Analysis of the AutoML Challenge series 2015-2018

Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, et al.

Analysis of the AutoML Challenge series 2015-2018

Isabelle Guyon GUYON@CHALEARN.ORG
UPSud/INRIA, U. Paris-Saclay, France and ChaLearn, USA
Lisheng Sun-Hosoya CECILE829@GMAIL.COM
UPSud, U. Paris-Saclay, France
Marc Boullé MARC.BOULLE@ORANGE.COM
Orange Labs, France
Hugo Jair Escalante HUGO.JAIR@GMAIL.COM
INAOE, Mexico
Sergio Escalera SERGIO@MAIA.UB.ES
Computer Vision Center and University of Barcelona, Barcelona, Spain
Zhengying Liu ZHENGYING.LIU@LRI.FR
UPSud, U. Paris-Saclay, France
Damir Jajetic DAMIR.JAJETIC@IN2.HR
IN2, Croatia
Bisakha Ray BISAKHA.RAY@NYUMC.ORG
NYU, USA
Mehreen Saeed MEHREEN.MEHREEN@GMAIL.COM
FAST, Pakistan
Michèle Sebag MICHELE.SEBAG@LRI.FR
CNRS, U. Paris-Saclay, France
Alexander Statnikov STATNIKOV@GMAIL.COM
American Express, USA
Wei-Wei Tu TUWW.CN@GMAIL.COM
4Paradigm Inc., Beijing, China
Evelyne Viegas EVELYNEV@MICROSOFT.COM
Microsoft Research, USA

Abstract

The ChaLearn AutoML Challenge[1] (NIPS 2015 - ICML 2016) consisted of six rounds of a machine learning competition of progressive difficulty, subject to limited computational resources. It was followed by one round of AutoML challenge (PAKDD 2018).

The AutoML setting differs from former model selection/hyper-parameter selection challenges, such as the one we previously organized for NIPS 2006: the participants aim to develop fully automated and computationally efficient systems, capable of being trained and tested without human intervention, with code submission.

This paper analyzes the results of these competitions and provides details about the datasets, which were not revealed to the participants. The solutions of the winners are systematically benchmarked over all datasets of all rounds and compared with canonical machine learning algorithms available in scikit-learn. All materials discussed in this paper (data and code) have been made publicly available at http://automl.chalearn.org/.

1. The authors are in alphabetical order of last name, except the first author who did most of the writing and the second author who produced most of the numerical analyses and plots.

Keywords: AutoML Challenge, machine learning, model selection, meta-learning, representation learning, hyper-parameter selection, any-time learning, fixed-time learning

1. Introduction

Until about ten years ago, machine learning (ML) was a discipline little known to the public. For ML scientists, it was a "sellers market": they were producing hosts of algorithms in search for applications and were constantly looking for new interesting datasets. Large internet corporations accumulating massive amounts of data such as Google, Facebook, Microsoft and Amazon have popularized the use of ML and data science competitions have engaged a new generation of young scientists in this wake. Nowadays, government and corporations keep identifying new applications of ML and with the increased availability of open data, we have switched to a "buyers market": everyone seems to be in need of a learning machine. Unfortunately however, learning machines are not yet fully automatic: it is still difficult to figure out which software applies to which problem, how to horseshoe-fit data into a software and how to select properly (hyper-)parameters. The ambition of the ChaLearn AutoML challenge series is to channel the energy of the ML community to reduce step by step the need for human intervention in applying ML to a wide variety of practical problems.

Full automation is an unbounded problem since there can always be novel settings, which have never been encountered before. Our first challenges AutoML1 were 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 execution times of less than 20 minutes per dataset on an 8 core $x86\_64$ machine with 56 GB RAM.

By doing that, we excluded unsupervised learning, active learning, transfer learning, and causal discovery problems, which are all very dear to us and have been addressed in past ChaLearn challenges (Guyon, 2011-2016), but which require each a different evaluation setting, thus making result comparisons very difficult. We did not exclude the treatment of video, images, text, and more generally time series and the selected datasets actually contain several instances of such modalities. However, they were first preprocessed in a feature representation, thus de-emphasizing feature learning. Still, learning from data pre-processed in feature-based representations already covers a lot of grounds and a fully automated method resolving this restricted problem would already be a major advance in the field.

Within this constrained setting, we included a variety of difficulties:

- **Different data distributions:** the intrinsic/geometrical complexity of the dataset.
- **Different tasks:** regression, binary classification, multi-class classification, multi-label classification.
- **Different scoring metrics:** AUC, BAC, MSE, F_1, etc. (see Section 4.2).

- **Class balance:** Balanced or unbalanced class proportions.
- **Sparsity:** Full matrices or sparse matrices.
- **Missing values:** Presence or absence of missing values.
- **Categorical variables:** Presence or absence of categorical variables.
- **Irrelevant variables:** Presence or absence of additional irrelevant variables (distractors).
- **Number P_{tr} of training examples:** Small or large number of training examples.
- **Number N of variables/features:** Small or large number of variables.
- **Ratio P_{tr}/N of the training data matrix:** $P_{tr} \gg N, P_{tr} = N$ or $P_{tr} \ll N$.

In this setting, the participants had to face many modeling/hyper-parameter choices. Some other, equally important, aspects of automating machine learning were not addressed in this challenge and are left for future research. Those include data "ingestion" and formatting, pre-processing and feature/representation learning, detection and handling of skewed/biased data, inhomogeneous, drifting, multi-modal, 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, selection of algorithms that satisfy arbitrary resource constraints at training and run time, the ability to generate and reuse workflows, and generating explicative reports.

This challenge series started with the NIPS 2006 "model selection game"[2] (Guyon et al., 2011), where the participants were provided with a machine learning toolbox based on the Matlab toolkit CLOP (Alamdari and Guyon, 2006) built on top of "the Spider" package (Weston et al., 2007). The toolkit provided a flexible way of building models by combining preprocessing, feature selection, classification and post-processing modules, also enabling the building of ensembles of classifiers. The goal of the game was to build the best hyper-model: the focus was on model selection, not on the development of new algorithms. All problems were feature-based binary classification problems. Five datasets were provided. The participants had to submit the schema of their model. The model selection game confirmed the effectiveness of cross-validation (the winner invented a new variant called cross-indexing) and pre-figured the need to focus more on search effectiveness with the deployment by novel search techniques such as particle swarm optimization.

New in the 2015/2016 AutoML challenge, we introduced the notion of "task": each dataset was supplied with a particular scoring metric to be optimized and a time budget. We initially intended to vary widely the time budget from dataset to dataset in an arbitrary way. We ended up fixing it to 20 minutes for practical reasons (Except for Round 0 where the time budget ranged from 100 to 300 seconds). However, because the datasets varied in size, this put pressure on the participants to manage their allotted time. Other elements of novelty included the freedom of submitting any Linux executable. This was made possible by using automatic execution on the open-source platform Codalab[3]. To help the participants we

2. http://clopinet.com/isabelle/Projects/NIPS2006/
3. http://competitions.codalab.org

provided a starting kit in Python based on the scikit-learn library (Pedregosa et al., 2011)[4]. This induced many of them to write a wrapper around scikit-learn. This has been the strategy of the winning entry "auto-sklearn" (Feurer et al., 2015a,b,c, 2018)[5]. Following the AutoML challenge, we organized a "beat auto-sklearn" game on a single dataset (madeline), in which the participants could provide hyper-parameters "by hand" to try to beat auto-sklearn. But nobody could beat auto-sklearn! Not even their designers. The participants could submit a json file which describes a sklearn model and hyper-parameter settings, via a GUI interface. This interface usefully allows researchers who want to compare their search methods with auto-sklearn to use the exact same set of hyper-models.

A large number of satellite events including bootcamps, summer schools, and workshops have been organized in 2015/2016 around the AutoML challenge.[6] The AutoML challenge was part of the official selection of the competition program of IJCNN 2015 and 2016 and the results were discussed at the AutoML and CiML workshops at ICML and NIPS in 2015 and 2016. Several publications accompanied these events: in (Guyon et al., 2015b) we describe the details of the design of the AutoML challenge[7]. In (Guyon et al., 2015a) and (Guyon et al., 2016) we review milestone and final results presented at the ICML 2015 and 2016 AutoML workshops. The 2015/2016 AutoML challenge had 6 rounds introducing 5 datasets each. We also organized a follow-up event for the PAKDD conference 2018[8] in only 2 phases, with 5 datasets in the development phase and 5 datasets in the final "blind test" round.

Going beyond the former published analyses, this paper presents systematic studies of the winning solutions on all the datasets of the challenge and conducts comparisons with commonly used learning machines implemented in scikit-learn. It provides unpublished details about the datasets and reflective analyses.

2. Problem Formalization and Overview

2.1. Scope of the Problem

This challenge series focuses on supervised learning in ML and, in particular, solving classification and regression problems, without any further human intervention, within given constraints. To this end, we released a large number of datasets pre-formatted in given feature representations (*i.e.,* each example consists of a fixed number of numerical coefficients; more in Section 3).

The distinction between input and output variables is not always made in ML applications. For instance, in recommender systems, the problem is often stated as making predictions of missing values for every variable rather than predicting the values of a particular variable (Ricci et al., 2011). In unsupervised learning (Ghahramani, 2004), the purpose is to explain data in a simple and compact way, eventually involving inferred latent variables (*e.g.,* class membership produced by a clustering algorithm).

4. http://scikit-learn.org/
5. https://automl.github.io/auto-sklearn/stable/
6. See http://automl.chalearn.org.
7. http://codalab.org/AutoML
8. https://www.4paradigm.com/competition/pakdd2018

In the following, only the strict supervised learning setting where data present themselves as identically and independently distributed input-output pairs, is considered. The models used are limited to fixed-length vectorial representations, excluding problems of time series prediction. Text, speech, and video processing tasks included in the challenge have been preprocessed in suitable fixed-length vectorial representations.

The difficulty of the proposed tasks lies on the data complexity (class imbalance, sparsity, missing values, categorical variables). The testbed is composed of data from a wide variety of domains. Although there exist ML toolkits that can tackle all these problems, it still requires considerable human effort to find, for a given dataset, task, evaluation metric, the methods and hyper-parameter settings that maximize performance subject to a computational constraint. The participant challenge is to create the *perfect black box* that removes human interaction, alleviating the shortage of data scientists in the coming decade.

2.2. Full Model Selection

In what follows, we refer to participant solutions as *hyper-models* to indicate that they are built from simpler components. For instance, for classification problems, participants might consider a hyper-model that combines several classification techniques such as nearest neighbors, linear models, kernel methods, neural networks, and random forests. More complex hyper-models may also include preprocessing, feature construction, and feature selection modules.

Generally, a predictive model of the form $y = f(\mathbf{x}; \boldsymbol{\alpha})$ has:

• a set of parameters $\boldsymbol{\alpha} = [\alpha_0, \alpha_1, \alpha_2, ..., \alpha_n]$;

• a learning algorithm (referred to as trainer), which serves to optimize the parameters using training data;

• a trained model (referred to as predictor) of the form $y = f(\mathbf{x})$ produced by the trainer;

• a clear objective function $J(f)$, which can be used to assess the model's performance on test data.

Consider now the model hypothesis space defined by a vector $\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_n]$ of hyper-parameters. The hyper-parameter vector may include not only variables corresponding to switching between alternative models, but also 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 (Schölkopf and Smola, 2001). Some authors refer to this problem as *full model selection* (Escalante et al., 2009; Sun et al., 2012), others as the CASH problem (Combined Algorithm Selection and Hyperparameter optimization) (Thornton et al., 2012). 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 participants have to devise algorithms capable of training the hyper-parameters $\boldsymbol{\theta}$. This may require intelligent sampling of the hyper-parameter space and splitting the available training data into subsets for both training and evaluating the predictive power of solutions—one or multiple times.

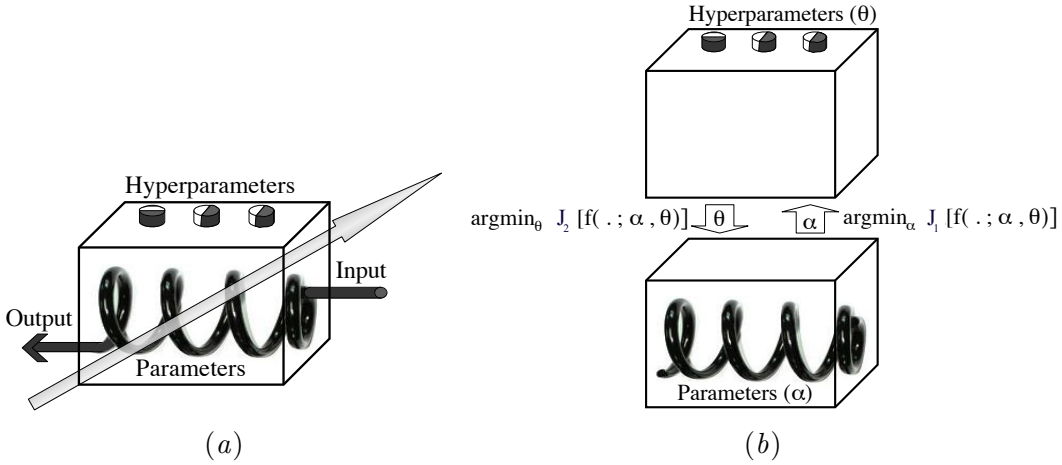


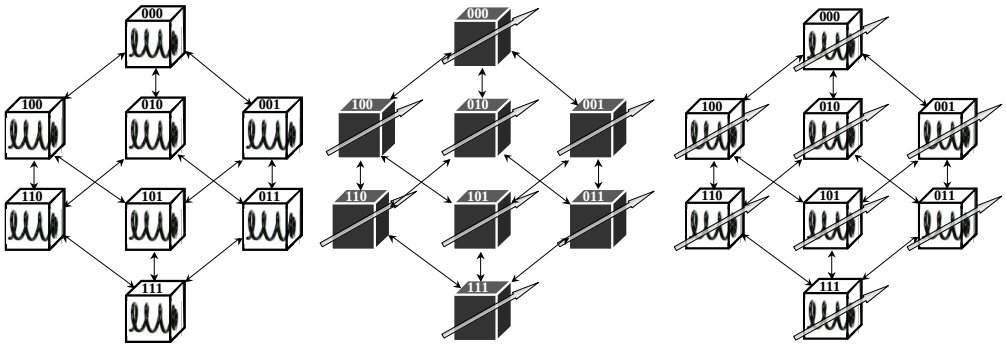
Figure 1: **Bi-level optimization.** (a) Representation of a learning machine with parameters and hyper-parameters to be adjusted. (b) De-coupling of parameter and hyper-parameter adjustment in two levels. The upper level objective J_2 optimizes the hyper-parameters $\boldsymbol{\theta}$; the lower objective J_1 optimizes the parameters $\boldsymbol{\alpha}$.

As an optimization problem, model selection is a bi-level optimization program (Colson et al., 2007; Dempe, 2002; Bennett et al., 2008); there is a lower objective J_1 to train the parameters $\boldsymbol{\alpha}$ of the model, and an upper objective J_2 to train the hyper-parameters $\boldsymbol{\theta}$, both optimized simultaneously (see Figure 1). As a statistics problem, model selection is a problem of multiple testing in which error bars on performance prediction ϵ degrade with the number of models/hyper-parameters tried or, more generally, the complexity of the hyper-model $C_2(\boldsymbol{\theta})$. A key aspect of AutoML is to avoid overfitting the upper-level objective J_2 by regularizing it, much in the same way as lower level objectives J_1 are regularized.

The problem setting also lends itself to using ensemble methods, which let several "simple" models vote to make the final decision (Breiman, 2001; Friedman, 2001; Caruana et al., 2004). In this case, the parameters $\boldsymbol{\theta}$ may be interpreted as voting weights. For simplicity we lump all parameters in a single vector, but more elaborate structures, such as trees or graphs can be used to define the hyper-parameter space (Thornton et al., 2013).

2.3. Optimization of Hyper-parameters

Everyone who has modeled data has had to face some common modeling choices: scaling, normalization, missing value imputation, variable coding (for categorical variables), variable discretization, degree of nonlinearity and model architecture, among others. ML has managed to reduce the number of hyper-parameters and produce *black-boxes* to perform tasks such as classification and regression (Hastie et al., 2001; Duda et al., 2001). Still, any real-world problem requires at least some preparation of the data before it can be fitted into an "automatic" method, hence requiring some modeling choices. There has been much progress on end-to-end automatic ML for more complex tasks such as text, image, video, and speech processing with deep-learning methods (Bengio et al., 2013). However, even these methods have many modeling choices and hyper-parameters.



(*a*) Fliter (*b*) Wrapper (*c*) Embedded

Figure 2: **Approaches to two-level inference.** (a) **Filter methods** select the hyper-parameters without adjusting the learner parameters. (No arrows indicates no parameter training.) (b) **Wrapper methods** select the hyper-parameters using trained learners, treating them as black-boxes. (c) **Embedded methods** use knowledge of the learner structure and/or parameters to guide the hyper-parameter search.

While producing models for a diverse range of applications has been a focus of the ML community, little effort has been devoted to the optimization of hyper-parameters. Common practices that include *trial and error* and grid search may lead to overfitting data for small datasets or underfitting data for large datasets. By overfitting we mean producing models that perform well on training data but perform poorly on unseen data, *i.e.,* models that do not generalize. By underfitting we mean selecting too simple a model, which does not capture the complexity of the data, and hence performs poorly both on training and test data. Despite well-optimized off-the-shelf algorithms for optimizing parameters, end-users are still responsible for organizing their numerical experiments to identify the best of a number of models under consideration. Due to lack of time and resources, they often perform model/hyper-parameter selection with ad hoc techniques. (Ioannidis, 2005; Langford, 2005) examine fundamental, common mistakes such as poor construction of training/test splits, inappropriate model complexity, hyper-parameter selection using test sets, misuse of computational resources, and misleading test metrics, which may invalidate an entire study. Participants must avoid these flaws and devise systems that can be blind tested.

An additional twist of our problem setting is that code is tested with limited computational resources. That is, for each task an arbitrary limit on execution time is fixed and a maximum amount of memory is provided. This places a constraint on the participant to produce a solution in a given time, and hence to optimize the model search from a computational point of view. In summary, participants have to jointly address the problem of over-fitting/under-fitting and the problem of efficient search for an optimal solution, as stated in (Jordan, 2013). In practice, the computational contraints have turned out to be far more challenging to challenge participants than the problem of overfitting. Thus the main contributions have been to devise novel efficient search techniques with cutting edge optimization methods.

2.4. Strategies of Model Search

Most practitioners use heuristics such as grid search or uniform sampling to sample $\boldsymbol{\theta}$ space, and use k-fold cross-validation as the upper-level objective J_2 (Dietterich, 1998). In this framework, the optimization of $\boldsymbol{\theta}$ is not performed sequentially (Bergstra and Bengio, 2012). All the parameters are sampled along a regular scheme, usually in linear or log scale. This leads to a number of possibilities that exponentially increases with the dimension of $\boldsymbol{\theta}$. k-fold cross-validation consists of splitting the dataset into k folds; $(k-1)$ folds are used for training and the remaining fold is used for testing; eventually, the average of the test scores obtained on the k folds is reported. Note that some ML toolkits currently support cross-validation. There is a lack of principled guidelines to determine the number of grid points and the value of k (with the exception of Dietterich (1998)), and there is no guidance for regularizing J_2, yet this simple method is a good baseline approach.

Efforts have been made to optimize continuous hyper-parameters with bilevel optimization methods, using either the k-fold cross-validation estimator (Bennett et al., 2008; Moore et al., 2011) or the leave-one-out estimator as the upper-level objective J_2. The leave-one-out estimator may be efficiently computed, in closed form, as a by-product of training only one predictor on all the training examples (*e.g.,* virtual-leave-one-out (Guyon et al., 2006b)). The method was improved by adding a regularization of J_2 (Cawley and Talbot, 2007). Gradient descent has been used to accelerate the search, by making a local quadratic approximation of J_2 (Keerthi et al., 2007). In some cases, the full $J_2(\boldsymbol{\theta})$ can be computed from a few key examples (Hastie et al., 2004; Park and Hastie, 2007). Other approaches minimize an *approximation* or an *upper bound* of the leave-one-out error, instead of its exact form (Opper and Winther, 2000; Vapnik and Chapelle, 2000). Nevertheless, these methods are still limited to specific models and continuous hyper-parameters.

An early attempt at full model selection was the *pattern search* method that uses k-fold cross-validation for J_2. It explores the hyper-parameter space by steps of the same magnitude, and when no change in any parameter further decreases J_2, the step size is halved and the process repeated until the steps are deemed sufficiently small (Momma and Bennett, 2002). (Escalante et al., 2009) addressed the full model selection problem using Particle Swarm Optimization, which optimizes a problem by having a population of candidate solutions (particles), and moving these particles around the hyper-parameter space using the particle's position and velocity. k-fold cross-validation is also used for J_2. This approach retrieved the winning model in $\sim 76\%$ of the cases. Overfitting was controlled heuristically with early stopping and the proportion of training and validation data was not optimized. Although progress has been made in experimental design to reduce the risk of overfitting (Ioannidis, 2005; Langford, 2005), in particular by splitting data in a principled way (Statnikov et al., 2008), to our knowledge, no one has addressed the problem of optimally splitting data.

While regularizing the second level of inference is a recent addition to the frequentist ML community, it has been an intrinsic part of Bayesian modeling via the notion of hyper-prior. Some methods of multi-level optimization combine importance sampling and Monte-Carlo Markov Chains (Andrieu et al., 1999). The field of Bayesian hyper-parameter optimization has rapidly developed and yielded promising results, in particular by using Gaussian processes to model generalization performance (Snoek et al., 2012; Swersky et al., 2013).

But Tree-structured Parzen Estimator (TPE) approaches modeling $P(\mathbf{x}|y)$ and $P(y)$ rather than modeling $P(y|\mathbf{x})$ directly (Bergstra et al., 2011, 2013) have been found to outperform GP-based Bayesian optimization for structured optimization problems with many hyper-parameters including discrete ones (Eggensperger et al., 2013). The central idea of these methods is to fit $J_2(\boldsymbol{\theta})$ to a smooth function in an attempt to reduce variance and to estimate the variance in regions of the hyper-parameter space that are under-sampled to guide the search towards regions of high variance. These methods are inspirational and some of the ideas can be adopted in the frequentist setting. For instance, the random-forest-based SMAC algorithm (Hutter et al., 2011), which has helped speed up both local search and tree search algorithms by orders of magnitude on certain instance distributions, has also been found to be very effective for the hyper-parameter optimization of machine learning algorithms, scaling better to high dimensions and discrete input dimensions than other algorithms (Eggensperger et al., 2013). We also notice that Bayesian optimization methods often combine with other techniques such as meta-learning and ensemble methods (Feurer et al., 2015b) in order to gain advantage in some challenge settings with time budget limit (Guyon et al., 2015a). Some of these methods consider jointly the two-level optimization and take time cost as a critical guidance for hyper-parameter search (Swersky et al., 2014; Klein et al., 2017).

Besides Bayesian optimization, several other families of approaches exist in the literature and have gained much attention with the recent rise of deep learning. Ideas borrowed from *reinforcement learning* have recently been used to construct optimal neural network architectures (Zoph and Le, 2016; Baker et al., 2016). These approaches formulate the hyper-parameter optimization problem in a reinforcement learning flavor, with for example states being the actual hyper-parameter setting (*e.g.,* network architecture), actions being added or deleting a module (*e.g.,* a CNN layer or a pooling layer), and reward being the validation accuracy. They can then apply off-the-shelf reinforcement learning algorithms (*e.g.,* RENFORCE, Q-learning, Monte-Carlo Tree Search) to solve the problem. Other architecture search methods use *evolutionary* algorithms (Real et al., 2017; Assunção et al., 2018). These approaches consider a set (population) of hyper-parameter settings (individuals), modify (mutate and reproduce) and eliminate unpromising settings according to their cross-validation score (fitness). After several generations, the global quality of the population increases. One important common point of reinforcement learning and evolutionary algorithms is that they both deal with the exploration-exploitation trade-off. Despite the impressive results, these approaches require huge amount of computational resources and some (especially evolutionary algorithms) are hard to scale. (Pham et al., 2018) recently proposed the weight sharing among child models to largely speed up (Zoph and Le, 2016) while achieving comparable results.

Note that splitting the problem of parameter fitting into two levels can be extended to multiple levels, at the expense of extra complexity—*i.e.,* need for a hierarchy of data splits to perform multiple or nested cross-validation (Efron, 1983), insufficient data to train and validate at the different levels, and increase of the computational load.

Table 1 shows a typical example of multi-level parameter optimization in a frequentist setting. We assume that we are using an ML toolbox with two learning machines: Kridge (kernel ridge regression) and Neural (a neural network a.k.a. "deep learning" model). At the top level we use a test procedure to assess the performance of the final model (this is not

Table 1:

Typical example of multi-level inference algorithm. The top-level algorithm Validation({GridCV(Kridge, MSE), GridCV(Neural, MSE)}, MSE) is decomposed into its elements recursively. Calling the method "train" on it using data D_{TrVa} results in a function f, then tested with test(f, MSE, D_{Te}). The notation $[.]_{CV}$ indicates that results are averages over multiple data splits (cross-validation). NA means "not applicable". A model family $\mathcal{F}$ of parameters $\boldsymbol{\alpha}$ and hyper-parameters $\boldsymbol{\theta}$ is represented as f($\boldsymbol{\theta}, \boldsymbol{\alpha}$). We derogate to the usual convention of putting hyper-parameters last, the hyper-parameters are listed in decreasing order of inference level. $\mathcal{F}$, thought of as a bottom level algorithm, does not perform any training: $train$(f($\boldsymbol{\theta}, \boldsymbol{\alpha}$)) just returns the function $f(\mathbf{x}; \boldsymbol{\theta}, \boldsymbol{\alpha})$.

Level	**Algorithm**	**Parameters**		**Optimization performed**	**Data split**
		Fixed	**Varying**		
NA	f	All	All	Performance assessment (no inference).	D_{Te}
4	Validation	None	All	Final algorithm selection using validation data.	$D = [D_{Tr}, D_{Va}]$
3	GridCV	model index i	$\boldsymbol{\theta}, \gamma, \boldsymbol{\alpha}$	10-fold CV on regularly sampled values of $\boldsymbol{\theta}$.	$D_{Tr} = [D_{tr}, D_{va}]_{CV}$
2	Kridge($\boldsymbol{\theta}$) Neural($\boldsymbol{\theta}$)	$i, \boldsymbol{\theta}$	$\gamma, \boldsymbol{\alpha}$	Virtual LOO CV to select regularization parameter γ	$D_{tr} = [D_{tr}^{\setminus\{d\}}, d]_{CV}$
1	Kridge($\boldsymbol{\theta}, \gamma$) Neural($\boldsymbol{\theta}, \gamma$)	$i, \boldsymbol{\theta}, \gamma$	$\boldsymbol{\alpha}$	Matrix inversion of gradient descent to compute $\boldsymbol{\alpha}$.	D_{tr}
0	Kridge($\boldsymbol{\theta}, \gamma, \boldsymbol{\alpha}$) Neural($\boldsymbol{\theta}, \gamma, \boldsymbol{\alpha}$)	All	None	NA	NA

an inference level). The top-level inference algorithm Validation({GridCV(Kridge, MSE), GridCV(Neural, MSE)}, MSE) is decomposed into its elements recursively. Validation uses the data split $D = [D_{Tr}, D_{Va}]$ to compare the learning machines Kridge and Neural (trained using D_{Tr} on the validation set D_{Va}, using the mean-square error (MSE) evaluation function. The algorithm GridCV, a grid search with 10-fold cross-validation (CV) MSE evaluation function, then optimizes the hyper-parameters $\boldsymbol{\theta}$. Internally, both Kridge and Neural use virtual leave-one-out (LOO) cross-validation to adjust γ and a classical L_2 regularized risk functional to adjust $\boldsymbol{\alpha}$.

Borrowing from the conventional classification of feature selection methods (Kohavi and John, 1997; Blum and Langley, 1997; Guyon et al., 2006b), model search strategies can be categorized into filters, wrappers, and embedded methods (see Figure 2). **Filters** are methods for narrowing down the model space, without training the learner. Such methods include preprocessing, feature construction, kernel design, architecture design, choice of prior or regularizers, choice of noise model, and filter methods for feature selection. Although some filters use training data, many incorporate human prior knowledge of the task or knowledge compiled from previous tasks. Recently, (Bardenet et al., 2013) proposed to apply collaborative filtering methods to model search. **Wrapper methods** consider learners as a black-box capable of learning from examples and making predictions once trained. They operate with a search algorithm in the hyper-parameter space (grid search or stochastic search) and an evaluation function assessing the trained learner's performance (cross-validation error or Bayesian evidence). **Embedded methods** are similar to wrappers, but they exploit the knowledge of the learning machine algorithm to make the search more efficient. For instance, some embedded methods compute the leave-one-out solution in a closed form, without leaving anything out, *i.e.,* by performing a single model training on all the training data (*e.g.,* (Guyon et al., 2006b)). Other embedded methods jointly optimize parameters and hyper-parameters (Keerthi et al., 2007; Moore et al., 2009, 2011).

In summary, many authors focus only on the efficiency of search, ignoring the problem of overfitting the second level objective J_2, which is often chosen to be k-fold cross-validation with an arbitrary value for k. Bayesian methods introduce techniques of overfitting avoidance via the notion of hyper-priors, but at the expense of making assumptions on how the data were generated and without providing guarantees of performance. In all the prior approaches to full model selection we know of, there is no attempt to treat the problem as the optimization of a regularized functional J_2 with respect to both (1) modeling choices and (2) data split. Much remains to be done to jointly address statistical and computational issues. The AutoML challenge series offers benchmarks to compare and contrast methods addressing these problems, free of the inventor/evaluator bias.

3. Data

We gathered a first pool of 70 datasets during the summer 2014 with the help of numerous collaborators and ended up selecting 30 datasets for the 2015/2016 challenge (see Table 2 and Appendix B), chosen to illustrate a wide variety of domains of applications: biology and medicine, ecology, energy and sustainability management, image, text, audio, speech, video and other sensor data processing, Internet social media management and advertising, market analysis and financial prediction. We preprocessed data to obtain feature representations

Rnd	DATASET	Task	Metric	Time	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
0	1 ADULT	multilabel	F1	300	3	1	0.16	0.011	1	0.5	9768	4884	34190	24	1424.58
0	2 CADATA	regression	R2	200	0	NaN	0	0	0	0.5	10640	5000	5000	16	312.5
0	3 DIGITS	multiclass	BAC	300	10	1	0.42	0	0	0.5	35000	20000	15000	1568	9.57
0	4 DOROTHEA	binary	AUC	100	2	0.46	0.99	0	0	0.5	800	350	800	100000	0.01
0	5 NEWSGROUPS	multiclass	PAC	300	20	1	1	0	0	0	3755	1877	13142	61188	0.21
1	1 CHRISTINE	binary	BAC	1200	2	1	0.071	0	0	0.5	2084	834	5418	1636	3.31
1	2 JASMINE	binary	BAC	1200	2	1	0.78	0	0	0.5	1756	526	2984	144	20.72
1	3 MADELINE	binary	BAC	1200	2	1	1.2e-06	0	0	0.92	3240	1080	3140	259	12.12
1	4 PHILIPPINE	binary	BAC	1200	2	1	0.0012	0	0	0.5	4664	1166	5832	308	18.94
1	5 SYLVINE	binary	BAC	1200	2	1	0.01	0	0	0.5	10244	5124	5124	20	256.2
2	1 ALBERT	binary	F1	1200	2	1	0.049	0.14	1	0.5	51048	25526	425240	78	5451.79
2	2 DILBERT	multiclass	PAC	1200	5	1	0	0	0	0.16	9720	4860	10000	2000	5
2	3 FABERT	multiclass	PAC	1200	7	0.96	0.99	0	0	0.5	2354	1177	8237	800	10.3
2	4 ROBERT	multiclass	BAC	1200	10	1	0.01	0	0	0	5000	2000	10000	7200	1.39
2	5 VOLKERT	multiclass	PAC	1200	10	0.89	0.34	0	0	0	7000	3500	58310	180	323.94
3	1 ALEXIS	multilabel	AUC	1200	18	0.92	0.98	0	0	0	15569	7784	54491	5000	10.9
3	2 DIONIS	multiclass	BAC	1200	355	1	0.11	0	0	0	12000	6000	416188	60	6936.47
3	3 GRIGORIS	multilabel	AUC	1200	91	0.87	1	0	0	0	9920	6486	45400	301561	0.15
3	4 JANNIS	multiclass	BAC	1200	4	0.8	7.3e-05	0	0	0.5	9851	4926	83733	54	1550.61
3	5 WALLIS	multiclass	AUC	1200	11	0.91	1	0	0	0	8196	4098	10000	193731	0.05
4	1 EVITA	binary	AUC	1200	2	0.21	0.91	0	0	0.46	14000	8000	20000	3000	6.67
4	2 FLORA	regression	ABS	1200	0	NaN	0.99	0	0	0.25	2000	2000	15000	200000	0.08
4	3 HELENA	multiclass	BAC	1200	100	0.9	6e-05	0	0	0	18628	9314	65196	27	2414.67
4	4 TANIA	multilabel	PAC	1200	95	0.79	1	0	0	0	44635	22514	157599	47236	3.34
4	5 YOLANDA	regression	R2	1200	0	NaN	1e-07	0	0	0.1	30000	30000	400000	100	4000
5	1 ARTURO	multiclass	F1	1200	20	1	0.82	0	0	0.5	2733	1366	9565	400	23.91
5	2 CARLO	binary	PAC	1200	2	0.097	0.0027	0	0	0.5	10000	10000	50000	1070	46.73
5	3 MARCO	multilabel	AUC	1200	24	0.76	0.99	0	0	0	20482	20482	163860	15299	10.71
5	4 PABLO	regression	ABS	1200	0	NaN	0.11	0	0	0.5	23565	23565	188524	120	1571.03
5	5 WALDO	multiclass	BAC	1200	4	1	0.029	0	1	0.5	2430	2430	19439	270	72

Table 2: **Datasets of the 2015/2016 AutoML challenge.** C=number of classes. Cbal=class balance. Sparse=sparsity. Miss=fraction of missing values. Cat=categorical variables. Irr=fraction of irrelevant variables. Pte, Pva, Ptr=number of examples of the test, validation, and training sets, respectively. N=number of features. Ptr/N=aspect ratio of the dataset.

Phase	DATASET	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
1	1 ADA	1	0.67	0	0	0	41471	415	4147	48	86.39
1	2 ARCENE	0.22	0.54	0	0	0	700	100	100	10000	0.01
1	3 GINA	1	0.03	0.31	0	0	31532	315	3153	970	3.25
1	4 GUILLERMO	0.33	0.53	0	0	0	5000	5000	20000	4296	4.65
1	5 RL	0.10	0	0.11	1	0	24803	0	31406	22	1427.5
2	1 PM	0.01	0	0.11	1	0	20000	0	29964	89	224.71
2	2 RH	0.04	0.41	0	1	0	28544	0	31498	76	414.44
2	3 RI	0.02	0.09	0.26	1	0	26744	0	30562	113	270.46
2	4 RICCARDO	0.67	0.51	0	0	0	5000	5000	20000	4296	4.65
2	5 RM	0.001	0	0.11	1	0	26961	0	28278	89	317.73

Table 3: **Datasets of the 2018 AutoML challenge.** All tasks are binary classification problems. The metric is the AUC for all tasks. The time budget is also the same for all datasets: 1200 s. Phase 1 was the development phase and phase 2 the final "blind test" phase.

(i.e., each example consists of a fixed number of numerical coefficients). Text, speech, and video processing tasks were included in the challenge, but not in their native variable length representations.

For the 2018 challenge, three datasets from the first pool (but unused in the first challenge) were selected and 7 new datasets collected by the new organizers and sponsors were added (see Table 3 and Appendix C).

Some datasets were obtained from public sources, but they were reformatted into new representations to conceal their identity, except for the final round of the 2015/2016 challenge and the final phase of the 2018 challenge, which included completely new data.

In the 2015/2016 challenge, data difficulty progressively increased from round to round. Round 0 introduced five (public) datasets from previous challenges illustrating the various difficulties encountered in subsequent rounds:

Novice. Binary classification problems only. No missing data; no categorical features; moderate number of features ($< 2,000$); balanced classes. Challenge lies in dealing with sparse and full matrices, presence of irrelevant variables, and various Ptr/N.

Intermediate. Binary and multi-class classification problems. Challenge lies in dealing with unbalanced classes, number of classes, missing values, categorical variables, and up to 7,000 features.

Advanced. Binary, multi-class, and multi-label classification problems. Challenge lies in dealing with up to 300,000 features.

Expert. Classification and regression problems. Challenge lies in dealing with the entire range of data complexity.

Master. Classification and regression problems of all difficulties. Challenge lies in learning from completely new datasets.

The datasets of the 2018 challenge were all binary classification problems. Validation partitions were not used because of the design of this challenge, even when they were

available for some tasks. The three reused datasets had similar difficulty as those of rounds 1 and 2 of the 2015/2016 challenge. However, the 7 new data sets introduced difficulties that were not present in the former challenge. Most notably an extreme class imbalance, presence of categorical features and a temporal dependency among instances that could be exploited by participants to develop their methods[9]. The datasets from both challenges are downloadable from http://automl.chalearn.org/data.

4. Challenge Protocol

In this section, we describe design choices we made to ensure the thoroughness and fairness of the evaluation. As previously indicated, we focus on supervised learning tasks (classification and regression problems), without any human intervention, within given time and computer resource constraints (Section 4.1), and given a particular metric (Section 4.2), which varies from dataset to dataset. During the challenges, the identity and description of the datasets is concealed (except in the very first round or phase where sample data is distributed) to avoid the use of domain knowledge and to push participants to design fully automated ML solutions. In the 2015/2016 AutoML challenge, the datasets were introduced in a series of rounds (Section 4.3), alternating periods of code development (Tweakathon phases) and blind tests of code without human intervention (AutoML phases). Either results or code could be submitted during development phases, but code had to be submitted to be part of the AutoML "blind test" ranking. In the 2018 edition of the AutoML challenge, the protocol was simplified. We had only one round in two phases: a development phase in which 5 datasets were released for practice purposes, and a final "blind test" phase with 5 new datasets that were never used before.

4.1. Time Budget and computational resources

The Codalab platform provides computational resources shared by all participants. We used up to 10 compute workers processing in parallel the queue of submissions made by participants. Each compute worker was equipped with 8 cores $x86\_64$. Memory was increased from 24 GB to 56 GB after round 3 of the 2015/2016 AutoML challenge. For the 2018 AutoML challenge computing resources were reduced, as we wanted to motivate the development of more efficient yet effective AutoML solutions. We used 6 compute workers processing in parallel the queue of submissions. Each compute worker was equipped with 2 cores $x86\_64$ and 8 GB of memory.

To ensure fairness, when a code submission was evaluated, a compute worker was dedicated to processing that submission only, and its execution time was limited to a given time budget (which may vary from dataset to dataset). The time budget was provided to the participants with each dataset in its *info* file. It was generally set to 1200 seconds (20 minutes) per dataset, for practical reasons, except in the first phase of the first round. However, the participants did not know this ahead of time and therefore their code had to be capable to manage a given time budget. The participants who submitted results —instead of code— were not constrained by the time budget since their code was run on

9. In RL, PM, RH, RI and RM datasets instances were chronologically sorted, this information was made available to participants and could be used for developing their methods.

their own platform. This was potentially advantageous for entries counting towards the Final phases (immediately following a Tweakathon). Participants wishing to also enter the AutoML (blind testing) phases, which required submitting code, could submit both results and code (simultaneously). When results were submitted, they were used as entries in the on-going phase. They did not need to be produced by the submitted code; *i.e.* if a participant did not want to share personal code, he/she could submit the sample code provided by the organizers together with his/her results. The code was automatically forwarded to the AutoML phases for "blind testing". In AutoML phases, result submission was not possible.

The participants were encouraged to save and submit intermediate results so we could draw learning curves. This was not exploited during the challenge. But we study learning curves in this paper to evaluate the capabilities of algorithms to quickly attain good performances.

4.2. Scoring Metrics

The scores are computed by comparing submitted predictions to reference target values. For each sample $i, i = 1 : P$ (where P is the size of the validation set or of the test set), the target value is a continuous numeric coefficient y_i for regression problems, a binary indicator in $\{0, 1\}$ for two-class problems, or a vector of binary indicators $[y_{il}]$ in $\{0, 1\}$ for multi-class or multi-label classification problems (one per class l). The participants had to submit prediction values matching as closely as possible the target values, in the form of a continuous numeric coefficient q_i for regression problems and a vector of numeric coefficients $[q_{il}]$ in the range $[0, 1]$ for multi-class or multi-label classification problems (one per class l).

The provided starting kit contains an implementation in Python of all scoring metrics used to evaluate the entries. Each dataset has its own scoring criterion specified in its *info* file. All scores are normalized such that the expected value of the score for a random prediction, based on class prior probabilities, is 0 and the optimal score is 1. Multi-label problems are treated as multiple binary classification problems and are evaluated using the average of the scores of each binary classification subproblem.

We first define the notation $\langle\cdot\rangle$ for the average over all samples P indexed by i. That is,

$$\langle y_i \rangle = (1/P) \sum_{i=1}^{P} (y_i). \tag{2}$$

The score metrics are defined as follows:

R^2. The coefficient of determination is used for regression problems only. The metric is based on the mean squared error (MSE) and the variance (VAR), and computed as

$$R^2 = 1 - MSE/VAR, \tag{3}$$

where $MSE = \langle (y_i - q_i)^2 \rangle$ and $VAR = \langle (y_i - m)^2 \rangle$, with $m = \langle y_i \rangle$.

ABS. This coefficient is similar to R^2 but based on the mean absolute error (MAE) and the mean absolute deviation (MAD), and computed as

$$ABS = 1 - MAE/MAD, \tag{4}$$

where $MAE = \langle \text{abs}(y_i - q_i) \rangle$ and $MAD = \langle \text{abs}(y_i - m) \rangle$.

BAC. Balanced accuracy is the average of class-wise accuracy for classification problems—and the average of *sensitivity* (true positive rate) and *specificity* (true negative rate) for binary classification:

$$BAC = \begin{cases} \frac{1}{2}[\frac{TP}{P} + \frac{TN}{N}], & \text{for binary} \\ \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{N_i}, & \text{for multi-class} \end{cases} \tag{5}$$

where P (N) is the number of positive (negative) examples, TP (TN) is the number of well classified positive (negative) examples, C is the number of classes, TP_i is the number of well classified examples of class i and N_i the number of examples of class i.

For binary classification problems, the class-wise accuracy is the fraction of correct class predictions when q_i is thresholded at 0.5, for each class. For multi-label problems, the class-wise accuracy is averaged over all classes. For multi-class problems, the predictions are binarized by selecting the class with maximum prediction value $\arg\max_l q_{il}$ before computing the class-wise accuracy.

We normalize the metric as follows:

$$|BAC| = (BAC - R)/(1 - R), \tag{6}$$

where R is the expected value of BAC for random predictions (i.e., $R = 0.5$ for binary classification and $R = (1/C)$ for C-class problems).

AUC. The area under the ROC curve is used for ranking and binary classification problems. The ROC curve is the curve of *sensitivity* vs. *1-specificity* at various prediction thresholds. The AUC and BAC values are the same for binary predictions. The AUC is calculated for each class separately before averaging over all classes. We normalize the metric as

$$|AUC| = 2AUC - 1. \tag{7}$$

F1 score. The harmonic mean of precision and recall is computed as

$$F1 = 2 * (precision * recall)/(precision + recall), \tag{8}$$

$$precision = true\ positive/(true\ positive + false\ positive) \tag{9}$$

$$recall = true\ positive/(true\ positive + false\ negative) \tag{10}$$

Prediction thresholding and class averaging is handled similarly as in BAC. We normalize the metric as follows:

$$|F1| = (F1 - R)/(1 - R), \tag{11}$$

where R is the expected value of F1 for random predictions (see BAC).

PAC. Probabilistic accuracy is based on the cross-entropy (or log loss) and computed as

$$PAC = \exp(-CE), \tag{12}$$

$$CE = \begin{cases} average\sum_l \log(q_{il}), & \text{for multi-class} \\ -\langle y_i \log(q_i), & \\ +(1 - y_i)\log(1 - q_i)\rangle, & \text{for binary and multi-label} \end{cases} \tag{13}$$

Class averaging is performed after taking the exponential in the multi-label case. We normalize the metric as follows:

$$|PAC| = (PAC - R)/(1 - R), \tag{14}$$

where R is the score obtained using $q_i = \langle y_i \rangle$ or $q_{il} = \langle y_{il} \rangle$ (i.e., using as predictions the fraction of positive class examples, as an estimate of the prior probability).

Note that the normalization of R^2, ABS, and PAC uses the average target value $q_i = \langle y_i \rangle$ or $q_{il} = \langle y_{il} \rangle$. In contrast, the normalization of BAC, AUC, and F1 uses a random prediction of one of the classes with uniform probability.

Only R^2 and ABS are meaningful for regression; we compute the other metrics for completeness by replacing the target values with binary values after thresholding them in the mid-range.

Table 4: **Phases of round n in the 2015/2016 challenge.** For each dataset, one labeled training set is provided and two unlabeled sets (validation set and test set) are provided for testing.

Phase in round [n]	**Goal**	**Duration**	**Submissions**	**Data**	**Leader-board scores**	**Prizes**
* AutoML[n]	Blind test of code	Short	NONE (code migrated)	New datasets, not downloadable	Test set results	Yes
Tweakathon[n]	Manual tweaking	Months	Code and/ or results	Datasets downloadable	Validation set results	No
* Final[n]	Results of Tweakathon revealed	Short	NONE (results migrated)	NA	Test set results	Yes

4.3. Rounds and Phases in the 2015/2016 challenge

The 2015/2016 challenge was run in multiple phases grouped in six rounds. Round 0 (Preparation) was a practice round using publicly available datasets. It was followed by five rounds of progressive difficulty (Novice, Intermediate, Advanced, Expert, and Master). Except for rounds 0 and 5, all rounds included three phases that alternated AutoML and Tweakathons contests. These phases are described in Table 4.

Submissions were made in Tweakathon phases only. The results of the latest submission were shown on the leaderboard and such submission automatically migrated to the following phase. In this way, the code of participants who abandoned before the end of the challenge had a chance to be tested in subsequent rounds and phases. New participants could enter at any time. Prizes were awarded in phases marked with a * during which there was no submission. To participate in phase AutoML[n], code had to be submitted in Tweakathon[n-1].

In order to encourage participants to try GPUs and deep learning, a GPU track sponsored by NVIDIA was included in Round 4.

To participate in the Final[n], code or results had to be submitted in Tweakathon[n]. If both code and (well-formatted) results were submitted, the results were used for scoring

rather than rerunning the code in Tweakathon[n] and Final[n]. The code was executed when results were unavailable or not well formatted. Thus, there was no disadvantage in submitting both results and code. If a participant submitted both results and code, different methods could be used to enter the Tweakathon/Final phases and the AutoML phases. Submissions were made only during Tweakathons, with a maximum of five submissions per day. Immediate feedback was provided on the leaderboard on validation data. The participants were ranked on the basis of test performance during the Final and AutoML phases.

We provided baseline software using the ML library Scikit-learn (Pedregosa et al., 2011). It uses ensemble methods, which improve over time by adding more base learners. Other than the number of base learners, the default hyper-parameter settings were used. The participants were not obliged to use the Python language nor the main Python script we gave as an example. However, most participants found convenient to use the main python script, which managed the sparse format, the any-time learning settings and the scoring metrics. Many limited themselves to search for the best model in the scikit-learn library. This shows the importance of providing a good starting kit, but also the danger of biasing results towards particular solutions.

4.4. Phases in the 2018 challenge

The 2015/2016 AutoML challenge was very long and few teams participated in all rounds. Further, even though there was no obligation to participate in previous rounds to enter new rounds, new potential participants felt they would be at a disadvantage. Hence, we believe it is preferable to organize recurrent yearly events, each with their own workshop and publication opportunity. This provides a good balance between competition and collaboration.

In 2018, we organized a single round of AutoML competition in two phases. In this simplified protocol, the participants could practice on five datasets during the first (development) phase, by either submitting code or results. Their performances were revealed immediately, as they became available, on the leaderboard.

The last submission of the development phase was automatically forwarded to the second phase: the AutoML "blind test" phase. In this second phase, which was the only one counting towards the prizes, the participants' code was automatically evaluated on five new datasets on the Codalab platform. The datasets were not revealed to the participants. Hence, submissions that did not include code capable of being trained and tested automatically were not ranked in the final phase and could not compete towards the prizes.

We provided the same starting kit as in the AutoML 2015/2016 challenge, but the participants also had access to the code of the winners of the previous challenge.

5. Results

This section provides a brief description of the results obtained during both challenges, explains the methods used by the participants and their elements of novelty, and provides the analysis of post-challenge experiments conducted to answer specific questions on the effectiveness of model search techniques.

Table 5: **Results of the 2015/2016 challenge winners.** $< R >$ is the average rank over all five data sets of the round and it was used to rank the participants. $< S >$ is the average score over the five data sets of the round. UP is the percent increase in performance between the average performance of the winners in the AutoML phase and the Final phase of the same round. The GPU track was run in round 4. Team names are abbreviated as follows: aad=aad_freiburg; djaj=djajetic; marc=marc.boulle; tadej=tadejs; abhi=abhishek4; ideal=ideal.intel.analytics; mat=matthias.vonrohr; lisheng=lise_sun; asml=amsl.intel.com; jlr44 = backstreet.bayes; post = postech.mlg_exbrain; ref=reference.

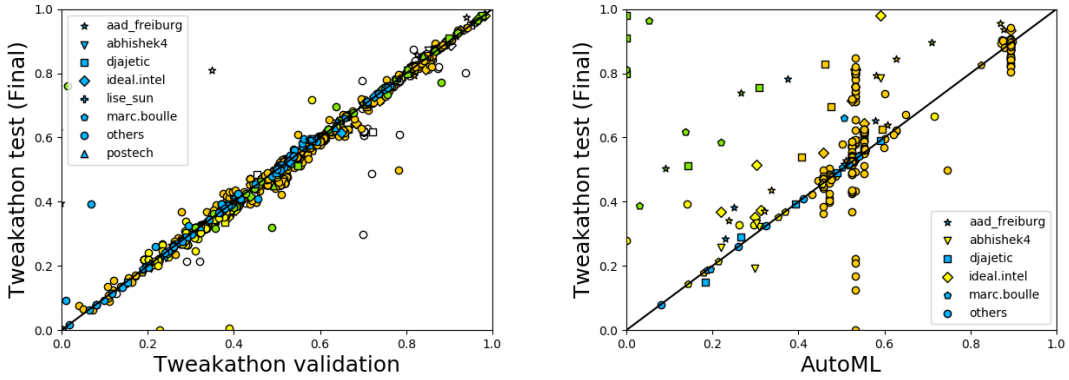
	AutoML				**Final**				
Rnd	Ended	Winners	$< R >$	$< S >$	Ended	Winners	$< R >$	$< S >$	UP (%)
0	NA	NA	NA	NA	02/14/15	1. ideal 2. abhi 3. aad	1.40 3.60 4.00	0.8159 0.7764 0.7714	NA
1	02/15/15	1. aad 2. jrl44 3. tadej	2.80 3.80 4.20	0.6401 0.6226 0.6456	06/14/15	1. aad 2. ideal 3. amsl	2.20 3.20 4.60	0.7479 0.7324 0.7158	15
2	06/15/15	1. jrl44 2. aad 3. mat	1.80 3.40 4.40	0.4320 0.3529 0.3449	11/14/15	1. ideal 2. djaj 3. aad	2.00 2.20 3.20	0.5180 0.5142 0.4977	35
3	11/15/15	1. djaj 2. NA 3. NA	2.40 NA NA	0.0901 NA NA	02/19/16	1. aad 2. djaj 3. ideal	1.80 2.00 3.80	0.8071 0.7912 0.7547	481
4	02/20/16	1. aad 2. djaj 3. marc	2.20 2.20 2.60	0.3881 0.3841 0.3815	05/1/16	1. aad 2. ideal 3. abhi	1.60 3.60 5.40	0.5238 0.4998 0.4911	31
GPU	NA	NA	NA	NA	05/1/16	1. abhi 2. djaj 3. aad	5.60 6.20 6.20	0.4913 0.4900 0.4884	NA
5	05/1/16	1. aad 2. djaj 3. post	1.60 2.60 4.60	0.5282 0.5379 0.4150	NA	NA	NA	NA	NA

Table 6: **Results of the 2018 challenge winners.** Each phase was run on 5 different datasets. We show the winners of the AutoML (blind test) phase and for comparison their performances in the Feedback phase. The full tables can be found at https://competitions.codalab.org/competitions/17767.

2. AutoML phase				**1. Feedback phase**			
Ended	Winners	$< R >$	$< S >$	Ended	Performance	$< R >$	$< S >$
03/31/18	1. aad_freiburg 2. narnars0 3. wlWangl 3. thanhdng 3. Malik	2.80 3.80 5.40 5.40 5.40	0.4341 0.4180 0.3857 0.3874 0.3863	03/12/18	aad_freiburg narnars0 wlWangl thanhdng Malik	9.0 4.40 4.40 14.0 13.8	0.7422 0.7324 0.8029 0.6845 0.7116

5.1. Scores obtained in the 2015/2016 challenge

The 2015/2016 challenge lasted 18 months (Dec. 8, 2014 to May 1, 2016). By the end of the challenge, practical solutions were obtained and open-sourced, such as the solution of the winners (Feurer et al., 2015b).

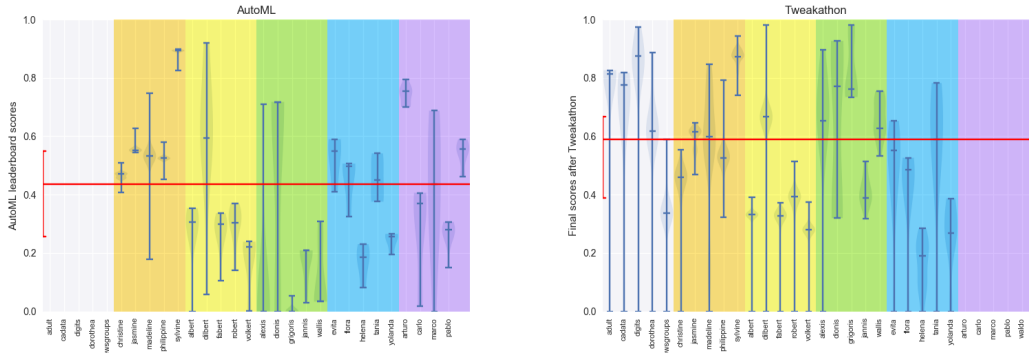


(*a*) Leaderboard overfitting? (*b*) Gap AutoML/Tweakathon?

Figure 3: **Performances of all participants in the 2015/2016 challenge.** We show the last entry of all participants in all phases of the 2015/2016 challenge on all datasets from the competition leaderboards. The symbols are color coded by round, as in Table 5. (a) **Overfitting in Tweakathons?** We plot the performance on the final test set vs. the performance on the validation set. The validation performances were visible to the participants on the leaderboard while they were tuning their models. The final test set performances were only revealed at the end of the Tweakathon. Except for a few outliers, most participants did not overfit the leaderboard. (b) **Gap between AutoML and Tweakathons?** We plot the Tweakathons vs. AutoML performance to visualize improvements obtained by manual tweaking and additional computational resources available in Tweakathons. Points above the diagonal indicate such improvements.

Table 5 presents the results on the test set in the AutoML phases (blind testing) and the Final phases (one time testing on the test set revealed at the end of the Tweakathon phases). Ties were broken by giving preference to the participant who submitted first. The table only reports the results of the top ranking participants. We also show in Figure 3 a comparison of the leaderboard performances of all participants. We plot in Figure 3(a) the Tweakathon performances on the final test set *vs.* those on the validation set, which reveals no significant overfitting to the validation set, except for a few outliers. In Figure 3(b) we report the performance in AutoML result (blind testing) *vs.* Tweakathon final test results (manual adjustments possible). We see that many entries were made in phase 1 (binary classification) and then participation declined as the tasks became harder. Some participants put a lot of effort in Tweakathons and far exceeded their AutoML performances (e.g. Djajetic and AAD Freiburg).

There is still room for improvement, as revealed by the significant differences remaining between Tweakathon and AutoML (blind testing) results (Table 5 and Figure 3-b). In Round 3, all but one participant failed to turn in working solutions during blind testing, because of the introduction of sparse datasets. Fortunately, the participants recovered, and, by the end of the challenge, several submissions were capable of returning solutions on all the datasets of the challenge. But learning schemas can still be optimized because,



(*a*) AutoML (test set) (*b*) Tweakathon (test set)

Figure 4: **Distribution of performance on the datasets of the 2015/2016 challenge (violon plots).** We show for each dataset the performances of participants at the end of AutoML and Tweakathon phases, as revealed on the leaderboard. The median and quartiles are represented by horizontal notches. The distribution profile (as fitted with a kernel method) and its mirror image are represented vertically by the gray shaded area. This is a slightly more refined version of the classical box plots. We show in red the median perfromance over all datasets and the corresponding quartiles. (a) **AutoML (blind testing).** The first 5 datasets were provided for development purpose only and were not used for blind testingin an AutoML phase. In round 3, the code of many participants failed because of computational limits. (b) **Tweakathon (manual tweaking).** The last five datasets were only used for final blind testing and the data were never revealed for a Tweakathon. Round 3 was not particularly difficult with additional compute power and memory.

even discarding Round 3, there is a 15 to 35% performance gap between AutoML phases (blind testing with computational constraints) and Tweakathon phases (human intervention and additional compute power). The GPU track offered (in round 4 only) a platform for trying Deep Learning methods. This allowed the participants to demonstrate that, given additional compute power, deep learning methods were competitive with the best solutions of the CPU track. However, no Deep Learning method was competitive with the limited compute power and time budget offered in the CPU track.

5.2. Scores obtained in the 2018 challenge

The 2018 challenge lasted 4 months (Nov. 30, 2017 to March 31, 2018). As in the previous challenge, top ranked solutions were obtained and open sourced. Table 6 shows the results of both phases of the 2018 challenge. As a reminder, this challenge had a feedback phase and a blind test phase, the performances of the winners in each phase are reported.

Performance in this challenge was slightly lower than that observed in the previous edition. This was due to the difficulty of the tasks (see below) and the fact that data sets in the feedback phase included three deceiving datasets (associated to tasks from previous challenges, but not necessarily similar to the data sets used in the blind test phase) out

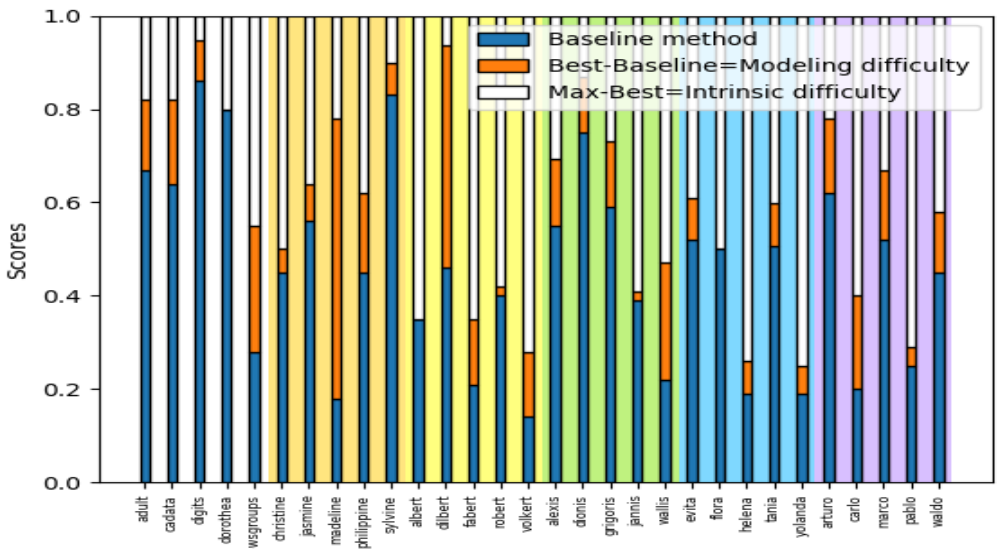


Figure 5: **Difficulty of tasks in the 2015/2016 challenge.** We consider two indicators of task difficulty (dataset, metric, and time budget are factored into the task): Intrinsic difficulty estimated by (1-the performance of the winners) and modeling difficulty (difference between the performance of the winner and the a baseline method, here SNB=Selective Naive Bayes). The best tasks should have a relatively low intrinsic difficulty and a high modeling difficulty to separate well participants.

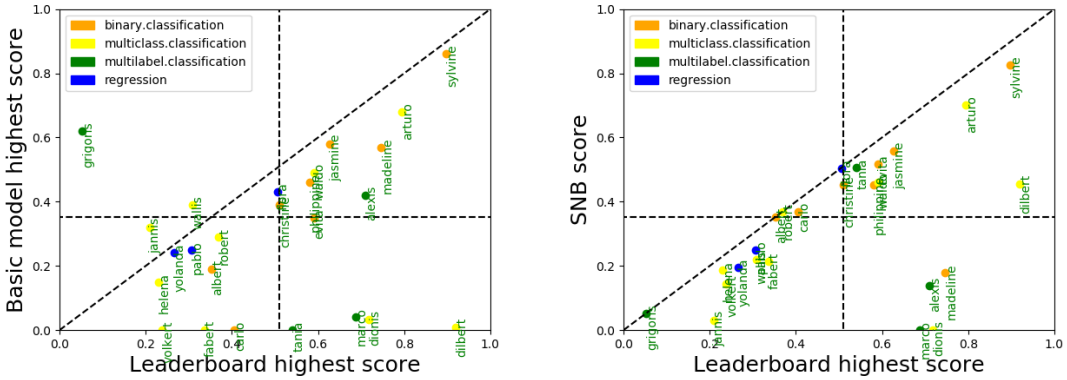
of five. We decided to proceed this way to emulate a realistic AutoML setting. Although harder, several teams succeeded at returning submissions performing better than chance.

The winner of the challenge was the same team that won the 2015/2016 AutoML challenge: AAD Freiburg (Feurer et al., 2018). Hence, the 2018 challenge helped to incrementally improve the solution devised by such team in the former challenge edition. Interestingly, the second place of the challenge proposed a solution that is similar in spirit to that of the winning team. For this challenge, there was a triple tie in the third place, prizes were split among the tied teams. Among the winners, two teams used the starting kit. Among other teams that participated in the challenge, most of them used either the starting kit or the solution open sourced by the AAD Freiburg team in the 2015/2016 challenge.

5.3. Difficulty of datasets/tasks

In this section, we assess dataset difficulty, or rather *task difficulty* since the participants had to solve prediction problems for given datasets, performance metrics, and computational time constraints. The tasks of the challenge presented a variety of difficulties, but those were not equally represented (Tables 2,3):

- **Categorical variables and missing data:** Few datasets had categorical variables in the 2015/2016 challenge (ADULT, ALBERT, and WALDO), and not very many



(*a*) Leaderboard overfitting? (*b*) Gap AutoML/Tweakathon?

Figure 6: **Modeling Difficulty *vs.* intrinsic difficulty**. For the AutoML phases of the 2015/2016 challenge, we plot an indicator of modeling difficulty *vs.* and indicator of intrinsic difficulty of datasets (leaderboard highest score). (a) Modeling difficulty is estimated by the score of the best untuned model (over KNN, NaiveBayes, RandomForest and SGD(LINEAR)). (b) Modeling difficulty is estimated by the score of the Selective Naive Bayes (SNB) model. In all cases, higher scores are better and negative / NaN scores are replaced by zero. The horizontal and vertical separation lines represent the medians. The lower right quadrant represents the datasets with low intrinsic difficulty and high modeling difficulty: those are the best datasets for benchmarking purposes.

variables were categorical in those datasets. Likewise, very few datasets had missing values (ADULT and ALBERT) and those included only a few missing values. So neither categorical variables nor missing data presented a real difficulty in this challenge, though ALBERT turned out to be one of the most difficult datasets because it was also one of the largest ones. This situation changed drastically for the 2018 challenge where five out of the ten datasets included categorical variables (RL, PM, RI, RH and RM) and missing values (GINA, PM, RL, RI and RM). These were among the main aspects that caused the low performance of most methods in the blind test phase.

- **Large number of classes.** Only one dataset had a large number of classes (DIONIS with 355 classes). This dataset turned out to be difficult for participants, particularly because it is also large and has unbalanced classes. However, datasets with large number of classes are not well represented in this challenge. HELENA, which has the second largest number of classes (100 classes), did not stand out as a particularly difficult dataset. However, in general, multi-class problems were found more difficult than binary classification problems.

- **Regression.** We had only four regression problems: CADATA, FLORA, YOLANDA, PABLO.

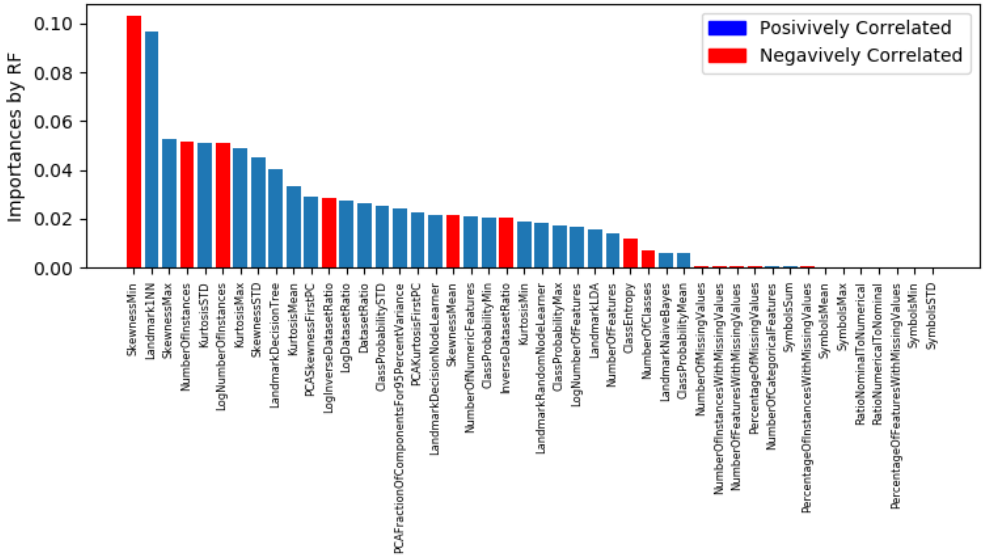


Figure 7: **Meta-features most predictive of dataset intrinsic difficulty (2015/2016 challenge data).** Meta-feature GINI importances are computed by a random forest regressor, trained to predict the highest participant leaderboard score using meta-features of datasets. Description of these meta-features can be found in Table 1 of the supplementary material of Feurer et al. (2015b). Blue and red colors respectively correspond to positive and negative correlations (Pearson correlations between meta features and score medians).

- **Sparse data.** A significant number of datasets had sparse data (DOROTHEA, FABERT, ALEXIS, WALLIS, GRIGORIS, EVITA, FLORA, TANIA, ARTURO, MARCO). Several of them turned out to be difficult, particularly ALEXIS, WALLIS, and GRIGORIS, which are large datasets in sparse format, which cause memory problems when they were introduced in round 3 of the 2015/2016 challenge. We later increased the amount of memory on the servers and similar datasets introduced in later phases caused less difficulty.

- **Large datasets.** We expected the ratio of the number N of features over the number P_{tr} of training examples to be a particular difficulty (because of the risk of overfitting), but modern machine learning algorithm are robust against overfitting. The main difficulty was rather the PRODUCT $N * P_{tr}$. Most participants attempted to load the entire dataset in memory and convert sparse matrices into full matrices. This took very long and then caused loss in performances or program failures. Large datasets with $N * P_{tr} > 20.10^6$ include ALBERT, **ALEXIS**, DIONIS, **GRIGORIS, WALLIS, EVITA, FLORA, TANIA, MARCO**, GINA, GUILLERMO, PM, RH, RI, RICCARDO, RM. Those overlap significantly with the datasets with sparse data (in

bold). For the 2018 challenge, all data sets in the final phase exceeded this threshold, and this was the reason of why the code from several teams failed to finish within the time budget. Only ALBERT and DIONIS were "truly" large (few features, but over 400,000 training examples).

- **Presence of probes:** Some datasets had a certain proportion of distractor features or irrelevant variables (probes). Those were obtained by randomly permuting the values of real features. Two-third of the datasets contained probes ADULT, CADATA, DIGITS, DOROTHEA, CHRISTINE, JASMINE, MADELINE, PHILIPPINE, SYLVINE, ALBERT, DILBERT, FABERT, JANNIS, EVITA, FLORA, YOLANDA, ARTURO, CARLO, PABLO, WALDO. This allowed us in part to make datasets that were in the public domain less recognizable.

- **Type of metric:** we used 6 metrics, as defined in section 4.2. The distribution of tasks in which they were used was not uniform: BAC (11), AUC (6), F1 (3), and PAC (6) for classification, and R2 (2) and ABS (2) for regression. This is because not all metrics lend themselves naturally to all types of applications.

- **Time budget:** Although in round 0 we experimented with giving different time budgets for the various datasets, we ended up assigning 1200 seconds (20 min) to all datasets in all other rounds. Because the datasets varied in size, this put more constraints on large datasets.

- **Class imbalance:** This was not a difficulty found in the 2015/2016 datasets. However, extreme class imbalance was the main difficulty for the 2018 edition. Imbalance ratios lower or equal to 1 to 10 were present in RL, PM, RH, RI, and **RM** datasets, in the latter data set class imbalance was as extreme as 1 to 1000. This was the reason why the performance of teams was low.

Figure 4 gives a first view of dataset/task difficulty for the 2015/2016 challenge. It captures, in a schematic way, the distribution of the participants' performance in all rounds on test data, in both AutoML and Tweakathon phases. One can see that the median performance over all datasets improves between AutoML and Tweakathon, as can be expected. Correspondingly, the average spread in performance (quartile) decreases. Let us take a closer look at the AutoML phases: The "accident" of round 3 in which many methods failed in blind testing is visible (introduction of sparse matrices and larger datasets)[10]. Round 2 (multi-class classification) appears to have also introduced a significantly higher degree of difficulty than round 1 (binary classification). In round 4, two regression problems were introduced (FLORA and YOLANDA), but it does not seem that regression was found significantly harder than multiclass classification. In round 5 no novelty was introduced. We can observe that, after round 3, the dataset median scores are scattered around the overall median. Looking at the corresponding scores in the Tweakathon phases, one can remark that, once the participants recovered from their surprise, round 3 was not particularly difficult for them. Rounds 2 and 4 were comparatively more difficult.

For the datasets used in the 2018 challenge, the tasks difficulty was clearly associated to extreme class imbalance, inclusion of categorical variables and high dimensionality in

10. Examples of sparse datasets were provided in round 0, but they were of smaller size.

terms of $N \times P_{tr}$. However, for the 2015/2016 challenge data sets we found that is was generally difficult to guess what makes a task easy or hard, except for dataset size, which pushed participants to the frontier of the hardware capabilities and forced them to improve the computational efficacy of their methods. Binary classification problems (and multi-label problems) are intrinsically "easier" than multiclass problems, for which "guessing" has a lower probablity of success. This partially explains the higher median performance in rounds 1 and 3, which are dominated by binary and multi-label classification problems. There is not a large enough number of datasets illustrating each type of other difficulties to draw other conclusions.

We ventured however to try to find summary statistics capturing overall taks difficulty. If one assumes that data are generated from an *i.i.d.*[11] process of the type:

$$y = F(\mathbf{x}, noise)$$

where y is the target value, $\mathbf{x}$ is the input feature vector, F is a function, and *noise* is some random noise drawn from an unknown distribution, then the difficuty of the learning problem can be separated in two aspects:

1. **Intrinsic difficulty**, linked to the amount of noise or the signal to noise ratio. Given an infinite amount of data and an unbiased learning machine $\hat{F}$ capable of identifying F, the prediction performances cannot exceed a given maximum value, corresponding to $\hat{F} = F$.

2. **Modeling difficulty**, linked to the bias and variance of estimators $\hat{F}$, in connection with the limited amount of training data and limited computational resources, and the possibly large number or parameters and hyper-parameters to estimate.

Evaluating the intrinsic difficulty is impossible unless we know F (but we don't). Our best approximation of F is the winners' solution. **We therefore use the winners' performance as an estimator of the best achievable performance.** This estimator may have both bias and variance: it is possibly biased because the winners may be under-fitting training data; it may have variance because of the limited amount of test data. Under-fitting is difficult to test. Its symptoms may be that the variance or the entropy of the predictions are less than those of the target values.

Evaluating the modeling difficulty is also impossible unless we know F and the model class. In the absence of knowledge on the model class, data scientists often use generic predictive models, agnostic with respect to the data generating process. Such models range from very basic highly biased models towards "simplicity" and smoothness of predictions (e.g. regularized linear models) to highly versatile unbiased models that can learn any function given enough data (e.g. ensembles of decision trees). To indirectly assess modeling difficulty, we resorted to use the difference in performance between *the method of the challenge winner* and that of (a) the best of four "untuned" basic models (taken from classical techniques provided in the scikit-learn library (Pedregosa et al., 2011) with default hyper-parameters) or (b) Selective Naive Bayes (SNB) (Boullé, 2007, 2009), a highly regularized model (biased towards simplicity), providing a very robust and simple baseline.

11. Independently and Identically Distributed samples.

Figures 5 and 6 give representations of our estimates of intrinsic and modeling difficulties for the 2015/2016 challenge datasets. It can be seen that the datasets of round 0 were among the easiest (except perhaps NEWSGROUP). Those were relatively small (and well-known) datasets. Surprisingly, the datasets of round 3 were also rather easy, despite the fact that most participants failed on them when they were introduced (largely because of memory limitations: scikit-learn algorithms were not optimized for sparse datasets and it was not possible to fit in memory the data matrix converted to a dense matrix). Two datasets have a small intrinsic difficulty but a large modeling difficulty: MADELINE and DILBERT. MADELINE is an artificial dataset very non-linear (clusters or 2 classes positionned on the vertices of a hyper-cube in a 5 dimensional space) so very hard for Naive Bayes and DILBERT is an image recognition dataset with images of objects rotated in all sorts of positions, also very hard for Naive Bayes. The datasets of the last 2 phases seem to have a large intrinsic difficulty compared to the modeling difficulty. But this can be deceiving because the datasets are new to the machine learning community and the performances of the winners may still be far from the best attainable performance.

We attempted to predict the intrinsic difficulty (as measured by the winners' performance) from the set of meta features used by AAD Freiburg for meta-learning (Feurer et al., 2015b), which are part of OpenML (Vanschoren et al., 2014), using a Random Forest classifier and ranked the meta features in order of importance (most selected by RF). The list of meta features is provided in Appendix A. The three meta-features that predict best dataset difficulty (Figure 7) are:

- LandmarkDecisionTree: performance of the decision tree classifier.
- Landmark1NN: performance of the nearest neighbor classifier.
- SkewnessMin: min over skewness of all features. Skewness measures the symmetry of a distribution. A positive skewness valuemeans that there is more weight in the left tail of the distribution.

5.4. Hyper-parameter optimization

Many participants used the scikit-learn (sklearn) package, including the winning group AAD Freiburg, which produced the auto-sklearn software. We used the auto-sklearn API to conduct post-challenge systematic studies of the effectiveness of hyper-parameter optimization. We compared the performances obtained with default hyper-parameter settings in scikit-learn and with hyper-parameters optimized with auto-sklearn[12], both within the time budgets as imposed during the challenge, for four "representative" basic methods: k-nearest neighbors (KNN), naive Bayes (NB), Random Forest (RF), and a linear model trained with stochastic gradient descent (SGD-linear[13]). The results are shown in Figure 8. We see that hyper-parameter optimization usually improves performance, but not always. The advantage of hyperparameter tuning comes mostly from its flexibility of switching the optimization metric to the one imposed by the task and from finding hyperparameters that work well given the current dataset and metric. However, in some cases it was not possible

12. we use sklearn 0.16.1 to mimic the challenge environment, and auto-sklearn 0.4.0.
13. we set the loss of SGD to be 'log' in scikit-learn for these experiments

to perform hyperparameter optimization within the time budget due to the data set size (score ≤ 0). Thus, there remains future work on how to perform thorough hyperparameter tuning given rigid time constraints and huge datasets.

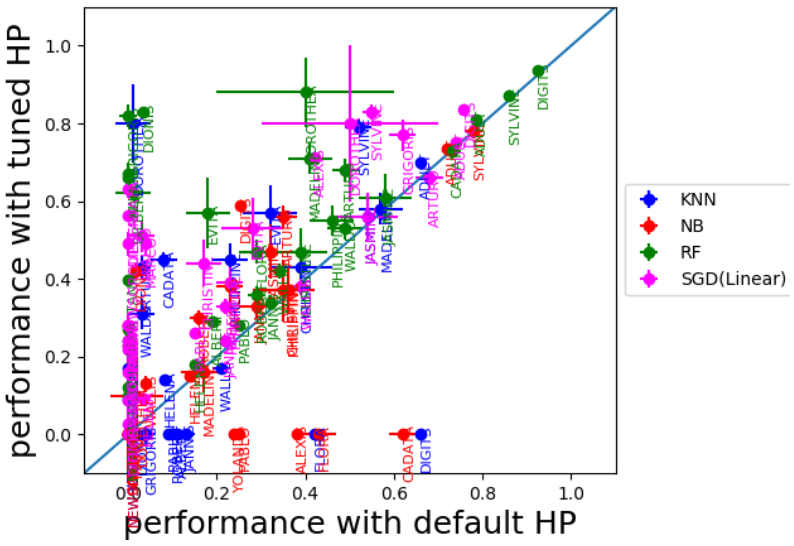
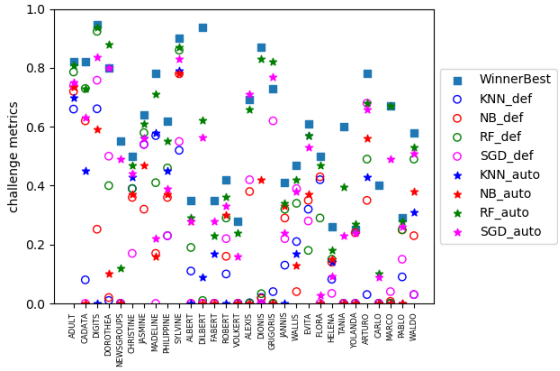
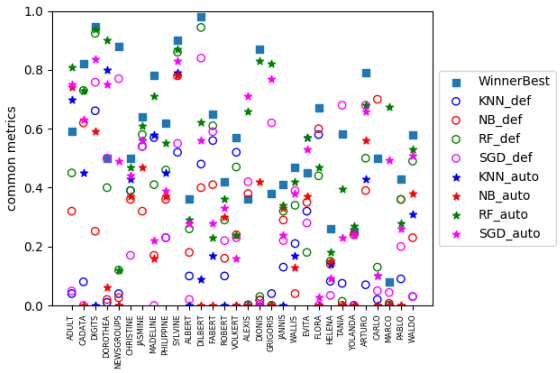


Figure 8: **Hyper-parameter tuning (2015/2016 challenge data).** We compare the performances obtained with default hyper-parameters and those with hyper-parameters optimized with auto-sklearn, within the same time budgets as given during the challenge. The performances of predictors which failed to return results in the alloted time are replaced by zero. Note that returning a prediction of chance level also resulted in a score of zero.

We also compared the performances obtained with different scoring metrics (Figure 9). Basic methods do not give a choice of metrics to be optimized, but auto-sklearn post-fitted the metrics of the challenge tasks. Consequently, when "common metrics" (BAC and R^2) are used, the method of the challenge winners, which is not optimized for BAC/R^2, does not usually outperfom basic methods. Conversely, when the metrics of the challenge are used, there is often a clear gap between the basic methods and the winners, but not always (RF-auto usually shows a comparable performance, sometimes even outperforms the winners).



(*a*) Challenge metrics

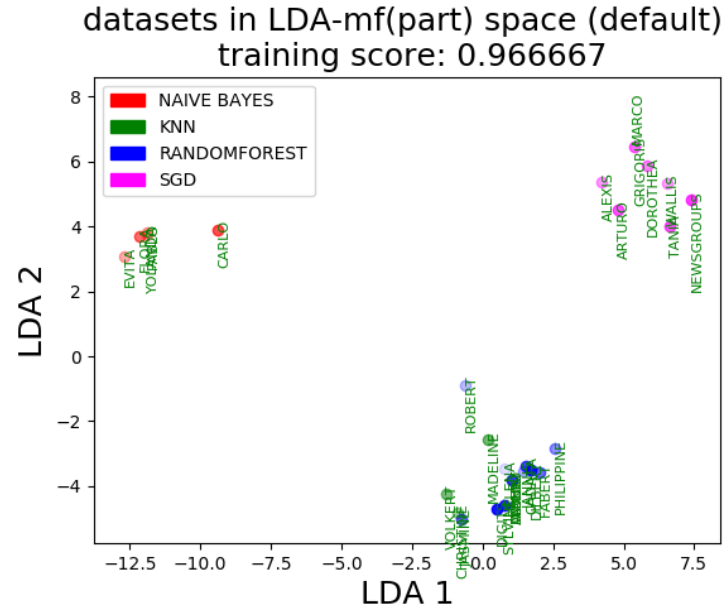


(*b*) Common metrics

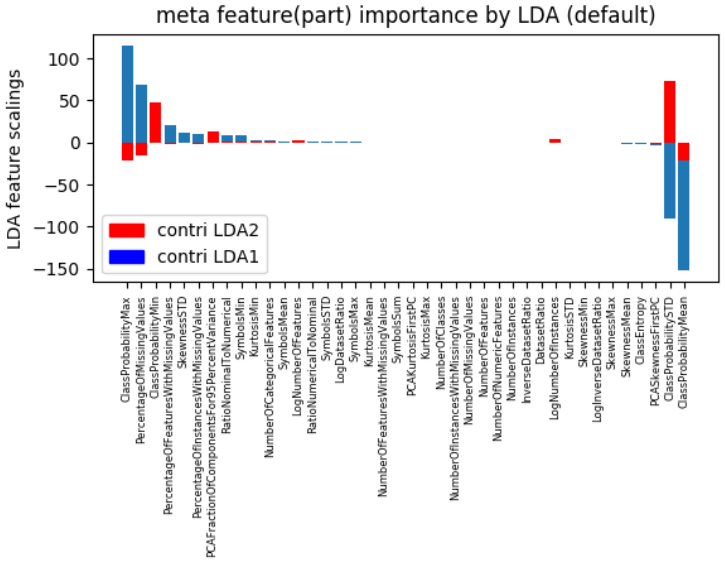
Figure 9: **Comparison of metrics (2015/2016 challenge).** (a) We used the metrics of the challenge. (b) We used the normalized balanced accuracy for all classification problems and the R^2 metric for regression problems. By comparing the two figures, we can see that the winner remains top ranking in most cases, regardless of the metric. There is no basic method that dominates all others. Although RF-auto (Random Forest with optimized HP) is very strong, it is sometimes outperformed by other methods. Plain linear model SGD-def sometimes wins when common metrics are used, but the winners perform better with the metrics of the challenge. Overall, the technique of the winners proved to be effective.

5.5. Meta-learning

One question is whether meta-learning (Brazdil et al., 2008) is possible, that is learning to predict whether a given classifier will perform well on future datasets (without actually training it), based on its past performances on other datasets. We investigated whether it is possible to predict which basic method will perform best based on the meta-learning



(a)



(b)

Figure 10: **Linear Discriminant Analysis.** (a) **Dataset scatter plot in principal axes**. We have trained a LDA using (X=meta features, except landmarks; y=which model won of four basic models (NB, SGD-linear, KNN, RF). The performance of basic models are measured using the common metrics. 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). (b) **Meta feature importances** computed as scaling factors of each LDA component

features of auto-sklearn (see Appendix A). We removed the "Landmark" features from the set of meta features because those are performances of basic predictors (albeit rather poor ones with many missing values), which would lead to a form of "data leakage".

We used four basic predictors:

- NB: Naive Bayes
- SGD-linear: Linear model (trained with stochastic gradient descent)
- KNN: K-nearest neighbors
- RF: Random Forest

We used the implementation of the scikit-learn library and with default hyper-parameter settings. In Figure 10, we show the two first Linear Discriminant Analysis (LDA) components, when training an LDA classifier on the meta-features to predict which basic classifier will perform best. The methods separate in three well distinct clusters, one of them grouping the non-linear methods that are poorly separated (KNN and RF) and the two others being NB and linear-SGD.

The features that are most predictive all have to do with "ClassProbability" and "PercentageOfMissingValues", indicating that the class imbalance and/or large number of classes (in a multi-class problem) and the percentage of missing value might be important, but there is a high chance of overfitting as indicated by an instability of the top ranking features under resampling of the training data.

5.6. Methods used in the challenges

A brief **description of methods** used in both challenges is provided in Appendices D and E, together with the results of a survey on methods that we conducted after the challenges. In light of the overview of Section 2 and the results presented in the previous section, we may wonder whether a dominant methodology for solving the AutoML problem has emerged and whether particular technical solutions were widely adopted. In this section we call "model space" the set of all models under consideration. We call "basic models" (also called elsewhere "simple models", "individual models", "base learners") the member of a library of models from which our hyper-models of model ensembles are built.

Ensembling: dealing with over-fitting and any-time learning. Ensembling is the big AutoML challenge series winner since it is used by over 80% of the participants and by all the top ranking ones. While a few years ago the hottest issue in model selection and hyper-parameter optimization was over-fitting, in present days the problem seems to have been largely avoided by using ensembling techniques. In the 2015/2016 challenge, we varied the ratio of number of training examples over number of variables (Ptr/N) by several orders of magnitude. Five datasets had a ratio Ptr/N lower than one (dorothea, newsgroup, grigoris, wallis, and flora), which is a case lending itself particularly to over-fitting. Although Ptr/N is the most predictive variable of the median performance of the participants, there is no indication that the datasets with $Ptr/N < 1$ were particularly difficult to the participants (Figure 5). Ensembles of predictors have the additional benefit of addressing in a simple way the "any-time learning" problem by growing progressively a

bigger ensemble of predictors, improving performance over time. All trained predictors are usually incorporated in the ensemble. For instance, if cross-validation is used, the predictors of all folds are directly incorporated in the ensemble, which saves the computational time of retraining a single model on the best HP selected and may yield more robust solutions (though slightly more biased due to the smaller sample size). The approaches differ in the way they weigh the contributions of the various predictors. Some methods use the same weight for all predictors (this is the case of bagging methods such as Random Forest and of Bayesian methods that sample predictors according to their posterior probability in model space). Some methods assess the weights of the predictors as part of learning (this is the case of boosting methods, for instance). One simple and effective method to create ensembles of heterogenous models was proposed by (Caruana et al., 2004). It was used successfully in several past challenges, *e.g.* (Niculescu-Mizil et al., 2009) and is the method implemented by the *aad_freibug* team, one of the strongest participants in both challenges (Feurer et al., 2015b). The method consists in cycling several times over all trained model and incorporating in the ensemble at each cycle the model which most improves the performance of the ensemble. Models vote with weight 1, but they can be incorporated multiple times, which de-facto results in weighting them. This method permits to recompute very fast the weights of the models if cross-validated predictions are saved. Moreover, the method allows optimizing the ensemble for any metric by post-fitting the predictions of the ensemble to the desired metric (an aspect, which was important in this challenge).

Model evaluation: cross-validation or simple validation. Evaluating the predictive accuracy of models is a critical and necessary building block of any model selection of ensembling method. Model selection criteria computed from the predictive accuracy of basic models evaluated from training data, by training a single time on all the training data (possibly at the expense of minor additional calculations), such as performance bounds, were not used at all, as was already the case in previous challenges we organized (Guyon et al., 2006a).Cross-validation was widely used and particularly K-fold cross-validation. However, basic models were often "cheaply" evaluated on just one fold to allow quickly discarding non promising areas of model space. This is a technique used more and more frequently to help speed-up search. Another speed-up strategy is to train on a subset of the training examples and monitor the learning curve. The "freeze-thaw" strategy (Swersky et al., 2014) halts training of models that do not look so promising on the basis of the learning curve, but may restart training them at a later point. This was used *e.g.* by (Lloyd, 2016) in the 2015/2016 challenge.

Model space: Homogeneous *vs.* heterogeneous. An unsettled question is whether one should search a large or small model space. The challenge did not allow us to give a definite answer to this question. Most participants opted for searching a relatively large model space, including a wide variety of models found in the scikit-learn library. Yet, one of the strongest entrants (the Intel team) submitted results simply obtained with a boosted decision tree (*i.e.* consisting of a homegeneous set of weak learners/basic models). Clearly, it suffices to use just ONE learning machine that is a universal approximator to be able to learn anything, given enough training data. So why include several? It is a question of rate of convergence: how fast we climb the learning curve. Including stronger basic models is one way to climb the learning curve faster. Our post-challenge experiments (Figure 9)

reveal that the scikit-learn version of Random Forest (an ensemble of homogeneous basic models – decision trees) does not usually perform as well as the winners' version, hinting that there is a lot of know-how in the Intel solution, which is also based on ensembles of decision tree, that is not captured by a basic ensemble of decision trees such as RF. We hope that more principled research will be conducted on this topic in the future.

Search strategies: Filter, wrapper, and embedded methods With the availability of powerful machine learning toolkits like scikit-learn (on which the starting kit was based), the temptation is great to implement **all-wrapper methods** to solve the CASH (or "full model selection") problem. Indeed, most participants went that route. Although a number of ways of optimizing hyper-parameters with **embedded methods** for several basic classifiers have been published (Guyon et al., 2006a), they each require changing the implementation of the basic methods, which is time consuming and error prone compared to using already debugged and well optimized library version of the methods. Hence practitioners are reluctant to invest development time in the implementation of embedded methods. A notable exception is the software of *marc.boulle*, which offers a self-contained hyper-parameter free solution based on Naive Bayes, which includes re-coding of variables (grouping or discretization) and variable selection. See Appendix D.

Multi-level optimization: Another interesting issue is whether multiple levels of hyper-parameters should be considered for reasons of computational effectiveness or overfitting avoidance. In the Bayesian setting, for instance, it is quite feasible to consider a hierarchy of parameters/hyper-parameters and several levels of priors/hyper-priors. However, it seems to be that for practical computational reasons, in the AutoML challenges, the participants use a shallow organization of hyper-parameter space and avoid nested cross-validation loops.

Time management: Exploration *vs.* exploitation tradeoff: With a tight time budget, efficient search strategies must be put into place to monitor the tradeoff exploration/exploitation. To compare strategies, we show in Appendix G learning curves for two top ranking participants who adopted very different methods: *Abhishek* and *aad_freiburg*. The former uses heuristic methods based on prior human experience while the latter initializes search with models predicted to be best suited by a meta-learning machine, then performs Bayesian optimization of hyper-parameters. *Abhishek* seems to often start with a better solution but performs a less efficient exploration. In contrast, *aad_freiburg* starts lower but often ends up with a better solution. Some elements of randomness in the search are useful to arrive at better solutions.

Preprocessing and feature selection: The datasets presented themselves with intrinsic difficulties that could be in part addressed by preprocessing or special modifications of algorithms: Sparsity, missing values, categorical variables, and irrelevant variables. Yet it appears that among the top ranking participants preprocessing has not been a focus of attention. They relied on the simple heuristics provided in the starting kit: replacing missing values by the median and adding a missingness indicator variable, one-hot-encoding of categorical variables. Simple normalizations were used. The irrelevant variables were ignored by 2/3 of the participants and no use of feature selection was made by top ranking participants. The methods used (involving ensembling) seem to be intrinsically robust against irrelevant variables. More details from the fact sheets are found in Appendix D.

Unsupervised learning: Despite the recent regain of interest in unsupervised learning spurred by the Deep Learning community, in the AutoML challenge series, unsupervised learning is not widely spread, except for the use of classical space dimensionality reduction techniques such as ICA and PCA. See Appendix D for more details.

Transfer learning and meta learning: To our knowledge, only the *aad_freiburg* relied on meta-learning to initialize their hyper-parameter search. To that end, they used datasets of OpenML[14]. The number of datasets released and the diversity of tasks did not allow the participants to perform effective transfer learning or meta learning.

Deep learning: The type of computations resources available in AutoML phases ruled out the use of Deep Learning, except in the GPU track. However, even in that track, the Deep Learning methods did not come ahead. One exception is the *aad_freiburg*, they used Deep Learning in Tweakathon rounds three and four and found it to be helpful for the datasets Alexis, Tania and Yolanda.

Task and metric optimization: There were four types of tasks (regression, binary classification, multi-class classification, and multi-label classification) and six scoring metrics (R2, ABS, BAC, AUC, F1, and PAC). Moreover, class balance and number of classes varied a lot for classification problems. Moderate effort has been put into designing methods optimizing specific metrics. Rather, generic methods were used and the outputs post-fitted to the target metrics by cross-validation or through the ensembling method.

Engineering: One of the big lessons of the AutoML challenge series is that most methods fail to return results in all cases, not a "good" result, but "any" reasonnable result. Reasons for failure include "out of time" and "out of memory" or various other failures (*e.g.* numerical instabilities). We are still very far from having "basic models" that run on all datasets. One of the strength of auto-sklearn is to ignore those models that fail and generally find at least one that returns a result.

Parallelism: The computers made available had several cores, so in principle, the participants could make use of parallelism. One common strategy was just to rely on numerical libraries that internally use such parallelism automatically. The *aad_freiburg* team used the differnt cores to lauch in parallel model search for different datasets (since each round included 5 datasets). These different uses of computer resources are visible in the learnign curves (see Appendix G).

6. Discussion

We briefly summarize the main questions we asked ourselves and the main findings:

1. **Was the provided time budget sufficient to complete the tasks of the challenge?** We drew learning curves as a function of time for the winning solution of *aad_freiburg* (auto-sklearn), see Appendix G). This revealed that for most dataset performances still improved well beyond the time limit imposed by the organizers. Although for about half the datasets the improvement is modest (no more that 20% of the score obtained at the end of the imposed time limit), for some datasets the improvement was very large (more than 2x the original score). The improvements are usually gradual, but sudden performance improvements occur. For instance, for

14. https://www.openml.org/

Wallis, the score doubled suddenly at 3x the time limit imposed in the challenge. As also noted by the authors of the package (Feurer et al., 2015b) auto-sklearn has a slow start but on the long run gets performances close to the best method.

2. **Are there tasks that were significantly more difficult than others for the participants?** Yes, there was a very wide range of difficulties for the tasks as revealed by the dispersion of the participants in terms of average (median) and variability (third quartile) of their scores. Madeline, a synthetic dataset featuring a very non-linear task, was very difficult for many participants. Other difficulties that caused failures to return a solution included large memory requirements (particularly for methods that attempted to convert sparse matrices to full matrices), and short time budget for datasets with large number of training examples and/or features or with many classes or labels.

3. **Are there meta-features of datasets and methods providing useful insight to recommend certain methods for certain types of datasets?** The *aad_freiburg* team used a subset of 53 meta-features (a superset of the simple statistics provided with the challenge datasets) to measure similarity between datasets. This allowed them to conduct hyper-parameter search more effectively by initializing the search with settings identical to those selected for similar datasets previously processed (a form of meta-learning). Our own analysis revealed that it is very difficult to predict the predictors' performances from the meta-features, but it is possible to predict relatively accurately which "basic method" will perform best. With LDA, we could visualize how datasets recoup in two dimensions and show a clean separation between datasets "preferring" Naive Bayes, linear SGD, or KNN, or RF. This deserves further investigation.

4. **Does hyper-parameter optimization really improve performance over using default values?** The comparison we conducted reveals that optimizing hyper-parameters rather than choosing default values for a set of four basic predictive models (K-nearest neighbors, Random Forests, linear SGD, and Naive Bayes) is generally beneficial. In the majority of cases (but not always), hyper-parameter optimization (hyper-opt) results in better performances than default values. Hyper-opt sometimes fails because of time or memory limitations, which gives room for improvement.

5. **How do winner's solutions compare with basic scikit-learn models?** They compare favorably. For example, the results of basic models whose parameters have been optimized do not yield generally as good results as running auto-sklearn. However, a basic model with default HP sometimes outperforms this same model tuned by auto-sklearn.

7. Conclusion

We have analyzed the results of several rounds of AutoML challenges.

Our design of the first AutoML challenge (2015/2016) was satisfactory in many respects. In particular, we attracted a large number of participants (over 600), attained results that are statistically significant, and advanced the state-of-the-art to automate machine learning.

Publicly available libraries have emerged as a result of this endeavor, including "auto-sklearn".

In particular, we designed a benchmark with a large number of diverse datasets, with large enough test sets to separate top ranking participants. It is difficult to anticipate the size of the test sets needed, because the error bars depend on the performances attained by the participants, so we are pleased that we made reasonable guesses. Our simple rule-of-thumb "N=50/E" where N is the number of test samples and E the error rate of the most depleted class seems to be widely applicable. Also concerning the datasets, we made sure that they were neither too easy nor too hard. This is important to be able to separate participants. To quantify this, we introduced the notion of "intrinsic difficulty" and "modeling difficulty". Intrinsic difficulty can be quantified by the performance of the best method (as a surrogate for the best attainable performance, *i.e.* the Bayes rate for classification problems). Modeling difficulty can be quantified by the spread in performance between methods. Our best problems have relatively low "intrinsic difficulty" and high "modeling difficulty". However, the diversity of the 30 datasets of our first 2015/2016 challenge is both a feature and a curse: it allows us to test the robustness of software across a variety of situations, but is makes meta-learning very difficult, if not impossible (datasets being very little redundant with respect to meta-features). Consequently, external meta-learning data must be used if meta-learning is to be explored. This was the strategy adopted by the AAD Freiburg team, which used the OpenML data for meta training. Likewise, we attached different metrics to each dataset. This contributed to making the tasks more realistic and more difficult, but also made meta-learning harder. In the second 2018 challenge, we diminished the variety of datasets and used a single metric.

With respect to task design, we learned that the devil is in the details. The challenge participants solve exactly the task proposed 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 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, the two versions of AutoML challenge we have run differ in the difficulty of transfer learning. In the 2015/2016 challenge, round 0 introduced 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. But in fact the datasets of round 0 were relatively easy. Then at 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 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. 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, it may be preferable to organize yearly events punctuated by workshops. This is a natural way of balancing competition and cooperation since workshops are a place of exchange. Participants are naturally 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 solution 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 easily reproduce the results without stumbling 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 computers may yield significant differences in results. Hopefully, with the generalization of the 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. For instance, we are preparing the NIPS 2018 Life Long Machine Learning challenge in which participants will be exposed to data whose distribution slowly drifts over time. We are also looking at a challenge of automatic machine learning where we will focus on transfer from similar domains.

Acknowledgments

Microsoft supported the organization of this challenge and donated the prizes and cloud computing time on Azure. This project received additional support from the Laboratoire d'Informatique Fondamentale

(LIF, UMR CNRS 7279) of the University of Aix Marseille, France, via the LabeX Archimede program, the Laboratoire de Recheche en Informatique of Paris Sud University, and INRIA-Saclay as part of the TIMCO project, as well as the support from the Paris-Saclay Center for Data Science (CDS). Additional computer resources were provided generously by J. Buhmann, ETH Zürich. This work has been partially supported by the Spanish project TIN2016-74946-P (MINECO/FEDER, UE) and CERCA Programme / Generalitat de Catalunya. The datasets released were selected among 72 datasets that were donated (or formatted using data publicly available) by the co-authors and by: Y. Aphinyanaphongs, O. Chapelle, Z. Iftikhar Malhi, V. Lemaire, C.-J. Lin, M. Madani, G. Stolovitzky, H.-J. Thiesen, and I. Tsamardinos. Many people provided feedback to early designs of the protocol and/or tested the challenge platform, including: 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 who contributed to the implementation of the Codalab platform and the sample code include E. Camichael, I. Chaabane, I. Judson, C. Poulain, P. Liang, A. Pesah, L. Romaszko, X. Baro Solé, E. Watson, F. Zhingri, M. Zyskowski. Some initial analyses of the challenge results were performed by I. Chaabane, J. Lloyd, N. Macia, and A. Thakur were incorporated in this paper. Katharina Eggensperger, Syed Mohsin Ali and Matthias Feurer helped with the organization of the Beat AutoSKLearn challenge. Matthias Feurer also contributed to the simulations of running auto-sklearn on 2015-2016 challenge datasets.

References

Amir Reza Saffari Azar Alamdari and Isabelle Guyon. Quick start guide for CLOP. Technical report, Graz University of Technology and Clopinet, May 2006.

Christophe Andrieu, Nando De Freitas, and Arnaud Doucet. Sequential MCMC for Bayesian model selection. In *IEEE Signal Processing Workshop on Higher-Order Statistics*, pages 130–134, 1999.

Filipe Assunção, Nuno Lourenço, Penousal Machado, and Bernardete Ribeiro. Denser: Deep evolutionary network structured representation. *arXiv preprint arXiv:1801.01563*, 2018.

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

Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michèle Sebag. Collaborative hyperparameter tuning. In *30th International Conference on Machine Learning*, volume 28, pages 199–207. JMLR Workshop and Conference Proceedings, May 2013.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

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.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

James Bergstra, Daniel Yamins, and David D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *30th International Conference on Machine Learning*, volume 28, pages 115–123, 2013.

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):273–324, December 1997.

Marc Boullé. Compression-based averaging of selective naive bayes classifiers. *Journal of Machine Learning Research*, 8:1659–1685, 2007. URL http://dl.acm.org/citation.cfm?id=1314554.

Marc Boullé. A parameter-free classification method for large scale learning. *Journal of Machine Learning Research*, 10:1367–1385, 2009. doi: 10.1145/1577069.1755829. URL http://doi.acm.org/10.1145/1577069.1755829.

Pavel Brazdil, Christophe Giraud Carrier, Carlos Soares, and Ricardo Vilalta. *Metalearning: Applications to data mining*. Springer Science & Business Media, 2008.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

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.

Gavin C. Cawley and Nicola L. C. Talbot. Preventing over-fitting during model selection via Bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8:841–861, April 2007.

F. Chollet. Keras. https://github.com/fchollet/keras, 2015.

Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel programming. *Annals of Operations Research*, 153:235–256, 2007.

Stephan Dempe. *Foundations of bilevel programming*. Kluwer Academic Publishers, 2002.

S. Dieleman, J. Schlter, C. Raffel, E. Olson, S. K. Snderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. De Fauw, M. Heilman, diogo149, B. McFee, H. Weideman, takacsg84, peterderivaz, Jon, instagibbs, Dr. Kashif Rasul, CongLiu, Britefury, and J. Degrave. Lasagne: First release. http://dx.doi.org/10.5281/zenodo.27878, August 2015.

Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

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

Bradley Efron. Estimating the error rate of a prediction rule: Improvement on cross-validation. *Journal of the American Statistical Association*, 78(382):316–331, 1983.

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.

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

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/falkner18a.html.

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*, 2015a.

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. 2015b. URL https://github.com/automl/auto-sklearn.

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, 2015c.

Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Practical automated machine learning for the automl challenge 2018. In *International Workshop on Automatic Machine Learning at ICML*, 2018. URL https://sites.google.com/site/automl2018icml/.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

Zoubin Ghahramani. Unsupervised learning. In *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Computer Science*, pages 72–112. Springer Berlin Heidelberg, 2004.

I. Guyon. *Challenges in Machine Learning book series*. Microtome, 2011-2016. URL http://www.mtome.com/Publications/CiML/ciml.html.

I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, T. Kam Ho, N. Macià, B. Ray, M. Saeed, A. Statnikov, and E. Viegas. AutoML challenge 2015: Design and first results. In *Proc. of AutoML 2015@ICML*, 2015a. URL https://drive.google.com/file/d/0BzRGLkqgrI-qWkpzcGw4bFpBMUk/view.

I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, T. Kam Ho, N. Macià, B. Ray, M. Saeed, A. Statnikov, and E. Viegas. Design of the 2015 ChaLearn AutoML challenge. In *International Joint Conference on Neural Networks*, 2015b. URL http://www.causality.inf.ethz.ch/AutoML/automl_ijcnn15.pdf.

I. Guyon, I. Chaabane, H. J. Escalante, S. Escalera, D. Jajetic, J. R. Lloyd, N. Macía, B. Ray, L. Romaszko, M. Sebag, A. Statnikov, S. Treguer, and E. Viegas. A brief review of the ChaLearn AutoML challenge. In *Proc. of AutoML 2016@ICML*, 2016. URL https://docs.google.com/a/chalearn.org/viewer?a=v&pid=sites&srcid=Y2hhbGVhcm4ub3JnfGF1dG9tbHxneDoyYThjZjhhNzRjMzI3MTg4.

Isabelle Guyon, Amir Reza Saffari Azar Alamdari, Gideon Dror, and Joachim Buhmann. Performance prediction challenge. In *the International Joint Conference on Neural Networks*, pages 1649–1656, 2006a.

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

Isabelle Guyon, Gavin Cawley, and Gideon Dror. *Hands-On Pattern Recognition: Challenges in Machine Learning, Volume 1*. Microtome Publishing, USA, 2011. ISBN 0971977712, 9780971977716.

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

Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

Carine Hue and Marc Boullé. A new probabilistic approach in rank regression with optimal bayesian partitioning. *Journal of Machine Learning Research*, 8:2727–2754, 2007. URL http://dl.acm.org/citation.cfm?id=1390332.

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

F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the conference on Learning and Intelligent OptimizatioN (LION 5)*, 2011.

John P. A. Ioannidis. Why most published research findings are false. *PLoS Medicine*, 2 (8):e124, August 2005.

D. Jajetic. Djajetic Implementation. https://github.com/djajetic/AutoML5, 2016a.

D. Jajetic. GPU_djajetic Implementation. https://github.com/djajetic/GPU_djajetic, 2016b.

Michael I. Jordan. On statistics, computation and scalability. *Bernoulli*, 19(4):1378–1390, September 2013.

S. Sathiya Keerthi, Vikas Sindhwani, and Olivier Chapelle. An efficient method for gradient-based adaptation of hyperparameters in SVM models. In *Advances in Neural Information Processing Systems*, 2007.

A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast bayesian hyperparameter optimization on large datasets. In *Electronic Journal of Statistics*, volume 11, 2017.

Ron Kohavi and George H. John. Wrappers for feature selection. *Artificial Intelligence*, 97 (1-2):273–324, December 1997.

John Langford. Clever methods of overfitting, 2005. Blog post at http://hunch.net/?p=22.

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

H. Mendoza, A. Klein, M. Feurer, J. Tobias Springenberg, and Frank Hutter. Towards automatically-tuned neural networks. In *ICML 2016 workshop on AutoML*, June 2016. URL https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxhdXRvbWwyMDE2fGd4OjMzYjQ4OWNhNTFhNzlhNGE.

Michinari Momma and Kristin P. Bennett. A pattern search method for model selection of support vector regression. In *In Proceedings of the SIAM International Conference on Data Mining*. SIAM, 2002.

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

Gregory M. Moore, Charles Bergeron, and Kristin P. Bennett. Nonsmooth bilevel programming for hyperparameter selection. In *IEEE International Conference on Data Mining Workshops*, pages 374–381, 2009.

Alexandru Niculescu-Mizil, Claudia Perlich, Grzegorz Swirszcz, Vikas Sindhwani, Yan Liu, Prem Melville, Dong Wang, Jing Xiao, Jianying Hu, Moninder Singh, et al. Winning the kdd cup orange challenge with ensemble selection. In *Proceedings of the 2009 International Conference on KDD-Cup 2009-Volume 7*, pages 23–34. JMLR. org, 2009.

Manfred Opper and Ole Winther. *Gaussian processes and SVM: Mean field results and leave-one-out*, pages 43–65. MIT, 10 2000. ISBN 0262194481. Massachusetts Institute of Technology Press (MIT Press) Available on Google Books.

Mee Young Park and Trevor Hastie. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69 (4):659–677, 2007.

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.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Quoc Le, and Alex Kurakin. Large-scale evolution of image classifiers. *arXiv preprint arXiv:1703.01041*, 2017.

Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

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

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2951–2959. 2012.

Alexander Statnikov, Lily Wang, and Constantin F Aliferis. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC Bioinformatics*, 9(1), 2008.

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.

L. Sun-Hosoya. Automl challenge: System description of lisheng sun. In *ICML 2016 workshop on AutoML*, June 2016. URL http://dx.doi.org/10.5281/zenodo.27878.

K. Swersky, J. Snoek, and R. P. Adams. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems 26*, pages 2004–2012, 2013.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

A. Thakur. AutoML challenge: Rules for selecting neural network architectures for automl-gpu challenge. In *ICML 2016 workshop on AutoML*, June 2016. URL https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxhdXRvbWwyMDE2fGd4OjNmY2M0N2JhZGViZWY3ZDY.

A. Thakur and A. Krohn-Grimberghe. AutoCompete: A framework for machine learning competitions. In *Proceedings of the International Conference on Machine Learning 2015, Workshop on Automatic Machine Learning*, 2015. URL https://docs.google.com/a/chalearn.org/viewer?a=v&pid=sites&srcid=Y2hhbGVhcm4ub3JnfGF1dG9tbHxneDo3YThhNmNiNDA0M2Q2NjM5.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, 2016.

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

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 847–855. ACM, 2013.

E. Tuv, A. Borisov, G. Runger, and K. Torkkola. Feature selection with ensembles, artificial variables, and redundancy elimination. *Journal of Machine Learning Research*, 10:1341–1366, January 2009.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Vladimir Vapnik and Olivier Chapelle. Bounds on error expectation for support vector machines. *Neural computation*, 12(9):2013–2036, 2000.

Jason Weston, Andre Elisseeff, Goekhan BakIr, and Fabian Sinz. Spider, 2007. http://mloss.org/software/view/29/.

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

Appendix A. Meta Features

We define the information and statistics provided in the public or private "info" files, reported in Table 2.

PUBLIC INFO:

- *task* = 'binary.classification', 'multiclass.classification', 'multilabel.classification', 'regression'
- *target_type* = 'Binary', 'Categorical','Numerical'
- *feat_type* = 'Binary', 'Categorical','Numerical'
- *metric* = 'bac', 'auc', 'f1', 'pac', 'abs', 'r2'
- *feat_num* = number of features
- *target_num* = number of columns of target file (one, except for multi-label problems)
- *label_num* = number of labels (number of unique values of the targets)
- *train_num* = number of trainign examples
- *valid_num* = number of validation examples (development test set)
- *test_num* = number of test examples (final test set)
- *has_categorical* = whether there are categorical variable (yes=1, no=0)

- $has\_missing$ = whether there are missing values (yes=1, no=0)
- is_sparse = whether the data are in sparse format (yes=1, no=0)

PRIVATE INFO:

- $real\_feat\_num$ = number of real features
- $probe\_num$ = number of fake features (probes)
- $frac\_probes$ = fraction of probes i.e. $probe_num/(probe\_num + real\_feat\_num)$
- $feat\_type\_freq$ = fraction of feature of each type 'Numerical', 'Categorical', or 'Binary'
- $train\_label\_freq$ = frequency of each label in trainign data
- $train\_label\_entropy$ = entropy of labels in training data
- $train\_sparsity$ = sparsity of training data (fraction of occurence of zero values)
- $train\_frac\_missing$ = fraction of missing values in training data
- The last 4 statistics are also calculated for the validation set and the test set
- $train\_data\_aspect\_ratio$ = ratio of number of training examples over number of features

We define the meta features as implemented in (Feurer et al., 2015a,b,c)[15]:

- ClassProbabilityMin = $min_{i=1...n}(p(Class_i)) = min_{i=1...n}(\frac{NumberOfInstances\_Class_i}{TotleNumberOfInstances})$
- ClassProbabilityMax = $max_{i=1...n}(p(Class_i)) = max_{i=1...n}(\frac{NumberOfInstances\_Class_i}{TotleNumberOfInstances})$
- ClassEntropy = $mean(-\sum_{i=1}^{n} p(Class_i)ln(p(Class_i)))$ where $p(Class_i)$ is the probability of having an instance of Class_i
- ClassOccurences = number of examples for each class
- ClassProbabilityMean = mean($\frac{ClassOcurrences}{NumberOfClasses}$)
- ClassProbabilitySTD = std($\frac{ClassOcurrences}{NumberOfClasses}$)
- DatasetRatio = $\frac{NumberOfFeatures}{NumberOfInstances}$
- InverseDatasetRatio = $\frac{NumberOfInstances}{NumberOfFeatures}$
- LogInverseDatasetRatio = log($DatasetRatio$)

15. Kurtosis, Skewness, KurtosisPCA and SkewnessPCA are intermediate metafeatures used to calculate some other metafeatures

- Landmark[Some_Model]: accuracy of [Some_Model] applied on dataset.
- LandmarkDecisionNodeLearner & LandmarkRandomNodeLearner: Both are decision tree with max_depth=1. 'DecisionNode' considers all features when looking for best split, and 'RandomNode' considers only 1 feature, where comes the term 'random'.
- Skewnesses: Skewness of each numerical features. Skewness measures the symmetry of a distribution. A skewness value > 0 means that there is more weight in the left tail of the distribution. Computed by scipy.stats.skew.
- SkewnessMax / SkewnessMin / SkewnessMean / SkewnessSTD: max / min / mean / std over skewness of all features.
- NumSymbols: Sizes of categorical features: for each categorical feature, compute its size (number of values in the category).
- SymbolsMax / SymbolsMin / SymbolsMean / SymbolsSTD / SymbolsSum = max / min / mean / std / sum over NumSymbols
- NumberOfCategoricalFeatures: Number of categorical features.
- NumberOfNumericFeatures: Number of numerical features
- RatioNumericalToNominal = $\frac{NumberOfNumericFeatures}{NumberOfCategoricalFeatures}$
- RatioNominalToNumerical = $\frac{NumberOfCategoricalFeatures}{NumberOfNumericFeatures}$
- Kurtosis = Fourth central moment divided by the square of the variance = $\frac{E[(x_i - E[x_i])^4]}{[E[(x_i - E[x_i])^4]]^2}$ where x_i is the i-th feature. Computed using scipy.stats.kurtosis.
- KurtosisMax / KurtosisMin / KurtosisMean / KurtosisSTD = max / min / mean / std of kurtosis over all features
- PCAKurtosis: Transform data by PCA, then compute the kurtosis
- NumberOfInstances = Number of examples
- NumberOfFeatures = Number of features
- NumberOfClasses = Number of classes
- LogNumberOfFeatures = $\log(NumberOfFeatures)$
- LogNumberOfInstances = $\log(NumberOfInstances)$
- MissingValues: Boolean matrix of dim (NumberOfInstances , NumberOfFeatures), indicating if an element of is a missing value.
- NumberOfMissingValues: Total number of missing value
- NumberOfInstancesWithMissingValues: Number of examples containing missing values.

- NumberOfFeaturesWithMissingValues: Number of features containing missing values.
- PCA: PCA decomposition of data.
- PCAFractionOfComponentsFor95PercentVariance: Fraction of PCA components explaining 95% of variance of the data.
- PCAKurtosisFirstPC: Kurtosis of the first PCA component.
- PCASkewnessFirstPC: Skewness of the first PCA component.

Appendix B. Datasets of the 2015/2016 AutoML challenge

ROUND 0

SET 0.1: ADULT

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multilabel	F1	3	1	0.16	0.011	1	0.5	9768	4884	34190	24	1424.58

This dataset was prepared by Isabelle Guyon from original data extracted by Barry Becker from the 1994 Census database. The data was donated to the UCI repository by Ron Kohavi: "Adult data set" (https://archive.ics.uci.edu/ml/datasets/Adult).

Past Usage: The Adult data set is among the most used marketing-style datasets. The ADA dataset is a version of it that was used previously used in the Performance Prediction challenge, the Model Selection game, and the Agnostic Learning vs. Prior Knowledge (ALvsPK) challenge.

Description: The original prediction task was to determine whether a person makes over 50K a year from census data. The problem was transformed into a multilabel problem by adding sex and race in the target values (for race, separate white form others).

Preparation: A set of reasonably clean records was extracted using the following conditions: $((AAGE > 16)$ and $(AGI > 100)$ and $(AFNLWGT > 1)$ and $(HRSWK > 0))$.

Representation: The features include age, workclass, education, etc.

SET 0.2: CADATA

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
regression	R2	0	NaN	0	0	0	0.5	10640	5000	5000	16	312.5

This dataset was prepared by Isabelle Guyon from original data provided by Kelley Pace and Ronald Barry: "California houses" (http://lib.stat.cmu.edu/datasets/).

Past Usage: Part of the StatLib datasets. Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions, Statistics and Probability Letters, 33 (1997) 291-297. It was submitted by Kelley Pace (kpace@unix1.sncc.lsu.edu). [9/Nov/99].

Description: These spatial data contain 20,640 observations on housing prices with 9 economic covariates.

Preparation: The original authors collected information on the variables using all the block groups in California from the 1990 Census. In this sample a block group on average includes 1425.5 individuals living in a geographically compact area. Naturally, the

geographical area included varies inversely with the population density. They computed distances among the centroids of each block group as measured in latitude and longitude. The final data contained 20,640 observations on 9 variables. The dependent variable is ln(median house value). For the purpose of the AutoML challenge, all samples were merged and the data were freshly randomly split in three sets: training, validation, and test. The order of the features was randomized, after adding a few distractor features (probes) that are permuted versions of real features.

Representation: Features.

SET 0.3: DIGITS

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multiclass	BAC	10	1	0.42	0	0	0.5	35000	20000	15000	1568	9.57

This dataset was prepared by Isabelle Guyon from original data provided by Yann LeCun, Corinna Cortes, and Chris Burges: "MNIST handwritten digit dataset" (http://yann.lecun.com/exdb/mnist/).

Past Usage: Many methods have been tried on the MNIST database, in its original data split (60,000 training examples, 10,000 test examples, 10 classes). This dataset was used in the NIPS 2003 Feature Selection Challenge under the name GISETTE and in the WCCI 2006 Performance Prediction Challenge and the IJCNN 2007 Agnostic Learning vs. Prior Knowledge Challenge under the name GINA, and in the ICML 2011 Unsupervised and Transfer Learning Challenge under the name ULE.

Description: This is a dataset of handwritten digits. It is a subset of a larger set made available from NIST. The digits in pixel representation have been size-normalized and centered in a fixed-size image by the authors. The data are quantized on 256 gray level values.

Preparation: For the purpose of the AutoML challenge, all samples were merged and the data were freshly randomly split in three sets: training, validation, and test. The order of the features (pixels) was also randomize, after adding a few distractor features (probes) that are permuted versions of real features.

Representation: Pixels.

SET 0.4: DOROTHEA

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	AUC	2	0.46	0.99	0	0	0.5	800	350	800	100000	0.01

This dataset was prepared by Isabelle Guyon from original data provided by DuPont Pharmaceuticals: "Feature selection challenge data" (http://www.cs.wisc.edu/~dpage/kddcup2001/).

Past Usage: DOROTHEA was prepared for the NIPS 2003 variable and feature selection benchmark by Isabelle Guyon, 955 Creston Road, Berkeley, CA 94708, USA (isabelle@clopinet.com). The dataset with which DOROTHEA was created is one of the KDD (Knowledge Discovery in Data Mining) Cup 2001. The original dataset and papers of the winners of the competition are available at: http://www.cs.wisc.edu/~dpage/kddcup2001/. DuPont Pharmaceuticals graciously provided this data set for the KDD Cup

2001 competition. All publications referring to analysis of this data set should acknowledge DuPont Pharmaceuticals Research Laboratories and KDD Cup 2001.

Description: Synopsis of the original data: One binary attribute (active A or inactive I) must be predicted. Drugs are typically small organic molecules that achieve their desired activity by binding to a target site on a receptor. The first step in the discovery of a new drug is usually to identify and isolate the receptor to which it should bind, followed by testing many small molecules for their ability to bind to the target site. This leaves researchers with the task of determining what separates the active (binding) compounds from the inactive (non-binding) ones. Such a determination can then be used in the design of new compounds that not only bind, but also have all the other properties required for a drug (solubility, oral absorption, lack of side effects, appropriate duration of action, toxicity, etc.). The original training data set consisted of 1909 compounds tested for their ability to bind to a target site on thrombin, a key receptor in blood clotting. The chemical structures of these compounds are not necessary for our analysis and were not included. Of the training compounds, 42 are active (bind well) and the others are inactive. To simulate the real-world drug design environment, the test set contained 634 additional compounds that were in fact generated based on the assay results recorded for the training set. Of the test compounds, 150 bind well and the others are inactive. The compounds in the test set were made after chemists saw the activity results for the training set, so the test set had a higher fraction of actives than did the training set in the original data split. Each compound is described by a single feature vector comprised of a class value (A for active, I for inactive) and 139,351 binary features, which describe three-dimensional properties of the molecule. The definitions of the individual bits are not included we only know that they were generated in an internally consistent manner for all 1909 compounds. Biological activity in general, and receptor binding affinity in particular, correlate with various structural and physical properties of small organic molecules. The task is to determine which of these properties are critical in this case and to learn to accurately predict the class value. In evaluating the accuracy, a differential cost model was used, so that the sum of the costs of the actives will be equal to the sum of the costs of the inactives.

Preparation: To prepare the data, we used information from the analysis of the KDD cup 2001 and the literature. There were 114 participants to the competition that turned in results. The winner of the competition is Jie Cheng (Canadian Imperial Bank of Commerce). His presentation is available at: http://www.cs.wisc.edu/~dpage/kddcup2001/Hayashi.pdf. The data was also studied by Weston and collaborators: J. Weston, F. Perez-Cruz, O. Bousquet, O. Chapelle, A. Elisseeff and B. Schoelkopf. "Feature Selection and Transduction for Prediction of Molecular Bioactivity for Drug Design". Bioinformatics. At lot of information is available from Jason Weston s web page, including valuable statistics about the data: http://www.kyb.tuebingen.mpg.de/bs/people/weston/kdd/kdd.html. To outperform these results, the paper of Weston et al., 2002, utilizes the combination of an efficient feature selection method and a classification strategy that capitalizes on the differences in the distribution of the training and the test set. First they select a small number of relevant features (less than 40) using an unbalanced correlation score that selects features that have non-zero entries only for positive examples. This score encodes the prior information that the data is unbalanced and that only positive correlations are likely to be useful. The score has an information theoretic motivation, see the paper for details.

Representation: The original data set was modified for the purpose of the feature selection challenge: The original training and test sets were merged. The features were sorted according to an unbalanced correlation critetion, computed using the original test set (which is richer is positive examples). Only the top ranking 100000 original features were kept. The all zero patterns were removed, except one that was given label ?1. For the second half lowest ranked features, the order of the patterns was individually randomly permuted (in order to create "random probes" or distrator features). The order of the patterns and the order of the features were globally randomly permuted to mix the original training and the test patterns and remove the feature order. The data was split into training, validation, and test set while respecting the same proportion of examples of the positive and negative class in each set.

SET 0.5: NEWSGROUPS

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multiclass	PAC	20	1	1	0	0	0	3755	1877	13142	61188	0.21

This dataset was prepared by Hugo Jair Escalante from original data provided by Ken Lang. The version we used was obtained from Deng Cai.: "TNW - 20 Newsgroups data set" (http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html).

Past Usage: The 20 NewsGroups data set is among the most used data sets for text categorization. It has been used to evaluate standard text categorization and recently it has been also widely used for the evaluation of cross domain text categorization.

Description: In this version of the data set the training and test documents were mixed in a single matrix, then plit into training, validation, and test set for the needs of the challenge.

Preparation: The data is organized into 20 different newsgroups (each newsgroup corresponds to a class), each corresponding to a different topic (see http://qwone.com/~jason/20Newsgroups/). Some of the newsgroups are very closely related to each other (e.g. comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), while others are highly unrelated (e.g misc.forsale / soc.religion.christian). .

Representation: Documents are represented by their bag-of-words using a term-frequency weighting scheme.

ROUND 1

SET 1.1: CHRISTINE

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	BAC	2	1	0.071	0	0	0.5	2084	834	5418	1636	3.31

This dataset was prepared by Isabelle Guyon from original data provided by Curt Breneman, Charles Bergeron, and Kristin Bennett: "Activation of pyruvate kynase" (http://www.causality.inf.ethz.ch/activelearning.php?page=datasets).

Past Usage: Active learning challenge, C dataset, see http://www.causality.inf.ethz.ch/activelearning.php.

Description: The task is to predict chemical activity of molecules. This is a two-class classification problems. The variables represent properties of the molecule inferred from its

structure. The problem is therefore to relate structure to activity (a QSAR=quantitative structure-activity relationship problem) to screen new compounds before actually testing them (a HTS=high-throughput screening problem). The problem is to predict the activation of pyruvate kynase, a well characterized enzyme, which regenerates ATP in glycolysis by catalyzing phosphoryl transfer from phosphoenol pyruvate to ADP to yield pyruvate and ATP.

Preparation: We modified the original data split and added probes.

Representation: Features/Attributes representing properties of molecules.

SET 1.2: JASMINE

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	BAC	2	1	0.78	0	0	0.5	1756	526	2984	144	20.72

This dataset was prepared by Isabelle Guyon from original data provided by Reza Farrahi Moghaddam, Mathias Adankon, Kostyantyn Filonenko, Robert Wisnovsky, and Mohamed Cheriet: "Arabic manuscripts" (http://www.causality.inf.ethz.ch/activelearning.php?page=datasets).

Past Usage: Active learning challenge, A dataset, see http://www.causality.inf.ethz.ch/activelearning.php.

Description: The task is to classify cursive script subwords from data in a feature representation extracted from Arabic Historical Manuscripts..

Preparation: We modified the original data split and added probes..

Representation: Features/Attributes.

SET 1.3: MADELINE

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	BAC	2	1	1.2e-06	0	0	0.92	3240	1080	3140	259	12.12

This dataset was prepared by Isabelle Guyon from original data provided by Isabelle Guypn: "Feature selection challenge data" (http://www.nipsfsc.ecs.soton.ac.uk/datasets/;https://archive.ics.uci.edu/ml/datasets/Madelon).

Past Usage: NIPS 2003 feature selection challenge. See Result analysis of the NIPS 2003 feature selection challenge, Isabelle Guyon, Steve R. Gunn, Asa Ben-Hur, Gideon Dror, 2004.

Description: MADELON is an artificial dataset containing data points grouped in 32 clusters placed on the vertices of a five dimensional hypercube and randomly labeled +1 or −1. The five dimensions constitute 5 informative features. 15 linear combinations of those features were added to form a set of 20 (redundant) informative features. Based on those 20 features one must separate the examples into the 2 classes (corresponding to the ±1 labels). We added a number of distractor feature called *probes* having no predictive power. The order of the features and patterns were randomized. See http://www.nipsfsc.ecs.soton.ac.uk/papers/NIPS2003-Datasets.pdf.

Preparation: To draw random data, the program takes the following steps: (1) Each class is composed of a number of Gaussian clusters. N(0,1) is used to draw for each cluster num_useful_feat examples of independent features. (2) Some covariance is added by

multiplying by a random matrix A, with uniformly distributed random numbers between -1 and 1. (3) The clusters are then placed at random on the vertices of a hypercube in a num_useful_feat dimensional space. The hypercube vertices are placed at values ± 1 $class\_sep$. (4) Redundant features are added. They are obtained by multiplying the useful features by a random matrix B, with uniformly distributed random numbers between -1 and 1. (5) Some of the previously drawn features are repeated by drawing randomly from useful and redundant features. Useless features (random probes) are added using N(0,1). (6)- All the features are then shifted and rescaled randomly to span 3 orders of magnitude. (7) Random noise is then added to the features according to N(0,0.1). (8) A fraction $flip\_y$ of labels are randomly exchanged.

Representation: Continuous valued features.

SET 1.4: PHILIPPINE

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	BAC	2	1	0.0012	0	0	0.5	4664	1166	5832	308	18.94

This dataset was prepared by Isabelle Guyon from original data provided by Emmanuel Faure, Thierry Savy, Louise Duloquin, Miguel Luengo Oroz, Benoit Lombardot, Camilo Melani, Paul Bourgine, and Nadine Peyrieras: "Mitosis classification" (http://www.causality.inf.ethz.ch/activelearning.php?page=datasets).

Past Usage: Active learning challenge, E dataset, see http://www.causality.inf.ethz.ch/activelearning.php.

Description: A feature representation of cells of zebrafish embryo to determine whether they are in division (meiosis) or not. All the examples are manually annotated.

Preparation: We modified the original data split and added probes.

Representation: Features extracted from video data.

SET 1.5: SYLVINE

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	BAC	2	1	0.01	0	0	0.5	10244	5124	5124	20	256.2

This dataset was prepared by Isabelle Guyon from original data provided by Department of Forest Sciences, Colorado: "Forest cover types" (https://archive.ics.uci.edu/ml/datasets/Covertype).

Past Usage: Active learning challenge, F dataset, see http://www.causality.inf.ethz.ch/activelearning.php.

Description: The tasks is to classify forest cover types. The original multiclass problem is brought back to Krummholz vs. other classes of trees.

Preparation: We modified the original data split and added probes.

Representation: Features/Attributes

ROUND 2

SET 2.1: ALBERT

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	F1	2	1	0.049	0.14	1	0.5	51048	25526	425240	78	5451.79

This dataset was prepared by Hugo Jair Escalante from original data provided by Olivier Chapelle (CRITEO): "Criteos Delayed Feedback in Display Advertising" (https://www.kaggle.com/c/criteo-display-ad-challenge/details/about-criteo?).

Past Usage: The data set used for the AutoML challenge was taken from the training-set partition of Criteos-Kaggle challenge. The challenge is runing and there are about 350 teams registered. A closely related data set is described in: O. Chapelle. Modeling delayed feedback in display advertising. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM Press, 2014.

Description: The data set is a small subset of the training set provided for the above mentioned challenge. The data set has been balanced (originally the class imbalance ratio was 70/30). .

Preparation: For the purpose of the AutoML challenge, missing values are denoted with NaN, the meaning of the variables has not been described yet (it contains sensitive data). Variables 1-13 are numeric, variables 14-39 are categorical.

Representation: Features related to click prediction, the semantics of the features has not been described elsewhere.

SET 2.2: DILBERT

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multiclass	PAC	5	1	0	0	0	0.16	9720	4860	10000	2000	5

This dataset was prepared by Hugo Jair Escalante from original data provided by Fu Jie Huang, Yann LeCun, Leon Bottou: "NORB data set (2 feature maps)" (http://cs.nyu.edu/~ylclab/data/norb-v1.0/).

Past Usage: This data set has been widely used for the evaluation of 3D object classifcation, it has been very popular recently for deep learning computer vision, the paper introducing the data set is: Yann LeCun, Fu Jie Huang, Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting, CVPR, 2004. (google scholar reports 380 citations).

Description: The data set has 48600 images comprising 5 categories, images come from 50 toys belonging to 5 categories. The objects were imaged by two cameras under 6 lighting conditions, 9 elevations (30 to 70 degrees every 5 degrees), and 18 azimuths (0 to 340 every 20 degrees). Images have been represented with features derived with a convolutional neural network. (TCNN with random weights).

Preparation: For the purpose of the AutoML challenge, all samples were merged (the standard procedure uses half of the images for training and half for testing). Images are represented with the upper layer of a (1-layer) Tiled Convolutional Neural Network (TCNN), no pretraining was performed, random weights were used (see A. Saxe code: http://web.stanford.edu/~asaxe/random_weights.html).

Representation: Features learned by a TCNN, we used 2 maps to erduce the dimensionality of the representation, the inputs to the TCNN are the pixels from the two stereo images, a 4x4 window wans considered.

SET 2.3: FABERT

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multiclass	PAC	7	0.96	0.99	0	0	0.5	2354	1177	8237	800	10.3

This dataset was prepared by Sergio Escalera from original data provided by Sergio Escalera, Xavier Baro, Jordi Gonzalez, Miguel A. Bautista, Meysam Madadi, Miguel Reyes, Victor Ponce: "LAP2014 Gesture Recognition Data set using Skeleton features" (http://sunai.uoc.edu/chalearn/).

Past Usage: The data from which the LAPSD was generated have been used by several people in two challenges (Multimodal Gesture Recognition and Looking at People Challenges), the number of registered participants exceeded 200 hundred (al least 20 people participated throughout the final stages and developed highly competitive methods). More information can be found in: Sergio Escalera et al. Multi-modal Gesture Recognition Challenge 2013: Dataset and Results. Proc. of ICMI 2013, pp. 445-452, 2013, and in Sergio Escalera et al. ChaLearn Looking at People Challenge 2014: Dataset and results, ECCV- Chalearn workshop 2014.

Description: This is a dataset of gesture recognition. It comprises all of the samples (training+validation+test) of the original data set, a total of 13845 samples. Skeleton information was used to represent gestures (BOW formulation). The original data set has 20 gesture classes, for this data set, the 20 gestures are grouped into 10 different classes (0. Perfetto and frieganiente, 1. Prendere, ok and noncenepiu, 2. Bounissimo, furbo, seipazo and cosatifarei, 3. Chevoui, daccordo and combinato, 4. Sonostufo and messidaccordo, 5. Vattene and Vieniqui, 6. Basta, 7. Fame, 8. Tantotempofa, 9. Cheduepalle.

Preparation: For the purpose of the AutoML challenge, all samples were merged. Skeleton frames were first described by the difference of world-coordinates of joint points and the head joint, and then clustered o generate a 400-words vocabulary, which was used to represent the videos.

Representation: Bag-of-Visual-Words using Skeleton coordinates, vocabulary of 400 codewords was considered.

SET 2.4: ROBERT

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multiclass	BAC	10	1	0.01	0	0	0	5000	2000	10000	7200	1.39

This dataset was prepared by Isabelle Guyon from original data provided by Antonio Torralba, Rob Fergus, and William T. Freeman, collected and made available publicly the 80 million tiny image dataset. Vinod Nair and Geoffrey Hinton collected and made available publicly the CIFAR datasets.: "Image classification (from Unsupervised and Transfer Learning Challenge)" (http://www.cs.toronto.edu/?kriz/cifar.html,http://groups.csail.mit.edu/vision/TinyImages/).

Past Usage: The data were used in the Unsupervised and Transfer Learning challenge: http://www.causality.inf.ethz.ch/unsupervised-learning.php.

Description: These are small pictures of objects, animals. etc. We merged the CIFAR-10 and the CIFAR-100 datasets. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. The CIFAR-100 dataset is similar to the CIFAR-10, except that it has 100 classes containing 600 images each. The 100 classes in the CIFAR-100 are grouped into 20 superclasses..

Preparation: The raw data came as 32x32 tiny images coded with 8-bit RGB colors (i.e. 3x32 features with 256 possible values). We converted RGB to HSV and quantized the

results as 8-bit integers. This yielded 30x30x3 = 900 x 3 features. We then preprocessed the gray level image to extract edges. This yielded 30 x 30 features (1 border pixel was removed). We then cut the images into patches of 10x10 pixels and ran kmeans clustering (an on-line version) to create 144 cluster centers. We used these cluster centers as a dictionary to create features corresponding to the presence of one the 144 shapes at one of 25 positions on a grid. This created another 144 x 25 = 3600 features. See http://jmlr.org/proceedings/papers/v27/supplemental/datasetsutl12a.pdf for details..

Representation: Bag of word features.

SET 2.5: VOLKERT

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multiclass	PAC	10	0.89	0.34	0	0	0	7000	3500	58310	180	323.94

This dataset was prepared by Hugo Jair Escalante from original data provided by J. Vogel and B. Schiele.: "VOGEL data set - image classification" (http://ccc.inaoep.mx/~hugojair/ebm/ebm_code_and_data.zip).

Past Usage: This data set has been used in a few publications for the evaluation of region labeling and image retrieval techniques. The data set was introduced in: J. Vogel, B. Schiele. Semantic Modeling of Natural Scenes for Content-Based Image Retrieval. Journal of Computer Vision, Vol. 72(2):133–157, 2007, this paper has ben cited around 250 times according to google scholar.

Description: Images are natural scenes from 6 different categories (coasts / rivers-lakes / forests / mountains / plains / sky-clouds). Each image has been divided in regions of 10x10 pixels each (grid segmentation), 100 regions per image were extracted. The goal of the task is to classify the regions. Regions are represented by a set of visual descriptors, and regions are labeled with one of 17 labels, associated to the scene categories.

Preparation: There are 70000 regions to be labeled with one of 17 labels, where every 100 regions (in the actual order of the X file) were extracted from the same image (each image corresponds to a single natural scene category). In the past 10 fold CV has been used for evaluation.

Representation: Images are represented by their edge and HSI-color histograms, as well as by texture features extracted from the co-occurrence matrix.

ROUND 3

SET 3.1: ALEXIS

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multilabel	AUC	18	0.92	0.98	0	0	0	15569	7784	54491	5000	10.9

This dataset was prepared by Isabelle Guyon from original data provided by The dataset was constructed from the KTH human action recognition dataset of Ivan Laptev and Barbara Caputo and the Hollywood 2 dataset of human actions and scenes of Marcin Marszalek, Ivan Laptev, and Cordelia Schmidt.: "Action recognition (from Unsupervised and Transfer Learning Challenge)" (http://www.nada.kth.se/cvap/actions/,http://www.irisa.fr/vista/Equipe/People/Laptev/download.html).

Past Usage: The data were used in the Unsupervised and Transfer Learning challenge: http://www.causality.inf.ethz.ch/unsupervised-learning.php.

Description: The data include video clips of people performing actions. The identification and recognition of gestures, postures and human behaviors has gained importance in applications such as video surveillance, gaming, marketing, computer interfaces and interpretation of sign languages for the deaf.

Preparation: The data were preprocessed into STIP features using the code of Ivan Laptev: http://www.irisa.fr/vista/Equipe/People/Laptev/download/stip-1.0-winlinux.zip. The final representation is a ?bag of STIP features?. Details are found in the report http://jmlr.org/proceedings/papers/v27/supplemental/datasetsutl12a.pdf.

Representation: Bag of word features.

SET 3.2: DIONIS

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multiclass	BAC	355	1	0.11	0	0	0	12000	6000	416188	60	6936.47

This dataset was prepared by Mehreen Saeed from original data provided by Sarmad Hussain and Qurat ul Ain Akram: "Urdu OCR dataset" (http://www.cle.org.pk/clestore/imagecorpora.htm).

Past Usage: http://www.cle.org.pk/Publication/papers/2013/Binarization%20and%20its%20Evaluation%20for%20Urdu%20Nastalique%20Document%20Images%208-3-1.pdf, http://www.cle.org.pk/Publication/papers/2014/AdaptingTesseract%20for%20Complex%20Scripts-%20an%20Example%20for%20Nastalique%203.10.pdf,www.UrduOCR.netandwww.cle.org.pk/clestore/imagecorpora.htm.

Description: This is a dataset of Urdu printed ligatures shapes with diacritics stripped off. The dataset has been derived from an original dataset found at : http://www.cle.org.pk/clestore/imagecorpora.htm by generating new images from existing ones. A subset of shapes is included in this dataset.

Preparation: For the purpose of the AutoML challenge, new shape images were created using elastic deformations, rotations, shear and scaling. Features were then extracted from the generated images.

Representation: DCT transform of image contours, spatial density computed by dividing each image in a 3x3 grid and computing the density for each cell, eigen values and eigen vector of (x,y) coordinates of foreground shape pixels.

SET 3.3: GRIGORIS

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multilabel	AUC	91	0.87	1	0	0	0	9920	6486	45400	301561	0.15

This dataset was prepared by Grigorios Tsoumakas et al. from original data provided by Grigorios Tsoumakas et al.: "WISE 2014 - Greek Media Monitoring Multilabel Classification" (https://www.kaggle.com/c/wise-2014).

Past Usage: The data set is being used in the WISE 2014 - Greek Media Monitoring Multilabel Classification, the challenge is being managed in the Kaggle platform, at the

moment of writing this file 121 teams have registered for the competition, these are teams that have made at least one submission.

Description: This is a multi-label classification competition for articles coming from Greek printed media. Raw data comes from the scanning of print media, article segmentation, and optical character segmentation, and therefore is quite noisy. Data was collected by scanning a number of Greek print media from May 2013 to September 2013. There are 301561 numerical attributes corresponding to the tokens encountered inside the text of the collected articles. Articles were manually annotated with one or more out of 203 labels (in this version, there are considered 200 labels only).

Preparation: For the purpose of the AutoML challenge, only the training subset of documents has been considered, a total of 200 labels are considered where each has at least one example.

Representation: Text are represented by their bag-of-words with a tfidf weighting scheme.

SET 3.4: JANNIS

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multiclass	PAC	4	0.8	7.3e-05	0	0	0.5	9851	4926	83733	54	1550.61

This dataset was prepared by Hugo Jair Escalante from original data provided by Hugo Jair Escalante, Michael Grubinger: "SAIAPR TC12 benchmark - main-branches classification" (http://imageclef.org/SIAPRdata).

Past Usage: Several methods for image annotation have been evaluated in the SAIAPR-TC12 collection (see http://scholar.google.com/scholar?oi=bibs&hl=en&cites=8812357429744542982); including region-level (this data) and image-level methods. A previous version of this collection (IAPR-TC12) has been widely used to benchmark multimodal image retrieval techniques in the CLEF forum. The data set is described in detail in the following publication: H. J. Escalante, et al. The Segmented and Annotated IAPR-TC12 Benchmark. Computer Vision and Image Understanding Journal, 114(4):419-428, 2010.

Description: In this version of the SAIAPR-TC 12 data set the goal is to classify image regions into one of the 4-most populated branches (Animals, Man-made objects, Persons, Landscape) of a hierarchy of concepts. Each instance is associated to a region of an image. Regions in images have been segmented manually, each region is described by a 27-dimensional verctor comprising the following visual-content attributes: area, boundary/area, width and height of the region, average and standard deviation in x and y, convexity, average, standard deviation and skewness in the RGB and CIE-Lab color spaces. In the past, 10-fold cross validation has been used for evaluation.

Preparation: For the purpose of the AutoML challenge, all regions are labeled by the first-level branch of the original labels.

Representation: Region area, boundary/area, width and height of the region, average and standard deviation in x and y, convexity, average, standard deviation and skewness in the RGB and CIE-Lab color spaces.

SET 3.5: WALLIS

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multiclass	BAC	11	0.91	1	0	0	0	8196	4098	10000	193731	0.05

This dataset was prepared by Hugo Jair Escalante from original data provided by Ana Cardoso Cachopo.: "C12 - the CADE 12 data set" (http://web.ist.utl.pt/~acardoso/datasets/).

Past Usage: This data set has been used to evaluate standard (single label) text categorization. There are no too much references using this data set, most work has been reported from Portuguese and Brazilian colleagues.

Description: The documents in the Cade12 correspond to a subset of web pages extracted from the CADE Web Directory, which points to Brazilian web pages classified by human experts.

Preparation: The data is organized into 12 classes each corresponding to a different webpage category (see http://web.ist.utl.pt/~acardoso/datasets/).

Representation: Documents are represented by their bag-of-words using a term-frequency weighting scheme.

ROUND 4

SET 4.1: EVITA

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	AUC	2	0.21	0.91	0	0	0.46	14000	8000	20000	3000	6.67

This dataset was prepared by Isabelle Guyon from original data provided by National Cancer Institute (NCI)DTP AIDS Antiviral Screen program: "HIV" (http://dtp.nci.nih.gov/docs/aids/aids_data.html).

Past Usage: This data set has been previous use in several challenges including the Performance Prediction Challenge under the name HIVA and the Causation and Prediction Challenge under the name SIDO.

Description: This is a problem of drug activity classification. The data contains descriptors of molecules, which have been tested against the AIDS HIV virus. The target values indicate the molecular activity (+1 active, -1 inactive).

Preparation: The features were reshuffled and a fresh data split was made.

Representation: The molecular descriptors were generated programmatically from the three dimensional description of the molecule, with several programs used by pharmaceutical companies for QSAR studies (Quantitative Structure-Activity Relationship). For example, a descriptor may be the number of carbon molecules, the presence of an aliphatic cycle.

SET 4.2: FLORA

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
regression	ABS	0	NaN	0.99	0	0	0.25	2000	2000	15000	200000	0.08

This dataset was prepared by C J Lin from original data provided by These data were collected primarily by Bryan Routledge, Shimon Kogan, Jacob Sagi, and Noah Smith. This version was obtained from C. J. Lin.: "E2006-tfidf 10-K Corpus" (https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html).

Past Usage: https://archive.ics.uci.edu/ml/datasets/YearPredictionMSD.

Description: Prediction of the release year of a song from audio features. Songs are mostly western, commercial tracks ranging from 1922 to 2011, with a peak in the year 2000s.

Preparation: The data were obtained by C J Lin. They respect the original representation..

Representation: Features: 12 = timbre average, 78 = timbre covariance.

SET 4.3: HELENA

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multiclass	BAC	100	0.9	6e-05	0	0	0	18628	9314	65196	27	2414.67

This dataset was prepared by Hugo Jair Escalante from original data provided by Hugo Jair Escalante, Michael Grubinger: "SAIAPR TC12 benchmark - top-100 frequent labels" (http://imageclef.org/SIAPRdata).

Past Usage: Several methods for image annotation have been evaluated in the SAIAPR-TC12 collection (see http://scholar.google.com/scholar?oi=bibs&hl=en&cites=8812357429744542982); including region-level (this data) and image-level methods. A previous version of this collection (IAPR-TC12) has been widely used to benchmark multimodal image retrieval techniques in the CLEF forum. The data set is described in detail in the following publication: H. J. Escalante, et al. The Segmented and Annotated IAPR-TC12 Benchmark. Computer Vision and Image Understanding Journal, 114(4):419-428, 2010.

Description: In this version of the SAIAPR-TC 12 data set the goal is to classify image regions into one of 100 labels (the top-100 more frequent ones). The original data set has about 276 labels, organized into a hierarchy of concepts, in this version of the data set the goal is to classify the leaf-labels of the hierarchy. Each instance is associated to a region of an image. Regions in images have been segmented manually, each region is described by a 27-dimensional verctor comprising the following visual-content attributes: area, boundary/area, width and height of the region, average and standard deviation in x and y, convexity, average, standard deviation and skewness in the RGB and CIE-Lab color spaces. In the past, 10-fold cross validation has been used for evaluation.

Preparation: For the purpose of the AutoML challenge, all regions are labeled by their leaf-label in the hierarchy of concepts.

Representation: Region area, boundary/area, width and height of the region, average and standard deviation in x and y, convexity, average, standard deviation and skewness in the RGB and CIE-Lab color spaces.

SET 4.4: TANIA

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multilabel	PAC	95	0.79	1	0	0	0	44635	22514	157599	47236	3.34

This dataset was prepared by Isabelle Guyon from original data provided by The original data were donated by Reuters and downloaded from: Lewis, D. D. RCV1-v2/LYRL2004: The LYRL2004 Distribution of the RCV1-v2 Text Categorization Test Collection (12-Apr- 2004 Version).: "Text classification (from REUTERS data)" (http://www.jmlr.org/papers/volume5/lewis04a/lyrl2004_rcv1v2_README).

Past Usage: The data were used in the Unsupervised and Transfer Learning challenge: http://www.causality.inf.ethz.ch/unsupervised-learning.php.

Description: We used a subset of the 800,000 documents of the RCV1-v2 data collection.

Preparation: The data were formatted in a bag-of-words representation. The representation uses 47,236 unique stemmed tokens, see http://jmlr.org/proceedings/papers/v27/supplemental/datasetsutl12a.pdf for details. We considered all levels of the hierarchy to select the most promising categories.

Representation: Bag-of-word features.

SET 4.5: YOLANDA

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
regression	R2	0	NaN	1e-07	0	0	0.1	30000	30000	400000	100	4000

This dataset was prepared by T. Bertin-Mahieux from original data provided by This data is a subset of the Million Song Dataset: http://labrosa.ee.columbia.edu/millionsong/ a collaboration between LabROSA (Columbia University) and The Echo Nest. Prepared by T. Bertin-Mahieux. This version was obtained from C. J. Lin.: "YearPredictionMSD Data Set " (https://archive.ics.uci.edu/ml/datasets/YearPredictionMSD).

Past Usage: The Million Song Dataset. Thierry Bertin-Mahieux, Daniel P.W. Ellis and Brian Whitman, Paul Lamere. https://archive.ics.uci.edu/ml/datasets/YearPredictionMSD; http://ismir2011.ismir.net/papers/OS6-1.pdf.

Description: Prediction of the release year of a song from audio features. Songs are mostly western, commercial tracks ranging from 1922 to 2011, with a peak in the year 2000s.

Preparation: The data were obtained by C J Lin. They respect the original representation.

Representation: Features: 12 = timbre average, 78 = timbre covariance.

ROUND 5

SET 5.1: ARTURO

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multiclass	F1	20	1	0.82	0	0	0.5	2733	1366	9565	400	23.91

This dataset was prepared by Sergio Escalera from original data provided by Sergio Escalera, Xavier Baro, Jordi Gonzalez, Miguel A. Bautista, Meysam Madadi, Miguel Reyes, V?ctor Ponce: "Multimodal Gesture Recognition Data set using audio features" (http://sunai.uoc.edu/chalearn/).

Past Usage: The data from which the ABGR was generated have been used by several people in two challenges (Multimodal Gesture Recognition and Looking at People Challenges), the number of registered participants exceeded 200 hundred (al least 20 people participated throughout the final stages and developed highly competitive methods). More information can be found in: Sergio Escalera et al. Multi-modal Gesture Recognition Challenge 2013: Dataset and Results. Proc. of ICMI 2013, pp. 445-452, 2013, and in Sergio Escalera et al. ChaLearn Looking at People Challenge 2014: Dataset and results, ECCV-Chalearn workshop 2014.

Description: This is a dataset of gesture recognition. It comprises all of the samples (training+validation+test) of the original data set, a total of 13664 samples. 20 classes of gestures were considered and only audio-based features are used to represent clips.

Preparation: For the purpose of the AutoML challenge, all samples were merged. Frames of the clip were first described by the 13 MEL coefficients extracted from the audio signal, and then clustered o generate a 200-words vocabulary, which was used to represent the videos.

Representation: Bag-of-Visual-Words using Mel coefficients coordinates, vocabulary of 200 codewords was considered.

SET 5.2: CARLO

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	PAC	2	0.097	0.0027	0	0	0.5	10000	10000	50000	1070	46.73

This dataset was prepared by Bisakha Ray from original data provided by Bisakha Ray, Javier Orlandi, Olav Stetter, Isabelle Guyon: "Connectomics-features-normal-1" (http://www.kaggle.com/c/connectomics/leaderboard).

Past Usage: Used for Connectomics challenge at http://www.kaggle.com/c/connectomics/leaderboard.

Description: This is a dataset of Connectomics Challenge. The outcome considered is presence or absence of connection.

Preparation: For the purpose of the AutoML challenge, all samples were merged and the data were freshly randomly split in three sets: training, validation, and test. The order of the features (pixels) was also randomize, after adding a few distractor features (probes) that are permuted versions of real features.

Representation: neuronal connection.

SET 5.3: MARCO

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multilabel	AUC	180	0.76	0.99	0	0	0	20482	20482	163860	15299	10.71

This dataset was prepared by Yin Aphinyanaphongs from original data provided by William Hersh: "Ohsumed TEXT dataset" (http://ir.ohsu.edu/ohsumed/ohsumed.html).

Past Usage: Many studies have used the ohsumed corpora for information retrieval research in the biomedical literature. See http://scholar.google.com/scholar?es_sm=91&um=1&ie=UTF-8&lr=&cites=13802943827211985373 for a listing of papers that cite this work.

Description: See the dataset url for more information. To summarize, these are biomedical articles from 1987 to 1991 from over 270 medical journals from the primary literature that contain titles, abstracts, human-assigned MeSH terms, publication types, authors, and source.

Preparation: The original dataset contains 348,566 references from MEDLINE. , the on-line medical information database, consisting of titles and/or abstracts from 270 medical journals over a five-year period (1987-1991). The available fields are title, abstract, MeSH indexing terms, author, source, and publication type. We applied the following steps in

order: (1) Filter references to contain an abstract, contain a title, and is of type "journal article." (2) Concatonate title (.T), abstract (.W), Author (.A), and Source (.S). (3) Replace all punctuation with blanks. (4) Remove stopwords defined in nltk.corpus. (5) Set minimum token occurence to 50. (6) Apply tf-idf to the resulting corpus. The classification targets are determined by ranking the top 200 mesh terms assigned to all the documents and building independent classification tasks for each MeSH term. See Google sheet at https://docs.google.com/spreadsheets/d/1Kihqtds6mYVWTwV4l5qRCNuUCZOnXkpM9zYbbtHT3ts/edit#gid=1366784086 for initial performance estimates on the various classification tasks.

Representation: Bag-of-word features.

SET 5.4: PABLO

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
regression	ABS	0	NaN	0.11	0	0	0.5	23565	23565	188524	120	1571.03

This dataset was prepared by Bisakha Ray from original data provided by Jaume Bacardit and Natalio Krasnogor: "The ICOS PSP benchmarks repository" (http://icos.cs.nott.ac.uk/datasets/psp_benchmark.html).

Past Usage: M. Stout, J. Bacardit, J.D. Hirst, N. Krasnogor Prediction of recursive convex hull class assignments for protein residues in Bioinformatics, 24(7):916-923, April 2008.

Description: This is a dataset of PSP benchmark repository. The outcome considered is protein structure prediction. It consists of 60 real-valued features for regression. The fold considered is TrainFold09w1.

Preparation: For the purpose of the AutoML challenge, all samples were merged and the data were freshly randomly split in three sets: training, validation, and test. The order of the features (pixels) was also randomize, after adding a few distractor features (probes) that are permuted versions of real features.

Representation: Protein structure features.

SET 5.5: WALDO

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
multiclass	BAC	4	1	0.029	0	1	0.5	2430	2430	19439	270	72

This dataset was prepared by Isabelle Guyon from original data prepared from various sources, all in the feature representation designed by Jose Fonollosa: "Cause-Effect Pairs challenge data (in Jarfo representation)" (http://www.causality.inf.ethz.ch/cause-effect.php?page=data).

Past Usage: The data were used in the cause-effet pairs challenge in their raw representation.

Description: We provided hundreds of pairs of real variables with known causal relationships from domains as diverse as chemistry, climatology, ecology, economy, engineering, epidemiology, genomics, medicine, physics. and sociology. Those were intermixed with controls (pairs of independent variables and pairs of variables that are dependent but not causally related) and semi-artificial cause-effect pairs (real variables mixed in various ways to produce a given outcome). The goal is to classify the pairs in one of 4 classes "A causes

B", B causes A", "A and B are independent" or "A and B are dependent but not causally related".

Preparation: One of the participant extracted features of the joint distribution of the variable pairs. Those feature (which we provide), include information theoretic features such as conditional entropy and results of independence tests.

Representation: Features.

Appendix C. Datasets of the 2018 AutoML challenge

C.1. PHASE 1: development

SET 1.1: ADA

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	AUC	2	1	0.33	0	0	0	41471	415	4147	48	86.39

This dataset is a version of the Adult data set used in round 0 of the 2015/2016 AutoML challenge. It was prepared by Isabelle Guyon from original data extracted by Barry Becker from the 1994 Census database. The data was donated to the UCI repository by Ron Kohavi: "Adult data set" (https://archive.ics.uci.edu/ml/datasets/Adult).

Past Usage: It was used previously used in the Performance Prediction challenge, the Model Selection game, and the Agnostic Learning vs. Prior Knowledge (ALvsPK) challenge. Adult, a version of ADA was used in round 0 of the 2015/2016 AutoML challenge.

Description: The task of ADA is to discover high revenue people from census data. This is a two-class classification problem. The raw data from the census bureau is known as the Adult database in the UCI machine-learning repository. The 14 original attributes (features) include age, workclass, education, education, marital status, occupation, native country, etc. Categorical features were eliminated and the original numerical features were preprocessed to obtain 48 attributes.

Preparation: A set of reasonably clean records was extracted using the following conditions: $((AAGE > 16)$ and $(AGI > 100)$ and $(AFNLWGT > 1)$ and $(HRSWK > 0))$.

Representation: The features include age, workclass, education, etc.

SET 1.2: ARCENE

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	AUC	2	0.78	0.54	0	0	0	700	100	100	10000	0.01

This dataset was made available by Isabelle Guyon. The tasks consist in distinguishing cancer versus normal patterns from mass-spectrometric data. This is a two-class classification problem with continuous input variables. More information on the dataset is available from this link: https://archive.ics.uci.edu/ml/datasets/Arcene

Past Usage: The Arcene dataset has been used previously in the NIPS 2003 feature selection challenge.

Description: The data were obtained from two sources: The National Cancer Institute (NCI) and the Eastern Virginia Medical School (EVMS). All the data consist of mass-spectra obtained with the SELDI technique. The samples include patients with cancer (ovarian or

prostate cancer), and healthy or control patients. Ovarian cancer samples comprise 253 spectra, including 91 controls and 162 cancer spectra. Regarding the prostate cancer, there are 253 normal samples and 69 disease samples. The number of original features is 15154.

Preparation: The samples were prepared as described in http://clopinet.com/isabelle/Projects/NIPS2003/Slides/NIPS2003-Datasets.pdf. After preprocessing, 3000 informative features and 7000 probes were included in the data set.

Representation: See http://clopinet.com/isabelle/Projects/NIPS2003/Slides/NIPS2003-Datasets.pdf.

SET 1.3: GINA

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	AUC	2	1	0.97	0.31	0	0	31532	315	3153	970	3.25

This dataset was prepared by Isabelle Guyon. The associated task is handwritten digit recognition. Specifically, the problem of separating the odd numbers from even numbers. This is a twoclass classification problem with sparse continuous input variables, in which each class is composed of several clusters. It is a problems with heterogeneous classes.

Past Usage: It was used previously used in the Performance Prediction challenge, the Model Selection game, and the Agnostic Learning vs. Prior Knowledge (ALvsPK) challenge.

Description: The dataset was formed with instances from the MNIST dataset that is made available by Yann LeCun at http://yann.lecun.com/exdb/mnist/.

Preparation: The following process was followed for preparing the data: Pixels that were 99% of the time white were removed. This reduced the original feature set of 784 pixels to 485. The original resolution (256 gray levels) was kept. The feature names are the (i,j) matrix coordinates of the pixels (in a 28x28 matrix). Two digit numbers were generated by dividing the datasets into to parts and pairing the digits at random. The task is to separate odd from even numbers. The digit of the tens being not informative, the features of that digit act as distracters.

Representation: Pixels from the images were used as features. More information on the dataset can be found in the following link: http://clopinet.com/isabelle/Projects/agnostic/Dataset.pdf

SET 1.4: GUILLERMO

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	AUC	2	0.67	0.53	0	0	0	5000	5000	20000	4296	4.65

This data set was prepared by Luis Pellegrin and Hugo Jair Escalante. It comprises preprocessed image-text pairs. Original data was obtained from the SAIAPR TC12 benchmark, provided and prepared by Michael Grubinger and Hugo Jair Escalante (http://imageclef.org/SIAPRdata).

Past Usage: The GUILLERMO data set was previously used in the RICATIM - Text Image Matching challenge.

Description: The prediction task consists of determining whether a pair of image - text is related. A word (text) is relevant to an image (and vice versa) if the word was used as label for the image in the original SAIAPR TC12 benchmark. Thus, the image labeling

problem is casted as one of binary classification. Images and words are encoded via learned representations as described below, both representations are concatenated to generate the input space of instances. Negative pairs were generated by sampling irrelevant labels.

Preparation: The data set was generated by sampling around 3,000 labeled images from the SAIAPR TC12 data set (formed by 20,000 images). The data set is almost balanced.

Representation: Images were represented by the response of a pretrained CNN (penultimate layer of VGG-16). Words were represented by their Word2Vec representation. An embedding of 200 dimensions was considered, the embedding was trained with the Wikipedia collection.

SET 1.5: RL

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	AUC	2	0.10	0.99	0.11	1	0	24803	0	31406	22	1427.5

This is a confidential dataset provided by the 4paradigm company, hence we cannot disclose confidential information about it. Although this dataset is publicly available as it was used for the feedback phase of the 2018 AutoML challenge.

Past Usage: This data set was specifically generated for the 2018 AutoML challenge.

Description: The RL data set is associated to a real-world recommendation task involving real users. Items can be: video, audio and activities recommendations, and labels are generated by clicks from users. Instances in this dataset are chronologically ordered, real recommendations and clicks of users from a small time period were considered.

Preparation: A small sample from real recommendations-clicks was taken for preparing this data set. The class imbalance ratio for this dataset was determined to resemble the actual imbalance ratio observed in practice in the associated recommendation task.

Representation: Processed numerical and categorical features encoding descriptive information were made available with this data set.

C.2. PHASE 2: final AutoML testing

SET 2.1: PM

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	AUC	2	0.01	1	0.11	1	0	20000	0	29964	89	224.71

This is a confidential dataset provided by the 4paradigm company, hence we cannot disclose confidential information about it.

Past Usage: This data set was specifically generated for the 2018 AutoML challenge.

Description: The PM data set is associated to a real-world click prediction task involving real users. More specifically a search-result-click through rate-prediction problem is considered. Instances in this dataset are chronologically ordered, real clicks of users from a small time period were considered.

Preparation: A small sample from real search-results-clicks was taken for preparing this data set. The class imbalance ratio for this dataset was determined to resemble the actual imbalance ratio observed in practice in the associated recommendation task.

Representation: Processed categorical features encoding descriptive information were made available with this data set.

SET 2.2: RH

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	AUC	2	0.04	0.59	0	1	0	28544	0	31498	76	414.44

This is a confidential dataset provided by the 4paradigm company, hence we cannot disclose confidential information about it.

Past Usage: This data set was specifically generated for the 2018 AutoML challenge.

Description: The RH data set is associated to a real-world recommendation task involving real users. Items can be: video, audio and activities recommendations, and labels are generated by clicks from users. Instances in this dataset are chronologically ordered, real recommendations and clicks of users from a small time period were considered.

Preparation: A small sample from real recommendations-clicks was taken for preparing this data set. The class imbalance ratio for this dataset was determined to resemble the actual imbalance ratio observed in practice in the associated recommendation task.

Representation: Processed numerical and categorical features encoding descriptive information were made available with this data set.

SET 2.3: RI

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	AUC	2	0.02	0.91	0.26	1	0	26744	0	30562	113	270.46

This is a confidential dataset provided by the 4paradigm company, hence we cannot disclose confidential information about it.

Past Usage: This data set was specifically generated for the 2018 AutoML challenge.

Description: The RI data set is associated to a real-world recommendation task involving real users. Items can be: video, audio and activities recommendations, and labels are generated by clicks from users. Instances in this dataset are chronologically ordered, real recommendations and clicks of users from a small time period were considered.

Preparation: A small sample from real recommendations-clicks was taken for preparing this data set. The class imbalance ratio for this dataset was determined to resemble the actual imbalance ratio observed in practice in the associated recommendation task.

Representation: Processed numerical and categorical features encoding descriptive information were made available with this data set.

SET 2.4: RICCARDO

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	AUC	2	0.33	0.51	0	0	0	5000	5000	20000	4296	4.65

This data set was prepared by Luis Pellegrin and Hugo Jair Escalante. It comprises preprocessed image-text pairs. Original data was obtained from the common objects in context collection (http://cocodataset.org/).

Past Usage: This data set was specifically generated for the 2018 AutoML challenge. It was built following a similar methodology as with the GUILLERMO data set above.

Description: The prediction task consists of determining whether a pair of image - text is related. A text (text could be either a word or the caption accompanying an image) is relevant to an image (and vice versa) if the text was used as caption (or word in the caption) for the image in the original MS COCO benchmark. Thus, the image captioning/labeling problem is casted as one of binary classification. Images and texts are encoded via learned representations as described below, both representations are concatenated to generate the input space of instances. Negative pairs were generated by sampling irrelevant labels.

Preparation: This data set was generated by sampling labeled images from the MS COCO data set. Texts were generated by either captions or words appearing in the captions. The data set is almost balanced.

Representation: Images were represented by the response of a pretrained CNN (penultimate layer of VGG-16). Texts were represented by their Word2Vec representation. An embedding of 200 dimensions was considered, the embedding was trained with the Wikipedia collection. For words, the direct embedding was used. For captions, the average embedding (over words appearing in the caption) was considered.

SET 2.5: RM

Task	Metric	C	Cbal	Sparse	Miss	Cat	Irr	Pte	Pva	Ptr	N	Ptr/N
binary	AUC	2	0.001	1	0.11	1	0	26961	0	28278	89	317.73

This is a confidential dataset provided by the 4paradigm company, hence we cannot disclose confidential information about it.

Past Usage: This data set was specifically generated for the 2018 AutoML challenge.

Description: The RM data set is associated to a real-world click prediction task involving real users. More specifically a search-result-click through rate-prediction problem is considered. Instances in this dataset are chronologically ordered, real clicks of users from a small time period were considered.

Preparation: A small sample from real search-results-clicks was taken for preparing this data set. The class imbalance ratio for this dataset was determined to resemble the actual imbalance ratio observed in practice in the associated recommendation task.

Representation: Processed categorical features encoding descriptive information were made available with this data set.

Appendix D. Methods of the 2015/2016 AutoML challenge

In this appendix, we first present the results of a survey we conduccted after the challenge, then briefly summarize the best methods based on fact sheets and papers presented at the ICML 2016 workshop where the winners presented their results.

D.1. Survey Analysis

Twenty-eight teams responded to a survey we conducted on methods used in the challenge.
Preprocessing. Preprocessing consisted in normalization, feature extraction, and dimensionality reduction. About one half of the respondents performed classical preprocessing

steps, including feature standardization, sample normalization, and replacement of missing values. This is consistent with the frequent use of ensembles of decision trees based on decision thresholds, which do not require complex preprocessing. Other preprocessing steps included grouping modalities for categorical variables (20%) and discretization (4%). Few participants also reported having used non-linear transforms such as log. Most participants did not perform any feature engineering, which can largely be explained by the fact that they did not know the application domain of the data sets. Those who reported using feature extraction either relied on the (embedded) feature learning of their algorithm (21%) or applied random functions (36%). More than 2/3 of the participants used dimensionality reduction, linear manifold transformations (e.g., PCA, ICA) being the most popular (43%). About 1/3 used feature selection alone. Other methods included non-linear dimensionality reduction (e.g., KPCA, MDS, LLE, Laplacian Eigenmaps) and clustering (e.g., K-means).
Predictor. The methods most frequently used involved (ensembles of) decision trees; 75% of the participants reported having used them, alone or in combination with other methods. The challenge setting lent itself well to such methods because each individual base learner trains rapidly and performance improves by increasing the number of learners, making such methods ideal any-time learning machines. Almost 1/2 of the participants used linear methods and about 1/3 used at least one of the following methods: Neural Nets, Nearest Neighbor, and Naive Bayes. The logistic loss was frequently used (75%). This may be due to the fact that producing probability-like scores is the most versatile when it comes to being able to be judged with a variety of loss functions. About 2/3 of the participants reported having used knowingly some form of regularization; two-norm regularization was slightly more popular than one-norm regularization.
Model selection and ensembling. About 2/3 of the respondents used one form of cross-validation for model selection; the rest used just the leaderboard. This may be due to the fact that the validation sets were not small for the most part. While K-fold cross-validation and leave-one-out remain the most popular, 20% of the respondents used the out-of-bag estimator of bagging methods and 10% used bi-level optimization methods. 4% reported transferring knowledge from phase to phase. However, such a strategy may be worth considering since both winners of phase AutoML5 used it. Only 18% of the respondents did not choose ensemble methods. For those who did, boosting and bagging were the most common—60% reported having used one of the two.
Implementation. Most respondents could not reliably evaluate how their methods scaled computationally. We are at least assured that they delivered results in less than 20 minutes on every data set, because this was the time limit for the execution. Most respondents claimed to have developed a simple method, easy to implement and parallelize (75% used multi-processor machines, 32% used algorithms run in parallel on different machines), but few claimed that their method was original or principled, and most relied on third-party libraries; scikit-learn, which was used in the starting kit, was frequently used. Luckily, this also resulted in code that was made available as open source—with only 10% exceptions. Python was used by 82% of the respondents. This is also explained by the fact that the starting kit was in Python. Although Codalab allows users to submit any Linux executable, the organizers provided no support for this. Even then, 25% used at least one of the following languages: C/C++, Java, or R, sometimes in combination with Python. The fact that the Codalab backend ran on Linux may also explain that 86% of the respondents ran on Linux;

others used Windows or MacOS. Memory consumption was generally high (more than half of the respondents used between 8 and 32 GB, and 18% used more that 32 GB). Indeed, when we introduced sparse data in Round 3, the sample code was memory demanding and we had to increase the memory on the server up to 56 GB. Unfortunately, this remained a problem until the end of the challenge—which we traced to an inefficient implementation of the data reader and of Random Forest for sparse matrices.

D.2. Fact Sheets

The methods of top ranking participants of the 2015/2016 challenge are briefly summarized.

ideal.intel.analytics and amsl.intel.com

The proprietary solution of the Intel team was presented by Eugene Tuv at the CiML workshop at NIPS, Montreal, December 2015 [16]. It is a fast implementation of tree-based methods in C/C++, which was developed to drive acceleration of yield learning in semiconductor process development. Using this software, the Intel team consistently has ranked high in ChaLearn challenges since 2003. The method is based on gradient boosting of trees built on a random subspace dynamically adjusted to reflect learned features relevance. A Huber loss function is used. No pre-processing was done, except for feature selection (Tuv et al., 2009). The classification method called Stochastic Gradient Tree and Feature Boosting selects a small sample of features at every step of the ensemble construction. The sampling distribution is modified at every iteration to promote more relevant features. The SGTFB complexity is of the order of $N_{tree}N_{tr}logN_{tr}logN_{feat}$, where N_{tree} is the number of trees, N_{tr} the number of training examples, and N_{feat} the number of features.

aad_freiburg

The open-source solution of AAD Freiburg uses a heterogeneous ensemble of learning machines (auto-sklearn (Feurer et al., 2015a,c)) combining the machine learning library scikit-learn (Pedregosa et al., 2011) with the state-of-the-art SMBO method SMAC to find suitable machine learning pipelines for a data set at hand. This is essentially a reimplementation of Auto-WEKA. To speed up the optimization process they employed a meta-learning technique (Feurer et al., 2015b) which starts SMAC from promising configurations of scikit-learn. Furthermore, they used the outputs of all models and combined these into an ensemble using ensemble selection. Their latest version uses a python reimplementation of SMAC (Hutter et al.) of Bayesian Optimization with Random Forests applied to a flexible configuration space describing scikit-learn. For the GPU version (Mendoza et al., 2016), they used the Java version of SMAC to tune auto-sklearn and deep neural networks implemented in Lasagne/Theano (Dieleman et al., 2015; Theano Development Team, 2016).

jrl44, backstreet.bayes, and lise_sun

16. http://ciml.chalearn.org/home

Freeze Thaw Ensemble Construction (Lloyd, 2016) of J. Lloyd (a.k.a. jrl44 and backstreet.bayes) is a modified version of the Freeze Thaw Bayesian optimization algorithm (Swersky et al., 2014) for ensemble construction. The strategy is to keep training the most promising members of an ensemble, while freezing the least promising ones, which may be thawed later. Probabilistic models based on Gaussian processes and decision trees are used to predict which ensemble member should be trained further. Joining late in the challenge, L. Sun made an entry in AutoML5 that ranked third using a similar approach (Sun-Hosoya, 2016).

abhishek4

AutoCompete of (Thakur and Krohn-Grimberghe, 2015) is an automated machine learning framework for tackling Machine Learning competitions. This solution performed well in late rounds of the AutoML challenge and won the GPU track (Thakur, 2016). The pipeline includes (1) stratified data splitting, (2) building features, (3) feature selection, (4) performing model and hyper-parameter selection (Random Forests, Logistic Regression, Ridge Regression, Lasso, SVM, Naive Bayes, and Nearest Neighbors), and (5) ensembling solutions. Search space is specified with prior knowledge on similar data sets (a form of meta-learning). Thakur found that this strategy is faster and yields comparable results to hyperopt.The underlying implementation is based purely on Python and scikit-learn with some modules in Cython. Their GPU solution is an advanced version of the AutoCompete solution, which uses Neural Networks built with Keras (Chollet, 2015).

djajetic

Djajetic (Jajetic, 2016a) is based on heterogeneous ensembles of models obtained by searching through model-space and adjusting hyper-parameters (HP) without any communication between models. Jajetic believes that this makes search more effective in non-convex search spaces. This strategy lends itself well to efficient and simple parallelization. The search space and ensembling properties for each individual model is defined in a separate Python script. Each model is trained and explores its own parameter space and only communicates its training error and best prediction results to the outside. The ensembling module operates in a hierarchical manner. It uses only the N best HP settings from each model, based on the training error, and only M best models from each model group. For the GPU track, Jajetic used a Neural Network (Jajetic, 2016b) based on the Lasagne and Theano libraries.

marc.boulle

Orange, the main French telecommunication operator, has developed the Khiops, which they made avaiable for licensing. The software was designed to address the needs of Orange to analyze their data accross a wide range of cases, without hyper-parameter tuning, and provide solutions that are robust and understandable with modest computational resources. Khiops exploits regularized methods for variable preprocessing, variable selection, variable construction for multi-table data mining, correlation analysis via k-coclustering, model averaging of selective naive Bayes classifiers and regressors. The classifier called Selective Naive Bayes (SNB) (Boullé, 2007, 2009) extends the Naive Bayes classifier using an

optimal estimation of the class conditional probabilities, a Bayesian variable selection and a Compression-based Model Averaging. The same framework was extended to regression in (Hue and Boullé, 2007). The Khiops tool was used throughout the challenge, using python scripts to be compliant to the challenge settings. Beyond the necessary but easy adaptation to the input/output requirements, the python scripts also had to manage the sparse format, the any-time learning settings and the scoring metrics, which were specific to the AutoML challenge and not supported by Khiops.

Appendix E. Methods of the 2018 AutoML challenge

E.1. Survey Analysis

Eleven teams responded to a survey we conducted on methods used in the 2018 challenge. The answers to this survey were consistent with the one reported in Appendix D.1. In the following we briefly summarize the main findings.
Preprocessing. 73% of teams applied feature standardization, 54% of teams applied a preprocessing to replace missing values, and 37% applied data normalization. Interestingly, the winning team applied data discretization and scaling in addition to the other preprocessing procedures. Regarding feature extraction, most teams adopted either trained feature extractors or random functions in the same proportion. More than half of the surveyed teams performed linear transformations of the input space, a third of teams performed feature selection. **Predictor.** Decision trees was the predictive model adopted by most participants (9 out of 11) that is 81%, the rest of teams used linear models. Hinge loss with 1 or 2 norm regularization was adopted in by most of the teams. **Model selection and ensembling.** As model selection criterion, the usual $k-$fold cross validation and the feedback obtained from the leader board were adopted by 50% of the teams each. Interestingly, all teams that filled in the survey adopted an ensemble methodology for generating the final predictor (mostly boosting-based ensembles). This is consistent with the answers observed in the previous edition of the challenge. **Implementation.** Python was used by all participants and about 20% of teams reported using the scikit-learn library (we believe that most, if not all, participants relied on this library, though).

E.2. Fact Sheets

The methods of the top ranking participants of the 2018 challenge are briefly summarized.

aad_freiburg

PoSH Auto-sklearn (*Portfolio Successive Halving* combined with Auto-sklearn) is the solution of the aad_freiburg team, which obtained the best performance in the 2018 challenge. PoSH Auto-sklearn uses a fixed portfolio of machine learning pipeline configurations on which it performs successive halving. If there is time left, it uses the outcome of these runs to warmstart a combination of Bayesian optimization and successive halving. Greedy submodular function maximization was used on a large performance matrix of ≈421 configurations run on ≈421 datasets to obtain a portfolio of configurations that performs well on a diverse set of datasets. To obtain the matrix, aad_freiburg used SMAC (Hutter et al.) to search the space of configurations offline, separately for each of the ≈421 datasets. The

configuration space was a subspace of the Auto-sklearn configuration space: dataset preprocessing (feature scaling, imputation of missing value, treatment of categorical values), but no feature preprocessing (this constraint due to the short time limits / resources in the competition), and one of SVM, Random Forest, Linear Classification (via SGD) or XGBoost. The combination of Bayesian optimization and successive halving is an adaptation of a newly developed method dubbed BO-HB (Bayesian Optimization HyperBand) (Falkner et al., 2018). The solution was further designed to yield robust results within the short time limits as follows: the number of iterations was used as a budget, except for the SVM, where the dataset size was the budget. If the dataset had less than 1000 data points, they reverted to simple cross-validation instead of successive halving. If a dataset had more than 500 features, they used univariate feature selection to reduce the number of features to 500. Lastly, for datasets with more than 45,000 data points, they capped the number of training points to retain decent computational complexity.

narnars0

The narnars0 team proposed an **Automated Machine Learning System for Voting Classifier with Various Tree-Based Classifiers.** This team based their solution in a voting ensemble formed with the following tree-based classifiers: gradient boosting, random forests, and extra-trees classifiers. They optimized the hyperparameters of tree-based classifiers by means of Bayesian optimization. Several machine learning models in scikit-learn were used to implement this system, including narnars0's own Bayesian optimization package, bayeso (`https://github.com/jungtaekkim/bayeso`), which was used to optimize a selection of hyperparameters of classifiers.

wlWangl

An AutoML solution resembling **Q-Learning** in reinforcement learning was proposed by the wlWangl team. This team considers the machine learning design pipeline as composed of three phases: data preprocessing, feature selection, and classification. Each phase associated to a set of methods. They view the candidate methods in each phase as the states of Q-Learning. The classification performance of the pipeline representing the reward. This team used Q-Learning to find the pipeline with the maximum reward. To further improve efficiency and robustness of the proposed method, they integrated meta learning and the ensemble learning into the method. Meta learning was used first to initialize the values of Q-Table for Q-Learning. Then, after the Q-Learning, the good discovered pipelines where ensembled with a stacking method.

thanhdng

The solution bt thanhdng was based on the ensemble solution provided as **starting kit** for the competition. Basically, this team adjusted the parameters of the ensemble (increasing the number of learning cycles and estimators).

Appendix F. Result tables of all 30 dataset of the 2015/2016 challenge

In this appendix, we provide result tables on which several graphs are based. In Table 7, we reran the code of the participants who made it available to us on all the datasets of the 2015/2016 challenge (the last version of code submitted to the challenge platform). In Figure 24 , we reran again these codes to compute their error bars with bootstrapping. In Tables 8 and 9, we ran four "basic models" with default hyper-parameter settings and with hyper-parameter optimization on all the datasets of the 2015/2016 challenge.

Table 7: **Systematic study of participants' methods:** The team abbreviations are the same as in the previous table. The colors indicate the rounds.

Datasets	aad	abhi	djaj	ideal	jrl44	lisheng	marc	ref
ADULT	0.82	0.82	0.81	0.83	0.81	0.8	0.81	0.82
CADATA	0.8	0.79	0.78	0.81	0.09	0.79	0.64	0.76
DIGITS	0.95	0.94	0.83	0.96	0.73	0.95	0.86	0.87
DOROTHEA	0.66	0.87	0.82	0.89	0.82	0.84	0.79	0.7
NEWSGROUPS	0.48	0.46	0.64	0.59	0.33	0.05	0.38	0.56
CHRISTINE	0.49	0.46	0.48	0.55	0.48	0.46	0.45	0.42
JASMINE	0.63	0.61	0.62	0.65	0.62	0.61	0.56	0.56
MADELINE	0.82	0.59	0.64	0.81	0.57	0.58	0.18	0.53
PHILIPPINE	0.66	0.53	0.52	0.72	0.52	0.52	0.45	0.51
SYLVINE	0.9	0.87	0.89	0.93	0.89	0.87	0.83	0.89
ALBERT	0.38	0.32	0.36	0.37	0.32	0.34	0.35	0.32
DILBERT	0.94	0.79	0.75	0.98	0.21	0.24	0.46	0.79
FABERT	0.36	0.19	0.33	0.35	0.03	0.18	0.21	0.24
ROBERT	0.46	0.33	0.33	0.51	0.21	0.4	0.37	0.36
VOLKERT	0.33	0.26	0.28	0.37	0.11	0.15	0.14	0.25
ALEXIS	0.75	0.65	0.67	0.76	0.62	0.68	0.62	0.64
DIONIS	0.9	0.32	0.75	0.93	0.02	0.87	0.81	0.31
GRIGORIS	0.73	0.76	0.8	0.97	0.54	0.88	0.96	0.75
JANNIS	0.55	0.38	0.41	0.42	0.24	0.36	0.39	0.4
WALLIS	0.71	0.63	0.74	0.71	0.12	0.23	0.58	0.62
EVITA	0.59	0.59	0.58	0.61	0.59	0.59	0.52	0.41
FLORA	0.5	0.51	0.5	0.53	0.02	0.42	0.51	0.37
HELENA	0.22	0.23	0.15	0.25	0.06	0.2	0.19	0.08
TANIA	0.47	0.76	0.39	0.73	0.53	0.6	0.66	0.54
YOLANDA	0.32	0.37	0.29	0.39	0.02	0.24	0.19	0.26
ARTURO	0.75	0.8	0.78	0.77	0.3	0.72	0.7	0.77
CARLO	0.45	0.37	0.43	0.18	0.36	0.4	0.37	0.14
MARCO	0.55	0.71	0.69	0.54	0.66	0.54	0.68	0.25
PABLO	0.3	0.29	0.31	0.27	0.03	0.29	0.25	0.28
WALDO	0.59	0.56	0.57	0.61	0.56	0.56	0.46	0.56

Appendix G. Learning Curve of all 30 datasets of the 2015/2016 challenge

In this appendix we show learning curves on all 30 datasets for two top ranking methods: **auto-sklearn** (aad_freiburg), as a representative of a Bayesian search method and **abhishek** as a representative of a heuristic method. In all figures (Figures G-G), we represent in yellow the learning curve of auto-sklearn within the time budget of the challenge; they are prolongated in green beyond the time budget. We represent in blue the learning curves of abhishek (it was not trivial for us to modify the code of abhishek to extend the learning

Table 8: Performances (original task metrics) of basic models using their **scikit-learn default HP setting**. All negative scores and NaN (due to the fact that algorithm didn't succeed in generating predictions within time limit) are brought to zero.

Rnd	DATASET	KNN	NAIVE BAYES	RANDOMFOREST	SGD(LINEAR)
0	ADULT	0.66±0.01	0.72±0.01	0.786±0.009	0.74±0.01
0	CADATA	0.08±0.03	0.62±0.03	0.73±0.02	0.0±0.0
0	DIGITS	0.661±0.007	0.252±0.007	0.924±0.004	0.758±0.007
0	DOROTHEA	0.01±0.04	0.02±0.06	0.4±0.2	0.5±0.2
0	NEWSGROUPS	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
1	CHRISTINE	0.39±0.07	0.36±0.06	0.39±0.06	0.17±0.04
1	JASMINE	0.54±0.05	0.32±0.06	0.58±0.06	0.54±0.07
1	MADELINE	0.57±0.05	0.17±0.05	0.41±0.05	0.0±0.0
1	PHILIPPINE	0.23±0.04	0.36±0.04	0.46±0.05	0.23±0.03
1	SYLVINE	0.52±0.03	0.78±0.02	0.86±0.01	0.55±0.02
2	ALBERT	0.11±0.02	0.0±0.0	0.19±0.02	0.0±0.0
2	DILBERT	0.0±0.0	0.0±0.0	0.01±0.04	0.0±0.0
2	FABERT	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
2	ROBERT	0.1±0.02	0.16±0.02	0.29±0.02	0.22±0.02
2	VOLKERT	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
3	ALEXIS	0.002±0.001	0.38±0.01	0.001±0.001	0.42±0.01
3	DIONIS	0.02±0.01	0.017±0.009	0.033±0.009	0.0±0.01
3	GRIGORIS	0.04±0.02	0.0±0.0	0.0±0.02	0.62±0.03
3	JANNIS	0.13±0.02	0.29±0.04	0.32±0.01	0.22±0.01
3	WALLIS	0.21±0.02	0.04±0.01	0.34±0.02	0.39±0.02
4	EVITA	0.32±0.06	0.35±0.07	0.18±0.05	0.28±0.07
4	FLORA	0.42±0.04	0.43±0.04	0.29±0.04	0.0±0.0
4	HELENA	0.082±0.009	0.14±0.01	0.15±0.01	0.034±0.006
4	TANIA	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
4	YOLANDA	0.0±0.0	0.24±0.01	0.0±0.0	0.0±0.0
5	ARTURO	0.03±0.02	0.35±0.03	0.49±0.03	0.68±0.03
5	CARLO	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
5	MARCO	0.0±0.004	0.007±0.002	0.0006±0.0003	0.04±0.01
5	PABLO	0.09±0.01	0.25±0.01	0.25±0.01	0.15±0.01
5	WALDO	0.03±0.03	0.23±0.03	0.49±0.04	0.03±0.03

curves beyond the time budget). The scores are computed using the task-specific metrics of the challenge.

We noticed that in about 2/3 of the cases, abhishek's learning curves start quite high but do not improve very much over time, they even sometimes go down, which may be an indication of overfitting. In about 80% of the cases, aad_freiburg's learning curves start lower that the learning curves of abhishek. Hence, in spite of their use of meta-learning, aad_freiburg did not come up with as good heuristic startign points. However, their hyperparameter search is more efficient: in about 1/2 of the cases, they end up higher at the end of the learning curve, within the time budget; in about 80% they end up higher if let run longer (green part of the curve).

These learning curves show that there is still a large margin for improvement in terms of combining techniques.

Table 9: Performances (original task metrics) of basic models using **auto-sklearn-tuned HP setting**. The time limit has been respected for this tuning. All negative scores and NaN (due to the fact that algorithm didn't succeed in generating predictions within time limit) are brought to zero.

Rnd	DATASET	KNN	NAIVE BAYES	RANDOMFOREST	SGD(LINEAR)
0	ADULT	0.748±0.009	0.74±0.01	0.808±0.007	0.777±0.009
0	CADATA	0.48±0.03	0.0±0.0	0.48±0.03	0.52±0.02
0	DIGITS	0.0±0.0	0.59±0.009	0.933±0.004	0.802±0.007
0	DOROTHEA	0.4±0.2	0.4±0.2	0.0±0.0	0.5±0.2
0	NEWSGROUPS	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
1	CHRISTINE	0.47±0.05	0.41±0.06	0.49±0.07	0.5±0.05
1	JASMINE	0.6±0.05	0.52±0.05	0.63±0.06	0.58±0.05
1	MADELINE	0.81±0.03	0.22±0.05	0.76±0.03	0.21±0.05
1	PHILIPPINE	0.55±0.04	0.39±0.04	0.58±0.03	0.45±0.04
1	SYLVINE	0.91±0.01	0.81±0.02	0.89±0.01	0.85±0.01
2	ALBERT	0.0±0.0	0.0±0.0	0.0±0.0	0.25±0.02
2	DILBERT	0.34±0.09	0.0±0.0	0.29±0.09	0.0±0.0
2	FABERT	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
2	ROBERT	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
2	VOLKERT	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
3	ALEXIS	0.0±0.0	0.42±0.01	0.11±0.01	0.267±0.008
3	DIONIS	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
3	GRIGORIS	0.0±0.02	0.55±0.03	0.0±0.02	0.0±0.0
3	JANNIS	0.34±0.02	0.24±0.02	0.33±0.01	0.32±0.04
3	WALLIS	0.26±0.02	0.26±0.02	0.34±0.02	0.18±0.01
4	EVITA	0.2±0.05	0.0±0.0	0.27±0.05	0.15±0.05
4	FLORA	0.0±0.002	0.0±0.0	0.0±0.0	0.0±0.001
4	HELENA	0.14±0.01	0.17±0.01	0.0±0.006	0.0±0.0
4	TANIA	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
4	YOLANDA	0.24±0.01	0.0±0.0	0.24±0.01	0.24±0.01
5	ARTURO	0.66±0.03	0.65±0.03	0.75±0.03	0.51±0.03
5	CARLO	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
5	MARCO	0.0±0.0	0.3±0.04	0.0±0.0	0.0±0.0
5	PABLO	0.25±0.01	0.0±0.0	0.25±0.01	0.25±0.01
5	WALDO	0.45±0.03	0.27±0.03	0.55±0.04	0.35±0.03

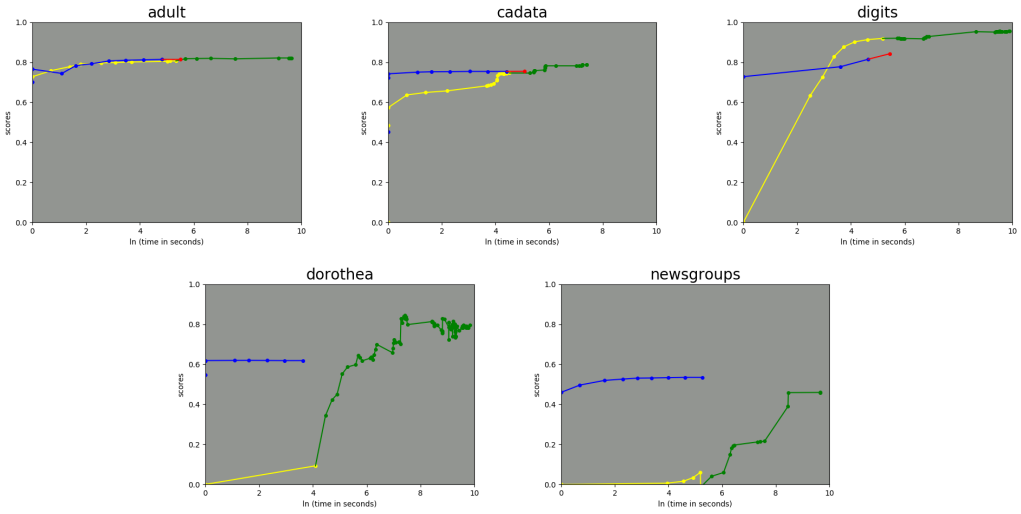


Figure 11: **Learning Curve of 'aad_freiburg' (yellow+green) and 'abhishek' (blue+red) for Round 0.**

We also show in Figures 17-23 all learning curves of a given round ovelaid for the same two high ranking participants ('aad_freiburg' (solid-dots) and 'abhishek' (solid-empty square)). This representation shows that the two optimization strategies differ in their management of time. The 'aad_freiburg' made use of parallelism. Since 4 cores were available on the computers used for the challenge, they started working on 4 (out of 5) datasets simultaneously and started on the fifth one by interrupting working on one of the other datasets, or interleaving work. In contrast, 'abhishek' processed one dataset after the other. For ease of visualisation, we connect the learning curves of 'abhishek' on the various datasets with a dashed line. The error bars were estimated by bootstrapping.

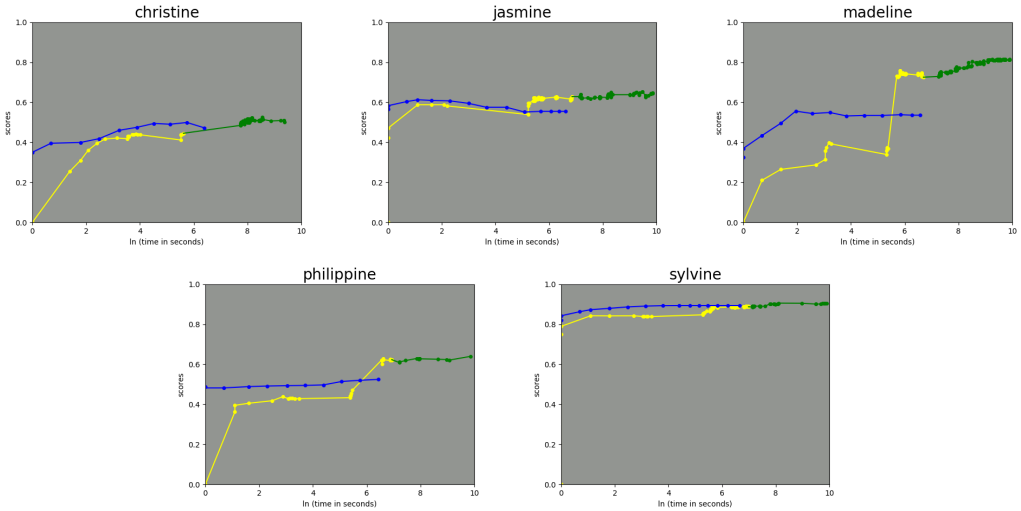


Figure 12: **Learning Curve of 'aad_freiburg' (yellow+green) and 'abhishek' (blue) for Round 1.**

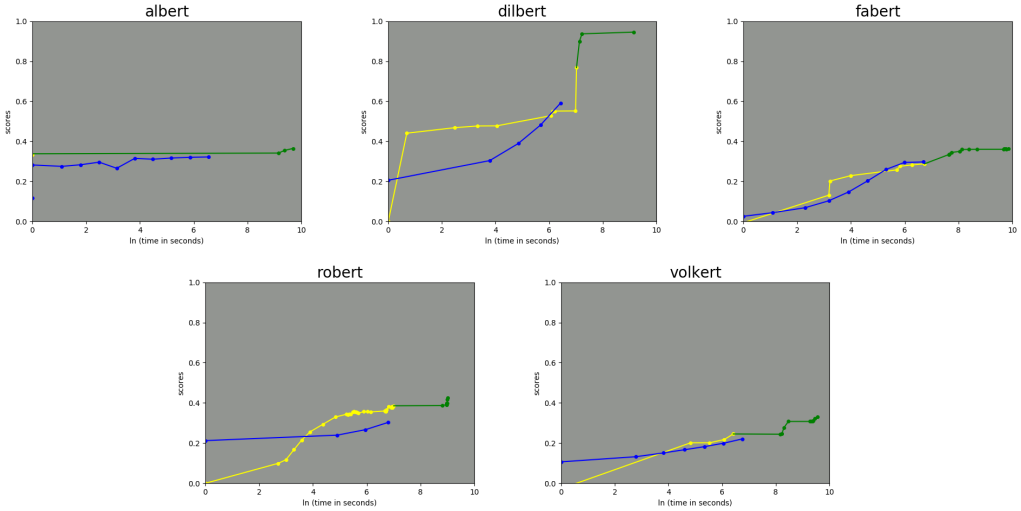


Figure 13: **Learning Curve of 'aad_freiburg' (yellow+green) and 'abhishek' (blue) for Round 2.**

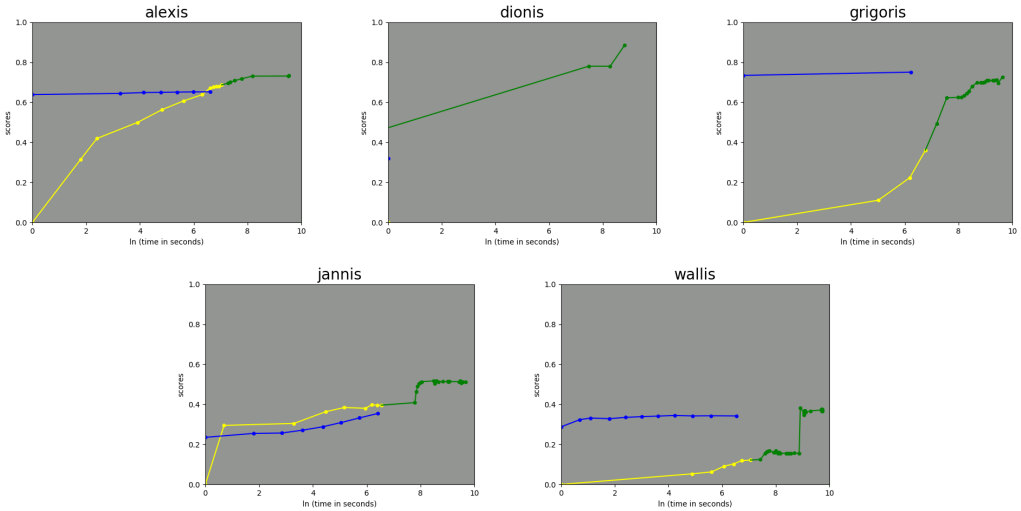


Figure 14: **Learning Curve of 'aad_freiburg' (yellow+green) and 'abhishek' (blue) for Round 3.**

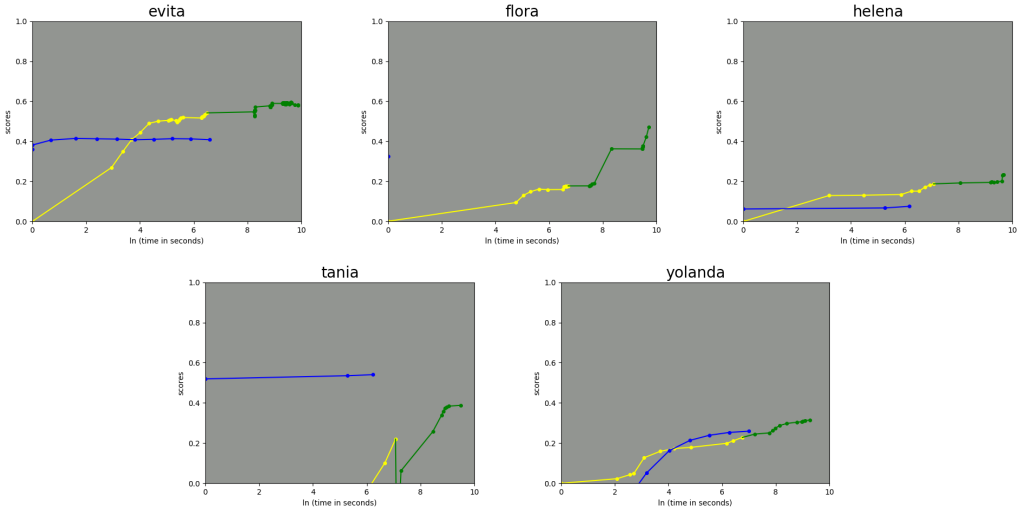


Figure 15: **Learning Curve of 'aad_freiburg' (yellow+green) and 'abhishek' (blue) for Round 4.**

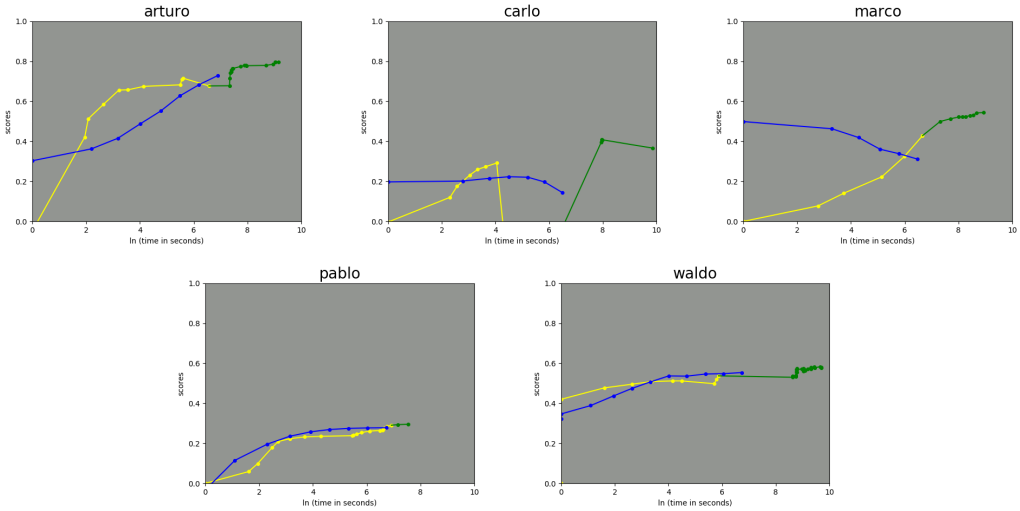


Figure 16: **Learning Curve of 'aad_freiburg' (yellow+green) and 'abhishek' (blue) for Round 5.**

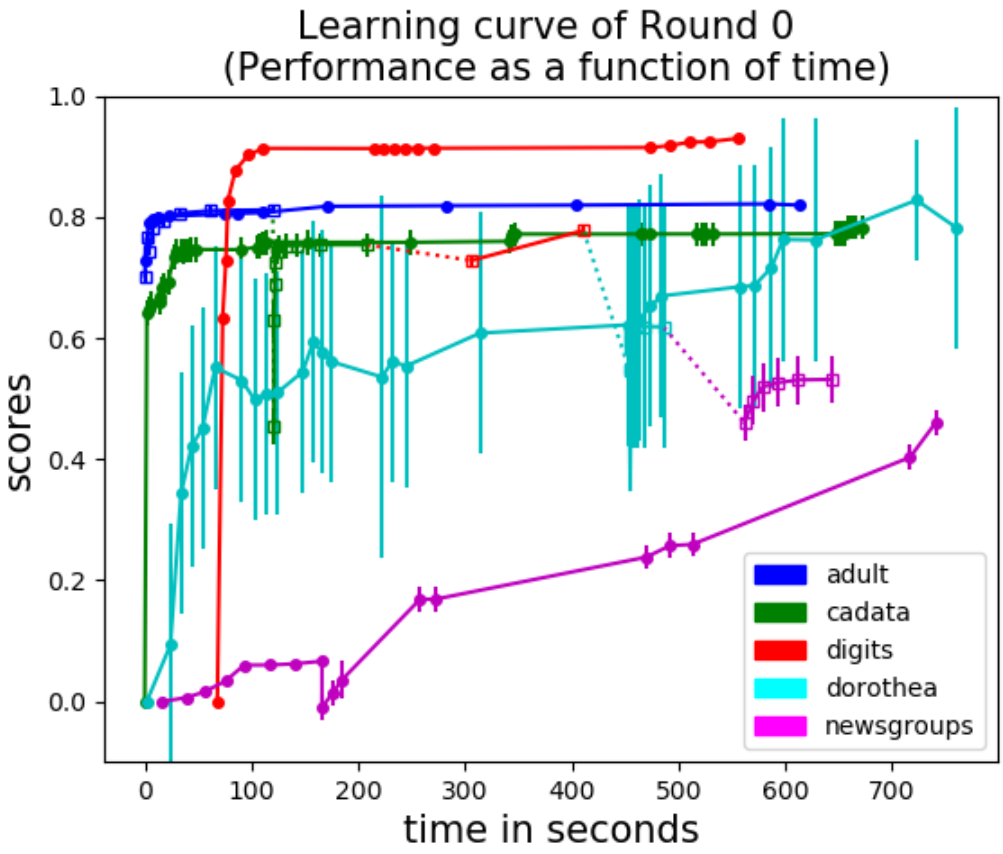


Figure 17: **Learning Curve of 'aad_freiburg' (solid-dots) and 'abhishek' (solid-empty square) for Round 0.**

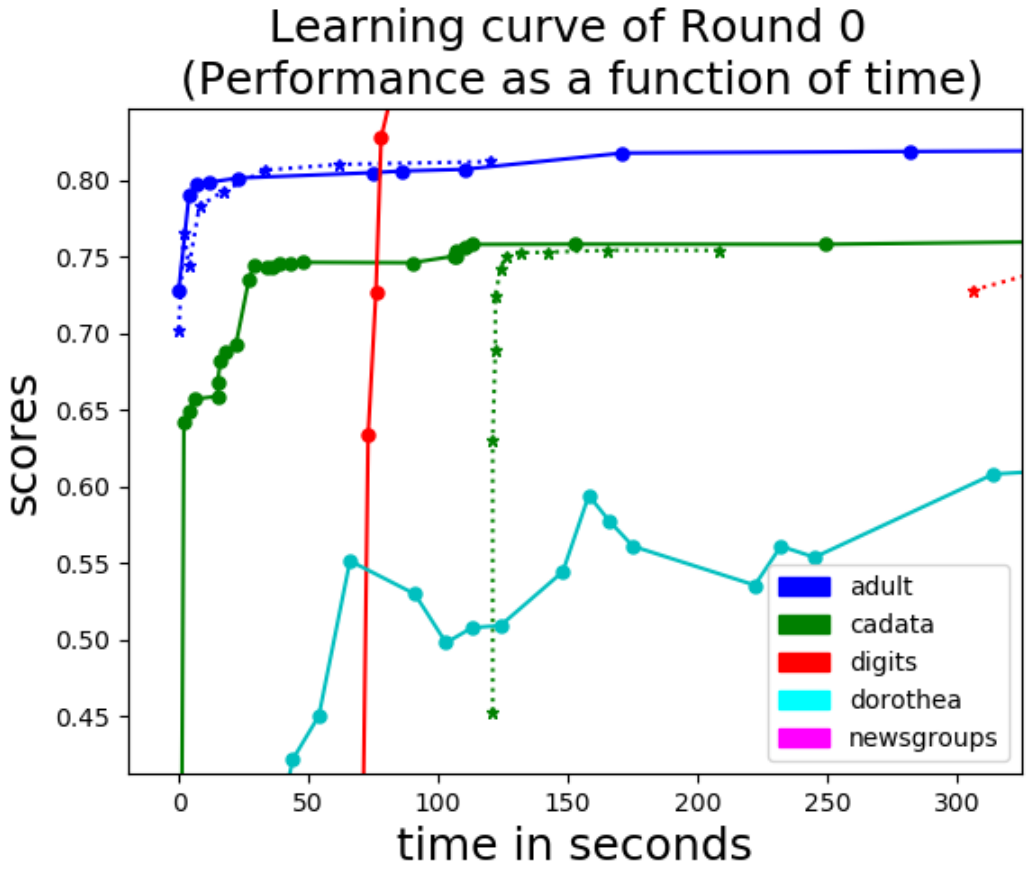


Figure 18: **Partial magnification of Figure 17**

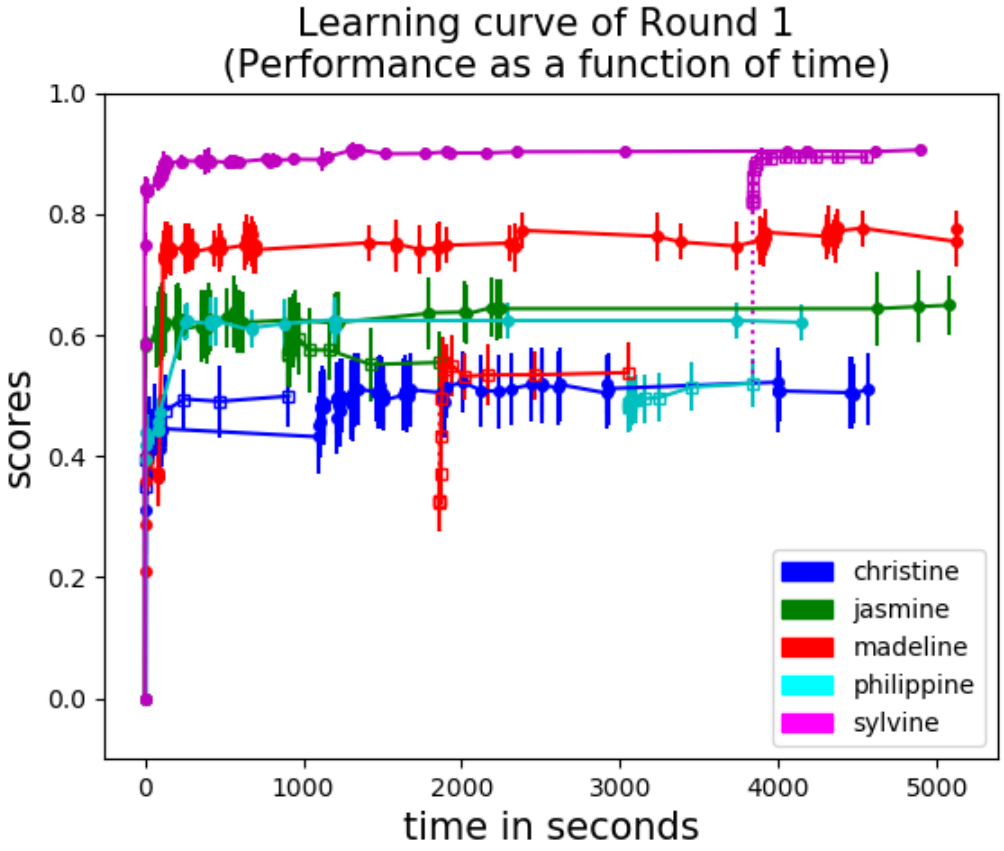


Figure 19: **Learning Curve of 'aad_freiburg' (solid-dots) and 'abhishek' (solid-empty square) for Round 1.**

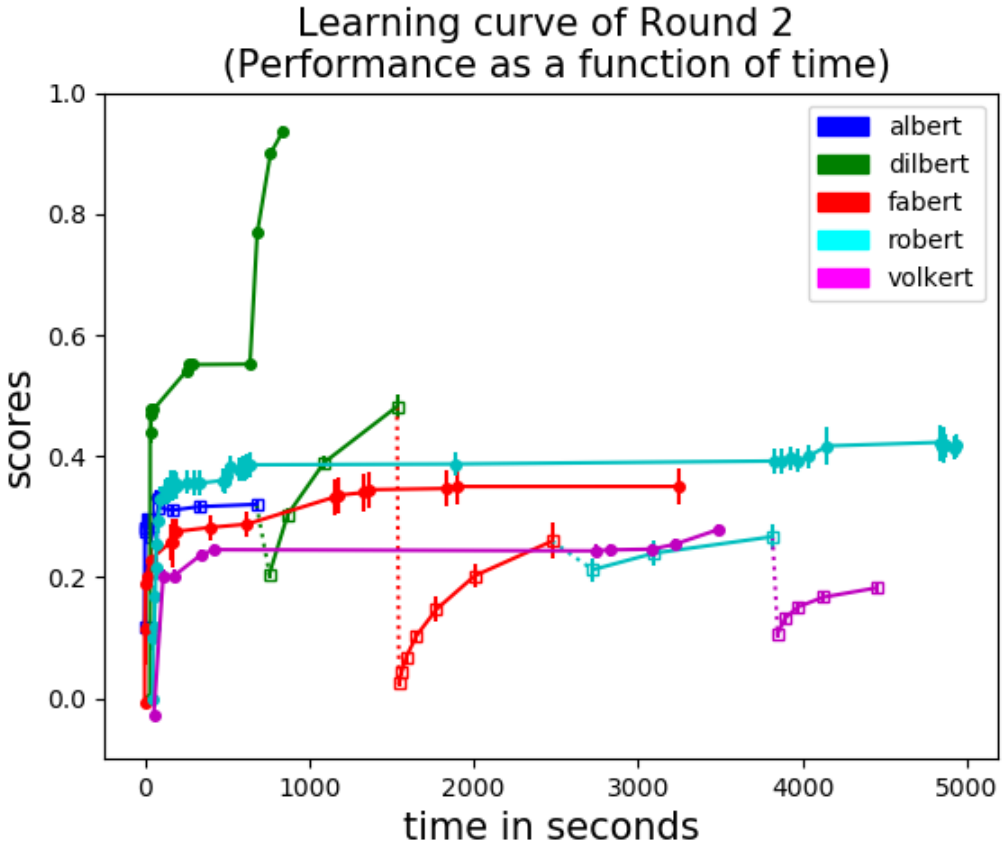


Figure 20: **Learning Curve of 'aad_freiburg' (solid-dots) and 'abhishek' (solid-empty square) for Round 2.**

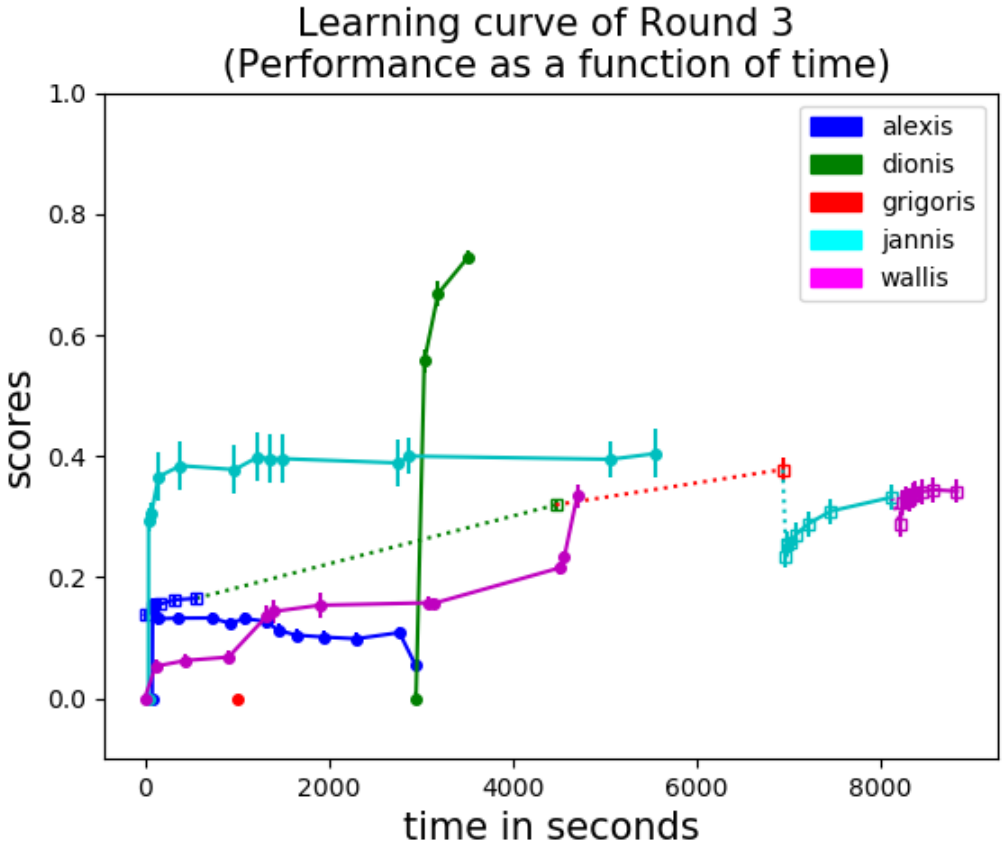


Figure 21: **Learning Curve of 'aad_freiburg' (solid-dots) and 'abhishek' (solid-empty squares) for Round 3.**

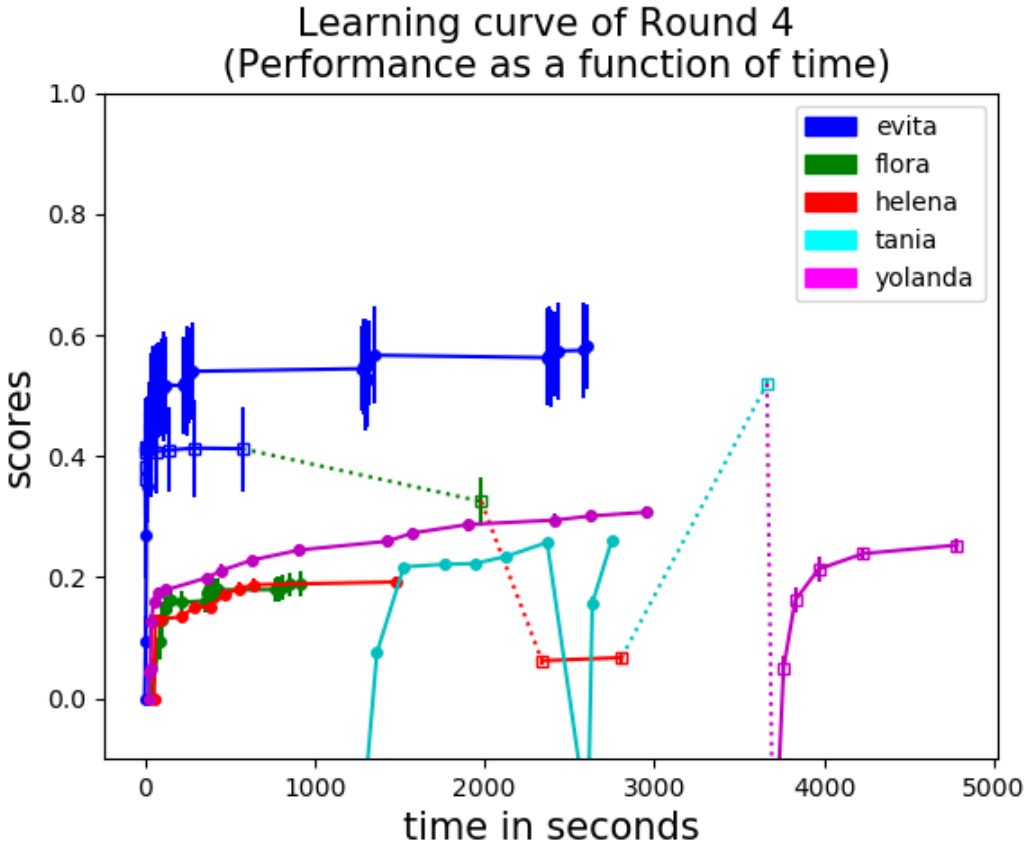


Figure 22: **Learning Curve of 'aad_freiburg' (solid-dots) and 'abhishek' (solid-empty square) for Round 4.**

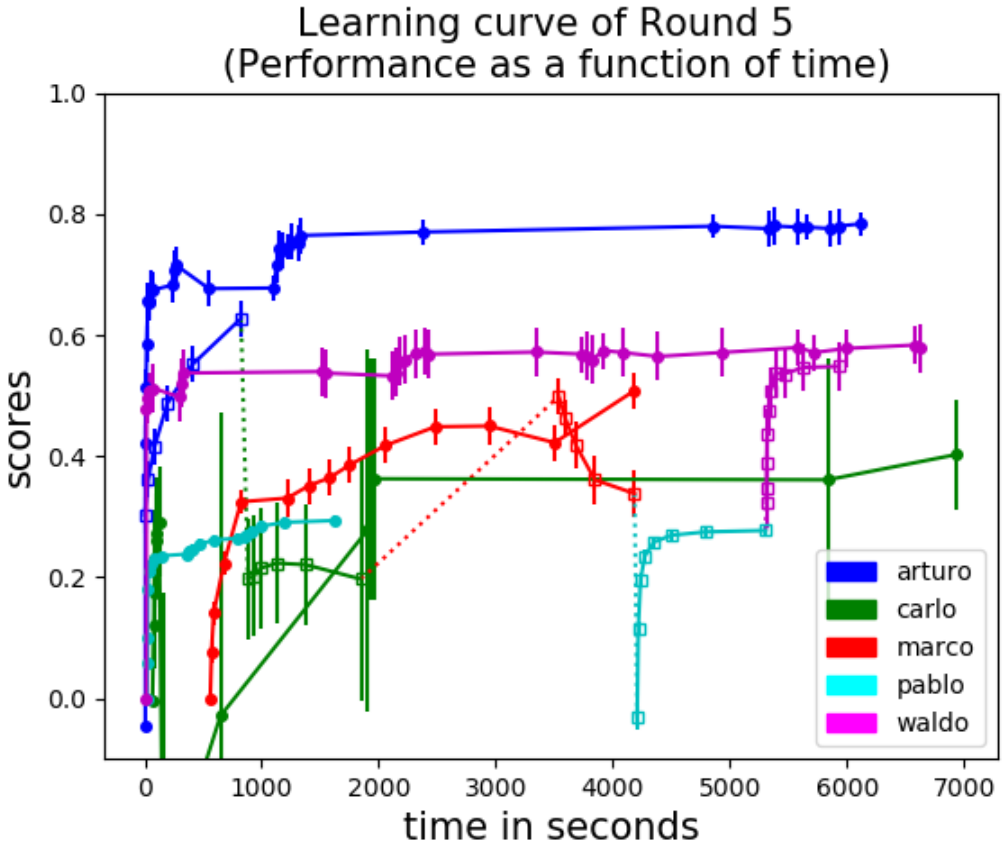


Figure 23: **Learning Curve of 'aad_freiburg' (solid-dots) and 'abhishek' (solid-empty squares) for Round 5.**

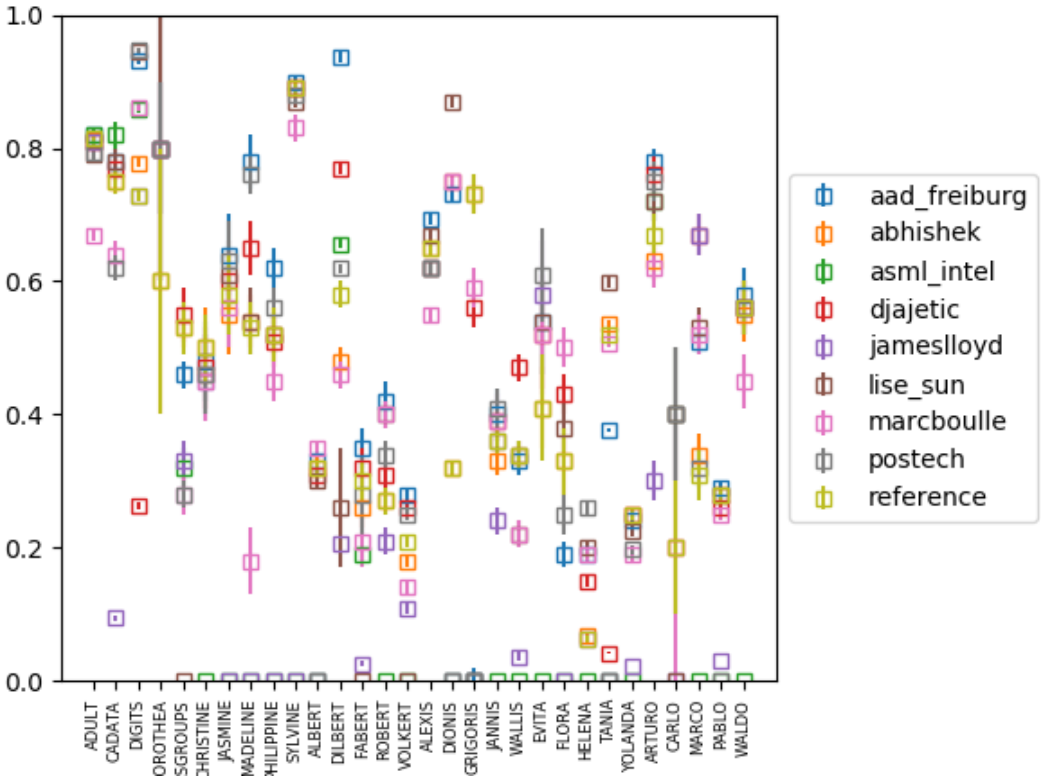


Figure 24: **Scores of participants' methods with error bars.**