Sharing RapidMiner workflows and experiments with OpenML

Jan N. van Rijn[1] and Joaquin Vanschoren[2]

[1] Leiden University, Leiden, Netherlands,
`j.n.van.rijn@liacs.leidenuniv.nl`
[2] Eindhoven University of Technology, Eindhoven, Netherlands,
`j.vanschoren@tue.nl`

Abstract. OpenML is an online, collaborative environment for machine learning where researchers and practitioners can share datasets, workflows and experiments. While it is integrated in several machine learning environments, it was not yet integrated into environments that offer a graphical interface to easily build and experiment with many data analysis workflows. In this work we introduce an integration into the popular RapidMiner environment, that will allow RapidMiner users to import data directly from OpenML and automatically share all their workflows and experiments. OpenML will then link these results to all other results obtained by other people, possibly with other tools, creating a single connected overview of the best workflows on a large set of machine learning problems. This is useful to learn and build on the results of others, to collaborate with many people online, and it provides a wealth of information to study how to construct workflows for new machine learning problems. We demonstrate the capabilities of this integration and identify several research opportunities.

Keywords: Meta Learning, Workflows, Algorithm Selection

1 Introduction

The field of meta-learning studies which Machine Learning algorithms work well on what kind of data. The algorithm selection problem is one of its most natural applications [24]: given a dataset, identify which learning algorithm (and which hyperparameter setting) performs best on it. Different approaches leverage meta-learning in different ways, such as building predictive meta-models based on data characterizations [3, 20], iteratively testing the most promising algorithms [17] and model-based hyperparameter optimization [8]. However, all these solutions focus on recommending just a single algorithm.

Even-though the obtained results are very useful, it has been widely recognized that the quality of the results can be markedly improved by also selecting the right pre-processing and post-processing operators [7, 19, 32]. For example, the quality of k Nearest Neighbour algorithms typically degrades when the number of features increases [9], so it makes sense to combine these algorithms with feature selection [14] or feature construction. The complete chain of pre-processing operators, algorithms and post-processing operators is typically referred to as a *workflow*.

Meta-learning research is built on the premise that the relation between data and a good algorithm can be learned. For this, a key prerequisite is to have access to a vast amount of executed experiments to serve as historical training data. Experiment databases [34] have been designed to collect previously executed experiments, divesting researchers from the burden of executing these many experiments over and over again. OpenML [26, 35] is an online platform for machine learning where researchers and practitioners can share datasets, workflows and experiments *automatically* from many machine learning environments, and build directly on each other's results. Hence, it provides an exceedingly rich resource for meta-learning research. However, it currently has limited support for workflows.

One of the additional challenges is the enormous increase of the search space: besides finding the right (order of) operators, each operator also has its own parameters to be tuned. Currently, there is only little work that addresses the question whether the relation between the dataset and the complete workflow of pre-processing, modelling and post-processing operators can be learned.

In order to foster progress in this challenging research area, we have integrated OpenML into RapidMiner. RapidMiner is a data analysis environment that has a graphical interface for users to easily experiment with many (slightly) different workflows, as well as support for generating machine learning workflows. By means of this integration, the full set of RapidMiner workflows can be shared on OpenML, as well as the ensuing experimental results as these workflows are run and evaluated on many input datasets. Collecting this information in an organized fashion will open up novel research opportunities in meta-learning.

2 Related Work

The algorithm selection problem has attracted a lot of attention. In meta-learning approaches, the data is characterised by so-called meta-features, over which a model can be built [3]. Significant effort has been devoted to creating these features, and they typically fall in one of the following categories [31]: statistical, information theoretic or landmarker [20].

Another way to find an appropriate algorithm is by subsampling the data: when training a algorithm on a small subset of the data, the time to execute and evaluate an algorithm is much lower. The underlying assumption is that if a algorithm works well on a small subsample of the data, it also works well on more data. Much research has been done to study which sample sizes and techniques are appropriate to obtain a reliable model [15, 21].

Even though these techniques work well, it has been correctly observed that learning curves do cross [16]. Some algorithms perform particularly well when trained on large amounts of data. In [25], a partial learning curve is built on small data samples to iteratively select the most promising algorithms in a time-constrained setting, showing significant time savings.

However, it usually doesn't suffice to recommend a single algorithm. Typically, the algorithm contains many hyperparameters that need to be tuned, and the model might benefit from certain pre-processing and post-processing operators. Many strategies have been proposed to optimise the hyperparameters of a given algorithm, including gradient decent methods [4], Bayesian optimization techniques [33] and genetic algorithms [13, 22]. However, most of these approaches do not leverage historical information on the performance of hyperparameter settings on previously seen problems. One simple way to do this is to use meta-learning to build a model that recommends parameter settings [30], or to view multiple algorithm configurations as individual algorithms [17]. Other research leverages meta-learning to recommend when certain hyperparameters should be tuned, or to predict a good initial parameter setting to speed up Bayesian optimization [8].

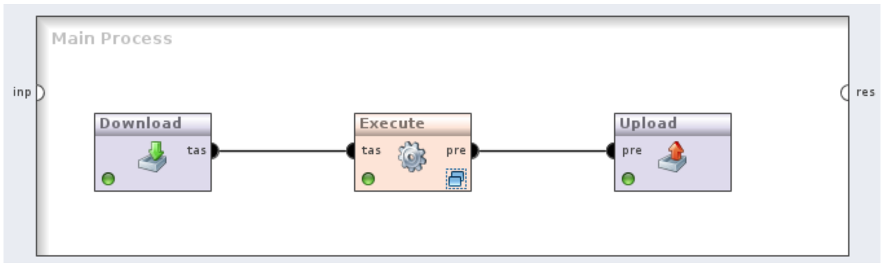
The task of selecting appropriate pre-processing and post-processing operators has been less studied in the literature. Case-based reasoning has been an early approach to select the most promising workflow out of a repository of previously successful workflows [10, 18]. Planning algorithms were also leveraged to construct and test possible workflows on the fly [1]. Most interestingly, the authors of [6, 12, 19] have independently from each other created a technique that exploits a meta-algorithm to predict what workflow to use. Their results suggests that the even the structure and operators of a workflow can be learned. In [7], the term *Full Model Selection Problem* was introduced, along with a particle swarm optimisation method to converge to a solution. The authors of [32] propose a framework in which these methods can be defined, along with particle swarm optimisation and genetic algorithm methods. Finally, graphical or other interfaces have emerged as a practical solution to manually construct and test many workflows, e.g. Weka [11], KNIME [2], ADAMS [23], and RapidMiner [28]. For a more complete overview of existing work in this area, see [29].

A common theme in this area is that a large body of workflows, and their evaluations on large numbers of datasets, is required for almost any of these methods to work well. In fact, it is often a limiting factor in demonstrating the practical applicability of these systems. By integrating OpenML and RapidMiner, it is our aim to work towards building the right infrastructure to foster large scale research in this direction.

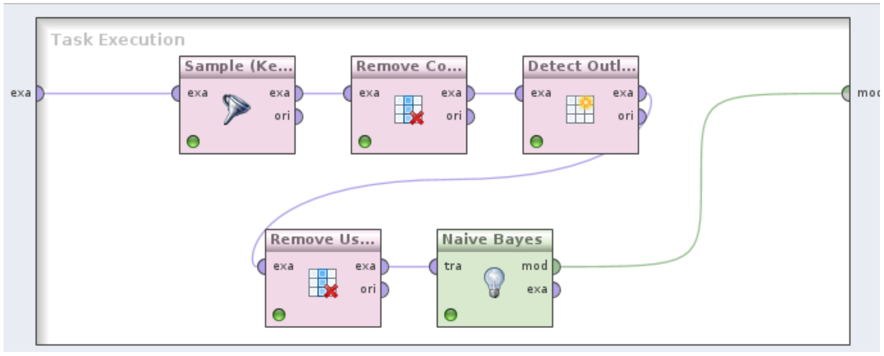
3 OpenML Connector

The integration[1] consists of three new RapidMiner operators: one for downloading OpenML tasks, one for executing them and one for uploading the results. Typically, they will be connected as shown in Figure 1(a). However, this modularization in three operators will likely be beneficial in some cases. The operators require an OpenML account to interact with the server.

[1] Available on `http://www.openml.org/`



(a) Main Workflow



(b) Subroutine solving OpenML task

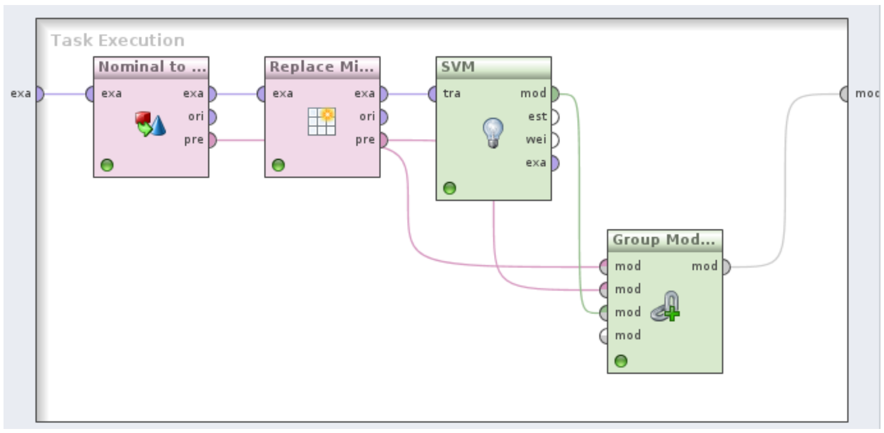
Fig. 1. Example of a RapidMiner workflow solving an OpenML task.

Download OpenML Task In order to make experiments reproducible, OpenML works with the concept of *tasks* [27, 35]. A task is a container that includes the input dataset(s), the data splits depending on the chosen evaluation procedure (e.g., cross-validation or holdout), and other necessary inputs. The "Download OpenML Task" operator downloads such tasks from OpenML and passes it to the output port.

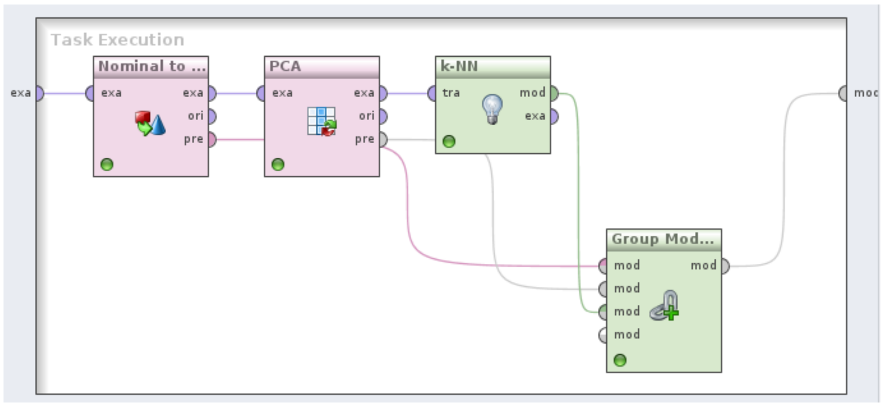
Execute OpenML Task The "Execute OpenML Task" is a so-called *super-operator*; it contains a sub-workflow that is expected to solve the task that is delivered at the input port. The subroutine is executed for each defined training set, and produces a model. This model is then used to predict the labels for the observations in the associated test set. An example of such a sub-workflow, including several pre-processing steps, is shown in Figure 1(b). The output of this super-operator is a data structure containing predictions for all instances in the test sets, and basic measurements such as run times.

Upload OpenML Task This operator uploads all relevant details of the workflow and the resulting predictions to OpenML. Details of the workflow are the set of all operators, and the parameter settings for each operators. The predictions contain the class label and confidences per class for classification tasks, or the predicted values for regression tasks. This enables OpenML to calculate all relevant performance measures, such as area under the ROC curve or RMSE. Also the run times for each fold are uploaded.

Example Workflows The main contribution of the RapidMiner plugin is that it automates the export of workflows to OpenML. Typically, a chain of operators is executed in order, possibly consisting of pre-processing operators that change the feature set. For example, the Support Vector Machine algorithm of RapidMiner can not operate on nominal features. In contrast to many WEKA algorithms, which automatically provide a workaround, in RapidMiner the work-

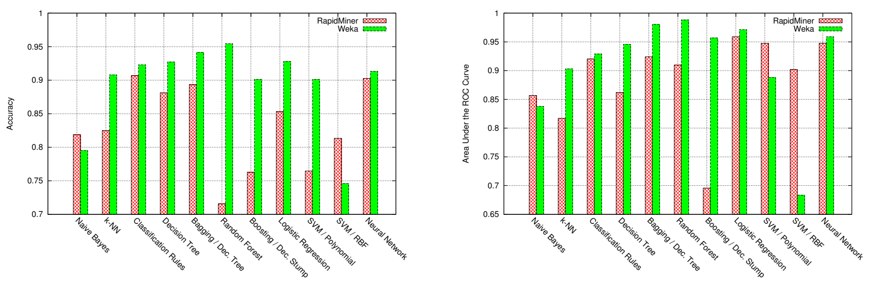


(a) SVM



(b) PCA / k-NN

Fig. 2. Subprocess of the Execute OpenML Task operator, combining various pre-processing models and the actual model using the “Group Models” operator.



(a) Accuracy (b) Area Under the ROC Curve

Fig. 3. Performance of RapidMiner and Weka algorithms on the "Spambase" dataset.

flow creator needs to define a solution. A possible solution could be to use the *Nominal to Numerical* operator. As this operator changes the feature set for each subsample of training set, the same pre-processing operations need to be performed on the test set. Figure 2 shows how the RapidMiner *Group Models* operator should be used to combine all pre-processing models and the algorithm model. This ensures that evaluation process is executed on the same features as constructed in the training set.

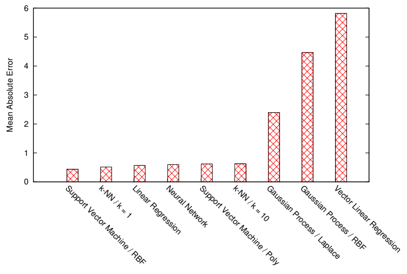
4 Research Opportunities

RapidMiner contains a large number of operators, covering a wide range of methods on various Machine Learning tasks. The OpenML integration thus opens up many research opportunities. We present some basic results obtained by using the RapidMiner plugin, and point out directions for future work.

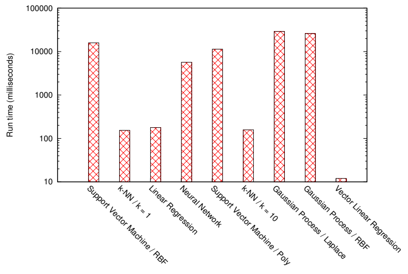
Decent Machine Learning research is conducted over a wide range of datasets. Yet, the results we present cover only a very small number of datasets. Therefore, no real conclusions can be drawn from these experiments, and interesting patterns should be addressed in future work.

Classification We can now easily compare the performance of various RapidMiner algorithms against similar algorithms implemented in a different workbench, for example WEKA, because evaluations of other workbenches are already available on OpenML. Figure 3 shows the result of 11 algorithm implementations on the "spambase" dataset.All algorithms are executed with their respective default parameter settings. In the case of ensembles the base-algorithm is denoted; in case of Support Vector Machines, the kernel was denoted.

One surprising observation is that performance differs significantly between different implementations of the same basic algorithms. For example, even a fairly simple algorithm like Naive Bayes does not yield the same performance in both workbenches. In particular the difference in performance of the Random



(a) Mean Absolute Error



(b) Run time

Fig. 4. Performance of regression algorithms on the "Wine Quality" dataset.

Forest and Boosting implementations of WEKA and RapidMiner is striking. Figure 3(a) shows the predictive accuracy of the algorithms. The results suggest that most WEKA algorithms are superior to their RapidMiner equivalents in terms of predictive accuracy. However, when measuring the Area Under the ROC Curve, most RapidMiner algorithms perform somewhat better, see for example the Support Vector Machines as shown in Figure 3(b).

The fact that two implementations of the same algorithm yield very different results can have various reasons. For example, different implementations can handle missing values differently, e.g., by replacing missing values by the mean of that attribute, or removing all observations that contain missing values.[2] If there is no parameter to control this behaviour, important aspects of model building are hidden from us. Finding these differences can therefore help us understand what kind of pre-processing steps are important for certain algorithms. Another possible explanation for why these results differ may be that the default parameter settings of the different implementations were optimized differently.

Regression RapidMiner also contains many algorithms than can perform regression tasks. In the next setup, we run some of these on the "Wine Quality" dataset [5].Figure 4 shows some results.

Figure 4(a) shows the *Mean Absolute Error* of all regression algorithms. There is a large group of algorithms with equivalent performance. Three algorithm perform eminently worse. Figure 4(b) shows run times. There seems no direct relation between good performance and higher run times.

It seems reasonable to assume that for regression tasks pre-processing steps become even more important, as many algorithms do not natively deal with irrelevant features (e.g., k-NN) or nominal values (e.g., Support Vector Machines). Fairly simple workflows can already make a big difference for regression tasks.

Learning Curves When running a algorithm on samples of increasing size, a learning curve can be constructed. These can be used to perform algorithm

[2] Note that the Spambase dataset used in Figure 3 has no missing features.

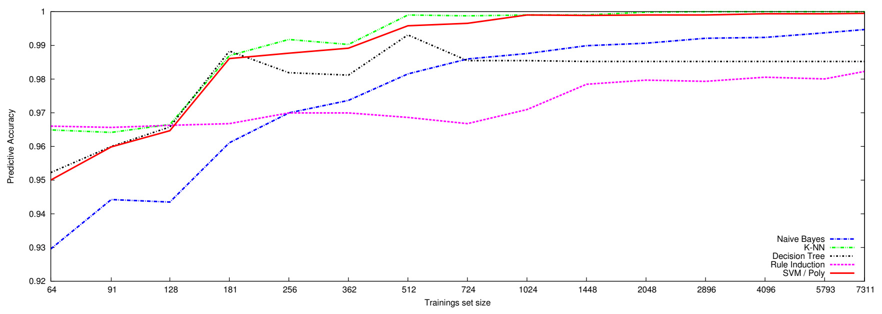


Fig. 5. Learning Curves of RapidMiner algorithms on the "Mushroom" dataset.

selection, as is done in [16, 25]. In [21] algorithm selection is used done by running algorithms on small samples of the data. The RapidMiner plugin can also operate on these data samples, and supports the creation of learning curves.

Figure 5 shows some of these curves. The x-axis shows the size of the sample, and the y-axis shows the performance on each sample. The curves behave as expected. In most cases, a larger sample size results in a higher accuracy. Furthermore, the curves do cross occasionally. An algorithm that performs well on a small sample is not necessarily competitive on larger samples, e.g., Rule Induction.

The Pairwise Comparison method described in [25] selects algorithms based on their performance on small samples. Enabling this method to operate on workflows would be a non-trivial but very useful extension.

Full Model Selection The authors of [7] first introduced this term, describing it as: given a set of pre-processing methods, feature selection algorithms and algorithms, select the combination of these that obtains the highest predictive accuracy for a given data set. The authors of [32] developed an uniform framework for solving this problem, and also came up with a competitive algorithm based on genetic algorithms. Workflows constructed in RapidMiner seem well fitted for this application, and the plugin introduced here could help with the experimentation and exploitation of such techniques.

Workflow Mining The authors of [19] propose a data mining advisor that attempts to extract knowledge from previously ran workflows to build new ones. Having access to a large repository of executed workflows gives the possibility of extracting knowledge about which components work well in combination with each other. By collecting a large set of workflow results in OpenML similar experiments can be conducted on large scale.

5 Conclusions

We have developed and presented an integration of the OpenML online collaboration platform within the RapidMiner workbench. OpenML currently contains over half a million experiments, yet few of those cover complex workflows. Recent work in meta-learning suggests that pre-processing and post-processing operators are an important part of successful algorithm selection solutions. It seems a logical next step to stimulate meta-learning and algorithm selection research on complete Machine Learning workflows, and this plugin is meant to enable and foster the automated sharing and collection of experiments that explore the performance of many possible workflows on many machine learning problems. Several opportunities for ongoing and future research have been identified, which can be pursued on an unprecedented scale through OpenML. We also hope to stimulate the integration of OpenML into other workbenches that deal with complex machine learning workflows.

Acknowledgments This work is supported by grant 600.065.120.12N150 from the Dutch Fund for Scientific Research (NWO).

References

1. Bernstein, A., Provost, F., Hill, S.: Toward Intelligent Assistance for a Data Mining Process: An Ontology-Based Approach for Cost-Sensitive Classification. Knowledge and Data Engineering, IEEE Transactions on 17(4), 503–518 (2005)
2. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz Information Miner. In: Data Analysis, Machine Learning and Applications. pp. 319–326. Springer Berlin Heidelberg (2008)
3. Brazdil, P., Gama, J., Henery, B.: Characterizing the Applicability of Classification Algorithms using Meta-Level Learning. In: Machine Learning: ECML-94. pp. 83–102. Springer (1994)
4. Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing Multiple Parameters for Support Vector Machines. Machine Learning 46(1-3), 131–159 (2002)
5. Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J.: Modeling wine preferences by data mining from physicochemical properties. Decision Support Systems 47(4), 547–553 (2009)
6. Diamantini, C., Potena, D., Storti, E.: Mining Usage Patterns from a Repository of Scientific Workflows. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. pp. 152–157. ACM (2012)
7. Escalante, H.J., Montes, M., Sucar, L.E.: Particle Swarm Model Selection. The Journal of Machine Learning Research 10, 405–440 (2009)
8. Feurer, M., Springenberg, T., Hutter, F.: Initializing Bayesian Hyperparameter Optimization via Meta-Learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. pp. 1128–1135 (2015)
9. Friedman, J.H.: On Bias, Variance, 0/1-Loss, and the Curse-of-Dimensionality. Data Mining and Knowledge Discovery 1(1), 55–77 (1997)

10. Goble, C.A., De Roure, D.C.: myExperiment: Social Networking for Workflow-using e-Scientists. In: Proceedings of the 2nd workshop on Workflows in support of large-scale science. pp. 1–2. ACM (2007)
11. 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)
12. Hilario, M., Nguyen, P., Do, H., Woznica, A., Kalousis, A.: Ontology-Based Meta-Mining of Knowledge Discovery Workflows. In: Meta-Learning in Computational Intelligence, pp. 273–315. Springer (2011)
13. Huang, C.L., Wang, C.J.: A GA-based feature selection and parameters optimization for support vector machines. Expert Systems with applications 31(2), 231–240 (2006)
14. Jain, A., Zongker, D.: Feature Selection: Evaluation, Application, and Small Sample Performance. Pattern Analysis and Machine Intelligence, IEEE Transactions on 19(2), 153–158 (1997)
15. John, G.H., Langley, P.: Static Versus Dynamic Sampling for Data Mining. In: In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 367–370. AAAI Press (1996)
16. Leite, R., Brazdil, P.: Predicting Relative Performance of Classifiers from Samples. In: Proceedings of the 22nd International Conference on Machine Learning. pp. 497–503. ACM (2005)
17. 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)
18. Morik, K., Scholz, M.: The MiningMart Approach to Knowledge Discovery in Databases. In: Intelligent Technologies for Information Analysis, pp. 47–65. Springer (2004)
19. Nguyen, P., Hilario, M., Kalousis, A.: Using Meta-mining to Support Data Mining Workflow Planning and Optimization. Journal of Artificial Intelligence Research 51, 605–644 (2014)
20. 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)
21. 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)
22. Reif, M., Shafait, F., Dengel, A.: Meta-learning for evolutionary parameter optimization of classifiers. Machine learning 87(3), 357–380 (2012)
23. Reutemann, P., Vanschoren, J.: Scientific workflow management with ADAMS. In: Machine Learning and Knowledge Discovery in Databases, pp. 833–837. Springer (2012)
24. Rice, J.R.: The Algorithm Selection Problem. Advances in Computers 15, 65118 (1976)
25. 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)
26. van Rijn, J.N., Bischl, B., Torgo, L., Gao, B., Umaashankar, V., Fischer, S., Winter, P., Wiswedel, B., Berthold, M.R., Vanschoren, J.: OpenML: A Collaborative Science Platform. In: Machine Learning and Knowledge Discovery in Databases, pp. 645–649. Springer (2013)

27. van Rijn, J.N., Umaashankar, V., Fischer, S., Bischl, B., Torgo, L., Gao, B., Winter, P., Wiswedel, B., Berthold, M.R., Vanschoren, J.: A Rapidminer extension for Open Machine Learning. In: RCOMM 2013. pp. 59–70 (2013)
28. Ritthoff, O., Klinkenberg, R., Fischer, S., Mierswa, I., Felske, S.: Yale: Yet another learning environment. In: LLWA 01-Tagungsband der GI-Workshop-Woche, Dortmund, Germany. pp. 84–92 (2001)
29. Serban, F., Vanschoren, J., Kietz, J.U., Bernstein, A.: A Survey of Intelligent Assistants for Data Analysis. ACM Computing Surveys (CSUR) 45(3), 31:1–31:35 (2013)
30. Soares, C., Brazdil, P., Kuba, P.: A Meta-Learning Method to Select the Kernel Width in Support Vector Regression. Machine Learning 54(3), 195–209 (2004)
31. Sun, Q., Pfahringer, B.: Pairwise meta-rules for better meta-learning-based algorithm ranking. Machine Learning 93(1), 141–161 (2013)
32. Sun, Q., Pfahringer, B., Mayo, M.: Towards a Framework for Designing Full Model Selection and Optimization Systems. In: Multiple Classifier Systems, pp. 259–270. Springer (2013)
33. Thornton, C., Hutter, F., Hoos, H., Leyton-Brown, K.: Auto-WEKA: Combined selection and Hyperparameter Optimization of Classification Algorithms. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 847–855. ACM (2013)
34. Vanschoren, J., Blockeel, H., Pfahringer, B., Holmes, G.: Experiment databases. A new way to share, organize and learn from experiments. Machine Learning 87(2), 127–158 (2012)
35. 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)