sample-based active testing and average ranking methods, accuracy-based and A3R-based versions ($N = 256$).

Next, let us analyze the effects of adopting A3R within the active testing strategy that uses sample-based tests. As the experiments described in the previous section have shown that there is not much difference between using sample number 4, 5 or 6, we have simply opted for the variant that uses the smallest sample number 4 (ASbT-4-A3R).

We compare our new sample-based active testing method ASbT-4-A3R that uses A3R in the selection process with ASbT-4-ACC which represents the method that uses accuracy instead. The results show that our method ASbT-4-A3R does not beat the previous method (ASbT-4-ACC).

For completeness, we also include another variant, ASbT-4-ACC-R. This variant works by first re-ranking all algorithms by evaluating them on a small sample of the new dataset. Hence, it starts later than the other algorithms, on average 500 seconds later, because it needs to run this process first. Then, in a second phase, it conducts the final evaluation using full CV tests. Merging the two phases, as is done in ASbT-4-ACC, results in lower losses sooner.

The average ranking method (AvgRank-A3R), which represents a much simpler method than the sample-based variant, achieves surprisingly good results which warrants further investigation.

One possible explanation is that our meta-dataset is perhaps too simple, including 53 algorithms with default parameters. In future work we will investigate

the effect of adopting the A3R measure in the ASbT method, while using more algorithms (in the order of hundreds).

5 Conclusions

We have described two novel algorithm selection methods. The first method uses fast sample-based tests to identify the most promising candidates, as its aim is to intelligently select the next algorithm to test on the new dataset. The second method is a rather simple one. It calculates an average ranking for all algorithms, but uses A3R as the measure to rank on. The novelty here lies in the use of A3R, instead of just accuracy.

The experimental results are presented in the form of loss-time curves. Since exhaustive testing is not always practically feasible, we focus on the behavior of the algorithm within smaller time interval. This is achieved by using a loss curves with $\log_{10}$ of time on the x-axis. This representation stresses the losses at the beginning of the curve, corresponding to the initial tests.

The results show that both methods lead to considerable time savings, when compared to the previous approaches that exploited just accuracy. The experimental results suggest that the new version of the average ranking method represents a very good alternative that could be used in practice.

The next challenge then is to explore ways on how to improve the ASbT method that uses A3R in selecting the best competitor for active testing method by using more algorithms in the order of hundreds.

Acknowledgments This work has been funded by Federal Government of Nigeria Tertiary Education Trust Fund under the TETFund 2012 AST$D Intervention for Kano University of Science and Technology, Wudil, Kano State, Nigeria for PhD Overseas Training. This work was also partially funded by FCT/MEC through PIDDAC and ERDF/ON2 within project NORTE-07-0124-FEDER-000059 and through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT Portuguese Foundation for Science and Technology within project FCOMP-01-0124-FEDER-037281.We wish to thank Carlos Soares for his useful comments on the method presented in this paper.

References

1. Abdulrahman, S.M., Brazdil, P.: Measures for Combining Accuracy and Time for Meta-learning. In: Meta-Learning and Algorithm Selection Workshop at ECAI 2014. pp. 49–50 (2014)
2. Brazdil, P., Soares, C.: A Comparison of Ranking Methods for Classification Algorithm Selection. In: Machine Learning: ECML 2000, pp. 63–75. Springer (2000)
3. Brazdil, P., Soares, C., Da Costa, J.P.: Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. Machine Learning 50(3), 251–277 (2003)

4. Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning: Applications to data mining. Springer Science & Business Media (2008)
5. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. The Journal of Machine Learning Research 7, 1–30 (2006)
6. Fedorov, V.V.: Theory of Optimal Experiments. Academic Press (1972)
7. Fürnkranz, J., Petrak, J.: An Evaluation of Landmarking Variants. In: Working Notes of the ECML/PKDD 2000 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning. pp. 57–68 (2001)
8. de Grave, K., Ramon, J., de Raedt, L.: Active Learning for Primary Drug Screening. In: Benelearn 08, The Annual Belgian-Dutch Machine Learning Conference. vol. 2008, pp. 55–56 (2008)
9. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
10. Leite, R., Brazdil, P.: Active Testing Strategy to Predict the Best Classification Algorithm via Sampling and Metalearning. In: ECAI. pp. 309–314 (2010)
11. Leite, R., Brazdil, P., Vanschoren, J.: Selecting Classification Algorithms with Active Testing. In: Machine Learning and Data Mining in Pattern Recognition, pp. 117–131. Springer (2012)
12. Long, B., Chapelle, O., Zhang, Y., Chang, Y., Zheng, Z., Tseng, B.: Active Learning for Ranking through Expected Loss Optimization. In: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. pp. 267–274. ACM (2010)
13. Neave, H.R., Worthington, P.L.: Distribution-free Tests. Unwin Hyman London (1988)
14. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Tell me who can learn you and I can tell you who you are: Landmarking various learning algorithms. In: Proceedings of the 17th International Conference on Machine Learning. pp. 743–750 (2000)
15. Provost, F., Jensen, D., Oates, T.: Efficient Progressive Sampling. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 23–32. ACM (1999)
16. Prudencio, R.B., Ludermir, T.B.: Active Selection of Training Examples for Meta-Learning. In: Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on. pp. 126–131. IEEE (2007)
17. Rice, J.R.: The Algorithm Selection Problem. Advances in Computers 15, 65–118 (1976)
18. van Rijn, J.N., Abdulrahman, S.M., Brazdil, P., Vanschoren, J.: Fast Algorithm Selection using Learning Curves. In: Advances in Intelligent Data Analysis XIV. Springer (2015)
19. Smith-Miles, K.A.: Cross-disciplinary Perspectives on Meta-Learning for Algorithm Selection. ACM Computing Surveys (CSUR) 41(1), 6:1–6:25 (2008)
20. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. ACM SIGKDD Explorations Newsletter 15(2), 49–60 (2014)