6th ICML Workshop on Automated Machine Learning (2019)

An Open Source AutoML Benchmark

Pieter Gijsbers[1*] P.GIJSBERS@TUE.NL
Erin LeDell[2*] ERIN@H2O.AI
Janek Thomas[3*] JANEK.THOMAS@STAT.UNI-MUENCHEN.DE
Sébastien Poirier[2*] SEBASTIEN@H2O.AI
Bernd Bischl[3] BERND.BISCHL@STAT.UNI-MUENCHEN.DE
Joaquin Vanschoren[1] J.VANSCHOREN@TUE.NL
[*] *The four authors contributed equally to the paper.*
[1] *Eindhoven University of Technology, Netherlands*
[2] *H2O.ai, United States*
[3] *Ludwig-Maximilians-University Munich, Germany*

Abstract

In recent years, an active field of research has developed around automated machine learning (AutoML). Unfortunately, comparing different AutoML systems is hard and often done incorrectly. We introduce an open, ongoing, and extensible benchmark framework which follows best practices and avoids common mistakes. The framework is open-source, uses public datasets and has a website with up-to-date results. We use the framework to conduct a thorough comparison of 4 AutoML systems across 39 datasets and analyze the results.

1. Introduction

Designing and tuning machine learning systems is a labor and time intensive task which requires extensive expertise. The field of automated machine learning (AutoML) is focused on automating this task. AutoML tools allow novice users to create useful machine learning models, while experts can use them to free up valuable time for other tasks. In recent years, many approaches have been developed for building and optimizing model learning pipelines, or building and optimizing deep neural networks. This paper focuses on the former.

The Need for Standardized Benchmarks

There is no universally best AutoML approach. Hence, we need comparisons to help practitioners select the right tools, and provide objective feedback to the research community. Unfortunately, many current comparisons are lacking in several ways. The selection of datasets is often too limited in scope, typically reusing the same (mostly small) datasets used in comparisons many years ago (Thornton et al., 2013). This increases the possibility of overfitting on a specific set of datasets, creates a bias towards older datasets rather than current challenges, and may fail to show individual tool's strengths or weaknesses. Authors may even knowingly or unknowingly select datasets on which current systems perform well. Finally, 'rival' methods may not have been run correctly, for instance by misunderstanding memory management and/or using insufficient compute resources (Balaji and Allen, 2018).

Tool	Back-end	Optimization	Meta-learning	Post-processing
Auto-WEKA	WEKA	Bayesian	-	-
auto-sklearn	scikit-learn	Bayesian	warm-start	ensemble selection
TPOT	scikit-learn	Genetic Programming	-	-
H2O AutoML	H2O	Random Search	-	stacked ensembles

Table 1: Simplified comparison of a selection of AutoML tools.

A New Hope

In this work, we present an open, extensible and ongoing AutoML benchmark to address these problems. The benchmark is completely *open source*[1], and allows anyone to *extend* it by adding or updating AutoML systems through pull requests. Finally, it is *ongoing* because we will update it with new benchmark datasets, run the experiments again when AutoML tools have substantial version updates. The benchmark is accompanied by a website which will show the latest results and other information.[2]

2. Related Literature

AutoML methods differ in their optimization method (e.g. Bayesian Optimization or Genetic Programming), the pipelines they generate (e.g. with or without fixed structure), the library of algorithms they select from, whether they use meta-learning to learn from runs on prior datasets, or whether they perform post-processing (e.g. ensemble construction).

Table 1 shows a simplified comparison of the AutoML tools compared in this paper. The first prominent AutoML tool was **Auto-WEKA** (Thornton et al., 2013), which used Bayesian optimization to select and tune the algorithms in a machine learning pipeline based on WEKA (Hall et al., 2009). **auto-sklearn** (Feurer et al., 2015) did the same using scikit-learn (Pedregosa et al., 2011) and added meta-learning to *warm-start* the search with the best pipelines on similar datasets, as well as ensemble construction. **TPOT** (Olson et al., 2016) optimizes scikit-learn pipelines via genetic programming, starting with simple ones and evolving them over generations. Finally, **H2O AutoML** (H2O.ai, 2017) optimizes H2O components by stacking the best solutions found by a random search.

Notable omissions from this list include autoxgboost (Thomas et al., 2018), which leverages Bayesian optimization to optimize gradient boosting models, OBOE (Yang et al., 2018), which uses low-rank approximation to predict the best pipelines, ML-Plan (Mohr et al., 2018), which optimizes WEKA-based pipelines using hierarchical planning, and Hyperband (Li et al., 2016), a bandit-based approach which aggressively selects configurations based on performance on subsamples of data. We do plan to include these in the next version of the benchmark. In some cases, we ran into technical issues and we are in contact with the authors to resolve them. There are also several AutoML systems which were sadly not yet open sourced at the time of writing, yet we hope that their authors will add them to the benchmark in the near future.

1. `https://github.com/openml/automlbenchmark/`
2. `https://openml.github.io/automlbenchmark/`

Prior efforts to create systematic AutoML benchmarks were sadly obfuscated by memory management and evaluation setup issues and lack strong baselines to interpret the results (Balaji and Allen, 2018). We instead opted to build an open-source framework in dialogue with the AutoML framework developers to ensure the tools were properly used. While we evaluate on fewer datasets, they are about 15 times larger on average. We followed best practices on how to construct and run good machine learning benchmarks (Bischl et al., 2017a) and general-purpose algorithm configuration libraries (Bischl et al., 2017b), but also extend them since evaluating AutoML systems comes with specific challenges.

3. Benchmark Design

Each benchmark task consists of a dataset, a metric to optimize, and specific resources to use. We will briefly explain our choice for each. More details are available on the website.

Datasets We selected 39 classification datasets from previous AutoML papers (Thornton et al., 2013), competitions (Isabelle et al., 2016), and machine learning benchmarks (Bischl et al., 2017a)[3], according to a predefined list of criteria.[4] To study the differences between AutoML systems, the datasets vary in the number of samples and features by orders of magnitude, and vary in the occurrence of numeric features, categorical features and missing values. We excluded datasets which were too easily solved with AutoML or did not represent typical AutoML scenario's (e.g. artificial datasets). The *current* list of datasets is available on OpenML (Vanschoren et al., 2014).[5] In line with our goal to regularly update the benchmark and avoid overfitting on one static set, we set aside some datasets for inclusion in the near future. We would also like to include even bigger datasets in subsequent versions, as well as regression datasets, currently omitted because of computational constraints.

Performance metrics The benchmark can be run with a wide range of measures at the user's discretion. For the results in this paper, area under the receiver operator curve (AUROC) is used for binary classification problems and log loss is used for multi-class classification problems[6], mainly because these are insightful, commonly used and supported by most AutoML tools. It is imperative that AutoML system optimize for the same metric they are evaluated on. The measures are estimated with ten-fold cross-validation.

Hardware choice and resource specifications To improve reproducibility and extensibilily, we opted to use standard *m5.2xlarge* instances available on Amazon Web Services (AWS).[7] These represent commodity level hardware which also keep the cost of running the benchmark down. It is not required to run the benchmark on AWS however, as the benchmark can also be run locally directly on the machine or from a docker container.

Frameworks and their configuration We required all AutoML tools to be open source. We selected the current set of tools based on popularity, ease of use and variety of underlying techniques. We do plan to include more tools in future work and encourage all developers

3. OpenML CC-18: `https://openml.github.io/OpenML/benchmark`
4. For more details, see: `https://openml.github.io/automlbenchmark/benchmark_datasets.html`
5. Study for this benchmark: `https://www.openml.org/s/218`
6. We use the implementations provided by scikit-learn 0.20
7. 32 GB memory, 8 vCPUs (Intel Xeon Platinum 8000 series Skylake-SP processor with a sustained all core Turbo CPU clock speed of up to 3.1 GHz). `https://aws.amazon.com/ec2/instance-types/m5/`

to add their AutoML tool to the benchmark framework. Baseline methods include a constant predictor, which always predicts the class prior, an *untuned* Random Forest[8], and a *tuned* Random Forest for which up to eleven unique values of *max_features* are evaluated with cross-validation (as time permits), and evaluated by refitting the final model with the optimal *max_features* values.

The AutoML tools were all used with their default hyperparameter values and search spaces, since most users will use them in this way. The exception are hyperparameters which specified available resources, which were fixed to a specific number of cores, memory and total runtime. This was done to allow a more practical comparison, and because it is practically impossible to homogenize the search spaces for each tool. It is important to realize that no two tools share the exact same search space or optimization method, so from this benchmark no conclusions can be drawn about those.

Open-source, extensible framework structure. In developing the benchmark framework, we made sure that it is easily extensible with new frameworks or datasets. New AutoML tools can be easily wrapped and included: each of the current tools required less than 100 lines of wrapper code. Adding a dataset that is hosted on OpenML only takes 3 lines of code. New additions are not evaluated automatically.

Meta-learning AutoML frameworks may use meta-learning to learn about good configurations across datasets. An AutoML framework which used datasets of the benchmark in its meta-learning process will have an unfair advantage on them. We did not decide how to resolve this issue, and leave it as future work.[9] From our selection of frameworks, only auto-sklearn uses meta-learning, and we indicate affected datasets in the results.

4. Results

We ran two benchmarks, using a time budget of 1 and 4 hours per fold respectively, for a total of around 8000 hours of computation time. The 4h 'raw' results are visualized in Figure 1. Those results are very similar to the 1h ones, bringing only slight score improvements for some frameworks, especially TPOT. Auto-WEKA is showing signs of overfitting when running longer, especially on multi-class problems. There is no AutoML system which consistently outperforms all others. On some datasets, the performance differences can be significant, but on others the AutoML methods are only marginally better than a Random Forest. The variance of the per-fold scores can be quite large. On the 'dionis' and 'helena' datasets, all frameworks perform worse than a Random Forest. Both have more more than 100, quite unbalanced classes, which seems to be a weak spot for current AutoML techniques, at least under log loss.

Because the scores vary across tasks, we also normalized them such that the constant predictor is 0 and the tuned random forest is 1. Shown in Table 2, these scores reflect the relative improvement over our strongest baseline, with a score greater than one being better than the strongest baseline. Generally, these scores are very similar across methods, all being relatively close to the tuned random forest baseline. None of the AutoML systems outperforms an untuned random forest across all problems, though in most cases they are

8. Unless specified otherwise, Random Forests were built with 2000 estimators and scikit-learn 0.20 defaults.
9. Take a look or join the discussion at `https://github.com/openml/automlbenchmark/issues/18`

better than a tuned random forest. Auto-WEKA has the poorest performance out of the tested AutoML packages under the tested conditions. Note that these results were obtained using a rather generous time budget. We hope to add anytime performance evaluation curves in the future, but this is not yet supported by many of the AutoML tools.

Finally, we can observe that on some datasets, some AutoML tools perform significantly better or worse than others. At the moment, we can't draw clear conclusions about which data properties explain this behavior beyond what we observed above. We aim to study this further by including more datasets.

Figure 1: Scores obtained on each dataset by each framework on each of ten folds. On the left are binary classification problems with their AUROC scores, on the right are multi-class classification problems with logloss. Opaque diamonds represent the average score across all folds.

Framework: Binary tasks:	auto-sklearn	Auto-WEKA	H2O AutoML	RandomForest	TPOT
adult	1.045	1.000	1.049	1.000	1.048
airlines	1.403	1.016	1.435	0.997	1.343
albert	1.009		1.115	1.001	0.981
amazon_employee...	0.972*	0.886	1.048	1.003	1.012
apsfailure	1.000	0.985	1.001	1.000	1.001
australian	1.010	1.015	0.909	1.010	1.011
bank-marketing	1.012	0.950	1.015	1.000	1.008
blood-transfusion	1.495	1.379	1.532	0.985	1.149
christine	1.072	0.998	1.048	0.988	1.029
credit-g	0.970*	0.829	0.991	1.004	0.924
guiellermo	1.004	0.934	1.024	0.999	0.878
higgs	1.018*	0.845	1.041	0.999	1.005
jasmine	0.987	0.939	1.001	0.998	1.004
kc1	0.999*	0.934	0.992	0.987	1.013
kddcup09_appetency	1.181*	1.043	1.176	1.016	1.134
kr-vs-kp	1.000*	0.959	1.000	0.999	0.999
miniboone	1.008	0.957	1.010	0.999	1.001
nomao	1.002	0.973	1.002	1.000	1.001
numerai28.6	1.679	1.544	1.730	1.042	1.428
phoneme	0.993*	0.998	1.005	1.000	1.015
riccardo	1.000	0.996	1.000	0.999	0.992
sylvine	1.013	0.985	1.011	0.999	1.023
Multi-class tasks:					
car	1.030	0.906	1.060	0.878	1.060
cnae-9	1.069	0.541	1.076	0.999	1.057
connect-4	1.184	-1.565	1.409	0.954	1.276
covertype	0.976	-0.361	0.856	0.944	0.933
dilbert	1.182	0.459	1.205	0.979	1.111
dionis	0.580	0.590		1.002	
fabert	1.026	-5.235	1.049	1.004	1.005
fashion-mnist	0.995	0.717	1.052	0.993	0.841
helena	0.660	-18.420	1.905	0.970	1.676
jannis	1.083	-1.989	1.065	0.973	0.987
jungle_chess...	1.299	-3.309	1.235	0.933	1.459
mfeat-factors	1.059*	0.789	1.053	0.992	1.018
robert	-0.001		1.545	1.000	0.640
segment	1.004	0.808	1.012	0.992	1.008
shuttle	1.000	0.979	1.000	1.000	1.000
vehicle	1.102	-4.630	1.166	0.986	1.099
volkert	1.002	-5.585	1.111	0.954	0.945

Table 2: Performance of AutoML frameworks, scaled between a constant class prior predictor (=0) and a tuned random forest (= 1). Missing values mean that no results were returned in time. *: the task was also included in meta-learning models.

5. Conclusion

We presented a novel benchmark for AutoML frameworks which is open-source, extensible both in terms of AutoML frameworks and tasks, and ongoing, publishing all the latest results online. Current results already highlight several avenues for further AutoML research. On some datasets, none of the frameworks outperforms a Random Forest within 4 hours, and high-dimensional or highly multi-class problems are often challenging. In future work, we will include more frameworks and tasks, especially larger datasets and regression tasks.

Acknowledgements

Pieter Gijsbers would like to acknowledge funding by the Data Driven Discovery of Models (D3M) program run by DARPA and the Air Force Research Laboratory.

References

A. Balaji and A. Allen. Benchmarking automatic machine learning frameworks. *CoRR*, abs/1808.06492, 2018. URL `http://arxiv.org/abs/1808.06492`.

B. Bischl, G. Casalicchio, M. Feurer, F. Hutter, M. Lang, R.G. Mantovani, J.N. van Rijn, and J. Vanschoren. Openml benchmarking suites. *arXiv preprint arXiv:1708.03731*, 2017a.

B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Frechtte, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney, and J. Vanschoren. Aslib: A benchmark library for algorithm selection. *arXiv preprint arXiv:1708.03731*, 2017b.

M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, pages 2962–2970, 2015.

H2O.ai. *H2O AutoML*, August 2017. URL `http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html`. H2O version 3.14.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656278. URL `http://doi.acm.org/10.1145/1656274.1656278`.

G. Isabelle, C. Imad, H.J. Escalante, S. Escalera, D. Jajetic, J.R. Lloyd, N. Maci, B. Ray, l. Romaszko, M. Sebag, A. Statnikov, S. Treguer, and E. Viegas. A brief review of the chalearn automl challenge: Any-time any-dataset learning without human intervention. In *Proceedings of the Workshop on Automatic Machine Learning*, volume 64 of *Proceedings of Machine Learning Research*, pages 21–30, New York, New York, USA, 24 Jun 2016. PMLR. URL `http://proceedings.mlr.press/v64/guyon_review_2016.html`.

L. Li, K.G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Efficient hyperparameter optimization and infinitely many armed bandits. *CoRR*, abs/1603.06560, 2016. URL `http://arxiv.org/abs/1603.06560`.

F. Mohr, M. Wever, and E. Hüllermeier. Ml-plan: Automated machine learning via hierarchical planning. *Machine Learning*, 107(8):1495–1515, Sep 2018. ISSN 1573-0565. doi: 10.1007/s10994-018-5735-z. URL `https://doi.org/10.1007/s10994-018-5735-z`.

R.S. Olson, R.J. Urbanowicz, P.C. Andrews, N.A. Lavender, L.C. Kidd, and J.H. Moore. *Applications of Evolutionary Computation: 19th European Conference, EvoApplications*

2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I, chapter Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, pages 123–137. Springer International Publishing, 2016. ISBN 978-3-319-31204-0. doi: 10.1007/978-3-319-31204-0_9. URL http://dx.doi.org/10.1007/978-3-319-31204-0_9.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

J. Thomas, S. Coors, and B. Bischl. Automatic gradient boosting. In *International Workshop on Automatic Machine Learning at ICML*, 2018.

C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proc. of KDD-2013*, pages 847–855, 2013.

J. Vanschoren, J. van Rijn, B. Bischl, and L. Torgo. OpenML: Networked science in machine learning. *SIGKDD Explor. Newsl.*, 15(2):49–60, 2014.

C. Yang, Y. Akimoto, D.W. Kim, and M. Udell. OBOE: collaborative filtering for automl initialization. *CoRR*, abs/1808.03233, 2018. URL http://arxiv.org/abs/1808.03233.