AutoML: state of the art with a focus on anomaly detection, challenges, and research directions

Maroua Bahri, Flavia Salutari, Andrian Putina, Mauro Sozio

International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-022-00309-0

REVIEW

AutoML: state of the art with a focus on anomaly detection, challenges, and research directions

Maroua Bahri[1] · **Flavia Salutari**[2] · **Andrian Putina**[2] · **Mauro Sozio**[2]

Received: 4 May 2021 / Accepted: 3 January 2022

Abstract
The last decade has witnessed the explosion of machine learning research studies with the inception of several algorithms proposed and successfully adopted in different application domains. However, the performance of multiple machine learning algorithms is very sensitive to multiple ingredients (e.g., hyper-parameters tuning and data cleaning) where a significant human effort is required to achieve good results. Thus, building well-performing machine learning algorithms requires domain knowledge and highly specialized data scientists. Automated machine learning (autoML) aims to make easier and more accessible the use of machine learning algorithms for researchers with varying levels of expertise. Besides, research effort to date has mainly been devoted to autoML for supervised learning, and only a few research proposals have been provided for the unsupervised learning. In this paper, we present an overview of the autoML field with a particular emphasis on the automated methods and strategies that have been proposed for unsupervised anomaly detection.

Keywords Machine learning · AutoML · Anomaly detection · Unsupervised learning · Hyper-parameter tuning

1 Introduction

Artificial intelligence (AI) is often used to describe a wide-ranging branch of computer science that aims to build smart machines capable of mimicking tasks that simulate human intelligence. The applications for AI are infinite and present in multiple various sectors. They include, but not limited to, robotics, healthcare industry, autonomous vehicles [13]. Machine learning is a fundamental subset of AI that refers to the development and the use of systems which are able to learn from the data and improve themselves without—or with minimal—external intervention. In fact, machine learning algorithms are characterized by learning from instances, known as "training instances," in order to build appropriate models and make predictions or decisions on a different set of unknown data, also called "test instances" [10].

All machine learning algorithms take, inter alia, instances as input, nevertheless their objectives can be different depending on the task to be accomplished. On the one hand, there is the supervised learning that consists of *classification* and *regression*. The former attempts to assign a class label, also called category, to unlabeled instances using the models that have been trained on a set of training instances, whose label membership is known. The latter instead attempts to predict continuous values rather than categorical ones [4]. On the other hand, there is the unsupervised learning, such as *clustering*, that involves grouping unlabeled instances into categories or clusters based on a similarity measure or distance. In this task, class labels are unknown. Finally, the semi-supervised learning is another major task that lies between the supervised and the unsupervised learning, where the fact that some labels are missing in a training set does not prevent them from being used to improve the predictive learning performance of models [17]. Besides, other approaches have been widely used in conjunction with machine learning systems and for visualization, such as dimensionality reduction [6,18].

Several machine learning algorithms have been developed and successfully adopted in different application domains.

✉ Maroua Bahri
maroua.bahri@inria.fr

Flavia Salutari
flavia.salutari@telecom-paris.fr

Andrian Putina
andrian.putina@telecom-paris.fr

Mauro Sozio
mauro.sozio@telecom-paris.fr

[1] MiMove, INRIA, Paris, France

[2] LTCI, Télécom Paris, IP-Paris, Palaiseau, France

Published online: 04 February 2022

The use of these traditional machine learning algorithms can potentially be costly, in terms of computational resources, and challenging. Moreover, algorithms have multiple hyper-parameters that are set prior to the learning process and can directly affect the performance of the models. However, this task requires domain knowledge and human expertise to perform manual hyper-parameter tuning (tries out hyper-parameter sets by hand), which is a hard and tedious task for non-expert as well as expert users. Another major drawback of the manual search is the fact that the process is not easily reproducible. The latter is definitely important, especially for the progress and improvement of scientific research in the machine learning field. Moreover, it is difficult to manage the manual hyper-parameter tuning task when the number of parameters and the range of values is high, i.e., if one aims to apply an algorithm with five hyper-parameters, then these hyper-parameters need to be manually tuned with different values on different datasets, and the best combination that achieves the highest performance will be chosen.

To cope with the aforementioned issues, automatic search techniques have been proposed under the hot *automated Machine Learning (autoML)* topic [11,36]. The autoML process comprises different tasks, such as feature selection, feature extraction, model selection, and hyper-parameter tuning. It makes easier the use of machine learning algorithms and simplifies the process of selecting the best model and identifying its corresponding optimal set of hyper-parameters. The automated processing evaluates the performance of the learned models according to a predefined quality metric, such as accuracy (for classification) and silhouette measure (for clustering). So far, the autoML community mostly focuses on the supervised learning, where multiple works have been proposed [6,21,25,79]. However, few research studies have been proposed for the unsupervised learning. This gap is present especially for unsupervised anomaly detection, due to the void of an unsupervised quality metric to evaluate anomaly detection algorithms [90].

Anomaly detection is a widely studied problem in machine learning and data mining. Roughly speaking, it consists of identifying instances that "significantly" deviate from the other points in a given dataset. Historically, anomaly detection (aka outlier detection) served mainly as a tool to filter out noise, which could impair the training of a machine learning algorithm. Nowadays, it plays perhaps a more noble role, as the machine learning community realized that anomalies are often associated with interesting or rare events. Applications of anomaly detection are vast, encompassing network security and maintenance, data storage, finance, and medicine. For example, anomaly detection algorithms provide important primitives in the detection of intrusions in networks, frauds in financial transactions, data leaks, as well as life-threatening situations in patient monitoring systems which generally consists of unlabeled data. A large number of approaches has been developed over the years. Supervised anomaly detection can be used when large amounts of labeled data are readily available. However, because of the difficulty in obtaining labeled data, unsupervised approaches preserve their appeal.

In this paper, we provide a review on autoML and its principal solutions. Among the others, we focus on providing an updated overview of the autoML and hyper-parameters optimization methods proposed for the unsupervised anomaly detection.

The important components of constructing an ML pipeline automatically are introduced and existing approaches are evaluated. This allows revealing the limitations of the current approaches and raising open research questions. We also briefly discuss some promising research directions. This review does not consider Neural Architecture Search (NAS)[1] methods where most existing work on NAS has focused on image classification [63] and is computationally expensive due to the large number of child models needed to achieve good performance. Moreover, the search space for automated anomaly detection is different from the search spaces defined by the existing NAS that should cover not only the configurations, but also the anomaly definitions [46]. We however refer to [22] for a recent in-depth survey devoted to NAS methods [22].

2 AutoML

The performance of a given machine learning method on a particular dataset depends on the quality of the algorithm, based on different criteria such as accuracy, memory usage, and running time, as well as its parameterization (i.e., the parameter settings). A data scientist is commonly challenged with a large number of choices. For example, the selection among a wide range of possible algorithms depending on the ML task (e.g., decision trees, Bayesian models, neural networks, k-means, etc.) in addition to tuning the different hyper-parameters of the selected algorithm in order to come up with the best performing configuration.

However, sometimes data scientists need to evaluate several algorithms and tune their different hyper-parameters to select the best performing model and fix the corresponding optimal hyper-parameter values, based on their assumptions and experience. This is indeed an expensive and tedious task. Meta-learning and autoML methods are used to cope with these issues.

During the last five years, autoML has attracted multiple researchers due to the importance of its application and became a *hot* topic. It consists of an automated process that

[1] Its goal is to find optimal neural architecture in a predefined search space to maximize the model performance.

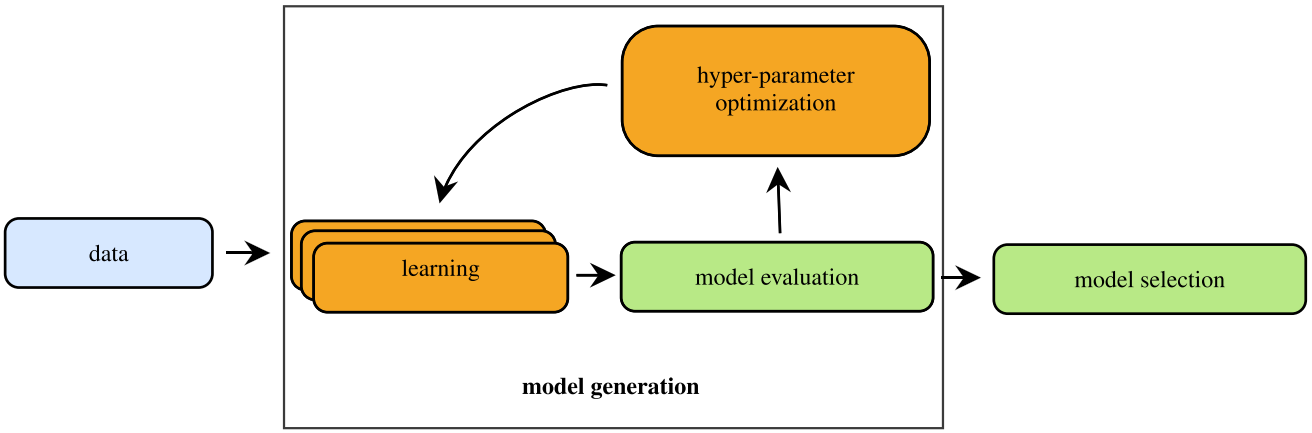

Fig. 1 Overview of the automated machine learning pipeline

performs algorithm selection and tunes hyper-parameters to improve the predictive performance of their model.

2.1 Challenges

Building high-quality machine learning models through autoML is an iterative, resource-intensive, and time-consuming process that involves different components (see Fig. 1). In the following, we list some challenges and/or problems that need to be tackled to improve the state-of-the-art ML algorithms:

- *Search space.* Defining a search space range for an hyper-parameter is not trivial. For instance, with imbalanced data distributions the search process may easily fall into local optima [71]. Often these search spaces are chosen arbitrarily without any validation leading to either inflated search spaces or spaces missing well-performing regions. In both cases, the AutoML process might be unable to find optimal hyper-parameter values. To cope with this issue, meta-learning (see Sect. 3) can be used to assess the importance of single hyper-parameters allowing to remove unimportant hyper-parameters regions from the search space or identify promising regions [36, 61,81].
- *Cold-start.* An important issue is the cold-start problem where the process may start with the worst model and/or a bad configuration, and spend too much time to get better results. This issue is related to the previous challenge to address. To warm-start the search process, one way is to use meta-learning where, based on data similarities, we start the search by a promising configuration (the one obtained on a very similar dataset). In a recent work [24], [24] showed the gain in terms of performance when using a meta-learner within an autoML system.
- *Running time.* The time is an important evaluation axis in autoML systems that approximately tells us how long do we wait before obtaining the recommended pipeline (i.e., model, hyper-parameter values, etc.). It can be restricted by setting a time limit that, when exceeded, the algorithm is then terminated and the best configuration obtained so far is selected [24].
- *High-dimensional data.* In several domains, such as biology and text documents, data may be high-dimensional which can potentially affect the performance of any algorithm, especially in terms of time and memory. To be more efficient and alleviate the use of resources, a dimension reduction process is then recommended in such cases through the feature extraction or selection component in the autoML pipeline as a preprocessing process which will precede the model generation component (see Fig. 1).
- *Scalability.* It is important to know if large datasets can be handled efficiently. Since several algorithms with different configuration will be simultaneously executed, so frameworks should offer parallelism or distributed computation to scale-up [21]. Most of the automated solutions coupled with machine learning library, e.g., Weka,[2] scikit-learn,[3] work on a single node which makes them inapplicable on large big data. In real-life practice, the scale of data generated daily is increasing continuously which led to the introduction of various distributed machine learning systems (e.g., Spark MLib [52]). Distributed solutions for the automated frameworks of autoML need to be explored to cope with tuning models over large datasets.
- *Evaluation metrics.* The models generated by the different algorithms selected in an automated framework need to be evaluated by measuring their effectiveness accord-

[2] https://www.cs.waikato.ac.nz/ml/weka/.

[3] https://scikit-learn.org/.

ing to a quality metric (or more than one). However, the choice of the measure is non-trivial since it often depends on different factors, such as the learning task (e.g., supervised or unsupervised), type of data, presence/absence of data labels, or data imbalance. The choice of the metric is crucial because the final model and its configuration are selected based on it. Nevertheless, there exists no unsupervised evaluation metric for the evaluation of the unsupervised anomaly detection algorithms. More research efforts and novel solutions are required to tackle the challenge of automatically building and tuning unsupervised anomaly detection algorithms.

– *Data streams.* Selecting the most appropriate algorithm and tuning its hyper-parameters is a difficult task when dealing with evolving data streams; therefore, automated approaches have not yet been well explored in the streaming area. Besides, many of these autoML approaches require several passes over the data, to come up with the best pipeline, and are not able to handle concept drifts. Incremental strategies and tuning techniques are required to make this automated task suitable to evolving data streams.

The aforementioned challenges are common between the offline—or batch—and the stream settings. However, the ML community started addressing them in the batch setting, but very few studies are proposed in the very challenging streaming context where more efforts should be made. This is because the latter environment has more requirements to take into account, such as processing in one-pass, i.e., an instance is processed only once and will be discarded.

2.2 Model generation

AutoML can be viewed as the process that makes the machine learning tasks easier by avoiding manual model selection and hyper-parameters tuning for non-machine learning experts as well as machine learning experts.

As shown in Fig. 1, model generation contains two main components, learning algorithms used (e.g., k-nearest neighbors, random forests algorithms) and hyper-parameter optimization techniques (presented in the following). For each iteration, we evaluate the model and its corresponding configuration using a quality metric (model evaluation). Other components can be added to the autoML pipeline displayed in Fig. 1, such as *feature engineering* before the use of the machine learning algorithms (learning step). This component consists in applying feature selection or feature extraction to reduce the input feature space—keep the relevant features [32], like applying principal component analysis, or linear discriminant analysis to improve the model performance. In practice, there is a crucial need for automating the dimension reduction process as processing high-dimensional data are considered as the most time consuming part.

We formalize the autoML problem as the Combined Algorithm Selection and Hyper-parameter problem (CASH) that has been addressed by different autoML approaches [3,26,43,57,72].

Definition 1 *CASH problem.* Let $\mathcal{A} = \{A^1, \ldots, A^a\}$ be a set of algorithms, each of them, $\mathcal{A}^i$, has a set of hyper-parameters in a domain Λ^i. So, if an algorithm A^i has p hyper-parameters, then the overall hyper-parameter space is denoted as $\Lambda^i = \Lambda^i_1, \ldots, \Lambda^i_p$. Given a dataset X, the goal is to roughly find the algorithm and hyper-parameter setting that minimizes the following loss:

$$A^\star, \lambda^\star \in \arg\min_{A^i \in \mathcal{A}, \lambda \in \Lambda^i} \frac{1}{K} \sum_{j=1}^{K} \mathcal{L}(A^i_\lambda, X^j_{\text{train}}, X^j_{\text{valid}}), \tag{1}$$

where $\mathcal{L}(A^i_\lambda, X^j_{\text{train}}, X^j_{\text{valid}})$ measures the loss obtained by algorithm A^i with hyper-parameter λ on X^j_{valid} when trained on X^j_{train} with K cross-validation folds.

Popular approaches and solution in automated optimization approaches include ParamILS [34], Sequential Model-based Algorithm Configuration (SMAC) [35], Gender-based genetic algorithm (GGA) [5], and Iterated Race (Irace) [51]. Irace is a well-known approach that uses an iterated racing procedure where the worst configurations are replaced by new ones for each iteration, also called race, by finding the most appropriate settings given a set of instances [51]. Algorithms based on the irace technique have some limitations. The most notable one is that irace is very costly in terms of time consumption and is primarily proposed for applications where the execution time is not an important axis. Though most of the applications commonly need to be efficient in terms of computational resources.

SMAC [35] is a tool for optimizing algorithm hyper-parameters across a set of instances. The main core consists of performing Bayesian optimization in conjunction with a simple racing mechanism on the instances to efficiently decide which of the two configurations performs better [35]. One main advantage of the SMAC is its ability to discard the worst performing parameter configurations quickly after the evaluation.

In [34,34] proposed ParamILS, an iterated local search method, for automatic hyper-parameter tuning. ParamILS starts by evaluating the default parameterization and some other configurations on a subset of instances, then the best configuration among those will be maintained and tested on a different subset of data. However, this method works only on categorical hyper-parameters and requires a discretization

process for numerical hyper-parameters to a finite number of values (an input user parameter). All continuous hyper-parameters have to be discretized.

The GGA [5] conducts a population-based local search whereby the separation of a competitive and a non-competitive gender balances exploitation and exploration of the parameter space. It is considered as one of the most competitive available tuners that is able to handle any type of parameter.

3 Meta-learning

Another application scenario is meta-learning which is the science that observes how different machine learning methods perform on a different set of datasets. Given a new unknown machine learning task, AutoML methods usually start from scratch to build a machine learning pipeline. Instead, models in this class learn from this experience to process new observations faster. The more similar the datasets are, the more types of meta-data we can leverage to define data similarity through meta-features [70,78,86] which describe characteristics of the datasets. There are different definitions of meta-learning, but all refer to the use meta-knowledge about learning method performance to improve the performance—or selection—of learning algorithms [78,80]. Thus, we do not need to start entirely from scratch to build autoML systems since promising configurations for a new unknown ML task are constructed leading to faster convergence with less trial and error.

Figure 2 shows a schema that summarizes how meta-learning works. Based on the observation of the performance of different machine learning algorithms with various configurations on different datasets, meta-learning extracts the promising configuration(s) with the highest performance for a new unknown dataset using similarities between the new data and training data (used within the meta-learners). [78] provides a survey exclusively on meta-learning.

More importantly, as mentioned before, meta-learning is complementary to autoML where it can be used to warm-start the search procedure for optimal models, i.e., the search procedure in autoML is more likely to start with promising initialization (since only the more relevant regions of the configuration space are explored), which results in a considerable boost in efficiency.

4 General automated approaches

In this section, we introduce the autoML and meta-learning approaches that have been proposed in machine leaning. The approaches have a common goal that intends to improve the performance of classical machine learning algorithms and could be also complementary.

In Fig. 3, we present an abstractive taxonomy that subdivides the autoML approaches into supervised, unsupervised, and semi-supervised approaches where we associate an example for each category. We review the novel approaches in each category with a particular attuning to the proposed solutions for unsupervised anomaly detection.

4.1 Supervised autoML

The basic autoML algorithms initially proposed for the supervised learning have been thoroughly discussed in few recent surveys on autoML [21,25,32].

Brazdil et al. presented the first solution based on meta-learning which aims to determine the closest training datasets to a given test dataset based on features' similarity, afterward it selects the best model by ranking the algorithms based on the performance of the neighboring datasets. However, this

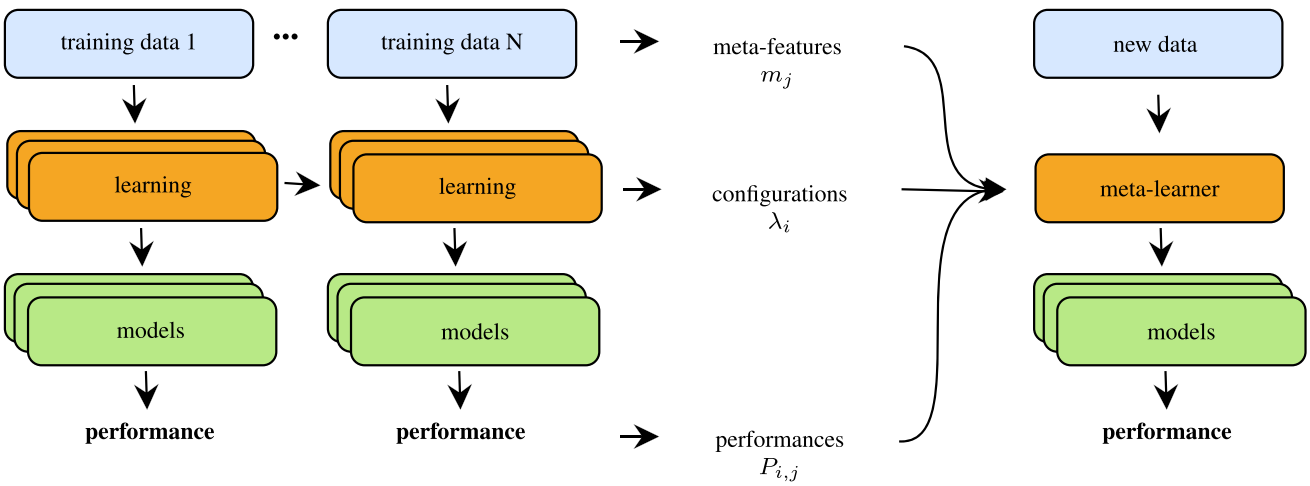

Fig. 2 Meta-learning schema: learning to learn

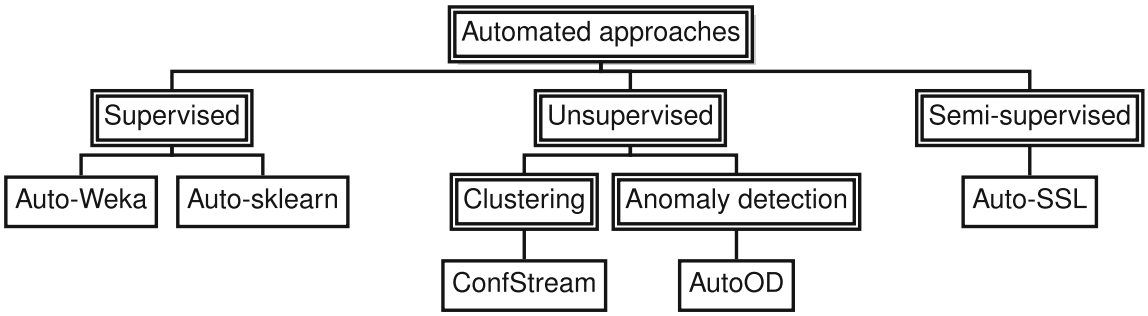

Fig. 3 Taxonomy of autoML with some well know approaches and tools

method works exclusively on classification algorithms and is not able to deal with unlabeled data.

It is in the last five years that the topic started to attract the attention of researchers, where multiple solutions and tools have been developed to serve the autoML problem. Auto-Weka[4] [44,72] is one of the earliest open source autoML solutions that has been implemented in Java on top of the well-known Weka machine learning software. Auto-Weka applies a Bayesian optimization using SMAC [35] to search over different classification and regression algorithms and their hyper-parameter settings. Nevertheless, there has been no new release since 2017 and auto-Weka is designed to work with supervised learning only excluding modern neural networks from its employed machine learning methods.

Similarly, auto-sklearn[5] [26] has been implemented in Python on top of the popular scikit-learn. More recently, auto-sklearn 2.0 [24] has been proposed to improve the efficiency of auto-sklearn using portfolios through meta-learning. It applies similar optimization techniques as auto-Weka and opts to improve the optimization process and to warm-start a Bayesian optimization by identifying similar datasets and using knowledge collected from the past. Similar to auto-Weka, this framework does not support deep neural networks nor unsupervised algorithms.

Hyperopt-Sklearn [43] is another autoML framework that is based on the scikit-learn machine learning library. This open-source framework uses hyperopt [9], which supports different optimization techniques, to define a search space over the possible configurations of scikit-learn learning algorithms. Auto-Keras[6] [39] is also an open source library for autoML that relies on a popular deep learning library called Keras. Auto-Keras provides functions to automatically search for hyper-parameters and architecture of deep learning models. This framework can tackle supervised algorithms and handle images and text data. Other software applications and library are proposed for supervised autoML as TPOT [58], H2O,[7] Microsoft Azure autoML,[8] and Amazon SageMaker.[9] We redirect the reader to a recent survey [32] on most of the proposed tools for supervised learning.

These tools have been proposed in static or batch environments under the assumption that the entire dataset is available before starting the process. Few approaches have been developed in the streaming framework where instances are not available *a priori*, and are coming one after one in an incremental way [7,53,75–77,79].

The first automated model selection approach uses meta-learning that periodically selects the best performing classifier based on the stream's characteristics [75]. Similarly, the BLAST [76,77] algorithm uses an ensemble of classifiers and selects the one that obtained the most accurate prediction on the last window for the next window of the evolving stream.

Different hyper-parameter optimization solutions are proposed in the batch setting, such as random search, grid search, Bayesian optimization, and heuristics. However, as far as we know, in the stream environment only one supervised solution exists named Self-Parameter Tuning (SPT) which adapts the Nelder-Mead technique [56] to data streams regression. A recent classification method has been proposed that tunes the parameters of the stream k-nearest neighbors algorithm by extending the SPT algorithm [79]. This approach could be generalized by including other classifiers in order to perform model selection on top of the hyper-parameter tuning task.

We recall that the aforementioned systems and methods deal with supervised methods only where instances are used to optimize the configuration and select better performing model (more accurate) using ground truth class labels and supervised learning objective functions.

[4] http://www.cs.ubc.ca/labs/beta/Projects/autoweka/.

[5] https://www.automl.org/automl/auto-sklearn/.

[6] https://autokeras.com/.

[7] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html.

[8] https://azure.microsoft.com/en-us/services/machine-learning/automatedml/.

[9] https://aws.amazon.com/fr/sagemaker/.

4.2 Clustering

Currently, autoML is completely focused on supervised learning. Even though some approaches may be applicable for the unsupervised tasks. In this section, we aim to review the research studies that have been proposed for the clustering task, where the hyper-parameter configuration and the model selection should be done without prior knowledge of data labels.

For the clustering task, the accuracy—or any other quality—measure (in the supervised task) can be obtained using external and internal validity measures, e.g., rand-index and silhouette coefficient, that do not require the access to the class label. Clustering algorithms evaluation is therefore made by measuring how similar instances in the same cluster are to each other and/or how dissimilar are instances belonging to different clusters. Similar to the supervised task, few attempts to improve the performance in the clustering task use ensemble and meta-learning (in this context also called meta-clustering) techniques [28,30,37,38,69].

To the best of our knowledge, De Souto et al. proposed the first meta-learning approach in the clustering task which provides a ranking for candidate algorithms after being applied to a given dataset. During the learning phase, the meta-learner learns from the datasets in a meta-train database.[10] In the test phase, given a new dataset, the meta-features that describe the data are extracted and based on their similarities to the existing datasets, the meta-learn produces a ranking. The latter could help non-expert users in the algorithm selection task [19]. Following this approach, few methods [23,54,60,65] have been proposed that are similarly based on meta-learning based on different meta-feature solutions.

A recent method, called ConfStream [15], has been proposed in the stream framework for automated algorithm configuration of clustering algorithms for evolving data streams. Inspired by the Blast method [77], ConfStream also uses an ensemble of a diverse set of stream clustering algorithms with different configurations. Based on the evaluation of every configuration and algorithm, the best performing configuration will be used to create a new one from it that will replace one of the configurations in the ensemble (if it is good enough). Carnein et al. u used the silhouette width measure to evaluate the clustering quality of every configuration. Therefore, the configuration with the best cluster quality is selected for the next window of the stream. However, similar to the Blast method [77], ConfStream uses periodically a small part of the stream (window) to select the model and the configuration that obtain the best performance and use them for the next window from the stream, and so on. Albeit it adopts a batch-incremental strategy, ConfStream is a very promising method for unsupervised autoML for data streams.

4.3 Anomaly detection

Anomaly detection,[11] also called outlier detection, is one of the cornerstone problems in data mining that, roughly speaking, consists of identifying the instances which deviate substantially from the rest of the data that show "normal" behavior. This problem has been widely studied over the last few decades [16,33] resulting in a large number of algorithms with varying levels of effectiveness. This task is crucial in a variety of application domains, such as fraud detection, intrusion detection, and healthcare system monitoring. Due to the absence of ground-truth labels in such applications, the unsupervised anomaly detection received an increasing attention in the recent years and overtook its supervised counterpart [62].

Generally, anomalies can be categorized as *point*, *contextual* or *collective* outliers. (i) Point outlier represents an instance that significantly deviates from all the instances in the dataset; (ii) contextual outlier is an instance that significantly deviates from the instances in a context, e.g., period in a time series; and (iii) collective outlier which is a subset of instances that significantly deviates from the entire dataset, i.e., individual instances of the subset might not be outliers but all of them, together, constitute a collective outlier.

In the literature, there exist a number of studies devoted to unsupervised anomaly detection, which is a relatively common scenario, in that ground truth labels are often unavailable. Detection algorithms can be essentially grouped in three classes: (i) *proximity/nearest neighbor*-based methods (e.g., *k*-NN, LOF [12]); (ii) *probabilistic/linear*-based methods (e.g., PPCA [73], OCSVM [67]); and (iii) *ensemble/isolation*-based methods (e.g., iForest [50], RHF [62]). The latter category include algorithms that, instead of profiling normal instances, isolate the anomalies by means of recursive splitting over the data through a random tree and by generating isolation forests.

In [31], authors compare the proximity-based anomaly detection algorithms on a diverse set of datasets and conclude that the initial assumption of whether the anomalies in the datasets are global or local is of great impact. They, therefore, recommend the use of global anomaly detection methods if no further information about the nature of anomalies in the dataset is provided. Similarly, a more recent comparative study [20] compares the most recent algorithms but describes the models generalization capacity using labels that most of the time are not available. The use of class labels with

[10] Meta-train database is a collection of historical datasets.

[11] In compliance with literature terminology, we use the terms *anomaly* and *outlier* interchangeably.

unsupervised anomaly detection algorithms is very common especially for evaluation.

As anomaly detection methods are generally unsupervised and typically lack of class labels, clustering techniques are often used to perform outlier detection, either for detecting anomalies or evaluation. Most of the current studies on anomaly detection focus on developing approaches that perform well on different data types and settings [1]. Nevertheless, these approaches tend to be highly sensitive to heir hyper-parameterization that might potentially affect the final results. To cope with this issue, lately researchers started tackling a more general problem for anomaly detection that concerns automated hyper-parameter tuning and model selection, i.e., what algorithm do we need to use on a given dataset [64].

4.3.1 Toward automated anomaly detection

A position paper that describes the main challenges associated with outlier ensembles was published in [2]. Aggarwal discussed an ensemble approach that combines anomaly scores of multiple anomaly detection algorithms in a sequential way while refining the data based on subspace analysis (e.g., subset selection, attribute-subset selection). The main idea of a sequential ensemble is that the successive algorithmic execution could help to provide concise insights regarding the scores. The advantages of ensemble-based methods are significant with the subspace analysis that induces further diversity.

In Table 1, we summarize some key characteristics of representative methods in the automated machine learning field for unsupervised anomaly detection. Since these methods are evaluated on diverse datasets, it is difficult to have an universal meta-analysis of their empirical performance. Instead, some main observations w.r.t. to the model design are summarized as follows: (i) *Search* indicates the searching techniques used; (ii) *Evaluation* refers to the nature of the evaluation techniques used to assess the proposed approaches; (iii) *Incremental* indicates if the method is incremental and could be applied on evolving data streams or not; and (iv) *Pros* and *Cons* represent the advantages and disadvantages of each method, respectively.

In [85], an outlier detection framework has been developed (Python toolbox), named PyOD, for scalable anomaly detection that includes more than 20 detectors with established outlier ensembles and network-based approaches. Nevertheless, this framework does not tackle problems as the optimal pipeline design, i.e., searching for optimal hyper-parameter setting. In [48], an automated Python system for Outlier Detection with Database Support (PyODDS) has been presented to automatically optimizes the detection pipeline for new datasets. It gives the ability to access to a wide range of anomaly detection algorithms. Given a new dataset, PyODDS searches for an optimal algorithm with its configuration based on a defined search space, i.e., a range of hyper-parameter settings. The main weakness of this framework is that despite the use of unsupervised anomaly detection algorithms in the search space, the comparison of the different algorithms and configurations is made using the F1-score, which is a supervised metric. Moreover, this system does not address the cold-start challenge, it just starts with the default configuration for each of its members and could not be used with evolving data streams since the used algorithms are static.

A framework called Locally Selective Combination in Parallel outlier ensembles (LSCP) has been introduced for unsupervised anomaly detection in [84]. LSCP starts with a diverse set of detection algorithms initialized with a range of hyper-parameters. All the detectors are firstly trained on a training set of instances and derive an outlier score matrix. Given a test instance, LSCP defines the local region by consensus of the nearest training instances (that appear more than $t/2$ times) in randomly selected feature of t sub-spaces. Then, it identifies the best performing detector in this local region by measuring the similarity between base detector scores and the pseudo ground truth (obtained by retrieving values associated with the local region). This method presents some limitations that makes it not ideal related to (i) the use of the same base algorithm with different configurations, (ii) the performance that may degrade when several attributes are irrelevant in a high-dimensional space, and (iii) the high time complexity due to the nearest neighbors search which is costly. This can be avoided by using dimension reduction techniques in order to extract relevant attributes and instead of computing pairwise distances in high-dimensional space, they can be obtained from the lower-dimensional representation of data.

A novel automated framework has been proposed recently for anomaly detection on time series data, named automated Time series Outlier Detection System (TODS) [45]. The functionalities provided through this system include several components, such as data processing, feature extraction, and various detection algorithms for time series. Authors claim that this system is able to provide the most suitable algorithm given a dataset (based on the F1-score) and is easily usable via a graphical user interface, but they do not precise the search technique employed by this system to discover the suitable pipeline.

In a recent work [86], Zhao et al. proposed MetaOD, a meta-learning approach for unsupervised outlier detection that performs model selection. This approach stands on the past performance of different detection models on existing anomaly detection benchmark datasets and carries over that to automatically select an efficient model for a new dataset. In fact, authors used an effective way to seize the similarity between the input and the historical datasets based on feature

Table 1 Overview of different unsupervised automated methods for anomaly detection.

Method	Search	Evaluation	Incremental	Pros	Cons
PyODDS [48]	Global optimization	Supervised	No	It is scalable and contains several AD algorithms	It does not address cold-start issue
LSCP [84]	Grid search	Supervised	No	It uses a similarity measure to assign ground truth labels	It is composed of only one base model where the performance may degrade when data have irrelevant attributes. It is also costly due to the neighbors' search
TODS [45]	Grid search	Supervised	No	It includes several components and detectors to the pipeline	The used search technique is not mentioned. It also uses a supervised evaluation measure
MetaOD [86]	–	Unsupervised	No	It handles the cold-start issue using meta-learning and meta-features	It discretizes the parameter space and is not a fully supervised method
Meta-AAD [83]	–	Unsupervised	Yes	It performs model selection using reinforcement learning and is applied to any unlabeled data. It is efficient since it uses only 6 features instead of the whole feature space	Many features are ignored and so less information is available
AutoOD [46]	RNN	Supervised	No	It is also built using reinforcement learning for tuning	It works with deep anomaly detection algorithms which are costly, and has been evaluated using ground truth labels

characteristics (meta-features) that capture statistical properties of data distributions. Given a new dataset, MetaOD identifies the datasets in the meta-train database that have similar characteristics to the new one, thereafter it focuses on the models (algorithms and their corresponding configuration) that perform well on those datasets. This task is similar to the recommendation task where we recommend to a new user the items liked by similar users. However, the proposed method discretizes the hyper-parameter space to make the overall search space tractable and the models are presented as {model, configuration} pairs. Moreover, the models trained in the meta-learn are evaluated using a supervised measure which makes MetaOD not fully unsupervised.

In [83], an Active Anomaly Detection framework (Meta-AAD) has been proposed to select the most proper model to optimize the number of detected anomalies. Meta-AAD leverages deep reinforcement learning to train a meta-learner that performs model selection. Finally, the trained models can be applied to any unseen unlabeled dataset for anomaly detection without the need for further tuning. Zha et al. empirically extracted only 6 features (in the meta-features extraction step) to compare similarities with the new dataset. But it would be nice to see the behavior of this framework using a feature extraction or selection technique with different feature set sizes, especially with high-dimensional data, 6 features are not enough.

In the same context, other simple meta-learning methods have been proposed to select the well-performing model given a test dataset [14,82]. For instance, Instance-Specific Algorithm Configuration (ISAC) [42] clusters the meta-train database and given the new data, it picks the closest cluster and selects the best model. However, this class of techniques is limited to one model class and not to a general case where we can select among different heterogeneous algorithms.

In [89], Zimek et al. discussed the underlying challenge and review some anomaly detection methods, mainly the ones based on approximate neighborhoods, that handle such data. A recursive binning and re-projection approach have been proposed in [29] that starts by recursively binning the data into k clusters, using a clustering method such as k-means. The data in each bin from the k bins is also clustered into k bins and so forth until a bin does not contain enough instances. Afterward, for each resulting bin, the neighbors are listed and authors used a nested loop to extract the n outliers. In this work, authors assumed that the number of anomalies n is an input parameter, which is generally not the

case for unsupervised anomaly detection algorithms. They also tested with a unique value, it would have been better if a range of different values of n has been experimented for results' consistency.

4.3.2 Neural network-based anomaly detection

In [8], Bergmann et al. applied the structure of convolutional Auto-Encoders for unsupervised defect segmentation on image data. More specifically, it utilizes the loss function based on structural similarity, and successfully examines inter-dependencies between local image regions to reveal the defective regions.

AnoGAN [66], a deep convolutional generative adversarial network, has been proposed to identify the anomalous image data. It is an unsupervised anomaly detection algorithm that learns a manifold of normal anatomical variability, and maps images to a latent space to estimate the anomaly scores. Applied to new images, the model assigns label and scores images while indicating their fit into the learned distribution.

In the same context, a region-based method for the localization and detection of anomalies in image data is proposed [55] which is based on convolutional neural networks and self-similarity. More specifically, the method exploits convolutional neural networks for describing the visual content of each subregion and the abnormality degree of each image region is obtained by estimating its similarity to a dictionary of anomaly-free subregions in a training set.

Building a strong deep neural network system is not a trivial task since it relies on human expertise to tune the hyper-parameters and design the neural architecture, which is time-consuming. In [46], authors proposed an Automated Outlier Detection (AutoOD) framework that aims to find the optimal neural network model within a predefined search space for unsupervised outlier detection. It is built upon a reinforcement learning-based approach that uses a Recurrent Neural Network (RNN) controller to choose blocks from the search space. The optimal model with the best performance on the validation set is used for the anomaly detection task. Nevertheless, this AutoOD framework is designed for the batch setting where active learning cannot be performed and only adapted to work with deep-structured outlier detection approaches, e.g., deep auto-encoder. Li et al. also used the ROC as quality measure which considers ground truth labels.

We redirect the reader to [59] for a survey that reviews the studies of deep anomaly detection with a comprehensive taxonomy of these methods.

4.3.3 Evaluation metrics

Quality assessment for the anomaly detection task is different from clustering since anomaly detection models are ranking the instances instead of grouping them as in the clustering task. Most of the aforementioned methods and tools use unsupervised anomaly detection algorithms, but their evaluation is performed using measures that require the access to data label. However, lack of access to ground truth labels in anomaly detection task makes evaluation and comparison inapplicable. Probably that is one of the main reasons why the state of the art on meta-learning, ensembles, and autoML for fully unsupervised anomaly detection is not very advanced [2].

In [88,89], authors discussed the most widely used quality measure to evaluate anomaly detection algorithms which is based on the Receiver Operating Characteristic (ROC) curves. This measure consists in a graphical plot that illustrates the true positive rate against the false positive rate. The resulting curves are usually turned into a measure by computing the Area Under this Curve (AUC) (also known as average precision) [88]. Besides, these quality measures require instances with known class labels, i.e., annotated anomalies. Moreover, The ROC and AUC measures completely neglect the anomaly scores as if they do not have any meaning.

Other measures, discussed in [68], evaluate anomaly detection algorithms using instances ranking and taking score information into account. Different studies use the top k anomalous instances [40]; however, k is an input parameter and the number of anomalies in a given dataset is supposed to be unknown. The authors do not provide an efficient way of how to choose the value of this key parameter. In other applications, we might fix a threshold and classify all the instances whose anomaly score exceeds it as outliers and the rest of the instances as normal. The major weakness of these metrics lies in the fact that they only evaluate the ranking without considering the corresponding scoring. If they are considering the score, we might need additional parameters such as the threshold value.

4.4 Semi-supervised AutoML

Up to this point, studies on autoML are mostly focusing on supervised machine learning problems using exclusively labeled data to perform classification or regression. However, in multiple real-world scenarios, unlabeled data may be available in addition to the labeled data and can help in building better learners. A very limited number of studies in automated semi-supervised learning can be found in the literature lack of standardized tools and systems as in supervised learning (e.g., Weka and scikit-learn).

Li et al. presented auto-SSL [47], an automated learning system for semi-supervised learning. First, they incorporated meta-learning with enhanced meta-features to help discriminating the instantiations which are likely to perform well. Then, under the assumption that high-quality hyper-

parameters lead to good predictive performance on unlabeled data with large margin separation, auto-SSL leverages a large margin separation method to fine-tune the hyper-parameters. This tool has been tested on 40 diverse datasets and compared against Support Vector Machines (SVM), auto-sklearn, Class Mass Normalization (CMN) [87] and Transductive SVM [41] where it shows discrete performance.

In [74], a semi-supervised co-ensembling method has been proposed that uses unlabeled instances to improve the performance of high-quality ensembles of state-of-the-art learners (from supervised autoML system). The procedure starts by employing semi-supervised learning after constructing supervised classifiers using an auto-sklearn system, where an ensemble of classifiers is firstly trained on labeled instances, then sub-ensembles of classifiers are used to pseudo-label the data for the remaining classifiers. Each classifier is then re-trained on the labeled and pseudo-labeled instances. This strategy achieved good predictive performance improvements on multi-class classification problems, but performance in terms of computational resources have not been addressed in the paper. Re-training ensembles in a multi-iteration way can be very costly.

5 Research directions

Although in the recent years, there has been increasing research efforts to tackle the challenges of the automated machine learning hot topic, however, several open challenges and research directions should be tackled to obtain efficient and effective AutoML systems. Currently, AutoML is completely focused on the supervised learning. Even though some proposed methods may be applicable for the unsupervised learning, researchers always test their proposed approaches for supervised learning. Dedicated research for unsupervised tasks, such as anomaly detection, could boost the development of the autoML topic for currently uncovered problems. Moreover, specialized methods could enhance the performance for those tasks.

In this section, we highlight some of these challenges and voids that need to be addressed to improve the state of the art of the autoML topic in the context of unsupervised anomaly detection.

As defined previously, the meta-learning includes any type of learning based on prior experience with other datasets. The field of meta-learning addresses the question "what learning methods work well on what data." However, defining the similarity between data is a key overarching challenge. Besides, when a new dataset represents a completely different event or some random noise, leveraging on prior experience will be ineffective. That is why, one needs to use a large diverse set of datasets in the meta-train database.

To extract the similarity between an input dataset and the datasets available in the meta-train database, we usually use meta-features that effectively describe the characteristics of data. Example of meta-features: (i) *simple* as the number of instances or features; (ii) *statistical* as kurtosis, mean, standard deviation of the attributes; or (iii) *information theoretic* such as class entropy, entropy of attributes, etc.We redirect the reader to surveys on the most commonly used meta-features in [70,78,86]. Besides, the base-learner(s) used for anomaly detection may be unsupervised but the meta-learning is supervised. Hence, a question which naturally arises is: *is it possible to make a meta-learning unsupervised?*. It is a question that has not been answered yet. We believe that this goal might be realizable if one starts by exploring the use of meta-features in order to come up with an unsupervised metric for anomaly detection.

Generally, supervised autoML systems and hyperparameter optimization problems are formulated as a black-box objective function that opts to improve the precision. Hence, many of these optimization techniques can be applied to unsupervised learning (including unsupervised anomaly detection and clustering) if we could find an appropriate objective function to assess the model's performance in an unsupervised way.

In [27], authors introduced a clustering-based approach that extends the k-means algorithm to perform both clustering and anomaly detection by adding a cluster to the k-means that contains anomalies. So, given the desired number of clusters k, the proposed approach partitions the data into $k + 1$ clusters. It uses a parameter that concerns the maximum number of outliers to prevent the approach from assigning many instances to the outlier group. If unknown, this parameter is fixed using a percentage from the size of the dataset (e.g., 0.5% of the dataset). Similar work has been proposed recently [49] that also performs outlier removal. We think that these unsupervised anomaly detection algorithms could be further explored, generalized, and used for a fully unsupervised autoML detection strategy, including the evaluation based on outlier removal.

In practice, there are still machine learning problems which have not been well explored yet by autoML due to their scale and which might require novel approaches. Here, by scale we mean both the size of the configuration space and the cost of model evaluations. For instance, very few works have been proposed for big data streams because of their evolving and incremental nature [7,15,79]. Most of the CASH solutions are proposed for the batch settings and more efforts need to be done to adapt them to the streaming framework. Another limitation of the automated solutions for machine learning (e.g., auto-Weka, auto-sklearn) is that they are linked to machine learning library (e.g., Weka, scikit-learn) that work exclusively on a single node which makes them inapplicable on massive datasets. Given the necessity of distributed

machine learning platforms, we are looking forward to new methods and strategies that fully exploit large-scale compute clusters. Except neural network solutions [22], so far no method has demonstrated scalability to several workers. We expect that more sophisticated and generalized approaches are required to further scale autoML to interesting and real problems.

6 Concluding remarks

In recent years, we witnessed the continuously ever-growing amount of data and the increasing proliferation of different groundbreaking machine learning models to analyze, mine and explore these data. Despite this, a number of tasks, as the selection of the best machine learning model and the manual tuning of their hyper-parameters, are still highly dependent on expert interventions.

This lately raised the interest of researchers to exploit the aspects of machine learning that can be automated and, hence, build systems capable of autonomously and automatically solving these tasks. Different solutions have been proposed in the literature. However, there are still several open challenges and research directions that needs to be tackled to achieve the ultimate goals of the AutoML and the question of whether an efficient and fully generalizable autoML framework is still remain open.

In this review, we presented the automated machine learning topic and provided an overview of the most promising and recent algorithms and tools in this field. After briefly reviewing autoML as a whole, providing a formal definition for it, this survey focuses on the automated approaches proposed for the supervised, unsupervised and semi-supervised learning. Among the different unsupervised tasks, a special emphasis is given to methods attempting to automatically tackle anomaly detection. We hope that this review provides useful resources for researchers and practitioners to understand the challenges, recent advances with more attention to anomaly detection, as well as some open research directions in the autoML topic.

Acknowledgements This work has been carried out in the frame of a cooperation between Huawei Technologies France SASU and Télécom Paris (Grant no. YBN2018125164).

Author Contributions MB, FS, MS and AP were involved in conceptualization. MB was involved for writing and also responsible for tables, figures and revisions.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interests.

References

1. Aggarwal, C.C.: Outlier analysis. In: Data Mining (2015)
2. Aggarwal, C.C.: Outlier ensembles: position paper. ACM SIGKDD (2013)
3. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: International Conference on Knowledge Discovery and Data Mining SIGKDD (2019)
4. Alpaydin, E.: Introduction to Machine Learning. MIT press (2020)
5. Ansótegui, C., Sellmann, M., Tierney, K.: A gender-based genetic algorithm for the automatic configuration of algorithms. In: International Conference on Principles and Practice of Constraint Programming, pp. 142–157. Springer (2009)
6. Bahri, M., Bifet, A., Maniu, S., Gomes, H.M.: Survey on feature transformation techniques for data streams. In: IJCAI (2020a)
7. Bahri, M., Veloso, B., Bifet, A., Gama, J.: Automl for stream k-nearest neighbors classification. In: IEEE BigData (2020b)
8. Bergmann, P., Löwe, S., Fauser, M., Sattlegger, D., Steger, C.: Improving unsupervised defect segmentation by applying structural similarity to autoencoders (2018) arXiv preprint arXiv:1807.02011
9. Bergstra, J., Yamins, D., Cox, D.D., et al.: A python library for optimizing the hyperparameters of machine learning algorithms. In: SciPy, vol. 13, p. 20 (2013)
10. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
11. Brazdil, P.B., Soares, C., da Costa, J.P.: Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. Mach. Learn. **50**(3), 251–277 (2003)
12. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: ACM SIGMOD International Conference on Management of Data, pp 93–104 (2000)
13. Bughin, J., Hazan, E., Ramaswamy, S., Chui, M., Allas, T., Dahlstrom, P., Henke, N., Trench, M.: Artificial intelligence: the next digital frontier? McKinsey Global Institute Report (2017)
14. Burnaev, E., Erofeev, P., Smolyakov, D.: Model selection for anomaly detection. In: ICMV (2015)
15. Carnein, M., Trautmann, H., Bifet, A., Pfahringer, B.: confstream: Automated algorithm selection and configuration of stream clustering algorithms. In: LION (2020)
16. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput Surv (CSUR) **41**(3), 1–58 (2009)
17. Chapelle, O., Scholkopf, B., Zien, A.: Semi-supervised learning. Trans Neural Netw (2009)
18. Cunningham, J.P., Ghahramani, Z.: Linear dimensionality reduction: survey, insights, and generalizations. JMLR (2015)
19. De Souto, M.C., Prudencio, R.B., Soares, R.G., De Araujo, D.S., Costa, I.G., Ludermir, T.B., Schliep, A.: Ranking and selecting clustering algorithms using a meta-learning approach. In: IJCNN, IEEE (2008)
20. Domingues, R., Filippone, M., Michiardi, P., Zouaoui, J.: A comparative evaluation of outlier detection algorithms: experiments and analyses. Pattern Recogn. **74**, 406–421 (2018)
21. Elshawi, R., Maher, M., Sakr, S.: Automated machine learning: state-of-the-art and open challenges (2019)
22. Elsken, T., Metzen, J.H., Hutter, F., et al.: Neural architecture search: a survey. J. Mach. Learn. Res. **20**(55), 1–21 (2019)
23. Ferrari, D.G., De Castro, L.N.: Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods. Inf. Sci. **301**, 181–194 (2015)
24. Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., Hutter, F.: Auto-sklearn 2.0: The next generation. arXiv preprint arXiv:2007.04074 (2020)

25. Feurer, M., Hutter, F.: Hyperparameter optimization. In: Automated Machine Learning, pp 3–33 (2019)
26. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M., Hutter, F.: Auto-sklearn: efficient and robust automated machine learning. In: Automated Machine Learning, pp. 113–134 (2019)
27. Gan, G., Ng, M.K.P.: K-means clustering with outlier removal. Pattern Recogn Lett (2017)
28. Ghosh, J., Acharya, A.: Cluster ensembles. DMKD (2011)
29. Ghoting, A., Parthasarathy, S., Otey, M.E.: Fast mining of distance-based outliers in high-dimensional datasets. DMKD (2008)
30. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. TKDD (2007)
31. Goldstein, M., Uchida, S.: A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PLoS ONE **11**(4), e0152173 (2016)
32. He, X., Zhao, K., Chu, X.: Automl: a survey of the state-of-the-art. Knowl. Based Syst. (2021)
33. Hodge, V., Austin, J.: A survey of outlier detection methodologies. Artif. Intell. Rev. **22**(2), 85–126 (2004)
34. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: Paramils: an automatic algorithm configuration framework. JAIR (2009)
35. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: LION (2011)
36. Hutter, F., Kotthoff, L., Vanschoren, J.: Automated Machine Learning. Springer (2019)
37. Iam-On, N., Boongoen, T.: Comparative study of matrix refinement approaches for ensemble clustering. Mach. Learn. (2015)
38. Jiang, Y., Verma, N.: Meta-learning to cluster. arXiv preprint arXiv:1910.14134 (2019)
39. Jin, H., Song, Q., Hu, X.: Auto-keras: An efficient neural architecture search system. In: ACM SIGKDD (2019)
40. Jin, W., Tung, A.K., Han, J.: Mining top-n local outliers in large databases. In: ACM SIGKDD (2001)
41. Joachims, T.: Transductive inference for text classification using support vector machines. In: Proceedings of the Sixteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc (1999)
42. Kadioglu, S., Malitsky, Y., Sellmann, M., Tierney, K.: Isac-instance-specific algorithm configuration. In: ECAI (2010)
43. Komer, B., Bergstra, J., Eliasmith, C.: Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In: ICML workshop on AutoML (2014)
44. Kotthoff, L., Thornton, C., Hoos, H.H., Hutter, F., Leyton-Brown, K.: Auto-weka 2.0: automatic model selection and hyperparameter optimization in weka. JMLR (2017)
45. Lai, K.H., Zha, D., Wang, G., Xu, J., Zhao, Y., Kumar, D., Chen, Y., Zumkhawaka, P., Wan, M., Martinez, D., Hu, X.: Tods: an automated time series outlier detection system (2020). eprint2009.09822
46. Li, Y., Chen, Z., Zha, D., Zhou, K., Jin, H., Chen, H., Hu, X.: Autood: automated outlier detection via curiosity-guided search and self-imitation learning. ICDE (2020a)
47. Li, Y.F., Wang, H., Wei, T., Tu, W.W.: Towards automated semi-supervised learning. In: AAAI (2019)
48. Li, Y., Zha, D., Venugopal, P., Zou, N., Hu, X.: Pyodds: An end-to-end outlier detection system with automated machine learning. In: WWW (2020b)
49. Liu, H., Li, J., Wu, Y., Fu, Y.: Clustering with outlier removal. IEEE Trans. Knowl. Data Eng. (2019)
50. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: international Conference on Data Mining, pp 413–422. IEEE (2008)
51. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Persp. (2016)
52. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al.: Mllib: machine learning in apache spark. J. Mach. Learn. Res. **17**(1), 1235–1241 (2016)
53. Minku, L.L.: A novel online supervised hyperparameter tuning procedure applied to cross-company software effort estimation. Empir. Softw. Eng. (2019)
54. Muravyov, S., Filchenkov, A.: Meta-learning system for automated clustering. In: PKDD/ECML AutoML workshop, pp 99–101 (2017)
55. Napoletano, P., Piccoli, F., Schettini, R.: Anomaly detection in nanofibrous materials by cnn-based self-similarity. Sensors **18**(1), 209 (2018)
56. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. (1965)
57. Nguyen, D.A., Kong, J., Wang, H., Menzel, S., Sendhoff, B., Kononova, A.V., Bäck, T.: Improved automated cash optimization with tree parzen estimators for class imbalance problems. In: 8th International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–9. IEEE (2021)
58. Olson, R.S., Moore, J.H.: Tpot: A tree-based pipeline optimization tool for automating machine learning. In: Workshop on Automatic Machine Learning, pp. 66–74. PMLR (2016)
59. Pang, G., Shen, C., Cao, L., Hengel, A.V.D.: Deep learning for anomaly detection: a review. ACM Comput. Surv. (CSUR) **54**(2), 1–38 (2021)
60. Poulakis, Y., Doulkeridis, C., Kyriazis, D.: Autoclust: A framework for automated clustering based on cluster validity indices. In: International Conference on Data Mining (ICDM), pp 1220–1225. IEEE (2020)
61. Probst, P., Boulesteix, A.L., Bischl, B.: Tunability: importance of hyperparameters of machine learning algorithms. J. Mach. Learn. Res. **20**(1), 1934–1965 (2019)
62. Putina, A., Sozio, M., Rossi, D., Navarro, J.M.: Random histogram forest for unsupervised anomaly detection. In: International Conference on Data Mining (ICDM), pp. 1226–1231. IEEE (2020)
63. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. AAAI Conf. Artif. Intell. **33**, 4780–4789 (2019)
64. Reunanen, N., Räty, T., Lintonen, T.: Automatic optimization of outlier detection ensembles using a limited number of outlier examples. Int. J. Data Sci. Anal. **10**, 377–394 (2020)
65. Sáez, J.A., Corchado, E.: A meta-learning recommendation system for characterizing unsupervised problems: On using quality indices to describe data conformations. IEEE Access **7**, 63247–63263 (2019)
66. Schlegl, T., Seeböck, P., Waldstein, S.M., Schmidt-Erfurth, U., Langs, G.: Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: International Conference on Information Processing in Medical Imaging, pp 146–157. Springer (2017)
67. Schölkopf, B., Williamson, R.C., Smola, A.J., Shawe-Taylor, J., Platt, J.C., et al.: Support vector method for novelty detection. NIPS Citeseer **12**, 582–588 (1999)
68. Schubert, E., Wojdanowski, R., Zimek, A., Kriegel, H.P.: On evaluation of outlier rankings and outlier scores. In: ICDM (2012)
69. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. JMLR (2002)
70. Sun, Q., Pfahringer, B.: Pairwise meta-rules for better meta-learning-based algorithm ranking. Mach. Learn. (2013)
71. Swirszcz, G., Czarnecki, W.M., Pascanu, R.: Local minima in training of neural networks. arXiv preprint arXiv:1611.06310 (2016)
72. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In: ACM SIGKDD (2013)

73. Tipping, M.E., Bishop, C.M.: Probabilistic principal component analysis. J. Roy. Stat. Soc. Ser. B **61**(3), 611–622 (1999)
74. Van Engelen, J.E., Hoos, H.H.: Semi-supervised co-ensembling for automl. In: Heintz, F., Milano, M., O'Sullivan, B. (eds.) Trustworthy AI—Integrating Learning, pp. 229–250. Optimization and Reasoning, Springer International Publishing, Cham (2021)
75. van Rijn, J.N., Holmes, G., Pfahringer, B., Vanschoren, J.: Algorithm selection on data streams. In: DS (2014)
76. van Rijn, J.N., Holmes, G., Pfahringer, B., Vanschoren, J.: Having a blast: meta-learning and heterogeneous ensembles for data streams. In: ICDM (2015)
77. van Rijn, J.N., Holmes, G., Pfahringer, B., Vanschoren, J.: The online performance estimation framework: heterogeneous ensemble learning for data streams. Mach. Learn. **107**(1), 149–176 (2018)
78. Vanschoren, J.: Meta-learning: a survey. arXiv preprint arXiv:1810.03548 (2018)
79. Veloso, B., Gama, J., Malheiro, B.: Self hyper-parameter tuning for data streams. In: DS (2018)
80. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Artif. Intell. Rev. (2002)
81. Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Hyperparameter search space pruning–a new component for sequential model-based hyperparameter optimization. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp 104–119. Springer (2015)
82. Xiao, Y., Wang, H., Zhang, L., Xu, W.: Two methods of selecting gaussian kernel parameters for one-class svm and their application to fault detection. Knowl. Based Syst. (2014)
83. Zha, D., Lai, K.H., Wan, M., Hu, X.: Meta-aad: Active anomaly detection with deep reinforcement learning. arXiv preprint arXiv:2009.07415 (2020)
84. Zhao, Y., Nasrullah, Z., Hryniewicki, M.K., Li, Z.: Lscp: Locally selective combination in parallel outlier ensembles. In: ICDM, SIAM (2019a)
85. Zhao, Y., Nasrullah, Z., Li, Z.: Pyod: A python toolbox for scalable outlier detection. JMLR (2019b)
86. Zhao, Y., Rossi, R.A., Akoglu, L.: Automating outlier detection via meta-learning. arXiv preprint arXiv:2009.10606 (2020)
87. Zhu, X., Ghahramani, Z., Lafferty, J.D.: Semi-supervised learning using gaussian fields and harmonic functions. In: ICML (2003)
88. Zimek, A., Campello, R.J., Sander, J.: Ensembles for unsupervised outlier detection: challenges and research questions a position paper. SIGKDD (2014)
89. Zimek, A., Schubert, E., Kriegel, H.P.: A survey on unsupervised outlier detection in high-dimensional numerical data. Stat. Anal. Data Mining ASA Data Sci. J. **5**(5), 363–387 (2012)
90. Zöller, M.A., Huber, M.F.: Benchmark and survey of automated machine learning frameworks. JAIR (2021)

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com