AutoML #4

Auto-sklearn: Automating Design Decision in AutoML

Mustafa Cavus

Dec 13, 2021

Agenda

1. Introduction

- The goal of AutoML is to get high performance without requiring the user to make decisions on the ML pipeline.
- These decisions can be classified as low and high level.
  - Low level decisions: model and hyperparameter configurations.
  - High level decisions: choosing the CV or HO, and choosing the use of SH or not.
- PoSH Auto-sklearn suffer from that the users have to manually set the arguments on a per-dataset basis.
- Therefore, Auto-sklearn is extended with a policy selector to automatically choose an optimization policy given a dataset.

1. Introduction

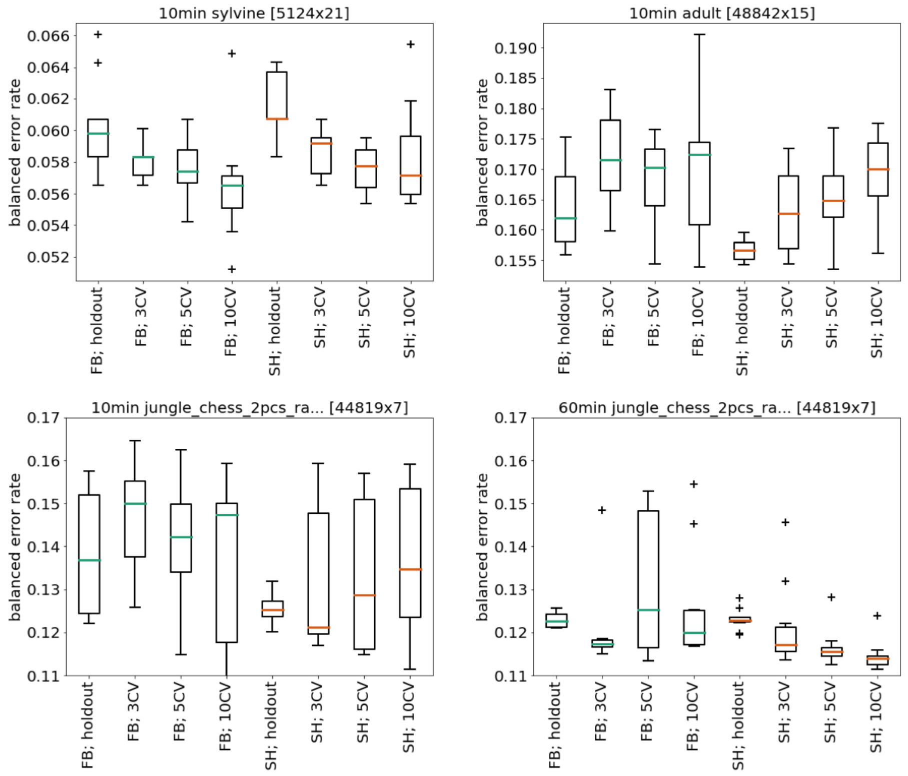


Figure 3: Final performance for BO using different model selection strategies averaged across 10 repetitions. Top row: Results for a optimization budget of 10 minutes on two different datasets. Bottom row: Results for a optimization budget of 10 and 60 minutes on the same dataset.

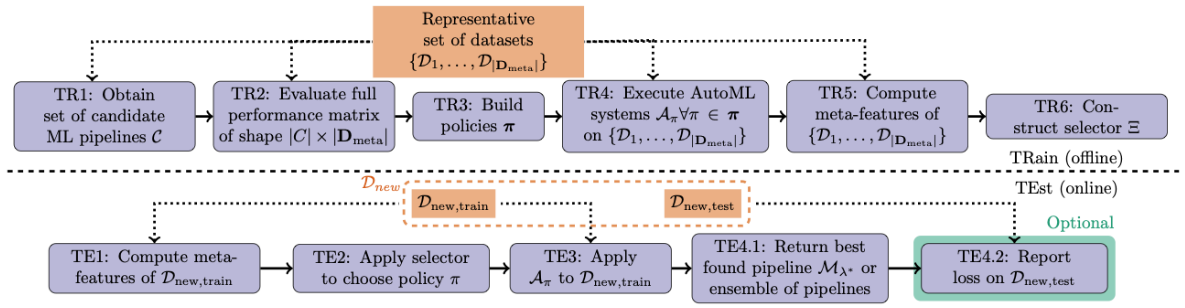
2. Automated Policy Selection

Consider there are different optimization policies $\pi \in \Pi$, the generalized error when an AutoML system A_π is not used with a fixed policy π, and a policy selector $\Xi : D \to \pi$:

$$\hat{GE}(A, \Xi, D_{meta}) = \frac{1}{|D_{meta}|} \sum_{d=1}^{|D_{meta}|} \hat{GE}(A_{\Xi(D_d)}(D_d), D_d)$$

2. Automated Policy Selection

A new layer on top of AutoML systems that automatically selects a policy π for a new dataset.



Representative set of datasets $\{\mathcal{D}_1, \dots, \mathcal{D}_{|\mathbf{D}_{meta}|}\}$
TR1: Obtain set of candidate ML pipelines $\mathcal{C}$
TR2: Evaluate full performance matrix of shape $|C| \times |\mathbf{D}_{meta}|$
TR3: Build policies $\boldsymbol{\pi}$
TR4: Execute AutoML systems $\mathcal{A}_\pi \forall \pi \in \boldsymbol{\pi}$ on $\{\mathcal{D}_1, \dots, \mathcal{D}_{|\mathbf{D}_{meta}|}\}$
TR5: Compute meta-features of $\{\mathcal{D}_1, \dots, \mathcal{D}_{|\mathbf{D}_{meta}|}\}$
TR6: Construct selector Ξ
TRain (offline)
TEst (online)
$\mathcal{D}_{new}$
$\mathcal{D}_{new,train}$
$\mathcal{D}_{new,test}$
Optional
TE1: Compute meta-features of $\mathcal{D}_{new,train}$
TE2: Apply selector to choose policy π
TE3: Apply $\mathcal{A}_\pi$ to $\mathcal{D}_{new,train}$
TE4.1: Return best found pipeline $\mathcal{M}_{\lambda^*}$ or ensemble of pipelines
TE4.2: Report loss on $\mathcal{D}_{new,test}$

Figure 4: Schematic overview of the proposed *Auto-sklearn 2.0* system with the training phase (TR1-TR6) above and the test phase (TE1-TE4) below the dashed line. Rounded boxes refer to computational steps while rectangular boxes depict the input data to the AutoML system.

There are eight different policies ({3-fold CV, 5-fold CV, 10-fold CV, holdout} × {SH, FB}).

2. Automated Policy Selection

Constructing the single best policy:

- A per-set algorithm selection can be used (Kerschke et al., 2019), which aims to find the single algorithm that performs best on average on set of problem instances;
- In this case it aims to find the combination of model selection and budget allocation that is best on average for the given set of meta-datasets.

Constructing the per-dataset policy selector:

- The policy selector design of HydraMIP (Xu et al., 2011) is followed.
- For each pair of AutoML policies, a random forest is fitted to predict whether policy π_A outperforms policy π_B given the current dataset's meta-features. Since the misclassification loss depends on the difference of the losses of the two policies (i.e. the ADTM when choosing the wrong policy), each meta-observation by their loss difference is weighted.
- To make errors comparable across different datasets (Bardenet et al., 2013), the individual error values are scaled for each dataset. At test time, all pairwise models are querried for the given meta-features, and use voting for Ξ to choose a policy π.

2. Automated Policy Selection

Meta-Features and Backup Strategy

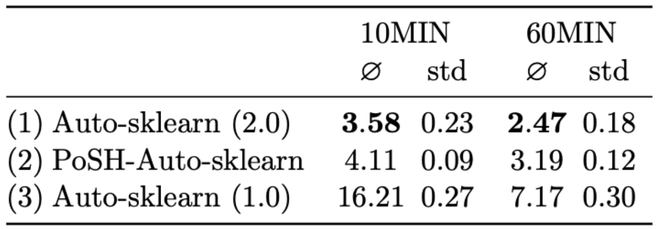
- A backup strategy is created by using meta-features in policy selection.
- Because of the run-time problem which is discussed in portfolio constructing phase, very simple and robust meta-features (number of data points and features) are used.
- If there is no dataset in the meta-datasets that has higher or equal values for each meta-feature, the system falls back to use holdout with SH.

3. Experimental Results

To compare the performance of the Auto-Sklearn 1.0, PoSh Auto-Sklearn and Auto-Sklearn 2.0, an experimental study is conducted with the following setup:

- Two opimization budgets are used: 10 min and 60 min.
- normalized balanced error rate (1 - balanced accuracy) is used to measure the performance.
- Two disjoint sets of datasets are used:
    1. D_{meta} is 208 datasets from OpenML with more than 500 and less than 1000000 samples with at least two attributes.
    2. D_{test} is 39 datasets selected for the AutoML benchmark proposed by Gijbers et al. (2019).

3. Experimental Results



	10MIN		60MIN	
	∅	std	∅	std
(1) Auto-sklearn (2.0)	**3.58**	0.23	**2.47**	0.18
(2) PoSH-Auto-sklearn	4.11	0.09	3.19	0.12
(3) Auto-sklearn (1.0)	16.21	0.27	7.17	0.30

Table 5: Final performance of *Auto-sklearn 2.0*, *PoSH Auto-sklearn* and *Auto-sklearn 1.0*. We report the normalized balanced error rate averaged across 10 repetitions on 39 datasets. We boldface the best mean value (per optimization budget) and underline results that are not statistically different according to a Wilcoxon-signed-rank Test ($\alpha = 0.05$).

- Auto-sklearn 2.0 achieves the lowest error for both optimization budgets.
- Auto-sklearn 2.0 reduces the relative error compared to Auto-sklearn 1.0 by 78% in 10 min budget.
- PoSH Auto-sklearn outperforms Auto-sklearn 1.0 in terms of the normalized balanced error rate, but that the additional step of selecting the model selection and budget allocation strategy gives Auto-sklearn 2.0 an edge.

3. Experimental Results

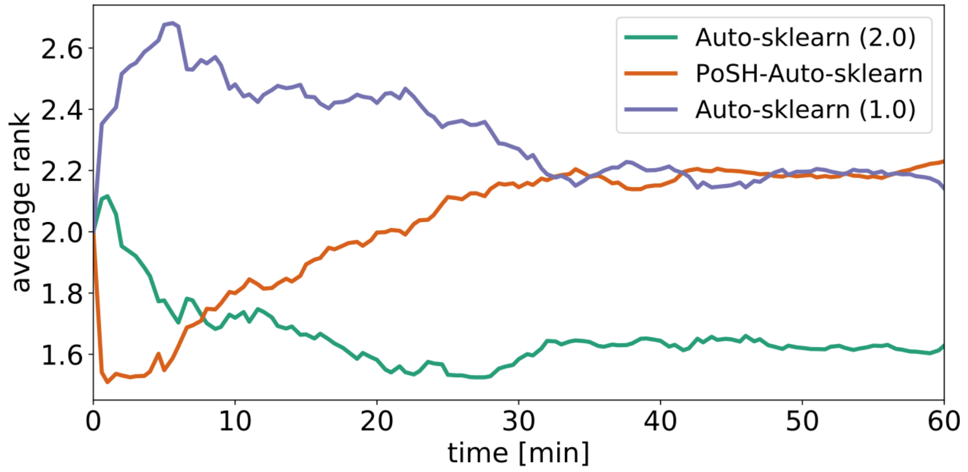


Figure 5: Performance over time. We report the normalized BER and the rank over time averaged across 10 repetitions and 39 datasets comparing our system to our previous AutoML systems.

- Average ranks (where failures obtain less weight compared to the averaged performance) are used the compared the performance of the versions.
- Auto-Sklearn 2.0 is still able to deliver best results.
- PoSH Auto-Sklearn should be preferred to Auto-Sklearn 1.0 for the first 30 min. and then converges to the roughly the same ranking.

4. Ablation Study

An ablation study is conducted to obtain the contribution of each improvements on Auto-Sklearn 1.0 by using the 39 datasets (Gijbers et al., 2019) which is used in the experimental study. In this study the answer of the following questions are explored:

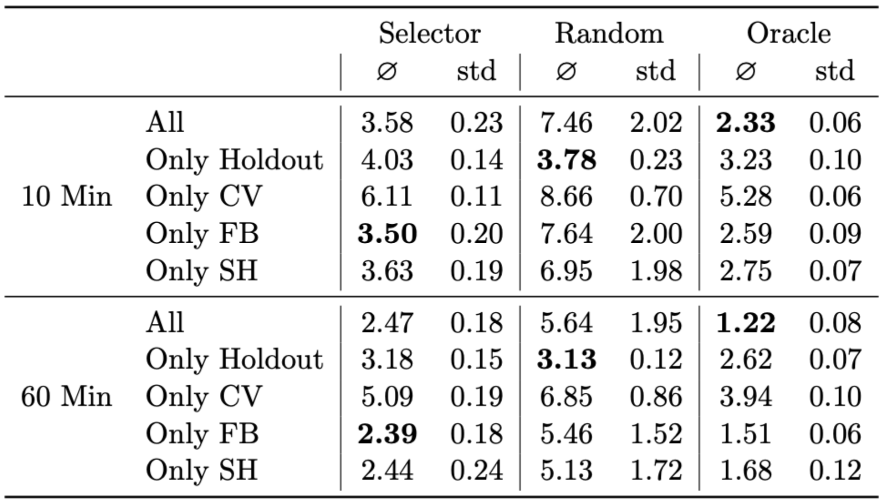
1. Is per-dataset selection needed?
2. Are different model selection strategies needed?
3. Is still warm-start bayesian optimization needed?

4.1. Is per-dataset selection needed?

- It is examined that how much performance is gain by having a model-based policy selector to decide between different AutoML strategies based on meta-features and how to construct this model-based policy selector, or whether it is sufficient to select a single strategy based on meta-training datasets.
- It is compared the performance of the full system using a model-based policy selector to using a single, static strategy (single best) and both, the model-based policy selector and the single best, without the fallback mechanism for out-of-distribution datasets.

4.2. Are different model selection strategies needed?

The different model selection strategies are compared. First model-based policy selectors on different subsets of the available eight combinations of model selection strategies and budget allocations: {3CV, 5CV, 10CV, holdout} x {SH, FB}.

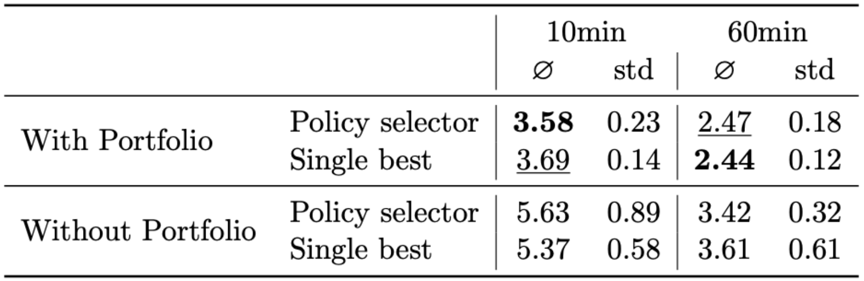


		Selector		Random		Oracle	
		∅	std	∅	std	∅	std
10 Min	All	3.58	0.23	7.46	2.02	**2.33**	0.06
	Only Holdout	4.03	0.14	**3.78**	0.23	3.23	0.10
	Only CV	6.11	0.11	8.66	0.70	5.28	0.06
	Only FB	**3.50**	0.20	7.64	2.00	2.59	0.09
	Only SH	3.63	0.19	6.95	1.98	2.75	0.07
60 Min	All	2.47	0.18	5.64	1.95	**1.22**	0.08
	Only Holdout	3.18	0.15	**3.13**	0.12	2.62	0.07
	Only CV	5.09	0.19	6.85	0.86	3.94	0.10
	Only FB	**2.39**	0.18	5.46	1.52	1.51	0.06
	Only SH	2.44	0.24	5.13	1.72	1.68	0.12

Table 7: Final performance (averaged normalized balanced error rate) for the full system and when not considering all model selection strategies.

4.3. Is still warm-start bayesian optimization needed?

- It is discussed the question whether is still need to add the additional complexity and invest resources to warm-start BO (and can therefore save the time to build the performance matrices to construct the portfolios).
- For this, the portfolio is removed from the AutoML system, meaning that it is directly started with BO and construct ensembles – both for creating the data the policy selector is trained on and for reporting performance.



		10min		60min	
		∅	std	∅	std
With Portfolio	Policy selector	**3.58**	0.23	2.47	0.18
	Single best	3.69	0.14	**2.44**	0.12
Without Portfolio	Policy selector	5.63	0.89	3.42	0.32
	Single best	5.37	0.58	3.61	0.61

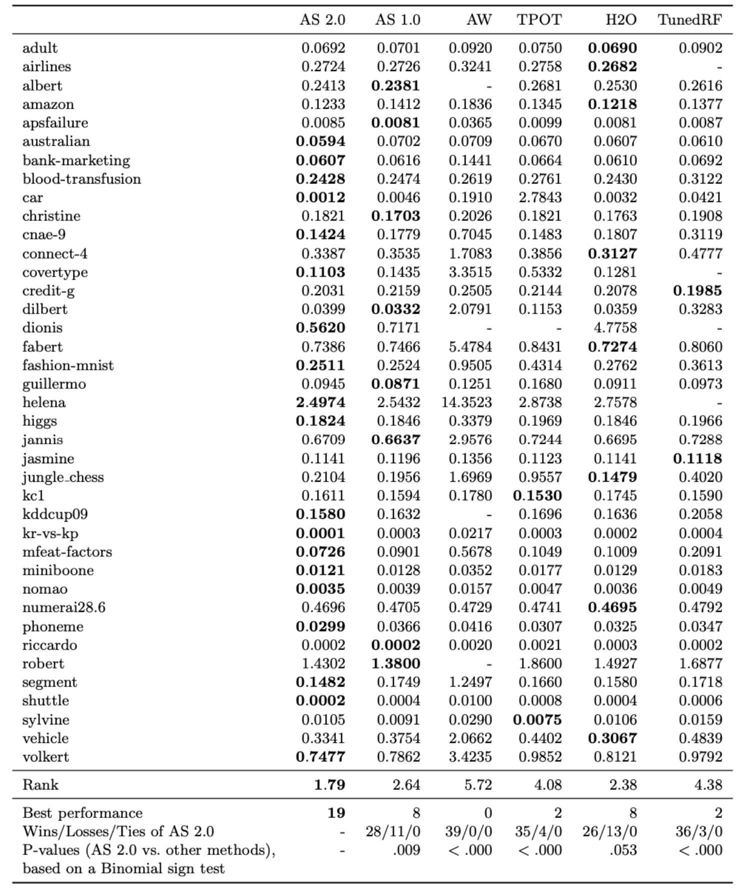
Table 8: Final performance (ADTM) after 10 and after 60 minutes with portfolios (top) and without (bottom). The row "with portfolio" and "policy selector" constitutes the full AutoML system including portfolios, BO and ensembles) and the row "without portfolios" and "policy selector" only removes the portfolios (both from the meta-data for model-based policy selector construction and at runtime). We boldface the best mean value (per optimization budget) and underline results that are not statistically different according to a Wilcoxon-signed-rank Test ($\alpha = 0.05$).

5. Comparison of AutoML Systems

The performance of Auto-sklearn 2.0, Auto-sklearn 1.0, Auto-WEKA (Thorthon et al., 2013), TPOT (Olsen et al., 2016), H2O AutoML (LeDell and Poirier, 2020), and random forest baseline hyperparameter tuning on 39 datasets (Gijbers et al., 2019).

- The log loss is used for multiclass datasets and $1 - \text{AUC}$ is used for binary datasets to measure the performance of the AutoML systems (lower is better).
- As an aggregate measure is the average rank is used (computed by averaging all folds and repetitions per dataset and then computing the rank).
- How often each framework is the winner on a dataset (champion), and give the losses, wins and ties against Auto-sklearn 2.0.
- The binomial sign test to compare the individual algorithms against Auto-sklearn 2.0.

5. Comparison of AutoML Systems



	AS 2.0	AS 1.0	AW	TPOT	H2O	TunedRF
adult	0.0692	0.0701	0.0920	0.0750	**0.0690**	0.0902
airlines	0.2724	0.2726	0.3241	0.2758	**0.2682**	-
albert	0.2413	**0.2381**	-	0.2681	0.2530	0.2616
amazon	0.1233	0.1412	0.1836	0.1345	**0.1218**	0.1377
apsfailure	0.0085	**0.0081**	0.0365	0.0099	0.0081	0.0087
australian	**0.0594**	0.0702	0.0709	0.0670	0.0607	0.0610
bank-marketing	**0.0607**	0.0616	0.1441	0.0664	0.0610	0.0692
blood-transfusion	**0.2428**	0.2474	0.2619	0.2761	0.2430	0.3122
car	**0.0012**	0.0046	0.1910	2.7843	0.0032	0.0421
christine	0.1821	**0.1703**	0.2026	0.1821	0.1763	0.1908
cnae-9	**0.1424**	0.1779	0.7045	0.1483	0.1807	0.3119
connect-4	0.3387	0.3535	1.7083	0.3856	**0.3127**	0.4777
covertype	**0.1103**	0.1435	3.3515	0.5332	0.1281	-
credit-g	0.2031	0.2159	0.2505	0.2144	0.2078	**0.1985**
dilbert	0.0399	**0.0332**	2.0791	0.1153	0.0359	0.3283
dionis	**0.5620**	0.7171	-	-	4.7758	-
fabert	0.7386	0.7466	5.4784	0.8431	**0.7274**	0.8060
fashion-mnist	**0.2511**	0.2524	0.9505	0.4314	0.2762	0.3613
guillermo	0.0945	**0.0871**	0.1251	0.1680	0.0911	0.0973
helena	**2.4974**	2.5432	14.3523	2.8738	2.7578	-
higgs	**0.1824**	0.1846	0.3379	0.1969	0.1846	0.1966
jannis	0.6709	**0.6637**	2.9576	0.7244	0.6695	0.7288
jasmine	0.1141	0.1196	0.1356	0.1123	0.1141	**0.1118**
jungle_chess	0.2104	0.1956	1.6969	0.9557	**0.1479**	0.4020
kc1	0.1611	0.1594	0.1780	**0.1530**	0.1745	0.1590
kddcup09	**0.1580**	0.1632	-	0.1696	0.1636	0.2058
kr-vs-kp	**0.0001**	0.0003	0.0217	0.0003	0.0002	0.0004
mfeat-factors	**0.0726**	0.0901	0.5678	0.1049	0.1009	0.2091
miniboone	**0.0121**	0.0128	0.0352	0.0177	0.0129	0.0183
nomao	**0.0035**	0.0039	0.0157	0.0047	0.0036	0.0049
numerai28.6	0.4696	0.4705	0.4729	0.4741	**0.4695**	0.4792
phoneme	**0.0299**	0.0366	0.0416	0.0307	0.0325	0.0347
riccardo	0.0002	**0.0002**	0.0020	0.0021	0.0003	0.0002
robert	1.4302	**1.3800**	-	1.8600	1.4927	1.6877
segment	**0.1482**	0.1749	1.2497	0.1660	0.1580	0.1718
shuttle	**0.0002**	0.0004	0.0100	0.0008	0.0004	0.0006
sylvine	0.0105	0.0091	0.0290	**0.0075**	0.0106	0.0159
vehicle	0.3341	0.3754	2.0662	0.4402	**0.3067**	0.4839
volkert	**0.7477**	0.7862	3.4235	0.9852	0.8121	0.9792
Rank	**1.79**	2.64	5.72	4.08	2.38	4.38
Best performance	**19**	8	0	2	8	2
Wins/Losses/Ties of AS 2.0	-	28/11/0	39/0/0	35/4/0	26/13/0	36/3/0
P-values (AS 2.0 vs. other methods), based on a Binomial sign test	-	.009	< .000	< .000	.053	< .000

Table 9: Results of the AutoML benchmark averaged across five repetitions. We report log loss for multiclass datasets and $1 - AUC$ for binary classification datasets (lower is better). AS is short for *Auto-sklearn* and AW for *Auto-WEKA*. *Auto-sklearn* has the best overall rank, the best performance in most datasets and, based on the P-values of a Binomial sign test, we gain further confidence in its strong performance.

5. Comparison of AutoML Systems

Results from Table 9:

- None of the AutoML systems is best on all datasets.
- Auto-sklearn 2.0 has the lowest average rank.
- Auto-sklearn 2.0 is followed by H2O AutoML and Auto-sklearn 1.0.
- According to both aggregate metrics, the TunedRF, Auto-WEKA and TPOT cannot keep up and lead to substantially worse results.
- Both versions of Auto-sklearn appear to be quite robust as they reliably provide results on all datasets, including the largest ones where several of the other methods fail.

Conclusions

- Portfolio construction is a useful way to reduce the optimization time instead of extracting meta-features for each of the new dataset.
- Successive Halving provides large speedups, but it could also too aggressively cut away good configurations that need a higher budget.
- Automated policy selection is useful improvement to increase the prediction performance of the AutoML system.
- One of the most important results for me, other than the improvements, is about the use of AutoML systems: Making it easy to use.

References

Feurer et al., Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning, 2021.

P. Gijsbers, E. LeDell, S. Poirier, J. Thomas, B. Bischl, and J. Vanschoren. An open source automl benchmark. In ICML Workshop on AutoML, 2019.

L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown. Hydra-MIP: Automated algorithm configuration and selection for mixed integer programming. In Proc. of RCRA workshop at IJCAI, 2011.

P. Kerschke, H. Hoos, F. Neumann, and H. Trautmann. Automated algorithm selection: Survey and perspectives. Evolutionary Computation, 27(1):3–45, 2019.

C. Thornton, F. Hutter, H. Hoos, and K. Leyton-Brown. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In Proc. of KDD'13, pages 847–855, 2013.

R. Olson and J. Moore. TPOT: A tree-based pipeline optimization tool for automat- ing machine learning. In Hutter et al. (2019).

E. LeDell and S. Poirier. H2o automl: Scalable automatic machine learning. In ICML W on AautoML, 2020.