Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 235–246
DOI: 10.15439/2023F708
ISSN 2300-5963 ACSIS, Vol. 35

Improving the Efficiency of Meta AutoML via Rule-based Training Strategies

Alexander Zender, Bernhard G. Humm, Tim Pachmann
0000-0002-6956-9049, 0000-0001-7805-1981, 0000-0003-4393-683X
Darmstadt University of Applied Sciences
Schöfferstr. 3, 64295 Darmstadt, Germany
Email: alexander.zender@h-da.de, bernhard.humm@h-da.de, tim.pachmann@icloud.com

***Abstract*—Meta Automated Machine Learning (Meta AutoML) platforms support data scientists and domain experts by automating the ML model search. A Meta AutoML platform utilizes multiple AutoML solutions searching in parallel for their best ML model. Using multiple AutoML solutions requires a substantial amount of energy. While AutoML solutions utilize different training strategies to optimize their energy efficiency and ML model effectiveness, no research has yet addressed optimizing the Meta AutoML process. This paper presents a survey of 14 AutoML training strategies that can be applied to Meta AutoML. The survey categorizes these strategies by their broader goal, their advantage and Meta AutoML adaptability. This paper also introduces the concept of rule-based training strategies and a proof-of-concept implementation in the Meta AutoML platform OMA-ML. This concept is based on the blackboard architecture and uses a rule-based reasoner system to apply training strategies. Applying the training strategy "top-3" can save up to 70% of energy, while maintaining a similar ML model performance.**

I. Introduction

MACHINE LEARNING (ML) is an important subdomain of artificial intelligence (AI), allowing programs to make predictions using models based on previous observations [1]. Creating effective ML models requires substantial knowledge and experience in the field of ML. Data scientists are a group of experts possessing those foundations. Generating an ML model involves several tasks, including data analysis, data preparation, feature engineering, model selection, validation, learning curve analysis and hyperparameter optimization.

The research field of *Automated Machine Learning (AutoML)* emerged to support data scientists and domain experts (professionals in a domain like medicine) with these tasks. AutoML aims to automate the model selection and hyperparameter optimization process leading to higher efficiency, and potentially better results [2]. Finding the best model and its hyperparameter optimization for a given problem is also known as the Combine Algorithm Selection and Hyperparameter Optimization (CASH) problem [3]. An AutoML solution programmatically searches for an ML pipeline by solving the CASH problem [4]. More progressive AutoML solutions also perform data preparation, feature engineering, and validation, allowing for the creation of entire ML pipelines [5]. There are a growing number of AutoML solutions available [6] offering automated solutions for ML tasks belonging to supervised learning (e.g. Auto-WEKA [7]) and unsupervised learning (e.g. AutoCluster [8]). Although AutoML aims to be accessible to users with and without ML and programming expertise, only a few AutoML solutions are targeted at domain experts [6]. Furthermore, due to the wide range of AutoML solutions and their constraint to mainly support only one major ML library (e.g. Auto-Keras [9] supporting Keras[1]), finding the most effective ML method for a given use case requires a trial and error approach.

Meta Automated Machine Learning (Meta AutoML) [10] is a concept that addresses such issues by integrating multiple Automated Machine Learning solutions into one ensemble. During a training session, the AutoML solutions search for their best ML model in parallel. Some AutoML solutions may use different approaches to automatically optimize the model search [3]. This internal optimization aims to reduce the training time and increase the effectiveness of the final ML model. Within the research field of AutoML, a wide range of different optimization approaches exist [11] e.g. meta-learning [12], or early-stopping [13].

Running multiple AutoML solutions in parallel is energy-inefficient. Therefore, improving the energy-efficiency of the Meta AutoML process is an important goal. AI approaches that use vast amounts of computation power to increase their performance can be labelled as red AI [14]. Meta AutoML falls into this red AI category, as it uses a massive amount of computation power to operate multiple AutoML solutions in parallel. It is important to optimize the Meta AutoML process and move it towards green AI [14]. However, on the Meta AutoML level, there has been no research on improving efficiency.

A simple Meta AutoML training strategy to improve energy efficiency is the "top-3" strategy. This strategy performs two successive training sessions. The first training session uses all AutoML solutions with a reduced sample set and training time. The second training session utilizes the full sample set and the remaining training time. This session uses the 3 best performing AutoML solutions found during the first training session. We use the top-3 strategy to demonstrate and evaluate the approach presented in this paper.

[1]https://keras.io

The contributions of this paper are two-fold: (a) A survey of AutoML training strategies that can be applied to Meta AutoML; (b) A concept for a rule-based training strategy component for Meta AutoML and its prototypical implementation in the platform OMA-ML [6] as a proof-of-concept. This paper uses the prototypical implementation to evaluate the effectiveness of a selected training strategy, namely the top-3 strategy. The evaluation compares CO_2 emission equivalents produced by the entire Meta AutoML training and the ML model prediction performance with and without using the training strategy.

This paper is structured as follows. Section II presents related work. Section III shows the survey on training strategies. Section IV is the paper's core, introducing the concept of rule-based training strategies. Section V briefly indicates aspects of the prototypical implementation in the platform OMA-ML. Section VI evaluates the concept and prototypical implementation. Section VII concludes the paper and discusses future work.

II. Related Work

A number of surveys exist that focus on state-of-the-art research about AutoML [6][3][15][16][17][18][19]. These surveys focus on the algorithms and approaches used to solve the CASH problem. They also review individual AutoML solutions. However, only one survey [6] offers a broader survey of existing AutoML solutions. Furthermore, we are not aware of a survey on AutoML training strategies.

The next important aspect of research relates to AutoML training strategies. There are two different categories of AutoML strategies: strategies that are applied during preprocessing and strategies that are applied during training of an ML model. Some AutoML solutions preprocess the dataset only once; e.g., Autogluon infers the process from the initial dataset [20]. Others may use intricate data preprocessing approaches; e.g., TPOT uses genetic programming [21] to find the best preprocessing workflow. Additionally, some AutoML solutions apply strategies to the ML model training process, e.g. early stopping (e.g. H2O: AutoML [22]) which stops the fitting process of a ML model based on a user-defined termination criterion [13]. Another strategy is multi-fidelity (e.g. Auto-Pytorch [23]). This aims to optimize the ML model and hyperparameter search by training models using lower-fidelity (e.g. less time, computation, data) to determine the best configuration to run high-fidelity training [24]. AutoML solutions can implement frameworks like BOHB to use multi-fidelity optimization [25]. Finally, some AutoML solutions implement strategies such as meta-learning [26] to further improve their training by learning from previous ones. While early stopping and multi-fidelity training strategies are used by AutoML solutions, they are not limited to AutoML. Data scientists use the concept of applying optimization strategies to improve their manual search for the best ML model.

There exists a collection of foundation literature introducing different strategies to improve the ML training [27][28]. The literature provides different optimization strategies depending on the dataset and general ML training. There are scientific surveys that compare different approaches used to optimize the individual steps of preprocessing [29][30]. Additionally, there are publications that present new approaches to optimize or replace existing preprocessing approaches [31][32].

The concept of improving the efficiency of Meta AutoML via rule-based training strategies is novel. We are not aware of any publication dealing with this issue. Currently, we are aware of two Meta AutoML platforms: OMA-ML [6] and Ensemble Squared [33]. Both Meta AutoML platforms use a meta layer to administer the built-in AutoML solutions. This abstraction layer allows a user to leverage multiple AutoML solutions simultaneously, without requiring previous knowledge about individual AutoML solutions or data science.

Ensemble Square does not use training strategies to optimize its Meta AutoML process. It uses all supported AutoML solutions for every training session by default.

III. A Survey of ML Training Strategies

In this section we compare 14 training strategies from AutoML solutions regarding their applicability to Meta AutoML.

A. Methodology

The training strategies are evaluated using the following criteria:

- **Category** of the training strategy:
  - *Data cleaning*: Identification and correction of flaws in the data;
  - *Data transformation*: Changing the scaling or distribution of the data;
  - *Complexity reduction*: Reducing the feature or sample size to reduce complexity;
  - *Infrastructure*: Adjusting the available hardware and computation power;
  - *Spot checking*: Using trial training sessions to determine the most viable solution;
  - *Training observation*: Actively supervising the ML model performance during the fit process;
  - *Meta-learning*: Learning from past training sessions to improve future trainings.
- **ML process phase**: The phase during which the strategy is applied:
  - *Pre-processing*: The preprocessing phase occurs before the actual ML model training and focuses on preparing the dataset;
  - *Training*: The training phase where the ML models are fitted to the training data;
  - *Post-processing*: the phase after training proper;
  - *Meta-level*: The Meta-Level is not a phase as such but covers the entire process.
- **Advantage**: The benefits of applying the strategy:
  - *Effectiveness*: The ML model's effectiveness may increase;
  - *Efficiency*: The amount of computation power required by the training may decrease.

TABLE I
OVERVIEW OF AUTOML TRAINING STRATEGIES

Strategy	Category	ML process phase	Advantage	Feasibility	References
Handling outliers	Data cleaning	Preprocessing	Effectiveness	Yes	[27][34][29]
Imputing missing values	Data cleaning	Preprocessing	Effectiveness	Yes	[27][31][32]
Omitting redundant samples	Data cleaning	Preprocessing	Effectiveness, Efficiency	Yes	[27][35]
Data sampling	Data cleaning	Preprocessing	Efficiency	Yes	[27][36]
Text feature encoding	Data transformation	Preprocessing	Effectiveness	Yes	[37]
Numerical feature scaling	Data transformation	Preprocessing	Effectiveness	Yes	[38][39]
Feature extraction	Complexity reduction	Preprocessing	Effectiveness, Efficiency	Partial	[40][41]
Feature selection	Complexity reduction	Preprocessing	Effectiveness, Efficiency	Yes	[42][43][27]
Dimensionality reduction	Complexity reduction	Preprocessing	Effectiveness, Efficiency	Yes	[44][30][45]
Hardware optimization	Infrastructure	Training	Efficiency	Yes	
Multi-fidelity optimization	Spot checking	Training	Efficiency	Yes	[28][46][24]
Top 3 optimization	Spot checking	Training	Efficiency	Yes	
Early Stopping	Training observation	Training	Efficiency	Partial	[13]
Meta-learning	Meta-learning	Meta level	Effectiveness, Efficiency	Yes	[47]

- **Feasibility**: Can the strategy be applied to Meta AutoML:
  - *Yes*: Can be applied to Meta AutoML;
  - *Partial*: Can be partially applicable to Meta AutoML.
  - *No*: Cannot be applied to Meta AutoML.

B. Strategies

Table I gives an overview of the training strategies survey.

1) Data cleaning: Data cleaning aims to remove or repair dirty data within the dataset [27]. The following can be issues within a dataset:

- *Outliers*: A value is an outlier if it significantly deviates from the other values;
- *Missing values*: When no data is available for a feature in a sample;
- *Redundancy*: Identical samples present in a dataset.

a) Handling outliers: Detecting and handling outliers is one of the first steps of data cleaning. Outliers represent noise within data [27]. If outliers are improperly handled, they can decrease the prediction performance of the ML model [34]. Statistical methods can be used to automatically identify and handle outliers (e.g. interquartile range, standard deviation [29]). Some AutoML solutions (e.g. Pycaret[2]) offer support to handle outliers from datasets. Handling outliers may increase the effectiveness of ML models.

b) Imputing missing values: Imputing missing values is important. Missing values can negatively impact the quality and accuracy of ML models by introducing bias [27]. Statistical methods can automatically impute missing values (e.g. k-nearest-neighbor-based approaches [31], neural networks [32]). Some AutoML solutions (e.g. MLJAR[3]) implement automatic imputation of missing values. Handling missing values may increase the effectiveness of ML models, as the potential bias from missing values is not introduced.

[2]https://github.com/pycaret/pycaret
[3]https://github.com/mljar/mljar-supervised

c) Omitting redundant samples: Duplicated samples can introduce bias in the model. Identical samples can negatively impact the search time of the training session and the ML model's effectiveness. Statistical methods can automatically detect duplicate samples (e.g. probabilistic matching [35]). Removing duplicate samples may increase the effectiveness of the ML model by removing potential bias. It may also increase the training efficiency by reducing the size of the dataset.

d) Data sampling: Sampling can adjust the dataset size to the training configuration to allow the most effective training (e.g. limit time or computation power) [27]. By sampling the dataset the training efficiency may increase and the ML model performance may not be negatively impacted [27]. Some AutoML solutions (e.g. FLAML) apply sampling to adjust the dataset size on the training configuration (e.g. limited time and large sample set by applying e.g. holdout [36]).

2) Data transformation: Data transformation aims to change the type or distribution of the data in a dataset. This includes transforming data into a format an ML algorithm can process [27].

a) Text feature encoding: Text features often can not be processed by ML algorithms and require encoding [37]. Text features may be disregarded without proper encoding or lead to errors during the ML model training. When confronted with text features, three categories of encoding strategies exist:

- *Binary encoding*: encodes the textual values into binary values, e.g. yes/no;
- *Ordinal encoding*: encodes the textual values into a finite set of discrete values with a rank or ordering between them. e.g. low/medium/high;
- *Nominal encoding*: encodes the textual values into a finite set of discrete values with no relationship between them. e.g. married/single/widowed.

Some AutoML solutions (e.g. Autokeras [9]) automatically encode text features. Encoding text features may increase the effectiveness of an ML model as it can consider those features while training instead of disregarding them.

b) Numerical feature scaling: Numerical features with a wide range of values may cause bias in the ML model. Some ML approaches are sensitive to the relative magnitude of features and may give more weight to features with larger values [38]. When confronted with numerical features displaying a wide range, two categories of scaling strategies can be applied:

- Normalization: This method resizes the data to a fixed value between 0 and 1;
- Standardization: This method resizes the data to have a median of 0 and a standard deviation of 1.

Some AutoML solutions (e.g. MLJAR) automatically scale numerical features using normalization or standardization functions. Scaling numerical features may increase the effectiveness of an ML model [39].

3) Complexity reduction: Complexity reduction aims to reduce the complexity of a dataset. The available methods can be divided into three categories:

- Feature extraction: Create new features from existing data that may have more meaningful information;
- Feature selection: Reduce the number of features by selecting the most relevant ones without changing the features themselves;
- Dimensionality reduction: Reduce the number of features by transforming the features into a lower dimensional space while preserving essential information.

a) Feature extraction: Feature extraction aims to create a subset of more meaningful features from the existing ones [40]. The construction of new features is highly specific to the data and data type of the dataset. Often, it is required to collaborate with domain experts who can group features correctly together. Automated feature engineering offers data scientists and AutoML solutions methods to automatically create candidate features derived from the original dataset e.g. Deep Feature Synthesis [41]. Using newly created features based on existing data may increase the performance of an ML model. Some AutoML solutions (e.g. MLJAR) apply automated feature engineering to create new and potentially more effective ML models. Multiple open-source libraries focus on automated feature engineering, e.g. Feature-engine[4]. Such tools can help data scientists quickly trial a wide range of different feature combinations. However, only a domain expert has the necessary understanding of the problem to determine the suitability of features.

b) Feature selection: Feature selection aims to reduce dataset complexity by removing non-useful features [42] and creating a feature subset that performs best under classification [43]. A dataset can contain irrelevant or noisy features (e.g. duplicated features) that may introduce bias into an ML model. By removing noisy features, the ML model's effectiveness and training efficiency may increase [27]. The existing methods can be classified into one of three categories:

- *Filter*: Filtering out undesirable features before learning by using heuristics based on the general data characteristics to evaluate the goodness of feature subsets;
- *Wrapper*: Methods that search the feature space for the best-performing subset. They assess the quality of features by training and evaluating a classifier with the subset;
- *Embedded*: Similarly to the wrapper method, the feature selection is performed during the learning process of the ML model.

Some AutoML solutions (e.g. MLJAR) apply automated feature selection to evaluate which feature is relevant for a given search.

c) Dimensionality reduction: Dimensionality reduction aims to reduce the dimensionality of the dataset. The number of features may be considered the dimensionality of the dataset. Dimensionality-reducing methods project the data into a lower-dimensional space that still preserves the most important properties of the original data. By reducing the complexity of the dataset the ML model's effectiveness and the training efficiency may increase. The existing methods can be divided into three categories:

- *Linear dimensionality reduction methods*: e.g. principal component analysis [44];
- *Non-linear dimensionality reduction methods*: e.g. multi-dimensional scaling [30];
- *Autoencoder methods*: e.g. using artificial neural networks [45].

C. Hardware optimization

Hardware optimization aims to choose the most suitable infrastructure for the task and use it in the most energy-efficient and resource-saving way possible. Optimizing the underlying hardware may reduce the amount of computation power invested during training by providing specialized hardware for ML. Another option is to limit the access to computation power to limit the used computation during training. Some AutoML solutions (e.g. MLJAR) can limit the maximum amount of RAM the solution uses.

D. Spot checking

Spot-checking aims to discover the approach that performs best for an ML task [28]. A spot-checking algorithm uses multiple trials to evaluate multiple ML algorithms on a given dataset to determine their performance. The spot-checking training aims to quickly assess the viability of a collection of ML models and decide which approach to use for further training.

a) Multi-fidelity optimization: The multi-fidelity strategy uses numerous training sessions with low-fidelity samples to evaluate the general trend of a system's behaviour, and a small number of high-fidelity samples to enhance the prediction accuracy in important regions [46]. Sequential multi-fidelity surrogate modelling is one multi-fidelity approach that limits the computational budget in addition to using low-fidelity

[4]https://feature-engine.trainindata.com/en/latest/

samples [24]. Some AutoML solutions (e.g. Auto-Pytorch) use multi-fidelity to improve their training.

b) Top-3 optimization: The top-3 strategy is a variation of the multi-fidelity strategy. The training is divided into two steps. The first is a pre-training with a low-fidelity configuration (limited data and time budget). After all candidates are trained, they are evaluated. The top 3 candidates are then trained with the full dataset and the remaining time. The benefit of top-3 is the increased efficiency of the training process, as only the best 3 ML models are fully trained. The top-3 strategy is not used by any existing AutoML solution, but it could be applied to AutoML.

1) Early-stopping: Early stopping is a strategy to intelligently terminate the training when a user-defined early stopping criterion is met. The standard early-stopping criterion is the loss on the validation set [13]. Some AutoML solutions (e.g. PyCaret) use early stopping to optimize training efficiency. On a Meta AutoML level, it is only partially possible to use early stopping. The main issue is that there is currently no interface available for Meta AutoML to extract the current status of the AutoML solution, except through text mining the console output. One issue with text mining is that some AutoML solutions produce limited or no relevant output during the training itself [6]. Additionally, no interface is available for the Meta AutoML process to halt the AutoML training. In most cases, the Meta AutoML could terminate the process using the underlying operating system, which leads to a complete loss of the ML model within the AutoML solution.

2) Meta-learning: Human domain experts derive knowledge from previous tasks by learning about the performance of ML algorithms. Meta-Learning mimics the perpetual process of "learning to learn" across similar tasks and transferring that prior knowledge to new tasks [47]. Some AutoML solutions (e.g. Auto-Sklearn) use meta-learning to form recommendations for ML algorithms and their configuration based on past training sessions. Meta-learning may increase training efficiency and the ML model effectiveness.

In the next section, we introduce a concept for implementing training strategies using a rule engine.

IV. A Concept for Rule-based Training Strategies

Before introducing the concept of rule-based training strategies, we define the optimization goals we aspire to achieve by applying training strategies:

- *Efficiency*: The energy consumption will be significantly reduced compared to a Meta AutoML training without applying a training strategy;
- *Effectiveness*: The ML model performance will not be significantly reduced.

The concept for achieving those goals uses a *blackboard architecture* [48]. The blackboard architecture is a problem-solving approach combining multiple specialized modules working collaboratively to solve a complex problem. The speech understanding system HEARSAY-II introduced and used this approach [49]. The blackboard architecture defines three components:

- *Blackboard*: a global data store that keeps the problem-solving state data;
- *Knowledge sources*: individual components with domain knowledge needed to solve a problem. Each component is represented as procedures, sets of rules or logic assertions and contributes information that will lead to a solution to the problem.
- *Controller*: A component that monitors the changes on the blackboard and decides what actions to take next. The controller decides on the order of invocation of the knowledge sources.

Fig. 1 shows a BPMN diagram [50] of the rule-based training strategies process. The process starts when a new Meta AutoML training is initiated.

A user initiates a new Meta AutoML training by configuring a new training within a Meta AutoML platform. A Meta AutoML platform is an application based on the Meta AutoML concept [10]. It unifies several AutoML solutions and allows the user to configure and start new training sessions without interacting with the AutoML solutions individually. One example of a Meta AutoML platform is OMA-ML [6]. The result of the user's training configuration is the *training configuration* data. For example, a training configuration could be represented as follows:

```
dataset : census_income.csv
task : tabular_classification
autoML_solutions_activated : [FLAML,
Autogluon, Autosklearn, TPOT, MLJAR,
AutoKeras, Pycaret, EvalML]
strategies_enabled : [top-3]
max_runtime : 60
```

In this example, the user configures the training for a tabular classification of the Census Income dataset [51] with a total runtime of 60 minutes. A total of 8 AutoML solutions are activated (FLAML, Autogluon, Autosklearn, TPOT, MLJAR, AutoKeras, Pycaret, EvalML) and one training strategy (top-3) is enabled.

When uploading new training data into the Meta AutoML platform, it is automatically analyzed. During this *dataset analysis*, the *analysis result* data is generated by extracting the properties of the *training data*. For example, the analysis result for the Census Income dataset could be as follows:

```
samples : 48000
features : 15
missing_values : none
duplicated_samples : 48
```

In this example, the Census Income dataset, is comprised of 48,000 samples and 15 features. It has no missing values and a total of 48 duplicated samples.

The training configuration and analysis result are added to the blackboard and represent the initial blackboard state data. The blackboard state data is the collection of the current state data of all the involved components and the current

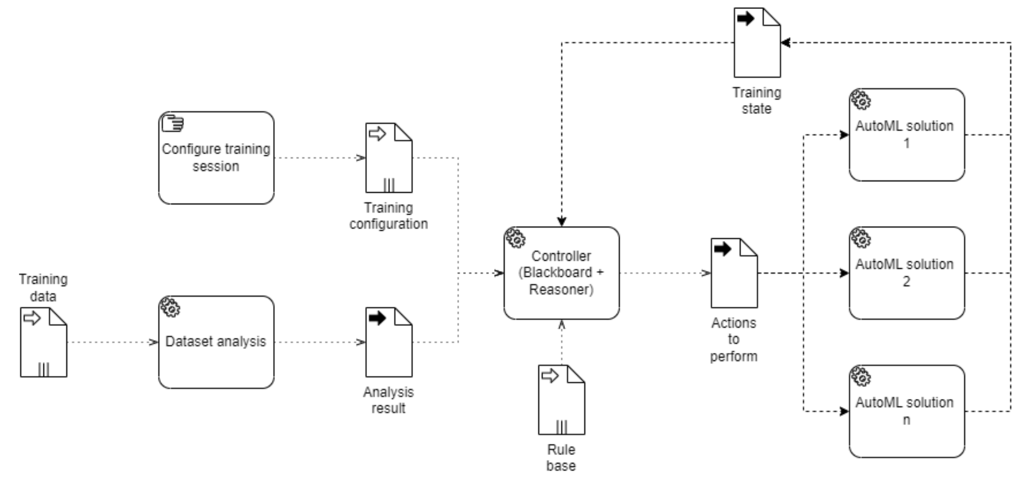


Fig. 1. Rule-based training strategies process as a BPMN diagram

training phase. When the blackboard is initialized the training phase is automatically set to 'preprocessing' and as the training progresses it is updated to 'training' and eventually to 'postprocessing'. For example, using the training configuration and analysis result from the previous examples, the initial blackboard state could be as follows:

```
phase: preprocessing
training_configuration:
    dataset : census_income.csv
    task : ...
analysis_result:
    samples : 48000
    features : ...
```

The blackboard and the reasoner component form together the *controller* module. The controller administers the Meta AutoML training process. It is responsible to advance the Meta AutoML training phase when the reasoner has no more rules to consider. During each phase, the reasoner will assess the *rule base* with the current blackboard state and execute matching rules. The rule base is a collection of all Meta AutoML training strategies supported by a Meta AutoML platform. A training strategy is composed of a condition and a collection of actions to perform if the condition matches. A training strategy is associated with one Meta AutoML training phase (see section III). For example, the definition of the top-3 strategy could be as follows:

```
if
phase == 'training' and
length(training_configuration.autoMl_
solutions_activated) > 3

then
do_top_3_training()
```

In this example, the condition for the top-3 strategy is that the Meta AutoML training phase is equal to 'training' and there are more than three AutoML solutions activated in the training configuration. If this is the case, the controller will execute the do_top_3_training function. During which, the controller instructs the *AutoML solutions* to begin the first training session. This initial training will use only 10% of the sample size, 10% of the max runtime and all activated AutoML solutions. For example, using the blackboard state from above, the *action to perform* by the top-3 strategy for the initial training could be represented as follows:

```
action : training
dataset : census_income.csv
task : tabular_classification
autoML_solutions_activated : [FLAML,
Autogluon, Autosklearn, TPOT, MLJAR,
AutoKeras, Pycaret, EvalML]
sample_size : 10%
max_runtime : 6
```

This action instructs the AutoML solution modules (FLAML, Autogluon, Autosklearn, TPOT, MLJAR, AutoKeras, Pycaret, EvalML) to perform tabular classification training, using 10% of the samples from the Census Income dataset, and a maximum runtime of 6 minutes.

During the training process, the AutoML solution modules notify the blackboard about their current *training state*. Each AutoML solution module represents the implementation of one AutoML solution used by the Meta AutoML platform. The training state represents information about the progression of the AutoML training process. For example, the information provided by the AutoML solution Autokeras during the initial training could be represented as follows:

```
AutoKeras:
    remaining_training_time : 2
    best_model_performance : 0.8
```

In this example, the remaining training time is 2 minutes, and the highest ML model performance is 80%. The ML model performance is measured using a standard ML metric for the current ML task. For example, the metric for tabular classification could be accuracy. This training state is updated throughout the AutoML solution training.

When the initial training for the top-3 strategy concludes, the controller evaluates the training state from all AutoML solutions and continues with the second training session using the complete sample set, the remaining 90% of the max runtime and the top-3 AutoML solutions from the preliminary training. For example, the final training state of the 8 AutoML solutions could be as follows:

```
FLAML:
    remaining_training_time : 0
    best_model_performance : 0.77
Autogluon:
    remaining_training_time : 0
    best_model_performance : 0.78
Autosklearn:
    remaining_training_time : 0
    best_model_performance : 0.87
TPOT:
    remaining_training_time : 0
    best_model_performance : 0.87
MLJAR:
    remaining_training_time : 0
    best_model_performance : 0.88
AutoKeras:
    remaining_training_time : 0
    best_model_performance : 0.92
Pycaret:
    remaining_training_time : 0
    best_model_performance : 0.95
EvalML:
    remaining_training_time : 0
    best_model_performance : 0.95
```

All AutoML solutions finished the preliminary training indicated by the remaining training time of 0. The controller based on the top-3 strategy decides that the best-performing AutoML solutions by best model performance are: AutoKeras, Pycaret and EvalML. The controller instructs these three AutoML solutions on further actions to perform. In this case, the action is to begin the second training using the remaining time and complete sample set. For example, the action to perform the second training could be as defined as follows:

```
action : training
dataset : census_income.csv
task : tabular_classification
autoML_solutions_activated : [AutoKeras,
Pycaret, EvalML]
sample_size : 1.0
max_runtime : 54
```

This action instructs the AutoML solutions AutoKeras, Pycaret and EvalML to perform a tabular classification training using the Census Income dataset with all the samples and a maximum runtime of 54 min.

As described above, the AutoML solution modules perform the training session and update their training state on the blackboard accordingly. When all three AutoML solutions conclude their training, the top-3 strategy ends. When the controller assesses that no more rules can be applied during this training, the Meta AutoML training concludes.

In the next section, we introduce the prototypical implementation of the rule-based training strategy within the OMA-ML platform.

V. Prototypical Implementation

The rule-based training strategy concept was implemented as a proof-of-concept in the Meta AutoML platform OMA-ML. OMA-ML is an open-source[5] platform providing users with a web application-based interface to configure the Meta AutoML training. Fig. 2 displays a screenshot of the training wizard used to configure the Meta AutoML training.

The training wizard displays the required and optional parameters for a Meta AutoML training. The minimal configuration OMA-ML requires comprises of the ML task (tabular classification), the target column ('class') and a maximum runtime (60 minutes).

For expert users, OMA-ML allows in-depth parametrization of the Meta AutoML training using optional parameters. The user may activate or deactivate individual ML libraries and associated AutoML solutions. For example, in Fig. 2 the following AutoML solutions are activated: Autogluon, Autosklearn, EvalML, FLAML, MLJAR, TPOT, Pycaret and Autokeras. Additionally, OMA-ML displays a collection of training strategies compatible with the current training configuration. For example, the top-3 strategy is available since more than 3 AutoML solutions are activated.

OMA-ML uses an ML ontology to display individual AutoML solution parameters. These are parameters AutoML solutions provide to fine-tune their search process. When the user clicks on the finish button, the Meta AutoML training begins.

The OMA-ML web application is developed in C# with the Blazor web framework[6]. OMA-ML follows a 3-layer architecture design. See Fig. 3 for an overview of the software architecture and technologies used.

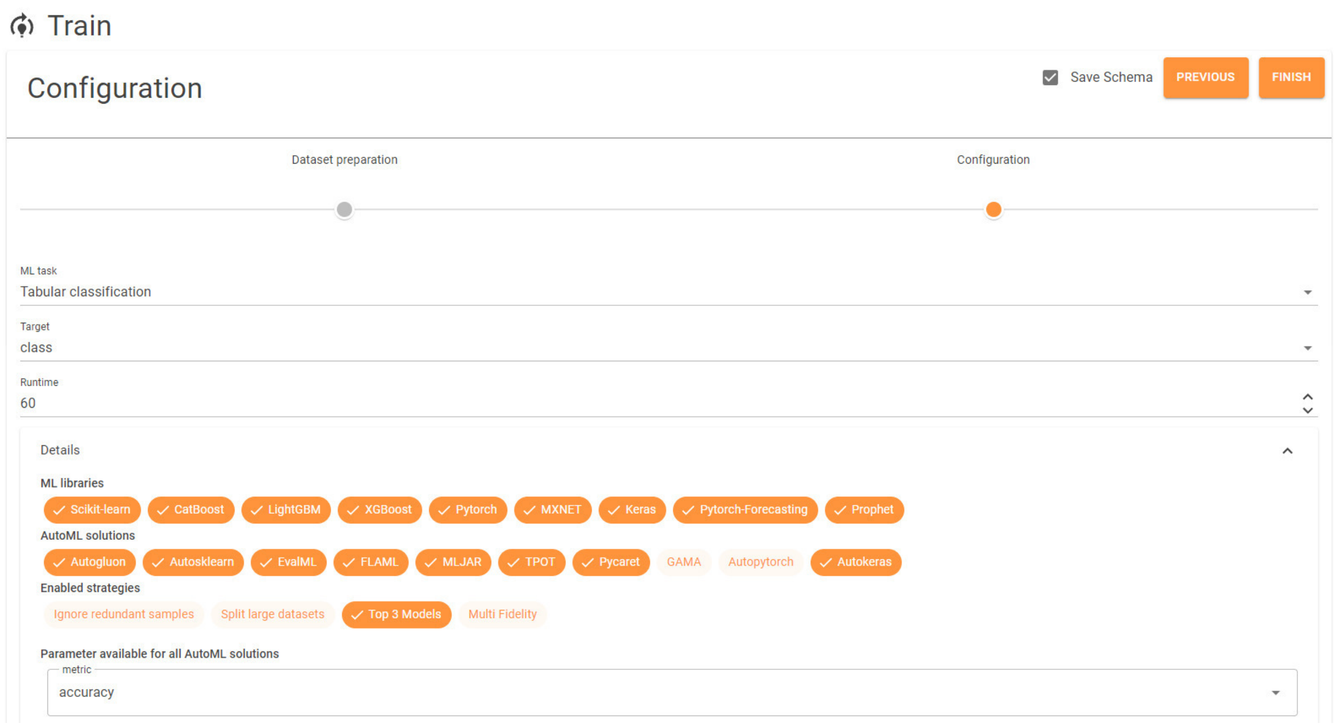
The presentation layer is connected to the logic layer using a gRPC[7] interface. The logic layer is developed in Python based on the blackboard architecture. The Controller component uses the library rule-engine[8] to reason over the rule base. The library rule-engine provides a grammar to create general-purpose rule objects from a logical expression that can be applied to arbitrary objects. The Meta AutoML training strategies conditions are modelled using this grammar. The training strategies are implemented within a rule base

[5]https://github.com/hochschule-darmstadt/MetaAutoML

[6]https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor

[7]https://grpc.io/

[8]https://pypi.org/project/rule-engine/



Train

Configuration

☑ Save Schema   PREVIOUS   FINISH

Dataset preparation — Configuration

ML task
Tabular classification

Target
class

Runtime
60

Details

ML libraries
✓ Scikit-learn ✓ CatBoost ✓ LightGBM ✓ XGBoost ✓ Pytorch ✓ MXNET ✓ Keras ✓ Pytorch-Forecasting ✓ Prophet

AutoML solutions
✓ Autogluon ✓ Autosklearn ✓ EvalML ✓ FLAML ✓ MLJAR ✓ TPOT ✓ Pycaret GAMA Autopytorch ✓ Autokeras

Enabled strategies
Ignore redundant samples Split large datasets ✓ Top 3 Models Multi Fidelity

Parameter available for all AutoML solutions
metric
accuracy

Fig. 2. Screenshot of the OMA-ML training configuration wizard

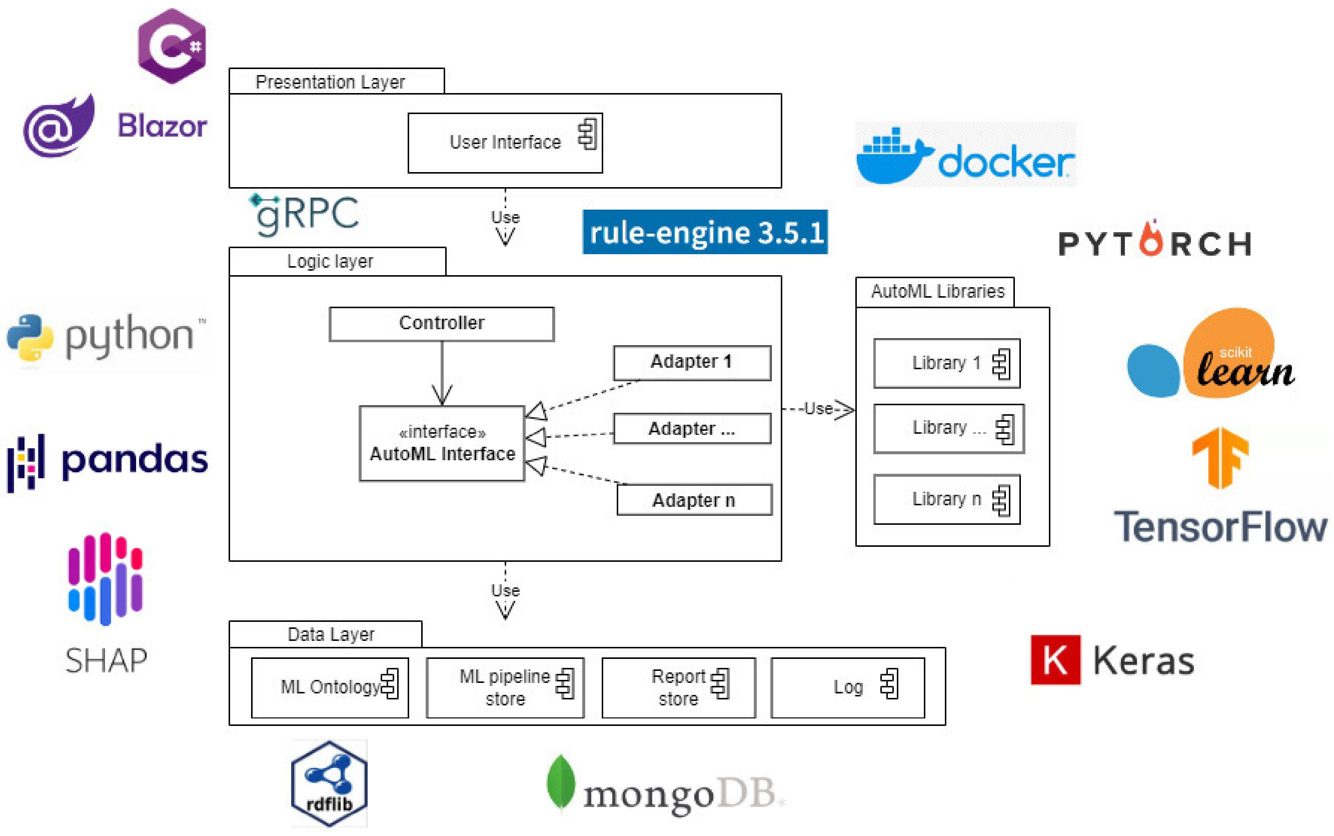


Fig. 3. OMA-ML software architecture and technologies (adapted from [6])

component. For example, the top-3 strategy implementation is as follows:

```
self.register_rule(
'training.top_3',
        Rule("phase == 'training'" and
        training_configuration['activated
        _auto_ml_solution'].length > 3,
        context=training_context),
        self.do_top_3
)
```

The register_rule functionality of the rule base permits rules to be registered. This method requires as parameters, the name, a condition in the rule-engine grammar and a function to execute when the condition matches. In the example above the top-3 strategy has:

- the name: 'training.top_3';
- the condition: the phase is training and the sum of all activated AutoML solutions must be greater than three;
- the function: 'do_top_3'.

Using the register_rule functionality, new training strategies can easily be added to the existing rule base. For example, the whole implementation of the top-3 strategy requires less than 50 lines of code. During a Meta AutoML training, the controller evaluates the rule base for any rule matching the state of the blackboard. Any matching rule invokes their respective function and interacts with the AutoML adapters. The AutoML adapters use the AutoML solutions (e.g. AutoKeras). They are based on the adapter-pattern and plug into the Controller, easing integration.

The logic layer uses the data layer to connect to various data stores. The ML ontology is located here and loaded using the Pyhton library RDFlib[9] into the Controller. SPARQL queries are used to interact with the ML ontology. Additionally, the document database MongoDB stores data generated by the Meta AutoML process. The ML pipelines generated by the AutoML adapters are saved in the ML pipeline store. Finally, any logs generated during the Meta AutoML training process are stored in a log storage.

VI. Evaluation

This section evaluates the concept and implementation of the rule-based training strategies within the Meta AutoML platform OMA-ML. By applying training strategies to Meta AutoML we aim to significantly reduce the energy consumption (efficiency) of the Meta AutoML process while avoiding a significant reduction in ML model performance (effectiveness). To evaluate these goals we compare two measures of quality:

- *Best ML model accuracy*: The highest accuracy of all the AutoML solutions found ML models;
- *Training CO_2-eq*: The sum of all the CO_2 equivalence produced by the AutoML solutions training.

[9]https://pypi.org/project/rdflib/

To measure the CO_2 equivalence, the Python library codecarbon[10] was used. Codecarbon measures the amount of CO_2-eq emitted by the individual AutoML solutions. Codecarbon tracks the power consumption of the underlying computational infrastructure, measured in kilowatt-hours. This value is multiplied by the carbon intensity of the electricity consumed for the computation. The carbon intensity is calculated as a weighted average of the emissions of the different energy sources used to generate the used electricity (e.g. natural gas, coal, wind) in the respective country[11], here Germany.

During the course of the experiment, five datasets from the open-source AutoML benchmark by Gijsbers et al. [52] were used:

- *adult*[12]: This dataset contains samples of more than 48,000 individuals and their socioeconomic properties extracted from the Census database in 1994. This dataset is a binary classification and aims to predict if an individual earns over 50k a year;
- *amazon*[13]: This dataset contains samples of more than 32,000 resource requests with the associated Amazon employee meta information from 2010 and 2011. This dataset is a binary classification and aims to predict if access to a resource was approved;
- *sylvine*[14]: This dataset contains more than 5,000 numerical samples; there is no definition of the origin of the data of this dataset. This dataset is a binary classification;
- *credit-g* [15]: This dataset contains samples of 1,000 individuals, their economic properties and loan requests. This dataset is a binary classification and aims to predict the credit score of an individual;
- *kc1*[16]: This dataset contains samples of more than 2,000 software modules and their quality metrics. This dataset is a binary classification and aims to predict if a software module has a defect.

OMA-ML performed two training sessions with each dataset. During the first training session, no optimization strategy was activated. For the second training session, the top-3 optimization strategy was activated. After every training session, the best ML model performance by accuracy [53] was logged as well as the AutoML solution which produced this ML model. Finally, the total CO_2-eq was calculated by taking the total of all AutoML solutions training CO_2-eq.

The training sessions were performed on an AMD Ryzen 7 5800H @ 3.20 GHz CPU with 64GB of RAM. A result summary of all training sessions can be seen in Table II.

The Meta AutoML training sessions applying the top-3 optimization all display a significant saving in CO_2-eq. The difference ranges from 59% with the adult dataset to 70% with the sylvine dataset. The performance of the best ML model

[10]https://github.com/mlco2/codecarbon
[11]https://mlco2.github.io/codecarbon/methodology.html
[12]https://openml.org/search?type=data&status=active&id=179
[13]https://openml.org/search?type=data&status=active&id=4135
[14]https://openml.org/search?type=data&status=active&id=41146
[15]https://openml.org/search?type=data&status=active&id=31
[16]https://openml.org/search?type=data&status=active&id=1067

TABLE II
EVALUATION SUMMARY

	Meta AutoML, no optimization			**Meta AutoML, top-3 optimization**			**Comparision**	
Dataset	**AutoML solution**	**accuracy**	**CO_2-eq [g]**	**AutoML solution**	**accuracy**	**CO_2-eq [g]**	**Difference accuracy**	**Saving CO_2-eq**
adult	FLAML	0.88	53.5	FLAML	0.87	22.1	-0.01	59%
amazon	Autogluon	0.95	45.8	Autogluon	0.95	16.6	0.00	64%
sylvine	Autosklearn	0.95	49.96	Autosklearn	0.95	15.0	0.00	70%
credit-g	FLAML	0.78	36.2	MLJAR	0.77	11.7	-0.01	68%
kc1	Autogluon	0.87	37.0	Autogluon	0.87	13.1	0.00	65%

is equal to or at worst one percentage point lower compared to the Meta AutoML training sessions without optimization. Additionally, the best AutoML solution is the same in both training sessions for four out of the five datasets. The only exception being the dataset credit-g, the best AutoML solution is FLAML in the training without optimization and MLJAR during the training with optimization. While FLAML did not produce the best ML model during the optimized training session, it was one of three AutoML solutions selected by the top-3 strategy to train a model in the second training session.

By using the top-3 training strategy it is possible to significantly (up to 70%) reduce the amount of CO_2-eq for a training session. While also achieving similar or slightly (1 percentage point) reduced results in the performance of the best ML model, the goals of this study can been regarded as achieved.

VII. Conclusion and Future Work

Meta AutoML provides users with or without data science knowledge access to automatically generated ML models. However, there is the major issue of massive computation power requirements for Meta AutoML. The concept of rule-based training strategies aims to optimize the energy efficiency and ML model effectiveness for Meta AutoML. The contribution of this paper is two fold. Firstly, we presented a survey of 14 AutoML training strategies, classifying them by their function category, ML phase, advantage and implementation feasibility for Meta AutoML. Most training strategies aim to optimize the dataset and only 3 strategies focus on the ML model fitting process. Twelve training strategies can be fully applied to the Meta AutoML process. The exception being the feature extraction and early-stopping strategy. While different approaches exist to automate feature extraction, only a domain expert can decide if the extracted features are relevant. Early stopping requires process information and an interface from the AutoML solutions to allow interaction by a Meta AutoML platform. Both of these are not supported by most AutoML solutions, making implementation of early stopping challenging. While AutoML solutions implement various training strategies already, on the Meta AutoML level however there has been no research on optimization using training strategies. The second contribution of this paper addresses this issue.

We presented the novel concept of rule-based training strategies. This concept uses the blackboard architecture to implement training strategies for Meta AutoML. This concept aims to significantly increase the efficiency of Meta AutoML by reducing the required computation power while not significantly reducing the best ML model performance. The Meta AutoML platform OMA-ML was used to implement a proof-of-concept. We evaluated the implementation by using the training strategy top-3. During the evaluation, five binary classification datasets were used. Using each dataset, two experiments were performed, one without an optimization strategy and one with the top-3 optimization strategy. Applying the top-3 optimization led to a saving of up to 70% CO_2-eq, with the best ML model having identical or slightly reduced performance (one percentage point).

The results show that rule-based training strategies can improve the Meta AutoML training efficiency significantly with only a slight reduction in ML model effectiveness. However, further evaluation is required. In future work, we aim to implement additional training strategies and perform extensive benchmark testing using a variety of ML tasks (e.g. regression, time series forecasting etc.) and various dataset types (e.g. texts, images, time series etc.).

Acknowledgment

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] S. J. Russell and P. Norvig, *Artificial intelligence: A modern approach*, fourth edition ed., ser. Pearson Series in Artificial Intelligence. Hoboken, NJ: Pearson, 2021. ISBN 9780134610993

[2] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, Eds., *Efficient and Robust Automated Machine Learning*. MIT Press, 2015. doi: 10.5555/2969442.2969547

[3] M.-A. Zöller and M. F. Huber, "Benchmark and survey of automated machine learning frameworks," *Journal of Artificial Intelligence Research*, vol. 70, pp. 409–472, 2021. doi: 10.1613/jair.1.11854

[4] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-weka: Combined selection and hyperparameter optimization of classification algorithms," in *KDD '13 : the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining : August 11-14, 2013, Chicago, Illinois, USA*, I. S. Dhillon, Ed. ACM, 2013. doi: 10.1145/2487575.2487629. ISBN 9781450321747 pp. 847–855.

[5] C. H. N. Larcher and H. J. C. Barbosa, “Auto-cve: a coevolutionary approach to evolve ensembles in automated machine learning,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. ACM Digital Library, M. López-Ibáñez, Ed. New York,NY,United States: Association for Computing Machinery, 2019. doi: 10.1145/3321707.3321844. ISBN 9781450361118 pp. 392–400.

[6] A. Zender and B. G. Humm, “Ontology-based meta automl,” *Integrated Computer-Aided Engineering*, vol. 29, no. 4, pp. 351–366, 2022. doi: 10.3233/ICA-220684

[7] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, “Auto-weka: Automatic model selection and hyperparameter optimization in weka,” in *Automated machine learning*, ser. The Springer Series on Challenges in Machine Learning, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 81–95. ISBN 978-3-030-05317-8

[8] Y. Poulakis, C. Doulkeridis, and D. Kyriazis, “Autoclust: A framework for automated clustering based on cluster validity indices,” in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020. doi: 10.1109/ICDM50108.2020.00153. ISBN 978-1-7281-8316-9 pp. 1220–1225.

[9] 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*, ser. ACM Digital Library, A. Teredesai, Ed. New York,NY,United States: Association for Computing Machinery, 2019. doi: 10.1145/3292500.3330648. ISBN 9781450362016 pp. 1946–1956.

[10] B. G. Humm and A. Zender, “An ontology-based concept for meta automl,” in *Artificial Intelligence Applications and Innovations*, ser. Springer eBook Collection, I. Maglogiannis, J. Macintyre, and L. Iliadis, Eds. Cham: Springer International Publishing and Imprint Springer, 2021, vol. 627, pp. 117–128. ISBN 978-3-030-79149-0

[11] G. Montavon, G. B. Orr, and K.-R. Müller, Eds., *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. ISBN 978-3-642-35288-1

[12] S. Thrun and L. Pratt, “Learning to learn: Introduction and overview,” in *Learning to Learn*, S. Thrun and L. Pratt, Eds. Boston, MA and s.l.: Springer US, 1998, pp. 3–17. ISBN 978-1-4613-7527-2

[13] L. Prechelt, “Early stopping — but when?” in *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7700, pp. 53–67. ISBN 978-3-642-35288-1

[14] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, “Green ai,” *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020. doi: 10.1145/3381831

[15] A. Doke and M. Gaikwad, “Survey on automated machine learning (automl) and meta learning,” in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2021. doi: 10.1109/ICCCNT51525.2021.9579526 pp. 1–5.

[16] Y.-W. Chen, Q. Song, and X. Hu, “Techniques for automated machine learning,” *ACM SIGKDD Explorations Newsletter*, vol. 22, no. 2, pp. 35–50, 2021. doi: 10.1145/3447556.3447567

[17] P. Ge, “Analysis on approaches and structures of automated machine learning frameworks,” in *2020 International Conference on Communications, Information System and Computer Engineering*. Piscataway, NJ: IEEE, 2020. doi: 10.1109/CISCE50729.2020.00106. ISBN 978-1-7281-9761-6 pp. 474–477.

[18] X. He, K. Zhao, and X. Chu, “Automl: A survey of the state-of-the-art,” *Knowledge-Based Systems*, vol. 212, p. 106622, 2021. doi: 10.1016/j.knosys.2020.106622

[19] Q. Yao, M. Wang, Y. Chen, W. Dai, Y.-F. Li, W.-W. Tu, Q. Yang, and Y. Yu, “Taking human out of learning applications: A survey on automated machine learning.” [Online]. Available: https://arxiv.org/pdf/1810.13306

[20] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola, “Autogluon-tabular: Robust and accurate automl for structured data.” [Online]. Available: https://arxiv.org/pdf/2003.06505

[21] T. T. Le, W. Fu, and J. H. Moore, “Scaling tree-based automated machine learning to biomedical big data with a feature set selector,” *Bioinformatics (Oxford, England)*, vol. 36, no. 1, pp. 250–256, 2020. doi: 10.1093/bioinformatics/btz470

[22] E. LeDell and S. Poirier, “H2o automl: Scalable automatic machine learning,” *7th ICML Workshop on Automated Machine Learning (AutoML)*, 2020. [Online]. Available: https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf

[23] L. Zimmer, M. Lindauer, and F. Hutter, “Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 9, pp. 3079–3090, 2021. doi: 10.1109/TPAMI.2021.3067763

[24] A. I. Forrester, A. Sóbester, and A. J. Keane, “Multi-fidelity optimization via surrogate modelling,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2088, pp. 3251–3269, 2007. doi: 10.1098/rspa.2007.1900

[25] S. Falkner, A. Klein, and F. Hutter, “Bohb: Robust and efficient hyperparameter optimization at scale,” in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 1437–1446. [Online]. Available: https://proceedings.mlr.press/v80/falkner18a.html

[26] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, “Auto-sklearn: Efficient and robust automated machine learning,” in *Automated machine learning*, ser. The Springer Series on Challenges in Machine Learning, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 113–134. ISBN 978-3-030-05317-8

[27] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*. Cham: Springer International Publishing, 2015, vol. 72. ISBN 978-3-319-10246-7

[28] J. Brownlee, *Machine learning mastery with Python: Understand your data, create accurate models and work projects end-to-end*, edition: v1.20 ed. [Australia]: [Jason Brownlee], 2021. ISBN 979-8540446273

[29] E. Panjei, Le Gruenwald, E. Leal, C. Nguyen, and S. Silvia, “A survey on outlier explanations,” *The VLDB journal : very large data bases : a publication of the VLDB Endowment*, vol. 31, no. 5, pp. 977–1008, 2022. doi: 10.1007/s00778-021-00721-1

[30] G. B. Rabinowitz, “An introduction to nonmetric multidimensional scaling,” *American Journal of Political Science*, vol. 19, no. 2, p. 343, 1975. doi: 10.2307/2110441

[31] P. Keerin, W. Kurutach, and T. Boongoen, “Cluster-based knn missing value imputation for dna microarray data,” in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2012. doi: 10.1109/ICSMC.2012.6377764. ISBN 978-1-4673-1714-6 pp. 445–450.

[32] I. A. Gheyas and L. S. Smith, “A neural network-based framework for the reconstruction of incomplete data sets,” *Neurocomputing*, vol. 73, no. 16-18, pp. 3039–3065, 2010. doi: 10.1016/j.neucom.2010.06.021

[33] J. Yoo, T. Joseph, D. Yung, S. A. Nasseri, and F. Wood, “Ensemble squared: A meta automl system.” [Online]. Available: https://arxiv.org/pdf/2012.05390

[34] M. Kalisch, M. Michalak, M. Sikora, Ł. Wróbel, and P. Przystałka, “Influence of outliers introduction on predictive models quality,” in *Beyond Databases, Architectures and Structures. Advanced Technologies for Data Mining and Knowledge Discovery*, ser. Communications in Computer and Information Science, S. Kozielski, D. Mrozek, P. Kasprowski, B. Małysiak-Mrozek, and D. Kostrzewa, Eds. Cham: Springer International Publishing, 2016, vol. 613, pp. 79–93. ISBN 978-3-319-34098-2

[35] J. Asher, D. Resnick, J. Brite, R. Brackbill, and J. Cone, “An introduction to probabilistic record linkage with a focus on linkage processing for wtc registries,” *International journal of environmental research and public health*, vol. 17, no. 18, 2020. doi: 10.3390/ijerph17186937

[36] C. Wang, Q. Wu, M. Weimer, and E. Zhu, “Flaml: A fast and lightweight automl library,” in *Proceedings of Machine Learning and Systems*, A. Smola, A. Dimakis, and I. Stoica, Eds., vol. 3, 2021, pp. 434–447. [Online]. Available: https://proceedings.mlsys.org/paper/2021/file/92cc227532d17e56e07902b254dfad10-Paper.pdf

[37] K. Potdar, T. S., and C. D., “A comparative study of categorical variable encoding techniques for neural network classifiers,” *International Journal of Computer Applications*, vol. 175, no. 4, pp. 7–9, 2017. doi: 10.5120/ijca2017915495

[38] J. Grus, *Data science from Scratch: First principles with Python*, 1st ed. Beijing and Köln: O’Reilly, 2015. ISBN 978-1-491-90142-7

[39] M. Ahsan, M. Mahmud, P. Saha, K. Gupta, and Z. Siddique, “Effect of data scaling methods on machine learning algorithms and model performance,” *Technologies*, vol. 9, no. 3, p. 52, 2021. doi: 10.3390/technologies9030052

[40] S. Khalid, T. Khalil, and S. Nasreen, “A survey of feature selection and feature extraction techniques in machine learning,” in *2014 Science and Information Conference*. IEEE, 2014. doi: 10.1109/SAI.2014.6918213. ISBN 978-0-9893193-1-7 pp. 372–378.

[41] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2015. doi: 10.1109/DSAA.2015.7344858. ISBN 978-1-4673-8272-4 pp. 1–10.

[42] A. Zheng and A. Casari, *Feature engineering for machine learning: Principles and techniques for data scientists*. Beijing and Boston and Farnham and Sebastopol and Tokyo and Beijing and Boston and Farnham and Sebastopol and Tokyo: O'Reilly, 2018. ISBN 978-1491953242

[43] A. Jain and D. Zongker, "Feature selection: evaluation, application, and small sample performance," *IEEE transactions on pattern analysis and machine intelligence*, vol. 19, no. 2, pp. 153–158, 1997. doi: 10.1109/34.574797

[44] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010. doi: 10.1002/wics.101

[45] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science (New York, N.Y.)*, vol. 313, no. 5786, pp. 504–507, 2006. doi: 10.1126/science.1127647

[46] Q. Zhou, M. Zhao, J. Hu, and M. Ma, *Multi-fidelity Surrogates: Modeling, Optimization and Applications*, 1st ed., ser. Engineering Applications of Computational Methods. Singapore: Springer Nature Singapore and Imprint Springer, 2023, vol. 12. ISBN 978-981-19-7212-6

[47] S. Thrun and L. Pratt, Eds., *Learning to Learn*. Boston, MA and s.l.: Springer US, 1998. ISBN 978-1-4613-7527-2

[48] H. Penny Nii, "An introduction to knowledge engineering, blackboard model, and age," 03.1980. [Online]. Available: https://purl.stanford.edu/cq570jp5428

[49] L. D. ERMAN, F. HAYES-ROTH, V. R. LESSER, and D. R. REDDY, "The hearsay-ii speech-understanding system: Integrating knowledge to resolve uncertainty," in *Readings in Artificial Intelligence*. Elsevier, 1981, pp. 349–389. ISBN 9780934613033

[50] "Iso/iec 19510:2013(en), information technology — object management group business process model and notation," 31.03.2022. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:19510:ed-1:v1:en

[51] R. Kohavi, "Census income," 1996.

[52] P. Gijsbers, E. LeDell, J. Thomas, S. Poirier, B. Bischl, and J. Vanschoren, "An open source automl benchmark." [Online]. Available: https://arxiv.org/pdf/1907.00909

[53] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009. doi: 10.1016/j.ipm.2009.03.002