AutoClues: Exploring Clustering Pipelines via AutoML and Diversification

AutoClues: Exploring Clustering Pipelines via AutoML and Diversification

Matteo Francia[1][0000−0002−0805−1051], Joseph Giovanelli[1][0000−0002−0990−3893], and Matteo Golfarelli[1][0000−0002−0437−0725]

University of Bologna, Cesena, Italy
{m.francia,j.giovanelli,matteo.golfarelli}@unibo.it

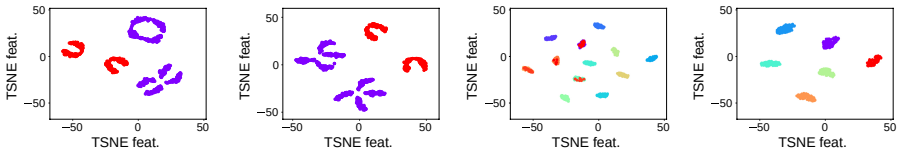
Abstract. AutoML has witnessed effective applications in the field of supervised learning – mainly in classification tasks – where the goal is to find the best machine-learning pipeline when a ground truth is available. This is not the case for unsupervised tasks that are by nature exploratory and they are performed to unveil hidden insights. Since there is no right result, analyzing different configurations is more important than returning the best-performing one. When it comes to exploratory unsupervised tasks – such as cluster analysis – different facets of the datasets could be interesting for the data scientist; for instance, data items can be effectively grouped together in different subspaces of features. In this paper, AutoClues explores and returns a dashboard of both relevant and diverse clusterings via AutoML and diversification. AutoML ensures that the explored pipelines for cluster analysis (including pre-processing steps) compute good clusterings. Then, diversification selects, out of the explored clusterings, the ones conveying different clues to the data scientists.

Keywords: AutoML, Clustering, Diversification

1 Introduction

Thanks to the abundant presence of data, machine learning (ML) has been employed in a variety of fields (e.g., urban mobility [27]). Depending on the scope of the analysis, the task is defined either supervised (i.e., leveraging a ground truth; e.g., classification) or unsupervised (i.e., without ground truth; e.g., cluster analysis). Data scientists design the workflow as a *ML pipeline*; namely, a series of pre-processing steps (e.g., features selection) are in charge of shaping the data so that the final analysis step can produce the best result. For each step of the pipeline, there are alternative algorithms (e.g., SPEC [31] for feature selection, KMeans [1] for cluster analysis), each with hyperparameters to tune (e.g., the number of features or clusters, respectively).

It is well known that, given the exponential search space, the tuning process is tedious and overwhelming. *Automated Machine Learning* (AutoML) aids in a smart exploration of the hyperparameter search space [14] and lets data scientists focus on analyzing and interpreting the extracted results. AutoML has been proven to be effective on supervised tasks where the ground truth eases



(a) Cl [28] AMI=0.41, sil=0.41 (b) FS + Cl AMI=0.41, sil=0.47 (c) FS + N + Cl AMI=0.9, sil=0.87 (d) FS + N + OR + Cl AMI=1.0, sil=0.92

Fig. 1: Approach motivation.
Feature Selection (FS) - Normalization (N) - Outlier Removal (OR) - Clustering (Cl)

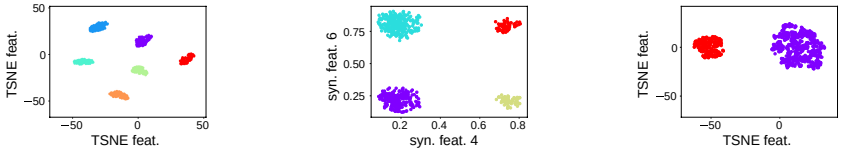
the evaluation of the hyperparameters optimality [25,8]; yet, when it comes to unsupervised tasks, the road is not paved yet [2]. In this paper, we focus on the unsupervised task of *(crisp) cluster analysis* [1] that returns a partitioning of the original dataset based on the similarity of data items. Cluster analysis is exploratory by nature since, given that no ground truth is available, there is no "correct" result [16] and the aim for the data scientist is to uncover clues hidden in the data. The two main limitations of the AutoML approaches in the literature are (i) auto-tuning is applied to the ML step only, disregarding the pre-processing ones, and (ii) only the most-performing pipeline configuration is returned; although this is reasonable in the supervised context, it provides limited information in unsupervised one due the exploratory nature of the analysis.

To overcome the previous limitations, we devise AutoClues: an *end-to-end* AutoML approach that provides a *dashboard* of *relevant* and *different* clusterings. In particular, we focus on the following contributions:

- *generalizing* AutoML formulation to deal with unsupervised ML pipelines;
- *tuning* a thorough ML pipeline to discover clusterings that would have been unrevealed otherwise;
- *diversifying* the generated clusterings to ensure that the dashboard is both high-quality and leads to different insights (i.e., disclose something new);
- *providing* a customizable generator of synthetic datasets for benchmarking in (crisp) cluster analysis.

To let the reader appreciate the novelty of AutoClues, we rely on a synthetic 10-dimensional (10D) dataset that includes 6 natural clusters. Figure 1a shows the t-SNE [19] visualization[1] of the clustering obtained applying AutoML4Clust [28], an approach of the literature, that solely tunes the clustering step Cl. Figures 1b to 1d show the clusterings obtained by tuning an ML pipeline that incrementally includes feature selection (FS), normalization (N), and outlier removal (OR). In (b), FS identifies the most relevant features; in (c), N standardizes such features thus avoiding bias due to different domain ranges; in (d), OR drops any data items that are not representative. It is apparent how the tuning improves throughout the different steps, making it possible for Cl to properly

[1] This dimensionality reduction visualizes high-dimensional clusterings in 2D, preserving distance proportions. We apply it with the default Scikit-learn hyperparameters.



(a) FS + N + OR + Cl
AMI=1.0, sil=0.92

(b) FS + OR + Cl
AMI=0.65, sil=0.86

(c) FS + OR + Cl
AMI=0.38, sil=0.74

Fig. 2: Relevant and diverse clusterings returned by AutoClues.
Feature Selection (FS) - Normalization (N) - Outlier Removal (OR) - Clustering (Cl)

detect the 6 natural clusters. To quantitatively understand the improvements, we rely on the silhouette index $sil \in [-1, 1]$ (the higher the better), an estimation of the "goodness" of a clustering considering solely the data itself; namely, the *separability* between the clusters and their *cohesion* [32]. While in real-case unsupervised problems the ground truth is not available, for our synthetic example we can also measure how the returned clusters match the synthetic ones through the adjusted mutual information [30] $AMI \in [0, 1]$ (the higher the better).

As to the limitation of returning only the most-performing pipeline configuration, Figure 2 depicts an example of an AutoClues dashboard with three different facets (clustering). Along with the expected clusters (a), the representation in (b) unveils 4 macro clusters in 2 original features, while the representation in (c) highlights a large gap of 2 main clusters. In real-world problems, those facets may help the data scientist to understand the data.

2 Related works

Although the active development of data pre-processing techniques in cluster analysis (e.g., outlier removal, feature selection), and the evidence of their benefits, there is no trace of automatic solutions that tunes a thorough ML pipeline.

Former approaches employed model-free techniques (i.e., without a model to drive the optimization) to tune the combination of number of clusters and number of features. Evolutionary algorithms are population-based heuristics inspired by biological evolution mechanisms (e.g., reproduction, mutation) or physical phenomena (e.g., particle swarm, black holes). The population is intended as the search space, individuals are configurations, and a mutation mechanism allows the modification of the current candidate, hence the exploration. Recent works that follow this modus operandi (i.e., MOGA [5], MODE-cf [12], TPE-AutoClust [6]) compose the candidate with a string that encodes information about both the feature selection and the clustering. Authors in [23] leverage simulated annealing, an algorithm that comes from a technique involving heating and controlled cooling of material in metallurgy. In [22], it is leveraged a gravitational search algorithm, inspired by the theory of Newtonian gravity.

Model-based techniques leverage past evaluations to fit a model and visit the most prominent configurations. Such techniques have been proven to achieve

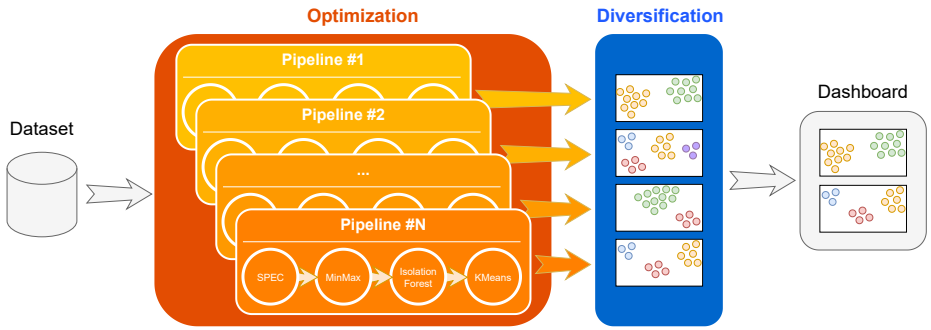


Fig. 3: Overview of AutoClues.

extremely good results, specifically in the (supervised) AutoML field where a pipeline has to be instantiated with different algorithms and hyperparameters. Authors in [28,21,18] apply such techniques on cluster analysis, but do not consider any pre-processing phase. In MALSS [15], the authors apply AutoML with simple (no-clustering-oriented) pre-processing, and still for the aim of suggesting the number of clusters. In [7], authors introduce a pipeline for unsupervised clustering, yet specifically tailored for multivariate time series originating from HPC job monitoring. Finally, such approaches retrieve just one solution.

3 AutoClues

AutoClues aims at (i) tuning the whole ML pipeline and (ii) suggesting multiple clusterings to provide the data scientist different facets of the datasets. In literature, these two problems are addressed through *optimization* and *diversification* techniques respectively. Figure 3 shows an overview of the architecture.

3.1 Formalization

Optimization We seek for the most "correct" pipeline. A pipeline is a sequence of steps that process a dataset in order to return a clustering and, at each step, an algorithm can be picked among several alternatives.

More formally, a *dataset* D is a collection of $|D|$ data items that is characterized by a set of features $\mathcal{F}$ (i.e., columns). An *Algorithm* A is a function that transforms a dataset D' into a new dataset D'' and has a set of (possibly empty) *hyperparameters* that regulates its behavior. Each hyperparameter has a domain, and the possible algorithm configurations are represented as the Cartesian product of all the hyperparameter domains, denoted with Λ_A. We call *algorithm instance* $\lambda_A \in \Lambda_A$ an (ordered) tuple in which each hyperparameter has been assigned with a value from its domain. For instance, a clustering algorithm is KMeans and two tunable hyperparameters can be: the number of clusters $k \in \mathbb{N}^+$ and the maximum iterations $iter \in \mathbb{N}^+$.

A pipeline *step* S is a set of algorithms that can be selected as alternatives to carry out the step goal, including the possibility to return the dataset as-is through the "Identity" algorithm I. The step domain is defined as $\Lambda_S = \Lambda_{A_1} \uplus \ldots \uplus \Lambda_{A_{|S|}}$, therefore the disjoint union of the domain of each algorithm[2]. λ_S denotes the algorithm instance selected for the step.

A *Pipeline* P is a sequence of steps, its domain is the Cartesian Product of the step domains $\Lambda_P = \Lambda_{S_1} \times \ldots \times \Lambda_{S_{|P|}}$, and a *pipeline instance* is an ordered tuple of algorithm instances $\lambda_P = (\lambda_{S_1}, \ldots, \lambda_{S_{|P|}}) \in \Lambda_P$. The input of λ_{A_i} is the output of $\lambda_{A_{i-1}}$ and the initial dataset D is the input of λ_{A_1}. In our use case, the last step is mandatory since it fulfills cluster analysis by returning a clustering.

A *(crisp) Clustering* is a partition of the dataset into a set of non-overlapping clusters $C = \{c_1, \ldots, c_{|C|}\}$ (i.e., groups of data items) by minimizing the distance of data items in the same cluster and maximizing the distance of different clusters. Given a pipeline domain Λ_P and a dataset D, the optimal instance is

$$\lambda_P^* = argmax_{\lambda_P \in \Lambda_P} rel(C) \tag{1}$$

where rel is a goodness metric for C obtained by applying λ_P on D.

Diversification The process of returning a set of relevant and diverse solutions – clusterings in our case – is known as diversification, a multi-objective optimization problem that can be formulated as follows. Let $\mathcal{C}$ be the set of clusterings that have been explored in the pipeline optimization process (Equation (1)), then our goal is selecting a set of α clusterings $\mathcal{C}^* \subseteq \mathcal{C}$ maximizing a *score* that represents a tradeoff between finding *relevance* and *diversity*.

$$\mathcal{C}^* = argmax_{\hat{\mathcal{C}} \subseteq \mathcal{C}, \alpha = |\hat{\mathcal{C}}|} score(\beta, \hat{\mathcal{C}}) \tag{2}$$

$$score(\beta, \hat{\mathcal{C}}) = (1 - \beta)\ rel(\hat{\mathcal{C}}) + \beta\ div(\hat{\mathcal{C}}) \tag{3}$$

where $\beta \in [0..1]$ is the tradeoff parameter and

$$rel(\hat{\mathcal{C}}) = (|\hat{\mathcal{C}}| - 1) \sum_{C \in \hat{\mathcal{C}}} rel(C) \tag{4}$$

$$div(\hat{\mathcal{C}}) = \sum_{C_i \in \hat{\mathcal{C}}} \sum_{C_j \in (\hat{\mathcal{C}} \setminus C_i)} div(C_i, C_j) \tag{5}$$

$div(\hat{\mathcal{C}})$ is the sum of pairwise clustering diversity comparisons and $rel(\hat{\mathcal{C}})$ is the sum of clustering relevances; $rel(\hat{\mathcal{C}})$ entails a multiplication factor $|\hat{\mathcal{C}}| - 1$ to make $rel(\hat{\mathcal{C}})$ and $div(\hat{\mathcal{C}})$ comparable.

3.2 Implementation

Table 1 reports the pipeline with steps, algorithms, and hyperparameters; AutoClues is available on GitHub [3]. The first step is *Feature selection*. Since cluster

[2] If an algorithm has no hyperparameters ($\Lambda_A = \varnothing$), we set a placeholder $\Lambda_A = \{1\}$.

[3] `https://github.com/big-unibo/autoclues`

Step	Algorithm	#Hyper.	$\lvert\Lambda_A\rvert$
Feature selection	SPEC [31]	1	$\lvert\mathcal{F}\rvert - 1$
	WKMeans [13]	2	$3 \cdot (\lvert\mathcal{F}\rvert - 1)$
	Pearson Filtering	1	10
Normalization	Standardization	0	1
	Robust Scaling	3	12
	MinMax	0	1
Outlier removal	Local Outlier Factor [3]	1	3
	Isolation Forest [17]	1	3
Clustering	KMeans [1]	1	$\sqrt{\lvert D\rvert}$
	Agglomerative clustering [20]	1	$\sqrt{\lvert D\rvert}$

Table 1: Steps and algorithms optimized by AutoClues.

analysis is particularly sensitive to non-informative and correlated features, we leverage algorithms from spectral family and based on the Pearson correlation, respectively. Follows, *Normalization* to adjust values on different scales and ensure that all the features contribute equally to the cluster formation. Here, the literature has a considerable consensus on the well-known techniques such as Standardization, Robust scaling, and Min-max. The last pre-processing step is *Outlier Removal* for discarding any data points that are not representative of the cluster, we leverage Local Outlier Factor [3] and and Isolation Forest [17]. Finally, the *Clustering* step. Since we focus on *crisp spherical* clustering algorithms, we consider KMeans [1] and Agglomerative Clustering [20] with complete linkage. As to the order of steps, in [11], the authors reduce the combinations of the former steps; we further constrained the order of the steps by computing several experiments and observing the impact of the alternatives.

Optimization To explore promising pipeline instances, we leverage Bayesian Optimization (BO) [14]. BO is iterative: as it explores hyperparameter configurations, it progressively builds an accurate model of the domain to decide the next configuration to explore. The exploration continues until a budget in terms of either iterations or time is reached.

Selecting the optimal pipeline instance requires the relevance metric (i.e., $rel(C)$) to evaluate the goodness of the retrieved clustering. In AutoClues, such a metric is customizable. Well-known metrics leveraged for spherical clustering are: the silhouette index (SIL) [32], contrasting the average distance to elements in the same cluster with the average distance to elements in other clusters, and the Davies-Bouldin Index (DBI) [4], computing the average similarity between clusters by considering their own size. However, clustering metrics show a bias toward lower dimensionalities i.e., yielding higher scores when fewer features are chosen [16,12]. To overcome this, we employ t-SNE [19] (with default Scikit-learn hyperparameters) to project the clusterings to a latent 2D space. Distances in this latent space are preserved, and we can compute the chosen metric atop, enabling a fair evaluation of clusterings across diverse feature spaces.

Diversification Implementing diversification in cluster analysis involves assessing the extent to which two clusterings differ from each other, hence how the returned dashboard looks. It is crucial to rely on a metric that considers not only shared cluster membership but also structural interrelationships.

Information theory introduces the concept of Mutual Information, quantifying the degree of dependence between two variables and, more specifically, Adjusted Mutual Information (AMI) allows for chance agreement[4]—providing more robust and meaningful measures. When applied to cluster analysis, AMI $\in [0, 1]$ considers the labels assigned to data points within clusters, assuming higher values when the clusters in one partition align with those in another. Since we need a diversity metric, we adapt the formula as in:

$$div(C_i, C_j) = 1 - AMI(C_i, C_j)$$

where C_i, C_j clusterings coming from different pipeline instances.

Finally, considering that the diversification problem is NP-hard, we compute it by exploiting the MMR heuristic solution[5] [29] that selects the best-performing clustering and iteratively adds the clusterings that most diversify the outcome.

4 Benchmark Generation and Empirical Evaluation

Evaluation in cluster analysis consists of assessing the approach performance in finding well-separated clusters and – if available – their alignment with a hypothetical ground truth. It is crucial to test on datasets that conform with the leveraged clustering algorithms, e.g., in our case, containing crisp spherical clusters. Yet, there is a lack of benchmarks (i.e., suites of datasets for fair comparisons) and the few available [9,10,26] are tailored to their specific scenarios. This translates into approaches relying upon datasets from supervised tasks, with no guarantees on the underlying clusters' shape.

In Section 4.1, we introduce a benchmarking generator and a suite of synthetic datasets. In Section 4.2, we leverage such a suite to assess the effectiveness and efficiency of AutoClues. Finally, in Section 4.3, we rely on real datasets to provide a comparison against other approaches in the literature.

4.1 Benchmark Generation

The synthetic benchmarking generator is available at `https:/github.com/big-unibo/clustering_benchmarking`. We create datasets of $|D|$ instances, defining $|C|$ natural hyper-spherical clusters in a space of $|\mathcal{F}|$ features. Then, we blur such clusters by posing common challenges faced by clustering algorithms. This includes noise on instances $\sigma(D)$, such as the presence of outliers, and noise on features $\sigma(F)$, such as irrelevant, correlated, or distorted features.

[4] In statistics, it serves as a baseline for assessing the significance in random variations.
[5] We use the default hyperparameter $\beta = 0.5$, and set α according to the test at hand.

Dataset	Characteristics							AutoClues Performance			
	$\|D\|$	$\|\mathcal{F}\|$	$\|\mathcal{C}\|$	$\sigma(D)$	$\sigma(\mathcal{F})$	SIL_N	SIL_B	SIL	AMI	Score	Div. time (s)
`syn1`	2905	2	3	0.15	0.50	0.72	0.48	0.79	1.0	4.12	$1.64 \cdot 10^3$
`syn2`	264	5	3	0.25	0.20	0.64	0.4	0.7	0.83	4.76	$1.74 \cdot 10^2$
`syn3`	900	7	12	0.11	0.14	0.88	0.6	0.92	1.0	4.39	$1.51 \cdot 10^3$
`syn4`	1446	2	13	0.22	1.00	0.59	0.31	0.85	0.95	3.92	$2.99 \cdot 10^3$
`syn5`	1673	4	8	0.17	0.25	0.71	0.54	0.81	0.99	4.22	$2.18 \cdot 10^3$
`syn6`	2905	2	3	0.15	0.50	0.72	0.61	0.84	0.87	4.63	$1.07 \cdot 10^3$
`syn7`	264	5	3	0.25	0.20	0.64	0.41	0.72	0.86	3.78	$1.67 \cdot 10^2$
`syn8`	1639	8	21	0.13	0.00	0.87	0.69	0.91	1.0	4.45	$3.13 \cdot 10^3$
`syn9`	525	3	2	0.21	0.00	0.66	0.42	0.69	0.87	3.96	$3.67 \cdot 10^2$
`syn10`	1446	2	13	0.22	1.00	0.59	0.32	0.86	0.97	4.03	$4.09 \cdot 10^3$
`syn11`	4813	10	3	0.27	0.40	0.46	0.17	0.09	0.42	2.92	$1.08 \cdot 10^2$
`syn12`	2905	2	3	0.15	0.50	0.72	0.57	0.79	1.0	4.48	$1.62 \cdot 10^3$
`syn13`	264	5	3	0.25	0.20	0.64	0.4	0.74	0.89	4.37	$2.75 \cdot 10^2$
`syn14`	525	3	2	0.21	0.00	0.66	0.22	0.71	0.9	4.38	$5.81 \cdot 10^2$
`syn15`	2905	2	3	0.15	0.50	0.72	0.61	0.81	1.0	4.06	$1.13 \cdot 10^3$
`syn16`	264	5	3	0.25	0.20	0.64	0.49	0.7	0.88	3.45	$1.84 \cdot 10^2$
`syn17`	900	7	12	0.11	0.14	0.88	0.41	0.92	1.0	4.62	$1.45 \cdot 10^3$
`syn18`	525	3	2	0.21	0.00	0.66	0.3	0.69	0.84	4.34	$2.85 \cdot 10^2$
`syn19`	2905	2	3	0.15	0.50	0.72	0.53	0.86	0.87	4.79	$1.28 \cdot 10^3$
`syn20`	264	5	3	0.25	0.20	0.64	0.44	0.7	0.88	3.74	$2.13 \cdot 10^2$

Table 2: Dataset characteristics and performance achieved by AutoClues.

To obtain datasets with different characteristics, we set boundaries for each of these dimensions and sample within them according to the Sobol sequence [24], a quasi-random low-discrepancy search converging to an equi-distributed coverage. In particular, the defined boundaries are: $|D|$ between 100 and 5000, $|F|$ between 2 and 10, $|C|$ between 2 and $\sqrt{|D|}$, $\sigma(D)$ between 0.1 and 0.3, and $\sigma(F)$ between 0 and 1. Table 2 provides a suite of 20 synthetic datasets.

Dataset complexity can be examined via the silhouette $SIL \in [0, 1]$, the higher the simpler. SIL_N measures the cohesion and separability of the natural clusters C in their original feature space, while SIL_B measures the silhouette of blurred clusters (i.e. after introducing noise (σ)). The former SIL_N indicates the presence of well-separated clusters. With the first quartile $Q1 = 0.64$, median $Q2 = 0.66$, and third quartile $Q3 = 0.72$, we observe that 25% of datasets are complex already at this stage. The latter SIL_B registers significantly lower values: `syn11` emerges as an especially complex dataset with $SIL_B = 0.17$ but, overall, we confirm the presence of a good distribution between more and less challenging datasets: $Q1 = 0.36$, $Q2 = 0.43$, and $Q3 = 0.55$.

4.2 Effectiveness and efficiency

We test AutoClues on the suite of generated synthetic datasets. For the optimization, we adopted the silhouette SIL index as a relevance objective metric and a budget of 7200 seconds (2 hours); for the diversification we set $\alpha = 3$ and

$\beta = 0.5$, resulting in a dashboard of 3 clusterings where relevance and diversity are weighted equally. Tests are run on a single core of an Intel Core i7 machine at 3.20 GHz and 64 GB of main memory. Given an AutoClues dashboard, Table 2 provides the maximum $SIL \in [0, 1]$ as the cohesion of the found clusters and the maximum $AMI \in [0, 1]$ as the alignment with the natural ones[6]. Besides, we report the dashboard score of Equation (2), summarizing the overall relevance and diversity in the dashboard, and the diversification computation time.

Effectiveness The achieved silhouette not only shows AutoClues' ability to find well-separated clusters in 19 cases out of 20, but it also demonstrates to overcome the silhouette of natural clusters SIL_N. This is achieved through pre-processing such as projecting natural clusters into more compact feature subsets and mitigating potential noise. The only critical exception is `syn11`, already highlighted in the previous section. AMI also confirms strong agreement between retrieved and natural clusters ($Q1 = 0.87$, $Q2 = 0.9$, $Q3 = 1$).

As to the dashboard, considering the experimental setting ($\alpha = 3$, $\beta = 0.5$, $rel, div \in [0, 1]$), the score is bounded in $[0, 6]$. Notably, given the inherent trade-off relationship between *rel* and *div* in the context of a diversification problem, it is noteworthy that scores at the boundaries are less likely to manifest, with values around 4 already acknowledged as high-quality [29]. Analogously, quartile values of the score ($Q1 = 4$, $Q2 = 4.34$, $Q3 = 4.47$) confirm Autoclues' ability to find relevant and diverse clusterings.

Efficiency Table 2 reports the computation time for diversification, when the optimization budget is set to 2 hours. We can observe 25% of datasets compute the dashboard in less than $Q1 = 2.44 \cdot 10^2$ seconds (4 minutes), 50% in less than $Q2 = 1.07 \cdot 10^3$ seconds (18 minutes), and 75% in less than $Q3 = 1.56 \cdot 10^3$ seconds (26 minutes). Besides, reducing the optimization budget leads to a decrease in diversification time, maintaining high performance. Figure 4 shows how AutoClues converges to the values reported in Table 2. Dashboards are generated at different snapshots during the 2-hour optimization process, and the same metrics are computed: SIL, AMI, dashboard score, and diversification time. We summarize the information by plotting the mean and standard deviation among the whole suite, the convergence is quantified as a progress ratio relative to the final achieved performance. Notably, the optimization time is illustrated in a logarithmic scale, highlighting already fast convergence. After 60 seconds of optimization, we have clusterings with SIL and AMI as good as 80% and 95% of the optimal and a dashboard almost 60% of the final within a negligible diversification time—roughly 2% of the total, 30 seconds on average.

Within 300 seconds of optimization, an average of 90% of SIL, 97% of AMI, and 75% of the dashboard score are registered with a diversification cost of 10%—on average, 1 minute and half. The trends of SIL and AMI saturated 100% right afterward, while both dashboard score and cost increase linearly until 900 seconds of optimization, in which the dashboard achieves 90% of its score within

[6] Metrics are computed on the original dataset (i.e., no t-SNE distortion)

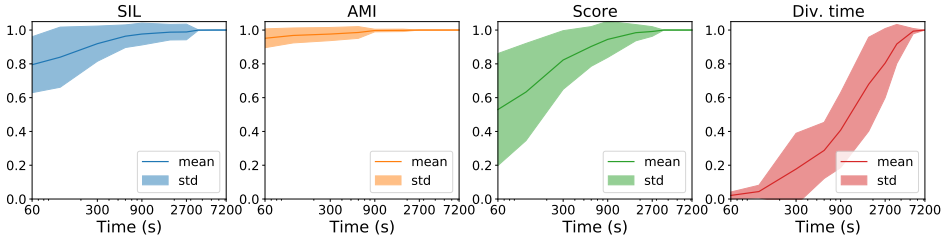


Fig. 4: AutoClues convergence through time (in log scale).

Dataset	Characteristics			Performance					
				DBI ↓				SIL ↑	
	$\|D\|$	$\|\mathcal{F}\|$	$\|C\|$	MOGA	MODE-cf	MALSS	AutoClues	MALSS	AutoClues
`blood`	748	4	2	-	-	0.3	**0**	0.73	**1**
`breast`	106	9	6	-	0.7	1.6	**0.54**	0.16	**0.60**
`ecoli`	327	7	5	-	0.92	**0.35**	0.46	**0.72**	0.46
`iris`	150	4	3	0.39	0.67	0.6	**0.38**	0.57	**0.71**
`seeds`	210	7	3	-	-	0.8	**0.4**	**0.45**	0.37
`thyroid`	215	5	3	-	-	0.64	**0.2**	0.6	**0.92**
`vehicle`	846	18	4	-	-	0.6	**0.15**	0.61	**0.72**
`wine`	178	13	3	**0.77**	1.22	1.4	1.01	0.28	**0.38**

Table 3: Comparison with other approaches in the literature.

a cost of 30%—5 minutes on average. After such a threshold, improvements in the score are not considered worth it for the computation cost. This is due to the increasing number of solutions to be evaluated during diversification, while relevant and diverse clusterings are already present in the dashboard.

4.3 Comparison

We compare AutoClues with state-of-the-art approaches in the literature against real datasets, considered as standard benchmarks. We selected the ones that provided either the performance on classification datasets (MOGA [5], MODE-cf [12]) or code for reproducibility (MALSS [15]). The former two approaches are evolutionary algorithms, and the latter is a general-purpose AutoML tool. These approaches do not provide a dashboard but only the best-performing clustering. Thus, for a fair comparison, we set $\alpha = 1$ to return the best clustering, and we adopt as relevance the same metric used in the competing approaches. MOGA and MODE-cf measure performance solely through the Davies–Bouldin Index (DBI [4], the less the better), MALLS also provides SIL (the higher the better).

Table 3 shows that AutoClues outperforms the reported approaches in 6 out of 8 datasets. According to DBI, MALSS achieves better performance on `seeds`, while MOGA on `wine`. As to SIL, MALSS is slightly more performant in `ecoli` and `seeds`. Yet, this is due to the fact that such approaches support the

computation of non-spherical clusterings while they are not currently included in AutoClues. Indeed, if we constraint MALSS to compute spherical clusters only, we observe a DBI value of 0.79 for `ecoli` while AutoClues achieves 0.46 (the less the better). As to SIL, MALLS achieves 0.4 and 0.45 for `ecoli` and `seeds` respectively; AutoClues outperforms on `ecoli` with $SIL = 0.46$ and confirms the previous result on `seeds` with $SIL = 0.37$.

5 Conclusion and Future Work

We introduced AutoClues, an end-to-end cluster analysis approach that leverages AutoML techniques to provide a diverse and relevant dashboard of clusterings. Our findings demonstrate that optimizing pre-processing significantly enhances performance, allowing AutoClues to overcome current state-of-the-art approaches. For future research, we plan to explore (i) meta-learning approaches to identify more effective pre-processing steps, (ii) integrating human feedback in the loop, and (iii) providing automatic explanations for the retrieved dashboard.

References

1. Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. Tech. rep., Stanford (2006)
2. Barlow, H.B.: Unsupervised learning. Neural computation **1**(3), 295–311 (1989)
3. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: Proc. of the 2000 ACM SIGMOD international conference on Management of data. pp. 93–104 (2000)
4. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-1**(2), 224–227 (1979)
5. Dutta, D., Dutta, P., Sil, J.: Simultaneous continuous feature selection and k clustering by multi objective genetic algorithm. In: 2013 3rd IEEE International Advance Computing Conference (IACC). pp. 937–942 (2013)
6. ElShawi, R., Sakr, S.: Tpe-autoclust: A tree-based pipline ensemble framework for automated clustering. In: 2022 IEEE International Conference on Data Mining Workshops (ICDMW). pp. 1144–1153 (2022)
7. Enes, J., Expósito, R.R., Fuentes, J., Cacheiro, J.L., Touriño, J.: A pipeline architecture for feature-based unsupervised clustering using multivariate time series from hpc jobs. Information Fusion **93**, 1–20 (2023)
8. Francia, M., Giovanelli, J., Pisano, G.: Hamlet: A framework for human-centered automl via structured argumentation. Future Generation Computer Systems **142**, 182–194 (2023)
9. Fränti, P., Sieranoja, S.: K-means properties on six clustering benchmark datasets (2018)
10. Gagolewski, M.: A framework for benchmarking clustering algorithms. SoftwareX **20**, 101270 (2022)
11. Giovanelli, J., Bilalli, B., Abelló, A.: Data pre-processing pipeline generation for autoetl. Information Systems **108**, 101957 (2022)
12. Hancer, E.: A new multi-objective differential evolution approach for simultaneous clustering and feature selection. Engineering Applications of Artificial Intelligence **87**, 103307 (2020)

13. Huang, J., Ng, M., Rong, H., Li, Z.: Automated variable weighting in k-means type clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(5), 657–668 (2005)
14. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: International conference on learning and intelligent optimization. pp. 507–523. Springer (2011)
15. Kamoshida, R., Ishikawa, F.: Automated clustering and knowledge acquisition support for beginners. Procedia Computer Science **176**, 1596–1605 (2020)
16. Lensen, A., Xue, B., Zhang, M.: Using particle swarm optimisation and the silhouette metric to estimate the number of clusters, select features, and perform clustering. In: European Conference on the Applications of Evolutionary Computation. pp. 538–554. Springer (2017)
17. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery from Data (TKDD) **6**(1), 1–39 (2012)
18. Liu, Y., Li, S., Tian, W.: Autocluster: Meta-learning based ensemble method for automated unsupervised clustering. In: 25th Pacific-Asia Conference, PAKDD 2021, May 11–14, Proc., Part III. p. 246–258. Springer-Verlag, Berlin, Heidelberg (2021)
19. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)
20. Murtagh, F., Contreras, P.: Algorithms for hierarchical clustering: an overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **7**(6) (2017)
21. Poulakis, Y., Doulkeridis, C., Kyriazis, D.: Autoclust: A framework for automated clustering based on cluster validity indices. In: ICDM. pp. 1220–1225. IEEE (2020)
22. Prakash, J., Singh, P.K.: Gravitational search algorithm and k-means for simultaneous feature selection and data clustering: a multi-objective approach. Soft Computing **23**(6), 2083–2100 (2019)
23. Saha, S., Spandana, R., Ekbal, A., Bandyopadhyay, S.: Simultaneous feature selection and symmetry based clustering using multiobjective framework. Appl. Soft Comput. **29**(C), 479–486 (apr 2015)
24. Sobol, I.: The distribution of points in a cube and the accurate evaluation of integrals (in russian) zh. Vychisl. Mat. i Mater. Phys **7**, 784–802 (1967)
25. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In: Proc. of the 19th ACM SIGKDD. pp. 847–855 (2013)
26. Thrun, M.C., Ultsch, A.: Clustering benchmark datasets exploiting the fundamental clustering problems. Data in brief **30**, 105501 (2020)
27. Toch, E., Lerner, B., Ben-Zion, E., Ben-Gal, I.: Analyzing large-scale human mobility data: a survey of machine learning methods and applications. Knowledge and Information Systems **58**(3), 501–523 (2019)
28. Tschechlov, D., Fritz, M., Schwarz, H.: Automl4clust: Efficient automl for clustering analyses pp. 343–348 (2021)
29. Vieira, M.R., Razente, H.L., Barioni, M.C., Hadjieleftheriou, M., Srivastava, D., Traina, C., Tsotras, V.J.: On query result diversification. In: 27th IEEE International Conference on Data Engineering (ICDE). pp. 1163–1174. IEEE (2011)
30. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: Proc. of the 26th annual international conference on machine learning. pp. 1073–1080 (2009)
31. Zhao, Z., Liu, H.: Spectral feature selection for supervised and unsupervised learning. In: Proc. of the 24th international conference on Machine learning (2007)
32. Zhu, L., Ma, B., Zhao, X.: Clustering validity analysis based on silhouette coefficient. Journal of Computer Applications **30**(2), 139–141 (2010)