An Ontology-Based Concept for Meta AutoML

Bernhard G. Humm, Alexander Zender

An Ontology-Based Concept for Meta AutoML

Bernhard G. Humm[1 [0000−0001−7805−1981]] and Alexander Zender[1 [0000−0002−6956−9049]]

Hochschule Darmstadt - University of Applied Sciences, Haardtring 100, 64295 Darmstadt, Germany
{bernhard.humm,alexander.zender}@h-da.de

Abstract. Automated machine learning (AutoML) supports ML engineers and data scientists by automating tasks like model selection and hyperparameter optimization. A number of AutoML solutions have been developed, open-source and commercial. We propose a concept called *OMA-ML* (Ontology-based Meta AutoML) that combines the strengths of existing AutoML solutions by integrating them (meta AutoML).
OMA-ML is based on a ML ontology that guides the meta AutoML process. It supports multiple user groups, with and without programming skills. By combining the strengths of AutoML solutions, it supports any number of ML tasks and ML libraries.

1 Introduction

Machine learning (ML) is an important sub-domain of artificial intelligence (AI), allowing to make predictions using models based on previous observations [22]. Engineering ML applications for practical use requires sound experience of ML engineers respectively data scientists. Tasks to be performed include data analysis, data preparation, feature engineering, model selection, validation, learning curve analysis and hyperparameter optimization. To support data scientists and also enable domain experts to create ML pipelines, the field of *automated ML (AutoML)* [28] has emerged. AutoML aims at automating model selection and hyperparameter optimization, leading to higher efficiency and, potentially, better results. More progressive AutoML solutions also perform data preparation, feature engineering, and validation, allowing to create entire ML pipelines automatically [15].

Currently, AutoML is focused on supervised ML [28]. There is a growing number of AutoML solutions available, both academic as well as commercial. Current state-of-the-art AutoML solutions target one concrete ML library and compute a ML pipeline for this library only, e.g., Autosklearn [5] for Scikit-learn [21], Auto-Keras [12] for Keras [3], and Google AutoML [8] for Tensorflow [1].

To the best of our knowledge, no existing AutoML solution combines multiple ML libraries.

The targeted user groups of AutoML solutions differ. Commercial solutions like RapidMiner Auto Model [14] or Google AutoML [8] offer a graphical user interface (GUI) usable for domain experts without programming skills (e.g.,

biologists), providing a workflow and deployment inside their ecosystem. Auto-WEKA [13] is an open source solution which also provides a GUI. Other open source solutions like Autosklearn [5], Auto-Keras [12], Auto-PyTorch [18], or TPOT [15] offer libraries that require programming skills.

This paper presents a concept for meta AutoML which supports multiple ML libraries by combining the strengths of several AutoML solutions. This concept is based on a ML ontology which guides meta AutoML. We call this method *OMA-ML* (Ontology-based Meta AutoML); the implementation is ongoing work in progress.

This paper is structured as follows. Section 2 presents related work.In Section 3, we introduce the basics of AutoML. Section 4 is the core of the paper, introducing meta AutoML and OMA-ML. Section 5 concludes the paper and indicates future work.

2 Related Work

The concept of meta AutoML is novel. We are only aware of one article [27] which presents a similar concept called *Ensemble Squared*, but this preprint is not a peer-reviewed publication. Like OMA-ML, Ensemble Squared uses third-party AutoML solutions which are invoked in parallel. Insofar, both OMA-ML and Ensemble Squared are meta AutoML approaches. A difference of our approach is the use of the ML ontology to guide various components of OMA-ML: the GUI for entering the AutoML configuration, dataset analysis and pre-processing, and the controller component of OMA-ML. We see considerable benefits in this approach regarding extensibility. Adding new AutoML solutions to OMA-ML can largely be accomplished by extending the ML ontology and implementing new adapters.

3 Basics of AutoML

3.1 Input and Output

Fig. 1 shows the inputs and outputs of AutoML as a BPMN diagram (Business Process Model and Notation [11]).

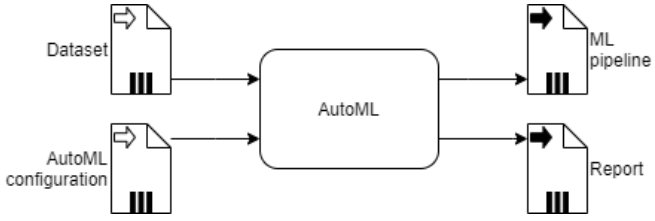

Fig. 1. AutoML input and output

AutoML requires the following input:

1. **Dataset**: the dataset for the ML task, e.g., a CSV file for classification on tabular data;
2. **AutoML configuration**:
   (a) ML task, e.g.,
      i. Classification or regression on tabular data;
      ii. Classification or regression on image and video data;
      iii. Classification or regression on text data;
   (b) ML target: e.g., label column in classification or regression tasks;
   (c) Additional configuration parameters (optional): e.g. maximum run-time, model performance or hardware restrictions.

AutoML produces the following output:

1. **ML pipeline**: The ML pipeline generated by AutoML is a piece of source code which can be executed to perform the ML task specified.
2. **Report**: A textual or graphical explanation of the AutoML result, including a listing of ML configurations and their respective performance measures.

AutoML solutions generate ML pipelines. A ML pipeline implements data preparation (e.g., feature selection, encoding or missing values imputation) the selected ML approach and its hyperparameter configuration [28].

AutoML solves the *Combined Algorithm Selection and Hyperparameter optimization (CASH) problem* [26]. Algorithms that solve the CASH problem search for the best ML approach and hyperparameter setting for a given ML task [28].

Different AutoML solutions use different algorithms to solve the CASH problem, e.g.:

1. Auto-Sklearn: SMAC [6] [10];
2. TPOT: Evolutionary algorithm [15];
3. H2O AutoML: Grid Search [16];
4. ATM: A combination of multi-armed bandit learning with Gaussian processes [24].

3.2 Example: Auto-sklearn

One of the more popular AutoML solutions by citations is Auto-sklearn [6]. It offers pipeline generation for classification and regression of tabular data.

Listing 1.1 displays a simple Auto-sklearn implementation. In Auto-sklearn, the ML tasks is specified by the class used, e.g., `AutoSklearnClassifier` for classification of tabular datasets. The AutoML process is triggered by executing the `fit` function.

Code Listing 1.1. Auto-sklearn example

```
import autosklearn.classification
cls = autosklearn.classification.AutoSklearnClassifier()
cls.fit(X_train, y_train)
predictions = cls.predict(X_test)
```

Without custom parameterization, Auto-sklearn will use its default configuration. Advanced users may customize the Auto-sklearn process [17] with a multitude of parameters, e.g.

1. Hardware usage, e.g., `memory_limit`;
2. Pipeline size or generation constrains, e.g., `ensemble_size`;
3. Pipeline scoring/metrics, e.g., `metric`;
4. Pre-processing constrains, e.g., `exclude_preprocessors`;
5. Runtime constraints, e.g., `time_left_for_this_task`;
6. Meta configuration (logging, save folder location, etc.), e.g., `output_folder`;

The Auto-sklearn result is a pipeline that can be used to make predictions using the `predict` function (see Listing 1.1). The `sprint_statistics` function display statistics about the found ML pipelines [17]:

1. Dataset name;
2. Metric used;
3. Best validation score;
4. Number of target algorithm runs;
5. Number of successful target algorithm runs;
6. Number of crashed target algorithm runs;
7. Number of target algorithm runs that exceeded the memory limit;
8. Number of target algorithm runs that exceeded the time limit.

3.3 Discussion of Existing AutoML Solutions

AutoML solutions are implemented on top of a specific ML library. They produce pipelines using software from this ML library that can be exported and imported into this ML library. Deciding on an AutoML solution results in a technology lock-in for the corresponding ML library. Comparing the performance between different ML libraries is not possible.

ONNX [25] is an open format for artificial neural networks (ANN) to enable interoperability between ML libraries. However, not every ML library supports ONNX. Furthermore ONNX does not support other ML model types besides ANN.

AutoML solutions target specific user groups. Most open source AutoML solutions target users with programming skills, e.g. in Python. Commercial AutoML solutions provide a GUI which also address users without programming skills, e.g., domain experts.

All existing AutoML solutions have their individual strengths. They all solve the CASH problem, support specific ML tasks, target specific user groups and generate ML pipelines for specific ML libraries. Meta AutoML allows combining the strengths of individual AutoML solutions, while alleviating their limitations: it may support various ML tasks, support various user groups and is technology-independent.

In the next section, we introduce OMA-ML, our concept for a meta AutoML. First, we present the goals we aspire for OMA-ML.

4 OMA-ML: An Ontology-based Concept for Meta AutoML

4.1 Goals for OMA-ML

By combining the strengths of individual AutoML solutions, we pursue the following goals for OMA-ML:

1. **AutoML**: OMA-ML shall perform AutoML, i.e., generate an executable ML pipeline and a report based on a configuration and a dataset.
2. **User Groups**: OMA-ML shall target user groups with and without programming skills. It shall provide a GUI which allows intuitive configuration of AutoML and interactive reporting. Additionally, it shall provide an API.
3. **Technology-independent**: OMA-ML shall support any number of ML libraries.
4. **ML tasks**: A wide range of ML tasks shall be supported.

4.2 Meta AutoML

Fig. 2 shows the concept of meta AutoML as a BPMN diagram.

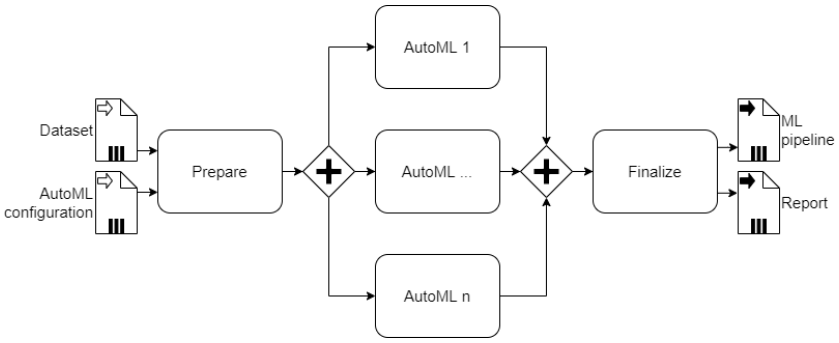

Fig. 2. Meta AutoML workflow

Similar to other AutoML solutions, the user enters the required input (dataset and AutoML configuration). The meta AutoML prepares various AutoML solutions to be executed in parallel. The results of the AutoML solutions are collected and the results of meta AutoML (ML pipeline and report) are finalized.

4.3 ML Ontology

An ontology is a formal, explicit specification of a shared conceptualization of a problem domain [23]. We are in the process of developing an ontology for the domain of ML [9]. One of the use cases of this ML ontology is to guide the meta

AutoML process. The ML ontology is modelled in RDF [4] using SKOS [20]. It currently consists of about 500 RDF triples, specifying 63 ML approaches, 15 ML tasks, 22 prediction performance measures, 9 AutoML solutions, 7 ML libraries, their characteristics and interrelationships, and more.

Fig. 3 shows some exemplary concepts of the ML ontology and their relationships.

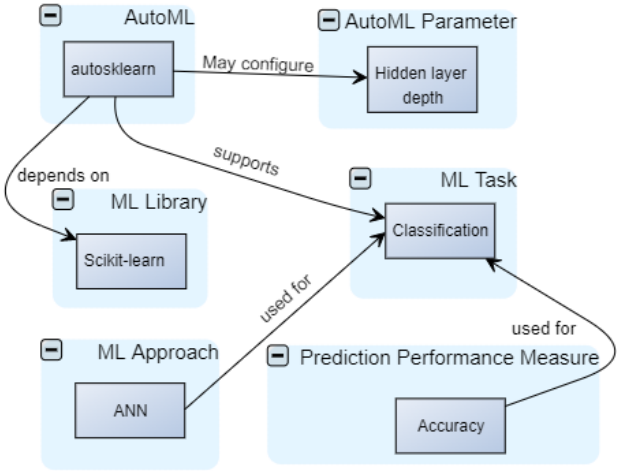

Fig. 3. Exemplary concepts the ML ontology

In this exemplary subset of the ML ontology it is expressed that the ML approach *ANN* can be used for *classification* tasks. *autosklearn* is an AutoML solution performing classification tasks. Autosklearn depends on the ML library *Sikit-learn* and allows to configure the *Hidden layer depth*. *Accuracy* can be used as a prediction performance measure for classification tasks.

The ML ontology is the information backbone of OMA-ML and is used in several components of OMA-ML, as shown in the next section.

4.4 OMA-ML Software Architecture

Figure 4 shows the software architecture of OMA-ML as a UML (Unified Modeling Language) component diagram.

OMA-ML is designed as a 3-layer-architecture.

1. **Presentation layer**: This is the user interface of OMA-ML. A GUI allows interaction and visualization. An ontology-guided wizard supports configuring OMA-ML. Additionally, an API provides batch access to OMA-ML.

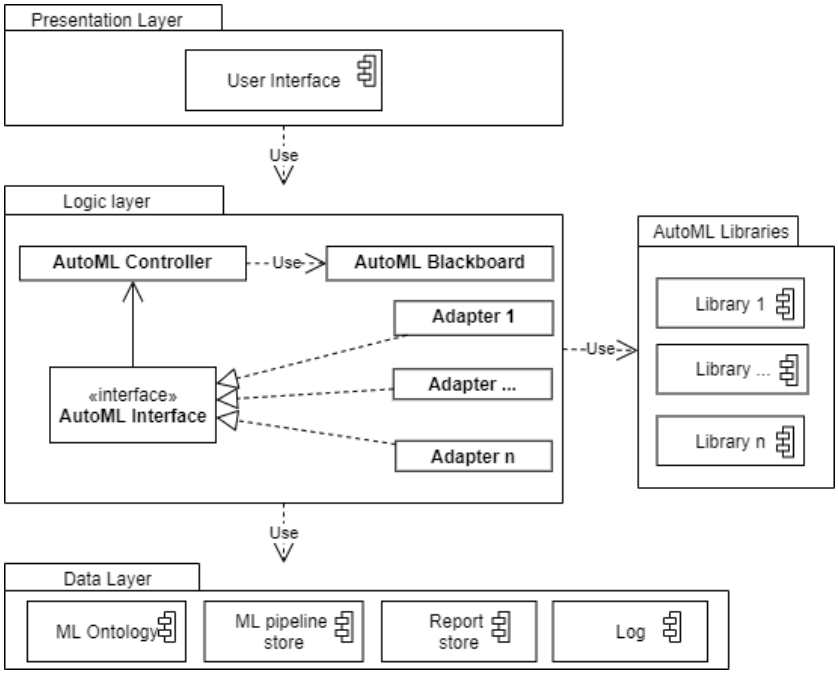

Fig. 4. OMA-ML software architecture

2. **Logic Layer**: This implements the control logic of OMA-ML, designed as a blackboard architecture [2]. The OMA-ML controller invokes individual AutoML libraries via the adapter pattern [7], thus providing a plug-in architecture for multiple AutoML solutions.
3. **Data Layer**: This layer provides access to the ML ontology (read access), the ML model store and AutoML logs (write access).

4.5 User Interface

In the GUI, a wizard guides the user to enter mandatory and optional AutoML configuration parameters. The wizard is based on the ML ontology, providing plausible configuration options only. For example, if the user selects AutoML solutions that produce ANN pipelines only, the wizard will only display configuration options for ANN.

Configuration parameters are as follows:

1. **Mandatory**:
   (a) **Dataset**: The dataset with labeled training data;
   (b) **ML Task**: The task the user wants to perform on the dataset, e.g., classification on tabular data (options from the ML ontology);
   (c) **ML target**: The name of the label column in the dataset.

2. **Optional**:
   (a) **Dataset schema**: Schema information on dataset columns including data types (e.g., int, float, string, date) and categories (e.g., numerical, categorical, textual) (options from the ML ontology);
   (b) **Scoring**: The prediction performance measure to be used as optimization target, e.g., accuracy (options from the ML ontology);
   (c) **AutoML solutions**: Usage restriction on particular AutoML solutions or ML libraries, e.g., autosklearn (options from the ML ontology);
   (d) **ML Model constraints**: Restrictions on ML approaches and custom configuration of ML approaches, e.g., ANN with maximum 10 hidden layers (options from the ML ontology);
   (e) **AutoML runtime constraints**: General meta AutoML constraints (monetary, time, hardware restriction) to influence the execution time, e.g., runtime limit 1 hour (options from the ML ontology);
   (f) **Training Type**: Training strategy for meta AutoML, e.g. use a subset of the dataset (options from the ML ontology);

After starting the OMA-ML process, the user interface is updated regularly with the current status of the AutoML processes which are executed in parallel. After termination of the OMA-ML process, the following output is provided:

1. **ML pipeline**: The user can download the successfully generated ML pipelines as a Python script and files specifying the pipeline structure. The Python scripts provide the following functionality:
   (a) Import the file specifying the pipeline structure;
   (b) Make predictions for a new, unlabeled dataset;
   (c) Save the prediction result.
2. **Report**:
   (a) Description of the used AutoML solutions, their produced ML pipelines and respective performance evaluations;
   (b) ML Pipeline leader board with scores.

When using OMA-ML in batch mode, the configuration file, including a link to the dataset can be passed to an API. The runtime state and output can be pulled from the API. Like in the online mode, the output consists of ML pipelines and reports.

4.6 OMA-ML Control Logic

The OMA-ML control logic is designed using the Blackboard pattern. Fig. 5 shows an overview of the OMA-ML control logic as a BPMN diagram.

When a new run of OMA-ML is triggered, the dataset is first analyzed, extracting the following metadata:

1. Number of rows and columns;
2. Data types of columns;
3. Missing values.

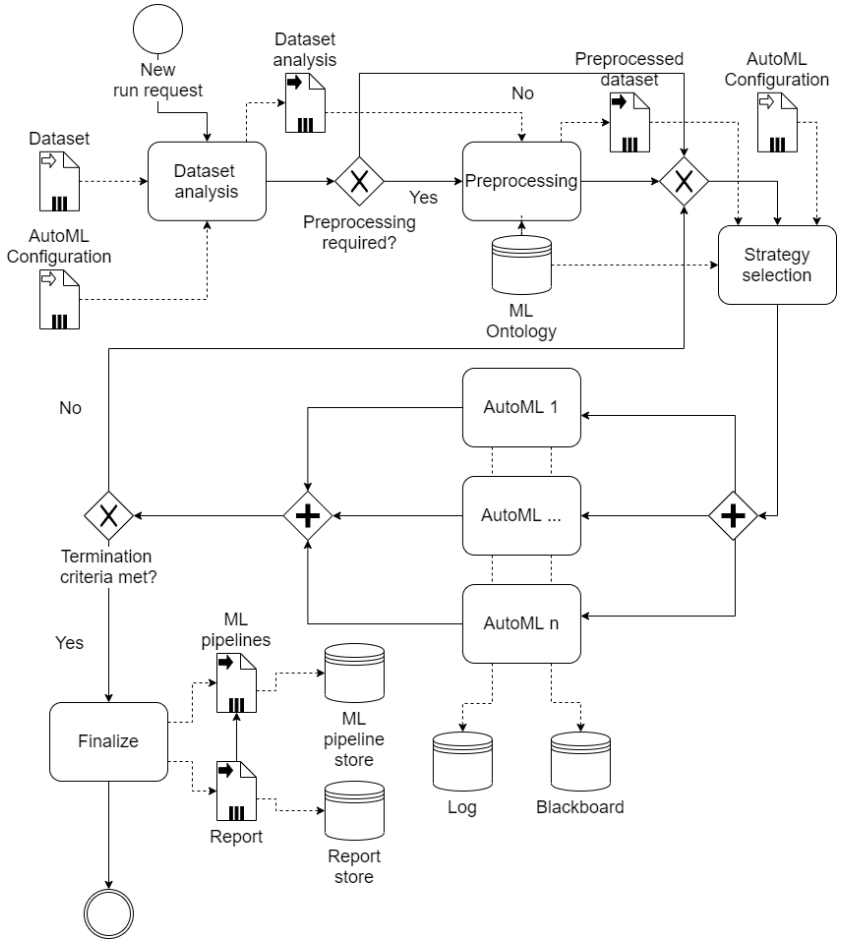

Fig. 5. OMA-ML Control Logic

Those metadata are needed for deciding whether pre-processing the dataset is necessary for individual AutoML solutions. The OMA-ML strategy selection is based on the ML ontology, taking into account the configuration and the dataset analysis result. It selects AutoML solutions which perform the ML tasks specified in the configuration. The dataset is pre-processed if needed. For example, if an AutoML solution requires numeric features only, but the dataset contains textual features, then the textual features are encoded. If the dataset is very large (e.g., 100 Mio. rows) and a small runtime limit is specified (e.g., 1 hour), then approaches with fast training times are selected or the dataset is downsized.

The selected AutoML solutions are triggered via their adapters in parallel by the OMA-ML controller. While executing AutoML, they continuously report their progress to the blackboard. The OMA-ML controller monitors the black-

board. After reaching the termination criteria (e.g., required accuracy is met, or run time limit is reached), the OMA-ML controller finalizes the OMA-ML run, saving the best performing executable ML pipelines to the ML pipeline store, generating a report, and storing it in the report store. Otherwise, the strategy may be altered, or alternative AutoML solutions may be triggered.

4.7 Logging

All OMA-ML runs are logged in a structured format, including the following data:

1. AutoML configuration;
2. Dataset analysis result;
3. OMA-ML strategy;
4. Hardware configuration (kernels, memory, processor, etc.);
5. AutoML actual run time (time spent);
6. Generated ML pipelines characteristics (accuracy, size, etc.).

With many OMA-ML runs, we expect the log data to be a valuable source of information. Data mining techniques may be used to gain insights to improve the OMA-ML controller's strategy selection. Using this log data additionally for supervised ML in the OMA-ML controller is subject to future work.

5 Conclusions and Future Work

In this paper, we have presented OMA-ML, a novel ontology-based concept for meta AutoML. OMA-ML combines the strengths of different AutoML solutions. It supports multiple ML libraries by employing multiple AutoML solutions via a plug-in architecture. This supports the extensibility of OMA-ML regarding future third-party AutoML developments. It supports multiple user groups, with and without programming skills. By combining the strengths of several AutoML solutions, it supports a wide range of ML tasks and ML libraries.

The implementation of OMA-ML is ongoing work in progress. OMA-ML is being implemented open-source as a micro-service architecture. It will be usable as a cloud service without installation. The GUI is implemented using the Blazor Framework [19]. The OMA-ML controller is implemented in Python. The OMA-ML batch access will be available via web services. Additionally, OMA-ML can be installed locally. The batch access is additionally available via a REST API. Individual AutoML libraries are installed in Docker containers and can be invoked in parallel.

As soon as the first OMA-ML release is available, we plan to thoroughly analyze it regarding usability and generated ML pipelines quality.

Future work includes the use of supervised ML on OMA-ML log data to improve the strategy selection in the OMA-ML controller. Furthermore, we plan to use learning curve analysis in the OMA-ML controller.

Acknowledgement

This work is funded by the German federal ministry of education and research (BMBF) in the program Zukunft der Wertschöpfung (funding code 02L19C157), and supported by Projektträger Karlsruhe (PTKA). The responsibility for the content of this publication lies with the authors.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems, https://arxiv.org/pdf/1603.04467
2. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture, A System of Patterns. Wiley Software Patterns Series, Wiley, s.l., 1. aufl. edn. (2013)
3. Chollet, F.: Keras (25032021), https://keras.io
4. Cyganiak, R., Wood, D., Lanthaler, M.: Rdf 1.1 concepts and abstract syntax (26032021), https://www.w3.org/TR/rdf11-concepts/
5. Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., Hutter, F.: Auto-sklearn 2.0: The next generation, https://arxiv.org/pdf/2007.04074
6. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M., Hutter, F. (eds.): Efficient and Robust Automated Machine Learning. MIT Press (2015). https://doi.org/10.5555/2969442.2969547
7. Gamma, E.: Design patterns: Elements of reusable object-oriented software. Addison-Wesley professional computing series, Addison-Wesley, Boston, 39. printing edn. (2011)
8. Google Cloud: Cloud automl (14012021), https://cloud.google.com/automl?hl=de
9. Humm, B.G., Bense, H., Fuchs, M., Gernhardt, B., Hemmje, M., Hoppe, T., Kaupp, L., Lothary, S., Schäfer, K.U., Thull, B., Vogel, T., Wenning, R.: Machine intelligence today: applications, methodology, and technology. Informatik Spektrum pp. 1–11 (2021). https://doi.org/10.1007/s00287-021-01343-1, https://link.springer.com/article/10.1007%2Fs00287-021-01343-1
10. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) Learning and Intelligent Optimization, Lecture Notes in Computer Science, vol. 6683, pp. 507–523. Springer Nature, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25566-3_40
11. Information technology — Object Management Group Business Process Model and Notation (2013), www.iso.org/standard/62652.html
12. Jin, H., Song, Q., Hu, X.: Auto-keras: An efficient neural architecture search system. In: Teredesai, A. (ed.) Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1946–1956. ACM Digital Library, Association for Computing Machinery, New York,NY,United States (2019). https://doi.org/10.1145/3292500.3330648

13. Kotthoff, L., Thornton, C., Hoos, H.H., Hutter, F., Leyton-Brown, K.: Auto-weka: Automatic model selection and hyperparameter optimization in weka. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds.) Automated machine learning, pp. 81–95. The Springer Series on Challenges in Machine Learning, Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_4
14. Lanio, K.: Rapidminer auto model (09032018), https://rapidminer.com/products/auto-model/
15. Le, T.T., Fu, W., Moore, J.H.: Scaling tree-based automated machine learning to biomedical big data with a feature set selector. Bioinformatics (Oxford, England) **36**(1), 250–256 (2020). https://doi.org/10.1093/bioinformatics/btz470
16. LeDell, E., Poirier, S.: H2o automl: Scalable automatic machine learning. 7th ICML Workshop on Automated Machine Learning (AutoML) (2020), https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf
17. Machine Learning Professorship Freiburg: Autosklearn documentation (16032021), https://automl.github.io/auto-sklearn/master/api.html
18. Mendoza, H., Klein, A., Feurer, M., Springenberg, J., Urban, M., Burkart, M., Dippel, M., Lindauer, M., Hutter, F.: Towards automatically-tuned deep neural networks: 7. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds.) AutoML: Methods, Sytems, Challenges, pp. 141–156. Springer (2018). https://doi.org/10.1007/978-3-030-05318-5_7
19. Microsoft: Blazor (26032021), https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor
20. Miles, A., Bechhofer Sean: Skos simple knowledge organization system namespace document (06082011), https://www.w3.org/2009/08/skos-reference/skos.html
21. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É.: Scikit-learn: Machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011), https://dl.acm.org/doi/10.5555/1953048.2078195
22. Russell, S.J., Norvig, P.: Artificial intelligence: A modern approach. Prentice Hall Series in Artificial Intelligence, Pearson, Upper Saddle River, 3. edition. global edition edn. (2016)
23. Studer, R., Benjamins, V., Fensel, D.: Knowledge engineering: Principles and methods. Data & Knowledge Engineering **25**(1-2), 161–197 (1998). https://doi.org/10.1016/S0169-023X(97)00056-6
24. Swearingen, T., Drevo, W., Cyphers, B., Cuesta-Infante, A., Ross, A., Veeramachaneni, K.: Atm: A distributed, collaborative, scalable system for automated machine learning. In: 2017 IEEE International Conference on Big Data (Big Data). pp. 151–162. IEEE (122017). https://doi.org/10.1109/BigData.2017.8257923
25. The Linux Foundation: Onnx (17032021), https://onnx.ai/
26. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-weka. In: Dhillon, I.S. (ed.) KDD '13 : the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining : August 11-14, 2013, Chicago, Illinois, USA. pp. 847–855. ACM (2013). https://doi.org/10.1145/2487575.2487629
27. Yoo, J., Joseph, T., Yung, D., Nasseri, S.A., Wood, F.: Ensemble squared: A meta automl system, https://arxiv.org/pdf/2012.05390
28. Zöller, M.A., Huber, M.F.: Benchmark and survey of automated machine learning frameworks. Journal of Artificial Intelligence Research **70**, 409–472 (2021). https://doi.org/10.1613/jair.1.11854