Towards Human-centric AutoML via Logic and Argumentation

Joseph Giovanelli[1], Giuseppe Pisano[1]

[1]*ALMA MATER STUDIORUM — Università di Bologna*

Abstract

In the last decade, we have witnessed an exponential growth in both the complexity and the number of Machine Learning (ML) techniques. As a consequence, leveraging such methods to solve real-case problems has become difficult for a Data Scientist (DS). Automated Machine Learning (AutoML) tools were devised to alleviate that task, but easily became as complex as the ML techniques themselves. The DS has started to rely on this kind of tools without understanding their functioning, thus loosing the control over the process.

In this vision paper, we propose HAMLET (Human-centric AutoMl via Logic and Argumentation), a framework that would help the DS to redeem her centrality. HAMLET is inspired to the well-known standard process model CRISP-DM. Iteration after iteration, the knowledge is augmented by acquiring more constraints about the problem until a suitable solution is found. HAMLET leverages Logic and Argumentation to merge both constraints and solutions in an uniformed human- and machine-readable medium. Not only it allows an easy exploration of the new knowledge at each iteration, but it also enforces a continuous revision via the AutoML tool and the confrontation between the DS and Domain Experts.

Keywords

AutoML, Logic, Argumentation, CRISP-DM, Data Scientist

1. Introduction

In relation to data platforms, it is well-known that Machine Learning (ML) plays a key role in the process of data analysis. As a matter of fact, it has been pervasively employed to cope with each and every type of real-case problems [1, 2, 3, 4]. The Data Scientist (DS) (i.e., a specialist of data analysis) starts by collecting raw data in an arbitrary format. Then she typically leverages a process model that will help her to translate the *knowledge* about the problem into *ML constraints*, and deploy the *solution*. CRISP-DM [5] is the most acknowledged standard process model and we will take it as a reference in the whole paper. A solution consists of a *ML pipeline*: a series of *Data Pre-processing transformations* and a *ML algorithm*. The DS can instantiate both with a large set of *techniques*, which have their own tunable hyper-parameters. These choices highly affect the performance of a solution.

Automated Machine Learning (AutoML) tools have been devised with the aim of assisting the DS during the ML pipeline instantiation. They leverage state-of-the-art optimisation approaches to smartly explore huge search spaces of solutions. AutoML has been demonstrated to provide accurate *performance*, even in a limited time budget. During the setting up of the search space, it is highly important to the DS to leverage the knowledge about the problem, considering all the ML constraints. Otherwise, it might lead the AutoML tool to retrieve *invalid solutions* (i.e., the *result* of those cannot be deemed *correct*). Besides, AutoML tools became that complex to make it difficult for the DS to understand their functioning, hence losing the control over the process. Researchers are aware of these problems [6]. There are some works that have prescribed to use a human-centric framework for AutoML [7, 8, 9], yet suggesting only design requirements. Alternatively, the authors in [10] have proposed a tool that visualises the best and the worst solutions retrieved by an AutoML tool.

We claim that the need of a human-centric framework for AutoML is real, and it is crucial for the DS to *augment* her knowledge via the retrieved solutions. At this purpose we propose HAMLET (Human-centric AutoMl via Logic and Argumentation), which leverages Logic and Argumentation to:

- structure the ML constraints and the AutoML solutions in a Logical Knowledge Base (LogicalKB);
- parse the structured LogicalKB into a human- and machine-readable medium called Problem Graph;
- leverage the Problem Graph to set up an AutoML search space;
- leverage the Problem Graph to allow both the DS and an AutoML tool to revise the current knowledge.

Figure 1 illustrates how CRISP-DM, AutoML, and HAMLET interact with each other. We remark that our framework allows the DS to never loose the control over the

Published in the Workshop Proceedings of the EDBT/ICDT 2022 Joint Conference (March 29-April 1, 2022), Edinburgh, UK
j.giovanelli@unibo.it (J. Giovanelli); g.pisano@unibo.it (G. Pisano)
0000-0002-0990-3893 (J. Giovanelli); 0000-0003-0230-8212 (G. Pisano)

process, and hence her centrality. Besides, HAMLET allows to visualise the knowledge in an human- and machine-readable format. As advocated in [11], the DS requires to understand the AutoML process in order to trust the proposed solutions.

The remain of the paper is structured as follows. Section 2 and Section 3 introduce the main notions of respectively AutoML and Argumentation. Section 4 illustrates our framework. Finally, Section 5 draws the conclusions and potential leveraging.

2. AutoML

AutoML tools have been conceived with the aim of lightening the DS in the overwhelming practise of finding the suitable solution for the case at hand. We recall that in the context of data platforms, a solution is a ML pipeline, defined as a series of Data Pre-processing transformations followed by a ML algorithm. In its early days, only the instantiation of the latter – the ML algorithm – was addressed. Auto-Weka [12] formalised the problem as Combined Algorithm Selection and Hyper-parameter Optimisation (CASH). In a nutshell, in order to find the most performing configuration, various ML algorithms – and related hyper-parameters – have to be tested over a dataset. Such a problem was successfully coped by leveraging Bayesian Optimisation (BO) [13], a sequential design strategy for global optimisation. The process involves several iterations, through which different configurations are explored. As the iterations advance, an increasingly accurate model is built on top of the previous explored configurations, with the aim of suggesting the most promising ones. The configurations keep being explored, and updating the model, until a budget in terms of either iterations or time is reached.

Recently, AutoML is no longer limited to optimise just the ML algorithm phase, but it includes Data Pre-processing as well. Indeed, with the aid of a series of transformations, it is possible to achieve better performance, unattainable with the most performing ML algorithm configuration [14]. In [15], the author formalised the problem as Data Pipeline Selection and Optimisation (DPSO). Each of the transformations can be instantiated with different techniques, which – analogously to the ML algorithms – have their own hyper-parameters. Auto-sklearn [16] includes Data Pre-processing already in its first versions. Yet, they fix the arrangement of the transformations a priori, without considering that the most performing arrangement changes according to the case and data at hand. Considering several arrangements translates into larger search spaces, not easy to explore.

In order to cope with ever larger research spaces, various expedients have been employed. Meta-learning (i.e., learning on top of learning) has been used to warm-start the Bayesian Optimisation (i.e., to boost the convergence process) by suggesting promising configurations (i.e., that worked well in previous similar real-case problems) [17]. Ensembling (i.e., construction of a high-performing solution combining several low-performing solutions; e.g., bagging, boosting, stacking) have been leveraged to enable AutoML tools to retrieve a solution that combines the best performing configurations, instead of retrieving just the best performing one [16]. Moreover, multi-fidelity methods (i.e., the use of several partial estimations to boost the time-consuming evaluation process) have been exploited to let AutoML tools explore as many configurations as possible.

All in all, the improvements made over the last years have yielded to be so substantial that AutoML is nowadays able to handle the entire ML pipeline instantiation. Yet, the stacking of complex mechanisms on top of each other unavoidably led to a less understanding of the process by the DS. We believe that the DS has the duty to revise and supervise the suggested solutions. Unfortunately, state-of-art AutoML tools overlook her role, and do not let that possible.

3. Logic & Argumentation

Logic can be defined as the abstract study of statements, sentences and deductive arguments [18]. From its birth, it has been developed and improved widely and now includes a variety of formalisms and technologies. Between all, Argumentation has proved itself an important tool for handling conflicting information (e.g., opinions, empirical data). This has led to a great number of researches trying to establish a computational model of logical arguments.

In Abstract Argumentation [19], a scenario can be represented by a directed graph. Each node represents an argument, and each edge denotes an attack by one argument to another. Each argument is regarded as atomic. There is no internal structure to an argument. Also, there is no specification of what is an argument or an attack. A graph can then be analysed to determine which arguments are acceptable according to some general criteria (i.e., semantics) [20].

A way to link Abstract Argumentation and logical formalisms has been advanced in the field of Structured Argumentation [21], where we assume a formal logical language for representing knowledge (i.e., a Logical Knowledge Base), and specifying how arguments and conflicts (i.e., attacks) can be derived from that knowledge. In the structured approach, premises and claims of the argument are made explicit, and the relationship between them is formally defined through rules internal to the formalism. We can build the notion of attack as a binary relation over structured arguments that de-

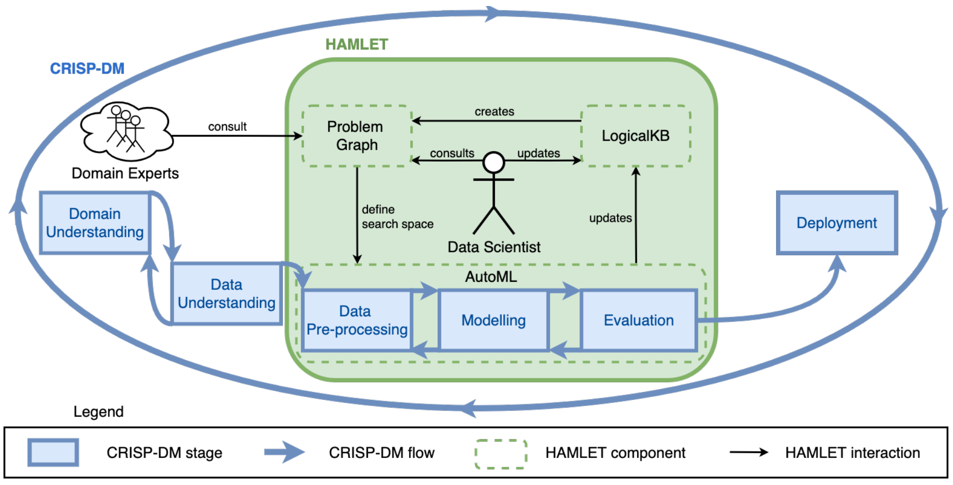


Figure 1: Integration of the HAMLET framework with the CRISP-DM standard process model and AutoML.

notes when one argument is in conflict with another (e.g., contradictory claims or premises). One of the main frameworks for Structured Argumentation is ASPIC$^+$ [22]. In this formalism arguments are built with two kinds of inference rules: strict rules, whose premises guarantee their conclusion, and defeasible rules, whose premises only create a presumption in favour of their conclusion. Then conflicts between arguments can arise from both *inconsistencies* in the Logical Knowledge Base and the defeasibility of the reasoning steps in an argument (i.e., a defeasible rule used in reaching a certain conclusion from a set of premises can also be attacked).

In our view, once defined the right logical language for encoding the DS and AutoML knowledge, a Structured Argumentation model (e.g., an ASPIC$^+$ instance [23]) would provide us with the formal machinery to build an Argumentation framework upon the data, while Abstract Argumentation would dispense the evaluation tools.

4. Towards a human-centric approach

Addressing ML problems encompasses the DS seeking for a solution, considering all the constraints of the case. She usually leverages a process model as CRISP-DM. The DS starts by collecting raw data in an arbitrary format. Then, in the first stage, Domain Understanding is conducted. The DS works in a close cooperation with Domain Experts, and enlists *domain-related constraints* (i.e., intrinsic of the problem). Follows Data Understanding, devoted to data analysis, and with the aim of extracting *data-related constraints* (i.e., defined by the data format). Domain and Data Understanding might be repeated many times, until the DS is satisfied by the acquired knowledge. Once she feels confident, she begins to investigate different solutions throughout the next stages: Data Pre-processing, Modelling, and Evaluation. Data Pre-processing and Modelling are conducted to effectively build the solution, while Evaluation offers a way to measure the performance of it. Finally, the process concludes with the Deployment stage (i.e., the actual implementation of the solution).

We recall that building a solution consists of instantiating a ML pipeline: a series of transformations – defined in the Data Pre-processing stage – and a ML algorithm—defined in the Modelling stage. Seeking the most correct and performing solution, the DS should consider the already known constraints – domain- and data-related – and some new she discovers in the Data Pre-processing and Modelling, respectively: *transformation- and algorithm-related constraints* (i.e., due to the intrinsic semantic of transformations and algorithms at hand).

Throughout the different stages, the DS acquires knowledge from different points of view (i.e., domain-, data-, transformation-, and algorithm-related). Besides, as illustrated in Figure 1, CRISP-DM might be iterated many times. The several *iterations* of the process aim at augmenting such a knowledge about the problem. Finally, the process is ruled by interactions between the DS and Domain Experts, discussing and arguing on both constraints and solutions.

4.1. AutoML and CRISP-DM

As described in Section 2, AutoML helps in finding a suitable ML pipeline instantiation (i.e., automatisation of

Data Pre-processing, Modelling, and Evaluation stages). However, such an automatisation unavoidably leads to a less overall understanding (i.e., the knowledge about the problem cannot be properly augmented throughout the process).

The definition of the search space has a huge impact on the correctness and performance of the solutions. The DS collects constraints to guarantee the correctness of the solution, anticipating the effect of each of them, and finally defining the search space.

EXAMPLE 1. Let us consider two transformations, namely Discretisation ($\mathcal{D}$) and Normalisation ($\mathcal{N}$), and a ML algorithm as Decision Tree ($\mathcal{DT}$). Based on the implementation, a possible algorithm-related constraint may be "require $\mathcal{D}$ when applying $\mathcal{DT}$". Accordingly, we consider a transformation-related constraint "no $\mathcal{N}$ in pipelines with $\mathcal{D}$". This leads to discard ML pipelines that contain $\mathcal{D}, \mathcal{N}$, and $\mathcal{DT}$:

$$\cdots \to \mathcal{N} \to \cdots \to \mathcal{D} \to \cdots \to \mathcal{DT}$$
$$\cdots \to \mathcal{D} \to \cdots \to \mathcal{N} \to \cdots \to \mathcal{DT}$$

In real-case problems, consider all the possible effects is overwhelming, and *inconsistencies* might occur. The problem exacerbates when it comes to cross-cutting issues, such as those related to *ethical* and *legal* fields. For instance, topics like racism and gender equality have to be treated separately, otherwise they could lead to social repercussions. As it is well-know, the authors of the boston-house dataset [24] engineered a feature assuming that racial self-segregation had a positive impact on house prices. A way of addressing such an issue is to encode some kind of ethical constraint (e.g., dropping that particular feature from the data). Furthermore, the ML result is expected to be compliant to the laws of the involved countries. To the best of our knowledge there is no attempt to properly treat such ML constraints, and hence ease the search space definition. Most of the tools are not customisable (i.e., weak-constrained search spaces, e.g., Auto-Weka, [12] Auto-Sklearn [16]), and others are far too permissive (i.e., no assistance at all; e.g., HyperOpt [25]). AutoML is not clear enough to provide the DS with a feedback that would help to augment her knowledge about the problem. We claim that a human-centric framework should provide the mechanisms to: *i)* help the DS to structure her knowledge about the problem in an effective search space; *ii)* augment the knowledge initially possessed by the DS with the one produced by the AutoML optimisation process.

4.2. The role of logic

The two identified requisites share a common need: encoding both the DS knowledge about the problem and the outcome of the AutoML tool in a uniform format. As a result, it would be possible to use the DS knowledge as an input for the optimisation process—search space definition. Then, this initial knowledge can be augmented with the possible solutions provided by an AutoML tool. These possible solutions can be exploited to derive new constraints (i.e., the awareness about the problem increases). We see the augmented knowledge as an awareness determined by an increased expertise on the correct constraints. The finding of such correct constraints leads to the finding of the correct solution—if exists. In other words, at each CRISP-DM iteration, the knowledge is encoded into the AutoML tool, which provides a feedback (i.e., augmented knowledge) in the same format.

Logic could be the key element in defining a common structure (i.e., a uniformed human- and machine-readable medium) on which the knowledge of both the DS and the AutoML tool can be combined fruitfully. In a way, our approach follows the steps of the well known logical based expert systems, of which it is possible to find a great number of successful examples [26]. In literature, it is also possible to find two well-known issues [27]: lack of scalability and difficulties in the definition of a sound knowledge base that encodes all the required pieces of information. Yet, we believe they do not affect our model. As to the former, the amount of the acquired knowledge (i.e. the problem constraints) through CRISP-DM iterations is not enough to label such a problem as a *big data problem*, and hence scalability should not be an issue. As to the latter, we believe that the analysis process would only benefit from the clearness given by such a structured investigation.

Logic would also provide the tools to cope with one of the distinctive features of the knowledge we want to deal with: the possible inconsistency. Indeed, the ML process is the product of possible attempts, validated or refuted by a consequent evaluation. Hence, the mechanism used to encode the knowledge is required to manage this constant revision process. This is the role of Argumentation—one of the main approaches for dealing with inconsistent knowledge and defeasible reasoning.

4.3. HAMLET

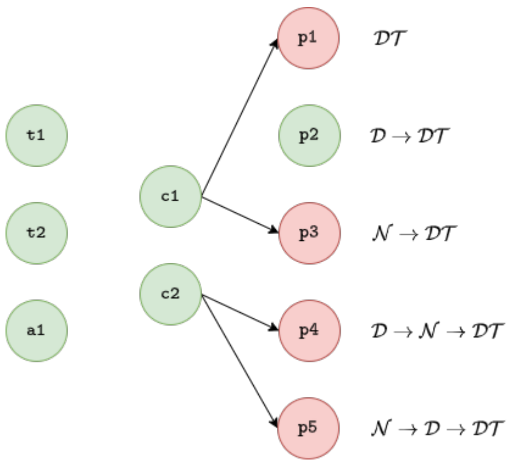
In the last paragraphs we identified two main requirements for a human-centric framework (i.e., structure the DS knowledge in a well-defined AutoML search space, and provide the solutions in accordance with the input knowledge). We also introduced Computational Logic – Argumentation in particular – as the main tool in our investigation. Let us now delve into details of how these pieces converge in our framework.

Figure 1 illustrates a scheme of HAMLET. The DS conducts the stages from Domain & Data Understanding to

Listing 1: Example of a LogicalKB using a logical formalism.

```
t1  :=>  transformation(discretisation).
t2  :=>  transformation(normalisation).
a1  :=>  algorithm(decision_tree).

c1  :=>  mandatory_transformation_for_algorithm([discretisation], decision_tree).
c2  :=>  invalid_transformation_set([normalisation, discretisation]).
```



t1
t2
a1
c1
c2
p1
p2
p3
p4
p5
$\mathcal{DT}$
$\mathcal{D} \to \mathcal{DT}$
$\mathcal{N} \to \mathcal{DT}$
$\mathcal{D} \to \mathcal{N} \to \mathcal{DT}$
$\mathcal{N} \to \mathcal{D} \to \mathcal{DT}$

Figure 2: Example of a Problem Graph. Green nodes are valid arguments, red ones are refuted.

Data Pre-processing & Modelling, and thus gathers all the constraints that represent the knowledge discovered so far. The Logical Knowledge Base (LogicalKB) provides a vehicle to encode such constraints. In particular, the DS leverages an intuitive logical language, and enlists the constraints one-by-one. In Section 3 we introduced the notion of Structured Argumentation as a formal tool to convert elements from a logical language into an Argumentation graph. Implementing and exploiting such a Structured Argumentation tool, HAMLET proceeds to resolve conflicts in the LogicalKB: the logical-encoded knowledge is transformed in a Problem Graph.

The benefit of the Problem Graph is two-fold. First of all, it can be leveraged by both the DS and Domain Experts to understand and summarise the current knowledge. Second of all, thanks to its nature, it is straightforward to convert such a graph of constraints into a space of possible solutions (i.e., exploiting Argumentation semantics, it is easy to obtain all the sets of arguments – constraints – which hold together). As a matter of fact, this feature would relieve the DS of the burden of manually considering all the effects of the possible constraints. It is important to notice that, although the increased degree of automatisation, the Problem Graph allows the DS and Domain Experts to correct, revise, and supervise the process. Accordingly, possible inconsistencies – due to diverging constraints – can be verified by the DS using her knowledge.

Once the knowledge has been accurately revised, an AutoML tool is leveraged to automatise the ML pipeline instantiation. Throughout the exploration, different solutions are tested, which contribute to augment the global knowledge about the problem. Accordingly, some of the originally encoded knowledge by the DS and Domain Experts might be refuted or found inconsistent. HAMLET is designed to enable a transparent augmentation of the knowledge in the Problem Graph according to the newfound solutions. The updating procedure is the same as the one employed by the DS during the constraint encoding phase. Specifically, the AutoML solutions are automatically transposed to our logical language in the form of new constraints, and then added to the LogicalKB. Of course, a change in the LogicalKB translates in a change in the Problem Graph, allowing the DS and Domain Experts to visualise and *argue* about it. The revision of the Graph is the key element in the process of augmenting the knowledge: the DS and Domain Experts can consult each other and discuss how the new insights relate with their initial knowledge. Indeed, thanks to the nature of the Problem Graph, it would be extremely easy to identify new possible conflicts and supporting arguments. Consequently, new constraints can be derived.

EXAMPLE 2. In Example 1 we introduce two possible ML constraints. We now provide their encoding in the LogicalKB, and the resulting Problem Graph. For the sake of clarity, we focus only on Discretisation ($\mathcal{D}$) and Normalisation ($\mathcal{N}$) as transformations, and Decision Tree ($\mathcal{DT}$) as the ML algorithm. Listing 1 contains the LogicalKB expressed in a logic language: `t1` and `t2` represent $\mathcal{D}$ and $\mathcal{N}$ respectively, `a1` represents $\mathcal{DT}$. We consider the algorithms-related constraint `c1`, namely "require $\mathcal{D}$ when applying $\mathcal{DT}$", and the trnasformation-related constraint `c2`, that is "no $\mathcal{N}$ in pipelines with $\mathcal{D}$". This LogicalKB is used to generate the Problem Graph shown in Figure 2, nodes represent arguments and edges represent attacks among them. There are five possible ML pipelines: $\mathcal{DT}$ (p1), $\mathcal{D} \to \mathcal{DT}$ (p2), $\mathcal{N} \to \mathcal{DT}$ (p3), $\mathcal{D} \to \mathcal{N} \to \mathcal{DT}$

(p4), $\mathcal{N} \rightarrow \mathcal{D} \rightarrow \mathcal{DT}$ (p5). With no constraints, we cannot discard any ML pipeline (i.e., there are no incompatibilities between the arguments). By introducing c1, attacks against p1 and p3 are generated (both pipelines contain $\mathcal{DT}$ but not $\mathcal{D}$). By introducing c2, attacks against p4 and p5 are generated (both pipelines contain $\mathcal{D}$ and $\mathcal{N}$). We can leverage a standard argumentation semantics (e.g., Dung's grounded semantics [19]) to evaluate the graph. In our case, all the arguments with no attacks are admissible. Among them, we retrieve the ones representing pipelines. p2 is the only valid pipeline, and it will be used to generate the AutoML search space.

Example 2 illustrates how HAMLET leverages Logic and Argumentation to handle the DS knowledge. The proposed logic formalism allows to easily encode the different ML constraints into a LogicalKB. We highlight that the Problem Graph generation is handled by an argumentation engine, which is available in the Supplementary Material [1]. The use of the Problem Graph allows to prune the considered ML pipeline for the AutoML search space. AutoML could update the Problem Graph by extracting constraints from the performed exploration, and transposing them into the LogicalKB. For instance, the DS may not have considered that data at hand contain missing values. AutoML could help in identifying transformation-related constraints such as: "require Imputation ($\mathcal{I}$) in all the pipelines". The resulting constraints might be in conflict with the previous knowledge. In our vision, the DS is able to visualise such inconsistencies through the Problem Graph, and resolve them.

We remark how our framework is compliant with the iterative nature of the CRISP-DM standard process model. This aspect is crucial when trying to solve real-case problems through the use of modern data platforms. Indeed, not only the different CRISP-DM stages can be executed several times, but the whole process can be iterated, bringing new information about the problem. We claim that our framework support and ease the adoption of the described resolution process model, by providing a tool that is both human- and machine-readable. The knowledge can be automatically handled throughout iterations, supporting the DS in the whole analysis, in a continuous revision of the problem constraints. At each iteration, a portion of the knowledge is known and other is discovered. Its integration into a unified augmented knowledge graph allows to: *i)* derive new constraints from the discovered knowledge, *ii)* jgcseamlessly visualise possible inconsistencies and conflicts. This naturally leads to a new iteration based on the new augmented knowledge. Besides, the entire process might be boosted with the aid of an external knowledge. In our vision, the DS community could create a shared LogicalKB derived from the available literature and similar real-case problems.

[1] https://queueinc.github.io/HAMLET-DATAPLAT2022/

5. Conclusions and potential leveraging

The increasing complexity in the state-of-the-art AutoML tools has led the DS to lose the control over the resolution process. We believe that human awareness about all the constraints and possible solutions of a ML problem is a fundamental aspect to consider, and consequently should play a key role in the design of next-generation data platforms. Accordingly, in this vision paper we present HAMLET, a human-centric AutoML framework based on Logic and Structured Argumentation. Logic is exploited to give a structure to the knowledge that the DS has to consider while deploying a solution. The advantage of such a choice is twofold. First of all, the logical encoding of the knowledge allows an easy exploration and verification of all the constraints that may apply to the case at hand—it is overwhelming for the DS to correctly handle the vast amount of them. Second of all, it provides a medium that is both human- and machine- readable. The DS and Domain experts can revise the knowledge, as well as the AutoML tool, thus creating a constant feedback cycle. We further remark that our framework could be able to address a wide range of AutoML-related challenges. We already highlighted a few of them: the embodiment of both ethical and legal constraints, and the construction of a shared knowledge among the DS community.

The road for future expansions is straightforward: we plan to extend this work providing a sound formalisation of HAMLET, and then a working implementation. It will be then possible to effectively quantify the benefits of our framework and test its efficacy on real-case problems.

References

[1] L. Zhou, S. Pan, J. Wang, A. V. Vasilakos, Machine learning on big data: Opportunities and challenges, Neurocomputing 237 (2017) 350–361. doi:10.1016/j.neucom.2017.01.026.

[2] P. Agrawal, R. Arya, A. Bindal, S. Bhatia, A. Gagneja, J. Godlewski, Y. Low, T. Muss, M. M. Paliwal, S. Raman, V. Shah, B. Shen, L. Sugden, K. Zhao, M.-C. Wu, Data platform for machine learning, in: Proceedings of the 2019 International Conference on Management of Data, SIGMOD '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1803–1816. URL: https://doi.org/10.1145/3299869.3314050. doi:10.1145/3299869.3314050.

[3] M. Francia, E. Gallinucci, M. Golfarelli, A. G. Leoni, S. Rizzi, N. Santolini, Making data platforms smarter with MOSES, Future Gener. Comput.

Syst. 125 (2021) 299–313. doi:10.1016/j.future.2021.06.031.

[4] C. Forresi, M. Francia, E. Gallinucci, M. Golfarelli, Optimizing execution plans in a multistore, in: L. Bellatreche, M. Dumas, P. Karras, R. Matulevičius (Eds.), Advances in Databases and Information Systems, Springer International Publishing, Cham, 2021, pp. 136–151.

[5] R. Wirth, J. Hipp, Crisp-dm: Towards a standard process model for data mining, in: Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining, volume 1, Springer-Verlag London, UK, 2000.

[6] D. Xin, E. Y. Wu, D. J. L. Lee, N. Salehi, A. G. Parameswaran, Whither automl? understanding the role of automation in machine learning workflows, in: CHI '21: CHI Conference on Human Factors in Computing Systems, ACM, 2021, pp. 83:1–83:16. doi:10.1145/3411764.3445306.

[7] Y. Gil, J. Honaker, S. Gupta, Y. Ma, V. D'Orazio, D. Garijo, S. Gadewar, Q. Yang, N. Jahanshad, Towards human-guided machine learning, in: Proceedings of the 24th International Conference on Intelligent User Interfaces, 2019, pp. 614–624.

[8] D. J.-L. Lee, S. Macke, A human-in-the-loop perspective on automl: Milestones and the road ahead, IEEE Data Engineering Bulletin (2020).

[9] D. Wang, J. D. Weisz, M. Muller, P. Ram, W. Geyer, C. Dugan, Y. Tausczik, H. Samulowitz, A. Gray, Human-ai collaboration in data science: Exploring data scientists' perceptions of automated ai, Proceedings of the ACM on Human-Computer Interaction 3 (2019) 1–24.

[10] J. P. Ono, S. Castelo, R. Lopez, E. Bertini, J. Freire, C. T. Silva, Pipelineprofiler: A visual analytics tool for the exploration of automl pipelines, IEEE Transactions on Visualization and Computer Graphics 27 (2021) 390–400.

[11] J. Drozdal, J. Weisz, D. Wang, G. Dass, B. Yao, C. Zhao, M. Muller, L. Ju, H. Su, Trust in automl: exploring information needs for establishing trust in automated machine learning systems, in: Proceedings of the 25th International Conference on Intelligent User Interfaces, 2020, pp. 297–307.

[12] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, K. Leyton-Brown, Auto-weka: Automatic model selection and hyperparameter optimization in weka, in: Automated Machine Learning, Springer, Cham, 2019, pp. 81–95.

[13] P. I. Frazier, A tutorial on bayesian optimization, CoRR abs/1807.02811 (2018). URL: http://arxiv.org/abs/1807.02811. arXiv:1807.02811.

[14] J. Giovanelli, B. Bilalli, A. Abelló, Effective data pre-processing for automl, in: K. Stefanidis, P. Marcel (Eds.), Proceedings of the 23rd International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP), volume 2840 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 1–10.

[15] A. Quemy, Data pipeline selection and optimization., in: DOLAP, 2019.

[16] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, F. Hutter, Auto-sklearn: efficient and robust automated machine learning, in: Automated Machine Learning, Springer, Cham, 2019, pp. 113–134.

[17] J. Giovanelli, B. Bilalli, A. Abelló, Data preprocessing pipeline generation for autoetl, Information Systems (2021) 101957.

[18] L. C. Paulson, Computational logic: its origins and applications, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 474 (2018). doi:10.1098/rspa.2017.0872.

[19] P. M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial Intelligence 77 (1995) 321–358. doi:10.1016/0004-3702(94)00041-X.

[20] P. Baroni, M. Caminada, M. Giacomin, An introduction to argumentation semantics, Knowledge Engineering Review 26 (2011) 365–410. doi:10.1017/S0269888911000166.

[21] P. Besnard, A. J. García, A. Hunter, S. Modgil, H. Prakken, G. R. Simari, F. Toni, Introduction to structured argumentation, Argument & Computation 5 (2014) 1–4. doi:10.1080/19462166.2013.869764.

[22] S. Modgil, H. Prakken, The *ASPIC*$^{+}$ framework for structured argumentation: a tutorial, Argument & Computation 5 (2014) 31–62. doi:10.1080/19462166.2013.869766.

[23] R. Calegari, G. Pisano, A. Omicini, G. Sartor, Arg2P: An argumentation framework for explainable intelligent systems, Journal of Logic and Computation (2021). doi:10.1093/logcom/exab089.

[24] D. Harrison, D. Rubinfeld, Hedonic housing prices and the demand for clean air, Journal of Environmental Economics and Management 5 (1978) 81–102. doi:10.1016/0095-0696(78)90006-2.

[25] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, D. D. Cox, Hyperopt: a python library for model selection and hyperparameter optimization, Computational Science & Discovery 8 (2015) 014008.

[26] H. Tan, A brief history and technical review of the expert system research, IOP Conference Series: Materials Science and Engineering 242 (2017). doi:10.1088/1757-899X/242/1/012111.

[27] P. K. Coats, Why expert systems fail, Financial Management 17 (1988) 77–86. URL: http://www.jstor.org/stable/3666074.