7th ICML Workshop on Automated Machine Learning (2020)

H2O AutoML: Scalable Automatic Machine Learning

E. LeDell ERIN@H2O.AI
H2O.ai, USA

S. Poirier SEBASTIEN@H2O.AI
H2O.ai, USA

Abstract

H2O is an open source, distributed machine learning platform designed to scale to very large datasets, with APIs in R, Python, Java and Scala. We present H2O AutoML, a highly scalable, fully-automated, supervised learning algorithm which automates the process of training a large selection of candidate models and stacked ensembles within a single function. The result of the AutoML run is a "leaderboard": a ranked list of models, all of which can be easily exported for use in a production environment. Models in the leaderboard can be ranked by numerous model performance metrics or other model attributes such as training time or average per-row prediction speed.

The H2O AutoML algorithm relies on the efficient training of H2O machine learning algorithms to produce a large number of models in a short amount of time. H2O AutoML uses a combination of fast random search and stacked ensembles to achieve results competitive with, and often better than, other frameworks which rely on more complex model tuning techniques such as Bayesian optimization or genetic algorithms. H2O AutoML trains a variety of algorithms (e.g. GBMs, Random Forests, Deep Neural Networks, GLMs), yielding a healthy amount of diversity across candidate models, which can be exploited by stacked ensembles to produce a powerful final model. The effectiveness of this technique is reflected in the OpenML AutoML Benchmark, which compares the performance of several of the most well known, open source AutoML systems across a number of datasets.

1. Introduction

There have been big strides in the development of user-friendly machine learning software which features simple, unified interfaces to a variety of machine learning algorithms (e.g. scikit-learn, H2O, caret, tidymodels, mlr). Although these tools have made it easy for non-experts to train machine learning models, there is still a fair bit of expertise that is required in order to achieve state-of-the-art results. Automatic machine learning or "AutoML" tools provide a simple interface to train a large number of models (or a powerful single model), can be a helpful tool for either a novice or advanced machine learning practitioner. Simplifying training and tuning of machine learning models by offering a single function to replace a process that would typically require many lines of code, frees the practitioner to focus on other aspects of the data science pipeline, such as data-preprocessing, feature engineering and model deployment.

H2O AutoML (H2O.ai, 2017) is an automated machine learning algorithm included in the H2O framework (H2O.ai, 2013) that is simple to use and produces high quality models that are suitable for deployment in a enterprise environment. H2O AutoML supports supervised training of regression, binary classification and multi-class classification models on tabular datasets. One of the benefits of H2O models is the fast scoring capabilities – many

H2O models can generate predictions in sub-millisecond scoring times. H2O AutoML offers APIs in several languages (R, Python, Java, Scala) which means it can be used seamlessly within a diverse team of data scientists and engineers. It is also available via a point-and-click H2O web GUI called Flow[1], which further reduces the barriers to widespread use of automatic machine learning. H2O also has tight integrations to big data computing platforms such as Hadoop[2] and Spark[3] and has been successfully deployed on supercomputers[4] in a variety of HPC environments (e.g. Slurm).

The H2O AutoML algorithm was first released in June, 2017 in H2O v3.12.0.1 so it has gone through many iterations of development over the past three years. It is widely used in industry and academia and has many advocates in the open source machine learning community. The algorithm is constantly evolving and improving in each new version of H2O, so this paper serves as a snapshot of the algorithm in time (May, 2020). A full list of in-development and planned improvements and new features is available on the H2O bug tracker website.[5]

2. H2O AutoML

H2O AutoML is a fully automated supervised learning algorithm implemented in H2O, the open source, scalable, distributed machine learning framework. H2O AutoML is available in Python, R, Java and Scala as well as through a web GUI. Though the algorithm is fully automated, many of the settings are exposed as parameters to the user, so that certain aspects of the modeling steps can be customized.

2.1 Data pre-processing

H2O AutoML currently provides the same type of automatic data-preprocessing that's provided by all H2O supervised learning algorithms. This includes automatic imputation, normalization (when required), and one-hot encoding for XGBoost models. H2O tree-based models (Gradient Boosting Machines, Random Forests) support group-splits on categorical variables, so categorical data can be handled natively. In experimental versions of the algorithm, we have benchmarked various automatic target encoding[6] strategies for high-cardinality features, though that's not yet available in the current stable release (H2O v3.30.0.3). Additional data pre-processing steps such as automatic text encoding using Word2Vec[7], as well as feature selection and feature extraction for automatic dimensionality reduction are all part of the H2O AutoML roadmap.

[1] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/flow.html
[2] https://hadoop.apache.org/
[3] https://spark.apache.org/
[4] https://scinet.usda.gov/user/geospatial/#tools-and-software
[5] https://0xdata.atlassian.net/issues/?filter=21603
[6] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-munging/target-encoding.html
[7] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/word2vec.html

2.2 Models

2.2.1 Base Models

H2O AutoML includes XGBoost Gradient Boosting Machines (GBM)[8], as well as H2O Gradient Boosting Machines (GBM)[9], Random Forests[10] (Default and Extremely Randomized Tree variety), Deep Neural Networks[11] and Generalized Linear Models (GLM)[12]. H2O offers a wrapper around the popular XGBoost software, so we are able to include this third-party algorithm in H2O AutoML. This also allows GPU acceleration of training.

The current version of H2O AutoML (H2O v3.30.0.3) trains and cross-validates (when `nfolds` > 1) models: three pre-specified XGBoost GBM models, a fixed grid of H2O GLMs, a default H2O Random Forest (DRF), five pre-specified H2O GBMs, a near-default H2O Deep Neural Net, an H2O Extremely Randomized Trees (XRT) model, a random grid of XGBoost GBMs, a random grid of H2O GBMs, and a random grid of H2O Deep Neural Nets. For each algorithm, we identified which hyperparamters we consider to be *most important*, defined ranges for those parameters and utilize random search to generate models. A list of the hyperparameters and the ranges we explored for each algorithm in the H2O AutoML process is documented in the user guide.[13] Which hyperparamters to consider, as well as their ranges, were decided upon based on benchmarking as well as the experience of expert data scientists (e.g. Kaggle Grandmasters[14]), and this is something we consistently try to improve upon over time via extensive benchmarking.

The pre-specified models are included to give quick, reliable defaults for each algorithm. The order of the algorithms, which can be customized by the user, is set to start with models that consistently provide good results (pre-specified XGBoost models) across a wide variety of datasets, followed by a tuned GLM for a quick reference point. From here we prioritize increasing the diversity across our set of models (for the sake of the final Stacked Ensembles) by introducing a few Random Forests, (H2O) GBM and Deep Learning models. After this set of prescribed models is trained and added to the leaderboard, we begin a random search across those same algorithms. The proportion of time spent on each algorithm in the AutoML run is explicitly defined to give some algorithms (e.g. XGBoost GBM, H2O GBM) more time than others (e.g. H2O Deep Learning), according to our perceived or estimated "value" of each task. In H2O v3.30.0.1, we introduced an experimental parameter, `exploitation_ratio`, which, if activated, fine-tunes the learning rate of the best XGBoost GBM model and best H2O GBM model and if a better model is found, adds it to the leaderboard.

2.2.2 Stacked Ensembles

After training the base models, two Stacked Ensemble models are trained using H2O's Stacked Ensemble algorithm[15]. Stacked Ensembles, also called *Stacking* or *Super Learning*,

[8] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/xgboost.html

[9] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/gbm.html

[10] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/drf.html

[11] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/deep-learning.html

[12] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/glm.html

[13] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html#random-grid-search-parameters

[14] https://www.kaggle.com/rankings

[15] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/stacked-ensembles.html

is a class of algorithms that involves training a second-level *metalearner* to find the optimal combination of the base learners. Stacked Ensembles perform particularly well if the base models are individually strong and make uncorrelated errors. Random search across a variety of algorithm families produces a very diverse set of base models, and when paired with stacking, produces powerful ensembles.

The *All Models* ensemble contains all the models, and the *Best of Family* ensemble contains the best performing model from each algorithm class/family. In particular, that includes one XGBoost GBM, H2O Random Forest, H2O Extremely Randomized Tree Forest, H2O GBM, H2O Deep Learning, and H2O GLM model. The Best of Family ensemble is optimized for production use cases since it only contains six (or fewer) base models and can generate predictions rapidly compared the All Models ensemble. The Best of Family ensemble will usually have slightly lower model performance, however it's usually still a jump in performance over the top base model. Generally, both of the ensembles produce better models than any individual model from the AutoML run, however in rare cases, models such as a simple GLM can rise to the top of the leaderboard.

By default, the metalearner in the Stacked Ensemble will be trained using the k-fold cross-validated predictions from the base learners. This version of stacking is called the *Super Learner* algorithm (Laan et al., 2007) and has been proven to represent an asymptotically optimal system for learning. In cases where the data is very large or there is a time-dependency across rows in the data, it is not advised to use cross-validation. Use of a holdout blending frame is the recommended way to train a Stacked Ensemble in this scenario (described in Appendix C).

2.3 API

The H2O AutoML interface is designed to have as few parameters as possible so that all the user needs to do is point to their dataset, identify the response column and optionally specify a time constraint or limit on the number of total models trained.

Python and R interfaces:

```
aml = H2OAutoML(max_runtime_secs=3600)
aml.train(y="response_colname", training_frame=train)
```

```
aml <- h2o.automl(y = "response_colname", training_frame = train,
                  max_runtime_secs = 3600)
```

In both the R and Python API, H2O AutoML uses the same data-related arguments, as the other H2O algorithms (e.g. `x`, `y`, `training_frame`). The user can configure values for `max_runtime_secs` and/or `max_models` to set explicit time or number-of-model limits on your run. The only required user-defined parameters of H2O AutoML are the data specification. If a stopping criteria (`max_runtime_secs` and/or `max_ models`), is not set by the user, the AutoML run will execute for 1 hour. For Python users, there is also scikit-learn compatible API for H2O AutoML[16] included in the H2O python package, which exposes the standard scikit-learn methods (e.g. `fit`, `predict`, `score`), and supports integration into scikit-learn (Pedregosa et al., 2011a) pipelines.

[16] https://github.com/h2oai/h2o-tutorials/blob/master/tutorials/sklearn-integration/README.md

2.4 Leaderboard

The AutoML object includes a *leaderboard* which ranks all models trained by model performance. They are ranked by cross-validated performance by default, unless a holdout frame to score and rank models for the leaderboard is provided via the `leaderboard_frame` argument. An example leaderboard is shown in Table 1. Other performance metrics, such as training time and per-row prediction speed can be computed and added to the leaderboard via the `H2OAutoML.get_leaderboard()` and `h2o.get_leaderboard()` functions in Python and R, respectively. In production use cases, the top performing model which can generate predictions faster than a certain speed threshold might be chosen over the top model on the leaderboard, ranked by model accuracy (which is usually a Stacked Ensemble).

3. Performance

3.1 Accuracy

The OpenML AutoML Benchmark (Gijsbers et al., 2019) is an ongoing, and extensible benchmarking framework which follows best practices and avoids common mistakes in machine learning benchmarking. The 2019 benchmark compares four popular AutoML systems across 39 classification datasets using the framework, and there is continued work to integrate more systems and datasets as well as extending the benchmark to include regression.

Open Source AutoML systems. Some of the most well known open source AutoML systems for tabular data are Auto-WEKA (Thornton et al., 2013), auto-sklearn (Feurer et al., 2015), TPOT (Olson et al., 2016) and H2O AutoML, which are the four systems evaluated in the 2019 OpenML AutoML Benchmark. Some other projects, not part of the original benchmark, but worth mentioning are hyperopt-sklearn (Bergstra et al., 2015), autoxgboost (Thomas et al., 2018), ML-Plan (Mohr et al., 2018), OBOE (Yang et al., 2018), GAMA (G. and V., 2019), TransmogrifAI (Salesforce.com, 2018), Auto-keras (Jin et al., 2019), and the newly released AutoGluon-Tabular (Erickson et al., 2020).

The 2019 edition of the benchmark, published in Gijsbers et al. (2019), concluded that no one system consistently out-performed (based on accuracy) all the other systems on *small* or *medium-sized* ($\leq$ 50,000 rows) classification datasets on simple hardware (8 CPU cores, 32G RAM). In Truong et al. (2019), another AutoML benchmark on 300 datasets which focused on shorter runtimes ($\leq$ 1 hour), concluded that H2O AutoML, Auto-keras and auto-sklearn performed better than the other systems. In particular, they noted that H2O AutoML "slightly outperforms the others for binary classification and regression, and quickly converges to the optimal results".

We used the OpenML AutoML benchmark framework to run an updated version[17] of the benchmark, removing Auto-WEKA due to poor results on the previous benchmark and adding AutoGluon. We added 5 datasets that were part of the "validation" set[18] in the benchmark, for a total of 44. We used the latest stable version of all the systems, as of May 2019. Figure 4 shows results which are very close among most tools for binary classification. In Figure 3, we can see that H2O AutoML performs favorably compared to most algorithms in a variety of data types. AutoGluon showed strong results for multiclass

[17] http://github.com/h2oai/h2o-automl-paper

[18] https://github.com/openml/automlbenchmark/blob/master/resources/benchmarks/validation.yaml

problems, however not as strong as stated in Erickson et al. (2020) because all non-default options in AutoGluon were turned off, as required by the OpenML benchmark.

3.2 Scalability & Speed

Due to the efficient implementations of the algorithms and the distributed nature of the H2O platform, H2O AutoML can scale to large datasets (e.g. 100M+ rows) as shown in Section E.1. Training of individual models is parallelized across CPU cores on a single machine, or across a cluster of networked machines in a multinode setting. XGBoost models, which are included in the AutoML algorithm by default, also support GPU acceleration for further speed-up in training. Since a significant portion of training is usually dedicated to XGBoost models, H2O AutoML benefits from GPU acceleration. Appendix A includes a more detailed discussion about the architecture and scalability of the H2O platform.

One of the benefits of building an AutoML system on top of a fast, scalable machine learning library, is that you can utilize speed and parallelism to train more models in the same amount of time as compared to AutoML libraries that are built with slower or less scalable underlying algorithms. As demonstrated in the OpenML AutoML benchmark results in Figure 4, this allows us to use simple, straight-forward techniques like random search and stacking to achieve excellent performance in the same amount of time as algorithms which use more complex tuning techniques such as Bayesian optimization or genetic algorithms.

4. Conclusion

We presented H2O AutoML, an algorithm for automatic machine learning on tabular data, part of the H2O machine learning platform. H2O excels in the areas of usability and scalability and has a very active and engaged user base in the open source machine learning community. Key aspects of H2O AutoML include its ability to handle missing or categorical data natively, it's comprehensive modeling strategy, including powerful stacked ensembles, and the ease in which H2O models can be deployed and used in enterprise production environments. The *leaderboard* features informative and actionable information such as model performance, training time and per-row prediction speed for each model trained in the AutoML run, ranked according to user preference.

H2O models, including stacked ensembles, can be inspected further using model interpretability tools featured in H2O such as partial dependence plots[19], Shapley values[20], and can also be used in conjunction with popular third-party interpretabilty tools such as lime[21]. H2O AutoML is a well-tested, widely-used, scalable, production-ready automatic machine learning system with APIs in R, Python, Java and Scala, and a web GUI. With such wide language support, it's a tool that can be used across diverse teams of statisticians, data scientists and engineers, making it a very practical choice in any heterogeneous team. Its ability to scale to very large datasets also means that it can handle a diverse set use-cases (both large and small data) across an organization, allowing consistent tooling and less overhead than relying on different tools for different problems.

[19] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/performance-and-prediction.html#partial-dependence-plots

[20] https://www.h2o.ai/blog/h2o-release-3-26-yau/

[21] https://github.com/marcotcr/lime & https://github.com/thomasp85/lime

References

James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David Cox. Hyperopt: A python library for model selection and hyperparameter optimization. *Computational Science Discovery*, 8:014008, 07 2015. doi: 10.1088/1749-4699/8/1/014008.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011. ISSN 1935-8237. doi: 10.1561/2200000016. URL `http://dx.doi.org/10.1561/2200000016`.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

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.

Pieter G. and Joaquin V. GAMA: Genetic automated machine learning assistant. *Journal of Open Source Software*, 4(33):1132, jan 2019. doi: 10.21105/joss.01132. URL `https://doi.org/10.21105/joss.01132`.

P. Gijsbers, E. LeDell, S. Poirier, J. Thomas, B. Bischl, and J. Vanschoren. An open source automl benchmark. *6th ICML Workshop on Automated Machine Learning*, 2019. URL `https://www.automl.org/wp-content/uploads/2019/06/automlws2019_Paper45.pdf`.

H2O.ai. *H2O: Scalable Machine Learning Platform*, 2013. URL `https://github.com/h2oai/h2o-3`. First version of H2O was released in 2013.

H2O.ai. *H2O AutoML*, June 2017. URL `http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html`. First released in H2O version 3.12.0.1.

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`.

Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, pages 1946–1956, 2019. URL `https://dl.acm.org/doi/10.1145/3292500.3330648`.

Mark Laan, Eric Polley, and Alan Hubbard. Super learner. *Statistical applications in genetics and molecular biology*, 6:Article25, 02 2007. doi: 10.2202/1544-6115.1309.

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, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011a.

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, 2011b.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 693–701. Curran Associates, Inc., 2011. URL `http://papers.nips.cc/paper/4390-hogwild-a-lock-free-approach-to-parallelizing-stochastic-gradient-descent.pdf`.

Salesforce.com. *TransmogrifAI*, 2018. URL `https://github.com/salesforce/TransmogrifAI`.

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.

Anh Truong, Austin Walters, Jeremy Goodsitt, Keegan Hines, C. Bayan Bruss, and Reza Farivar. Towards automated machine learning: Evaluation and comparison of automl approaches and tools. *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, Nov 2019. doi: 10.1109/ictai.2019.00209. URL `https://arxiv.org/abs/1908.05557`.

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`.

Appendix A. H2O Distributed Machine Learning Platform

The H2O machine learning platform was designed with very large training sets in mind, so each of the algorithms available in H2O are scalable to larger-than-memory datasets and are trained in a fully parallelized fashion across multiple nodes and cores.

A.1 Architecture

The "H2O Cluster" is the Java process running H2O across one or many computers. The first task you do when working with H2O is to initialize the cluster, either locally, or on a remote machine. Once the cluster is running, data can be imported from disk into distributed data frames called "H2O Frames", and model training is also performed in memory inside the H2O cluster. The distributed data frame architecture as well as the in-memory computing aspect are similar to Apache Spark, though H2O's implementation (2013) pre-dates Spark (2014). This is why H2O implements its own distributed computing infrastructure versus relying upon a popular framework like Spark. At the time H2O was first introduced, MapReduce and Hadoop were very popular and so H2O is designed to work seamlessly with "big data" platforms such as Hadoop and (later) Spark.

H2O's core is written in highly optimized Java code, using primitive types (no Java objects) which gives FORTRAN-like speed. Inside H2O, a lock-free distributed key/value store (DKV) is used to read and write data (frames, models, objects) asynchronously across all nodes and machines. The algorithms are implemented on top of H2O's distributed Map/Reduce framework and utilize the Java Fork/Join framework for multi-threading. The data is read in parallel and is distributed across the cluster and stored in memory in a columnar format in a compressed way. H2O's data parser has built-in intelligence to guess the schema of the incoming dataset and supports data ingest from multiple sources in various formats.[22]

A.2 Scalability & Parallelism

To scale training to datasets that cannot fit inside the memory (RAM) of a single machine, you simply add compute nodes to your H2O cluster. The training set will be distributed across multiple machines, row-wise (a full row is contained in a single node, so different rows will be stored on different nodes in the cluster). The Java implementations of the distributed machine learning algorithms inside H2O are highly optimized and the training speeds are further accelerated by parallelized training. To reduce communication overhead between nodes, data is compressed and uncompressed on the fly.

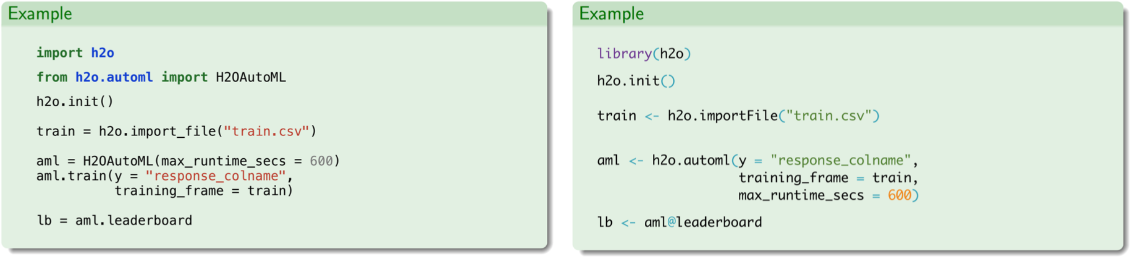
Within the algorithm implementations, there are many optimizations made to speed up the algorithms. Examples of such optimizations include pre-computing histograms (used extensively in tree-based methods), so they are available on-demand when needed. H2O GBM and Random Forest utilize group-splits for categorical columns, meaning that one-hot encoding (large memory cost) or label encoding (loss of categorical nature) is not necessary. Cutting edge optimization techniques such as the *alternating direction method of multipliers* (ADMM) (Boyd et al., 2011), an algorithm that solves convex optimization problems by breaking them into smaller pieces, are used extensively, providing both speed

[22] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/getting-data-into-h2o.html

and scalability. Optimization techniques are also selected dynamically based on data size and shape for further speed-up. For example, the H2O GLM uses a *iteratively reweighted least squares method* (IRLSM) with a Gram Matrix approach, which is efficient for tall and narrow datasets and when running lambda search via a sparse solution. For wider and dense datasets (thousands of predictors and up), the *limited-memory Broyden-Fletcher-Goldfarb-Shanno* (L-BFGS) solver scales better, so in that case, it will be used automatically.[23] H2O Deep Learning[24] includes many optimizations for speeding up training such as the HOGWILD! (Recht et al., 2011) approach to parelleizing stochastic gradient descent. When the associated optimization problem is sparse, meaning most gradient updates only modify small parts of the decision variable, then HOGWILD! achieves a nearly optimal rate of convergence. Here we have listed a few notable examples, but there are many other optimizations included in H2O algorithms, designed to promote speed and scalability.

Random search is an embarrassingly parallel task, which offers additional opportunity for speed-up. The current stable version of H2O AutoML (H2O 3.30.0.3) parallelizes training within a single model and partially for cross-validation across all cores of the H2O cluster, however the random search is executed serially, training a single model at any given time. H2O grid/random searches have a `parallel` argument which allows the user to specify how many models will be trained at once on the H2O cluster. Automatic parallelization of model training, in which we dynamically decide how many models to train in parallel, is a work in progress and is planned for a future release of H2O AutoML. The goal is to maximize the number of models that can be trained at once, given training set size and compute resources, without overloading the system.

Appendix B. H2O AutoML API



Example

```
import h2o
from h2o.automl import H2OAutoML
h2o.init()

train = h2o.import_file("train.csv")

aml = H2OAutoML(max_runtime_secs = 600)
aml.train(y = "response_colname",
          training_frame = train)

lb = aml.leaderboard
```

Example

```
library(h2o)
h2o.init()

train <- h2o.importFile("train.csv")

aml <- h2o.automl(y = "response_colname",
                  training_frame = train,
                  max_runtime_secs = 600)

lb <- aml@leaderboard
```

Figure 1: Basic, complete code examples of the H2O AutoML Python and R APIs.

Appendix C. H2O AutoML Customization

Though H2O AutoML is designed to be a fully automatic hyperparamter-free algorithm, discrete pieces of the algorithm can be modified, turned on/off or re-ordered. Other than controlling the length of the AutoML run, the most significant way you modify the algorithm is by turning on or off or re-ordering the constituent algorithms.

[23] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/glm.html#irlsm-and-l-bfgs

[24] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/deep-learning.html

Families of algorithms, including the Stacked Ensembles, can be switched off using the `exclude_algos` argument. This is useful if you have constraints of preferences about the algorithms that you'd like to train. Diversity among the set of base models generally increases the performance of the Stacked Ensemble, though in some cases, it's practical or beneficial to turn some algorithms off. The H2O AutoML function can also be used as a utility to tune a single algorithm (e.g. `"XGBoost"`) by specifying a single algorithm (and optionally `"StackedEnsemble"`) with the `include_algos` argument. The `modeling_plan` argument offers the ability to re-order discrete steps of the algorithm, such as the grid searches or pre-specified models. The max runtime allowed per model can also be specified via the `max_runtime_secs_per_model` argument.

The leaderboard will be scored using cross-validated metrics by default, but the user can pass in a test frame to the `leaderboard_frame` argument, which will then be used exclusively to score and rank models for the leaderboard. If the user prefers using a holdout frame to train the metalearner of the Stacked Ensemble instead of using cross-validation predictions (for example, if cross-validation is turned off), that frame can be passed to a `blending_frame` argument. Row weights and cross-validation fold row assignment can be specified via the `weights_column` and `fold_column` arguments.

The `nfolds` parameter allows the user to modify how many folds to use for cross-validation. Cross-validation is used to train Stacked Ensembles, as well as to rank models on the leaderboard. Fewer folds allow more models to train in the total AutoML runtime, however more folds generally lead to strong Stacked Ensemble models, so there's a trade-off to be considered. On larger datasets (anything over a million rows, for example), especially in time-limited scenarios, it can be advantageous to turn off cross-validation and train the Stacked Ensemble metalearner with the blending frame instead of a cross-validation predictions frame.

There are arguments related to early stopping of the individual algorithms, grid searches and AutoML run itself. The model performance metric which should be used for early stopping (e.g. `"AUC"`, `"logloss"`, `"RMSE"`) is specified via `stopping_metric`, and the `sort_metric` specifies which metric the models on the leaderboard should be sorted by. The `stopping_tolerance` and `stopping_rounds` parameters allow fine tuning of the early stopping sensitivity.

An experimental argument called `exploitation_ratio` was introduced in H2O v3.30.0.1, which allows the user to specify the budget ratio (between 0 and 1) dedicated to the *exploitation* (vs *exploration*) phase. By default, the exploitation phase is disabled, however, if activated, the exploitation phase will fine-tune the best XGBoost GBM and the best H2O GBM found during exploration (random search) and add the tuned models to the leaderboard. The tuned models will also be included in the set of base models for Stacked Ensembles. Additional fine-tuning may be added to other model types in the future.

Monotonic constraints which control the nature of the relationship between an input variable and the prediction can be passed through to supporting algorithms (XGBoost and H2O GBMs) via the `monotonic_constraints` argument.[25] This may be useful or necessary in certain industrial or scientific applications. Partial dependence plots, included in H2O, can be used to observe and validate the monotonicity.

[25] https://gist.github.com/ledell/91beb929dcdb04a964f5f580faa48a93

There are three arguments related to controlling for class imbalance which can be switched on manually, however we plan to address class imbalance in a more automatic fashion in future version. The remainder of the the arguments to the H2O AutoML function are related to keeping or deleting certain data artifacts, exporting models, and setting a random seed.

Appendix D. H2O AutoML Output

D.1 Leaderboard

Each AutoML run generates a leaderboard. The models are ranked by a default metric based on the problem type (the second column of the leaderboard). In binary classification problems, that metric is AUC, and in multiclass classification problems, the metric is mean per-class error. In regression problems, the default sort metric is deviance. The user can change the default leaderboard ranking metric via the `sort_metric` argument.

	model_id	auc	logloss	aucpr	mpce	rmse	mse
1	StackedEnsemble_AllModels_AutoML_20200519_023639	0.788	0.553	0.806	0.311	0.433	0.187
2	StackedEnsemble_BestOfFamily_AutoML_20200519_023639	0.787	0.554	0.804	0.318	0.433	0.188
3	GBM_5_AutoML_20200519_023639	0.782	0.558	0.802	0.320	0.436	0.190
4	GBM_2_AutoML_20200519_023639	0.778	0.563	0.796	0.334	0.438	0.191
5	GBM_1_AutoML_20200519_023639	0.777	0.563	0.799	0.356	0.438	0.192
6	GBM_3_AutoML_20200519_023639	0.775	0.565	0.795	0.328	0.439	0.192
7	GBM_grid__1_AutoML_20200519_023639_model_1	0.773	0.568	0.791	0.323	0.440	0.194
8	XGBoost_grid__1_AutoML_20200519_023639_model_4	0.773	0.568	0.789	0.322	0.440	0.194
9	GBM_4_AutoML_20200519_023639	0.772	0.569	0.793	0.337	0.441	0.194
10	XGBoost_grid__1_AutoML_20200519_023639_model_3	0.772	0.570	0.791	0.342	0.441	0.194
11	GBM_grid__1_AutoML_20200519_023639_model_2	0.770	0.569	0.789	0.370	0.441	0.194
12	XGBoost_3_AutoML_20200519_023639	0.770	0.573	0.785	0.338	0.442	0.195
13	DRF_1_AutoML_20200519_023639	0.765	0.580	0.782	0.336	0.445	0.198
14	XRT_1_AutoML_20200519_023639	0.760	0.585	0.777	0.339	0.448	0.200
15	XGBoost_grid__1_AutoML_20200519_023639_model_1	0.759	0.585	0.774	0.359	0.447	0.200
16	XGBoost_2_AutoML_20200519_023639	0.755	0.606	0.774	0.361	0.455	0.207
17	XGBoost_1_AutoML_20200519_023639	0.746	0.620	0.765	0.360	0.460	0.211
18	DeepLearning_grid__2_AutoML_20200519_023639_model_1	0.732	0.614	0.738	0.375	0.459	0.211
19	XGBoost_grid__1_AutoML_20200519_023639_model_2	0.730	0.814	0.748	0.397	0.497	0.247
20	DeepLearning_1_AutoML_20200519_023639	0.704	0.631	0.708	0.395	0.468	0.219
21	DeepLearning_grid__1_AutoML_20200519_023639_model_1	0.690	0.670	0.699	0.401	0.480	0.230
22	GLM_1_AutoML_20200519_023639	0.683	0.639	0.681	0.397	0.473	0.223

Table 1: Example of a binary classification Leaderboard. Taken from the H2O AutoML User Guide code examples section. http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html#code-examples

D.2 Log

When using Python or R clients, you can also access meta information through the event log (`event_log`), an H2OFrame with selected AutoML backend events generated during training. This information can also be viewed during the AutoML run by setting `verbose = "info"` in the AutoML function. There is another object generated called `training_info`, a simple dictionary/list exposing data that could be useful for post-analysis (various timings).

Appendix E. Benchmark Results

E.1 Ablation Study: CV-Stacking vs Blending

We compared different settings of H2O AutoML, specifically related to stacking, as the training set size increases. We used a version of the the *Airlines* dataset[26] which contains flight statistics for all flights in the U.S. from 1987 - 2013, totaling slightly more than 150 million rows. The goal is to predict departure delay, encoded as a binary response. There are a number of different types of features in this dataset (categorical, binary, numeric), which contain pre-flight information that may be useful in trying to predict whether a flight will be delayed.

We evaluated the difference between using 5-fold cross-validation (`nfolds = 5`) and a 10% blending frame (with `nfolds = 0`) to train the metalearner of the Stacked Ensembles. We prepared training sets of increasing sizes: 10k, 100k, 1M, 10M, 100M rows. The same test set of 100k rows was used to evaluate all the AutoML runs. The code and results from these benchmarks are available on Github.[27].

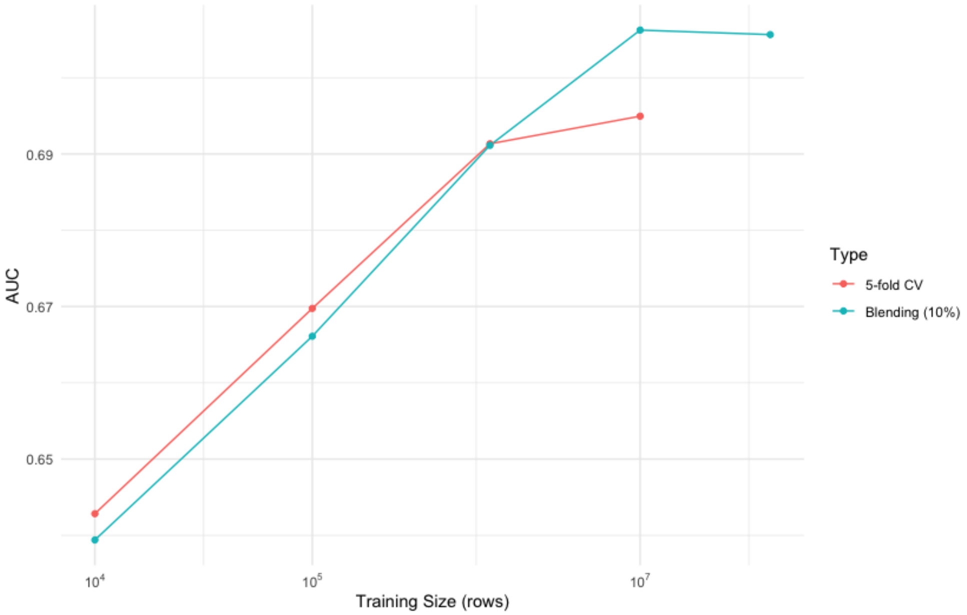


Figure 2: H2O AutoML scaling from 10,000 to 100M rows on the *Airlines* binary classification dataset on a single machine. Stacking with 5-fold cross-validated predictions versus stacking with a 10% blending frame partitioned from the training set.

As training set size increases, the added burden of doing *k*-fold cross-validation for purposes of metalearning, becomes less effective. At a certain point, the computational cost of doing cross-validation out-weighs the benefits of training the metalearner on more data and a Stacked Ensemble trained with a simple 10% holdout frame can out-perform the ensembles trained on the cross-validated predicted values from the base learners.

[26] https://www.transtats.bts.gov/Fields.asp?Table_ID=236

[27] http://github.com/h2oai/h2o-automl-paper

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 2: List of the AutoML tools in the OpenML AutoML Benchmark (2019).

With smaller data, or when your *data-size-to-compute-resources ratio* is high, H2O AutoML will typically produce a better Stacked Ensemble model using cross-validation, however, for larger datasets, especially in time-constrained scenarios (< 1 hour), it's recommended to reduce the number of cross-validation folds (e.g. `nfolds = 3`) or skip cross-validation completely and instead use a *blending frame* to train the Stacked Ensemble. The `blending_frame` argument allows the user to pass a holdout frame for purposes of training the metalearner in the Stacked Ensemble. A dynamic strategy, based on data to compute ratio, for choosing the number of folds or using a blending frame is planned for a future release of H2O AutoML.

Appendix F. OpenML AutoML Benchmark

A high level summary of the tools in the 2019 OpenML AutoML Benchmark, the first and only published edition of the benchmark thus far, is shown in Table 2.

The first prominent AutoML tool was Auto-WEKA, which used Bayesian optimization to select and tune the algorithms in a machine learning pipeline based on WEKA (Hall et al., 2009). It was followed by auto-sklearn, which uses the same basic technique on top of scikit-learn (Pedregosa et al., 2011b) and added meta-learning to warm-start the search with the best pipelines on similar datasets, as well as ensemble construction. TPOT optimizes scikit-learn pipelines via genetic programming, starting with simple ones and evolving them over generations. As noted in the main text, we removed Auto-WEKA from the benchmark since it was not competitive with the other tools, so the results shown in Figures 3 and 4 include auto-sklearn, TPOT, H2O AutoML and AutoGluon-Tabular.

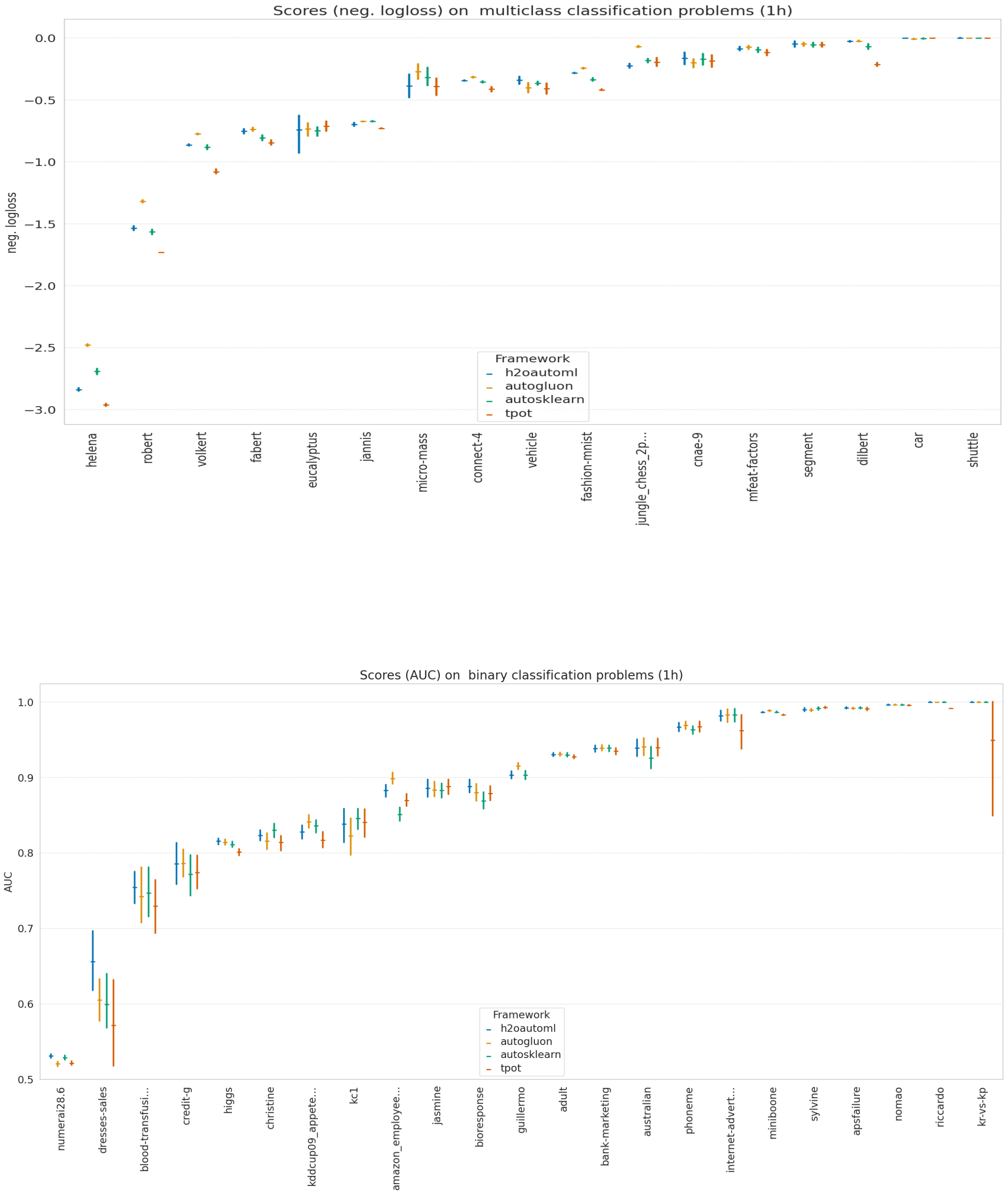


Figure 3: Updated OpenML AutoML Benchmark Results on binary and multiclass classification datasets (1 hour). AUC shown is 10-fold cross-validated AUC (the better scores are on the top). We added 5 datasets that were part of the "validation" in the benchmark, for a total of 44.

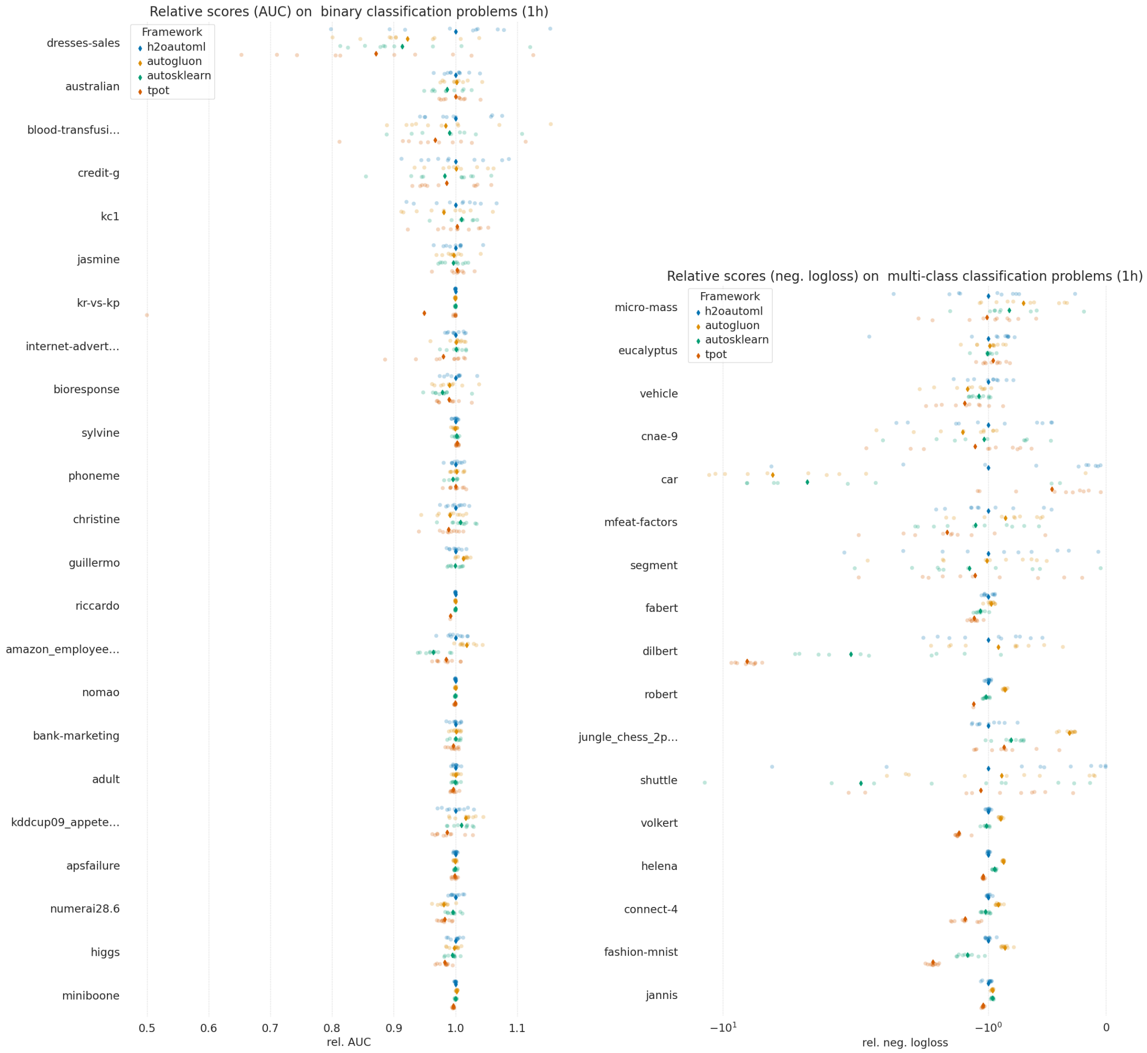


Figure 4: Updated OpenML AutoML Benchmark Results on binary and multiclass classification datasets (1 hour). The bold points are the 10-fold cross-validated values and the scores for each fold are also shown (the better scores are on the right). Both plots show the relative scores, as compared to H2O AutoML. We added 5 datasets that were part of the "validation" in the benchmark, for a total of 44.