A Survey on AutoML Methods and Systems for Clustering

YANNIS POULAKIS, CHRISTOS DOULKERIDIS, and DIMOSTHENIS KYRIAZIS,
Department of Digital Systems, University of Piraeus, Piraeus, Greece

Automated Machine Learning (AutoML) aims to identify the best-performing machine learning algorithm along with its input parameters for a given dataset and a specific machine learning task. This is a challenging problem, as the process of finding the best model and tuning it for a particular problem at hand is both time-consuming for a data scientist and computationally expensive. In this survey, we focus on unsupervised learning, and we turn our attention on AutoML methods for clustering. We present a systematic review that includes many recent research works for automated clustering. Furthermore, we provide a taxonomy for the classification of existing works, and we perform a qualitative comparison. As a result, this survey provides a comprehensive overview of the field of AutoML for clustering. Moreover, we identify open challenges for future research in this field.

CCS Concepts: • **Computing methodologies → Cluster analysis;**

Additional Key Words and Phrases: Unsupervised learning, clustering, automated machine learning, meta-learning, algorithm selection, hyperparameter tuning

ACM Reference Format:
Yannis Poulakis, Christos Doulkeridis, and Dimosthenis Kyriazis. 2024. A Survey on AutoML Methods and Systems for Clustering. *ACM Trans. Knowl. Discov. Data.* 18, 5, Article 120 (February 2024), 30 pages. https://doi.org/10.1145/3643564

1 INTRODUCTION

Lately, the unprecedented rate of data generation on a daily basis has made an ever-increasing volume of datasets available. From IoT sensors that capture measurable variables of natural phenomena to services that collect information on digital processes, and from user-generated content in social media to open data initiatives, the produced data hold information that can be turned into actionable insights after applying suitable data-mining techniques. In this way, data-driven knowledge discovery methods are invaluable tools towards developing models from data that can be used to optimize various aspects of everyday life.

Machine learning algorithms are increasingly used to learn models from the wealth of available input data. These algorithms are trained upon collected data to perform certain tasks, such as classification, clustering, and regression. However, many alternative algorithms are available for a machine learning task and no single algorithm can be deemed as a universally optimal solution,

The research leading to the results presented in this article has received funding from the European Union's funded Projects MobiSpaces under grant agreement no 101070279 and Green.DAT.AI under grant agreement no 101070416.
Authors' addresses: Y. Poulakis, C. Doulkeridis, and D. Kyriazis, Department of Digital Systems, University of Piraeus, Karaoli & Dimitriou Str. 80, Piraeus, Greece, 18534; e-mails: {gpoul, cdoulk, dimos}@unipi.gr.

ACM 1556-4681/2024/02-ART120
https://doi.org/10.1145/3643564

as per the *"no free lunch theorem"* [1]. Additionally, each algorithm is accompanied with a set of parameters, whose values strongly affect the quality of the learned model, according to the dataset at hand.

This plethora of available options increases the complexity of designing machine learning pipelines, while it also requires expertise, is time-consuming, and is computationally expensive for the evaluation of different models. This realization is the motivation behind **automated machine learning (AutoML)**, a collection of methods that aim to alleviate these shortcomings by automating the machine learning modeling process. These methods include optimization techniques, such as Bayesian or genetic optimization, as well as meta-learning techniques, both described in the later sections of this article.

Nevertheless, AutoML methods are thoroughly studied for supervised learning, leaving its counterpart unsupervised learning less explored. This is mainly due to the lack of an optimization target to be used by the respective methods. While classification and regression models are trained on the true values of the target variable, in clustering the true clusters are typically not known beforehand. In turn, this hinders the evaluation of result quality by means of one objective measure. Instead, different internal **cluster validity indices (CVIs)** [2] are used, which may be obscure and oftentimes subjective.

However, designing clustering pipelines can greatly benefit from AutoML. The task of algorithm selection for clustering is challenging as many clustering algorithms exist and their performance depends on several dataset characteristics. Such characteristics are the geometrical properties of the datasets, the density distribution of their data points, data dimensionality, and other statistical properties of the data. The acquisition of this kind of information requires domain knowledge that can provide insights on which exactly of these characteristics and how they are related to algorithm performance. Therefore, access to algorithm selection methods proves valuable for clustering practitioners.

Furthermore, clustering algorithms require a set of input parameters to be specified prior to execution. While the majority of the popular clustering algorithms require the number of clusters to be provided as input, others, such as DBSCAN, require a radius and a number of points to be provided for defining data neighborhoods. Setting appropriate values for such input parameters is a cumbersome and time-consuming task, especially when multiple interrelated parameters need to be set. Additionally, in the specific case of clustering with neural networks through self-organizing maps [3, 4], unique hyperparameters are required such as the number of layers and their respective neurons. Therefore, in addition to algorithm selection, it is crucial to correctly set the values of necessary hyperparameters for each clustering algorithm.

Lately, considerable effort has been invested in developing methods for automated clustering. This has resulted in various prototypes and techniques, each with its own different merits. However, what is missing is to provide a common framework that includes all design choices in a comprehensive way. Moreover, it is necessary to identify the commonalities and differences between existing approaches to provide a deeper understanding of the research field of AutoML for clustering. Effectively, this is expected to help new researchers to acquire a better understanding of the field faster, but also experienced practitioners that want to test novel methods and techniques.

Motivated by this lack of clear and concise understanding of AutoML for unsupervised learning, in this work, we aim to provide a comprehensive review of applications of AutoML methods and systems for clustering. The scope of our survey is not intended to be a complete tutorial on AutoML methods and challenges, but rather a systematic review of AutoML applications in an unsupervised learning context. The contributions of this study can be listed as follows:

(1) Classification of existing methods for automated clustering based on a taxonomy.
(2) Qualitative comparison of existing systems based on a number of common characteristics such as the number of meta-features, the category of novel meta-features proposed in each work, and the optimization technique used.
(3) Identification and elucidation of open problems within the field of automated machine learning for clustering along with promising research directions.

As such, the remainder of this article is structured as follows: in Section 2, we present an overview of methods and techniques as used in the context of AutoML. Then, in Section 3, we propose a taxonomy for classifying the existing works in AutoML for clustering. In Section 4, we present a mapping of existing methods and prototypes for AutoML to the taxonomy together with detailed description of their operation. Section 5 presents a qualitative comparison of the related work, while Section 6 discusses open research problems in the field. We conclude the article in Section 7.

2 OVERVIEW

In this section, we present the preliminaries of AutoML for clustering. First, a brief overview of the clustering problem is provided and then we proceed to address the two main tasks of automated machine learning, namely, algorithm selection and hyperparameter tuning. Algorithm selection is typically addressed by meta-learning techniques, whereas hyperparameter tuning is dealt with optimization algorithms.

2.1 Clustering

The main goal of clustering is to partition a dataset into groups that contain similar instances but are dissimilar to instances in different groups. Dissimilarity is based on a distance measure, such as the Euclidean or cosine distance. While the choice of a distance measure usually depends on dataset characteristics, recent works propose automated methods for the recommendation of such measures for clustering algorithms [5]. Formally, a clustering schema is represented as a set of subsets $\{C_1, \ldots, C_k\}$ that contain instances of S such that: $S = \cup_{i=1}^{k} C_i$ and $C_i \cap C_j = \emptyset$ for $i \neq j$. This notation indicates that each instance belongs to one and only one subset of the partitioning schema. Although this is the most popular clustering approach, also referred to as *"crisp clustering"* in the literature, there exists another category of algorithms [6, 7] that assign a membership value to each instance associating it with more than one subset in the formed partitioning, also known as *"fuzzy"* or *"soft"* clustering. All of the works included in this study focus on the first category of clustering algorithms.

Several metrics have been defined to assess the quality of the resulting partitioning of a clustering algorithm. These metrics are classified into internal and external CVIs [2]. Internal CVIs measure intricate properties of a clustering schema, such as inter- and intra-cluster separability or the density of the produced clusters [2, 8].

External CVIs rely on the availability of ground truth, which consists of the cluster label for each instance in the dataset. As the availability of ground truth is not always given in unsupervised learning, internal CVIs are most frequently used in real-world applications. Nevertheless, determining which internal CVI to use to assess clustering quality still remains a challenge among practitioners.

2.2 Algorithm Selection

2.2.1 Problem Overview. Every category of machine learning tasks includes a set of algorithms that act as potential solutions. In clustering, it is commonly accepted that algorithms such as K-Means are more reliable on clustering datasets that form spherical geometries. Instead,

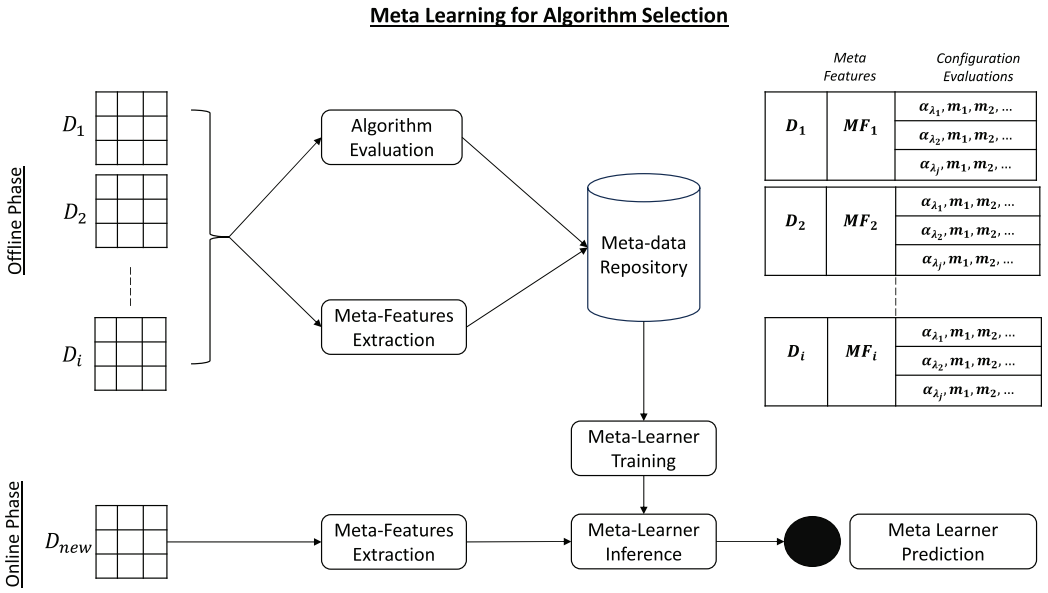

Fig. 1. The offline and online phases of meta-learning for algorithm selection. In the offline phase, a set of algorithm configurations is examined on a collection of datasets to produce meta-data and train the meta-learner. In the online phase, the meta-learner is applied to make the necessary predictions.

density-based algorithms, such as DBSCAN and OPTICS, are more effective on complex geometries of varying density. Likewise, several clustering algorithms exist, each with its own advantages. This realization is the rationale behind what is known as the Algorithm Selection problem [9], which is formally defined as follows: Given a set of learning algorithms A and a dataset D, the goal of Algorithm Selection is to select an algorithm $\alpha \in A$ that minimizes a loss metric L.

$$\alpha^* = argmin_{\alpha \in A} L(\alpha, D) \tag{1}$$

As Equation (1) presents the problem of algorithm selection under the scope of clustering, L could be any of the clustering evaluation metrics that exist. Which cluster validity index to use for identifying the best-performing algorithms remains one of the most challenging aspects of applying AutoML techniques for clustering. Notice that in the case of supervised learning, the equation is slightly modified to support cross-validation by averaging the loss calculated from different splits (D_{train} and D_{test}) of the dataset D.

2.2.2 Meta-learning for Algorithm Selection. Typically, in AutoML for clustering, algorithm selection is tackled with the use of meta-learning. Meta-learning [10], or *"learning to learn,"* is the process of extracting insights from the examination of various solved tasks to guide the design or selection of future models, also capable for problem-solving of the same or a similar task. Each task is associated with a set of measurable characteristics known as meta-features and a set of measured results from the evaluation of different solutions to the task. Meta-learners are predictive models that are trained on meta-features and evaluation measures, which are then used on newly encountered tasks to predict well-performing solutions.

Meta-learning for algorithm selection is divided into an offline and an online phase, as presented in Figure 1. The tasks examined in this case refer to *clustering a specific dataset* for a variety of datasets and the potential solutions are the clustering algorithms evaluated on selected metric(s).

Formally, in the offline phase, for a collection of datasets $\{D_i\}$, a set of meta-features MF is extracted and a set of algorithm configurations is evaluated according to a set of evaluation metrics M. Afterwards a meta-learner model is trained on the set of meta-features for each dataset. The output of the meta-learner is one of the following: (1) the best-performing algorithm α^*, (2) an algorithm ranking $[a^{1st}, a^{2nd}, \ldots, a^{last}]$, or (3) the value of an evaluation metric $m \in M$. Then, the meta-learner is applied during the online phase to make predictions for the dataset at hand.

2.3 Hyperparameter Tuning

2.3.1 Problem Overview. Although automated algorithm selection alone provides a significant advantage in modeling machine learning pipelines, most of the algorithms also require various parameters to be set that directly affect their performance. In clustering algorithms, these parameters vary from the number of output clusters to the selection of the distance metric to be used to measure instance similarity. In popular frameworks that include machine learning algorithm implementations, such as scikit-learn [11], default values exist for each parameter of an algorithm. However, in most cases, selecting appropriate values results in improved performance.

Formally, the problem of hyperparameter optimization is expressed as follows: Let a denote a machine learning algorithm. A vector of hyperparameters is denoted by $\lambda \in \Lambda$, and algorithm a with its hyperparameters instantiated to λ is denoted by a_λ. Given a dataset D our goal is to find:

$$\lambda^* = argmin_{\lambda \in \Lambda} L(a_\lambda, D). \tag{2}$$

In addition, both algorithm selection and hyperparameter tuning can be tackled simultaneously through optimization methods to solve the problem known in the literature as **Combined Algorithm Selection and Hyperparameter Tuning (C.A.S.H.)** [12]. While this approach is common in supervised learning, these processes are usually decoupled in automated clustering. In this case, the equation to be solved is formulated as follows:

$$a^*, \lambda^* = argmin_{a \in A, \lambda \in \Lambda} L(a_\lambda, D). \tag{3}$$

2.3.2 Optimization Methods for Hyperparameter Tuning. To address the problem of hyperparameter tuning, several optimization techniques have been proposed in the literature that test a set of parametric settings and select the one that minimizes the desired metric. Although exhaustive strategies, such as grid search, are applicable, the techniques presented in this section have been proven to be more efficient. Unlike meta-learning for algorithm selection, where mainly the offline phase is processing-intensive and the online phase is only the inference, hyperparameter tuning is a single-step process that can be very costly.

In the remainder of this subsection, we present three of the most prominent optimization algorithms used in hyperparameter tuning, namely, Bayesian, Genetic, and Bandit-based optimization. Note that each pseudo-code block that accompanies their respective discussions is presented through the scope of tuning the hyperparameters of a single algorithm. However, these algorithms are also able to tackle the C.A.S.H. problem by considering a parametric space that is a combination of the parametric spaces of the considered algorithms.

Bayesian Optimization. The most prominent state-of-the-art optimization technique is Bayesian optimization, which is also used widely in supervised AutoML. This optimization technique targets the global optimization of black-box functions that are expensive to evaluate. Such an expensive evaluation is that of machine learning algorithms trained with specific parametric values on a dataset and the black-box aspect refers to the lack of a functional form between parameters and the loss metric. For a thorough tutorial on Bayesian optimization, we refer to References [13, 14].

ALGORITHM 1: Bayesian Optimization Algorithm

Input: parametric space (Λ), Data set (D), iterations (n)
Output: λ^*, m^*

Select *initial configurations according to design* $\lambda_1,\ldots,\lambda_j$
Evaluate *selected configurations,* $m_1, \ldots, m_j \leftarrow L(\alpha_{\lambda_1}, D),\ldots,L(\alpha_{\lambda_j}, D)$
Set $m^* \leftarrow \min(m_1, \ldots, m_j)$, $\lambda^* \leftarrow argmin_{\lambda\in\Lambda} L(\alpha_\lambda, D)$
Fit *surrogate model S over* $m_1, \ldots, m_j$ *and* $\lambda_1, \ldots, \lambda_j$
while $i < n$ **do**
  Select *Next configuration for evaluation according to acquisition function* U, $\lambda_i = argmax_{\lambda\in\Lambda} U(\lambda_i)$
  Evaluate *new configuration,* $m_i \leftarrow L(\alpha_{\lambda_i})$
  if $m_i < m^*$ **then**
    $\lambda^* \leftarrow \lambda_i, m^* \leftarrow m_i$
  end
  Update *surrogate model, S*
  $i \leftarrow i + 1$
end

Algorithm 1 is an iterative algorithm that chooses points in a defined search space to be evaluated according to an exploration-exploitation mechanism. At each iteration a probabilistic model, also known as surrogate model, is fitted upon the observations made, i.e., the evaluation of parametric configurations. A popular choice for surrogate models is Gaussian processes [15, 16]. Other models have also been suggested in the literature, such as tree parzen esimators [17] and random forests [18].

Which point to evaluate at each iteration is decided with the use of an acquisition function. The most popular choice is the expected improvement [19, 20]. Improvement at point x is expressed as shown in Equation (4) and its expected value, when considering a Gaussian surrogate model, is depicted in Equation (5), where,

— σ is the standard deviation of the Gaussian process.
— μ is the mean of the Gaussian process.
— ϕ is the probability density function of the standard normal distribution.
— Φ is the cumulative distribution function of the standard normal distribution.
— f^* is the best evaluation so far.

$$u(x) = max(0, f^* - f(x)) \tag{4}$$

$$E[u(x)] = (f^* - \mu) \cdot \Phi\left(\frac{f^* - \mu}{\sigma}\right) + \sigma \cdot \phi\left(\frac{f^* - \mu}{\sigma}\right) \tag{5}$$

Genetic Optimization. Another genre of optimization algorithms used in the literature [21] for hyperparameter tuning is that of population-based algorithms, also known as evolutionary algorithms and more specifically genetic optimization. Evolutionary algorithms draw inspiration from Charles Darwin's theory of evolution and are based on the "*survival of the fittest*" paradigm. As such, the terminology used to describe the procedure originates from biology.

Genetic optimization algorithms start by evaluating a given set of points in the search space, also known as population, that are potential solutions to the problem at hand. In the hyperparameter tuning domain, this population includes different parametric configurations. The evaluation refers to measuring specific metrics, such as cluster validity indices, of machine learning tasks. After the initial population is evaluated, a selection strategy is applied to select a subset of configurations,

ALGORITHM 2: Genetic Optimization Algorithm

Input: parametric space (Λ), Data set (D), iterations (n), crossover probability (cp), mutation probability (mp)
Output: λ^*, m^*
Initialize *population* = $\lambda_1, \ldots, \lambda_j, \lambda \in \Lambda$
while $i < n$ **do**
  Evaluate *population*, $m_1, \ldots, m_j \leftarrow L(\alpha_{\lambda_1}, D)$,...,$L(\alpha_{\lambda_j}, D)$
  Select *configurations for crossover*, $T \in \Lambda$
  Apply *crossover*, *offsprings* $\leftarrow$ *crossover*(T, cp)
  Apply *mutation*, *offsprings*$' \leftarrow$ *mutation(offsprings, mp)*
  define new generation of population
  $i \leftarrow i + 1$
end
Set $m^* \leftarrow \min(m_1, \ldots, m_j)$, $\lambda^* \leftarrow argmin_{\lambda \in \Lambda} L(\alpha_\lambda, D)$

ALGORITHM 3: Successive Halving Algorithm

Input: parametric space (Λ), Data set (D), budget (b)
Output: λ^*, m^*
Set $no\_configurations \leftarrow length(\Lambda)$
while $no\_configurations > 1$ **do**
  Allocate budget uniformly to each configuration, $b' \leftarrow \frac{b}{no\_configurations}$
  Evaluate *each configuration*, $m_i \leftarrow L(\alpha_{\lambda_i}(b'), D)$
  Rank configurations according to m_i
  keep the top half of the ordered configurations, $\Lambda' \leftarrow \frac{\Lambda}{2}$
  $no\_configurations \leftarrow length(\Lambda')$
end

labeled as ancestors, to produce new configurations. This operation is known as crossover. Finally, the produced configurations may be mutated, i.e., altered randomly, to increase the probability of strengthening weaker configurations [22].

This procedure is iterative and goes on for a fixed number of iterations or until a predefined evaluation threshold has been reached. The basic structure of the genetic optimization algorithm is shown in Algorithm 2. Genetic algorithms are widely customizable, since various ancestor selection, crossover, and mutation techniques have been proposed in the literature. For a more thorough explanation of this algorithmic family, we refer the interested reader to References [23, 24].

Bandit-based Optimization. Hyperparameter tuning has also been related to the *multi-armed bandits* [25] problem. According to this problem, there is a need to allocate a fixed budget of resources to the evaluation of different solutions to maximize the expected gain. A key aspect of this approach is that budget allocation for each solution can be changed during the optimization process. The intuition behind identifying the hyperparameter optimization problem as the *multi-armed bandits* problem is that partially trained models can still provide an indication for the quality of their respective fully trained versions. To this end, in Reference [26], successive halving is proposed, an algorithm that uniformly allocates a fixed budget on a set of configurations and eliminates the half worst-performing ones. This procedure is repeated until a single configuration is deemed as the best and ensures that the budget is exponentially allocated to more promising configurations. In Reference [27], successive halving is extended to tackle the need for defining

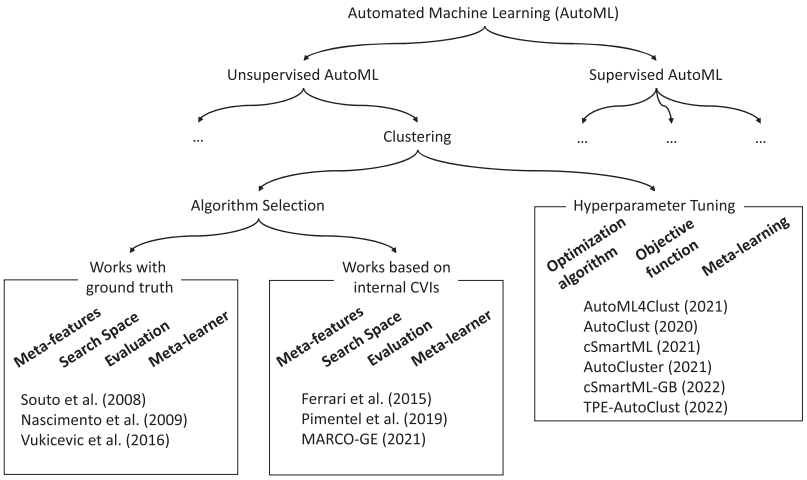

Fig. 2. The taxonomy of works followed in the present study.

how many configurations should be considered in an optimization cycle and how much budget should be allocated to each.

3 CLASSIFICATION TAXONOMY

In this section, we present a taxonomy for the classification of existing research in AutoML for clustering. As shown in Figure 2, we first separate the works according to their primary focus: algorithm selection or hyperparameter tuning. The works on algorithm selection are further classified in two categories, based on the availability and use of ground truth in the datasets. In the following, we describe in more detail the main design choices in each of the two main categories.

3.1 Algorithm Selection

We identify four different design choices for tackling algorithm selection and use them as the basis of comparison of relevant works later in this study.

Meta-features. The first crucial choice when designing a meta-learning framework is which meta-features to choose. Usually, meta-features come from domains such as statistics and information theory, they are extracted from landmarking techniques [28], or they are hand-crafted to capture specific domain knowledge, such as the technology used to gather data [29]. In clustering, it is essential that these meta-features capture the properties of a dataset related with the performance of clustering algorithms to make meaningful recommendations. This includes aspects such as the existence of geometric patterns, the distribution of density, and any formed neighborhoods of instances.

Search Space Exploration. The second differentiating factor is how in the offline phase information on algorithm performance for the available datasets is gathered. First, there is the configuration space design, i.e., which algorithms to consider and how their related parametric search spaces are defined. Then, there is the search strategy over this configuration space. As this procedure is part of the offline phase, the computational budget is usually not a major problem, and therefore search strategies, such as grid search, over large configuration spaces are conducted. Alternatives include random search, which has been showcased to outperform grid search in certain cases [30], or the optimization techniques presented in the previous section (Bayesian, Genetic, and Bandit-based optimization), which are preferred due to speed and better efficiency over real-valued parameters or unequally important parameters.

Evaluation. Perhaps the most challenging aspect in meta-learning for algorithm selection is how to identify the algorithm performance rankings or (in a similar context) which algorithm exhibits the best performance. This problem is due to the unsupervised nature of clustering and the lack of one commonly accepted cluster validity index. Strategies to overcome this problem include (a) selecting subsets of validity indices [31], (b) combining them to create a unified view of algorithm performance (usually through rankings) [32], or (c) using datasets with known "true" clusters, so evaluation measures such as the Adjusted Rand Index [33] can be used.

Meta-learner. Finally, the meta-learner is the model that is trained over the meta-features and configuration evaluations to be used to make the prediction of the best algorithm in the online phase.

In many cases, the k-nearest neighbors [34] is used as meta-learner, due to its simplicity and lack of training phase. However, other machine learning classifiers have also been used, such as random forests. When the aim is to predict the performance ranking of the available clustering algorithms, either multiple models can be used, each trained to predict the individual rank of each algorithm, or a classifier capable of predicting directly a ranked list, such as XGBoost [35].

3.2 Hyperparameter Tuning

For hyperparameter tuning, we identify three design choices that have impact on both computational efficiency and quality of results.

Optimization algorithm. As discussed in the previous section, there exist various optimizations methods that have been used for hyperparameter tuning. While some bring excellent results in terms of configuration selection, others are more efficient in terms of computational complexity. Furthermore, genetic algorithms can also be used to optimize multi-objective criteria, which can prove useful when considering multiple internal cluster validity indices. Therefore, we identify the optimization algorithm as the first differentiating factor among works for automated clustering.

Objective function. The next design choice is the loss function for the optimization algorithm, i.e., which evaluation metric to optimize when searching for the best configuration.

This is an inherent problem of clustering evaluation, which directly affects the optimization process and hinders the transition of these methods to the unsupervised domain. Evidently, one could arbitrarily select a single one of the available metrics, but it is not clear if that is capable of indicating the configuration that produces the best clustering.

Meta-learning. Finally, it should be noted that there are cases in the literature where meta-learning is used in combination with an optimization technique to warm-start the procedure [36]. This is likely to provide benefits, for example, by selecting a subset of the most promising algorithms. Another example would be narrowing down the search space of individual parameters. In both cases, the search space is restricted to the most promising configurations. In contrast, if this restriction of the search space fails to include the actual optimal configuration, then the optimization process will fail to identify it. Additional use cases of meta-learning prior to hyperparameter tuning also include the selection of a cluster validity index for optimization and budget selection.

4 MAPPING AUTOML SYSTEMS TO THE TAXONOMY

Recently, automated machine learning approaches for clustering have attracted the attention of the research community. As a result, several system prototypes have been proposed, presented in Table 1, so we describe in this section how they operate and we provide an overview of their distinguishing features.

We classify the relevant works in two main categories: First, in Section 4.1, we present the related works that perform solely algorithm selection, which we further classify according to their

Table 1. Characteristics of Studies Included in this Article

	Release	No. Data sets	Code Available
Souto et al. [29]	2008	32	✗
Nascimento et al. [37]	2009	35	✗
Ferrari et al. [38]	2015	84	✗
Vukicevic et al. [39]	2016	30	✗
Pimentel et al. [40]	2019	219	✗
AutoClust [41]	2020	24	✗
Marco-GE [42]	2021	210	✓
cSmartML [21]	2021	27	✓
AutoCluster [43]	2021	33	✗
AutoML4Clust [44]	2021	29	✗
TPE-AutoClust [45]	2022	29	✗
cSmartML-GB [46]	2022	29	✗

evaluation strategy to distinguish approaches that exploit labels (ground truth) in the datasets from other approaches that do not rely on such an assumption. Afterwards, in Section 4.2, we examine fully fledged systems that additionally consider hyperparameter optimization either as a joint problem with algorithm selection or independently.

The comparison of the works is based on the design choices defined in Section 3 (Figure 2) and presented in Tables 3 and 4, respectively. Finally, a complete listing of meta-features and internal cluster validity measures used in each work are presented in Tables 5 and 6.

4.1 Methods for Algorithm Selection

AutoML methods that focus on algorithm selection can be classified in two categories: (a) methods that rely on the availability of ground truth and exploit this information to perform the algorithm selection and (b) methods that do not require this information and (instead) operate using internal **cluster validity indices (CVIs)**.

4.1.1 Works that Exploit Ground Truth Labels. The works presented in this section propose meta-learning approaches for algorithm selection that rely on the availability of ground truth, i.e., *a priori* information about cluster membership of each instance. Having information about the true clusters enables the evaluation of clustering partitions produced by different algorithms using external validity indices, similar to the evaluation of supervised learning. However, this limits the applicability to cases where the cluster labels are available in the datasets, which is rarely the case in unsupervised settings.

Souto et al. [29] To the best of our knowledge, the first work that addressed algorithm selection for clustering via meta-learning is that of Souto et al. [29]. A set of 8 meta-features is proposed for dataset representation, 7 of which measure different descriptive characteristics of the data, while the last one is domain-specific, indicating the technology used to gather the data. The meta-features include: the logarithm of number of records, the logarithm of the ratio of number of records over attributes, the percentage of missing values, and the percentage of attributes kept after filtering. Also, another 3 meta-features are extracted from the pairwise generalized distances of instances [47]: (i) the proportion of these distances within 50% of a Chi-Squared distribution with degrees of freedom equal to the number of attributes, based on the assumption that they should follow this distribution if the dataset comes from a multivariate normal distribution; (ii) the skewness of this vector; and (iii) the percentage of these values that are more distant than two standard deviations from the mean, indicating potential outliers.

Additionally, for each dataset, 7 algorithms are evaluated with default parameters, except for the distance function and the number of clusters. The distance functions evaluated for each algorithm are Euclidean, Cosine, and Pearson's correlation. For the algorithms that require the number of output clusters as a parameter, this value is set to the number of clusters reported as ground truth. Clustering performance is measured by the **Corrected Rand index (cR)**, a metric that examines the agreement between predicted and true clusterings. Afterwards, each algorithm is ranked according to its performance on a per dataset basis. Finally, on top of this information, the meta-features and the algorithm performance rankings, 7 regressors are trained, each to predict an individual algorithm's performance rank. Their combination produces the final rank prediction of algorithms.

The term *rank prediction* of algorithms is used to refer to predicting the relative rank of the evaluated clustering algorithms, instead of predicting just the best-performing algorithm. Essentially, given a metric m that is used to evaluate the clustering quality for a dataset d, if $m(a, d) < m(a', d)$ for any two algorithms a, a' (i.e., a' is better than a based on $m()$), then a' is ranked higher than a.

The approach in Reference [29] is empirically evaluated using 32 microarray datasets for gene expression with known ground truth. The meta-learner is evaluated with the leave-one-out methodology, and the predictions are compared using **Spearman's Rank Correlation Coefficient (SRCC)** against the ideal rankings, obtained in the offline phase by ranking the algorithms according to their best evaluations for each dataset. The proposed algorithm ranking prediction strategy is also compared to always recommending the default ranking, which is the average of algorithm performance rankings for each dataset in the repository. Based on a statistical significance test of the mean SRCC of the two approaches, the conclusion is that the proposed approach provides a rank prediction that is more correlated to the ideal ranking.

Nascimento et al. [37] Later, Nascimento et al. [37] extended the work in Reference [29] by including a set of 5 additional meta-features, which—however—rely on the existence of ground truth in datasets. Practically, these meta-features measure the size of true clusters, the classification error obtained after applying a k-nearest neighbors classifier, and the similarity of the cluster instance distribution to the uniform distribution. The search space consists of seven different algorithms that are tested using various dissimilarity measures, namely: Pearson's Correlation Coefficient, Cosine, Spearman's Correlation Coefficient, and four versions of the Euclidean distance (original, standardized, scaled, and ranked). Algorithm performance is evaluated using the **Corrected Rand index (cR)**, but any combination of algorithm and dissimilarity measure that was unable to predict the actual number of clusters was disregarded from comparison. Finally, a set of six different classification models are compared as meta-learners on their performance for algorithms' rank prediction.

It is noteworthy that out of all the algorithms evaluated in Reference [37], only three algorithms, namely, **Finite Mixture of Gaussians (FMG), K-Means (KM)**, and **Spectral Clustering (SP)**, achieved best results across all datasets and they did in an imbalanced fashion. To avoid overfitting of the meta-learner, the meta-features are replicated to achieve a uniform distribution of classes. The experimental evaluation is based on 30 out of 35 microarray datasets, after excluding those for which a tie between clustering algorithms was observed. A set of six different classifiers from WEKA [48] are tested as meta-learners (using the "leave-one-out" methodology) based on their accuracy in predicting the best algorithm. The results show that Random Forests manage to outperform other meta-learners in terms of accuracy. Finally, the extraction of interpretable knowledge from the meta-learning analysis is examined. To this end, the MLRules algorithm [49] is applied to generate a set of 100 weighted rules in an attempt to associate the decision of algorithm selection with the most crucial meta-features. The 10 most important rules are presented,

Table 2. Reusable Components Described in the Work of Vukicevic et al. [39] to Form Hybrid Models that Define the Algorithm Search Space

Sub-process	*Available Reusable Components*
Representative Initialization	DIANA, RANDOM, XMEANS, GMEANS, PCA, KMEANS++, SPSS
Distance Measure	EUCLIDEAN, CITY, CORREL, COSINE
Representative Update Procedure	MEAN, MEDIAN, ONLINE
Cluster Evaluation	AIC, BIC, SILHOUETTE, COMPACT, XB-INDEX, CONNECTIVITY

with the first one suggesting that if the percentage of outliers is over a threshold, then FMG should be selected.

Vukicevic et al. [39] This study mainly differs from prior works in terms of the meta-learning task it aims to solve. The proposed meta-learner is trained to predict directly the performance of each algorithm on a specific dataset as measured by an external cluster validity index, instead of predicting the best algorithm(s) directly. As a result, the proposed meta-features additionally characterize the algorithm, instead of only the dataset. More specifically, two new sets of meta-features are introduced and used in combination with the ones in Reference [37].

The first set of meta-features describes the algorithms directly and independently of the datasets. They indicate how sub-processes included in the training of a clustering algorithm are solved, such as which distance measure is used to calculate instance similarity (i.e., the parameterizations of algorithms). However, these meta-features are effective only between algorithms that share some common sub-processes. Therefore, this study includes only representative clustering algorithms based on a so-called *component approach* [50, 51]. The reusable components are shown in Table 2, which are combined by a genetic algorithm [52] to form 504 combinations that are considered as individual models.

The second set of meta-features includes the measured values of several internal cluster validity indices. These meta-features vary per combination of dataset and algorithm, as different values would be produced for algorithms a_1, a_2 when applied on the same dataset D. In practice, this approach may often be prohibitively expensive, due to the computational cost of running the candidate algorithms prior to extracting meta-features. All of the models in the search space are evaluated offline on each dataset according to the **Adjusted Mutual Information (AMI)** between the predicted clustering and the ground truth. The meta-learner trained upon these observations in the offline phase is a regression model to predict the AMI for a new dataset.

A series of experiments are conducted on the same datasets as in Reference [37]. They measure the quality of the meta-learner by calculating the difference between its prediction and the actual performance score that was found in the offline phase. Recall that the meta-learner is trained to predict the algorithm performance in terms of Adjusted Mutual Information and is therefore treated as a regression problem. The difference is ultimately measured through **Round Mean Squared Error (RMSE)** and **Mean Absolute Error (MAE)**. Datasets are distinguished into three groups, depending on the microarray technology used to gather the data: one for each chip and one that takes into consideration both. Additionally, a set of experiments is conducted that targets meta-feature selection to examine potential improvements in meta-learning performance and identify the most relevant meta-features. This is achieved with the use of an evolutionary-based feature selection algorithm, although the exact methodology is not stated. The results present improvements for most meta-learning algorithms, with the best being SVM and Neural Networks across the three distinguished groups of datasets. For these two best-performing meta-learning algorithms, the percentage of meta-features selected after feature selection is examined for each of the two newly introduced sets of meta-features. Most of the meta-features based on the reusable components are selected, but fewer than 50% of the internal CVI meta-features are retained.

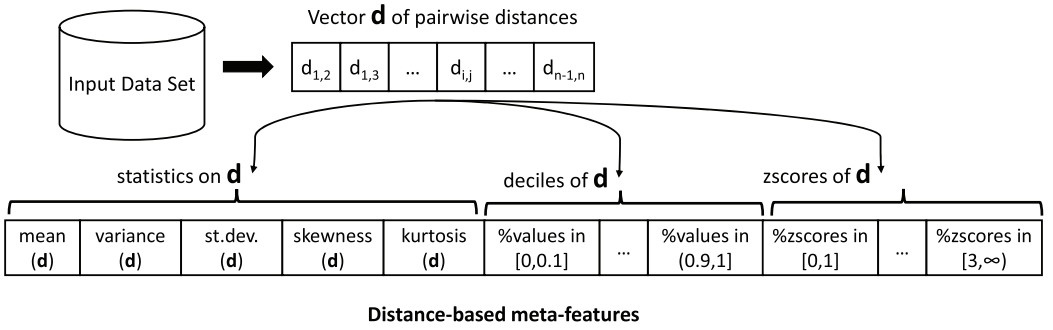

Fig. 3. Illustration of distance-based meta-features by Ferrari et al. [38].

4.1.2 Works Based on Internal Cluster Validity Indices. In the general case, clustering is used for situations where there is no knowledge about the existence of clusters or even the number of clusters in the dataset. Therefore, several methods for algorithm selection are oblivious to the existence of ground truth information. In this section, we review these methods, which are closer to the needs of real-life applications, where data are not labelled and its manual annotation is either too costly or even infeasible.

Ferrari et al. [38] One of the first approaches to tackle algorithm selection for clustering in an unsupervised manner is Reference [38]. The main two novelties introduced in this study are: (i) a new set of *distance-based* meta-features based on pairwise distances of instances and (ii) a new method for ranking algorithm performance.

The meta-features are practically statistical properties (deciles and z-scores) computed over the distance distribution of the dataset. The intuition is that the distance distribution in a dataset plays an important role in deciding the best clustering algorithm. In more detail, as shown in Figure 3, all pairwise Euclidean distances are computed and stored in a vector, then this vector's values are normalized in the range of $[0, 1]$, and finally the statistical properties are computed. Additionally, a set of attribute-based meta-features is used, based on the ones proposed for classification problems in References [53, 54].

For the association of the datasets with algorithm performance, a set of seven different algorithms were evaluated on 10 internal validity indices to jointly produce the algorithm performance ranking. The remaining problem is how to combine the algorithm rankings for different CVIs into a global algorithm ranking. To this end, two methods are proposed, called the *score ranking* and *winner ranking*. The former is based on a point-per-rank system that assigns points to each algorithm according to their position in the rankings. Instead, the latter compares for each pair of algorithms the number of "wins" over all CVIs, and the ranking is based on the number of wins. Finally, a KNN classifier is employed as meta-learner to perform rank prediction for a newly encountered dataset, based on the most similar ones in the meta-repository.

The approach is tested on 84 datasets, where the KNN meta-learner is used with the number of neighbors k varying between 2 and 10. The meta-learner is evaluated based on its predictive ability as measured by the Spearman's Rank Correlation on the predicted and actual best rankings. The meta-features examined in the study are tested both independently, as attribute-based and distance-based, as well as jointly to create a hybrid vector of meta-features. The experimentation revealed that the distance-based and the hybrid-based versions produced better algorithm rankings than the attribute-based version.

Pimentel et al. [40] In this approach, a new set of meta-features is introduced based on *instance correlation*. First, the dataset instances are transformed to instance rankings. Although not directly

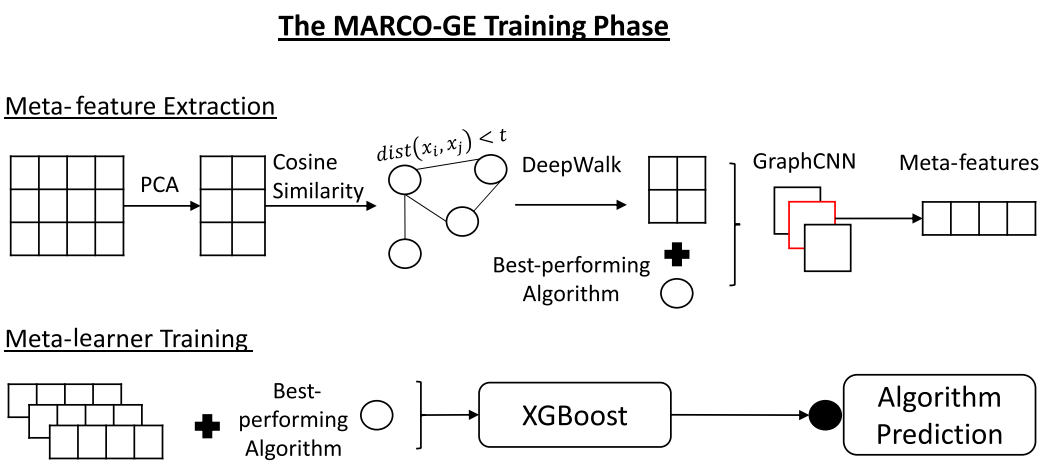

Fig. 4. The MARCO-GE meta-learning approach visualized.

explained, it is assumed that the instance ranking is the vector that consists of the rank of each individual feature value of a dataset instance. Afterwards, the pairwise correlation of the instance rankings is calculated by Spearman's Rank Correlation coefficient. The resulting vector is finally normalized and the meta-features, same as in Reference [38], are extracted from the pairwise correlation distribution. Not only are they the same meta-features extracted from a different vector, but throughout the study they are combined to create a hybrid version, named **correlation and distance (CaD)**. Additional to meta-features, each dataset is associated with the algorithms performance ranking. A set of 10 algorithms is evaluated by 10 internal cluster validity indices, each providing a different ranking, which are then averaged to form an aggregate version. Over these meta-data, the CaD meta-features and the algorithm rankings, a classification model is chosen as the meta-learner to be trained upon, responsible for recommendations.

The proposed methodology is evaluated on a collection of 219 datasets. Two meta-learners, namely, KNN and Random Forests classifiers, are tested for various parameter values of the number of neighbors and trees, respectively. Results measure their predictive performance on algorithm rankings, as measured by three metrics, SRCC, ARI, and AMI, against true rankings. Results are compared against different sets of meta-features from related works (statistical, distance, evaluation) and two baseline recommendation methodologies: the average and majority algorithm rankings of the meta-repository. Finally, the authors examine the meta-features importance by identifying the relative frequency of each, used in the ensemble of trees of the random forest meta-learner to create node splits.

MARCO-GE [42]. This is one of the most recent meta-learning frameworks for algorithm selection that suggests that the dataset characterization sub-problem can be approached by examining the graph representation of a dataset (Figure 4). The rationale behind the approach is that graphs hold information on both local and global neighborhoods of a dataset, thus directly affecting the performance of clustering. First, PCA is applied on the dataset and then a similarity graph is constructed, where vertices correspond to instances and edges connect two vertices if the similarity of the instances is above a certain threshold.

The graphs consist of (i) a node matrix, (ii) a degree matrix, (iii) an adjacency matrix, and (iv) a node feature representation matrix that is calculated with the DeepWalk algorithm [55], which provides the latent representation of graph nodes according to a predefined fixed size of dimensions. This graph representation of a dataset along with the respective best-performing algorithm act as

the training instances of a classification graph convolutional neural network that learns to model their relationship. The network consists of 3 convolutional layers with RELU activation function and a readout layer. The first 3 layers are responsible for learning high-level node representations, while the latter aggregates them to a single graph representation. In this graph representation, the output of the nodes in the readout layer act as the meta-features for a dataset's description.

To retrieve the best-performing algorithm in the offline phase, 17 algorithms are evaluated on each dataset after parameter optimization. The algorithms undergo the Bayesian optimization procedure with the use of Hyperopt [56]. The optimization target metric for the hyperparameter tuning is a linear combination of internal cluster validity indices. At each iteration, internal cluster validity measurements are calculated and stored. Afterwards, they are normalized as per the min-max normalization technique. Finally, the current optimization trials' indices values are averaged to provide the target optimization metric.

The meta-features extracted through the graph convolutional network and the algorithm performance rankings are used to train the meta-learner, XGBoost algorithm, that learns to predict algorithm rankings.

Experiments are conducted on 210 datasets, and the proposed methodology is tested both on predicting individual rankings produced by the examination of each available internal CVI and their averaged ranking. The new set of meta-features, drawn from the dataset's graph representation, are compared against some proposed from related works, namely, the distance-based [38], and the combination of distance and correlation-based meta-features (CaD) [40]. The meta-learner performance is tested using the leave-one-out procedure, measuring prediction performance through **Spearman's Rank Correlation Coefficient (SRCC)** and **Mean Reciprocal Rank (MRC)**, which measures the rank of the first correct recommendation against actual rankings. Overall, the MARCO-GE approach shows significant advantages over the two.

4.2 AutoML Systems for Clustering

Compared to algorithm selection, less work has been conducted targeting hyperparameter tuning for clustering models. To the best of our knowledge, six systems exist that provide this additional feature separately or combined with algorithm selection: AutoML4Clust [44], AutoClust [41], cSmartML [21], cSmartML-Glassbox [46], AutoCluster [43], and TPE-AutoClust [45]. In this section, we describe these systems, followed by their comparison in Table 4, with emphasis on the design choices identified previously in this survey: (i) the inclusion of meta-learning, (ii) the selection of optimization method, and finally (iii) the selection of the objective function.

AutoML4Clust [44]. Tschechlov et al. [44] examine how the typical optimization procedures used in AutoML for supervised learning can be applied into the unsupervised domain to solve the C.A.S.H. tuning problem. Four methods for hyperparameter optimization are evaluated, namely, random search [30], Bayesian optimization [12], Hyperband [27], and BOHB [57]. The first three have already been presented in Section 2.3.2. BOHB combines Bayesian optimization with Bandit-based methods to get the best of two worlds. These algorithms are in turn tested to optimize three internal, namely, cluster validity indices, **Calinski-Harabasz index (CH), Silhouette index (SIL)** and **Davies-Bouldin index (DB)**, which are assumed as the user input in the pipeline.

Experiments are designed to evaluate the approach on the following aspects: (i) the impact of budget allocation to the framework, (ii) the overall quality of clusterings recommended as measured by the **Adjusted Mutual Information (AMI)** against the true clusters known as ground truth, and (iii) the quality differences when using different internal CVI for optimization. Overall, the framework is shown to produce high-quality recommendations after no more than 60 loops of the optimizer. Also, their finding is that the Calinski-Harabasz index achieves the best results over

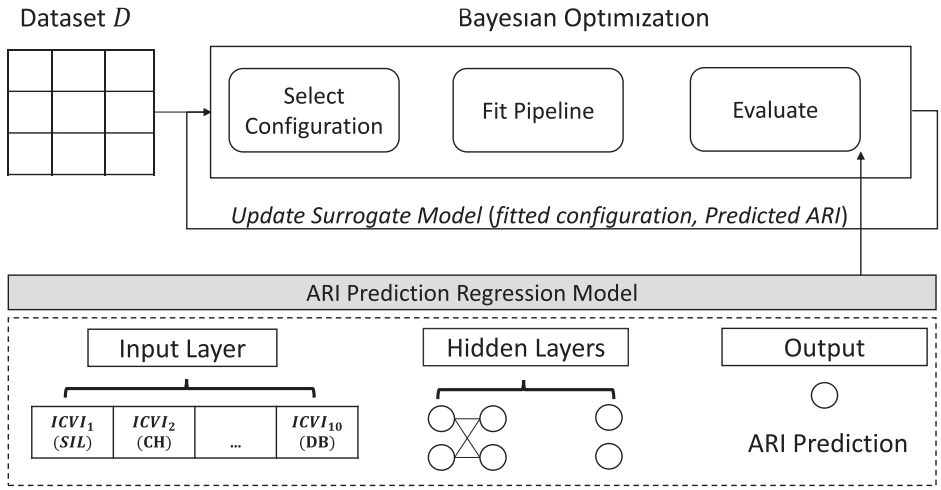

Fig. 5. The AutoClust [41] hyperparameter tuning phase.

every optimization method, while the Silhouette index is excluded from comparison completely, as it is stated to provide the most inaccurate results.

AutoClust [41]. Poulakis et al. [41] introduce AutoClust, a framework that targets algorithm selection and hyperparameter tuning in sequence. For the former, a meta-learning system is employed that solves the problem described in Section 2.2. The meta-features proposed in this study are measurements of the internal cluster validity indices as produced by applying Meanshift algorithm [58] on each dataset. According to the authors, Meanshift is selected due to its non-parametric nature. The datasets in this study come with the ground truth available, thus the Adjusted Rand Index is used for identifying the best-performing algorithm, and the configuration exploration strategy used is grid search over the parametric spaces of eight clustering algorithms.

Hyperparameter tuning is tackled by Bayesian optimization with tree-parzen estimators, as included in the Hyperopt library [56], which is modified to optimize a combination of internal CVI (Figure 5). This combination of indices is achieved by leveraging the meta-learning repository created in the offline phase, which includes both the internal validity indices and Adjusted Rand Index (external) for the evaluation of the different algorithm configurations. Both are used to train a regression neural network model to capture their relationship, which is then used in the evaluation process of the Bayesian optimization to combine the internal validity indices by predicting ARI. This ARI prediction acts as a proxy loss metric that the algorithm aims to optimize.

To evaluate the algorithm selection process, the introduced meta-features are compared against the ones found in Reference [38] by training a KNN meta-learner with both and measuring the respective accuracy and top-3 accuracy. For hyperparameter tuning, the methodology of combining internal cluster validity indices in the optimization process is tested against optimizing each individual available CVI with Hyperopt. Results on 24 real datasets validate the benefits of the approach.

cSmartML [21]. In this work, an AutoML framework for clustering is introduced that follows the paradigm of splitting algorithm selection and hyperparameter tuning into two phases. The main novelties in this work are (i) the selection of internal cluster validity indices to be used for hyperparameter tuning and (ii) the usage of genetic algorithms for hyperparameter tuning that utilizes more than one validity metric to optimize, following the multi-objective optimization paradigm.

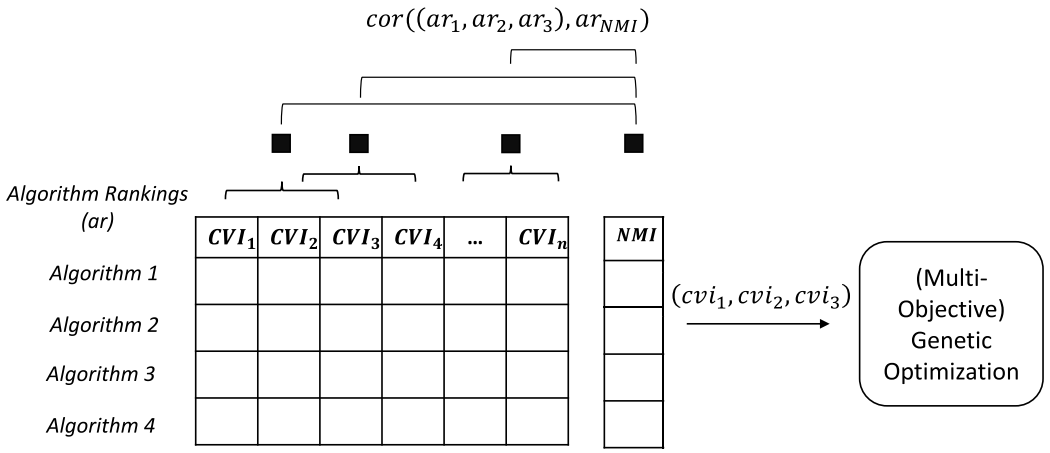

Fig. 6. The cSmartML [21] CVI selection phase.

During the offline phase of meta-learning, for a collection datasets that come with ground truth available, the meta-features described in Reference [38] are extracted. Additionally, a set of algorithm configurations is evaluated on 12 internal CVIs (Table 6) and 1 external, namely, **Normalized Mutual Information (NMI)**. For each combination of 3 different of the available internal CVIs, three individual algorithm rankings are produced, which are then merged to a single one, as depicted in Figure 6. These new merged rankings are afterwards compared in terms of correlation, as measured with the SRCC, with the algorithm ranking that is produced by evaluating configurations according to NMI. Therefore, the meta-model is trained to suggest the highest-ranked clustering algorithm according to the set of 3 CVI, which is best correlated to NMI and that set is also recommended to be optimized during the hyperparameter tuning stage.

The output of the meta-learning process is provided as input to a genetic algorithm for hyperparameter optimization. More specifically, the MuPlusLambda evolutionary algorithm [59] is employed, developed in the Python package DEAP [60], which follows the genetic algorithm structure presented in Section 2.3.2. The most notable difference between the genetic algorithm structure discussed is the selection of the best configurations at each generation according to multiple internal CVIs observations, as in this work the problem is treated as a multi-objective optimization. To select the best configuration, the NSGA-II scheme [61] is utilized, which performs a nondominated-sorting and crowding distance comparison.

One interesting finding of the experimentation process is that for 15 out of the 27 synthetic and real datasets in total, the multi-objective ranking is more correlated to the objective one that comes from the external evaluation of the clusterings than the rankings produced by individual internal CVIs. Additionally, the final framework performance is tested against several baseline methods. Each method is evaluated by measuring the NMI of the clustering achieved by the recommended algorithm configuration to the one that is known as ground truth clustering. The baselines include clustering by all available algorithms with default parameter values, tuning their parameters with grid and random search and the approach of AutoML4Clust [44]. The configurations selected by these baselines are indicated by a random internal CVI, which in most cases performs worse compared to the proposed methodology.

cSmartML-Glassbox [46]. Elshawi et al. have also proposed a variant called cSmartML-Glassbox [46], where the focus falls upon improvement of the user experience and the AutoML pipeline efficiency as depicted in Figure 7. The framework consists of several components: The

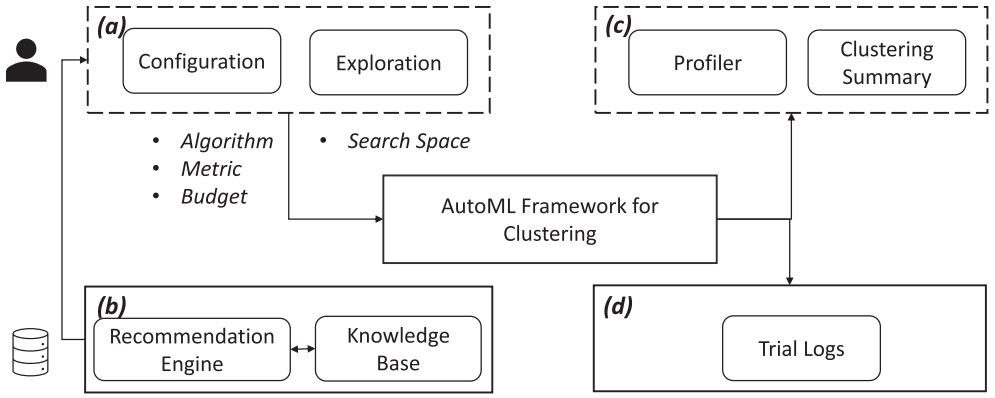

Fig. 7. The cSmartML-Glassbox [46] framework.

main interaction point between user and the framework is (a) a panel for configuration, for the user to select the algorithm(s), metric, and budget. Additionally, the user at this step can adjust the search space in the hope of achieving a more time-effective search process. Afterwards comes a meta-learning model that is the basis for a (b) time-budget recommendation engine. Essentially, this is the typical meta-learning procedure but with the target variable being the time budget necessary for the AutoML methods to find adequate results. The meta-features used include those found in Reference [38] and a set of landmarker meta-features, 12 internal cluster validity indices as calculated after from the results of 3 different clustering algorithms fitted on the data. This results in total 36 different meta-features. The target variable, i.e., the time budget that was enough to find the maximum performance as measured through Adjusted Rand index, is retrieved from running the framework on 200 synthetic and real datasets on 8 different time budgets.

Additional features of the framework include (c) visualization tools: a clustering visualization and summary panel that displays color-coded instances to represent the found clusters, the values of validity indices calculated for the resulting partitioning, a panel that displays the independent parameter to performance relationship, and how performance varies in a hyperpartition view. The latter refers to parameters that have only a handful of categorical values such as the affinity of spectral clustering that can be RBF or k-nearest neighbors. Finally comes the (d) exploration logger where AutoML trials can be saved, ultimately to help cases where the AutoML process is resumed in the future or rerun to accommodate for the non-deterministic nature of the AutoML algorithms.

The framework is evaluated in a usability study with the help of test-users. The first set of results is acquired from questionnaires answered from 23 machine learning practitioners with a basic machine learning and data science knowledge level that aim to capture the framework's ease of use, user understanding of the profiler, and willingness to use. The second part of the experimentation targets four machine learning experts and how they can leverage their expertise to enhance the AutoML pipeline by fine-tuning the search space through the framework. Finally, the recommendation engine is tested with different regression models as meta-learners, with the results showing the neural network regressors as the most successful.

AutoCluster [43]. Liu et al. [43] present a framework that incorporates meta-learning for algorithm selection and cluster ensembles to produce a final clustering for a dataset. For algorithm selection a new set of five different meta-features is proposed, four of which are based on landmarking. Landmark meta-features refer to certain measurable characteristics observed from fitted models, such as the number of leaves that are found in the resulting hierarchical tree from the training of the agglomerative clustering algorithm. The other one refers to the spatial

randomness of the data as measured by the Hopkin's statistical test [62], which is made against the null hypothesis that data are generated by a Poisson distribution. Furthermore, a set of additional 19 meta-features, addressed in Reference [63], are extracted to produce the final meta-feature vector. This vector is calculated for each dataset and paired with the respective best-performing algorithm, indicated in a joint fashion by Adjusted Rand Index and 3 internal CVIs in the offline phase. When a new dataset is encountered, a selection of the most similar datasets provides the final suggestion by majority voting.

After an algorithm has been recommended, it is optimized through grid search and a selection of the best configurations according to three internal validity indices (Silhouette, Calinski-Harabasz, Davies-Bouldin) are identified so the respective clusterings can be retrieved. This collection of clusterings is treated as an ensemble and realized through openEnsemble [64].

The experimentation process is based on comparing the resulting clustering partition of the proposed methodology (i) to the ones produced by several algorithms with default values and (ii) with those that are the product of an optimization procedure that also considers algorithm selection as a hyperparameter solving essentially the C.A.S.H. problem (Section 2.3.2). A number of 15 and 21 datasets out of 33 in total achieved improved performance via AutoCluster. Complementary experiments test the added benefits of the cluster ensembles and the need to contain diverse configurations and the meta-feature importance by applying the F statistical test.

TPE-AutoClust [45]. One of the most recent frameworks is TPE-AutoClust [45], in which automated clustering is achieved through a combination of meta-learning and genetic optimization. Meta-learning in this context is responsible for selecting the initial population of the genetic algorithm, pipelines that include both a preprocessor and a clustering algorithm. The meta-features that describe the datasets include the attribute and distance-based sets found in Reference [38] and an additional set of eight landmark meta-features that stem from the evaluation of clustering algorithms with internal CVIs. Specifically, three algorithms, namely, MeanShift, DBSCAN, and OPTICS, are evaluated with the scikit-learn's default parameters on the Calinski Harabasz and Davies-Bouldin indices and the Silhouette coefficient. The values of these internal metrics are finally used as meta-features. Meta features are combined with the best-performing pipeline for each dataset. To find that pipeline in the offline phase, a genetic optimization algorithm is employed optimizing Adjusted Rand index. When a new dataset is encountered, it is associated with the top-performing pipelines of the 50 most similar datasets to be used in the consequent step.

After the initial population has been selected through meta-learning, the genetic algorithm takes place. The pipelines are evaluated at each generation on the three internal indices also used in the previous step, and the top-rated ones are selected according to the NSGA-II scheme [61] for the respective genetic operations, crossover and mutation. This procedure is repeated until the time budget is exhausted and the top configurations are finally selected for ensemble construction. The ensemble construction takes place with the top-5 performing pipelines identified according to each of the available internal metrics by the majority voting scheme.

The framework is evaluated as a whole on 12 real datasets from UCI repository and 15 synthetic datasets against exhaustive searching, the results of algorithms with default values and 2 frameworks, namely, cSmartML and AutoML4Clust instantiated on time budgets of 10 and 30 minutes for each dataset. Performance of each approach is compared on the ARI measured of the proposed solution of each to the actual best clustering found in the offline phase. Additionally, results are tested for significance with the Wilcoxon statistical test. An additional set of experiments is conducted to evaluate the performance of meta-learning, where different versions of the framework are compared according to the meta-features used. Finally, an experiment is conducted to test the ensembling operation effectiveness, where the results provided by the genetic algorithm for each internal CVI are compared against the solution of the ensemble.

Table 3. Comparison of Meta-learning Systems

	No. Algorithms	No. Meta-features	Novel Meta-features Category	Algorithm Evaluation	Meta-learner
Souto et al. [29]	7	8	Descriptive	(External) CRI	7 SVM Regressors
Nascimento et al. [37]	7	13	Descriptive	(External) CRI	Random Forrest
Ferrari et al. [38]	7	28	Distance-based	(Internal) 10 CVI	KNN
Vukicevic et al. [39]	504*	24	Internal CVI, Reusable Components	(External) AMI	SVM Regressor
Pimentel et al. [40]	10	38	Correlation-based	(Internal) 10 CVI	KNN, Random Forest
MARCO-GE [42]	17	*	Graph latent Representation	(Internal) 10 CVI	XGBoost
AutoClust [41]	8	10	Landmark	(External) ARI	KNN
cSmartML [21]	8	19	Landmark	(Internal) 12 CVI & (External) AMI	KNN
cSmartML-Glassbox [46]	8	55	-	(Internal) 12 CVI & (External) ARI	MLP Regressor
AutoCluster [43]	6	24	Landmark, Data Distribution	(Internal) 3 CVI & (External) ARI	KNN
TPE-AutoClust [45]	6	28	Landmark	(Internal) 3 CVI & (External) ARI	KNN**

Exact meta-features and internal validity indices are reported in Tables 5 and 6, respectively. * The work of Vukicevic et al. [39] is associated with a very high number of algorithms, which in reality are different combinations of solutions to sub-processes of similarly structured algorithms, namely, representative based. ** Multiple nearest neighbors were selected instead of concluding to one solution through majority voting.

5 COMPARISON

In Table 3, we first examine the different meta-learning approaches. While all the works that focus on algorithm selection tackle the problem via meta-learning, five out of six clustering AutoML systems (Section 4.2) also incorporate meta-learning prior to hyperparameter tuning. In these cases the meta-learning task to be solved is not necessarily algorithm selection. This is feasible because algorithm selection can be tackled as part of the optimization procedure. Instead, we perceive additional meta-learning tasks, such as (a) the selection of a subset of internal cluster validity indices to optimize, (b) the prediction of the necessary time budget for hyperparameter tuning to find adequate results, or even (c) the selection of the initial population for the genetic optimization algorithms. While the task may vary between the works focused solely on algorithm selection and AutoML systems, we identify that the meta-learning aspect of each work can be effectively compared according to the specifications identified in Section 3, leading to the comparison of 11 works in Table 3 (out of the 13 total works).

One thing to consider is the number of available clustering algorithms per work ("No. Algorithms," Table 3). As the number of algorithms increases, it is expected that the number of classes for the meta-learner to predict increases. This in turn makes the classification task of the best algorithm more challenging. Several works opt to include variants of traditional clustering algorithms as independent models. This is mostly true in the works that focus solely on algorithm selection. Instead, in approaches where hyperparameter tuning takes place, these variants are expressed through parameters to be selected (linkage of the Agglomerative Clustering, etc.). The work of Vukicevic et al. [39] clearly stands out in terms of the number of algorithms considered. In reality, this number refers to the product of different combinations of sub-processes (see Section 4.1.1) that are used to train representative clustering algorithms (KMeans, KMedoids, etc.), and they are not used as the target variable of the meta-learning model, rather as training instances to predict their performance.

The number of meta-features used to build the meta-learning model also varies per work ("No. Meta-features," Table 3). Typically, most works introduce new meta-features and in some cases also include experiments or methods for the selection of them.

The choice of meta-learner varies, depending on the meta-learning task (regressors for predicting algorithm evaluation metric, classification models for predicting the best-performing algorithm, etc.). The most popular classification model is KNN, probably due to its simplicity and efficiency. Moreover, it is an algorithm that is still under active research with several advancements [65–68] including research for the distance to be used and the selection of the number of neighbors k. Additional work has also been made to reduce time complexity of the KNN algorithm,

Table 4. A Comparison of the AutoML Systems Developed for Clustering

Framework	Meta-learning	Hyperparameter Tuning	Objective Function
AutoClust [41]	✓	Bayesian Optimization	A regression model that predicts ARI from 10 ICVI
cSmartML [21]	✓	Genetic Optimization (multi-objective)	Selection of 3 ICVI in the meta-learning phase
cSmartML-GB [46]	✓	Bayesian Optimization	User Input ICVI
AutoCluster [43]	✓	Grid Search and Ensembling	Three different ICVI
AutoML4Clust [44]	✗	Bayes, Random, Hyperband, BOHB	User input ICVI
TPE-AutoClust [45]	✓	Genetic Optimization (multi-objective)	Three different ICVI

for example, through hyper-representations, as shown in Reference [69]. Regressors, however, are selected on a per-case basis.

Additionally it can be observed that 8 out of 11 meta-learning approaches are developed based on the existence of ground truth. Some of them, usually the earlier works on gene expression data, use only external metrics for the evaluation process, while others try to gain the best of both worlds. For example, cSmartML [21] compares the algorithm performance indicated by internal cluster validity indices to that obtained by Adjusted Mutual Information to select the most correlated indices. In AutoCluster [43], algorithms are annotated according to both Adjusted Rand Index and internal CVIs, and the meta-learner makes a recommendation by taking both into consideration. Clearly, this is a main differentiating factor among existing works, and it is still an open research problem, despite considerable efforts to move towards completely unsupervised solutions.

Moreover, most of the corresponding works utilize a subset of existing internal CVIs or a combination of them. In the majority of works, at least 10 CVIs are selected, but no subset of CVIs clearly stands out, as shown in Table 6. In fact, apart from the three most prominent indices, namely, Silhouette, Calinski-Harabasz, and Davies-Bouldin, the others tend to vary in terms of usage frequency. This is a clear indication that there is a lack of a fundamental methodology for the selection process, which largely depends on personal preferences and index popularity.

Further, in Table 4, we provide a comparison of the methods for hyperparameter tuning. Two notable observations can be made: First, all efforts, other than AutoML4Clust [44], employ meta-learning prior to hyperparameter tuning to select promising algorithms. In the case of cSmartML [21], **cluster validity indices (CVIs)** are additionally used for optimization. Second, similar to the meta-learning approaches, it is a common practice to combine multiple internal cluster validity indices to evaluate algorithm configurations. AutoClust [41] achieves this combination by training a regression model over internal CVIs, cSmartML uses multi-objective genetic optimization, and AutoCluster [43] defines a cluster ensemble based on the optimization of three different internal CVIs. The choice of optimization method is intertwined with how each work utilizes more than one index, such as in cSmartML, where the multi-objective genetic optimization is optimizing the three selected indices in the meta-learning phase.

Finally, in Table 5, we provide an overview of all the meta-features encountered in an attempt to create a potential point of reference for future research in the field. Each of the meta-features that we have identified can be mapped into one of the following five distinct categories:

— Several descriptive meta-features exist that take into account statistical measurements, such as the number of instances and features and the number of missing values.
— Some meta-features require external ground truth and thus can only be used when this information is available, like the ones that consider the size of ground truth clusters.
— Distance-based meta-features also exist, which are computed over the pairwise Euclidean distances.
— Correlation-based meta-features are computed over the vector of the Spearman's rank correlation coefficient computed over all pairwise instance rankings of a dataset. Instance rankings consist of the rank of each instance value according to the respective dataset feature.

Table 5. Meta-features Encountered throughout the Studies Considered in This Survey

Meta-features	Souto et al.	Nascimento et al.	Vukicevic et al.	Ferrari et al.	Pimentel et al.	Marco-GE	AutoClust	AutoCluster	CsmartML	TPE-AutoClust	cSmartML-GB
(Log) No. instances	✓	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗
(Log) No. features	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
% of discrete attributes	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
(Log) ratio of instances to the number of features	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗
(Log) ratio of features to the number of instances	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
% of missing values	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
Hopkins statistic	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Proportion of T^2 that are within 50% of the Chi-squared distribution	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
Skewness of the T^2 vector	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
% of attributes kept after selection filter	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
% of outliers ∗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
(Min, Max, Mean, std) class skewness	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
(Min, Max, Mean, std) class kurtosis	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
(Min, Max, Mean, std) discrete feature entropy	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
Mean feature correlation	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
Mean discrete feature concentration	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
PCA 95% deviations	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Skewness & kurtosis of the 1st pc of PCA	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
No. clusters with size inferior to k	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
No. clusters with size superior to k	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
Normalized Relative Entropy of cluster distribution	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
Classification error obtained by the KNN algorithm ($k\tilde{3}$)	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
ICVI measured by MeanShift	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
ICVI as measured by each algorithm configuration	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
ICVI measured by MeanShift, DBSCAN, OPTICS	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
Instance distance to closest cluster center (KMeans)	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
No. leaves (Agglomerative Clustering)	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Reachability instances (OPTICS)	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Distances to become core points (OPTICS)	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Mean, Variance, Std. Deviation, Skewness, Kurtosis of the distance vector **d**.	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓
% of values in 10 formed bins of equal size in the range of the normalized distance vector **d**	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓
% of values with absolute zscore of the normalized distance vector in 4 formed bins	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓
Mean, Variance, Std. Deviation, Skewness, Kurtosis of the correlation vector **c**	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
% of values in 10 formed bins of equal size in the range of the normalized correlation vector **c**	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
% of values with absolute zscore of the normalized correlation vector **c** in 4 formed bins	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Microarray technology used for gathering data	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
Reusable components, 1 meta-feature for each of the table x	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
300 meta-features from the previous to last graph CNN layer	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗

If a parenthesis precedes the meta-feature, then it indicates that a variant may have been considered instead.
Additionally, for certain meta-features such as the percentage of outliers, calculation may differ among works.

— Last, there is a group of application-specific meta-features that can only be utilized in a specific context. Examples include the indicator of technology used to gather the microarray gene expression data [29, 37] or the meta-features that stem from the representation of dataset through graphs [42].

6 OPEN CHALLENGES

While the topic of AutoML for clustering is actively researched, several challenges still remain open. Some challenges are associated with improvements of solutions to the sub-problems identified in this survey, such as the selection of meaningful meta-features. Other challenges relate to new

Table 6. Internal Cluster Validation Indices Used across the Clustering AutoML Methodologies

	Ferrari et al	Pimentel et al.	MARCO	AutoClust	AutoML4Clust	AutoCluster	cSmartML*	TPE-AutoClust
Bezdek-Pal [71]	✓	✗	✓	✗	✗	✗	✗	✗
Banfield-Raferty [72]	✗	✗	✗	✗	✗	✗	✓	✗
Hubbert-Levin (C Index) [73]	✓	✗	✓	✓	✗	✗	✗	✗
Calinski-Harabasz [74]	✓	✓	✓	✓	✓	✓	✗	✓
CDBW [75]	✗	✗	✗	✓	✗	✗	✗	✗
Davies Bouldin [76]	✓	✓	✓	✓	✓	✓	✓	✓
Dunn Index [77]	✓	✓	✓	✓	✗	✗	✓	✗
Friedman-Rasky [78]	✓	✗	✗	✗	✗	✗	✗	✗
Handl-Knowles-Kell [79]	✓	✗	✓	✗	✗	✗	✗	✗
Hubbert-T (Modified) [80]	✗	✗	✗	✗	✗	✗	✓	✗
I-Index [81]	✗	✗	✗	✗	✗	✗	✓	✗
SD-Scat [2]	✗	✓	✓	✗	✗	✗	✗	✗
SD-Dis [2]	✗	✓	✗	✗	✗	✗	✗	✗
SDbw [2]	✗	✗	✗	✓	✗	✗	✓	✗
Silhouette [82]	✓	✓	✓	✓	✓	✓	✗	✓
Gamma [83]	✓	✓	✗	✗	✗	✗	✗	✗
McClain Rao [84]	✗	✗	✗	✓	✗	✗	✓	✗
Milligan-Cooper [85]	✓	✗	✓	✗	✗	✗	✗	✗
PBM Index [86]	✗	✗	✗	✗	✗	✗	✓	✗
Ratkowsky Lance [87]	✗	✗	✗	✓	✗	✗	✓	✗
Ray-Turi [88]	✗	✓	✗	✗	✗	✗	✓	✗
Scott Symons [89]	✗	✗	✗	✗	✗	✗	✓	✗
Tau [90]	✗	✓	✗	✓	✗	✗	✗	✗
Xie-Beni [91]	✗	✓	✓	✗	✗	✗	✗	✗

*Both variations of cSmartML include the same set of meta-features.

topics, such as efficient handling of big volumes of data and clustering specifics like subspace clustering.

6.1 CVI Selection

One of the biggest challenges when designing an AutoML pipeline for clustering is the design of an evaluation strategy. The problem is that in an unsupervised setting, several **cluster validity indices (CVIs)** exist in the literature, and no single CVI is universally accepted as a suitable evaluation metric. As a result, the selection of an appropriate combination of CVIs is quite challenging and substantially impacts the resulting quality of AutoML. In this survey, we observed that this decision affects two discrete steps in the employment of meta-learning and optimization methods, respectively: (i) the evaluation of clustering algorithms in the offline phase of meta-learning to identify algorithm performance and (ii) the objective function to guide the optimization process.

Table 6 maps the different CVIs to the research papers that adopt one or more of them. Although some prove to be popular choices, such as the Silhouette and Dunn indices, it is clear that there is a lack of consensus on the exact subset of indices to select. While some studies provide insights on the suitability of certain CVIs [70], it is still unclear which ones to use to identify the best-performing algorithm on real datasets. Moreover, even though using more than one CVI seems a reasonable approach, it is not straightforward how to aggregate different CVIs into one evaluation measure. Existing approaches presented in this survey have tackled this problem by averaging each algorithm's performance over different CVIs or by trying to learn a relation among the CVIs. Identifying which CVIs are suitable for the AutoML task is a grand challenge for future works in this field.

We believe that a deeper understanding of the interrelationship between CVIs would be useful, instead of simply using all available CVIs. While some CVIs can be considered as general-purpose and can be used for many clustering algorithms, other CVIs make implicit assumptions on the

generated clusters. It is therefore necessary to systematically categorize CVIs to identify complementary CVIs and eventually focus on a carefully selected subset of CVIs that are appropriate for AutoML. Furthermore, approaches that try to learn the best way to aggregate these CVIs, going beyond plain averaging, seem to be promising.

6.2 Clustering-specific Meta-features

The selection of meta-features to be used in AutoML systems is another challenging task. As discussed throughout Section 3 and summarized in Table 5, different sets of meta-features are explored by different systems, aiming to acquire suitable representations for datasets for predicting the performance of clustering algorithms. For clustering, it is a common perception that meta-features should in some way include information about the distances of the instances, which is at the core of many clustering algorithms. However, other meta-features derived from categories such as landmarking have also been proven to provide decent results for algorithm selection. Furthermore, some recent works, notably References [92] and [93], have proposed task-agnostic meta-feature extraction with promising results. How to combine existing sets of meta-features, identify correlations and redundant meta-features for clustering, or even decide against the use of some meta-features are open questions.

Moreover, it is our belief that new approaches to meta-feature extraction are necessary to better capture similarities between datasets, where the concept of similarity is based on demonstrating similar performance for the same clustering algorithm. One could argue in favor of applying recent successful results from AI and Deep Learning for the task of meta-feature extraction. The main challenge to overcome when considering such approaches is how to train a model to extract meta-features from datasets of varying shapes. We believe this topic to be of great research interest, which can potentially enable automated meta-feature extraction, where models can learn meaningful per-task meta-features.

Another question relates to understanding the intrinsic properties that different datasets share, when all provide good results for a specific clustering algorithm. Evidently, the study of meta-feature extraction will concern many researchers in AutoML in the near future.

6.3 AutoML for Big Data

A challenge that is common in AutoML for both supervised and unsupervised learning is how to create scalable solutions to support big data. The focus of most works presented in this study is on quality of results, in terms of suggesting the best model configuration, rather than minimizing the execution time of the respective methodologies. However, very recently, improving the execution time of AutoML based on sampling techniques has been proposed [94]. Optimizing AutoML methodologies in terms of execution time and hardware requirements will not only make them cost-efficient, saving cloud-based expenses, but will also make them approachable to academic researchers and students alike.

To that end, we identify some sub-processes to be considerably more costly than others. For example meta-feature extraction is typically more expensive than meta-learner inference. Therefore, it is important when selecting a subset of meta-features to include to also consider their extraction time. This holds especially true for meta-features that fall under the landmarking category where one would first fit one or more machine learning algorithms to extract some set of meta-features.

Hyperparameter tuning is another expensive process due to the necessary amount of algorithm evaluations. As all of the optimization algorithms, presented throughout this study, require a budget to be predefined, developing a systematic method for identifying a reasonable budget could potentially lead to better performance. Moreover, development of early stopping criteria for the optimization algorithms can help avoid unnecessary evaluations.

We believe that choosing a subset D' of the original dataset D to perform algorithm selection and hyperparameter tuning is a promising approach. While Reference [94] is a solid step in that direction, it still remains to be tested whether it can be applied in the unsupervised setting. We foresee that certain challenges may arise, such as how to ensure cluster geometry is persistent through the sample. However, validating this argument and providing structured methodologies to solve it, if it exists, will have a valuable impact on the computational cost of AutoML systems.

In conclusion, we believe that it is of great essence to study more closely the execution time of future methodologies for AutoML and to apply them to much larger datasets. This will resemble much more real-world problems and will make AutoML approaches more practical.

6.4 Multi-view and Subspace Clustering

For multidimensional datasets it may be infeasible to achieve a reasonable clustering over the entire set of attributes, yet meaningful clusters may exist in subspaces. Thus, several subspace clustering algorithms have been proposed in the literature, which are further divided into top-down and bottom-up approaches [95]. This aspect of clustering is still unexplored, to our knowledge, in the context of AutoML. It is our belief that AutoML pipelines for clustering could be designed to examine clusterings in different subspaces of a dataset. Not only will this enhance the final clustering quality, but it will also provide results that are easier to interpret.

Similar to subspace clustering, multi-view clustering [96] gains traction lately in the literature. The main idea is that heterogeneous data coming from different sources, but describing closely related phenomena, represent different views that when examined in a combined fashion can provide better insights on certain tasks. Such an example is images that are accompanied with a text description, in which case both hold necessary semantic information to perform clustering. Of course, this concept applies to both supervised and unsupervised clustering. In the case of unsupervised clustering, many subcategories fall under the domain of multi-view clustering [97], such as multi-kernel learning, multi-view graph clustering, and so on, and each category includes a set of proposed approaches.

Although new systems could be developed to select and tune algorithms appropriate for each of these clustering sub-domains, we believe that using algorithm selection and hyperparameter tuning by subspace could prove an efficient alternative. Intuitively, for certain subspaces of a dataset, different algorithm configurations could be more suited and as such each could benefit from the AutoML procedure. Of course, this kind of design is by nature costly in terms of execution time, and further research is needed to test both its validity and feasibility.

7 CONCLUSIONS

AutoML for clustering still poses major challenges, mostly due to the uncertainty of identifying the best clustering result. Various other task-specific design choices also play a crucial role, such as which meta-features to use, how to explore the search space, how to perform evaluation properly, and how to apply a suitable meta-learner. In this survey, we have discussed some of the most prominent methods used in the literature, and we classified meta-learning approaches for algorithm selection and system prototypes that also include hyperparameter tuning. It is our firm belief that clustering for AutoML needs to be further explored and developed to tackle the aforementioned shortcomings, since a lot of generated data are unlabelled. Getting unsupervised learning AutoML on par with supervised learning will enable practitioners to build systems that tackle machine learning modeling in a holistic manner, capable of also creating complex pipelines that combine both supervised and unsupervised learning.

REFERENCES

[1] David H. Wolpert and William G. Macready. 1997. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* 1, 1 (1997), 67–82.

[2] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. 2001. On clustering validation techniques. *J. Intell. Inf. Syst.* 17, 2-3 (2001), 107–145.

[3] Teuvo Kohonen. 1990. The self-organizing map. *Proc. IEEE* 78, 9 (1990), 1464–1480.

[4] Juha Vesanto and Esa Alhoniemi. 2000. Clustering of the self-organizing map. *IEEE Trans. Neural Netw. Learn. Syst.* 11, 3 (2000), 586–600.

[5] Xiaoyan Zhu, Yingbin Li, Jiayin Wang, Tian Zheng, and Jingwen Fu. 2020. Automatic recommendation of a distance measure for clustering algorithms. *ACM Trans. Knowl. Discov. Data* 15, 1, Article 7 (Dec. 2020), 22 pages.

[6] Andrea Baraldi and Palma Blonda. 1999. A survey of fuzzy clustering algorithms for pattern recognition. I. *IEEE Trans. Syst. Man Cybern. Part B* 29, 6 (1999), 778–785.

[7] Abdelkarim Ben Ayed, Mohamed Ben Halima, and Adel M. Alimi. 2014. Survey on clustering methods: Towards fuzzy clustering for big data. In *6th International Conference of Soft Computing and Pattern Recognition (SoCPaR'14)*. IEEE, 331–336.

[8] Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza, Jesús M. Pérez, and Iñigo Perona. 2013. An extensive comparative study of cluster validity indices. *Pattern Recognit.* 46, 1 (2013), 243–256.

[9] John R. Rice. 1976. The algorithm selection problem. *Adv. Comput.* 15 (1976), 65–118.

[10] Joaquin Vanschoren. 2019. Meta-learning. In *Automated Machine Learning - Methods, Systems, Challenges*, Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren (Eds.). Springer, 35–61.

[11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* 12 (2011), 2825–2830.

[12] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13)*, Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy (Eds.). ACM, 847–855.

[13] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2016), 148–175.

[14] Eric Brochu, Vlad M. Cora, and Nando de Freitas. 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR* abs/1012.2599 (2010).

[15] Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. 2022. Recent advances in Bayesian optimization. *CoRR* abs/2206.03301 (2022).

[16] Mitchell McIntire, Daniel Ratner, and Stefano Ermon. 2016. Sparse Gaussian processes for Bayesian optimization. In *32nd Conference on Uncertainty in Artificial Intelligence (UAI'16)*, Alexander Ihler and Dominik Janzing (Eds.). AUAI Press.

[17] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *25th Annual Conference on Neural Information Processing Systems*, John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger (Eds.). ACM, 2546–2554.

[18] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *5th International Conference on Learning and Intelligent Optimization (LION'11) (Lecture Notes in Computer Science)*, Carlos A. Coello Coello (Ed.), Vol. 6683. Springer, 507–523.

[19] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient global optimization of expensive black-box functions. *J. Glob. Optim.* 13, 4 (1998), 455–492.

[20] Dawei Zhan and Huanlai Xing. 2020. Expected improvement for expensive optimization: A review. *J. Glob. Optim.* 78, 3 (2020), 507–544.

[21] Radwa El Shawi, Hudson Lekunze, and Sherif Sakr. 2021. cSmartML: A meta learning-based framework for automated selection and hyperparameter tuning for clustering. In *IEEE International Conference on Big Data (Big Data'21)*, Yixin Chen, Heiko Ludwig, Yicheng Tu, Usama M. Fayyad, Xingquan Zhu, Xiaohua Hu, Suren Byna, Xiong Liu, Jianping Zhang, Shirui Pan, Vagelis Papalexakis, Jianwu Wang, Alfredo Cuzzocrea, and Carlos Ordonez (Eds.). IEEE, 1119–1126.

[22] Christodoulos A. Floudas and Panos M. Pardalos (Eds.). 2009. *Encyclopedia of Optimization, Second Edition*. Springer.

[23] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. 2021. A review on genetic algorithm: Past, present, and future. *Multim. Tools Appl.* 80, 5 (2021), 8091–8126.

[24] Ye Tian, Langchun Si, Xingyi Zhang, Ran Cheng, Cheng He, Kay Chen Tan, and Yaochu Jin. 2022. Evolutionary large-scale multi-objective optimization: A survey. *ACM Comput. Surv.* 54, 8 (2022), 174:1–174:34.

[25] Joannès Vermorel and Mehryar Mohri. 2005. Multi-armed bandit algorithms and empirical evaluation. In *16th European Conference on Machine Learning (ECML'05) (Lecture Notes in Computer Science)*, João Gama, Rui Camacho, Pavel Brazdil, Alípio Jorge, and Luís Torgo (Eds.), Vol. 3720. Springer, 437–448.

[26] Kevin G. Jamieson and Ameet Talwalkar. 2016. Non-stochastic best arm identification and hyperparameter optimization. In *19th International Conference on Artificial Intelligence and Statistics (AISTATS'16) (JMLR Workshop and Conference Proceedings)*, Arthur Gretton and Christian C. Robert (Eds.), Vol. 51. JMLR.org, 240–248.

[27] Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.* 18, 1 (2017), 185:1–185:52.

[28] Bernhard Pfahringer, Hilan Bensusan, and Christophe G. Giraud-Carrier. 2000. Meta-learning by landmarking various learning algorithms. In *17th International Conference on Machine Learning (ICML'00)*, Pat Langley (Ed.). Morgan Kaufmann, 743–750.

[29] Marcílio Carlos Pereira de Souto, Ricardo Bastos Cavalcante Prudêncio, Rodrigo G. F. Soares, Daniel S. A. de Araujo, Ivan G. Costa, Teresa Bernarda Ludermir, and Alexander Schliep. 2008. Ranking and selecting clustering algorithms using a meta-learning approach. In *International Joint Conference on Neural Networks (IJCNN'08)*. IEEE, 3729–3735.

[30] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* 13, 10 (2012), 281–305.

[31] Caroline Tomasini, Leonardo R. Emmendorfer, Eduardo Nunes Borges, and Karina S. Machado. 2016. A methodology for selecting the most suitable cluster validation internal indices. In *31st Annual ACM Symposium on Applied Computing*, Sascha Ossowski (Ed.). ACM, 901–903.

[32] Vasyl Pihur, Susmita Datta, and Somnath Datta. 2007. Weighted rank aggregation of cluster validation measures: A Monte Carlo cross-entropy approach. *Bioinformatics* 23, 13 (2007), 1607–1615.

[33] Matthijs J. Warrens and Hanneke van der Hoef. 2022. Understanding the adjusted rand index and other partition comparison indices based on counting object pairs. *J. Classif.* 39, 3 (2022), 487–509.

[34] Gongde Guo, Hui Wang, David A. Bell, Yaxin Bi, and Kieran Greer. 2003. KNN model-based approach in classification. In *OTM Confederated International Conferences, (CoopIS, DOA, and ODBASE ('03) (Lecture Notes in Computer Science)*, Robert Meersman, Zahir Tari, and Douglas C. Schmidt (Eds.), Vol. 2888. Springer, 986–996.

[35] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 785–794.

[36] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. 2014. Using meta-learning to initialize Bayesian optimization of hyperparameters. In *International Workshop on Meta-learning and Algorithm Selection co-located with 21st European Conference on Artificial Intelligence (MetaSel@ECAI'14) (CEUR Workshop Proceedings)*, Joaquin Vanschoren, Pavel Brazdil, Carlos Soares, and Lars Kotthoff (Eds.), Vol. 1201. CEUR-WS.org, 3–10.

[37] André C. A. Nascimento, Ricardo Bastos Cavalcante Prudêncio, Marcílio Carlos Pereira de Souto, and Ivan G. Costa. 2009. Mining rules for the automatic selection process of clustering methods applied to cancer gene expression data. In *19th International Conference on Artificial Neural Networks (ICANN'09) (Lecture Notes in Computer Science)*, Cesare Alippi, Marios M. Polycarpou, Christos G. Panayiotou, and Georgios Ellinas (Eds.), Vol. 5769. Springer, 20–29.

[38] Daniel Gomes Ferrari and Leandro Nunes de Castro. 2015. Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods. *Inf. Sci.* 301 (2015), 181–194.

[39] Milan Vukicevic, Sandro Radovanovic, Boris Delibasic, and Milija Suknovic. 2016. Extending meta-learning framework for clustering gene expression data with component-based algorithm design and internal evaluation measures. *Int. J. Data Min. Bioinform.* 14, 2 (2016), 101–119.

[40] Bruno Almeida Pimentel and André C. P. L. F. de Carvalho. 2019. A new data characterization for selecting clustering algorithms using meta-learning. *Inf. Sci.* 477 (2019), 203–219.

[41] Yannis Poulakis, Christos Doulkeridis, and Dimosthenis Kyriazis. 2020. AutoClust: A framework for automated clustering based on cluster validity indices. In *20th IEEE International Conference on Data Mining (ICDM'20)*, Claudia Plant, Haixun Wang, Alfredo Cuzzocrea, Carlo Zaniolo, and Xindong Wu (Eds.). IEEE, 1220–1225.

[42] Noy Cohen-Shapira and Lior Rokach. 2021. Automatic selection of clustering algorithms using supervised graph embedding. *Inf. Sci.* 577 (2021), 824–851.

[43] Yue Liu, Shuang Li, and Wenjie Tian. 2021. AutoCluster: Meta-learning based ensemble method for automated unsupervised clustering. In *25th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'21) (Lecture Notes in Computer Science)*, Kamal Karlapalem, Hong Cheng, Naren Ramakrishnan, R. K. Agrawal, P. Krishna Reddy, Jaideep Srivastava, and Tanmoy Chakraborty (Eds.), Vol. 12714. Springer, 246–258.

[44] Dennis Tschechlov, Manuel Fritz, and Holger Schwarz. 2021. AutoML4Clust: Efficient autoML for clustering analyses. In *24th International Conference on Extending Database Technology*, Yannis Velegrakis, Demetris Zeinalipour-Yazti, Panos K. Chrysanthis, and Francesco Guerra (Eds.). OpenProceedings.org, 343–348.

[45] Radwa El Shawi and Sherif Sakr. 2022. TPE-AutoClust: A tree-based pipeline ensemble framework for automated clustering. In *IEEE International Conference on Data Mining Workshops (ICDM'22)*, K. Selçuk Candan, Thang N. Dinh, My T. Thai, and Takashi Washio (Eds.). IEEE, 1144–1153.

[46] Radwa El Shawi and Sherif Sakr. 2022. cSmartML-Glassbox: Increasing transparency and controllability in automated clustering. In *IEEE International Conference on Data Mining Workshops (ICDM'22)*, K. Selçuk Candan, Thang N. Dinh, My T. Thai, and Takashi Washio (Eds.). IEEE, 47–54.

[47] R. A. Johnson and D. W. Wichern. 2007. *Applied Multivariate Statistical Analysis*. Pearson Prentice Hall.

[48] Mark A. Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explor.* 11, 1 (2009), 10–18.

[49] Krzysztof Dembczynski, Wojciech Kotlowski, and Roman Slowinski. 2008. Maximum likelihood rule ensembles. In *25th International Conference on Machine Learning (ICML'08) (ACM International Conference Proceeding Series)*, William W. Cohen, Andrew McCallum, and Sam T. Roweis (Eds.), Vol. 307. ACM, 224–231.

[50] Milan Vukicevic, Boris Delibasic, Milos Jovanovic, Milija Suknovic, and Zoran Obradovic. 2011. Internal evaluation measures as proxies for external indices in clustering gene expression data. In *IEEE International Conference on Bioinformatics and Biomedicine (BIBM'11)*, Fang-Xiang Wu, Mohammed Javeed Zaki, Shinichi Morishita, Yi Pan, Stephen Wong, Anastasia Christianson, and Xiaohua Hu (Eds.). IEEE Computer Society, 574–577.

[51] Milan Vukicevic, Boris Delibasic, Milos Jovanovic, Milija Suknovic, and Zoran Obradovic. 2012. A method for design of data-tailored partitioning algorithms for optimizing the number of clusters in microarray analysis. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'12)*. IEEE, 252–259.

[52] Boris Delibasic, Milan Vukicevic, Milos Jovanovic, Kathrin Kirchner, Johannes Ruhland, and Milija Suknovic. 2012. An architecture for component-based design of representative-based clustering algorithms. *Data Knowl. Eng.* 75 (2012), 78–98.

[53] Donald Michie, David J. Spiegelhalter, and Charles C. Taylor. 1994. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

[54] Alexandros Kalousis. 2002. *Algorithm Selection via Meta-learning*. Ph.D. Dissertation. University of Geneva.

[55] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*, Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (Eds.). ACM, 701–710.

[56] James Bergstra, Daniel Yamins, and David D. Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *30th International Conference on Machine Learning (ICML'13) (JMLR Workshop and Conference Proceedings)*, Vol. 28. JMLR.org, 115–123.

[57] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and efficient hyperparameter optimization at scale. In *35th International Conference on Machine Learning (ICML'18) (Proceedings of Machine Learning Research)*, Jennifer G. Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 1436–1445.

[58] Dorin Comaniciu and Peter Meer. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* 24, 5 (2002), 603–619.

[59] William F. Punch. 2001. Book review: Genetic programming—An introduction: On the automatic evolution of computer programs and its applications. *Genet. Program. Evolv. Mach.* 2, 2 (2001), 193–195.

[60] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary algorithms made easy. *J. Mach. Learn. Res.* 13 (2012), 2171–2175.

[61] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* 6, 2 (2002), 182–197.

[62] Amit Banerjee and Rajesh N. Davé. 2004. Validating clusters using the Hopkins statistic. In *IEEE International Conference on Fuzzy Systems (FUZZ'04)*. IEEE, 149–153.

[63] Yu-Feng Li, Hai Wang, Tong Wei, and Wei-Wei Tu. 2019. Towards automated semi-supervised learning. In *33rd AAAI Conference on Artificial Intelligence (AAAI'19), 31st Innovative Applications of Artificial Intelligence Conference (IAAI'19), 9th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI'19)*. AAAI Press, 4237–4244.

[64] Tom Ronan, Shawn Anastasio, Zhijie Qi, Pedro Henrique S. Vieira Tavares, Roman Sloutsky, and Kristen M. Naegle. 2018. OpenEnsembles: A Python resource for ensemble clustering. *J. Mach. Learn. Res.* 19 (2018), 26:1–26:6.

[65] Qian Jiang, Xin Jin, Shin-Jye Lee, and Shaowen Yao. 2019. A new similarity/distance measure between intuitionistic fuzzy sets based on the transformed isosceles triangles and its applications to pattern recognition. *Expert Syst. Applic.* 116 (2019), 439–453.

[66] Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, and Ruili Wang. 2018. Efficient kNN classification with different numbers of nearest neighbors. *IEEE Trans. Neural Netw. Learn. Syst.* 29, 5 (2018), 1774–1785.

[67] S. Zhang, J. Li, and Y. Li. 2023. Reachable distance function for KNN classification. *IEEE Trans. Knowl. Data Eng.* 35, 07 (July 2023), 7382–7396.

[68] Shichao Zhang and Jiaye Li. 2023. KNN classification with one-step computation. *IEEE Trans. Knowl. Data Eng.* 35, 3 (2023), 2711–2723.
[69] Shichao Zhang, Jiaye Li, Wenzhen Zhang, and Yongsong Qin. 2022. Hyper-class representation of data. *Neurocomputing* 503 (2022), 200–218.
[70] Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, and Junjie Wu. 2010. Understanding of internal clustering validation measures. In *10th IEEE International Conference on Data Mining (ICDM'10)*, Geoffrey I. Webb, Bing Liu, Chengqi Zhang, Dimitrios Gunopulos, and Xindong Wu (Eds.). IEEE Computer Society, 911–916.
[71] James C. Bezdek and Nikhil R. Pal. 1998. Some new indexes of cluster validity. *IEEE Trans. Syst. Man Cybern. Part B* 28, 3 (1998), 301–315.
[72] Jeffrey D. Banfield and Adrian E. Raftery. 1993. Model-based Gaussian and non-Gaussian clustering. *Biometrics* 49, 3 (1993), 803–821.
[73] Lawrence J. Hubert and Joel R. Levin. 1976. A general statistical framework for assessing categorical clustering in free recall. *Psychol. Bull.* 83, 6 (1976), 1072–1080.
[74] Tadeusz Caliński and Jerzy Harabasz. 1974. A dendrite method for cluster analysis. *Commun. Statist.-Theor. Meth.* 3, 1 (1974), 1–27.
[75] Maria Halkidi and Michalis Vazirgiannis. 2008. A density-based cluster validity approach using multi-representatives. *Pattern Recognit. Lett.* 29, 6 (2008), 773–786.
[76] David L. Davies and Donald W. Bouldin. 1979. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.* 1, 2 (1979), 224–227.
[77] J. C. Dunn. 1973. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *J. Cybern.* 3, 3 (1973), 32–57.
[78] Jerome H. Friedman and Lawrence C. Rafsky. 1979. Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests. *Ann. Statist.* 7, 4 (1979), 697–717.
[79] Julia Handl, Joshua D. Knowles, and Douglas B. Kell. 2005. Computational cluster validation in post-genomic data analysis. *Bioinformatics* 21, 15 (2005), 3201–3212.
[80] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *J. Classif.* 2, 1 (1985), 193–218.
[81] Ujjwal Maulik and Sanghamitra Bandyopadhyay. 2002. Performance evaluation of some clustering algorithms and validity indices. *IEEE Trans. Pattern Anal. Mach. Intell.* 24, 12 (2002), 1650–1654.
[82] Peter Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 20, 1 (1987), 53–65.
[83] Frank B. Baker and Lawrence J. Hubert. 1975. Measuring the power of hierarchical cluster analysis. *J. Amer. Statist. Assoc.* 70, 349 (1975), 31–38.
[84] John O. McClain and Vithala R. Rao. 1975. CLUSTISZ: A program to test for the quality of clustering of a set of objects. *J. Market. Res.* 12, 4 (1975), 456–460.
[85] Glen W. Milligan and Martha C. Cooper. 1985. An examination of procedures for determining the number of clusters in a data set. *Psychometrika* 50, 2 (1985), 159–179.
[86] Malay Kumar Pakhira, Sanghamitra Bandyopadhyay, and Ujjwal Maulik. 2004. Validity index for crisp and fuzzy clusters. *Pattern Recognit.* 37, 3 (2004), 487–501.
[87] David A. Ratkowsky and Geoffrey N. Lance. 1978. A criterion for determining the number of groups in a classification. *Austral. Comput. J.* 10, 4 (1978), 115–117.
[88] S. Ray and R. H. Turi. 1999. Determination of number of clusters in k-means clustering and application in colour image segmentation. In *4th International Conference on Advances in Pattern Recognition and Digital Techniques*. Narosa Publishing House, 137–143.
[89] A. J. Scott and M. J. Symons. 1971. Clustering methods based on likelihood ratio criteria. *Biometrics* 27, 2 (1971), 387–397.
[90] Leo A. Goodman and William H. Kruskal. 1954. Measures of association for cross classifications. *J. Amer. Statist. Assoc.* 49, 268 (1954), 732–764.
[91] Xuanli Lisa Xie and Gerardo Beni. 1991. A validity measure for fuzzy clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* 13, 8 (1991), 841–847.
[92] Noy Cohen-Shapira and Lior Rokach. 2021. TRIO: Task-agnostic dataset representation optimized for automatic algorithm selection. In *IEEE International Conference on Data Mining (ICDM'21)*, James Bailey, Pauli Miettinen, Yun Sing Koh, Dacheng Tao, and Xindong Wu (Eds.). IEEE, 81–90.
[93] Hadi S. Jomaa, Lars Schmidt-Thieme, and Josif Grabocka. 2021. Dataset2Vec: Learning dataset meta-features. *Data Min. Knowl. Discov.* 35, 3 (2021), 964–985.
[94] Teddy Lazebnik, Amit Somech, and Abraham Itzhak Weinberg. 2022. SubStrat: A subset-based optimization strategy for faster AutoML. *Proc. VLDB Endow.* 16, 4 (2022), 772–780.

[95] Lance Parsons, Ehtesham Haque, and Huan Liu. 2004. Subspace clustering for high dimensional data: A review. *SIGKDD Explor.* 6, 1 (2004), 90–105.
[96] Lele Fu, Pengfei Lin, Athanasios V. Vasilakos, and Shiping Wang. 2020. An overview of recent multi-view clustering. *Neurocomputing* 402 (2020), 148–161.
[97] Yan Yang and Hao Wang. 2018. Multi-view clustering: A survey. *Big Data Min. Anal.* 1, 2 (2018), 83–107.

Received 17 April 2023; revised 16 January 2024; accepted 19 January 2024