Benchmarking AutoML Clustering Frameworks

Matheus Camilo[1] **Biagio Licari**[1] **Gabriel Marques Tavares**[2,3] **Sylvio Barbon Junior**[1]

[1]Department of Engineering and Architecture, University of Trieste, Italy
[2]LMU Munich, Germany
[3]Munich Center for Machine Learning (MCML), Munich, Germany

Abstract The surge of frameworks for automated unsupervised clustering problems exposed the notable gap in performance assessment, unified datasets and methodologies for this field. The lack of standardization and proper clustering goal setting obscures the applicability and suitability of such solutions. Therefore, we propose a benchmark to bridge this gap by offering a comparative analysis of AutoML frameworks for clustering, using several criteria and a comprehensive set of benchmarking problems. Four prominent AutoML unsupervised frameworks (AutoML4Clust, Autocluster, cSmartML, and ML2DAC) were compared following our methodology. By extending the evaluation beyond quantitative metrics, this research contributes to a more nuanced understanding of the applicability and performance of AutoML for a diverse set of clustering problems. Our analysis shows the evident demand for effort in the direction of pipeline synthesis (i.e., search and optimization of complete pipelines), clustering goal definition and suitable analysis dimensions.

1 Introduction

In the evolving landscape of data analysis, clustering serves as a foundational technique with widespread applications in tasks such as pattern recognition (Baraldi and Blonda, 1999), image segmentation (Mittal et al., 2021) and anomaly detection (Chandola et al., 2009). The inherent complexity and diversity of datasets demand a certain level of expertise and ability from the data practitioners who wish to produce effective clustering solutions. This prompts the integration of Automated Machine Learning (AutoML) methods into the clustering domain. The primary goal of AutoML is to make machine learning more accessible to individuals with varying levels of expertise by automating the complex and time-consuming aspects of machine learning development (Hutter et al., 2019).

The automated generation of unsupervised clustering solutions presents a formidable challenge. The datasets can often exhibit high-dimensionality, varying shapes, and sizes of clusters (Assent, 2012; Strehl and Ghosh, 2003), as well as the presence of noise and outliers (Ester et al., 1996). The dynamic nature of real-world data introduces a level of intricacy that demands sophisticated methodologies for discerning meaningful patterns. Since unsupervised clustering lacks clear targets, there is no single metric that describes every dataset equally well (Bonner, 1964; Ezugwu et al., 2022). Consequently, assessing the efficacy and performance of current AutoML frameworks designed for unsupervised learning remains an open issue, highlighting the need for further research and standardization efforts in this domain.

Furthermore, the proliferation of AutoML frameworks for unsupervised clustering presents a paradox to AutoML's fundamental objective of simplifying machine learning (Hutter et al., 2019). While these frameworks aim to streamline model development and deployment, their abundance introduces a counter-intuitive complexity, as each tool offers diverse capabilities, strengths, and limitations. Practitioners must possess not only a deep understanding of the clustering datasets they are working with but also comprehensive knowledge of the functionalities and performance of various AutoML frameworks. Thus, navigating this landscape demands a blend of domain expertise,

technical proficiency, and informed decision-making to leverage AutoML for unsupervised learning tasks effectively.

However, while supervised learning has garnered significant attention in the literature, the evaluation of AutoML techniques for unsupervised problems remains considerably understudied. To bridge this gap, we provide a benchmarking to evaluate AutoML clustering frameworks to assess their usability, clustering performance, and scalability. By examining usability aspects, such as installation process and documentation quality, we aim to provide insights into the ease of use and overall user experience of these tools. Evaluating clustering performance across diverse datasets and algorithms would identify the strengths and weaknesses of AutoML frameworks in terms of clustering quality and accuracy. Additionally, by analyzing scalability characteristics we assess their ability to handle large-scale datasets and efficiently utilize computational and time resources. Overall, this work offers valuable guidance for selecting the most suitable AutoML framework for clustering tasks, thereby advancing automated clustering techniques and enhancing accessibility to machine learning technology.

In the context of clustering, we selected four AutoML frameworks, namely: *ML2DAC* (Treder-Tschechlov et al., 2023a), *Autocluster* (Liu et al., 2021), *cSmartML* (ElShawi et al., 2021) and, finally, *AutoML4Clust* (Tschechlov et al., 2021). Our experiments showed that while usability aspects can vary between the frameworks, the clustering performance remains relatively consistent across most, with scalability favoring ML2DAC and AutoML4Clust. The experiments compared the AutoML clustering frameworks by leveraging a varied collection of 100 clustering datasets, ensuring comprehensive stress testing. The main contributions of this paper are:

- The first benchmarking of AutoML frameworks for clustering.
- Introduction of quantitative and qualitative criteria for AutoML frameworks for clustering.
- Development of a set of one hundred clustering datasets for comprehensive framework benchmarking.

2 Theoretical background

Algorithm Selection (AS), Hyperparameter Optimization (HPO), and Combined Algorithm Selection and Hyperparameter Optimization (CASH) are fundamental tasks in AutoML. Here, we provide formal definitions for each of these tasks.

AS involves selecting the optimal algorithm A^* from a set of candidate algorithms $\mathcal{A}$ to solve a given problem instance represented by a dataset D. Formally, AS aims to optimize (i.e., maximize) a performance function $\mathcal{F}(\cdot,\cdot)$:

$$A^* \in \arg\max_{A \in \mathcal{A}} \mathcal{F}(D, A) \tag{1}$$

where $\mathcal{F}(D, A)$ denotes the performance of using algorithm A to process the problem instance D.

HPO aims to find the optimal hyperparameters θ^* that optimize a given objective function $\mathcal{F}(\cdot,\cdot)$, considering a learning model M and a dataset D:

$$\theta^* \in \arg\max_{\theta \in \Theta} \mathcal{F}(M(D, \theta)) \tag{2}$$

where Θ denotes the hyperparameter space.

CASH integrates the tasks of AS and HPO. Given a set of candidate algorithms $\mathcal{A}$, and their associated hyperparameter spaces Θ_A, CASH aims to find both the optimal algorithm A^* and its corresponding hyperparameters θ^* that maximizes the objective function $\mathcal{F}(\cdot,\cdot,\cdot)$:

$$A^*, \theta^* \in \arg\max_{A \in \mathcal{A}, \theta \in \Theta_A} \mathcal{F}(D, A, \theta) \tag{3}$$

3 Related Work

Reviewing the literature, we considered as related the works that focused mainly on comparing AutoML methods and frameworks. Notable among these is Gijsbers et al. (2019), which introduces a benchmarking framework specifically tailored for AutoML. This benchmarking platform incorporates diverse evaluation criteria, such as predictive performance metrics, computational resource consumption, and ease of use, to evaluate four AutoML frameworks on 39 classification datasets. Another important work was presented by Balaji and Allen (2018), which evaluates AutoML frameworks in the context of regression in addition to classification. It compares the predictive performance of four AutoML frameworks on 30 regression datasets and 57 classification datasets.

Moreover, Zoller and Huber (2021) combined the benchmarking of popular AutoML frameworks with a very thorough survey of AutoML methods. The benchmarking evaluates the predictive performance of 6 AutoML frameworks in performing CASH on 73 classification datasets. The survey portion provides insights regarding features, capabilities, and applicability across different domains of AutoML methods for supervised machine learning.

Poulakis et al. (2024) presented a systematic review of AutoML methods for unsupervised learning. The survey includes recent research works in automated clustering and proposes a taxonomy for categorizing existing methods. By performing a qualitative comparison, the survey offers an overview of the field of AutoML for clustering, highlighting the current state-of-the-art and potential research directions. However, no quantitative comparison was provided, limiting the analysis landscape.

The lack of standardized benchmarking criteria and datasets for evaluating AutoML solutions in clustering is the primary motivation for this research. Establishing a set of qualitative and quantitative evaluation criteria, and clustering datasets is crucial to promote transparency and replicability in research findings and pave the way for future research.

4 Benchmarking Design

4.1 Datasets

To properly compare the performance of AutoML frameworks, it is important to assemble a set of datasets that represent a diverse range of clustering scenarios. In the context of clustering, this is not a well-defined or simple process. According to Von Luxburg et al. (2012), there are basically three types of datasets that are commonly used to evaluate clustering algorithms: (i) real world data, (ii) classification datasets and (iii) artificial data.

Using real world datasets to evaluate clustering algorithms may sound like a sensible approach for every case, although it comes with its own limitations. The main one is the lack of generalization, since it enables only the evaluation of algorithms considering their relevance to particular contexts or objectives (e.g., healthcare, finance, social networks).

Using classification datasets to evaluate clustering algorithms can be misleading since class labels in classification datasets are predefined categories, which may not align with the intrinsic groupings identified by clustering algorithms. This discrepancy can obscure the true performance of clustering algorithms and hinder the identification of meaningful clusters within the data (Fränti and Sieranoja, 2018).

The usage of synthetic data serves to assess the statistical performance of clustering algorithms within specified assumptions about the data generation process. However, it is essential to note that this method does not inherently gauge the practical *utility* of clustering, in the sense that the *usefulness* depends on specific domains or applications.

In our proposed benchmarking, we do not aim to set specific applications or clustering objectives to evaluate the performance of clustering algorithms, we only want to assess the statistical performance (see Sections 5.2 and 5.3). In this sense, labeled clustering synthetic data aligns better

with our objectives, as it allows the manipulation of desired parameters for study while mitigating the influence of other factors.

We synthesized[1] a set of 100 labeled clustering datasets, by sampling data from probabilistic mixture models based on dataset archetypes defined by the combinations of the features and ranges presented in Table 1. In a practical sense, an archetype is a high-level representation of the overall geometry of a dataset. Once an archetype is defined, it is straightforward to generate different datasets with similar structures. This characteristic makes these kinds of benchmarks more reproducible, since it is possible to replicate new data based on pre-defined archetypes (Zellinger and Bühlmann, 2023). The archetypes that were obtained by the combinations of the features from Table 1 are described in the Appendix 6 and displayed in Figure 4.

Table 1: Features and ranges of proposed benchmarking clustering datasets.

Feature	Range	Description
Dimensions	2 - 100	Refers to the number of dimensions of the dataset
Clusters	2 - 35	Refers to the number of clusters in the dataset
Samples	150 - 5000	Quantity of data points
Aspect Ref	1-10	Determines how elongated or spherical the clusters are
Aspect Max Min	1-10	Determines how much the Aspect Ref varies between clusters
Radius Max Min	1-10	Determines the volume variation in clusters
Imbalance Ratio	1 - 3	Refers to the disproportionate sizes between clusters

4.2 Selection Criteria

After reviewing the state-of-the-art on AutoML for clustering, we identified a set of 20 frameworks related to general clustering tasks, as presented in Table 2. It is possible to organize the current frameworks into four different tasks: AS, AS and HPO in sequence, CASH and Pipeline Synthesis (PS). PS automatically selects and configures the appropriate preprocessing techniques, feature engineering methods, clustering algorithms, and hyperparameters to build an optimal pipeline for a specific dataset (Karmaker et al., 2021).

Considering the limitations inherent in AS-only approaches for fully addressing clustering tasks (specially, the heavy impact of hyperparameters in clustering algorithms), we opted not to include frameworks solely dedicated to this task. Instead, we focused on frameworks capable of recommending a candidate solution, favoring a single tuned clustering algorithm over ranking or multiple solutions. This constraint narrowed down our selection of frameworks. Additionally, we prioritized frameworks with complete transparency and reproducibility support to ensure compliance with all evaluation criteria. In line with these principles, only four frameworks met our stringent criteria[2][3]: Autocluster, cSmartML, AutoML4Clust, and ML2DAC.

4.3 Evaluation Criteria

To conduct a thorough evaluation of the selected AutoML frameworks, we have established a comprehensive set of evaluation criteria. They are:

- **General comparison**. Although the selected frameworks share a common goal of simplifying and automating the clustering process, they have different approaches and strategies to achieve

[1]Datasets were generated using *Repliclust*. A Python package for generating synthetic clustering datasets, based on high-level geometric representations of whole classes of datasets.

[2]GitHub: `https://github.com/wywongbd/autocluster`, `https://github.com/DataSystemsGroupUT/csmartml`, `https://github.com/tschechlovdev/Automl4Clust`, `https://github.com/tschechlovdev/ml2dac`

[3]Python Package Index: `https://pypi.org/project/autocluster/`

Table 2: Summary of state-of-the-art AutoML frameworks for clustering grouped by AutoML task.

Task	Frameworks
AS	Ferrari and de Castro (2015), Pimentel and de Carvalho (2019), CBOvalue (Adam et al., 2015), de Souto et al. (2008), Pimentel and de Carvalho (2018), Nascimento et al. (2009), Ferrari and de Castro (2012), Soares et al. (2009), Fernandes et al. (2021), Pimentel and de Carvalho (2019), Gabbay et al. (2021), MARCO-GE (Cohen-Shapira and Rokach, 2021)
AS → HPO	AutoClust (Poulakis et al., 2020), **Autocluster** (Liu et al., 2021), **cSmartML** (ElShawi et al., 2021)
CASH	**AutoML4Clust** (Tschechlov et al., 2021), **ML2DAC** (Treder-Tschechlov et al., 2023b), COBS (Van Craenendonck and Blockeel, 2017), MASSCAH (Shalamov et al., 2018)
PS	TPE-AutoClust (ElShawi and Sakr, 2022)

this goal. The first criterion evaluates the design principles by comparing and contrasting: (i) optimization strategies (techniques to explore the search space); (ii) application of techniques to increase resource efficiency (e.g., parallelization, meta-learning, warm start); (iii) the degree of automation in the selection of algorithms, preprocessing techniques, and hyperparameters; Furthermore, we also evaluate the usability (Nichols and Twidale, 2002) and the diversity of clustering algorithms - search space composition - (Dongkuan Xu, 2015).

- **Clustering performance**. We employ specific Cluster Validity Indices (CVI) designed to measure the quality and similarity of clustering results (Vendramin et al., 2010). These metrics are fundamental in our evaluation and fall into two distinct categories: internal and external. Internal metrics assess how well the clusters are formed within the dataset, measuring attributes such as cohesion and separation. On the other hand, external metrics measure the clustering performance by comparing the results to ground truth labels.
- **Scalability**. We evaluated the scalability and consistency (Boden et al., 2017) of AutoML frameworks by testing their performance regarding memory consumption and on datasets of varying dimensions and instances.

4.4 Frameworks Configuration

All frameworks were executed in an isolated environment using the latest version of Anaconda (ana, 2020). Each framework was built using the exact packages and versions specified in their respective documentations. The experiments were conducted on a Dell Precision 5820 Tower X-Series, equipped with an Intel Core i9-10980XE CPU boasting 18 cores and 36 threads, reaching a maximum clock speed of 4.6 GHz. It featured 62.5 GiB of RAM, powered by two SK Hynix PC801 NVMe 1TB drives. The system was augmented with an NVIDIA Quadro T1000 Mobile GPU and operated on Ubuntu 22.04.2 LTS (Jammy Jellyfish).

5 Results[4]

5.1 General comparison

Autocluster automatically recommends a clustering algorithm using a distance-based strategy, considering meta-features such as simple, statistical, PCA-based, and landmarkers obtained from both a meta-database and the meta-features extracted from the new clustering problem. After having selected the closest candidates approaches (i.e., AS task), it conducts a grid-search for HPO

[4]The repository with the experiments that we conducted can be found `https://github.com/biagiolicari/Benchmark_result_AutoML`.

to enhance the performance of CVIs including the Calinski-Harabasz Index (Caliński and Harabasz, 1974), Davies-Bouldin Index (DBS) (Davies and Bouldin, 1979), and Silhouette Coefficient (SIL) (Rousseeuw, 1987). Each CVI contributes to a vote regarding the best score, employing a majority voting strategy to recommend the optimal algorithm. It is worth noting that this framework does not have preset hyperparameters, allowing flexibility in selecting different clustering algorithms. However, it requires a reliable meta-database capable of supporting the distance-based strategy. Additionally, the limitations of grid search could potentially lead to inefficient optimization.

AutoML4Clust approaches CASH by limiting its search space to only k-based clustering algorithms and the hyperparameter tuning to only the number of clusters (k). The framework optimizes internal CVIs through techniques such as Bayesian Optimization (BO), HyperBand optimization (Li et al., 2016), and the combination of HyperBand and BO (BOHB) (Falkner et al., 2018). It allows for multiple optimization tasks to be run concurrently during the optimization phase. However, this is only possible for the BOHB and HyperBand optimizators. The framework introduces a budget for time or iterations as a hyperparameter to limit the exploratory space during optimization. Unlike other frameworks, AutoML4Clust focuses solely on the recommendation phase. A proper selection of the budget is mandatory, as the optimization phase involves combinations of all algorithms and hyperparameters without optimized strategies.

The framework cSmartML capitalizes on past task experiences, maintaining a repository of dataset meta-features and evaluations to enhance its performance. Users provide datasets and time budget constraints, initiating the selection and tuning process. Using meta-features and eight clustering techniques, cSmartML assesses clustering quality using twelve internal indices and multiple objective functions. It ranks configurations based on performance correlations with external validation using Normalized Mutual Information and Spearman's rank correlation. Notably, cSmartML creates a set of hyper-partitions for a selected algorithm, which are then fine-tuned in parallel using an evolutionary algorithm. Unlike Autocluster and AutoML4Clust, cSmartML follows a tuning strategy based on two different levels of hyperparameter impact: main and conditional levels. While cSmartML introduces a strategy to reduce the HPO task, it operates in two different stages, AS then HPO, without taking advantage of mapping the problem as in traditional CASH.

ML2DAC is a framework that leverages a meta-knowledge repository built during an initial learning phase, drawing insights from various pre-clustered datasets and best CVI candidates for a particular task. This framework demands a very costly offline phase to build the meta-dataset, considering meta-feature extraction, diverse CVIs for further recommendation and additional meta-information. In the subsequent application phase, it identifies similar datasets from its repository, retrieves configurations for warm start, and selects the CVI using a trained model. Selecting warm start configurations and narrowing the search space reduce complexity and optimize performance using BO. By executing multiple configurations, the framework calculates the selected CVI values to identify the best-performing configuration. The offline phase is balanced by a lightweight online phase, employing suitable strategies to enhance recommendation performance. However, this framework does not support the dynamic increment of algorithms or CVIs, necessitating a reset of the offline phase from scratch. Table 3 presents an overview of the design strategy for performance-boosting methods and the proposed architecture of each framework.

Usability. We evaluated the usability of each framework by assessing their documentations and installation processes. The summary of our evaluation is presented in Table 4, where AutoML4Clust and cSmartML presented the least updated and usable frameworks mainly due to outdated installation processes or lack of comprehensive examples of usage and documentation. Autocluster and ML2DAC on the other hand equip users with the necessary resources to easily install and utilize their frameworks, by providing their frameworks as packages through Python's package-management system.

Search space. The adaptability of AutoML frameworks greatly influences clustering effectiveness. Our analysis of supported clustering algorithms (see Table 5) concluded that Autocluster

Table 3: Framework design strategies.

Framework	Performance boosting methods	Architecture
AutoML4Clust	Time or Iterations Budget	Online (optimization)
cSmartML	Time Budget, hyperpameters hierarchy tuning based on two levels	Offline (meta-features) and online (optimization)
Autocluster	-	Offline (meta-features) and online (optimization)
ML2DAC	Iterations Budget and warm start	Offline (meta-features and CVIs extensive calculation) and online (optimization)

Table 4: Summary of Documentation Evaluation among AutoML Frameworks.

Framework	Last Update	Examples	Documentation	Installation
AutoML4Clust	Oct 13, 2020	Repository provides usage examples, and detailed configurations.	Clear documentation with brief explanations.	Challenging installation process due to outdated packages and programming language.
cSmartML	Dec 27, 2021	Sparse examples in repository; GUI image available.	Lacks comprehensive documentation.	User-friendly interface, intuitive for interaction.
Autocluster	Jan 9, 2023	Repository showcases usage examples and customizations.	Detailed documentation and reports.	Easy installation and usage with provided examples.
ML2DAC	Sep 18, 2023	Repository contains usage demos and customizations.	Covers installation and result reproduction.	Simple installation and execution with provided demos.

and ML2DAC presented the greater search space range for AS. Autocluster and ML2DAC embrace diverse clustering algorithms, leaning towards density-based and affinity propagation models. In contrast, cSmartML prioritizes hierarchy models with some density-based and graph theory adoption. AutoML4Clust's search space is composed of only three partitioning-based clustering algorithms and a Gaussian Mixture Model (GMM) (Bishop, 2006).

Table 5: Clustering algorithms supported by each framework.

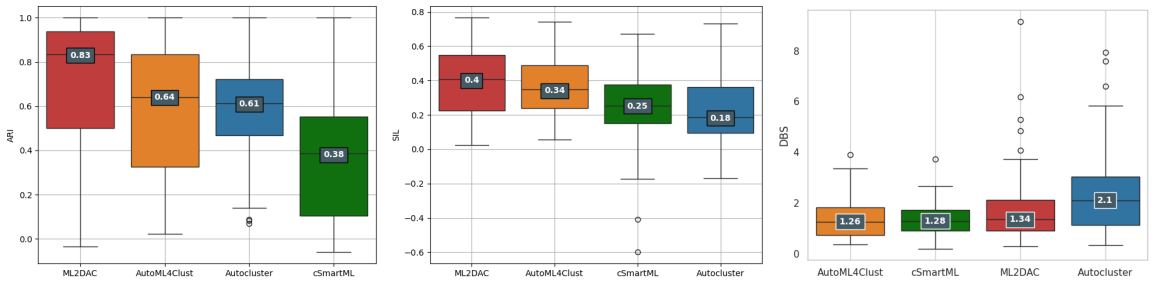
	AutoML4Clust	Autocluster	cSmartML	ML2DAC
Affinity Propagation	✗	✓	✓	✓
OPTICS	✗	✓	✓	✓
Agg. Clustering	✗	✓	✓	✓
MeanShift	✗	✓	✓	✓
GMM	✓	✓	✗	✓
Spectral Clustering	✗	✓	✓	✓
Mini Batch KMeans	✓	✓	✗	✓
K-medoids	✓	✗	✗	✗
DBSCAN	✗	✓	✓	✓
KMeans	✓	✓	✓	✓
Birch	✗	✓	✓	✓

5.2 Clustering performance

For the experiments, we instantiated all frameworks with their default configurations. We set a limit of 5 hours for the maximum execution time, after which the execution was considered failed. We also specified the amount of memory available as resource constraints and limited it to 16GiB. Additionally, we declared the user intent by choosing the optimization mode to maximize the quality of the result based on the internal CVIs[5] SIL and DBS, since they are common for every framework in the benchmark. Additionally, for the external CVI, we selected the Adjusted Random Index (ARI) (Hubert and Arabie, 1985), considering its ability to evaluate clustering solutions without assuming a priori knowledge, making it versatile and applicable to diverse datasets and algorithms (Santos and Embrechts, 2009).

The meta-CVI functionality of cSmartML and ML2DAC was utilized to obtain their results. The meta-learning capabilities of ML2DAC, cSmartML, and Autocluster were also used. We did not impose any additional constraints or customization to the evaluated frameworks' configuration parameters. This approach aimed to mirror the practical application of these systems closely. To conclude, we decided to use cSmartML's single optimization mode instead of its multi-objective optimization capability to match the other frameworks.

We ran each framework five times on the group of datasets, where we measured the performance regarding the internal and external CVIs, as displayed in Figure 1. Regarding ARI and SIL, ML2DAC produced the best results and AutoML4Clust the second best, while being closely followed by Autocluster in ARI and followed by cSmartML in SIL. The optimal DBS is 0. Therefore, AutoML4Clust obtained the best results, closely followed by cSmartML and ML2DAC, while Autocluster obtained the worst results.

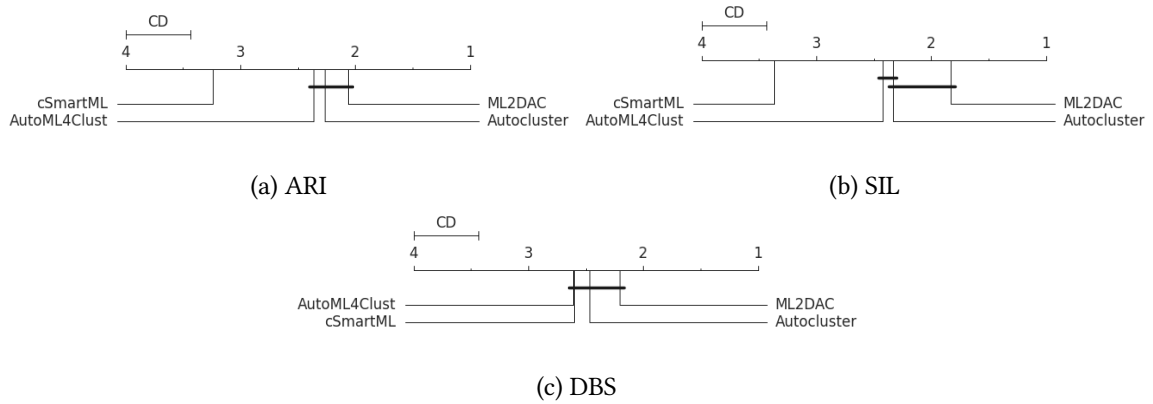


(a) ARI (b) SIL (c) DBS

Figure 1: Boxplot showing the performance of the selected frameworks, sorted from best to worse result based on the median for (a) ARI, (b) SIL, (c) DBS.

To truly assess the difference in cluster quality between the selected frameworks, we employed the non-parametric Friedman test at $p < 0.05$ to identify any statistically significant deviations in the average rank distributions. After conducting a statistical analysis, we use the Nemenyi post-hoc test to identify pairs that show significant differences. Lower ranks in the diagram indicate better performance. Figure 2 shows the Critical Distance (CD) diagrams (Demar, 2006) for each benchmarking metric. ML2DAC achieves better ranks across all settings, however, it is not statistically different than other frameworks in all analysis dimensions. Only for ARI and SIL we notice that the best three frameworks are statistically different than the worst framework. This analysis highlights that in practical scenarios, most frameworks have quite similar performance, and none of them can be considered optimal or statistically advantageous.

[5]The formalization for every CVI used can be found in the Appendix A.



(a) ARI

(b) SIL

(c) DBS

Figure 2: CD plot for respectively ARI, SIL, and DBS scores.

5.3 Scalability

Execution time. The results in Figure 3a indicate that AutoML4Clust outperforms other frameworks in terms of speed, with an average execution time of 1.11 minutes across all datasets and all executions. ML2DAC also performs well, with an average execution time of 2.27 minutes, although it is not as fast as AutoML4Clust. In comparison, Autocluster and cSmartML take much longer to execute, with times of 8.92 and 18.04 minutes, respectively. When comparing Autocluster and cSmartML, Autocluster is almost 2 times faster. Overall, while AutoML4Clust is the fastest, ML2DAC presents a balanced option for clustering tasks, combining speed and quality.

Memory usage. Analysing the memory usage across various AutoML frameworks for clustering has highlighted significant disparities that can impact framework selection. Autocluster is the framework that requires less memory usage, with an average memory consumption of 3.74 GB. AutoML4Clust requires an average of 4.72 GB. ML2DAC requires approximately 5.31 GB. Whereas cSmartML uses an average memory usage of approximately 5.68 GB. These variations underscore the importance of a comprehensive evaluation process that takes into account performance metrics and system resource limitations in selecting an AutoML framework.

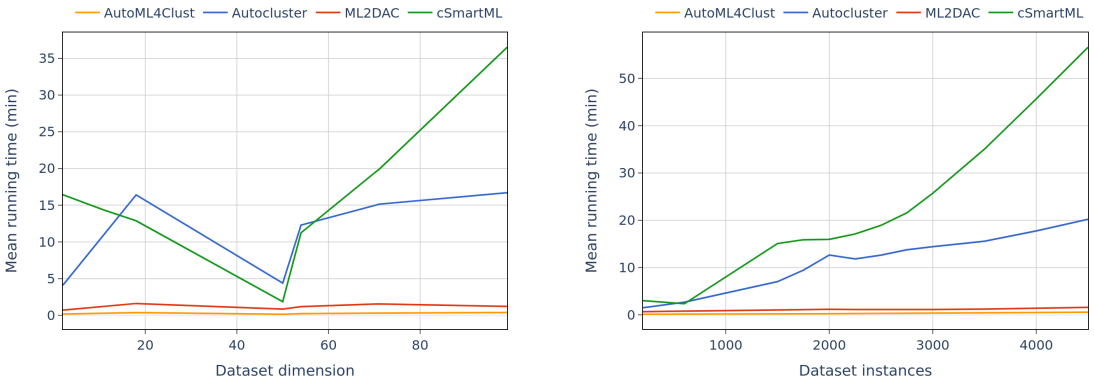
Scalability over dataset. Here, a comparison is made between the performance of frameworks concerning increasing dataset dimensions and instances. As indicated in Figure 3, AutoML4Clust exhibits almost constant execution time, together with ML2DAC, demonstrating stable performance. cSmartML, on the other hand, displays an exponential rise. cSmartML appears to be the most affected by larger datasets, indicating potential scalability challenges.

6 Discussions

In this study, we conducted experiments to evaluate three main criteria of four AutoML frameworks designed for clustering.

The first criterion involved qualitative general comparisons assessing the different design principles of each framework and their ease of use, aswell as usability and search space composition. According to the criteria that we proposed to evaluate this criterion, ML2DAC obtained the superior performance by balancing offline model training with meta-learning and warm start to produce high quality and resource-efficient optimizations, on a diverse search space. Furthermore it provides users with reproducible results, and easy installation process and documentation that helps lower the learning-curve.

Moving on to the second criterion, we evaluated quantitative clustering performance. The experiments revealed that the frameworks generally obtained statistically similar results, except



(a) Scalability over dataset dimensions.

(b) Scalability over dataset instances.

Figure 3: Average performance of frameworks over (a) dataset dimension and (b) dataset instances.

for cSmartML, which exhibited statistically lower ARI and SIL scores. This underscores that in practical scenarios, none of the frameworks can be deemed optimal or statistically advantageous.

In terms of the third criterion, scalability performance, the experiments demonstrated that both CASH frameworks, ML2DAC and AutoML4Clust, consistently delivered faster results across datasets of different sizes compared to Autocluster and cSmartML. Moreover, the most effective techniques capitalized on BO, distinguishing them from the evolutionary optimization employed by cSmartML and the grid search utilized by Autocluster. However, it is important to mention that AutoML4Clust presents a limited algorithm space compared to ML2DAC, as detailed in Table 5. The improved results of ML2DAC are attributed to the warm start strategy.

It is important to note that there may be limitations in the design and execution of benchmarks for AutoML frameworks. These frameworks differ in their design choices, including code libraries, search space, and optimization techniques, which can affect their performance. It is also worth noting that all the experiments were conducted within a time budget, which can limit insights regarding the scalability and performance of frameworks. Additionally, it is important to note that only the results produced by the final model of each framework are recorded.

7 Conclusion

In conclusion, our novel benchmarking analysis sheds light on the strengths and limitations of AutoML frameworks for clustering tasks. While usability and search space vary among the frameworks, clustering performance remains relatively consistent across most, with scalability favoring CASH approaches. These findings provide valuable insights for researchers and practitioners in selecting the most suitable AutoML framework for clustering tasks, emphasizing the importance of considering both qualitative and quantitative aspects in framework evaluation and selection. Additionally, our contribution to developing a diverse set of clustering datasets for benchmarking AutoML frameworks holds significant value in the field. This collection, meticulously curated to encompass a wide spectrum of data distributions, dimensions, and complexities, enables us the conduct comprehensive evaluations of AutoML frameworks across diverse scenarios.

8 Broader Impact Statement

After careful reflection, the authors have determined that this work presents no notable negative impacts to society or the environment.

References

(2020). Anaconda software distribution.

Adam, A., Blockeel, H., Vanschoren, J., Brazdil, P., Giraud-Carrier, C., and Kotthoff, L. (2015). Dealing with overlapping clustering: A constraint-based approach to algorithm selection.

Assent, I. (2012). Clustering high dimensional data. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(4):340–350.

Balaji, A. and Allen, A. (2018). Benchmarking automatic machine learning frameworks. *ArXiv*, abs/1808.06492.

Baraldi, A. and Blonda, P. (1999). A survey of fuzzy clustering algorithms for pattern recognition. i. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(6):778–785.

Bishop, C. M. (2006). Pattern recognition and machine learning. *Springer google schola*, 2:645–678.

Boden, C., Spina, A., Rabl, T., and Markl, V. (2017). Benchmarking data flow systems for scalable machine learning. *Proceedings of the 4th ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond*.

Bonner, R. E. (1964). On some clustering techniques. *IBM journal of research and development*, 8(1):22–32.

Caliński, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58.

Cohen-Shapira, N. and Rokach, L. (2021). Automatic selection of clustering algorithms using supervised graph embedding. *Information Sciences*, 577:824–851.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227.

de Souto, M. C. P., Prudencio, R. B. C., Soares, R. G. F., de Araujo, D. S. A., Costa, I. G., Ludermir, T. B., and Schliep, A. (2008). Ranking and selecting clustering algorithms using a meta-learning approach. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 3729–3735.

Demar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

Dongkuan Xu, Y. T. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*.

ElShawi, R., Lekunze, H., and Sakr, S. (2021). csmartml: A meta learning-based framework for automated selection and hyperparameter tuning for clustering. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 1119–1126.

ElShawi, R. and Sakr, S. (2022). Tpe-autoclust: A tree-based pipline ensemble framework for automated clustering. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1144–1153.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Ezugwu, A. E., Ikotun, A. M., Oyelade, O. O., Abualigah, L., Agushaka, J. O., Eke, C. I., and Akinyelu, A. A. (2022). A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743.

Falkner, S., Klein, A., and Hutter, F. (2018). Bohb: Robust and efficient hyperparameter optimization at scale.

Fernandes, L. H. d. S., Lorena, A. C., and Smith-Miles, K. (2021). Towards understanding clustering problems and algorithms: An instance space analysis. *Algorithms*, 14(3).

Ferrari, D. G. and de Castro, L. N. (2012). Clustering algorithm recommendation: A meta-learning approach. In Panigrahi, B. K., Das, S., Suganthan, P. N., and Nanda, P. K., editors, *Swarm, Evolutionary, and Memetic Computing*, pages 143–150, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ferrari, D. G. and de Castro, L. N. (2015). Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods. *Information Sciences*, 301:181–194.

Fränti, P. and Sieranoja, S. (2018). K-means properties on six clustering benchmark datasets. *Applied intelligence*, 48:4743–4759.

Gabbay, I., Shapira, B., and Rokach, L. (2021). Isolation forests and landmarking-based representations for clustering algorithm recommendation using meta-learning. *Information Sciences*, 574:473–489.

Gijsbers, P., LeDell, E., Thomas, J., Poirier, S., Bischl, B., and Vanschoren, J. (2019). An open source automl benchmark. *arXiv preprint arXiv:1907.00909*.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2:193–218.

Hutter, F., Kotthoff, L., and Vanschoren, J. (2019). *Automated machine learning: methods, systems, challenges*. Springer Nature.

Karmaker, S. K., Hassan, M. M., Smith, M. J., Xu, L., Zhai, C., and Veeramachaneni, K. (2021). Automl to date and beyond: Challenges and opportunities. *ACM Computing Surveys (CSUR)*, 54(8):1–36.

Li, L., Jamieson, K. G., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2016). Efficient hyperparameter optimization and infinitely many armed bandits. *CoRR*, abs/1603.06560.

Liu, Y., Li, S., and Tian, W. (2021). Autocluster: Meta-learning based ensemble method for automated unsupervised clustering. In Karlapalem, K., Cheng, H., Ramakrishnan, N., Agrawal, R. K., Reddy, P. K., Srivastava, J., and Chakraborty, T., editors, *Advances in Knowledge Discovery and Data Mining*, pages 246–258, Cham. Springer International Publishing.

Mittal, H., Pandey, A. C., Saraswat, M., Kumar, S., Pal, R., and Modwel, G. (2021). A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets. *Multimedia Tools and Applications*, pages 1–26.

Nascimento, A. C. A., Prudêncio, R. B. C., de Souto, M. C. P., and Costa, I. G. (2009). Mining rules for the automatic selection process of clustering methods applied to cancer gene expression data. In Alippi, C., Polycarpou, M., Panayiotou, C., and Ellinas, G., editors, *Artificial Neural Networks – ICANN 2009*, pages 20–29, Berlin, Heidelberg. Springer Berlin Heidelberg.

Nichols, D. M. and Twidale, M. B. (2002). Usability and open source software.

Pimentel, B. A. and de Carvalho, A. C. (2019). A new data characterization for selecting clustering algorithms using meta-learning. *Information Sciences*, 477:203–219.

Pimentel, B. A. and de Carvalho, A. C. P. L. F. (2018). Statistical versus distance-based meta-features for clustering algorithm recommendation using meta-learning. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Pimentel, B. A. and de Carvalho, A. C. P. L. F. (2019). Unsupervised meta-learning for clustering algorithm recommendation. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Poulakis, Y., Doulkeridis, C., and Kyriazis, D. (2020). Autoclust: A framework for automated clustering based on cluster validity indices. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1220–1225.

Poulakis, Y., Doulkeridis, C., and Kyriazis, D. (2024). A survey on automl methods and systems for clustering. *ACM Transactions on Knowledge Discovery from Data*.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Santos, J. M. and Embrechts, M. (2009). On the use of the adjusted rand index as a metric for evaluating supervised classification. In *International conference on artificial neural networks*, pages 175–184. Springer.

Shalamov, V., Efimova, V., Muravyov, S., and Filchenkov, A. (2018). Reinforcement-based method for simultaneous clustering algorithm selection and its hyperparameters optimization. *Procedia Computer Science*, 136:144–153. 7th International Young Scientists Conference on Computational Science, YSC2018, 02-06 July2018, Heraklion, Greece.

Soares, R. G. F., Ludermir, T. B., and De Carvalho, F. A. T. (2009). An analysis of meta-learning techniques for ranking clustering algorithms applied to artificial data. In Alippi, C., Polycarpou, M., Panayiotou, C., and Ellinas, G., editors, *Artificial Neural Networks – ICANN 2009*, pages 131–140, Berlin, Heidelberg. Springer Berlin Heidelberg.

Strehl, A. and Ghosh, J. (2003). Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 15(2):208–230.

Treder-Tschechlov, D., Fritz, M., Schwarz, H., and Mitschang, B. (2023a). Ml2dac: Meta-learning to democratize automl for clustering analysis. *Proceedings of the ACM on Management of Data*, 1(2):1–26.

Treder-Tschechlov, D., Fritz, M., Schwarz, H., and Mitschang, B. (2023b). Ml2dac: Meta-learning to democratize automl for clustering analysis. *Proc. ACM Manag. Data*, 1(2).

Tschechlov, D., Fritz, M., and Schwarz, H. (2021). Automl4clust: Efficient automl for clustering analyses.

Van Craenendonck, T. and Blockeel, H. (2017). Constraint-based clustering selection. *Machine Learning*, 106(9):1497–1521.

Vendramin, L., Campello, R. J. G. B., and Hruschka, E. R. (2010). Relative clustering validity criteria: A comparative overview. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 3.

Von Luxburg, U., Williamson, R. C., and Guyon, I. (2012). Clustering: Science or art? In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 65–79. JMLR Workshop and Conference Proceedings.

Zellinger, M. J. and Bühlmann, P. (2023). repliclust: Synthetic data for cluster analysis.

Zoller, M.-A. and Huber, M. F. (2021). Benchmark and survey of automated machine learning frameworks. *Journal of artificial intelligence research*, 70:409–472.

Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

   (b) Did you describe the limitations of your work? [Yes]

   (c) Did you discuss any potential negative societal impacts of your work? [Yes]

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? `https://2022.automl.cc/ethics-accessibility/` [Yes]

2. If you ran experiments...

   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources)? [Yes]

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? [Yes]

   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [Yes]

   (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? [Yes]

   (e) Did you report the statistical significance of your results? [Yes]

   (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? [Yes]. Tabular

   (g) Did you compare performance over time and describe how you selected the maximum duration? [Yes]

   (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

   (i) Did you run ablation studies to assess the impact of different components of your approach? [No]

3. With respect to the code used to obtain your results...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes]

   (b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [No]

   (c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes]

   (d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes]

   (e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes]

4. If you used existing assets (e.g., code, data, models)...

(a) Did you cite the creators of used assets? [Yes]

(b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [No]

(c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]

5. If you created/released new assets (e.g., code, data, models)...

   (a) Did you mention the license of the new assets (e.g., as part of your code submission)? [No]

   (b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [Yes]

6. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [Yes]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [No]

7. If you included theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [Yes]

   (b) Did you include complete proofs of all theoretical results? [No]

A CVIs Formalization

A.1 Silhouette

The SIL score, as introduced by Rousseeuw (1987), ranges from -1 to 1, where a higher score indicates better-defined clusters. A score close to 1 suggests that the data point is well-matched to its own cluster and poorly matched to neighboring clusters, signifying a strong and appropriate clustering. Conversely, a score near -1 implies that the point may be assigned to the wrong cluster, while a score around 0 indicates overlapping clusters. Let the i, j be n-dimensional feature vectors (data points) and $i, j \in C_I$; C as a given cluster of the dataset D; $d(i, j)$ as the average Euclidian distance between data points i and j; $|C_i|$ as the number of data points in cluster C_I and $|N|$ is the total number of data points present in the dataset D. The Silhouete Score for i can be calculated using (4) if and only if $|C_I| > 1$.

$$\mathcal{SIL}(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{4}$$

Where, $a(i)$ is the average distance from the i-th data point to the other data points in the same cluster, given by (5).

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, j \neq i} d(i, j) \tag{5}$$

And, $b(i)$ is the smallest average distance between the i-th data point of C_I and the k-th data point of C_K, where $C_I \neq C_K$, given by (6).

$$b(i) = \min_{K \neq I} \frac{1}{|C_K|} \sum_{k \in C_K} d(i, k) \tag{6}$$

Finally, the overall SIL score for the entire dataset D is the average of the SIL scores for all data points, as given by (7):

$$\mathcal{SIL}_D = \frac{1}{|N|} \sum_{i=1}^{N} S(i) \tag{7}$$

A.2 Davies-Bouldin Score

The DBS, as introduced by Davies and Bouldin (1979), is defined as the ratio of compactness within clusters and the separation between clusters. A DBS closer to 0 indicates better clustering, with well-defined and separated clusters. Let C_I be a cluster of a dataset D; σ the average Euclidian distance from each data point of C_I to its centroid A_I; $d(C_I, C_K)$ be the Euclidian distance between the A_I and A_K; and the data-point $i \in C_I$; $|C|$ be the total number of clusters in D; and $|C_I|$ be the total number of datapoints in C_I.

Where, the compactness of C_I is given by (8):

$$\sigma_I = \frac{1}{|C_I|} \sum_{i=1}^{|C_I|} ||i - A_I|| \tag{8}$$

The distance between the centroids is given by (9):

$$d(C_I, C_K) = ||A_I - A_K|| \tag{9}$$

And, the DBS for the entire dataset D is calculated using the equation (10):

$$\mathcal{DBS}_D = \frac{1}{|C|} \sum_{I=1}^{|C|} \max_{K \neq I} \left(\frac{\sigma_I + \sigma_K}{d(C_I, C_K)} \right) \tag{10}$$

A.3 Adjusted Random Index

Hubert and Arabie (1985) The Rand Index (RI) is a measure of similarity between two clusterings. It is particularly useful in evaluating the performance of clustering algorithms when the ground truth is known. RI considers pairs of samples and classifies them as either concordant or discordant based on whether they are placed in the same or different clusters in both clusterings. It ranges from 0 to 1, where 1 indicates perfect agreement between the true and predicted clusterings, and 0 indicates no agreement beyond that expected by chance. Let n be the number of elements in the set $S = \{o_1, ..., o_n\}$, where $U = \{u_1, ..., u_R\}$ and $V = \{v_1, ..., v_C\}$ represent two different partitions of S into subsets R and CS;

The Rand Index is calculated using the formula (11):

$$RI = \frac{\alpha + \beta}{\binom{n}{2}} \tag{11}$$

For $1 \leq i, j \leq n, i \neq j, 1 \leq k, k_1, k_2 \leq r, k_1 \neq k_2, 1 \leq l, l_1, l_2 \leq s, l_1 \neq l_2$,:

- $\alpha = |S^*|$, where $S^* = \{(o_i, o_j) \mid o_i, o_j \in U_k; o_i, o_j \in V_l\}$
- $\beta = |S^*|$, where $S^* = \{(o_i, o_j) \mid o_i \in U_{k_1}; o_j \in U_{k_2}; o_i \in V_{l_1}; o_j \in V_{l_2}\}$

The ARI is a essentially an expansion of of RI. The ARI accounts for chance agreement, providing a normalized measure that ranges from -1 to 1. A score of 1 indicates perfect agreement, 0 suggests agreement expected by chance, and negative values imply worse-than-chance agreement. The formula for the ARI is given by:

$$\mathcal{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - [\sum_i \binom{\alpha_i}{2} \sum_j \binom{\beta_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{\alpha_i}{2} + \sum_j \binom{\beta_j}{2}] - [\sum_i \binom{\alpha_i}{2} \sum_j \binom{\beta_j}{2}]/\binom{n}{2}} \tag{12}$$

This formula normalizes the Rand Index by accounting for random chance, making the ARI a valuable tool for assessing clustering accuracy in various fields, including biology, image analysis, and social sciences.

B Synthetic datasets archetypes

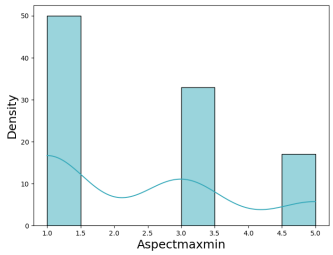
Archetype	Dim	Clusters	Instances	Aspectref	Aspectmaxmin	Radius	Imbalance
Archetype 0	53	20	350	1.5	1	3	2
Archetype 1	10	3	350	3.0	1	3	2
Archetype 2	11	12	400	3.0	3	3	2
Archetype 3	53	4	3000	1.5	1	5	1
Archetype 4	10	2	3500	1.5	3	1	2
Archetype 5	11	10	2500	5.0	1	1	2
Archetype 6	10	2	400	1.5	1	1	1
Archetype 7	5	2	2750	5.0	5	3	1
Archetype 8	10	2	1500	1.5	1	5	1
Archetype 9	99	15	450	3.0	3	1	2
Archetype 10	50	21	500	3.0	1	3	2
Archetype 11	52	2	200	3.0	3	5	1
Archetype 12	54	13	1750	1.5	3	1	1

Continued on the next page

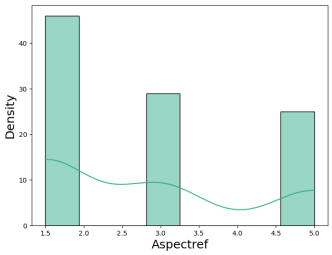
Archetype	Dim	Clusters	Instances	Aspectref	Aspectmaxmin	Radius	Imbalance
Archetype 13	53	11	1500	3.0	1	5	1
Archetype 14	2	4	2750	5.0	1	5	2
Archetype 15	10	2	2000	5.0	1	1	1
Archetype 16	51	18	200	3.0	3	5	1
Archetype 17	52	20	450	1.5	3	5	2
Archetype 18	18	4	2250	1.5	5	1	1
Archetype 19	10	4	400	5.0	5	1	2
Archetype 20	18	4	2500	1.5	1	5	1
Archetype 21	53	15	2000	1.5	1	3	1
Archetype 22	18	4	4000	1.5	3	3	2
Archetype 23	2	14	1500	1.5	1	3	2
Archetype 24	3	3	200	3.0	1	5	1
Archetype 25	5	2	4500	1.5	5	5	1
Archetype 26	71	5	2250	1.5	1	1	2
Archetype 27	54	18	2250	3.0	1	5	1
Archetype 28	70	19	3500	3.0	1	5	1
Archetype 29	71	6	2500	3.0	1	3	2
Archetype 30	70	21	4000	1.5	1	1	1
Archetype 31	18	4	4500	1.5	1	3	2
Archetype 32	2	4	3500	5.0	5	1	1
Archetype 33	18	4	500	5.0	5	5	1
Archetype 34	53	13	4500	3.0	3	1	2
Archetype 35	3	3	2000	1.5	5	3	1
Archetype 36	2	15	3500	1.5	1	5	2
Archetype 37	2	16	600	5.0	1	3	2
Archetype 38	5	3	200	5.0	1	3	1
Archetype 39	3	4	300	3.0	3	5	1
Archetype 40	2	14	3500	1.5	3	1	2
Archetype 41	5	2	4000	1.5	1	3	1
Archetype 42	18	4	2000	5.0	1	5	1
Archetype 43	71	19	600	3.0	3	1	1
Archetype 44	54	8	200	1.5	3	3	2
Archetype 45	54	2	200	1.5	3	3	2
Archetype 46	11	10	3500	1.5	5	3	2
Archetype 47	2	6	3000	5.0	5	1	2
Archetype 48	18	5	2500	1.5	1	1	1
Archetype 49	5	3	4000	1.5	1	3	2
Archetype 50	99	18	1750	1.5	3	3	1
Archetype 51	52	7	300	3.0	1	5	2
Archetype 52	11	13	2500	5.0	1	3	1
Archetype 53	11	12	300	1.5	1	1	1
Archetype 54	99	10	200	3.0	3	3	2
Archetype 55	11	12	2500	5.0	5	3	1
Archetype 56	54	18	3000	3.0	3	5	1
Archetype 57	53	19	4500	3.0	3	5	1
Archetype 58	52	19	450	1.5	1	5	2

Continued on the next page

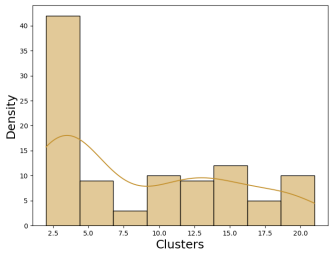
Archetype	Dim	Clusters	Instances	Aspectref	Aspectmaxmin	Radius	Imbalance
Archetype 59	2	3	4000	5.0	5	5	1
Archetype 60	54	18	3000	3.0	3	3	2
Archetype 61	54	11	1750	1.5	3	1	1
Archetype 62	2	5	4500	1.5	1	5	2
Archetype 63	10	2	4500	1.5	3	3	1
Archetype 64	2	4	550	1.5	1	3	1
Archetype 65	10	2	4500	5.0	1	1	1
Archetype 66	2	14	2500	5.0	5	3	1
Archetype 67	3	3	1750	3.0	1	5	1
Archetype 68	11	11	2000	3.0	3	3	1
Archetype 69	3	3	3000	5.0	1	3	2
Archetype 70	11	12	200	1.5	1	1	1
Archetype 71	10	3	3500	5.0	1	5	2
Archetype 72	71	11	2250	3.0	3	1	1
Archetype 73	2	15	200	3.0	3	5	1
Archetype 74	3	3	1750	1.5	5	3	2
Archetype 75	2	3	2000	5.0	1	1	1
Archetype 76	18	5	200	1.5	3	5	1
Archetype 77	51	15	2000	1.5	3	3	2
Archetype 78	99	10	3500	3.0	3	3	2
Archetype 79	11	11	2500	1.5	3	3	2
Archetype 80	3	4	350	5.0	1	3	1
Archetype 81	71	7	2500	1.5	3	1	1
Archetype 82	5	3	350	1.5	5	3	1
Archetype 83	70	13	4500	3.0	1	5	2
Archetype 84	3	3	300	5.0	5	5	1
Archetype 85	5	2	3000	5.0	5	3	2
Archetype 86	71	2	2000	1.5	3	5	1
Archetype 87	70	15	400	3.0	3	3	1
Archetype 88	11	10	200	5.0	1	3	1
Archetype 89	18	5	1500	5.0	5	5	2
Archetype 90	5	3	1500	1.5	1	1	2
Archetype 91	10	4	450	3.0	1	3	1
Archetype 92	18	5	550	1.5	1	5	1
Archetype 93	51	20	400	1.5	1	1	2
Archetype 94	2	6	1750	1.5	1	5	2
Archetype 95	5	2	3000	3.0	1	3	2
Archetype 96	52	20	2250	1.5	3	1	1
Archetype 97	2	16	1750	3.0	3	3	2
Archetype 98	2	14	4000	1.5	1	5	2
Archetype 99	11	13	3500	5.0	1	1	1



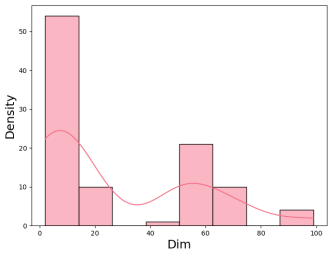
(a) Min and Max Aspect Ratio



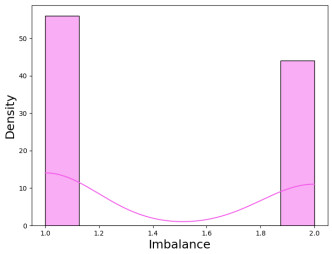
(b) Aspect Ratio



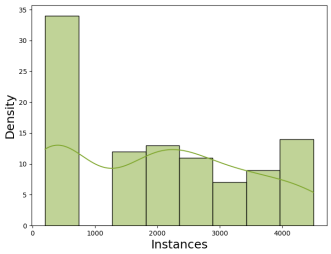
(c) Number of Clusters



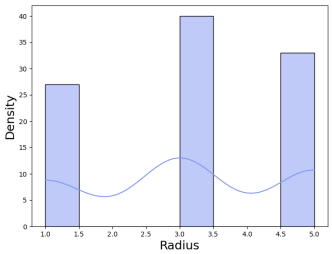
(d) Number of Dimensions



(e) Imbalance Ratio



(f) Number of Instances



(g) Radius

Figure 4: Feature distribution of the 100 clustering datasets used for evaluating AutoML frameworks.