Lessons learned from the AutoML challenge

Lisheng Sun-Hosoya, Isabelle Guyon, Michèle Sebag

Lessons learned from the AutoML challenge

Lisheng Sun-Hosoya*[1,2], Isabelle Guyon[1,2,3], et Michele Sebag[1,2]

[1]LRI, CNRS UMR 8623, Université Paris Sud
[2]INRIA Paris Saclay
[3]Chalearn

Abstract

We give a brief account of the main findings of our post-hoc analysis of the first AutoML challenge (2015-2016). This competition, which took place in 2015-2016 challenged the participants to submit code that solve classification and regression problems from fixed-length feature representations, without any human intervention. This paper is a digest of a book chapter to be published in the Springer Series on Challenges in Machine Learning [Geaar]. All datasets, code of the winners, and challenge results are found at: http://automl.chalearn.org.

Keywords: **Automatic ML, meta learning, transfer learning, Bayesian optimization**

1 Background

Machine Learning has achieved considerable successes in recent years and an ever-growing number of disciplines rely on it. However, this success crucially relies on human intervention in many steps (data preprocessing, feature engineering, model selection, hyperparameter optimization, etc.). As the complexity of these tasks is often beyond non-experts, the rapid growth of machine learning applications has created a demand for off-the-shelf or reusable methods, which can be used easily and without expert knowledge. The objective of AutoML (Automatic Machine Learning) challenges is to push research towards creating "universal learning machines" capable of learning and making predictions without human intervention. This means that the participants must deliver code, which is blind tested on datasets never released before.

The overall AutoML problem covers a wide range of difficulties, which cannot be addressed all at once in a single challenge. To name only a few: data "ingestion" and formatting, pre-processing and feature/representation learning, detection and handling of skewed/biased data, inhomogeneous, drifting, multimodal, or multi-view data (hinging on transfer learning), matching algorithms to problems (which may include supervised, unsupervised, or reinforcement learning, or other settings), acquisition of new data (active learning, query learning, reinforcement learning, causal experimentation), management of large volumes of data including the creation of appropriately sized and stratified training, validation, and test sets[1], selection of algorithms that satisfy arbitrary resource constraints at training and run time, the ability to generate and reuse workflows, and generating explicative reports.

Therefore, restricting the scope of a particular challenge is of great importance to ensure that the field progresses swiftly and intermediate milestones of immediate practical interest are reached. Our first challenge was limited to:

- **Supervised learning** problems (classification and regression).
- **Feature vector** representations.
- **Homogeneous datasets** (same distribution in the training, validation, and test set).
- **Medium size datasets** of less than 200 MBytes.
- **Limited computer resources** with an execution times of less than 20 minutes per dataset on an 8 core $x86\_64$ machine with 56 GB RAM.

Within this constrained setting, the testbed was composed of 30 datasets from a wide variety of application domains (medical diagnosis, speech recognition, credit

*cecile829@gmail.com

[1]In AutoML, the test sets were designed to be large enough such that the performance of participants are well separated, please refer to the appendix of [Geaar] for the error bars of winners' performance.

rating, prediction of drug toxicity/efficacy, classification of text, prediction of customer satisfaction, object recognition, protein structure prediction, action recognition in video data) and ranged across different types of complexity (class imbalance, sparsity, missing values, categorical variables). In this limited framework, there remain many modeling choices.

Many robust learning machines with a reduced number of hyper-parameters have emerged in the recent years in an effort to produce *perfect black-boxes* to perform tasks such as classification and regression [HTF01, DHS01]. But the availability of toolboxes rich in such models, *e.g.* Weka [HFH+09] or scikit-learn [PVG+11], has not eliminated modeling choices. Similarly to AI, which has endeavored to pass the Turing test, ML has undertaken the task of beating the "no free lunch theorem", stating that no model can be superior to all others on every task. Tools like AUTO-SKLEARN [FKE+15b, FKE+15a, FSH15][2], a wrapper around the scikit-learn library built by the winners of the AutoML challenge have made big strides towards that goal. The objective of this paper is to summarize where we are in this race. More details can be found in our full report [Geaar].

2 The AutoML setting

For the purpose of this paper, a predictive model (or model for short) has the form $y = f(\mathbf{x}) = f(\mathbf{x}; \boldsymbol{\alpha})$ with a set of parameters $\boldsymbol{\alpha} = [\alpha_0, \alpha_1, \alpha_2, ..., \alpha_n]$ trainable with a learning algorithm (trainer). The trained model (predictor) $y = f(\mathbf{x})$ produced by the trainer is evaluated by an objective function $J(f)$, used to assess the model performance on test data.

A model hypothesis space defined by a vector $\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_n]$ of hyper-parameters, which may include both categorical variables corresponding to switching between alternative models and other modeling choices such as preprocessing parameters, type of kernel in a kernel method, number of units and layers in a neural network, or training algorithm regularization parameters [SS01]. Some authors refer to this problem as *full model selection* [EMS09, SPM12], others as the CASH problem (Combined Algorithm Selection and Hyperparameter optimization) [THHL12].

We will then denote hyper-models as

$$y = f(\mathbf{x}; \boldsymbol{\theta}) = f(\mathbf{x}; \boldsymbol{\alpha}(\boldsymbol{\theta}), \boldsymbol{\theta}), \tag{1}$$

where the model parameter vector $\boldsymbol{\alpha}$ is an implicit function of the hyper-parameter vector $\boldsymbol{\theta}$ obtained by using a trainer for a fixed value of $\boldsymbol{\theta}$, and training data composed of input-output pairs $(\mathbf{x}_i, y_i)$.

The goal of the challenge participants is to devise algorithms capable of training the hyper-parameters $\boldsymbol{\theta}$.

[2] https://automl.github.io/auto-sklearn/stable/

3 Statistical complexity *vs.* computational complexity

The dilemma of model selection is to avoid "searching too hard" (and falling in the trap of over-fitting) and "not searching hard enough" (and falling in the trap of under-fitting). One often refers to as **statistical complexity** all ailments related to the "curse of dimensionality" or solving ill-posed problems, in which not enough training data is available to ensure good generalization. Another notion of complexity, complementing the first one, is **computational complexity**: exploring exhausively (or very intensively) a very large model space by evaluating "all" (or very many) models is generally infeasible because of the combinatorial nature of the problem of probing simultaneouly several hyper-parameters. Success in the AutoML challenge depends on addessing both types of complexity.

Best practices for model selection converge towards the Ockham's razor principle, which prescribes limiting model complexity to the minimum necessary to explain the data, or *shave off unnecessary parameters*. This has been grounded in theory over the past few years in such frameworks as regularization, Bayesian priors, Minimum Description Length, Structural Risk Minimization (SRM), and the bias/variance tradeoff [Ris78, GBD92, HTF01, DHS01, Vap98]. With modern learning machines designed to regularize and prevent overfitting, good practitioners usually do not over-train their models. Indeed, in the challenge, our analyses revealed no over-fitting of models. Two means of combatting over-fitting are pervasive: using cost functions **penalizing model complexity** and **ensembling**. The latter has been adopted by all winners, see Section 6.

The most pressing problem in today's AutoML research is therefore that of under-fitting, which cannot be solved by brute-force search if computational resources are limited. In the next section, we review what "clever search" entails in today's state-of-the art.

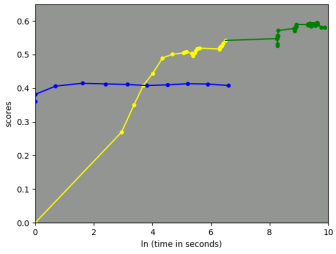

Figure 1: **Learning Curve of 'aad_freiburg' (yellow) and 'abhishek' (blue) for the evita dataset.** Abhishek *starts with a better solution but performs a less efficient exploration. In contrast,* aad_freiburg *starts lower but ends up with a better solution. In green: the 'aad_freiburg' learning curve keeps improving beyond the time limit imposed in the challenge (20 min).*

4 Heuristic search *vs.* Bayesian optimization

Model search (including hyper-parameter search, jointly referred to as HP search) involves two necessary components: (1) a means of estimating the performance of the model on future data (estimator), and (2) a strategy for exploring the search space (policy).

For problem 1, there is presently a large consensus for the choice of estimators. Obviously, when vast amounts of training data are available, reserving a single subset of the training data for validation is simple and efficient. Otherwise, common practice is to use some form of cross-validation (CV). **K-fold CV** and its variants are a favorite, in front of **bootstrapping** (*e.g.,* bagging used in Random Forests). For K, most people use K=10, although there is no clear theoretical foundation for this choice. It has been known for decades that a special kind of CV estimator, the leave-one-out estimator, can be efficiently approximated by training a single model (*e.g.,* virtual-leave-one-out [GGNZ06]). Yet such methods are not applicable to all algorithms and require dedicated code, so they are not popular amongst practitionners, which prefer using plain CV in a wrapper setting. Likewise, techniques of bilevel optimization, optimizing simultaneously parameters and hyper-parameters (*e.g.,* [BKJH08, MBB11]) have not gained in popularity for the same reason.

Problem 2 is presently the main focus in AutoML research: develop efficient search policies. The 'aad_freiburg' team (who developped *AUTO-SKLEARN* [FKE+15b, FKE+15a, FSH15] and dominated both the 2015-2016 AutoML challenge and its 2018 sequel) used a method inspired by **Bayesian optimization** [EFH+13]. The key idea is to guide the search with a "cheap" evaluation of models. CV is thus used to evaluate only a few candidate points in HP space. A predictor of model performance in HP space is built with a form of active learning. Random Forest (RF) regressors lend themselves particularly well to this exercise and have superseded Gaussian processes [HHMR]. One reason is that they are based on decision trees, which are hierarchical in nature, thus making it easy to map a hierarchy of hyper-parameters. Another reason is that, as an ensemble method, RF yields also an estimator of the variance of the predictions. Armed with an estimation of the expectation and the variance of the model to be evaluated, Bayesian optimization methods estimate the expected benefit of effectively training and testing a new model. Typically, this is a function expressing the exploration/exploitation tradeoff, *i.e.,* you want to explore regions of high variance (where your predictions are least confident) but not waste too much time exploring if there are low hanging fruits (models with good performances candidates for winning).

Another form of Bayesian optimization, which was very strong in the first AutoML challenge, is "freeze-thaw" (introduced by J. Lloyd in the first phases whose code was overtaken by S. Sun, placing 3rd in the final phase) [Llo16]. Other top ranking participants used various forms of **heuristic search** with performances that ended up nearly as good. It is difficult to tell apart at this stage the influence of various factors in the success of methods. Initialization played an important role. We show a typical example of learning curve in Figure 1. Given enough time, the Bayesian optimization method ('aad_freiburg') ends up with better performance, but Abhishek has a better initialization.

The strongest contender to such "clever search" methods is plain **grid search** applied to models having only very few hyper-parameters. Grid search applied to gradient tree boosting is a typical illustration of such approach, which has been very successful in challenges, since at least 2006 [Lut06]. The Intel team produced very good results with such methods in the AutoML challenge.

Finally, **search free methods** are worth mentionning. Marc Boullé applied the Selective Naive Bayes (SNB) [Bou07, Bou09] extending the Naive Bayes method for classification and regression. His software developed by Orange Labs and in use in production, was used in the challenge with minimal adaptation to make it compliant to the challenge settings. Without any further tuning, it returned a solution with honorable results, within the time limit of the challenge. It is therefore a very strong baseline.

5 Meta Learning

Meta-learning aims at defining some general principles over different datasets[3]. The 'aad_freiburg' team investigated meta-learning applied to the initialization of Bayesian search. Specifically, they considered 140 datasets from `openml.org`[VvRBT13] (a platform which allows to systematically run algorithms on datasets) and they defined meta-features of datasets, including simple statistics characterizing input and output space, and the performance of a few landmark algorithms such as one nearest neighbor (1NN) and decision tree. They also ran AUTO-SKLEARN on these datasets and recorded the best performing algorithm for each dataset. Given a new dataset, and considering its neighbors in terms of meta-features, they can then initialize the HP search for the new dataset with the algorithms performing best on its neighbors, resulting in significant improvements compared to random initialization of the HP search.

To further understand the success of this method, we investigated which meta-features are most predictive of the best performing models. To that end we performed the following experiment using scikit-learn. We excluded *landmark models* from the set of meta-features and built a linear discriminant classifier (LDA) to predict which of four *basic models* would perform best on the 30 datasets of the challenge, thus defining a 4-class classification problem. Basic models included Naive Bayes (NB), Stochastic Gradient Descent linear model (SGD-linear), K=Nearest neighbor (KNN), and Random Forest (RF), with default hyper-parameter settings. The results shown in Figure 2 reveal three clusters in the space of the two first LDA components. Further, the features that contribute most to the first two LDA components are the fraction of missing values and features characterizing the distribution of target values.[4]

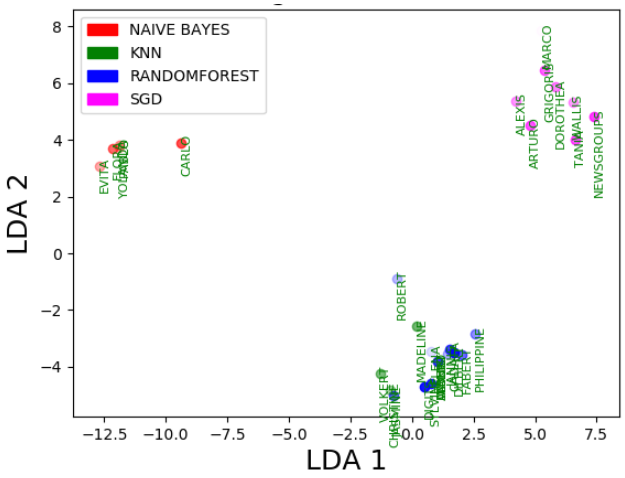

Figure 2: **Linear Discriminant Analysis.** *(a)We trained LDA using (X=meta features, except landmarks; y=which model won of four basic models (NB, SGD-linear, KNN, RF). The models were trained with default hyper parameters. In the space of the two first LDA components, each point represents one dataset. The colors code for the winning basic models. The color transparency reflects the scores of the corresponding winning model (better is darker).*

Although the 'aad_freiburg' team showed in their 2015 paper [FKE+15a] that this initialization faired better than random initialization, there is still room for improvement. Indeed, learning curves of the first AutoML (of which an example is shown in Figure 1) have revealed that other competitors had far better initializations.

For the 2018 edition of the AutoML challenge, the 'aad_freiburg' team introduced a novel strategy: *Portfolio Successive Halving*-AUTO-SKLEARN. As a form of meta-learning, they created a fixed portfolio of machine learning pipelines using over 400 datasets. At each period, the less successful pipelines (as estimated from the Bayesian optimization model), are discarded along the so-called *Bayesian Optimization HyperBand* (BO-HB) [FKH17, FKH18].

6 Towards AutoML: engineering *vs.* principles

Overall, we can ask ourselves whether the AutoML challenge helped pushing "the science of AutoML"

[3]Meta-learning differs from transfer learning, which is concerned with transferring models/knowledge among tasks. Transfer learning can take various forms depending on the type of information that overlaps between tasks [PY10], *i.e.,* similarity of input space distribution and/or similarity of outputs/labels. In the AutoML challenge framework however, the diversity among the application domains and types of learning difficulties hinders transfer learning, that will not be considered further in the paper. Actually, the help of using transfer learning in such competition is an open question. We have seen in other past challenges lending themselves to transfer learning that most (if not all) the participants did not do any transfer learning, even through on the long run transfer learning proved to be useful. The main problem may be that challenges are time constrained and transfer learning pays off only if you do it right and have enough time.

[4]Given the small size of available datasets (only 30 datasets in total), the LDA is trained on all datasets, and the model has no generality.

and whether some design guidelines have emerged and whether the "no free lunch" theorem has been beaten.

One thing is sure: ensembling always helps[5], whether you choose a homogeneous ensemble like Random Forests or a heterogeneous ensemble, built *e.g.,* with the method of [CNMCK04]. Other design choices regarding meta-learning and search are still evolving. However, what drives most progress in the field is the emergence of simple new concepts that researchers can share and re-implement to reproduce results. In that respect, Bayesian optimization has been helpful.

To reproduce the results of the challenge, there is one good news and one bad news. The good news is that all the code is open-sourced[6]. The bad news is that if you want to write your own code based on *e.g.,* scikit-learn and the principles outlined in this paper, it will be a significant amount of engineering. First, most methods of scikit-learn will die on you for many datasets (out-of-memory or out-of-time). Second, there are a lot of tricks of the trade to perform meta-learning and get Bayesian optimisation to work.

So it may be far easier to create your own code from scratch and create your own "universal approximator" with a handful of hyper-parameters tunable with grid search. But be careful, many other people have tried; scikit-learn is full of such models. Figure 3 shows the performance of "pure models" *vs.* the performances of the challenge winners. They are lagging behind. It is not so easy to beat the "no free lunch" theorem.

Other engineering aspects play an important role. In the third round of the challenge when large sparse datasets were introduced, the vast majority of methods failed blind testing by either running out of time or out of memory. Ironically, the winners won thanks to proper exception handling (they returned random results rather than failing). During the "tweakathon" phase that followed blind testing the participants had an opportunity to fix their code. This revealed that the tasks of round 3 were not particularly difficult, once engineering problems were dealt with.

You may also be tempted to go beyond Bayesian optimization in the line of research pursuing heterogeneous model search, by performing better meta-learning or even by learning policies with reinforcement learning. Some ideas along these lines have been proposed in the literature [ZL16, BGNR16] and for the first time in 2018, one of the top participants (wiWangl) used Q-Learning to learn machine learning pipelines.

[5]This phenomenon has also been observed heavily in Kaggle competitions[Kag], please refer to our overview in appendix 1.

[6]http://automl.chalearn.org

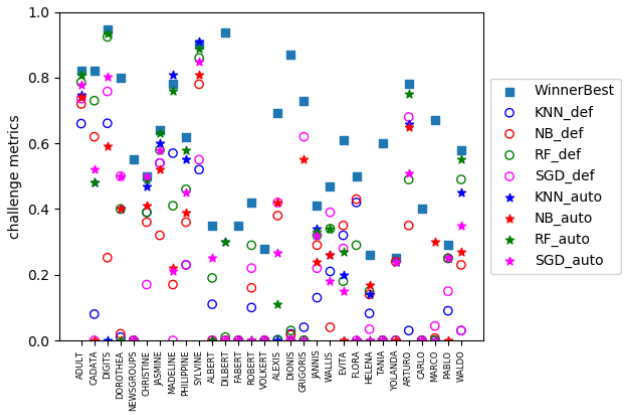

Figure 3: **Comparison of methods (2015-2016 challenge) including basic methods (-def suffix), basic methods with optimised HP (-auto suffix), and challenge winners.** *Winners in general win over basic methods, even with optimized HPs. There is no basic method that dominates all others. Though RF-auto (Random Forest with optimised HP) is very strong, it is often outperformed by other methods and sometimes by RF-def (Random Forest with default HP). Thus, under the tight computational constraints of the challenge, optimizing HP does not always pay. For KNN though, time permitting, optimizing HP generally helps by a long shot. Interestingly, KNN wins, even over the challenge winners, on some datasets.*

You may be tempted to use neural networks or deep learning. Unfortunately, for such time constrained challenge, even with GPUs (we provided GPUs in round 4 of the first AutoML challenge), they are not among the best performing methods.

In conclusion, it is fair to say that the winners provided a well engineered solution satisfying the constraint of the challenge in terms of time budget and robustness to algorithm failures, but for any new proposed task, manually selected and fined tuned algorithms may still perform better.

7 Discussion: challenge and benchmark design

The diversity of the 30 datasets of our first AutoML challenge was both a feature and a curse: it allowed us to test the robustness of software across a variety of situations, but made meta-learning difficult (datasets being different with respect to meta-features). Like-

wise, we attached different metrics (loss functions) to each dataset. This contributed to making the tasks more realistic and more difficult, but also made meta-learning harder. Consequently external datasets must be used if meta-learning is to be explored for the AutoML challenge tasks. As previously mentioned, this was the strategy adopted by the AAD Freiburg team, which used the OpenML data for meta training. They used over 400 datasets for meta-learning in last challenge edition.

With respect to task design, we learned that the devil is in the details. Challenge participants generally solve exactly the task proposed by the organizers, to the point that their solution may not be adaptable to seemingly similar scenarios. In the case of the AutoML challenge, we pondered whether the metric of the challenge should be the area under the learning curve (plotting performance as a function of time) or one point on the learning curve (the performance obtained after a fixed maximum computational time elapsed). We ended up favoring the second solution for practical reasons. Examining after the challenge the learning curves of some participants, it is quite clear that the two problems are radically different, particularly with respect to strategies mitigating "exploration" and "exploitation". This prompted us to think about the differences between "fixed time" learning (the participants know in advance the time limit and are judged only on the solution delivered at the end of that time) and "any time learning" (the participants can be stopped at any time and asked to return a solution). Both scenarios are useful: the first one is practical when models must be delivered continuously at a rapid pace, e.g. for marketing applications; the second one is practical in environments when computational resources are unreliable and interruption may be expected (e.g. people working remotely via an unreliable connection). This will influence the design of future challenges.

Also regarding task design, both AutoML challenges differ in the sequence of difficulties tackled in each round. In the 2015/2016 challenge, round 0 introduced five datasets representing a sample of all types of data and difficulties (types of targets, sparse data or not, missing data or not, categorical variables of not, more examples than features or not). Then each round ramped up difficulty introducing each time 5 new datasets. But in fact the datasets of round 0 were relatively easy. Then during each round, the code of the participants was blind tested on data that were one notch harder than in the previous round. Hence transfer was quite hard. In the 2018 challenge, we had only 2 phases, each with 5 datasets of similar difficulty and the datasets of the first phase were each matched with one corresponding dataset on a similar task in the second phase. As a result, transfer was made simpler.

Concerning the starting kit and baseline methods, we provided code that ended up being the basis of the solution of the majority of participants (with notable exceptions from industry such as Intel and Orange who used their own "in house" packages). Thus, we can question whether the software provided biased the approaches taken. Indeed, all participants used some form of ensemble learning, similarly to the strategy used in the starting kit. However, it can be argued that this is a "natural" strategy for this problem. But, in general, the question of providing enough starting material to the participants without biasing the challenge in a particular direction remains a delicate issue.

From the point of view of challenge protocol design, we learned that it is difficult to keep teams focused for an extended period of time and go through many challenge phases. We attained a large number of participants (over 600) over the whole course of the AutoML challenge, which lasted over a year (2015/2016) and was punctuated by several events (such as hackathons). However, few teams participated to all challenge rounds and despite our efforts to foster collaboration, the general spirit was competitive. It may be preferable to organize yearly events punctuated by workshops. This is a natural way of balancing competition an cooperation since workshops are a place of exchange where participants get rewarded by the recognition they gain via the system of scientific publications. As a confirmation of this conjecture, the second instance of the AutoML challenge (2017/2018) lasting only 4 months attracted nearly 300 participants.

One important novely of our challenge design was "code submission". Having the code of the participants executed on the same platform under rigorously similar conditions is a great step towards fairness and reproducibility, as well as ensuring the viability of solutions from the computational point of view. We have imposed to the winners to release their code under an open source licence to win their prizes. This was good enough an incentive to obtain several publicly available software as the "product" of the challenges we organized. In our second challenge (AutoML 2018), we have made use of dockers. Distributing the docker makes it possible for anyone downloading the code of the participants to reproduce easily the results without stubling upon installation problems due to inconsistencies in computer environments and libraries. Still the hardware may be different and we find that, in post-challenge evaluations, changing computer may yield

significant differences in results. Hopefully, with the generalization of use of cloud computing that is becoming more affordable, this will become less of an issue.

The AutoML challenge series is only beginning. Several new avenues are under study. We are preparing the NIPS 2018 Life Long Machine Learning challenge in which participants will be exposed on data whose distribution slowly drifts over time. We are also preparing a challenge of automatic machine learning from raw data with Google Zurich, priviledging transfer from similar domains.

Acknowledgments

This paper summarizes the work of a large number of people, including those involved in data donation and formatting Y. Aphinyanaphongs, O. Chapelle, Z. Iftikhar Malhi, V. Lemaire, C.-J. Lin, M. Madani, B. Ray, G. Stolovitzky, H.-J. Thiesen, and I. Tsamardinos, those involved early on in challenge design K. Bennett, C. Capponi, G. Cawley, R. Caruana, G. Dror, T. K. Ho, B. Kégl, H. Larochelle, V. Lemaire, C.-J. Lin, V. Ponce López, N. Macia, S. Mercer, F. Popescu, D. Silver, S. Treguer, and I. Tsamardinos, the software developers and beta-testers E. Camichael, I. Chaabane, I. Judson, C. Poulain, P. Liang, A. Pesah, L. Romaszko, X. Baro Solé, E. Watson, F. Zhingri, M. Zyskowski, and the main challenge organizers H. J. Escalante, S. Escalera, M. Saeed A. Statnikov, W. W. Tu and E. Viegas and result analysers I. Chaabane, J. Lloyd, N. Macia, and A. Thakur, M. Boullé, Z. Liu, and D. Jajetic.

References

[BGNR16] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

[BKJH08] Kristin P. Bennett, Gautam Kunapuli, and Jong-Shi Pang Jing Hu. Bilevel optimization and machine learning. In *Computational Intelligence: Research Frontiers*, volume 5050 of *Lecture Notes in Computer Science*, pages 25–47. Springer, 2008.

[Bou07] Marc Boullé. Compression-based averaging of selective naive bayes classifiers. *Journal of Machine Learning Research*, 8:1659–1685, 2007.

[Bou09] Marc Boullé. A parameter-free classification method for large scale learning. *Journal of Machine Learning Research*, 10:1367–1385, 2009.

[CNMCK04] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *21st International Conference on Machine Learning*, pages 18–. ACM, 2004.

[DHS01] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2nd edition, 2001.

[EFH+13] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, 2013.

[EMS09] Hugo Jair Escalante, Manuel Montes, and Luis Enrique Sucar. Particle swarm model selection. *Journal of Machine Learning Research*, 10:405–440, 2009.

[FKE+15a] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Proceedings of the Neural Information Processing Systems*, pages 2962–2970. 2015.

[FKE+15b] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Methods for improving bayesian optimization for automl. In *Proceedings of the International Conference on Machine Learning 2015, Workshop on Automatic Machine Learning*, 2015.

[FKH17] Stefan Falkner, Aaron Klein, and Frank Hutter. Combining hyperband and bayesian optimization. In *BayesOpt 2017 NIPS Workshop on Bayesian Optimization*, 2017.

[FKH18] Stefan Falkner, Aaron Klein, and Frank Hutter. Practical hyperparameter optimization. In *International Conference on Learning Representations 2018 Workshop track*, 2018.

[FSH15] M. Feurer, J.T. Springenberg, and F. Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1128–1135, 2015.

[GBD92] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[Geaar] I. Guyon et al. Analysis of the AutoML challenge series 2015-2018. In F. Hutter et al., editor, *Automatic Machine Learning*. Springer series in Challanges in Machine Learning, 2017, to appear.

[GGNZ06] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti Zadeh, editors. *Feature extraction, foundations and applications*. Studies in Fuzziness and Soft Computing. Physica-Verlag, Springer, 2006.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The Weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[HHMR] F. Hutter, H. Hoos, K. Murphy, and S. Ramage. Sequential Model-based Algorithm Configuration (SMAC). http://www.cs.ubc.ca/labs/beta/Projects/SMAC/.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. Springer, 2nd edition, 2001.

[Kag] Kaggle. https://www.kaggle.com/. Accessed: 2018-06-07.

[Llo16] J. Lloyd. Freeze Thaw Ensemble Construction. https://github.com/jamesrobertlloyd/automl-phase-2, 2016.

[Lut06] R. W. Lutz. Logitboost with trees applied to the WCCI 2006 performance prediction challenge datasets. In *Proc. IJCNN06*, pages 2966–2969, Vancouver, Canada, July 2006. INNS/IEEE.

[MBB11] Gregory Moore, Charles Bergeron, and Kristin P. Bennett. Model selection for primal SVM. *Machine Learning*, 85(1-2), October 2011.

[PVG+11] 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, 2011.

[PY10] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knoweledge and Data Engineering*, 22(10):1345–1359, October 2010.

[Ris78] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, September 1978.

[SPM12] Quan Sun, Bernhard Pfahringer, and Michael Mayo. Full model selection in the space of data mining operators. In *Genetic and Evolutionary Computation Conference*, pages 1503–1504, 2012.

[SS01] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[THHL12] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Automated selection and hyperparameter optimization of classification algorithms. *CoRR*, abs/1208.3719, 2012.

[Vap98] Vladimir N. Vapnik. *Statistical learning theory*. Wiley, 1998.

[VvRBT13] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

[ZL16] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A Usage of ensembling in Kaggle competitions

Competition	Use ensemble?
TensorFlow Speech Recognition Challenge (ends 2018-01-17)	1st place: ? 2nd place: Yes 3rd place: Yes
Corporacion Favorita Grocery Sales Forecasting (ends 2018-01-16)	1st place: Yes 2nd place: ? 3rd place: Yes
Cdiscounts Image Classification Challenge (ends 2017-12-15)	1st place: Yes 2nd place: Yes
Porto Seguros Safe Driver Prediction (ends 2017-11-30)	1st place: ? 2nd place: ? 3rd place: ?
Carvana Image Masking Challenge (ends 2017-09-28)	1st place: Yes 3rd place: No
Instacart Market Basket Analysis (ends 2017-08-15)	2nd place: ? 3rd place: ?
Planet: Understanding the Amazon from Space (ends 2017-07-21)	1st place: Yes 3rd place: Yes
Mercedes-Benz Greener Manufacturing (ends 2017-07-1)	1st place: ? 2nd place: Yes
Sberbank Russian Housing Market (ends 2017-06-30)	1st place: Yes
Intel & MobileODT Cervical Cancer Screening (ends 2017-06-22)	6th place: Yes 9th place: Yes
Quora Question Pairs (ends 2017-06-07)	1st place: ? 2nd place: Yes 3rd place: Yes
Google Cloud & YouTube-8M Video Understanding Challenge (ends 2017-06-07)	1st place: Yes 5th place: Yes
Data Science Bowl 2017 (ends 2017-04-13)	1st place: ? 2nd place: Yes
The Nature Conservancy Fisheries Monitoring (ends 2017-04-13)	8th place: Yes 9th place: Yes
Dstl Satellite Imagery Feature Detection (ends 2017-03-08)	5th place: ? 9th place: ?
Outbrain Click Prediction (ends 2017-01-19)	2nd place: Yes 3rd place: Yes 4th place: Yes

Table 1: Overview of the usage of ensemble techniques in winner solutions of 2017-2018 Kaggle competitions[Kag]. Source: http://ndres.me/kaggle-past-solutions/. We list only competitions with winner solution overviews available. The usage is marked as 'Yes'/ 'No' if it is mentioned directly in the winner's solution overview, otherwise it is marked as '?'. Up to 23 winners out of 36 have employed ensembling in their winning solutions, only 1 winner declared the non-use of ensembling.