Open source AutoML: machine learning for everyone?

Andrea Lanari

February 18, 2021

Abstract

During the past decade, Machine Learning (ML) popularity has been skyrocketing: year by year, the models are becoming more and more accurate and they are finding applications in a huge variety of fields. Nowadays, even if ML is widely used, developing a good Machine Learning system is not so easy: in particular, the model choice and hyperparameters tuning are two aspects that only experts can handle successfully through many tests. Therefore, researchers in the field are tackling this problem making use of a new method: Automated Machine Learning (AutoML). AutoML is the possibility to delete the "expert phase" in favor of a more complex system able to find the best approach independently. In this work, we focus on the open source tools already available, trying to compare their performance and their actual autonomy.

1 Introduction: definitions and domain

For the past ten years, Machine Learning (ML) and its sub-subject (in particular Deep Learning (DL)) has been introduced in several fields and nowadays they are so common that some new smartphones have a dedicated processor (called NPU) in order to improve ML and DL tasks such as image classification, object detection and many others.
Even if ML has managed to take the next step into the "real application" world, developing these types of systems is still challenging: indeed, to build a good machine learning model, an expert that designs the best configuration through the trial-and-error approach is necessary, with a consequent expense of resources and time. For this reason, lately a new branch of Machine Learning, called Automated Machine Learning (AutoML), has started to develop quickly.

Among the various possible definitions, according to [22], Automated Machine Learning is "a subject designed to reduce the demand for data scientists and enable domain experts to automatically build ML applications without much requirement for statistical and ML knowledge". In other words, the main goal is to minimize as much as possible the human participation in the Machine Learning process.
Starting from the name, AutoML is the combination of two big branches of computer science: machine learning and autonomous systems. In order to better understand the domain under consideration, it may be helpful to remember some definitions. First of all, the definition of Machine Learning is "the study of computer algorithms that improve automatically through experience" [1]. On the contrary, it is not easy to give a single definition about autonomous systems: indeed, it includes several concepts distributed in many subjects. Despite this, we are able to describe them with some adjectives common in all fields such as "self-government", "well-defined bordered" and "self-determination".
Currently, several companies/universities/communities have developed a solution related to AutoML: from Google to Amazon, but also Uber, MIT and Weka. In this work, we will first try to give a general overview about the current state of the art of AutoML, focusing on open source tools. Then we will provide a comparison between them, underlining the relation between their actual autonomy and the results obtained.

2 Related work

As previously mentioned, AutoML has increased a lot its popularity in the last ten years, the topic has been a matter of research in many respects, producing much literature on the subject. In order to understand the starting point, the survey proposed from [13] is probably the most complete work about AutoML right now.
An important inspiration linked to the Open Source AutoML is presented in [14], where an AutoML Benchmark is described with some performance test: however, that tool is not used in this work.
Another important related work is [18], that was used as a reference regarding the autonomy evaluation.

3 Auto Machine Learning classification

In the literature, many ways have been suggested in order to develop AutoML systems. We will now give an overview of the available tools from the point of view of machine learning, and their classification. According to [13], there are also different classification schemes regarding automated machine learning tools. In this paper, we propose the division into five different categories: feature engineering, optimization algorithm selection, model selection, general full scope and NAS (Neural Architecture Search). As can be easily deduced from the class

names, this partition is based on the problem setup: each category is one or more phases of the machine learning pipeline. Many of the works we will cite cannot always be classified in only one group; often they intersect more than one. Here is the list with the categories, their description and some examples:

- **Feature Engineering**: it is the process of transforming raw data into features that better represent the underlying problem. FeatureTool [7], for example, is one of the most famous service about feature engineering: it uses Deep Feature Synthesis to perform this task on relational and temporal data. Another important work related to this topic is ExploreKit [8], which first generates a very large space of possible solutions combined with the initial one, and then reduces it through a machine learning algorithm. Some general full scope and NAS tools also permit to figure out this problem such as Auto-Weka [4] or AutoSKLearn [6].

- **Optimization Algorithm Selection**: it is the phase in which, after selecting a model, we try to find the best setting for it. Regarding this part, we can mention HpBandSter [12] that use a bayesan optimization or hyperopt-sklearn [5] built over the famous SkLearn Python package. Also in this case, there are some frameworks that include the possibility to optimize the hyperparameters, although this is not their only feature: NNI [21] is an example.

- **Model Selection**: this catagory has more than one phase of machine learning pipeline included. For example, automl-gs [23] enables to generate the entire sequence of a ML solution, without writing a line of code. ATM (Auto Tune Models) [11] solve classification problems by providing only a csv input and a target: they not only manage the configuration of the hyperparameters, but also independently choose what is the best model for them.

- **Neural architecture search**: these types of systems are designed to build the best possible architecture of neural networks, in order to perform deep learning on the input. Moreover, they also usually perform a first phase of feature engineering. Projects such as Microsoft's NNI [21] and AutoKeras [15] are probably the most important ones.

- **General full scope**: in this case, AutoML reaches the highest level of autonomy: in fact, these tools manage independently all aspects of learning from feature engineering to hyperparameters selection. For this category, AutoSkLearn [6] can provide an excellent example.

Having looked at AutoML from a machine learning perspective, now we will focus on how these tools are self-governed.
According to [17], they can be distinguished by how the controller is implemented: basic techniques and experienced ones.

Regarding basic methods, the literature shows many different methods to develop a controller for an AutoML tool. For example, TPOT [9] proposes a solution inspired by biology evolution: after doing a first part of feature engineering, the main idea is to create and evaluate several different generations of tree-based ML pipeline, in order to generate the best possible model. In particular, the algorithm is based on a genetic algorithm that, firstly, creates the first generation of 100 random tree-based ML pipelines. At this point, the algorithm is composed by two iterative phases: the first one is the evaluation one where it selects the best 20 ML pipelines of a generation through the accuracy. In the second part, instead, the algorithm creates a new generation starting from the selected elements.
Another approach is presented on AutoKeras [15] which makes use of Bayesian Optimization in order to optimize the best neural network architecture to solve the problem. More specifically, this algorithm, after building a Gaussian Process model, follows these 3 steps iteratively: first of all, The Gaussian Process Model is trained with configuration and performance, then it generates the next architecture to observe, optimizing a well-defined acquisition function. Finally, the neural architecture is trained in order to be evaluated and then the probabilistic model is updated (step 1).
MLjarSupervised [25] uses another different method, which is the greedy search presented in [2]. The main idea in this work is to find the best model through the selection of a local optimum: they start with an empty ensemble, which is filled with all the models that maximize the metrics used. Finally, the system selects the best model among the set created.

On the other hand, the second group, which makes use of experience-based techniques, such as meta-learning and transfer-learning, is less common.
A very good example of this kind of methods is AutoSkLearn. In both of its versions it makes use of meta-learning: the first one makes use of a k-nearest dataset, which provides the initial configuration where to start with Bayesian Optimization [6]. The second version [20], instead of using one k-nearest dataset, creates a portfolio of possible ML pipelines, which are evaluated directly through a comparison matrix, in order to use the best model.

4 The open source AutoML system

After seeing the proposed classification, in this section, we will focus on the general full scope and NAS projects, giving a description for each one tested on the proposed experiment. In fact, our goal is to evaluate the tools that can be most used by a wide audience, as they require very little training related to machine learning. Following is a list of all the instruments analyzed:

- AutoKeras: it is a NAS system based on Keras, a famous open source Python's library of TensorFlow. The framework uses Bayesian optimization to guide the network morphism for efficient neural architecture search. With just a few lines of code in Python, it allows to solve problems such

as classification and regression with datasets that consist of images, text and structured data.

- AutoSklearn: built above the library SkLearn, it provides a different approach compared to the existing AutoML methods: in fact, it automatically takes into account past performance on similar datasets and constructs ensembles from the models evaluated during the optimization. In particular, AutoSklearn makes use of 15 classifiers, 14 feature preprocessing methods, 4 data preprocessing methods, giving rise to a structured hypothesis space with 110 hyperparameters [6].

- Autogluon: developed by Amazon, Autogluon is a toolkit for deep learning, thus belonging to the category of NAS [19]. This is one of the most recent instruments and as the paper shows, it is also considered among the most performing ones at the moment.

- h2O: part of h2O.ai project, AutoML h2O is a neural architecture search system. What's unique about this tool is that it is part of a larger framework dedicated to machine learning. For this reason all the pipelines are developed using their specific library [24].

- Ludwig: developed by Uber, Ludwig is a "flexible, extensible and easy to use toolbox which allows users to train deep learning models and use them for obtaining predictions without writing code" [16]. Ludwig introduces a different approach based on a configuration file and data types: the first aspect would encourage reuse of sub-model code, while the second would enable inexperienced users to obtain actual models and increase productivity for more experienced users.

- mljarSupervised: it is a automated machine learning Python package that works only with tabular datasets: the main objective is to save time for data scientists. It is a general full scope tool: it abstracts the common way to preprocess the data, construct the machine learning models, and perform hyper-parameters tuning to find the best model. Moreover, we can see exactly how the ML pipeline is composed [25].

- Tpot: probably one of the first tool related to AutoML, Tpot (Tree-Based Pipeline Optimization Tool) combines a flexible expression tree representation of pipelines with stochastic search algorithms such as genetic programming, making use of the Python-based scikit-learn library as its ML menu [9].

5 The proposed comparison

After seeing the open source tools currently available, in this section, we will provide a comparison between them in terms of autonomy and results.
Before describing how the experiment works, in order to obtain a reliable comparison, we will make use only of those AutoML softwares which include all

steps of the machine learning process, such as NAS and general full scope. That said, the experiment aims to evaluate autonomy and machine learning through two different scenarios based on tabular datasets: the first one asks to complete a multi-classification task about the price range for smartphones, while the second is a regression task correlated with the life expectancy around the world.
The main idea is straightforward: both scenarios have their own dataset (.csv files), already split in test and training sets, that are used to train the different AutoML tools, so as to predict the label (classification) or the value (regression) for the test set.
During this, each tool is set up following its own documentation, trying to intervene as little as possible, relying on the default parameters. Once obtained the predicted labels or values, we started to evaluate these tools on two different sides: from the Machine Learning point of view and the autonomy one. Regarding ML, we decided to compare the prediction and the real value by using the mean accuracy, f1 score and confusion matrix for the classification and variance scores, the mean absolute error and actual vs. predicted plots, for the regression.
From the autonomy viewpoint, we took inspiration from what has been proposed in [18], trying to outline it for the AutoML systems. In the paper, the researchers suggest to evaluate the autonomy level through the external intervention needed. In order to do this, they proposed the following,

$$L = M_G / M_I,$$

where L is the level of autonomy, M_G represents the measure of the set of Goals G and M_I is the measure of the needed interventions I. Our idea is to assign a level 100 of autonomy if you can run a tool that predicts perfectly the results, after having received an acceptable training set, a target and a test set for each task considered and supported. Basically, we want to solve an AutoML problem just by typing a single line of code:

```
Automl_Tool_Name training test "target"
```

For this purpose, we decided to calculate M_G with this simple formula:

$$M_G = 100 \cdot \sum_{t=1}^{n} M_t$$

where n is the number of ML tasks considered and M_t is a metric for the task in the interval $[0-1]$. In our case, we use the accuracy for the classification and the variance score for the regression.
Before discussing about M_I, we need to define what is considered an "Intervention". Here we suggest a list of a possible interpretation:

- instantiation of an object;
- type definition;
- type conversion;

- call a function;
- parameter passage;
- run a script.

Consequently, we calculated M_I as

$$M_I = (n + I)/5 \cdot n,$$

where I are the number of interventions and n is the number of tasks. In this way, a perfect AutoML system invoked with line code 1 obtains the maximum level of autonomy, that is 100.

6 Results and Conclusion

6.1 AutoML tool results

- **AutoKeras**:

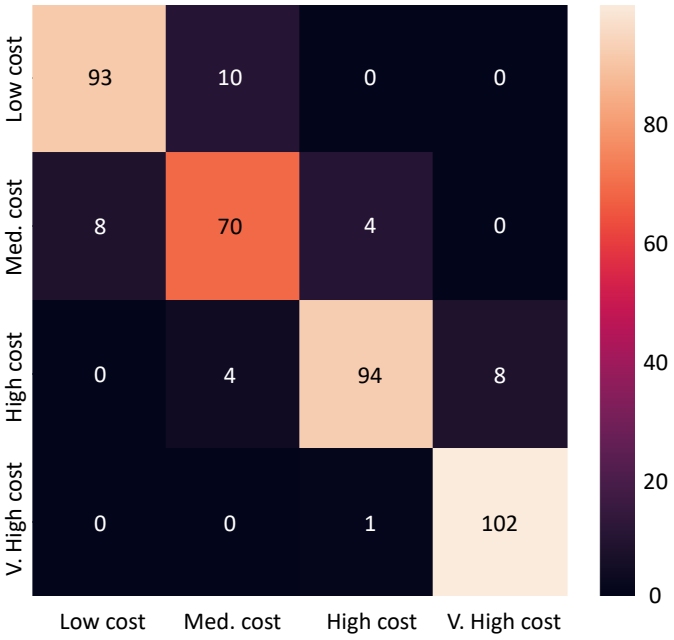

Figure 1: Confusion matrix

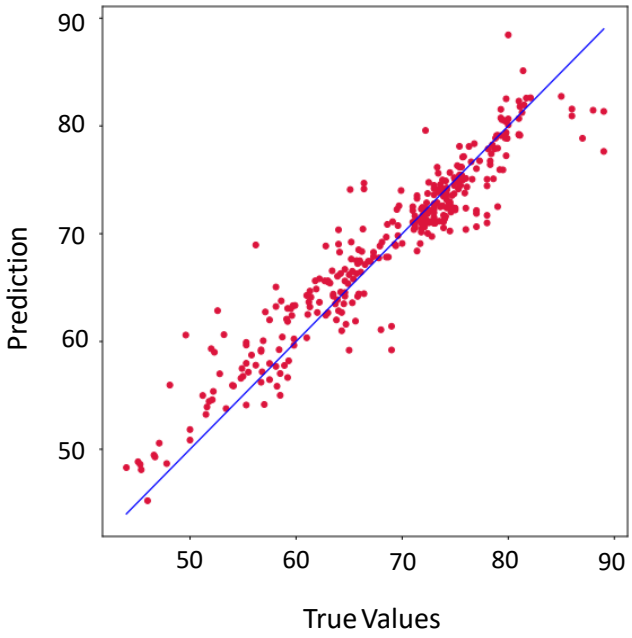

Figure 2: Predicted vs real values plot

Classification	Regression	Autonomy
Mean Accuracy: 0.8975	Variance score: 0.889336118	Level: 74
F1macro: 0.89361498	Mean absolute error: 2.18497534	Intervention: 25

- **Autogluon**:

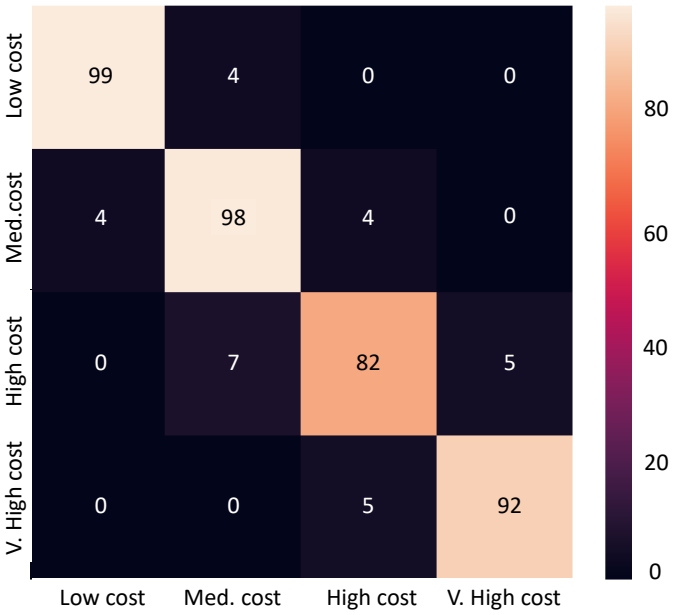

Figure 3: Confusion matrix

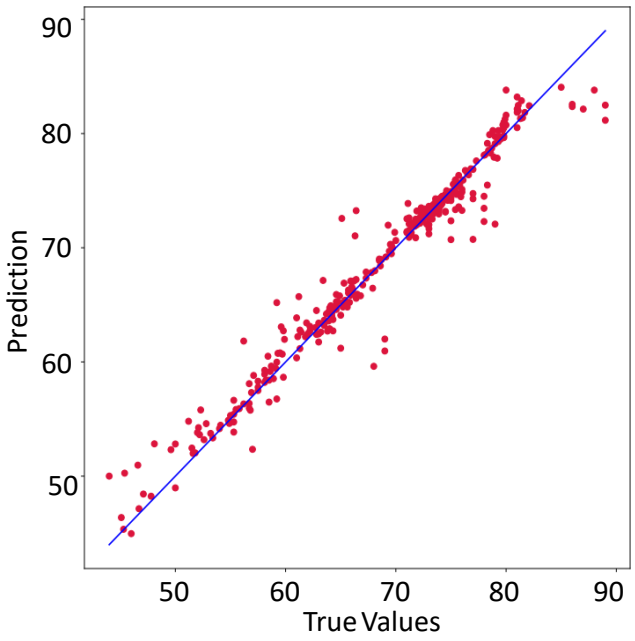

Figure 4: Predict vs real plot

Classification	Regression	Autonomy
Mean Accuracy: 0.9275	Variance score: 0.95667207	Level: 87
F1macro: 0.92693326	Mean absolute error: 1.1236627	Intervention: 20

- **AutoSkLearn**:

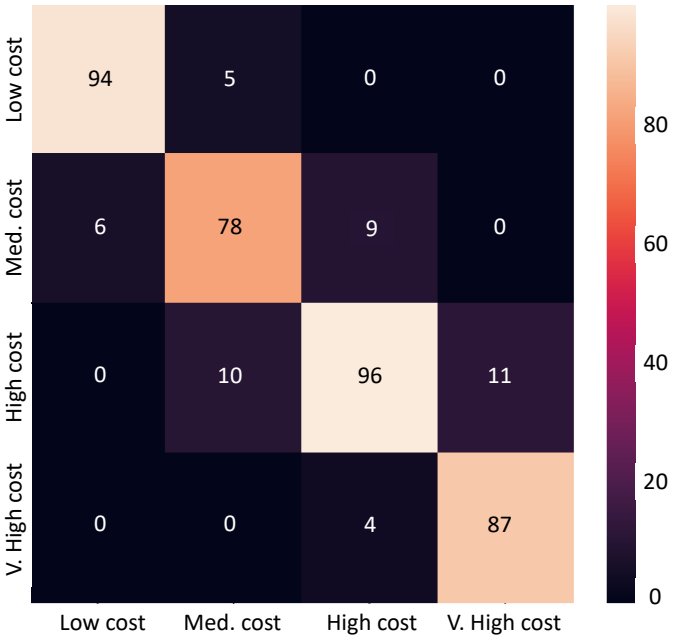

Figure 5: Confusion matrix

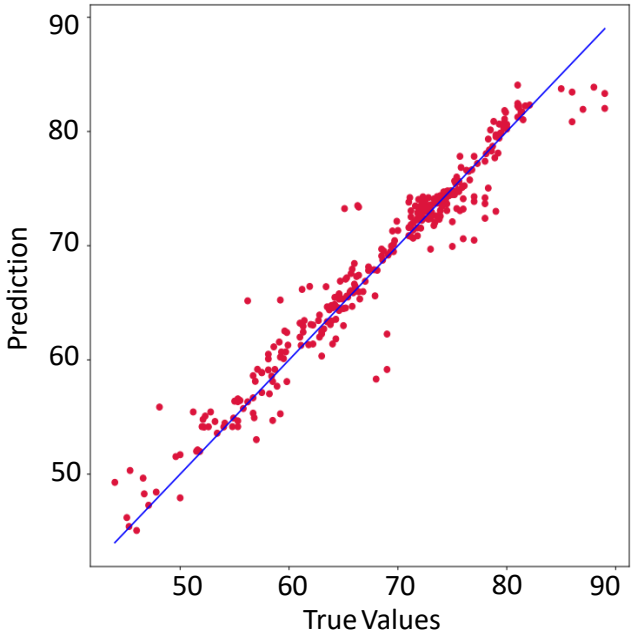

Figure 6: Predict vs real plot

Classification	**Regression**	**Autonomy**
Mean Accuracy: 0.8875	Variance score: 0.94449481	Level: 67
F1macro: 0.88840643	Mean absolute error: 1.38578796	Intervention: 25

- **Ludwig**:

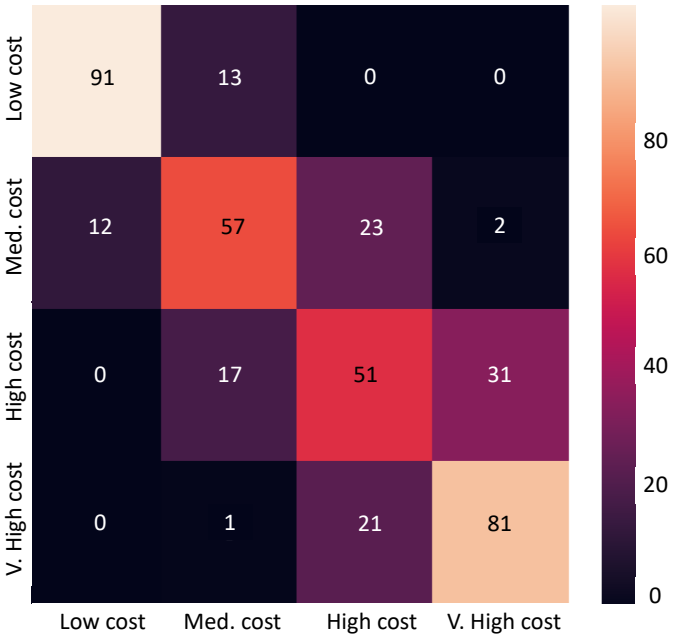

Figure 7: Confusion matrix

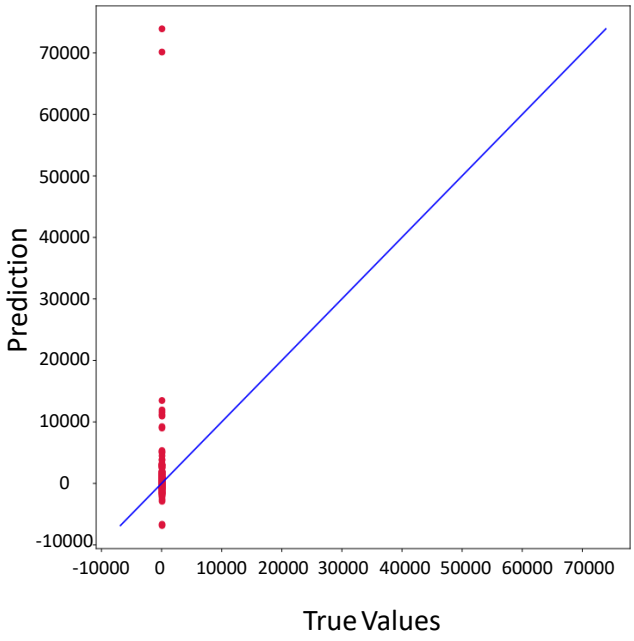

Figure 8: Predicted vs real values plot

Classification	**Regression**	**Autonomy**
Mean Accuracy: 0.51	Variance score: -17507.8463983	Level: 7
	can't perform the task	
F1macro: 0.501669978		Intervention: 75
	Mean absolute error: 369.54470812	

- **TPOT**:

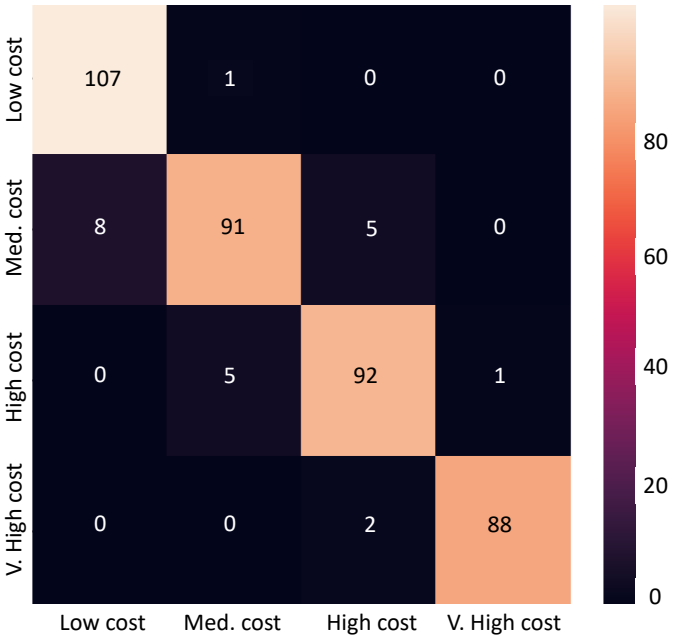

Figure 9: Confusion matrix

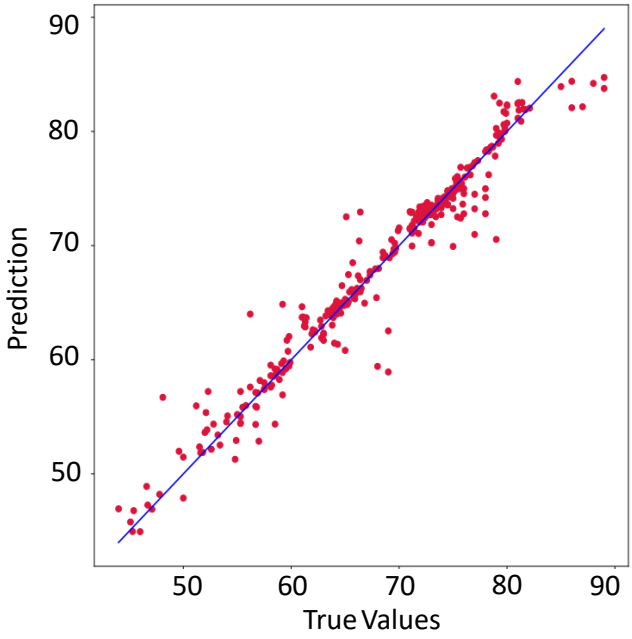

Figure 10: Predicted vs real values plot

Classification	**Regression**	**Autonomy**
Mean Accuracy: 0.945	Variance score: 0.95368282	Level: 70
F1macro: 0.94559107	Mean absolute error: 1.16024478	Intervention: 25

- **MLJarSupervised**:

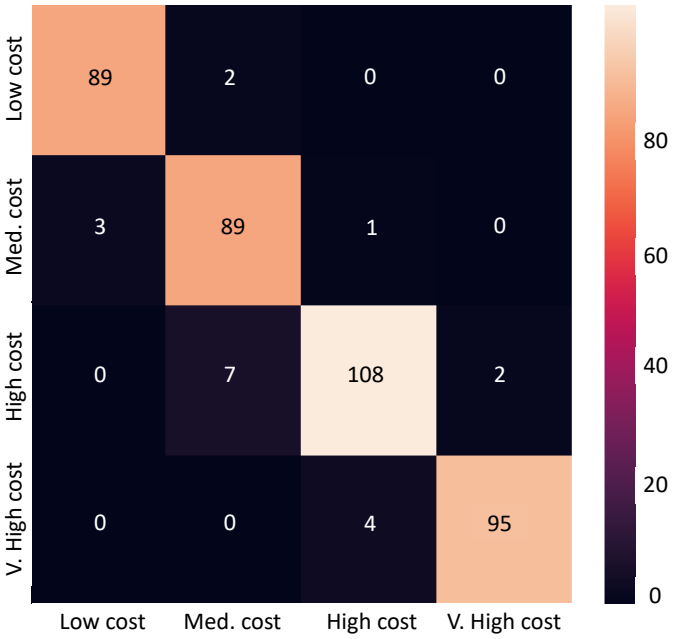

Figure 11: Confusion matrix

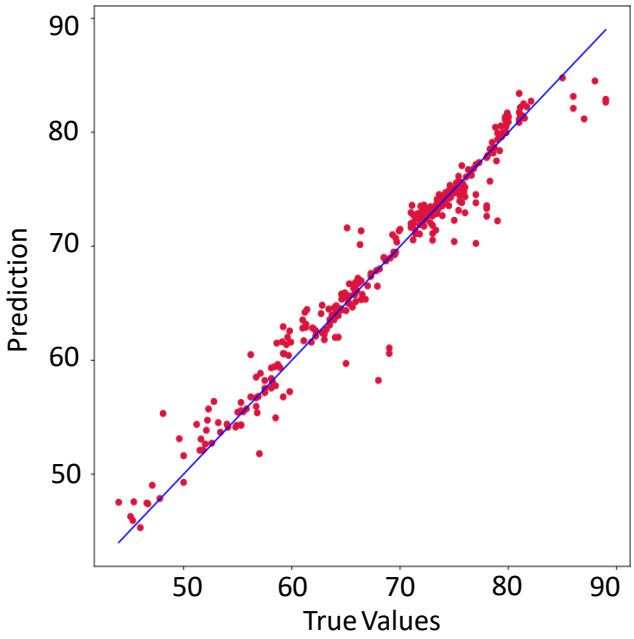

Figure 12: Predicted vs real values plot

Classification	**Regression**	**Autonomy**
Mean Accuracy: 0.9525	Variance score: 0.95727202	Level: 83
F1macro: 0.95328324	Mean absolute error: 1.15920469	Intervention: 21

- **H2O**:

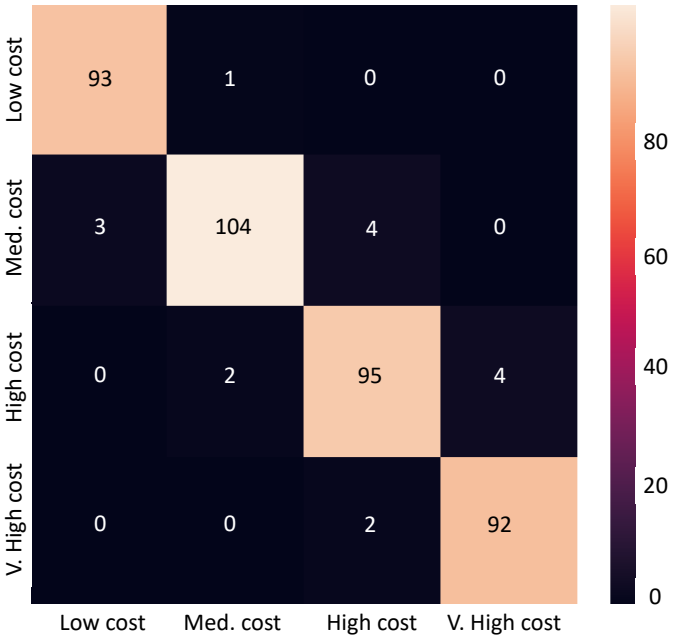

Figure 13: Confusion matrix

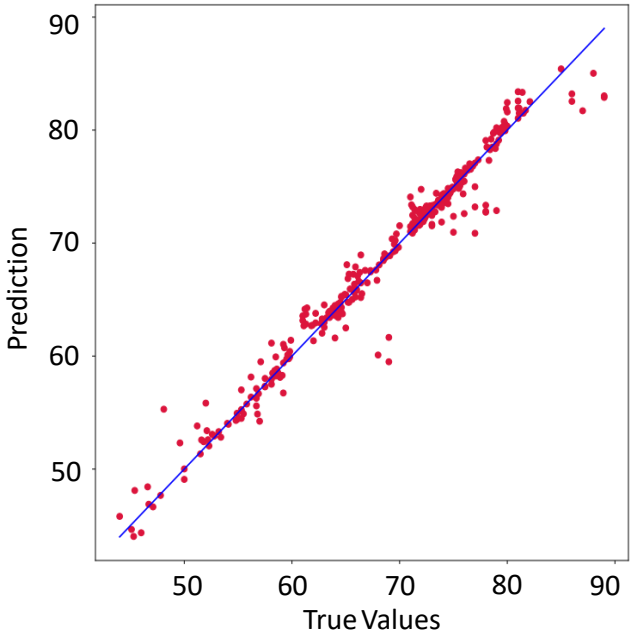

Figure 14: Predicted vs real values plot

Classification	**Regression**	**Autonomy**
Mean Accuracy: 0.965	Variance score: 0.9674258	Level: 59
F1macro: 0.96052273	Mean absolute error: 0.95082071	Intervention: 31

The results show that easy tasks like these can be generally performed by AutoML system. Indeed, except for Ludwig, all the others achieve good results, with a relative low number of interventions.
In particular, the H2O system has the best ML overall performance in both tasks, followed by Autogluon. Regarding the autonomy, H2O has to work in its environment, increasing the number of interventions required. In terms of autonomy level, Autogluon achieves the best result, combining a good ML performance with only 10 interventions more than necessary.

6.2 Conclusion

As we have seen, automated machine learning is nowadays a reality: the many tools available confirm this, even if they are not so popular. Although the goals to achieve were not difficult, the results show how effective these tools are: a few instructions are enough to easily complete such tasks with a very good result.
On the other hand, even if these tools are very powerful, they will most likely be able to help during the process of building a machine learning pipeline, rather than opening up this world to a wider audience: indeed, it is necessary to have some competence in order to manage the data and interpret the results, without blindly relying on machines.
In conclusion, this work confirms how much automated machine learning can help to speed up the trial-and-error process used to build this kind of models, although they are not yet sufficient to completely eliminate the figure of the expert.

References

[1] Tom M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2.

[2] Rich Caruana et al. "Ensemble Selection from Libraries of Models". In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 18. ISBN: 1581138385. DOI: 10.1145/1015330.1015432. URL: https://doi.org/10.1145/1015330.1015432.

[3] F. Alonso et al. "Towards a set of Measures for Evaluating Software Agent Autonomy". In: (2009), pp. 73–78. DOI: 10.1109/MICAI.2009.15.

[4] C. Thornton et al. "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms". In: *Proc. of KDD-2013*. 2013, pp. 847–855.

[5] Brent Komer, James Bergstra, and Chris Eliasmith. "Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn". In: Jan. 2014, pp. 32–37. DOI: 10.25080/Majora-14bd3278-006.

[6] Matthias Feurer et al. "Efficient and Robust Automated Machine Learning". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015, pp. 2962–2970. URL: https://proceedings.neurips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf.

[7] James Max Kanter and Kalyan Veeramachaneni. "Deep feature synthesis: Towards automating data science endeavors". In: (2015), pp. 1–10.

[8] Gilad Katz, Eui Shin, and Dawn Song. "ExploreKit: Automatic Feature Generation and Selection". In: Dec. 2016, pp. 979–984. DOI: 10.1109/ICDM.2016.0123.

[9] Randal S. Olson et al. "Automating Biomedical Data Science Through Tree-Based Pipeline Optimization". In: *Applications of Evolutionary Computation*. Ed. by Giovanni Squillero and Paolo Burelli. Cham: Springer International Publishing, 2016, pp. 123–137. ISBN: 978-3-319-31204-0.

[10] Corinna Cortes et al. *AdaNet: Adaptive Structural Learning of Artificial Neural Networks*. 2017. arXiv: 1607.01097 [cs.LG].

[11] Thomas Swearingen et al. "ATM: A distributed, collaborative, scalable system for automated machine learning". In: *2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, December 11-14, 2017*. 2017, pp. 151–162. DOI: 10.1109/BigData.2017.8257923. URL: https://doi.org/10.1109/BigData.2017.8257923.

[12] Stefan Falkner, Aaron Klein, and Frank Hutter. "BOHB: Robust and Efficient Hyperparameter Optimization at Scale". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, Oct. 2018, pp. 1437–1446. URL: http://proceedings.mlr.press/v80/falkner18a.html.

[13] Quanming Yao et al. "Taking Human out of Learning Applications: A Survey on Automated Machine Learning". In: *ArXiv* (2018).

[14] Pieter Gijsbers et al. "An Open Source AutoML Benchmark". In: (July 2019).

[15] Haifeng Jin, Qingquan Song, and Xia Hu. *Auto-Keras: An Efficient Neural Architecture Search System*. 2019. arXiv: 1806.10282 [cs.LG].

[16] Piero Molino, Yaroslav Dudin, and Sai Sumanth Miryala. *Ludwig: a type-based declarative deep learning toolbox*. 2019. arXiv: 1909.07930 [cs.LG].

[17] Quanming Yao et al. *Taking Human out of Learning Applications: A Survey on Automated Machine Learning*. 2019. arXiv: 1810.13306 [cs.AI].

[18] Panos Antsaklis. "Autonomy and metrics of autonomy". In: *Annual Reviews in Control* 49 (2020), pp. 15–26. ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2020.05.001. URL: https://www.sciencedirect.com/science/article/pii/S1367578820300274.

[19] Nick Erickson et al. *AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data*. 2020. arXiv: 2003.06505 [stat.ML].

[20] Matthias Feurer et al. *Auto-Sklearn 2.0: The Next Generation*. 2020. arXiv: 2007.04074 [cs.LG].

[21] Quanlu Zhang et al. "Retiarii: A Deep Learning Exploratory-Training Framework". In: *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*. 2020, pp. 919–936.

[22] Marc-André Zöller and Marco F. Huber. *Benchmark and Survey of Automated Machine Learning Frameworks*. 2021. arXiv: 1904.12054 [cs.LG].

[23] URL: https://github.com/minimaxir/automl-gs.

[24] URL: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html.

[25] URL: https://supervised.mljar.com/.