AutoML #3

PoSH Auto-sklearn: Portfolio Successive Halving

Mustafa Cavus

Nov 29, 2021

Reference Paper

Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning

Matthias Feurer[1] FEURERM@CS.UNI-FREIBURG.DE
Katharina Eggensperger[1] EGGENSPK@CS.UNI-FREIBURG.DE
Stefan Falkner[2] STEFAN.FALKNER@DE.BOSCH.COM
Marius Lindauer[3] LINDAUER@TNT.UNI-HANNOVER.DE
Frank Hutter[1,2] FH@CS.UNI-FREIBURG.DE

[1] *Department of Computer Science, Albert-Ludwigs-Universität Freiburg*
[2] *Bosch Center for Artificial Intelligence, Renningen, Germany*
[3] *Institute of Information Processing, Leibniz University Hannover*

- Part I: Portfolio Successive Halving in PoSH Auto-sklearn
- Part II: Automating Design Decisions in AutoML

Agenda

1. Reminders
2. Portfolio Building
3. Budget Allocation: Successive Halving
4. Experimental Results

1. Reminder

In AutoML, generate pipeline $M_\lambda : x \rightarrow y$ automatically produces predictions minimizing the expected generalization error GE for data from the distribution P_d:

$$GE(M_\lambda) = E_{(x,y)\sim P_d}[L(M_\lambda(x), y)]$$

Since a dataset can only be observed through a set of n independent observations $D_d = \{(x_1, y_1), \ldots, (x_n, y_n)\} \sim P_d$, we can only emprically approximate the generalization error on sample data:

$$\hat{GE}(M_\lambda, D_d) = \frac{1}{n} \sum_{i=1}^{n} L(M_\lambda(\mathbf{x}_i), y_i)$$

In practice we have access to two disjoint, finite samples which we from now on denote D_{train} and D_{test}. For searching the best ML pipeline, we only have access to D_{train}, however, in the end performance is estimated once on D_{test}. AutoML systems use this to search for the best M_λ^*:

$$M_{\lambda^*} \in argmin_{\lambda \in} \hat{GE}(M_\lambda, D_{train})$$

where $\hat{GE}(M_\lambda, D_{train})$ is estimated by holdout or k-fold CV.

The Story Behind Portfolio Building

- Finding the optimal solution to the time-bounded optimization problem requires searching a large space of possible ML pipelines as efficiently as possible.

  $\mu_{\lambda^*} \in argmin_{\lambda \in \Lambda} \hat{GE}(\mu_\lambda, D_{train}) \quad s.t. (\sum t_{\lambda_i}) < T$

- BO is a strong approach for this, but it starts from scratch for every new problem.

- A better solution is to warmstart BO with ML pipelines that are expected to work well, as done in the k-nearest dataset (KND) approach of Auto-sklearn 1.0.

The Story Behind Portfolio Building

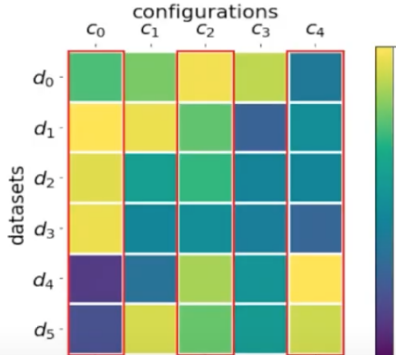
A better solution is to warmstart BO with ML pipelines that are expected to work well, as done in the k-nearest dataset (KND) approach has the following problems:

- It is time consuming since it requires to compute meta-features describing the characteristics of datasets.
- It adds complexity to the system as the computation of the meta-features must also be done with a time and memory limit.
- Many meta-features are not defined with respect to categorical features and missing values, making them hard to apply for most datasets.
- It is not immediately clear which meta-features work best for which problem.
- In the KND approach, there is no mechanism to guarantee that we do not execute redundant ML pipelines.

The Story Behind Portfolio Building

- They indicated suffered from these issues in the first AutoML challenge, failing on one track due to running over time for the meta-feature generation.
- Thus they removed landmarking meta-features due to their potentially high runtime.
- Propose a meta-feature-free approach which does not warmstart with a set of configurations specific to a new dataset.

2. Why Portfolio?



Goal: Meta-learning without meta-features

Idea: Construct a portfolio

Portfolio: A set of complementary configurations that covers as many diverse datasets as possible and minimize the risk of failure when facing a new task.

2.1. Portfolio Building Approach

For a given finite set of candidate pipelines $C = \{\lambda_1, \lambda_2, \ldots, \lambda_l\}$, assume that there exist a set of datasets $D_{meta} = \{D_1, D_2, \ldots, D_{|D_{meta}|}\}$, a portfolio P consisting a subset of the pipelines in C that performs well on D_{meta}, can be build using the following algorithm:

Algorithm 1: Greedy Portfolio Building

1: **Input:** Set of candidate ML pipelines $\mathcal{C}$, $\mathbf{D}_{\text{meta}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_{|\mathbf{D}_{\text{meta}}|}\}$, maximal portfolio size p, model selection strategy S
2: $\mathcal{P} = \emptyset$
3: **while** $|\mathcal{P}| < p$ **do**
4: $\quad \boldsymbol{\lambda}^* = \text{argmin}_{\boldsymbol{\lambda} \in \mathcal{C}} \widehat{GE}_S(\mathcal{P} \cup \{\boldsymbol{\lambda}\}, \mathbf{D}_{\text{meta}})$
`// Ties are broken favoring the model trained first.`
5: $\quad \mathcal{P} = \mathcal{P} \cup \boldsymbol{\lambda}^*, \ \mathcal{C} = \mathcal{C} \setminus \{\boldsymbol{\lambda}^*\}$
6: **end while**
7: **return** Portfolio $\mathcal{P}$

3. Budget Allocation

Two key components of any efficient AutoML system are:

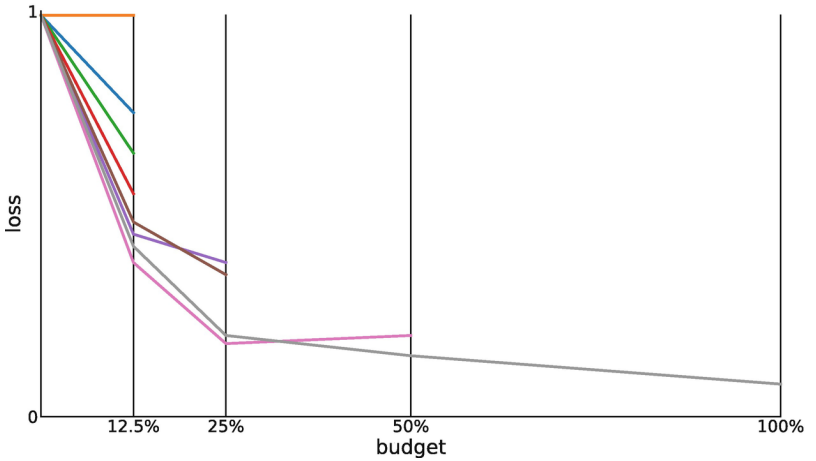
1. Model selection
2. Budget allocation strategies.

Which serve the following purposes:

- Approximate the generalization error of a single ML pipeline by using the holdout or k-fold CV.
- Decide how many resources to allocate for each pipeline evaluation.

3.1. Approach: Successive Halving

AutoML systems evaluate each pipeline under the same resource limitations and on the same budget. To increase efficiency, successive halving (SH) is used to allocate more resources to promising pipelines and prune poor-performing pipelines.



SH provides large speedups, but it could also too aggresively cut away good configurations that need a higher budget. Thus, SH to work best for large datasets, for which there is not enough time to train many ML pipelines for the full budget.

4. Experimental Study

In the experimental study, the improvements for PoSH Auto-sklearn are validated:

1. The performance of using a portfolio, previous KND approach and no warmstarting BO are compared.

2. The performance of the PoSH Auto-sklearn and Auto-sklearn 1.0. are compared.

4.1. Experimental Setup

- Two opimization budgets are used:
  - short - 10 min.
  - long - 60 min.
- balanced error rate (1 - balanced accuracy) is used
- linear transformation is used for datasets on different scales
- time (1/10 optimization budget) and memory (4 GB) are limited for each ML pipeline.

4.2. Datasets

Two disjoint sets of datasets are used:

1. D_{meta} for building portfolios (208 datasets from OpenML with more than 500 and less than 1000000 samples with at least two attributes.)
2. D_{test} for evaluating the method (39 datasets selected for the AutoML benchmark proposed by Gijbers et al., 2019).

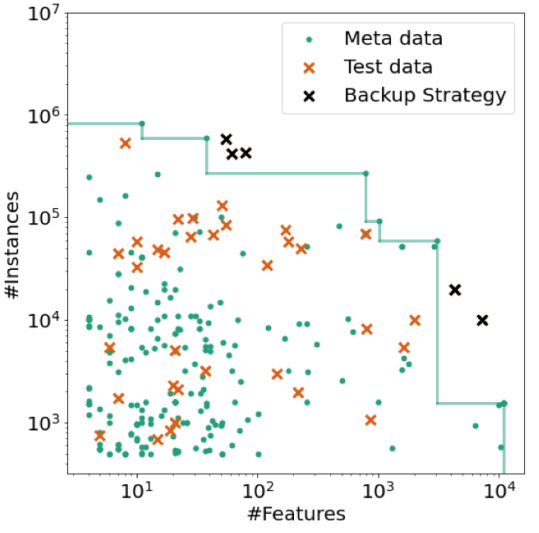


Figure 2: Distribution of meta and test datasets. We visualize each dataset w.r.t. its metafeatures and highlight the datasets that lie outside our meta distribution using black crosses.

4.3. Meta-Data Generation

For each optimization budget, four performance matrices are created. Each matrix refers to one way of assessing the generalization error of a model:

- holdout
- 3-fold CV
- 5-fold CV
- 10-fold CV

4.4. Results

4.4.1. Portfolio - KND - No warmstart BO

	10 minutes			60 minutes		
	BO	KND	Port	BO	KND	Port
holdout	5.98	5.29	**3.70**	3.84	3.98	**3.08**
SH; holdout	5.15	4.82	**4.11**	3.77	3.55	**3.19**
3CV	8.52	7.76	**6.90**	6.42	6.31	**4.96**
SH; 3CV	7.82	7.67	**6.16**	6.08	5.91	**5.17**
5CV	9.48	9.45	**7.93**	6.64	6.47	**5.05**
SH; 5CV	9.48	8.85	**7.05**	6.19	5.83	**5.40**
10CV	16.10	15.11	**12.42**	10.82	10.44	**9.68**
SH; 10CV	16.14	15.10	**12.61**	10.54	10.33	**9.23**

Table 2: Averaged normalized balanced error rate. We report the aggregated performance across 10 repetitions and 39 datasets of our AutoML system using only Bayesian optimization (*BO*), or BO warmstarted with k-nearest-datasets (*KND*) or a greedy portfolio (*Port*). Per line, we boldface the best mean value (per model selection and budget allocation strategy and optimization budget, and underline results that are not statistically different according to a Wilcoxon-signed-rank Test ($\alpha = 0.05$)).

4.4.2. PoSH Auto-sklearn vs Auto-sklearn 1.0

	10MIN		60MIN	
	∅	std	∅	std
(1) PoSH-Auto-sklearn	**4.11**	0.09	**3.19**	0.12
(2) Auto-sklearn (1.0)	16.21	0.27	<u>7.17</u>	0.30

Table 3: Final performance of *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 ($\alpha = 0.05$).

4.4.3. Raw results for the short budget

Task ID	Name	holdout	SH; holdout	3CV	SH; 3CV	5CV	SH; 5CV	10CV	SH; 10CV
167104	Australian	0.1721	0.1569	0.1622	0.1617	0.1583	0.1602	**0.1556**	0.1559
167184	blood-transfusion	0.3641	**0.3610**	0.3725	0.3666	0.3689	0.3722	0.3674	0.3689
167168	vehicle	0.2211	0.2267	0.2017	0.2093	0.2172	0.2052	0.2310	**0.1870**
167161	credit-g	0.2942	**0.2841**	0.2939	0.2955	0.2942	0.2911	0.2939	0.2934
167185	cnae-9	0.0658	0.0680	0.0651	0.0616	**0.0550**	0.0629	0.0626	0.0553
189905	car	0.0049	0.0049	0.0097	0.0029	0.0047	0.0017	0.0023	**0.0009**
167152	mfeat-factors	0.0152	0.0164	0.0141	**0.0107**	0.0150	0.0117	0.0153	0.0149
167181	kc1	0.2735	0.2688	0.2720	0.2713	0.2547	0.2660	**0.2477**	0.2719
189906	segment	0.0666	0.0687	0.0681	**0.0620**	0.0664	0.0621	0.0643	0.0671
189862	jasmine	0.2044	0.2051	**0.1982**	0.1986	0.2010	0.2027	0.2043	0.2027
167149	kr-vs-kp	**0.0067**	0.0077	0.0093	0.0085	0.0079	0.0078	0.0071	0.0080
189865	sylvine	0.0592	0.0594	0.0600	0.0608	0.0582	0.0582	**0.0560**	0.0578
167190	phoneme	0.1231	0.1245	0.1168	0.1160	0.1152	0.1136	**0.1129**	0.1144
189861	christine	0.2670	0.2621	0.2608	0.2556	**0.2517**	0.2567	0.2587	0.2645
189872	fabert	0.3387	0.3399	0.3140	0.3120	**0.3096**	0.3204	0.3180	0.3172
189871	dilbert	0.0241	0.0248	0.0258	0.0220	**0.0191**	0.0211	0.0303	0.0647
168794	robert	**0.5489**	0.5861	0.5762	0.5583	0.5854	0.5873	0.6230	0.6230
168797	riccardo	0.0035	0.0052	0.0067	0.0054	**0.0027**	**0.0027**	0.5000	0.5000
168796	guillermo	0.2186	**0.2102**	0.2311	0.2228	0.2165	0.2837	0.5000	0.5000
75097	Amazon	**0.2361**	0.2431	0.2526	0.2526	0.2379	0.2385	0.2448	0.2443
126026	nomao	0.0353	0.0381	0.0360	0.0345	**0.0312**	0.0331	0.0403	0.0401
189909	jungle_chess	0.1212	0.1251	0.1232	0.1156	0.1280	0.1180	**0.1134**	0.1141
126029	bank-marketing	0.1397	0.1436	0.1402	0.1407	**0.1352**	0.1435	0.1378	0.1362
126025	adult	0.1579	0.1575	0.1553	**0.1540**	0.1591	0.1562	0.1545	0.1585
75105	KDDCup09	0.2450	0.2495	**0.2449**	0.2512	0.2525	0.2487	0.2497	0.2456
168795	shuttle	0.0093	0.0086	0.0085	**0.0084**	0.0085	0.0087	0.0088	0.0084
168793	volkert	0.3735	0.3724	0.3939	**0.3703**	0.3720	0.3775	0.3867	0.3957
189874	helena	0.7483	0.7624	**0.7475**	0.7478	0.7483	0.7534	0.7476	0.7575
167201	connect-4	0.2629	0.2721	0.2653	0.2630	**0.2537**	0.2565	0.3003	0.2771
189908	Fashion-MNIST	**0.1050**	0.1098	0.1217	0.1195	0.1181	0.1153	0.1437	0.1437
189860	APSFailure	0.0384	0.0402	0.0355	0.0364	0.0355	**0.0354**	0.0410	0.0455
168792	jannis	0.3654	0.3685	0.3648	0.3655	0.3651	**0.3567**	0.3912	0.3881
167083	numerai28.6	0.4776	0.4765	0.4752	0.4749	**0.4747**	0.4775	0.4789	0.4788
167200	higgs	0.2736	0.2764	0.2730	0.2742	**0.2724**	0.2744	0.2832	0.2844
168798	MiniBooNE	**0.0581**	0.0589	0.0691	0.0633	0.0585	0.0644	0.0691	0.0685
189873	dionis	**0.1172**	0.1205	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
189866	albert	**0.3135**	0.3171	0.3469	0.3277	0.5000	0.3354	0.3714	0.3703
75127	airlines	0.3423	0.3424	0.3450	**0.3384**	0.3419	0.3429	0.3408	0.3456
75193	covertype	0.0568	0.0564	0.0683	0.0600	**0.0548**	0.0556	0.2519	0.2527

Table 13: Results from Table 2 for 10 minutes using portfolios. Numbers are the balanced error rate. We boldface the lowest error.

4.4.4. Raw results for the long budget

Task ID	Name	holdout	SH; holdout	3CV	SH; 3CV	5CV	SH; 5CV	10CV	SH; 10CV
167104	Australian	0.1742	0.1674	0.1623	0.1626	0.1598	0.1608	0.1625	**0.1557**
167184	blood-transfusion	0.3648	**0.3618**	0.3631	0.3641	0.3689	0.3692	0.3684	0.3692
167168	vehicle	0.2125	0.2344	0.1702	0.1944	**0.1657**	0.1960	0.1959	0.2151
167161	credit-g	0.2922	**0.2895**	0.3035	0.2957	0.3056	0.2978	0.3008	0.2931
167185	cnae-9	0.0733	0.0761	0.0560	0.0616	0.0537	0.0536	0.0675	**0.0518**
189905	car	0.0036	0.0013	0.0037	0.0098	**0.0007**	0.0012	0.0008	0.0010
167152	mfeat-factors	0.0169	0.0186	0.0130	**0.0117**	0.0139	0.0132	0.0151	0.0122
167181	kc1	0.2728	0.2739	0.2680	0.2724	0.2678	0.2804	**0.2546**	0.2576
189906	segment	0.0708	0.0692	0.0647	0.0635	**0.0588**	0.0596	0.0621	0.0613
189862	jasmine	0.2048	0.2049	0.1995	0.1989	0.1995	**0.1976**	0.1995	0.1980
167149	kr-vs-kp	0.0060	0.0080	0.0081	0.0068	0.0064	0.0068	0.0055	**0.0053**
189865	sylvine	0.0590	0.0591	0.0584	0.0587	0.0577	0.0578	**0.0573**	0.0573
167190	phoneme	0.1222	0.1237	0.1152	0.1155	0.1111	0.1130	0.1117	**0.1105**
189861	christine	0.2673	0.2666	0.2575	0.2584	**0.2532**	0.2575	0.2549	0.2588
189872	fabert	0.3381	0.3319	0.3099	0.3097	0.3119	0.3080	**0.3027**	0.3071
189871	dilbert	0.0200	0.0200	**0.0132**	0.0146	0.0185	0.0209	0.0212	0.0149
168794	robert	0.5273	0.5199	0.5238	**0.5183**	0.5456	0.5605	0.5652	0.5407
168797	riccardo	0.0029	**0.0016**	0.0018	0.0019	0.0025	0.0076	0.5000	0.5000
168796	guillermo	**0.2012**	0.2025	0.2057	0.2081	0.2100	0.2039	0.5000	0.5000
75097	Amazon	0.2376	0.2394	0.2338	0.2381	0.2431	0.2384	**0.2312**	0.2324
126026	nomao	0.0352	0.0353	0.0334	0.0331	0.0320	0.0327	**0.0313**	0.0319
189909	jungle_chess	0.1214	0.1221	0.1154	0.1172	0.1171	0.1153	**0.1108**	0.1141
126029	bank-marketing	0.1388	0.1398	0.1380	0.1392	0.1382	0.1382	**0.1370**	0.1380
126025	adult	0.1546	0.1541	0.1550	0.1540	0.1550	0.1550	0.1539	**0.1538**
75105	KDDCup09	0.2492	**0.2461**	0.2477	0.2532	0.2466	0.2488	0.2617	0.2485
168795	shuttle	0.0136	0.0107	0.0125	0.0093	0.0084	**0.0063**	0.0127	0.0087
168793	volkert	0.3600	0.3673	**0.3449**	0.3551	0.3496	0.3487	0.3581	0.3563
189874	helena	0.7449	0.7494	**0.7331**	0.7369	0.7407	0.7404	0.7562	0.7452
167201	connect-4	0.2539	0.2556	0.2382	0.2428	0.2370	0.2373	0.2416	**0.2369**
189908	Fashion-MNIST	0.1010	**0.0971**	0.1046	0.1066	0.1102	0.1105	0.1191	0.1075
189860	APSFailure	0.0362	0.0374	0.0345	0.0364	0.0372	0.0347	0.0343	**0.0334**
168792	jannis	0.3670	0.3638	0.3589	0.3576	0.3584	0.3565	**0.3473**	0.3572
167083	numerai28.6	0.4765	0.4763	0.4774	0.4770	0.4750	0.4767	0.4755	**0.4743**
167200	higgs	0.2712	0.2734	0.2718	**0.2680**	0.2696	0.2680	0.2701	0.2683
168798	MiniBooNE	0.0576	0.0583	0.0560	**0.0536**	0.0571	0.0565	0.0560	0.0608
189873	dionis	**0.0961**	0.1068	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
189866	albert	0.3116	0.3168	0.3170	0.3172	**0.3094**	0.3199	0.3183	0.3186
75127	airlines	0.3403	0.3410	**0.3375**	0.3401	0.3388	0.3390	0.3398	0.3399
75193	covertype	0.0537	0.0519	0.0496	0.0496	**0.0454**	0.0461	0.0458	0.0459

Table 14: Results from Table 2 for 60 minutes using portfolios. Numbers are the balanced error rate. We boldface the lowest error.

See you on Dec 13

Part II: Automating Design Decisions in AutoML