A Comparison of AutoML Tools for Machine Learning, Deep Learning and XGBoost

Luís Ferreira
EPMQ - IT
CCG ZGDV Institute
ALGORITMI Center
University of Minho
Guimarães, Portugal
luis.ferreira@ccg.pt

André Pilastri
EPMQ - IT
CCG ZGDV Institute
Guimarães, Portugal
andre.pilastri@ccg.pt

Carlos Manuel Martins
WeDo Technologies
Braga, Portugal
carlos.mmartins
@mobileum.com

Pedro Miguel Pires
WeDo Technologies
Braga, Portugal
pedro.mpires
@mobileum.com

Paulo Cortez
ALGORITMI Center
Dep. Information Systems
University of Minho
Guimarães, Portugal
pcortez@dsi.uminho.pt

***Abstract*—This paper presents a benchmark of supervised Automated Machine Learning (AutoML) tools. Firstly, we analyze the characteristics of eight recent open-source AutoML tools (Auto-Keras, Auto-PyTorch, Auto-Sklearn, AutoGluon, H2O AutoML, rminer, TPOT and TransmogrifAI) and describe twelve popular OpenML datasets that were used in the benchmark (divided into regression, binary and multi-class classification tasks). Then, we perform a comparison study with hundreds of computational experiments based on three scenarios: General Machine Learning (GML), Deep Learning (DL) and XGBoost (XGB). To select the best tool, we used a lexicographic approach, considering first the average prediction score for each task and then the computational effort. The best predictive results were achieved for GML, which were further compared with the best OpenML public results. Overall, the best GML AutoML tools obtained competitive results, outperforming the best OpenML models in five datasets. These results confirm the potential of the general-purpose AutoML tools to fully automate the Machine Learning (ML) algorithm selection and tuning.**

***Index Terms*—Automated Deep Learning (AutoDL), Automated Machine Learning (AutoML), Benchmarking, Neural Architecture Search (NAS), Software, Supervised Learning.**

I. Introduction

A Machine Learning (ML) application includes typically several steps: data preparation, feature engineering, algorithm selection and hyperparameter tuning. Most of these steps require trial and error approaches, especially for non-ML-experts. More experienced practitioners often use heuristics to exploit the vast dimensional space of parameters [1]. With the increasing number of non-specialists working with ML [2], in the last years there has been an attempt to automate several components of the ML workflow, giving rise to the concept of Automated Machine Learning (AutoML) [3].

This paper focuses on the selection of the best supervised ML algorithm and its hyperparameter tuning. The comparison study considers eight recent open-source AutoML technologies: Auto-Keras, Auto-PyTorch, Auto-Sklearn, AutoGluon, H2O AutoML, rminer, TPOT, and TransmogrifAI. To assess these tools, we use twelve popular datasets retrieved from the OpenML platform, divided into regression, binary and multi-class classification tasks. In particular, we design three main scenarios for the benchmark study: General ML (GML) algorithm selection; Deep Learning (DL) selection and XGBoost (XGB) hyperparameter tuning. Each tool is measured in terms of its predictive performance (using an external 10-fold cross-validation) and computational cost (measured in terms of time elapsed). Moreover, the best AutoML tools are further compared with the best public OpenML predictive results (which are assumed as the "gold standard").

The paper is organized as follows. Section 2 presents the related work. Next, Section 3 describes the AutoML tools and datasets. Section 4 details the benchmark design. Then, Section 5 presents the obtained results. Finally, Section 6 discusses the main conclusions.

II. Related Work

The state-of-the-art works that compare AutoML tools can be grouped into three major categories. The first category includes publications that introduce a novel AutoML tool and then compared it with existing ones. The second category (similar to our work) is related with comparison of distinct tools, without proposing a new AutoML framework. Finally, the third category (less approached) focuses on the characteristics of the technologies rather than their predictive performances.

Table I summarizes the related works using the following columns: **Ref.** – the study reference; **Cat.** – the AutoML study category; **Dat.** – the number of analyzed datasets; **Tools** – the number of compared AutoML tools; **GML** – if General ML algorithms (not DL) were tested, such as Naïve Bayes (NB), Support Vector Machine (SVM) or XGB; **DL** – if DL was included in the comparison; **Ext.** – the external validation method used (if any); **C.** – if computational effort was measured; and **Description** – brief explanation of the comparison approach. The majority of the related works (14 studies) are from the year 2020, which confirms that AutoML tool comparison is a hot research topic. Some studies explore a large number of datasets [4], [5]. Our comparison adopts 12 datasets, which is below the two mentioned works but is still higher than used in eleven other studies (e.g., [6], [7]). More importantly, we consider eight AutoML technologies, which is a number only outperformed by [8] (which tested only one dataset) and [9] (which did not use any datasets).

In particular, we benchmark the following recent tools: Auto-PyTorch - only studied in [10] and compared in [9]; rminer – not considered by the related works; and Transmogrifai – only compared in [11]. Most works target GML. There are four studies that only address DL (e.g., [6], [12]). Similar to our approach, there are seven studies that consider both GML and DL. Of the 21 surveyed works, only 12 employ an external validation. Most of these studies (8 of 12) use a single holdout train test split, which is less robust than a 10-fold cross-validation (adopted in four works). In addition, only 9 studies measure the computational effort. Furthermore, few studies contrast the AutoML results with the best human configured results. Kaggle competition results were included in [6], [13], [14]. This work adopts open science (OpenML) best results, which was only performed in [15].

TABLE I
SUMMARY OF THE RELATED WORK (AUTOML TOOL COMPARISON).

Year	Ref.	Cat.	Dat.	Tools	GML	DL	Ext.	C.	Description
2019	[16]	1	8	5	✓				new AutoML tool
2019	[6]	1	2	4		✓	HO	✓	new AutoML tool
2019	[17]	1	53	4	✓	✓	10CV	✓	new AutoML tool
2019	[18]	2	39	4	✓	✓			AutoML benchmark
2019	[19]	2	5	3	✓	✓			AutoML benchmark
2019	[3]	2	n.d.	n.d.	✓	✓	HO	✓	AutoML competition
2019	[4]	2	300	6	✓	✓	HO		AutoML benchmark
2020	[5]	1	175	2	✓		HO	✓	new AutoML tool
2020	[7]	1	3	2	✓				new AutoML tool
2020	[13]	1	50	6	✓		10CV	✓	new AutoML tool
2020	[20]	1	39	2	✓		10CV	✓	new AutoML tool
2020	[21]	1	5	2		✓	HO	✓	new AutoML tool
2020	[12]	1	3	2		✓	HO		new AutoML tool
2020	[22]	1	130	3	✓			✓	new AutoML tool
2020	[10]	1	8	4	✓	✓			new AutoML tool
2020	[15]	2	12	4	✓				AutoML benchmark
2020	[11]	2	3	2	✓		HO	✓	AutoML benchmark (risk management)
2020	[14]	2	137	5	✓		10CV		survey and benchmark
2020	[8]	3	1	12		✓	HO		literature review
2020	[23]	3	0	7	✓				qualitative comparison
2020	[9]	3	0	18	✓	✓			qualitative comparison
this work	-	2	12	8	✓	✓	10CV	✓	benchmark

n.d. - not disclosed.
10CV - 10-fold Cross-Validation (CV).
HO - Hold-Out (HO) validation.

III. MATERIALS AND METHODS

A. AutoML Tools

This study compares eight recent open-source AutoML tools. Whenever possible, all tools were executed with their default values, in order to prevent any bias towards a particular tool, while also corresponding to a natural non-ML-expert choice. When available in the tool documentation, we show the number of hyperparameters ($\mathcal{H}$) tuned by the AutoML.

1) Auto-Keras: a Python library based on the Keras module and that is focused on an automatic DL Neural Architecture Search (NAS) [24]. The search is performed by using a Bayesian Optimization, with the tool automatically tuning the number of dense layers, units, type of activation functions used, dropout values and other DL hyperparameters. In this work, we adopt Auto-Keras version 1.0.7, which is used in the DL scenario (Section IV).

2) Auto-PyTorch: another AutoML tool specifically focused on NAS [10]. Auto-PyTorch version 0.0.2 uses the PyTorch framework and a multi-fidelity optimization to search the parameters of the best architecture (e.g., network type, number of layers, activation function). Similarly to Auto-Keras, we use Auto-PyTorch only in the second DL scenario.

3) Auto-Sklearn: an AutoML library built on top of the Scikit-Learn ML framework. The choice of algorithms and hyperparameters implemented by Auto-Sklearn takes advantage of recent advances in Bayesian optimization, meta-learning, and Ensemble Learning [25]. We use Auto-SkLearn version 0.7.0 in the first GML scenario, since it does not implement an automated DL or XGB. All ML algorithms (when available for the task type) were tested: AdaBoost ($\mathcal{H} = 4$), Bernoulli ($\mathcal{H} = 2$) and Multinomial NB ($\mathcal{H} = 2$), Gaussian NB ($\mathcal{H} = 0$), Decision Tree (DT) ($\mathcal{H} = 4$), Extremely Randomized Trees (XRT) ($\mathcal{H} = 5$), Gradient Boosting Machine (GBM) ($\mathcal{H} = 6$), k-Nearest Neighbors (k-NN) ($\mathcal{H} = 3$), Linear Discriminant Analysis (LDA) ($\mathcal{H} = 4$), Linear SVM (LSVM) ($\mathcal{H} = 4$), Kernel based SVM (KSVM) ($\mathcal{H} = 7$), Passive Aggressive ($\mathcal{H} = 3$), Quadratic Discriminant Analysis (QDA) ($\mathcal{H} = 2$), Random Forest (RF) ($\mathcal{H} = 5$) and a Multiple Linear Regression (MR) classifier ($\mathcal{H} = 10$).

4) AutoGluon: a Python AutoML toolkit focused on DL [26]. In this work, we consider the tabular prediction feature of AutoGluon version 0.0.13. The tabular prediction executes several ML algorithms and then returns a Stacked Ensemble that uses the distinct ML models in multiple layers. In the GML scenario (Section IV), ensemble includes all non DL algorithms: GBM, CatBoost Boosted Trees, RF, Extra Trees (XT), k-NN and MR. For the DL scenario, the AutoGluon uses a DL dense architecture that uses heuristics to set the hidden layer sizes, employing also ReLU activation functions, dropout regularization and batch normalization layers [26].

5) H2O AutoML: the H2O open-source module for AutoML [27]. The tool adopts a grid search to perform the ML model selection. In this paper, we use H2O AutoML version 3.30.1.2 for the three comparison scenarios: GML, DL and XGB. In GML, all ML algorithms were explored (except DL): Generalized Linear Model (GLM) ($\mathcal{H} = 1$), GBM ($\mathcal{H} = 8$), RF ($\mathcal{H} = 0$), XRT ($\mathcal{H} = 0$), XGB ($\mathcal{H} = 9$) and two Stacked Ensembles: Best – with only the best models per ML family; and All – with all trained algorithms. For the DL scenario, the H2O tool uses a fully connected multi-layer perceptron trained with a stochastic gradient descent back-propagation algorithm. The searched $\mathcal{H} = 7$ hyperparameters include the number of hidden layers and hidden units per layer, the learning rate, the number of training epochs, activation functions and input and hidden layer dropout values. Finally, for the XGB scenario, the tool tunes the same $\mathcal{H} = 9$ hyperparameters of GML.

6) rminer: package of the R tool that is intending to facilitate the use of ML algorithms [28]. In its most recent version (1.4.6), rminer implements AutoML functions. The

rminer AutoML executions can be completely customized by the user, who can define the searched ML algorithms, hyperparameter ranges and validation metrics of the assumed grid search. For less experienced users, rminer includes three predefined AutoML search templates (https://CRAN.R-project.org/package=rminer). Similarly to H2O, we test this tool in the GML and XGB scenarios. In GML, we used the `"automl3"` template, which searches the best model among: GLM ($\mathcal{H} = 2$), Gaussian kernel SVM ($\mathcal{H} = 2$ for classification and $\mathcal{H} = 3$ for regression), shallow multilayer perceptron (with one hidden layer, $\mathcal{H} = 1$), RF ($\mathcal{H} = 1$), XGB ($\mathcal{H} = 1$) and a Stacked Ensemble ($\mathcal{H} = 2$, similar to H2O Stacked Best).

7) TPOT: a tool written in Python and that automates several ML phases (e.g., feature selection, algorithm selection) by using a Genetic Programming [29]. The GML scenario tested all TPOT version 0.11.5 algorithms: DT, RF, XGB, (multinomial) Logistic Regression (LR) and k-NN. TPOT was not included in the third comparison scenario (XGB, Section IV) because the tool does not allow the selection of a single algorithm, such as XGB.

8) TransmogrifAI: an AutoML tool for structured data and that runs on top of Apache Spark [30]. TransmogrifAI version 0.7.0 uses a grid search to perform the search of the best ML model. In the GML scenario, the tool was tested with all its ML algorithms: NB, DT, Gradient Boosted Trees (GBT), RF, MR, LR and LSVM.

9) Summary: Table II summarizes the AutoML tools that were used. For each tool, we detail the base ML **Framework**, available Application Programming Interface (**API**) programming **Language**, compatible **Operating Systems**, and if it supports **DL** (Auto-Keras and Auto-PyTorch only address DL).

TABLE II
DESCRIPTION OF THE COMPARED AUTOML TOOLS.

AutoML Tool	Framework	API Lang.	Operating Systems	DL	Scenario GML	DL	XGB
Auto-Keras	Keras	Python	MacOs Linux Windows	Yes (only)		✓	
Auto-PyTorch	PyTorch	Python	MacOs Linux Windows	Yes (only)		✓	
Auto-Sklearn	Scikit-Learn	Python	Linux	No	✓		
AutoGluon	PyTorch	Python	MacOS (P.) Linux	Yes	✓	✓	
H2O AutoML	H2O	Java Python R	MacOs Linux Windows (P.)	Yes	✓	✓	✓
rminer AutoML	rminer	R	MacOs Linux Windows	No	✓		✓
TPOT	Scikit-Learn	Python	MacOs Linux Windows	No	✓		
TransmogrifAI	Spark (MLlib)	Scala	MacOs Linux Windows	No	✓		

P. - partially supported (with less capabilities).

TABLE III
DESCRIPTION OF THE SELECTED OPENML DATASETS.

Dataset	Task	Rows	Cols.	Classes/ levels	Target values
Cholesterol	regression	303	14	152	[126, 564]
Churn	binary	5000	21	2	{0,1}
Cloud	regression	108	7	94	[0, 6]
Cmc	multi-class	1473	10	10	{0,1,...,9}
Credit	binary	1000	21	2	{0,1}
Diabetes	binary	768	9	2	{0,1}
Dmft	multi-class	797	5	6	{0,1,...,5}
Liver disorders	regression	345	6	16	[0, 20]
Mfeat	multi-class	2000	7	10	{0,1,...,9}
Plasma	regression	315	14	257	[179, 1727]
Qsar	binary	1055	42	2	{0,1}
Vehicle	multi-class	846	19	4	{0,1,...,3}

B. Data

The analyzed datasets (Table III) were retrieved from OpenML [31]. The data selection criterion was defined as selecting the most downloaded datasets that did not include missing data and that reflected three supervised learning tasks: regression, binary and multi-class classification. The datasets reflect different numbers of instances (**Rows**), input variables (**Cols.**) and output target response values (**Classes/levels**, from 2 to 257; the last column details the **Target** domain values).

IV. BENCHMARK DESIGN

The comparison study assumes three main scenarios (Table II). The first **GML** scenario executes all ML algorithms from the AutoML tools except DL, aiming to perform a more horizontal ML family agnostic search. DL was discarded since some of the tools do not implement DL (Table II), the training of DL models often requires a higher computational effort and the second scenario is exclusively devoted to DL. The second **DL** scenario focuses on NAS, as implemented by the Auto-Keras, Auto-PyTorch, AutoGluon and H2O AutoML tools. Finally, the third scenario is more vertical, considering only the **XGB** algorithm. XGB was selected since it is a recently proposed non DL algorithm that includes a large number of hyperparameters (e.g., H2O documentation mentions 40 hyperparameters of which only $\mathcal{H} = 9$ are tunned). In this scenario, we test H2O and rminer, since they are AutoML tools that allow to run the single XGB algorithm.

For every predictive experiment, the datasets were equally divided into tens folds, used for the external cross-validation. In order to create validation sets (to select the best ML algorithms and hyperparameters), we adopted an internal 5-fold validation. For instance, if the data contains 100 instances, then in the first external 10-fold iteration 90 examples are used by the tool for fitting purposes (model selection and training), with the remaining 10 instances being used for the external testing. The 90 fit examples are further divided into 5 folds.

In the first internal fold, each ML is trained with 72 instances and 18 are used for validation purposes (allowing to select the best model). Since neither Auto-Keras nor Auto-PyTorch natively support cross-validation during the fitting phase, we used a simpler holdout train (75%) and test (25%) set split to select and fit the models.

In all three scenarios the same measures are used to evaluate the performance of the external 10-fold test set predictions. Popular prediction measures were selected: regression - Mean Absolute Error (MAE) ($\in [0.0,\infty[$, where 0.0 denotes a perfect predictor); binary classification - Area Under the receiver operating characteristic Curve (AUC) ($\in$ [0.0,1.0], where 1.0 denotes the ideal classifier); multi-class classification - Macro F1-score ($\in$ [0.0,1.0], where 1.0 denotes the perfect model). Whenever allowed by the AutoML tool, we adopted the same measures for the internal AutoML validation set model comparison. The exceptions were with multi-class datasets and the Auto-Keras and Auto-PyTorch tools, which did not allow to use a Macro F1-score validation, thus the default loss function was adopted for these tools.

All experiments were executed using an Intel Xeon 1.70GHz server with 56 cores and 2TB of disk space. For each external fold, we also recorded the computational effort (in terms of time elapsed) for the AutoML fit (model selection and training). When the AutoML tool allowed to specify a time limit for training, the chosen time was one hour (3,600 s). Also, for the tools that implement an early stopping AutoML parameter, we fixed the value to three rounds. To aggregate the distinct external 10-fold results, we compute the average values. We also provide the 10-fold average t-distribution 95% confidence intervals, which can be used to attest if the tool differences are statistically significant (e.g., by checking if two confidence intervals do not overlap). Nevertheless, given that there is a very large number of comparisons, to select the best tool for each task, we adopt a lexicographic approach [32], which considers first the best average predictive performance (with a precision up to 1% or 0.01 points) and then the average computational effort (precision in s). To facilitate the lexicographic regression analysis, we compute the Normalized MAE (NMAE) score, which is a scale independent measure, where $NMAE = MAE/(\max(y) - \min(y))$ and y denotes the output target.

V. Results

Figures 1 and 2 summarize the main scenario (GML) results. In total, there were 12 (dataset) $\times$ 6 (tools) $\times$ 10 (folds) = 720 AutoML executions. Figure 1 presents the average computational effort (in s) for each external 10-fold iteration. Figure 2 shows the average external test scores (grouped in terms of the binary, multi-class and regression tasks). To facilitate the visualization of the regression scores, in the right of Figure 2 we use the NMAE score in the y-axis.

For GML, Auto-Sklearn always requires the maximum allowed computational effort (3,600 s), followed by TPOT (average of 858 s per external fold and dataset). The other tools are much faster: AutoGluon – lowest average value (70 s), best in 5 datasets; H2O – second average value (158 s), best in 5 datasets; TransmogrifAI – third best average (317 s); rminer – fourth best average (408 s), best in 2 datasets. Regarding the prediction performances, there is a high overall correlation between the validation and test scores (not shown in Figure 2, although the same effect is present in Tables IV and V), when considering all tool execution values: 0.75 – binary; 0.90 – multi-class; and 0.92 – regression. For binary classification, and when considering the test set results, the AutoML differences are smaller for churn (maximum difference of 3 *percentage points* - *pp*) and higher for the other datasets (10 *pp* for diabetes, 15 *pp* for credit and 16 *pp* for qsar). TransmogrifAI is the best tool in 3 of the datasets (churn, credit and qsar), also obtaining the best average AUC per dataset (88%). An almost identical average (87%) is achieved by H2O (best in churn and credit), rminer (best in diabetes) and TPOT (best in churn). AutoGluon and Auto-Sklearn produced the worst overall results (average AUCs per dataset of 78% and 80%). Turning to multi-class tasks, the AutoML differences (best tool test result minus the worst one) are smaller when compared with the binary task: 4 *pp* – Cmc; 5 *pp* – Dmft; 6 *pp* – Mfeat; and 8 *pp* – Vehicle. The best test dataset average is obtained by AutoGluon (Macro F1-Score 58%), followed by Auto-Sklearn, H2O and TPOT (Macro F1-Score of 57%), then TransmogrifAI (56%) and finally rminer (53%). In terms of datasets, the best results were: Cmc – Auto-Sklearn (54%); Dmft – TransmogrifAI (24%); Mfeat – AutoGluon, Auto-Sklearn and TPOT (74%); and vehicle - AutoGluon and Auto-Sklearn (82%). As for the regression tasks, the AutoML tool differences for each dataset are very small, corresponding to 1 *pp* in terms of NMAE for all three datasets. In effect, all tools obtain the same average NMAE per dataset (9%). Using the lexicographic selection (Section IV), the GML tool recommendation is: binary - TransmogrifAI; multi-class - AutoGluon; regression – rminer.

The DL benchmark consisted of 12 (dataset) $\times$ 4 (tools) $\times$ 10 (folds) = 480 AutoML executions. Table IV shows the average DL 10-fold results ($\pm$ the 95% confidence intervals) in terms of the external computational effort (**Time**), internal validation (**Val.**) and test scores (**Test**). The Auto-Keras and Auto-PyTorch validation scores are omitted, since they are not disclosed by the tools. Regarding execution time, AutoGluon is much faster than the other tools, requiring an average fit time of just 24 s. The second fastest DL tool is Auto-Keras (average of 984 s), followed by H2O (3,458 s) and then Auto-PyTorch (3,600 s). As for the prediction performances, the average test values per dataset are: binary (AUC) - H2O (85%), Auto-PyTorch (77%); AutoGluon (72%) and Auto-Keras (69%); multi-class (Macro F1-score) - Auto-Gluon (57%), Auto-PyTorch (56%); H2O (50%) and Auto-Keras (43%); regression (NMAE) - H2O (10%); Auto-PyTorch and AutoGluon (11%); Auto-Keras (13%). While only four tools are compared, larger differences among the tools were obtained for the DL scenario when compared with GML: binary - ranging from 11 *pp* (Qsar) to 24 *pp* (credit); multi-class - from 4 *pp* to 30 *pp*; and regression – from 2 *pp* to 10 *pp*.

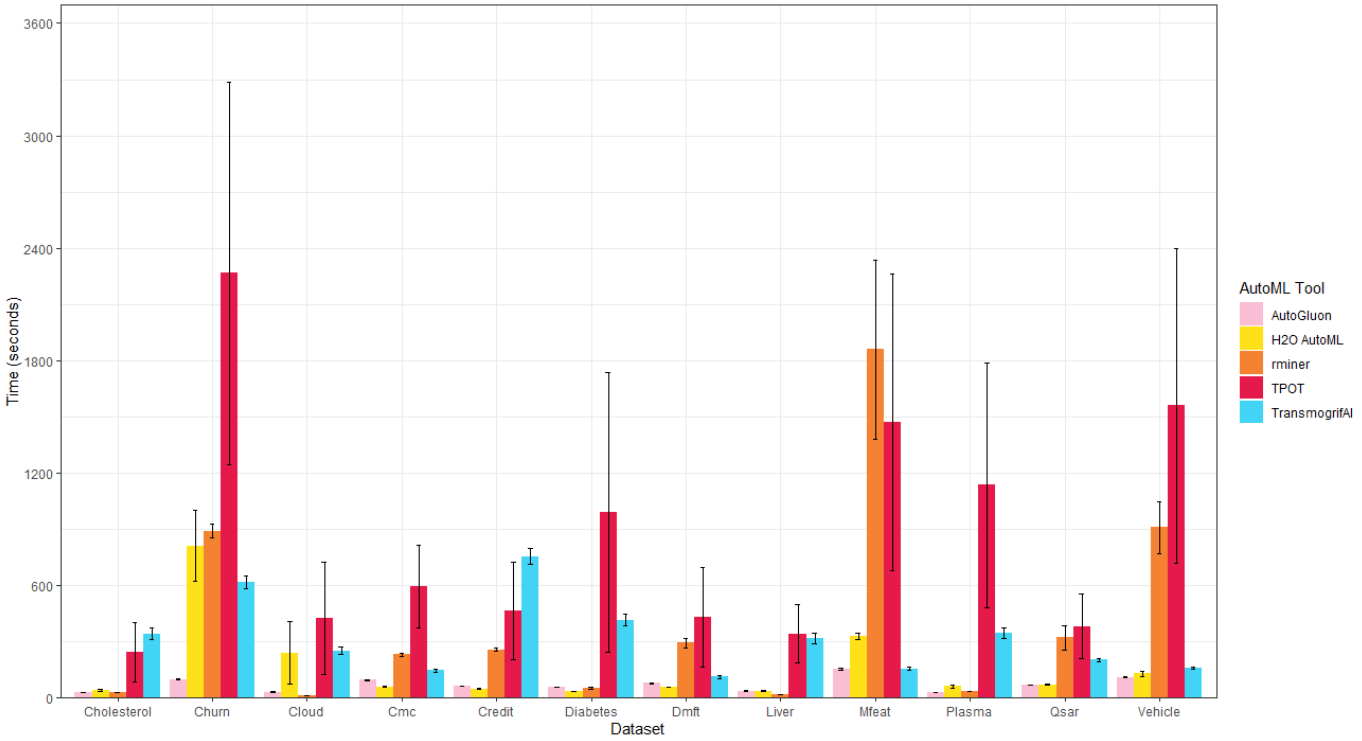

Fig. 1. Execution time (y-axis) for the GML scenario (bars denote external 10-fold average values with 95% confidence intervals; the Auto-Sklearn values were omitted from the graph because they are always constant and equal to 3,600 s).

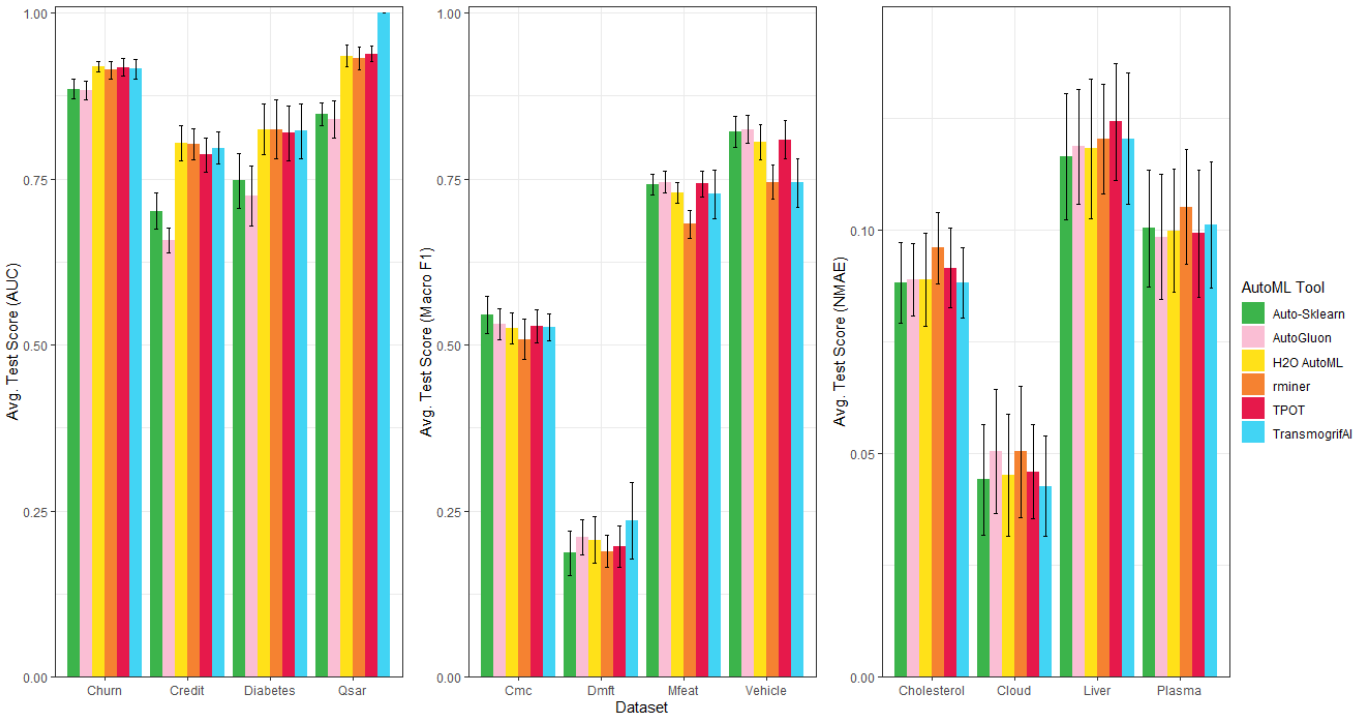

Fig. 2. Predictive results for the GML scenario (bars denote external 10-fold average values with 95% confidence intervals).

The lexicographic recommendation for DL is: binary - H2O; multi-class - AutoGluon; regression – H2O. However, when considering both GML and DL scenarios, the lexicographic selection favors GML tools.

TABLE IV
RESULTS FOR THE DL SCENARIO (BEST VALUES IN BOLD).

Dataset	Tool	Time	Measure	Val.	Test
Churn	Auto-Keras	2294±309	AUC	-	0.74±0.04
	Auto-PyTorch	3600±000		-	0.81±0.08
	AutoGluon	**0041**±003		0.90±0.00	0.80±0.02
	H2O AutoML	3600±000		**0.92**±0.00	**0.92**±0.02
Credit	Auto-Keras	1498±333		-	0.52±0.02
	Auto-PyTorch	3600±000		-	0.68±0.04
	AutoGluon	**0021**±002		0.77±0.01	0.57±0.03
	H2O AutoML	3600±000		**0.78**±0.00	**0.76**±0.04
Diabetes	Auto-Keras	0828±179		-	0.70±0.04
	Auto-PyTorch	3600±000		-	0.73±0.03
	AutoGluon	**0022**±001		0.79±0.01	0.65±0.05
	H2O AutoML	3600±000		**0.84**±0.01	**0.82**±0.03
Qsar	Auto-Keras	1154±336		-	0.82±0.03
	Auto-PyTorch	3600±000		-	0.87±0.02
	AutoGluon	**0026**±003		0.91±0.00	0.84±0.03
	H2O AutoML	3600±000		**0.92**±0.00	**0.92**±0.02
Cmc	Auto-Keras	0884±139	Macro F1	-	0.45±0.04
	Auto-PyTorch	3600±000		-	0.51±0.04
	AutoGluon	**0027**±003		**0.55**±0.00	**0.53**±0.02
	H2O AutoML	3600±000		**0.55**±0.00	0.48±0.06
Dmft	Auto-Keras	0478±045		-	0.16±0.04
	Auto-PyTorch	3600±000		-	**0.18**±0.04
	AutoGluon	**0018**±002		0.21±0.01	0.17±0.02
	H2O AutoML	3600±000		**0.23**±0.01	0.14±0.04
Mfeat	Auto-Keras	0713±095		-	0.44±0.04
	Auto-PyTorch	3600±000		-	0.73±0.02
	AutoGluon	**0032**±002		**0.76**±0.00	**0.74**±0.01
	H2O AutoML	3600±000		0.69±0.01	0.58±0.06
Vehicle	Auto-Keras	0708±053		-	0.65±0.05
	Auto-PyTorch	3600±000		-	**0.85**±0.02
	AutoGluon	**0038**±003		**0.85**±0.01	0.82±0.03
	H2O AutoML	3600±000		0.82±0.01	0.78±0.03
Cholesterol	Auto-Keras	0665±059	MAE	38.4±3.2	57.7±29.3
	Auto-PyTorch	3600±000		59.5±3.9	60.5±24.8
	AutoGluon	**0013**±002		39.0±0.5	**39.4**±4.0
	H2O AutoML	2509±279		**37.3**±0.4	39.9±4.1
Cloud	Auto-Keras	0549±572		**0.00**±0.00	0.88±0.36
	Auto-PyTorch	3600±000		0.20±0.04	**0.30**±0.10
	AutoGluon	**0013**±001		0.39±0.03	0.43±0.12
	H2O AutoML	3256±437		0.25±0.01	0.32±0.11
Liver Disorders	Auto-Keras	0879±797		2.67±0.29	2.60±0.37
	Auto-PyTorch	3600±000		2.54±0.14	**2.41**±0.35
	AutoGluon	**0018**±003		3.39±0.20	3.44±0.33
	H2O AutoML	3326±546		**2.23**±0.04	2.68±0.33
Plasma	Auto-Keras	1156±111		**127**±006	170±020
	Auto-PyTorch	3600±000		175±091	191±097
	AutoGluon	**0014**±002		156±004	155±022
	H2O AutoML	3600±000		151±003	**160**±017

Table V presents the XGB scenario results (total of 240 AutoML executions). Both tools require a low computational effort, with rminer presenting the faster fit times (average of 5 s), while H2O requires around 16 times more computation (average of 80 s). Both tools also present similar predictive performances, with the tool differences ranging from 0 to 2 *pp*. The lexicographic selection for XGB favors: binary - rminer (average AUC of 86%); multi-class - H2O (average Macro F1-score of 55%); regression - rminer (average NMAE of 9%). When considering both DL and XGB scenarios, the lexicographic choice favors rminer XGB for the binary classification and regression tasks, while AutoGluon DL is the selected tool for multi-class. When analyzing all three scenarios, the overall lexicographic selection is: binary – TransmogrifAI GML; multi-class – AutoGluon GML; regression - rminer XGB.

TABLE V
RESULTS FOR THE XGB SCENARIO (BEST VALUES IN BOLD).

Dataset	Tool	Time	Measure	Val.	Test
Churn	H2O	356±182	AUC	**0.93**±0.00	**0.92**±0.01
	rminer	**006**±000		0.92±0.00	**0.92**±0.01
Credit	H2O	021±002		**0.79**±0.01	**0.79**±0.02
	rminer	**004**±000		0.77±0.01	**0.79**±0.02
Diabetes	H2O	013±002		**0.83**±0.01	**0.82**±0.05
	rminer	**003**±001		0.81±0.01	**0.82**±0.04
Qsar	H2O	037±004		**0.93**±0.00	**0.93**±0.01
	rminer	**004**±000		**0.93**±0.00	**0.93**±0.02
Cmc	H2O	035±002	Macro F1	**0.53**±0.01	**0.53**±0.02
	rminer	**006**±000		**0.53**±0.01	0.52±0.03
Dmft	H2O	034±002		**0.23**±0.01	**0.21**±0.04
	rminer	**009**±000		0.20±0.01	0.19±0.02
Mfeat	H2O	121±009		**0.72**±0.06	**0.71**±0.02
	rminer	**015**±001		**0.72**±**0.01**	**0.71**±0.02
Vehicle	H2O	088±012		**0.77**±0.00	**0.77**±0.04
	rminer	**007**±000		0.76±0.01	0.75±0.03
Cholesterol	H2O	025±005	MAE	**39.1**±0.5	**39.2**±4.2
	rminer	**002**±000		42.7±1.0	42.4±5.1
Cloud	H2O	167±120		**0.28**±0.02	**0.30**±0.08
	rminer	**002**±000		0.31±0.01	**0.30**±0.08
Liver Disorders	H2O	020±004		**2.29**±0.04	**2.35**±0.32
	rminer	**002**±000		2.49±0.05	2.45±0.30
Plasma	H2O	042±008		**154**±003	**154**±022
	rminer	**003**±001		171±004	170±021

Finally, we contrast the best main GML scenario results (which consider more ML algorithms and AutoML tools) with the best public OpenML results (Table VI). For each dataset, we show in rounded brackets the best GML AutoML tool and the type of OpenML modeling (the algorithm name or "Pipeline", with the latter denoting a ML workflow that includes a data preparation step). While the best OpenML result includes predictions for all external 10-fold instances, we do not know the exact validation and testing procedures adopted. Thus, rather than assuming a "correct" comparison, we use here the best OpenML results as a "gold standard", denoting a proxy to the best results that can be achieved when using a human expert ML modeling. The column **Attempts** from Table VI denotes the number of human ML attempts, termed as a "run" in OpenML. The higher the number of attempts, the stronger is our assumption that the gold standard was reached. While all 12 datasets have high download numbers,

TABLE VI
COMPARISON BETWEEN BEST GML SCENARIO AND BEST OPENML RESULTS (BEST VALUES IN BOLD).

Dataset	Measure	Best		OpenML
		AutoML	OpenML	Attempts
Churn	AUC	0.919 (H2O AutoML)	**0.930** (Pipeline)	5,132
Credit		**0.803** (H2O AutoML)	0.800 (Ranger)	419,021
Diabetes		0.825 (rminer)	**0.842** (XGB)	132,164
Qsar		**1.000** (TransmogrifAI)	0.938 (XGB)	147,659
Cmc	Macro F1	0.545 (Auto-Sklearn)	**0.572** (Pipeline)	21,446
Dmft		0.236 (TransmogrifAI)	**0.262** (Pipeline)	19,445
Mfeat		0.745 (AutoGluon)	**0.772** (Pipeline)	22,136
Vehicle		0.820 (AutoGluon)	**0.870** (Pipeline)	23,532
Cholesterol	MAE	**38.59** (TransmogrifAI)	38.60 (Cforest)	160
Cloud		**0.256** (TransmogrifAI)	0.268 (Pipeline)	5
Liver Disorders		2.329 (Auto-Sklearn)	**2.309** (Pipeline)	77
Plasma		**152.3** (AutoGluon)	152.6 (Pipeline)	15

the attempts distribution is highly unbalanced towards the classification tasks, particularly the binary ones (e.g., Credit has more than 419,000 attempts). The results from Table VI confirm the quality of the AutoML. In effect, the tools obtained prediction scores that are close to the best OpenML results in seven datasets (e.g., the maximum difference is 2 and 5 *pp* for the binary and multi-task classification tasks). More importantly, the AutoML outperformed the best OpenML for three regression tasks and for two highly modeled binary datasets.

VI. CONCLUSIONS

In this paper, we benchmark eight recent open-source supervised learning Automated Machine Learning (AutoML) tools: Auto-Keras, Auto-PyTorch, Auto-Sklearn, AutoGluon, H2O AutoML, rminer, TPOT and TransmogrifAI. A large set of computational experiments was held by considering an external 10-fold cross-validation, twelve datasets and three tool comparison scenarios. Each tool was benchmarked by measuring its computational effort and predictive scores. We retrieved popular datasets from the OpenML platform, which were equally grouped into regression, binary and multi-class classification tasks. The three comparison scenarios were: General Machine Learning (GML) - with a broad range of classical ML algorithms; Deep Learning (DL) - focusing on tools with DL Neural Architecture Search (NAS) capabilities; and XGBoost (XGB) - considering a single XGB algorithm hyperparameter tuning.

To select the best tools for each scenario, we adopted a lexicographic approach, which considers first, for each task, the best average predictive score and then the lowest computational effort. For GML, the lexicographic selection favors TransmogrifAI for binary classification, AutoGluon for multi-class classification and rminer for regression. For DL, the selection is H2O for the binary and regression tasks and AutoGluon for regression. As for the XGB scenario, rminer is the best overall option for binary and regression tasks, while H2O is recommended for multi-class tasks.

A global overall analysis, considering all three scenarios, favors the GML approach, which produced the best predictive scores. This result should be taken with some caution, since GML explored more ML algorithms and AutoML tools. Nevertheless, the slightly lower AutoML DL predictive performances might be explained by two factors. Firstly, the analyzed datasets are relatively "small", with the largest dataset containing only 5,000 instances. And it is known that DL tends to produce better results (when compared with shallow methods) when modeling big data [33]. Secondly, the AutoML tools with DL capabilities are more recent and thus might be still immature when compared with GML tools. For instance, the tested Auto-PyTorch and AutoGluon versions are still in their zero dot something versions (e.g., 0.0.2). To further measure the quality of the GML AutoML modeling, we compared the best GML results with the best predictions publicly available at the OpenML platform. The OpenML comparison confirmed that current GML AutoML tools provide competitive results, producing close predictions in seven datasets and even outperforming the human ML modeling in five datasets.

In future work, we intend to enlarge the comparison by considering more open-source AutoML technologies and datasets. In particular, we wish to analyze big data, where DL can potentially produce better predictions. We also plan to benchmark ML frameworks for specific infrastructure settings, such as involving edge computing.

ACKNOWLEDGMENTS

This work was executed under the project Opti-Edge: 5G Digital Services Optimization at the Edge, Individual Project, NUP: POCI-01-0247-FEDER-045220, co-funded by the Incentive System for Research and Technological Development, from the Thematic Operational Program Competitiveness of the national framework program - Portugal2020.

REFERENCES

[1] Y. He, J. Lin, Z. Liu, H. Wang, L. Li, and S. Han, "AMC: automl for model compression and acceleration on mobile devices," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11211. Springer, 2018, pp. 815–832.

[2] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-weka: combined selection and hyperparameter optimization of classification algorithms," in *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, I. S. Dhillon, Y. Koren, R. Ghani, T. E. Senator, P. Bradley, R. Parekh, J. He, R. L. Grossman, and R. Uthurusamy, Eds. ACM, 2013, pp. 847–855.

[3] I. Guyon, L. Sun-Hosoya, M. Boullé, H. J. Escalante, S. Escalera, Z. Liu, D. Jajetic, B. Ray, M. Saeed, M. Sebag, A. R. Statnikov, W. Tu, and E. Viegas, "Analysis of the automl challenge series 2015-2018," in *Automated Machine Learning - Methods, Systems, Challenges*, ser. The Springer Series on Challenges in Machine Learning, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Springer, 2019, pp. 177–219.

[4] A. Truong, A. Walters, J. Goodsitt, K. E. Hines, C. B. Bruss, and R. Farivar, "Towards automated machine learning: Evaluation and comparison of automl approaches and tools," in *31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019*. IEEE, 2019, pp. 1471–1479.

[5] P. Das, N. Ivkin, T. Bansal, L. Rouesnel, P. Gautier, Z. S. Karnin, L. Dirac, L. Ramakrishnan, A. Perunicic, I. Shcherbatyi, W. Wu, A. Zolic, H. Shen, A. Ahmed, F. Winkelmolen, M. Miladinovic, C. Archembeau, A. Tang, B. Dutt, P. Grao, and K. Venkateswar, "Amazon sagemaker autopilot: a white box automl solution at scale," in *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, S. Schelter, S. Whang, and J. Stoyanovich, Eds. ACM, 2020, pp. 2:1–2:7.

[6] J. Z. Liang, E. Meyerson, B. Hodjat, D. Fink, K. Mutch, and R. Miikkulainen, "Evolutionary neural automl for deep learning," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, A. Auger and T. Stützle, Eds. ACM, 2019, pp. 401–409.

[7] N. Dhir, T. Rudny, D. Zilli, and A. Tosi, "An automatic type-inferential general latent feature model," in *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*. IEEE, 2020, pp. 1–8.

[8] J. Waring, C. Lindvall, and R. Umeton, "Automated machine learning: Review of the state-of-the-art and opportunities for healthcare," *Artif. Intell. Medicine*, vol. 104, p. 101822, 2020.

[9] Y. Chen, Q. Song, and X. Hu, "Techniques for automated machine learning," *SIGKDD Explor.*, vol. 22, no. 2, pp. 35–50, 2020. [Online]. Available: https://doi.org/10.1145/3447556.3447567

[10] L. Zimmer, M. Lindauer, and F. Hutter, "Auto-pytorch tabular: Multi-fidelity metalearning for efficient and robust autodl," *CoRR*, vol. abs/2006.13799, 2020. [Online]. Available: https://arxiv.org/abs/2006.13799

[11] L. Ferreira., A. Pilastri., C. Martins., P. Santos., and P. Cortez., "An automated and distributed machine learning framework for telecommunications risk management," in *Proceedings of the 12th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, INSTICC. SciTePress, 2020, pp. 99–107.

[12] H. A. Neto, R. C. Alves, and S. V. Campos, "Nasirt: Automl based learning with instance-level complexity information," *arXiv preprint arXiv:2008.11846*, 2020.

[13] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola, "Autogluon-tabular: Robust and accurate automl for structured data," *arXiv preprint arXiv:2003.06505*, 2020.

[14] M. Zöller and M. F. Huber, "Benchmark and survey of automated machine learning frameworks," *arXiv preprint arXiv:1904.12054*, 2020. [Online]. Available: http://arxiv.org/abs/1904.12054

[15] M. Hanussek, M. Blohm, and M. Kintz, "Can automl outperform humans? an evaluation on popular openml datasets using automl benchmark," *CoRR*, vol. abs/2009.01564, 2020. [Online]. Available: https://arxiv.org/abs/2009.01564

[16] I. Drori, L. Liu, Y. Nian, S. C. Koorathota, J. S. Li, A. K. Moretti, J. Freire, and M. Udell, "Automl using metadata language embeddings," *arXiv preprint arXiv:1910.03698*, 2019.

[17] C. Wang and Q. Wu, "Flo: Fast and lightweight hyperparameter optimization for automl," *arXiv preprint arXiv:1911.04706*, 2019.

[18] P. Gijsbers, E. LeDell, J. Thomas, S. Poirier, B. Bischl, and J. Vanschoren, "An open source automl benchmark," *arXiv preprint arXiv:1907.00909*, 2019.

[19] R. Gimeno Saborit, "Benchmark auto machine learning," Universitat Autonoma de Barcelona, Spain, Tech. Rep., 2019.

[20] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-sklearn 2.0: The next generation," *arXiv preprint arXiv:2007.04074*, 2020.

[21] K. Jing, J. Xu, and H. Xu, "NASABN: A neural architecture search framework for attention-based networks," in *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*. IEEE, 2020, pp. 1–7.

[22] A. Yakovlev, H. F. Moghadam, A. Moharrer, J. Cai, N. Chavoshi, V. Varadarajan, S. R. Agrawal, T. Karnagel, S. Idicula, S. Jinturkar, and N. Agarwal, "Oracle automl: A fast and predictive automl pipeline," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 3166–3180, 2020.

[23] I. Xanthopoulos, I. Tsamardinos, V. Christophides, E. Simon, and A. Salinger, "Putting the human back in the automl loop," in *Proceedings of the Workshops of the EDBT/ICDT 2020 Joint Conference, Copenhagen, Denmark, March 30, 2020*, ser. CEUR Workshop Proceedings, A. Poulovassilis, D. Auber, N. Bikakis, P. K. Chrysanthis, G. Papastefanatos, M. A. Sharaf, N. Pelekis, C. Renso, Y. Theodoridis, K. Zeitouni, T. Cerquitelli, S. Chiusano, G. Vargas-Solar, B. Omidvar-Tehrani, K. Morik, J. Renders, D. Firmani, L. Tanca, D. Mottin, M. Lissandrini, and Y. Velegrakis, Eds., vol. 2578. CEUR-WS.org, 2020. [Online]. Available: http://ceur-ws.org/Vol-2578/ETMLP5.pdf

[24] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1946–1956.

[25] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 2962–2970. [Online]. Available: http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning

[26] Auto-Gluon, "AutoGluon: AutoML Toolkit for Deep Learning — AutoGluon Documentation 0.0.1 documentation," 2020. [Online]. Available: https://autogluon.mxnet.io/

[27] D. Cook, *Practical machine learning with H2O: powerful, scalable techniques for deep learning and AI*. " O'Reilly Media, Inc.", 2016.

[28] P. Cortez, "Data mining with neural networks and support vector machines using the r/rminer tool," in *Industrial Conference on Data Mining*. Springer, 2010, pp. 572–583.

[29] R. S. Olson, R. J. Urbanowicz, P. C. Andrews, N. A. Lavender, L. C. Kidd, and J. H. Moore, *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I*. Springer International Publishing, 2016, ch. Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, pp. 123–137.

[30] Salesforce, "Transmogrifai," 2019, https://transmogrif.ai/.

[31] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "Openml: networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013.

[32] A. A. Freitas, "A critical review of multi-objective optimization in data mining: a position paper," *SIGKDD Explor.*, vol. 6, no. 2, pp. 77–86, 2004. [Online]. Available: https://doi.org/10.1145/1046456.1046467

[33] A. Ng, *Machine Learning Yearning*. deeplearning.ai, 2020. [Online]. Available: https://www.deeplearning.ai/machine-learning-yearning/