AutoML Challenge 2015: Design and First Results

Isabelle Guyon GUYON@CHALEARN.ORG
Berkeley, CA, USA

Kristin Bennett BENNEK@RPI.EDU
RPI, USA

Gavin Cawley G.CAWLEY@UEA.AC.UK
UEA, UK

Hugo Jair Escalante HUGO.JAIR@GMAIL.COM
INAOE, Mexico

Sergio Escalera SERGIO@MAIA.UB.ES
CVC, UAB & UB, Spain

Tin Kam Ho THO@US.IBM.COM
IBM, USA

Núria Macià MACIA.NURIA@GMAIL.COM
Andorra

Bisakha Ray BISAKHA.RAY@NYUMC.ORG
NYU, USA

Mehreen Saeed MEHREEN.MEHREEN@GMAIL.COM
FAST, Pakistan

Alexander Statnikov STATNIKOV@GMAIL.COM
NYU, USA

Evelyne Viegas EVELYNEV@MICROSOFT.COM
Microsoft, USA

Abstract

ChaLearn is organizing the Automatic Machine Learning (AutoML) contest 2015, which challenges participants to solve classification and regression problems *without any human intervention*. Participants' code is automatically run on the contest servers to train and test learning machines. However, there is no obligation to submit code; half of the prizes can be won by submitting prediction results only. Datasets of progressively increasing difficulty are introduced throughout the six rounds of the challenge. (Participants can enter the competition in any round.) The rounds alternate phases in which learners are tested on datasets participants have not seen (AutoML), and phases in which participants have limited time to tweak their algorithms on those datasets to improve performance (Tweakathon). This challenge will push the state of the art in fully automatic machine learning on a wide range of real-world problems. The platform will remain available beyond the termination of the challenge: http://codalab.org/AutoML.

Keywords: AutoML Challenge, machine learning, model selection, meta-learning, representation learning, active learning

1. Introduction

The *AutoML* Challenge is designed to promote research on reducing or removing the need for human interaction in applying machine learning (ML) to practical problems. This refers to all aspects of automating the ML process beyond model selection, hyper-parameter optimization, and model search. Automation is desired for data loading and formatting, detection and handling of skewed data and missing values, selection of learning representation and feature extraction, matching algorithms to problems, acquisition of new data (active learning), creation of appropriately sized and stratified training, validation, and test sets, selection of algorithms that satisfy resource constraints at training and run time, the ability to generate and reuse workflows, meta-learning and learning transfer, and explicative reports. Such automation is crucial for both robots and lifelong autonomous ML.

In Guyon et al. (2015) we describe the details of the design of the AutoML challenge[1] as part of the official IJCNN 2015 competition program. Milestone and final results will be discussed at various workshops, including at the ICML 2015 AutoML workshop. This challenge advances the theoretical underpinnings of model selection by treating it as a joint problem of optimization and statistics. It provides a novel framework for numerical experimentation that reduces user intervention and allows practitioners to evaluate approaches on a large set of problems. In this paper we report some first results from this year's competition.

The new addition of this year's competition is code submission by participants for automatic execution on the open-source platform Codalab[2]. This ensures (1) no human intervention and (2) fair competition since all learning machines (also referred to as learners) are trained and tested on datasets unknown to participants using the same resources. However, there is no obligation to submit code; half of the prizes can be won by submitting prediction results alone. There are six rounds (Prep, Novice, Intermediate, Advanced, Expert, and Master) in which datasets of progressive difficulty are introduced, five per round. The rounds alternate AutoML phases in which submitted code is tested on the Codalab platform with new datasets and Tweakathon phases in which participants improve their methods by tweaking them on those same datasets. During Tweakathon phases participants are free to use their own computational resources.

2. Challenge Design

This challenge 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 are releasing 30 datasets[3] pre-formatted in given feature representations (i.e., each example consists of a fixed number of numerical coefficients). Data present themselves as input-output pairs that are identically and independently distributed. The models used are limited to fixed-length vectorial representations. Hence, we do not incorporate problems of time series prediction. Text, speech, and video processing tasks included in the challenge are not presented in their native data representations; datasets have been preprocessed

1. http://codalab.org/AutoML
2. codalab.org
3. The nature of the data will remain confidential until the challenge is over.

in suitable fixed-length vectorial representations. The difficulty of the challenge 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, and available computational time, the methods and hyper-parameter settings that maximize performance. The participant's challenge is to create the *perfect black box* that removes human interaction.

2.1. Tasks

This challenge is concerned with regression and binary, multi-class, and multi-label classification problems from data formatted in fixed-length feature-vector representations. Each task is associated with a dataset coming from a real application. The domains of application are drawn from 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. All datasets present themselves in the form of data matrices with examples in rows and features in columns. The goal is to predict a target value. For instance, in a medical dataset, examples may represent patient records and features may represent results of laboratory analyses; the goal may be to predict the diagnosis of the patient (positive or negative). The identity and description of the datasets is concealed (except in round 0) to avoid the use of domain knowledge and to push participants to design fully automated ML solutions. In addition, the tasks are constrained by a time budget and a scoring metric.

2.2. Time Budget

The Codalab platform provides computational resources shared by all participants. To ensure fairness, when a code submission is evaluated, its execution time is limited to a given time budget, which varies from dataset to dataset. The time budget is provided with each dataset in its *info* file. The organizers reserve the right to adjust the time budget by supplying participants with updated *info* files. Participants who submit results—instead of code—are not constrained by the time budget since their code is run on their own platform. This may be advantageous for entries counting towards the Final phases (immediately following a Tweakathon). Participants wishing to also enter the AutoML phases, which require submitting code, can submit both results and code (simultaneously). The results do not need to be produced by the submitted code; if a participant does not want to share personal code, he/she can submit the sample code provided by the organizers together with his/her results.

2.3. Scoring Metrics

The score is 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). Participants must submit prediction values matching as closely as possible the target value, 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 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,

$$\langle y_i \rangle = (1/P) \sum_i (y_i). \tag{1}$$

The score metrics are described in the followings.

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

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

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

$$ABS = 1 - MAE/MAD, \tag{3}$$

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. 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.

$$|BAC| = (BAC - R)/(1 - R), \tag{4}$$

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

$$|AUC| = 2AUC - 1. \tag{5}$$

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

$$F1 = 2 * (precision * recall)/(precision + recall), \tag{6}$$

$$precision = true\ positive/(true\ positive + false\ positive) \tag{7}$$

$$recall = true\ positive/(true\ positive + false\ negative) \tag{8}$$

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

$$|F1| = (F1 - R)/(1 - R), \tag{9}$$

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

$$PAC = \exp(-CE), \tag{10}$$

$$CE = \begin{cases} average \sum_l \log(q_{il}), & \text{for multi-class} \\ -\langle y_i \log(q_i), & \\ +(1 - y_i)\log(1 - q_i)\rangle, & \text{for binary and multi-label} \end{cases} \tag{11}$$

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

$$|PAC| = (PAC - R)/(1 - R), \tag{12}$$

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.

2.4. Leaderboard Score Calculation

Participants (or their submitted code) provide prediction results for the withheld target values for all datasets in each round. The scoring program supplied by the organizers is run on the server to compute the scores. For each dataset, participants are ranked in decreasing order of performance—based on the scoring metric associated with the given task. The overall score is computed by averaging the rank obtained on every dataset and shown on the leaderboard in column $\langle rank \rangle$.

The results of the last submission are used to compute the leaderboard results. Therefore, participants must resubmit older entries if they want them to be counted as final. In phases marked with a [+], participants with the three smallest $\langle rank \rangle$ are eligible for prizes (subject to compliance with the Terms and Conditions).

Table 1: **Phases of round n.** 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

2.5. Estimation of Performance on Future Data

In the Performance Prediction Challenge (Guyon et al., 2006) participants had to both design models that generalized well on future data and predict their performance. This is a fundamental aspect of the AutoML field. However, the challenge does not address this issue to limit the complexity of design and evaluation.

Along with the learning curves as a function of time spent searching for the best hyper-model, we could draw learning curves as a function of the number of training examples. Such curves are useful in evaluating whether having more training examples significantly improves performance. This may be part of the post-challenge analysis.

2.6. Rounds and Phases

The challenge is run in multiple phases grouped in six rounds. Round 0 (Preparation) is a practice round with publicly available datasets which is followed by five rounds of progressive difficulty (Novice, Intermediate, Advanced, Expert, and Master). Except for rounds 0 and 5, all rounds include three phases that alternate AutoML and Tweakathons contests. These phases are described in Table 1.

Submissions are made in Tweakathon phases only. The results of the latest submission are shown on the leaderboard and such submission automatically migrates to the next phase. Submitting code makes it possible to participate in subsequent phases without new submissions. However, participating in previous rounds is not a prerequisite for entering new rounds. Prizes are awarded in phases marked with a [+] during which there is no submission.

2.7. Code vs. Result Submission

To participate in phase AutoML[n], code must be submitted in Tweakathon[n-1]. To participate in the Final[n], code or results must be submitted in Tweakathon[n]. If both code and (well-formatted) results are submitted, the results are used for scoring rather than re-running the code in Tweakathon[n] and Final[n]. The code is executed when results are unavailable or not well formatted. Thus, there is no disadvantage in submitting both results and code. When submitting results and code, different methods can be used to enter the Tweakathon/Final phases and the AutoML phases. Submissions are made only during Tweakathon with a maximum of five submissions per day. Immediate feedback is provided on the leaderboard on validation data. Participants rank on the basis of test performance during the Final and AutoML phases.

3. Data

Every round consists of five datasets spanning a range of difficulties. Data difficulty progressively increases from round to round. Datasets may be recycled, but are reformatted into new representations, except for the final round, which includes completely new data.

The datasets used in the challenge present the following range of difficulty, which requires demanding hyper-parameter choices:

Data distributions. Different intrinsic/geometrical complexity.

Tasks. Regression, binary classification, multi-class classification, and multi-label classification.

Scoring metrics. See Section 2.3.

Class imbalance. Balanced vs. unbalanced class proportions.

Sparsity. Full vs. sparse matrices.

Missing values. Presence vs. absence of missing values.

Categorical variables. Presence vs. absence of categorical features.

Irrelevant variables. Presence vs. absence of additional irrelevant data.

Number of training examples (P_{tr}). Small vs. large number of training examples.

Number of features (N). Small vs. large number of features.

Aspect ratio of the training data matrix (P_{tr}/N). $P_{tr} \gg N, P_{tr} \ = N$, or $P_{tr} \ll N$.

Round 0 uses five datasets from previous challenges to define tasks illustrating a few of the aforementioned difficulties. The details of the datasets are given in Guyon et al. (2015).

All datasets have been pre-formatted in a fixed-length feature-based representation. In the following rounds described below, the number of variables and samples vary between thousands and millions.

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 P_{tr}/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.

Table 2: Round 0 data: results of phase Tweakathon 0 on validation set. Rank is shown in parentheses for each data set. The best entry has the lowest average rank.

Participant	**<Rank>**	**DS 1**	**DS 2**	**DS 3**	**DS 4**	**DS 5**
ideal.intel	1.2	0.82 (2)	0.81 (1)	0.96 (1)	0.90 (1)	0.60 (1)
abhishek	3.2	0.82 (4)	0.79 (4)	0.94 (3)	0.85 (3)	0.45 (2)
aad.freiburg	3.4	0.82 (3)	0.80 (2)	0.94 (4)	0.80 (5)	0.42 (3)
reference	7.0	0.81 (8)	0.78 (5)	0.81 (8)	0.70 (8)	0.35 (6)

Table 3: Round 0 data: results of phase Final 0 on test set.

Participant	**<Rank>**	**DS 1**	**DS 2**	**DS 3**	**DS 4**	**DS 5**
ideal.intel.analytics	1.40	0.8262 (1)	0.8132 (2)	0.9632 (2)	0.8877 (1)	0.5894 (1)
abhishek	43.60	0.8178 (4)	0.7924 (4)	0.9394 (5)	0.8716 (2)	0.4608 (3)
aad.freiburg	4.00	0.8172 (6)	0.8107 (3)	0.9751 (1)	0.8580 (5)	0.3958 (5)
reference	18.40	0.8140 (8)	0.7759 (24)	0.8151 (13)	0.6178 (26)	0.3345 (21)

Table 4: Round 1 data: results of phase AutoML 1 on test set.

Participant	**<Rank>**	**DS 1**	**DS 2**	**DS 3**	**DS 4**	**DS 5**	**Duration**
aad.freiburg	2.80	0.5096 (1)	0.6059 (4)	0.6270 (3)	0.5802 (1)	0.8778 (5)	5988 (3)
jrl	443.80	0.4856 (2)	0.6276 (1)	0.5993 (5)	0.5292 (3)	0.8711 (8)	5987 (4)
tadej	4.20	0.4309 (9)	0.6207 (3)	0.7468 (1)	0.5549 (2)	0.8749 (6)	2728 (61)
reference	5.20	0.4568 (6)	0.5524 (8)	0.5324 (6)	0.5244 (4)	0.8934 (2)	4366 (20)

Table 5: Round 1 data: results of phase Tweakathon 1 on validation data.

Participant	**<Rank>**	**DS 1**	**DS 2**	**DS 3**	**DS 4**	**DS 5**
sjahandideh	2.40	0.5588 (1)	0.6958 (1)	0.8593 (1)	0.7067 (4)	0.9192 (5)
ideal.intel.analytics	2.60	0.5564 (2)	0.6844 (2)	0.8453 (5)	0.7376 (2)	0.9254 (2)
aad.freiburg	3.20	0.5276 (5)	0.6768 (3)	0.8553 (4)	0.8182 (1)	0.9251 (3)
reference	32.40	0.4628 (27)	0.5627 (29)	0.5276 (58)	0.5163 (31)	0.8895 (17)

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

4. Baseline Software and Preliminary Results

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 are used.

As of June 2015, more than 300 people registered and downloaded data, and more than 60 teams are actively participating. The top ranking participants significantly outperform the reference entry in rounds 0 and 1 (see Table 2). Results on the first few phases that are available as of early July are listed in Tables 3, 4, 5, and 6 respectively. The methods of several participating teams are described in Feurer et al. (2015), Stajner (2015), and Thakur and Krohn-Grimberghe (2015).

Table 6: Round 1 data: results of phase Final 1 on test set.

Participant	**<Rank>**	**DS 1**	**DS 2**	**DS 3**	**DS 4**	**DS 5**
aad.freiburg	2.20	0.5269 (4)	0.6378 (2)	0.8457 (2)	0.7925 (1)	0.9364 (2)
ideal.intel.analytics	3.20	0.5537 (1)	0.6458 (1)	0.8130 (8)	0.7153 (3)	0.9344 (3)
asml.intel.com	4.60	0.5441 (2)	0.6310 (3)	0.8191 (6)	0.6569 (7)	0.9280 (5)
reference	34.20	0.4722 (17)	0.5524 (33)	0.5324 (63)	0.5244 (37)	0.8934 (21)

5. Discussion

Based on results of past challenges, we designed a competition where the strategy is to (1) reduce the search space with filter methods, (2) reduce the number of hyper-parameters using versions of the algorithms that optimize them with embedded methods, and (3) use an ensemble method to grow an ever improving ensemble until the computational budget is exhausted.

A few participants asked for more computational resources. In round 0 we provided one hour of computing per submission on an 8-core machine; we will progressively ramp up this time up to 10 hours throughout the challenge. Future editions might run on Hadoop to allow participants to process larger datasets and face the Big Data era.

In the post-challenge analysis, we plan to systematically test the winning entries on these datasets semi-synthetically modified to cover a wider range of problem complexity (e.g., varying the proportion of training examples, missing data, and distractor variables). We intend to relate the results to a set of data complexity measures proposed in Ho and Basu (2002).

Acknowledgments

Microsoft supported the organization of this challenge and donated the prizes. 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. Computing resources were provided generously by J. Buhmann, ETH Zürich. The datasets released were selected among 72 datasets that were donated (or formatted using data publicly available) 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: M. Boullé, C. Capponi, R. Caruana, G. Dror, B. Kégl, H. Larochelle, V. Lemaire, C.-J. Lin, V. Ponce López, S. Mercer, F. Popescu, D. Silver, A. Statnikov, and I. Tsamardinos. The software developers who contributed to the implementation of the Codalab platform and the sample code include E. Camichael, I. Judson, C. Poulain, P. Liang, A. Pesah, X. Baro Solé, E. Watson, M. Zyskowski. This work has utilized computing resources at the High Performance Computing Facility of the Center for Health Informatics and Bioinformatics at the NYU Langone Medical Center.

References

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Methods for improving bayesian optimization for AutoML. In *AutoML Workshop, International Conference on Machine Learning 2015*, 2015.

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, 2006.

Isabelle Guyon, Kristin Bennett, Gavin Cawley, Hugo Jair Escalante, Sergio Escalera, Tin Kam Ho, Núria Macià, Bisakha Ray, Mehreen Saeed, Alexander Statnikov, and Evelyne Viegas. Design of the 2015 chalearn automl challenge. In *The International Joint Conference on Neural Networks*, July 2015.

Tin Kam Ho and Mitra Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, March 2002.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, October 2011.

Tadej Stajner. Autokit: Automatic machine learning via representation and model search. In *AutoML Workshop, International Conference on Machine Learning 2015*, 2015.

Abhishek Thakur and Artus Krohn-Grimberghe. Autocompete: A framework for machine learning competitions. In *AutoML Workshop, International Conference on Machine Learning 2015*, 2015.