Journal of Theoretical and Applied Information Technology
31st May 2025. Vol.103. No.10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SOL-AUTOCLUST: A SMART ONLINE-LEARNING AUTOMATED CLUSTERING FRAMEWORK

IBRAHIM GOMAA [1] , HODA M. O. MOKHTAR [2] , NEAMAT EL-TAZI [3], ALI ZIDANE [4]

[1,2, 3, 4] Faculty of Computers and Artificial Intelligence, Cairo University, Cairo, Egypt.

[2]Faculty of Computing and Information Sciences, Egypt University of Informatics, Cairo, Egypt.

E-mail: [1]i.gomaa@fci-cu.edu.eg, [2]h.mokhtar@fci-cu.edu.eg, [3]n.eltazi@fci-cu.edu.eg, [4]a.zidane@fci-cu.edu.eg

ABSTRACT

The automation of machine learning has predominantly focused on supervised tasks, leaving unsupervised clustering, a critical component of exploratory data analysis, significantly underdeveloped by existing Auto-ML frameworks. Current approaches often limit their scope to dataset characteristics, neglecting the crucial influence of algorithmic suitability (e.g., robustness to outliers) and user-defined requirements (e.g., interpretability needs). This oversight leads to suboptimal clustering outcomes, particularly when dealing with complex, high-dimensional, or noisy data. To address these limitations, this research introduces SOL-Auto-Clust, a novel end-to-end automated clustering framework that makes a key contribution by holistically integrating three fundamental dimensions: inherent data characteristics, intrinsic algorithmic traits, and explicit user-defined objectives. By employing a meta-feature architecture, SOL-Auto-Clust dynamically generates customized clustering pipelines, addressing both data intricacies and real-world application requirements. Extensive evaluation across diverse datasets highlights the framework's ability to simplify clustering processes and produce reliable, insightful outcomes, marking a significant step towards human-aligned Auto-ML for unsupervised learning.

Keywords: *Automated Machine Learning (Auto-ML), Automated Clustering, Unsupervised Learning, CASH*

1. INTRODUCTION

Machine learning (ML) and artificial intelligence (AI) have attracted more attention from both the business and research communities. Both ML and AI can be adopted in various domains to provide fast and efficient results compared to traditional programming. They can solve complex problems and extract hidden patterns from the given data. In general, the machine learning process depends on the volume of the training data. The more data used for model training, the more accurate result you get. Therefore, data is the most important part of the process of building a machine learning model. Recently, data volumes have increased exponentially because of the spread of the internet, social media, devices, data sources, and different kinds of applications. Consequently, artificial intelligence and machine learning have been broadly adapted in various fields, such as image classification [1-3] text classification [4], clustering analysis [5-9] speech recognition [10], predictive analytics [11-15] and recommendation systems [17, 18].

Clustering is considered one of the most used machine learning tasks, along with unsupervised learning, that involves the process of partitioning unlabeled data or data points into distinct clusters according to the similarities of data characteristics and features [20]. The aim behind clustering is to uncover concealed patterns or relationships based on shared attributes. Clustering primary purpose is to extract meaningful insights from large datasets and facilitate data organization. By organizing the data, it becomes easier to analyze and comprehend the underlying patterns present within the dataset. In addition, clustering is widely used in various applications and domains, such as healthcare [21-23], customer segmentation [26-28], finance [29-31], education [32, 33], campaign analysis, and marketing [24, 25]. Generally, there are many challenges facing data scientists when building clustering models. These challenges include a wide range of possible clustering algorithms that data scientists need to select from, such as K-means, Agglomerative Clustering, DBScan, and Optic. In addition, data scientists need to tune a set of hyper-parameters for the selected clustering algorithm. Moreover, they need to select the appropriate

measure to evaluate the model's performance. Furthermore, the clustering process depends on specifying the exact number of clusters for each dataset to be clustered. However, when dealing with real-world datasets, it is not expected to have this knowledge. Therefore, several studies have been proposed to reduce the manual development of ML processes and employ the automation concept instead.

In general, manual development of the machine learning process is both resource- and time-intensive and often depends on the specialization level of the data scientists and machine learning engineers. Thus, eliminating human interference from the ML process loop and activating the concepts of "Human Not in the Loop" and Auto-ML have attracted more attention. Human Not in the Loop and Auto-ML encourage finding a way to automate the process of building machine learning pipelines and reducing human intervention. Moreover, Auto-ML can help in providing high-performance machine learning solutions for a given problem within an acceptable time limit.

Although the topic of auto-ML has attracted more attention to automating the workflow of building machine learning models with many systems, unsupervised learning and particularly automated clustering, are much less addressed. Automating the clustering process is challenging because of the subjectivity of evaluating clustering quality due to a lack of ground truth, and clustering is very much a human construct. In addition, most existing auto-ML frameworks that automate the clustering process depend on data characteristics without considering users' needs or clustering algorithm characteristics. However, depending only on data characteristics cannot guarantee selecting the optimal clustering algorithms. For example, assume we have a new dataset (d_n) to be clustered, and the two most similar datasets to d_n are d_1 and d_2, as shown in figure 1. After identifying the similarity between d_n and d_1, d_2, it is found that d_n is more similar to d_1, and consequently, the recommended clustering algorithm to be used in clustering d_n is the k-means. In this example, the recommendation process depended only on measuring the similarity between datasets without validating whether the recommended algorithm suited the characteristics of the dataset (d_n) or not. The accuracy of the recommendation process may be compromised because it relies solely on the characteristics of the data without considering the specific traits of the clustering algorithms. This oversight becomes evident in the given example, where the recommended algorithm is k-means, which is not well-suited for datasets that contain outliers. Moreover, some algorithms ignore the outliers, which may be real data points that represent special cases for users. For example, if we have a dataset for fraudulent and nonfraudulent transactions, some clustering algorithms that can't deal with outliers will consider fraudulent transactions as outlier points and ignore them as they are a minority and don't have the same behavior as the majority of the population. All of these hypotheses and assumptions must be taken into consideration during the clustering algorithm selection process.

Dataset	No. of instances	No. of features	No. of missing values	No. of categorical features	No. of numeric features	No. of binary features	Outliers' flag	Clustering algorithm
d_1	200	10	0	0	5	5	0	K-means
d_2	300	30	0	5	15	10	1	Birch
d_n	200	9	0	0	5	4	1	?

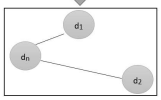

Figure 1: An Illustrative Example of Similarities Between Datasets

In this paper, we address the challenges and limitations of the existing auto-clustering frameworks and how to overcome them. "SOL-AutoClust" is presented as a new efficient online auto-clustering framework that focuses on the complete pipeline of the clustering process instead of focusing only on automating the process of Combined Algorithm Selection and Hyper-parameter tuning (CASH). The CASH part costs 20% of the time that the data scientists spend building a machine learning pipeline for a certain problem [34]. Moreover, the proposed framework considers data characteristics, clustering algorithms' characteristics, and users' needs during the recommendation process of optimal clustering pipeline selection. Furthermore, the SOL-AutoClust framework automates the process of determining the optimal number of clusters for the given dataset.

The rest of this paper is organized as follows: In Section 2, we present a brief review of related work. In Section 3, Section 4, and Section 5, we propose our solution for recommending a complete clustering pipeline for any given dataset. The experimental results are presented in Section 6. Finally, Section 7 concludes and proposes directions for possible future work.

2. RELATED WORK

Machine learning is a pivotal scientific discipline that finds applications across various domains, providing solutions to intricate problems. This section examines existing automated machine

learning (Auto-ML) frameworks, offering a concise overview of each framework's functionality. Each of the presented frameworks aims to automate specific tasks within the machine learning process, either partially or entirely. Notably, recent advancements have led to the development of several auto-ML frameworks that utilize powerful machine learning packages that readily accessible.

2.1 Auto-ML in Supervised Learning

Auto-WEKA [35], implemented on top of the widely used data mining tool WEKA [36], was the pioneering Auto-ML framework. Auto-WEKA focuses on automating the selection of machine learning algorithms and tuning hyperparameters, addressing what is known as the Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem. By leveraging the SMAC [37] optimization algorithm and feature selection algorithms within WEKA, Auto-WEKA provided solutions for the CASH problem. The subsequent release, Auto-WEKA 2.0 [38], integrated updates into the WEKA ecosystem and expanded its support to include regression problems.

Auto-sklearn [39], built on the popular Python machine learning library Scikit-Learn [55], stands as one of the most well-known auto-ML frameworks. It also considers the CASH problem, further enhancing the approach by incorporating meta-learning [40]. This meta-learning approach involves recognizing dataset characteristics and recommending the most suitable machine learning pipeline based on those characteristics. Auto-sklearn achieves this by computing a set of meta-features, such as the number of instances, features, classes, and data skewness [56]. During the training stage, Auto-sklearn evaluates 140 different datasets from the OpenML repository [41] to calculate meta-features and utilizes Bayesian optimization [42] to determine the best-performing machine learning pipelines for each dataset. When presented with a new dataset, Auto-sklearn calculates its meta-features and selects the stored machine learning pipelines from the nearest 25 datasets based on the L1 distance measurement [16]. L1 distance between two datasets d_i and d_j in meta-feature space can be computed as in Eq. (1). Auto-sklearn also incorporates automated ensemble model construction, retaining all considered models from the training stage and employing a post-processing method to create an ensemble, mitigating the risk of overfitting. Comparative studies have shown that Auto-sklearn outperforms the Auto-WEKA system in approximately 86% of cases [43].

$$L1_D(d_i, d_j) = \sum_{k=1}^{n} |d_{ik} - d_{jk}| \quad (1)$$

where n represents meta-features

TPOT [44] has since been adapted to handle a wide range of machine learning problems. It is an open-source auto-ML framework that automates tasks such as feature preprocessing, model selection, and hyperparameter optimization. Similar to Auto-sklearn, TPOT gained popularity due to its integration with the Scikit-Learn library. TPOT utilizes a genetic programming algorithm [45], to construct genetic programming trees, combining algorithms within machine learning pipelines. Additionally, TPOT employs different algorithms for various tasks, including feature preprocessing (e.g., Principal Component Analysis, Scalers), feature selection (e.g., Recursive Feature Elimination, Variance Thresholds), and classification (e.g., K-Nearest Neighbors, Decision Tree, Random Forest). Overall, TPOT stands as one of the most prominent auto-ML frameworks available today.

While the aforementioned auto-ML frameworks dominate the open-source landscape, several other frameworks have emerged in recent years. H2O [46], for instance, is an open-source distributed automated machine learning framework that facilitates training large-scale machine learning models. It provides server-based training accessible through APIs in multiple programming languages such as R, Python, Java, and Scala. H2O encompasses feature engineering, data preprocessing, model selection, and hyperparameter optimization, employing fast random search and stacked ensembles to optimize the recommended pipeline.

LightAutoML [47], an open-source auto-ML framework primarily designed for the financial sector, automates various steps in the machine learning pipeline, including feature engineering, data preprocessing, model selection, and hyperparameter optimization.

AMLBID [48] is another open-source auto-ML framework that utilizes a meta-learning-based approach to automate machine learning model building for industrial data.

ATM [49], an example of a distributed and scalable auto-ML framework, enables machine learning users to upload datasets, select from a range of machine learning algorithms, and define a search space for hyperparameters. By leveraging Bayesian optimization and meta-learning techniques, ATM recommends optimal machine learning pipelines for the given datasets.

ML-Plan [50], relying on hierarchical task networks (HTNs) [51], automates algorithm selection and configuration tasks within the machine learning process.

AlphaD3M [52] employs reinforcement learning techniques to optimize machine learning pipelines. It utilizes iterative experiments with different pipelines, incorporating actions such as inserting, deleting, or replacing pipeline parts to discover or recommend the optimal pipeline.

Furthermore, SmartML [53], the first auto-ML framework for automating classification problems using the R programming language and its associated package, constructs a knowledge base comprising meta-features and algorithm performance across different training datasets. When provided with a dataset, SmartML extracts its meta-features and compares them with those stored in the knowledge base using the nearest neighbor technique. The most similar datasets are then used to identify the best-performing algorithms, ultimately recommending the most suitable algorithm. SmartML employs SMAC Bayesian Optimization [54] to optimize the hyperparameters of the selected algorithm.

2.2 Auto-ML in Unsupervised Learning

Several methods [57-63] assist novice analysts in choosing a promising clustering algorithm for unsupervised learning tasks. These approaches employ meta-learning, which involves learning from previous experiences to select the most suitable algorithm for a new dataset. However, these methods solely focus on the clustering algorithm and disregard the corresponding hyperparameters.

Various techniques exist for optimizing the hyperparameters of clustering algorithms. These techniques utilize a fixed clustering algorithm with different hyperparameters and require a prior definition of promising values within the hyperparameter search space. They can be categorized into two groups: exhaustive methods and non-exhaustive methods. Exhaustive methods include the entire search space of clustering algorithms and hyperparameters in order to identify the optimal clustering result. This selection is based on a predefined Cluster Validity Index (CVI), such as Calinski and Harabasz index (CH) [64], Davies-Bouldin index (DB) [65], Silhouette Width (SW) [66]. On the other hand, non-exhaustive methods only consider a subset of hyperparameter values. Examples of non-exhaustive methods include LOG-Means [67], g-means [68], HDBSCAN [69], and X-means [70]. However, both types of methods do not address the CASH problem as they only focus on a single algorithm.

In a recent study on auto-ML solutions for clustering, researchers in [71] introduced AutoML4Clust, an auto-ML approach that effectively addresses the CASH problem by exploring the configuration space. The authors investigate various optimizers and CVIs in combination and find that no single CVI is universally suitable for all types of datasets. However, AutoML4Clust does not provide a solution for reducing the potentially large configuration space and does not assist in selecting the appropriate CVI. Furthermore, although it efficiently explores configurations using optimizers, it does not explore methods to identify well-performing configurations early in the process. AutoClust [72] and AutoCluster [73], on the other hand, employ meta-learning to sequentially address the CASH problem by narrowing down the configuration space to a single algorithm and subsequently optimizing its hyperparameters. Both approaches use meta-learning to identify datasets with similar characteristics and select the algorithm that performs best based on these datasets. For this purpose, they utilize landmarking [74] meta-features to describe dataset characteristics, which involve executing one or multiple clustering algorithms and utilizing the clustering results as meta-features. Then they proceed to optimize the hyperparameters of the selected algorithm.

ML2DAC [75] is an auto-ML approach that automates the clustering process. By leveraging the core principles of meta-learning, ML2DAC automates the clustering process by identifying well-performing configurations and recommending appropriate clustering algorithms based on a chosen validity index.

Table 1: Comparison Between State-Of-The-Art Clustering Auto-ML Frameworks.

	AutoML4Clust	AutoClust	ML2DAC
Year	2021	2020	2023
Language	Python	Python	Python
Supported data type	Tabular	Tabular	Tabular
Data size	Small	Small	Small
Feature preprocessing	No	No	No
User technicality level	High	High	High
Algorithms' characteristics	No	No	No
Computational Power	High	High	High

In table 1, a feature comparison between the most recent and popular state of the art frameworks is presented.

While all of the existing Auto-ML Frameworks provide partial or complete ML pipeline automation, each one works differently and targets different algorithms or dataset structures. Although all of these auto-ML frameworks offer varying degrees of automation for machine learning pipelines, each framework operates differently and focuses on different algorithms or dataset structures. While some frameworks incorporate meta-learning approaches and multiple CVIs, certain challenges remain despite these features. Among those challenges are the following:

Challenge 1: The effectiveness of these frameworks is confined within the boundaries of simple and small datasets. While they excel in handling these modest-sized datasets, their capabilities begin to diminish when confronted with more complex and larger-scale data. These frameworks provide efficient and reliable solutions for straightforward and compact data analysis tasks, but their limitations become apparent when faced with the intricacies and voluminous nature of larger datasets. Therefore, it is crucial to consider alternative approaches or more robust frameworks when dealing with more extensive and intricate data sets to ensure accurate and comprehensive analysis.

Challenge 2: These frameworks exhibit limitations in their approach by solely focusing on the characteristics of the dataset, neglecting the crucial aspects of clustering algorithms' characteristics and users' needs. By disregarding these vital factors, these frameworks fail to provide comprehensive and tailored solutions for clustering tasks. The success of clustering algorithms depends not only on the properties of the dataset itself but also on the specific characteristics of the algorithms employed and the requirements and preferences of the users. By disregarding these aspects, these frameworks miss the opportunity to optimize clustering outcomes, potentially leading to suboptimal results. Therefore, it is essential to consider the broader context, encompassing both dataset characteristics and the specific requirements of clustering algorithms and users, to ensure more effective and personalized clustering solutions.

Challenge 3: These frameworks possess an inherent limitation in their applicability, as they are specifically tailored for structured datasets. While they excel in handling data that adheres to a well-defined schema or format, their effectiveness diminishes when confronted with unstructured or semi-structured data. Structured datasets, characterized by organized and predefined relationships among data elements, align seamlessly with the capabilities of these frameworks. However, when faced with unstructured datasets that lack a pre-defined schema or have irregular data patterns, these frameworks may struggle to extract meaningful insights or perform accurate analyses. Therefore, it is crucial to explore alternative approaches or adapt these frameworks to accommodate the complexities of unstructured data when dealing with such datasets to ensure comprehensive and reliable data processing and analysis.

Challenge 4: The execution of auto-ML experiments using these frameworks demands substantial computational power. The intricate nature of automated machine learning (auto-ML) tasks, which involve processes such as feature engineering, model selection, hyperparameter tuning, and ensemble learning, requires significant computational resources to handle the complexity and volume of data involved. These frameworks rely on intensive computations and iterative algorithms to explore and optimize the wide range of possibilities inherent in auto-ML. Consequently, to ensure efficient and timely completion of auto-ML experiments, it becomes imperative to deploy these frameworks on systems equipped with substantial computational capabilities, such as high-performance computing clusters or cloud-based infrastructures. By harnessing the requisite computational power, these frameworks can effectively tackle the demanding nature of auto-ML experiments, enabling researchers and practitioners to enhance the efficiency and effectiveness of their machine learning workflows.

Despite the progress in Auto-MLAuto-ML frameworks, a critical gap persists in automating unsupervised learning, particularly clustering, which remains under-addressed compared to supervised learning. Existing frameworks, such as Auto-WEKA, Auto-sklearn, and TPOT, predominantly focus on supervised tasks, while unsupervised solutions like AutoML4Clust, AutoClust, and ML2DAC exhibit limitations: they either narrowly address the Combined Algorithm Selection and Hyperparameter (CASH) problem, rely solely on dataset characteristics, or ignore user-specific requirements. Furthermore, these frameworks struggle with scalability, computational efficiency, and adaptability to unstructured data, as highlighted in Challenges 1–4. The literature reveals that no current approach holistically integrates data characteristics, clustering algorithm traits (e.g., outlier sensitivity), and users need to recommend an

end-to-end clustering pipeline. This gap underscores the need for a robust Auto-ML framework that bridges these dimensions, ensuring optimal clustering outcomes across diverse real-world applications. Our proposed framework directly tackles this gap by introducing an Auto-ML framework that uniquely combines data characteristics and clustering algorithms characteristics with user-centric customization, aiming to significantly improve automated clustering for real-world applications.

In the next section, we will discuss the proposed SOL-AutoClust framework, which is proposed to tackle these challenges.

3. PROPOSED AUTO-ML FRAMEWORK: SOL-AUTOCLUST

In this section, we introduce a novel automated machine learning framework (SOL-AutoClust) that aims to overcome the limitations of the existing frameworks. SOL-AutoClust is an auto-ML framework that is built on top of the Python library Scikit-Learn. SOL-AutoClust includes all steps of the clustering process pipeline, including data preprocessing, feature engineering, and clustering algorithm selection. Currently, the algorithms' search space of the proposed SOL-AutoClust framework consists of 10 different clustering algorithms. The different algorithms that have been used in the complete pipeline lifecycle are shown in table 2.

As shown in table 2, the algorithm selection in our clustering pipeline was systematically designed to ensure robustness, scalability, and adaptability across diverse datasets, aligning with the core research objective of generalizable and reproducible outcomes. Preprocessing techniques—including Ordinal/One Hot Encoders, multiple Imputers, Log Transformer, and Min Max Scaler—address heterogeneous data types, missing values, and feature normalization, while feature selection methods (Variance Threshold, Collinearity Removal, and Random Forest-based selection) and engineering strategies (PCA, Polynomial Features) optimize dimensionality reduction and non-linear relationship modeling. The clustering phase integrates a spectrum of algorithms to rigorously evaluate varied data structures: K-means variants for spherical clusters, Affinity Propagation/ Mean-shift for non-parametric shapes, DBSCAN/OPTICS for density-based patterns, Spectral Clustering/Gaussian Mixtures for complex manifolds, and hierarchical methods (Agglomerative, BIRCH) for multi-scale analysis.

In general, SOL-AutoClust framework automates the process of recommending the appropriate clustering pipeline for the given dataset, depending on both the dataset and the clustering algorithms' characteristics. The pipeline steps of the online recommendation process are shown in figure 2. This recommendation process consists of three main steps: dataset characterization, appropriate algorithms identification, and automatic pipeline recommendation. More details about the recommendation pipeline are presented in the following subsections.

3.1 Data Repository

Mapping configuration and clustering algorithms' characteristics are stored in the repository to be utilized in the online recommendation process. Mapping configuration is used to classify the given dataset based on its size and dimensionality. The mapping configuration boundaries for each type is shown in table 3. On the other hand, clustering algorithms' characteristics are stored in the repository to be used while selecting the appropriate clustering algorithms based on the characteristics of the given dataset. The clustering algorithms and their characteristics are presented in table 4.

3.2 Dataset Characterization

In this step, the characteristics of the given dataset are extracted and mapped using the mapping configuration. These characteristics of the given dataset are mapped to be compatible with the algorithms' characteristics, consequently facilitating the process of appropriate clustering algorithm selection. A sample of these characteristics and their descriptions is shown in table 5. "Number of instances" and "Number of features" characteristics are used to classify the dataset into scale size and dimensionality degree based on the boundaries in the mapping configuration.

3.3 Appropriate Algorithms Identification

After extracting dataset characteristics, in this step, the extracted characteristics are compared against the characteristics of clustering algorithms so, the appropriate algorithms, are selected to be executed. An example of selecting appropriate clustering algorithms for a given dataset is shown in figure 3.

3.4 Automatic Pipeline Recommendation

After identifying the relevant set of clustering algorithms that suit the characteristics of the given dataset, this relevant set is used to select the optimal pipeline. The process of optimal pipeline recommendation consists of three main tasks: pruning clustering algorithms, data preprocessing, and optimal pipeline recommendation.

Table 5: Datasets Characteristics Sample.

Feature	Description
Number of instances (Cardinality)	# rows in the dataset
Number of features (Dimensionality)	Number of columns in the dataset
Anomalies data?	Whether the dataset contains outliers or not
Number of missing values (Null Count)	Total number of missing values across the dataset
Number of numeric features (Numerical Degree)	Number of numerical columns
Number of categorical features (Categorical Degree)	Number of categorical columns
Number of binary features (Binary Degree)	Number of binary (with two distinct values) columns

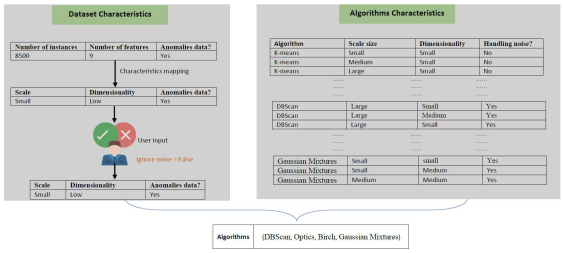

Figure 3: An example of Selecting Relevant Algorithms

3.4.1 Clustering algorithms pruning

The pruning process is adapted to select the dominant clustering algorithms from those in the relevant set. The pruning step depends on three dimensions: complexity, scalability, and the number of input parameters to select the dominant clustering algorithms. We outline the pruning process in algorithm 1.

3.4.2 Data preprocessing

This task aims to clean the dataset and perform a set of preprocessing steps on the given dataset to increase the accuracy of the clustering process. These preprocessing steps are classified into two types: mandatory steps and optional steps. The mandatory and optional steps are shown in table 6.

Table 6: Data Preprocessing Steps.

Mandatory steps	Optional steps
Categorical feature encoding	Feature selection
Feature scaling	Feature engineering
Missing data imputation	Removing outliers

3.4.3 Optimal pipeline recommendation

After cleaning the dataset by applying the mandatory preprocessing steps, the dominant clustering algorithms are used to cluster the cleaned dataset. If there were outliers in the given dataset and the user input was ignoring them, a preliminary iteration is executed to remove these outliers. Then, each algorithm in the dominant set is executed four times: the first iteration is without applying any optional step; the second iteration is by applying feature selection; the third iteration is by applying feature engineering; and the fourth iteration is by applying both feature selection and feature engineering. The silhouette score is calculated for each iteration for all dominant clustering algorithms, and the pipeline that achieves the minimum silhouette score is recommended for the given dataset.

4. EXPERIMENTAL EVALUATION

In this section, we present an evaluation of SOL-Auto-Clust's ability to automatically recommend the best clustering pipeline for a given dataset. We implement all of our code in Python version 3.11.

4.1 ML Pipeline Recommendation Evaluation

In this section, we demonstrate how combining data characteristics and clustering characteristics improves SOL-Auto-Clust's ability to find a suitable clustering pipeline for a given dataset. To accomplish this, we evaluated SOL-Auto-Clust by comparing it to the clustering Auto-ML pipeline generation platforms discussed in existing literature. In general, there are two types of clustering evaluation measures or metrics. Internal measures do not require any ground truth to assess the quality of clusters. They are based solely on the data and the clustering results. External measures compare the clustering results to ground truth labels. Choosing the best metrics for evaluating a clustering model depends on the specific objectives of your analysis. If the true cluster labels (ground truth) are unknown, internal metrics such as silhouette score (SC) [19] are useful. However, if the true cluster labels are known, external metrics like adjusted rand index (ARI) [86] can be used. Our proposed framework provides the two types to serve all cases. We performed different experiments where we tracked multiple evaluation measures such as adjusted rand

index (ARI) and silhouette score (SC). We compared SOL-Auto-Clust with three open-source Auto-Clust pipeline generation platforms: ML2DAC [75], Auto-Clust [72], and Auto-Cluster [73].

4.1.1 Datasets

We used 16 public datasets with different characteristics to compare our proposed framework to the existing frameworks. We used a total of 16 public datasets with different characteristics in order to conduct a comprehensive evaluation of our proposed framework in comparison to the existing frameworks. We performed 4 experiments for every dataset utilizing each framework. The details about the datasets used in the evaluation are presented in table 7.

4.1.2 Solution Environment and Hardware

The proposed framework was implemented in Python 3.11 using a virtual machine with Ubuntu 20.04, 32 GB of RAM, and 8 cores to compare it with ML2DAC [75], Auto-Clust [72], and Auto-Cluster [73].

4.2 Evaluation Results and Analysis

In this section, we shall delve into an examination of the conducted experimental outcomes, aiming to compare our innovative SOL-Auto-Clust framework with the existing frameworks available. For every dataset, we conducted four experiments with each framework. The results obtained were consistent across SOL-Auto-Clust, Auto-Clust, and Auto-Cluster; however, there was significant variability observed in the outcomes for ML2DAC across different experiments. The performance of our proposed framework compared to Auto-Clust and Auto-Cluster across all evaluation datasets is displaied in table 8. The clustering algorithms recommended and the number of clusters generated by SOL-Auto-Clust, Auto-Clust, and Auto-Cluster are shown in table 9. Conversely, table 10 presents the performance of ML2DAC on each evaluation dataset over four runs. Additionally, table 11 illustrates the recommended clustering algorithms and the number of clusters generated by ML2DAC in these runs. The discrepancies in ML2DAC's results are attributed to its reliance on two configurations: warm-starting configurations, which remain constant for each dataset in every run, and dynamic configurations determined by the Bayesian Optimization optimizer, which can vary between runs.

In the results of the experiments, presented in table 8 and table 10 it is shown that the proposed framework, SOL-Auto-Clust, outperforms the ML2DAC, Auto-Clust, and Auto-Cluster frameworks in terms of adjusted rand index (ARI) and silhouette score (SC) measures.

In figure 4 and figure 5, it is shown that ML2DAC, Auto-Clust, and Auto-Cluster face significant challenges when recommending suitable clustering algorithms for datasets that contain outliers. These approaches struggle to handle datasets such as "Covid-19", "Abalone", "Oil-Spill", "Air quality health impact", "Marketing Campaign", "Advanced IoT Agriculture", and "Diabetes prediction" which are known to have outliers. As a result, the recommended algorithms are limited in their effectiveness and can only provide accurate clustering results for clean datasets.

Another noteworthy observation is that ML2DAC, Auto-Clust, and Auto-Cluster encounter difficulties in suggesting appropriate clustering algorithms for high-dimensional datasets, specifically demonstrated in the "Covid-19" dataset. High-dimensional data presents unique challenges due to the increased complexity and the curse of dimensionality. The failure of these approaches to address these challenges further emphasizes their limitations in accurately recommending suitable algorithms in such scenarios. Auto-Cluster utilized 150 datasets consisting of a maximum of 5000 samples and 50 features to generate its meta-data. Thus, Auto-Cluster is constrained to working with small-scale and low-dimensional datasets.

On the other hand, the evaluation results presented in table 9 and table 11 compare the recommended clustering algorithms and the number of clusters generated by the proposed framework with Auto-Clust, Auto-Cluster and ML2DAC. The findings highlight some significant observations. Firstly, it is evident that ML2DAC tends to produce a large number of clusters for most datasets. For instance, the "CO2-Emission" and "Drug" datasets have 164 and 88 clusters by average respectively, despite containing 935 and 200 records respectively. This results in a situation where the number of clusters is approximately 25% of the total records, presenting challenges in terms of interpretability.

Consequently, the evaluation results provide valuable insights into the limitations of ML2DAC, Auto-Clust, and Auto-Cluster in comparison to the proposed framework. These limitations primarily relate to the handling of outliers, addressing high-dimensional datasets, and accurately determining the number of clusters. The proposed framework demonstrates superior performance in these areas, highlighting the importance of considering these factors when recommending clustering algorithms.

In general, there is a noticeable superiority to our proposed framework in terms of adjusted rand

index (ARI) and silhouette score (SC) measures when recommending the appropriate pipeline for a given dataset. This superiority is due to its reliance on an online learning approach. Furthermore, combining data characteristics and clustering algorithms' characteristics empowers the proposed framework to dominate the existing frameworks in the recommendation process.

Additionally, depending solely on the outcomes derived from similar datasets can be deceptive, as the two datasets may share numerous characteristics but possess subtle differences that could profoundly influence the performance of clustering algorithms. This highlights the importance of considering subjective evaluations in the form of online learning. By actively incorporating ongoing feedback and adapting the clustering algorithms in real-time, online learning provides a more effective approach to handle such nuanced variations. It allows for continuous adjustments and refinements based on the evolving understanding of the data, mitigating the risk of being misled by superficial similarities between groups. In this way, online learning offers a dynamic and responsive framework that can better capture the intricate nuances of data and optimize the clustering process, leading to more accurate and reliable results.

Furthermore, a key factor contributing to the superiority of the proposed framework is its holistic approach, which encompasses the entire pipeline of data preprocessing and clustering. Unlike existing frameworks that solely focus on the Cluster Analysis Selection and Hyperparameter (CASH) part, the proposed framework considers the complete set of pipeline steps. This includes crucial tasks such as encoding categorical features, feature scaling, feature selection, data cleaning, and outlier detection and handling. By considering these essential preprocessing steps, the proposed framework ensures that the data is appropriately prepared and optimized for clustering analysis. Only after completing these preprocessing steps does the framework proceed to the CASH part, where it intelligently selects the most suitable clustering algorithms and hyperparameters. This comprehensive approach enhances the accuracy and robustness of the clustering process, as it addresses potential issues and inconsistencies in the data prior to applying the clustering algorithms. In essence, the proposed framework goes beyond the limitations of existing frameworks by incorporating the entire pipeline, resulting in more reliable and effective clustering outcomes.

4.3 Difference from Prior Work

Prior unsupervised approaches like AutoML4Clust and ML2DAC focus narrowly on the Combined Algorithm Selection and Hyperparameter (CASH) problem, often relying on single Cluster Validity Indices (CVIs) or static meta-features. For instance, AutoML4Clust [71] explores configuration spaces but fails to reduce search complexity or adapt CVIs dynamically, while AutoClust [72] and AutoCluster [73] prioritize dataset similarity via landmarking without integrating algorithm-specific traits (e.g., outlier robustness). In general, most prior works overlook user-centric requirements, such as interpretability needs, and struggle with computational efficiency when scaling to large datasets.

In contrast, SOL-Auto-Clust introduces three key innovations. First, it unifies data characteristics (e.g., dimensionality, outlier density), algorithm traits (e.g., sensitivity to noise, scalability), and user-defined objectives (e.g., interpretability) into a new framework, enabling complete pipeline recommendations. Second, it employs a dynamic multi-CVI strategy to evaluate clustering quality, addressing the limitation of single-CVI dependency in AutoML4Clust. Third, it automates end-to-end tasks, including outlier-aware preprocessing and optimal cluster count determination. Experiments demonstrate SOL-Auto-Clust's superiority in handling high-dimensional data and outliers, outperforming benchmarks in Adjusted Rand Index (ARI) and Silhouette Score. Furthermore, the framework minimizes computational overhead by employing smart configuration pruning that leverages meta-feature extraction. Furthermore, while SOL-Auto-Clust performs exceptionally on structured data, its inability to handle unstructured formats (such as text and images) currently poses a limitation, which is planned to be addressed in future work. These advancements and limitations position SOL-Auto-Clust as a scalable, user-aligned solution that bridges critical gaps in automated clustering while acknowledging avenues for further refinement.

5. CONCLUSION AND FUTURE WORK

The SOL-Auto-Clust framework successfully tackles the important problem of automating unsupervised clustering for current Auto-ML solutions. By holistically integrating data characteristics, algorithm properties, and user needs, SOL-Auto-Clust overcomes the fragmented approaches of prior works, such as the narrow CASH focus of AutoML4Clust or AutoCluster's

dependence on dataset similarity. Empirical validation across diverse and complex datasets confirms its superior performance, particularly in handling high-dimensional and outlier-rich scenarios. The framework's automated determination of cluster counts and outlier-aware preprocessing significantly reduces manual intervention, a key bottleneck in earlier methodologies. While the current reliance on structured data meta-features suggests an avenue for future development regarding unstructured data, the inherent scalability of SOL-Auto-Clust underscores its practical utility. Future work will focus on expanding its capabilities to unstructured data and enhancing interpretability, further solidifying its role in human-aligned machine learning pipelines.

REFERENCES:

[1] K. He, X. Zhang, S. Ren, J. Sun (2016) Deep Residual Learning for Image Recognition. In the Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778.

[2] Sheykhmousa, Mohammadreza, et al. (2020) Support Vector Machine Versus Random Forest for Remote Sensing Image Classification: A Meta-Analysis and Systematic Review. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 13, pp. 6308–6325.

[3] M. A. Chandra, S. S. Bedi (2021) Survey on SVM and their application in image classification. International Journal of Information Technology, vol. 13, no. 5, pp. 1–11.

[4] Minaee, Shervin, et al. (2021) Deep Learning--based Text Classification: A Comprehensive Review. ACM computing surveys, CSUR, 54(3), 1-40.

[5] Takahashi, K. Z. (2023) Molecular cluster analysis using local order parameters selected by machine learning. Physical Chemistry Chemical Physics, 25(1), 658-672.

[6] Vincent, Tom, et al. (2023) Data cluster analysis and machine learning for classification of twisted bilayer graphene. Carbon, 201, 141-149.

[7] Keogh, Tracey M., et al. (2023) Cluster analysis reveals distinct patterns of childhood adversity, behavioral disengagement, and depression that predict blunted heart rate reactivity to acute psychological stress. Annals of Behavioral Medicine, 57(1), 61-73.

[8] Ali, Nafees, et al. (2023) Classification of reservoir quality using unsupervised machine learning and cluster analysis: Example from Kadanwari gas field, SE Pakistan. Geosystems and Geoenvironment, 2(1), 100123.

[9] Hong, Yejin, Sungmin Yoon, and Sebin Choi. (2023) Operational signature-based symbolic hierarchical clustering for building energy, operation, and efficiency towards carbon neutrality. Energy, 265, 126276.

[10] A. Graves, A. Mohamed, G. Hinton (2013) Speech recognition with deep recurrent neural networks. In IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 6645–6649.

[11] Abdulkareem, Nasiba M., et al. (2021) COVID-19 World Vaccination Progress Using Machine Learning Classification Algorithms. Qubahan Academic Journal, vol. 1, no. 2, Art. no. 2.

[12] R. Sujitha, V. Seenivasagam (2021) Classification of lung cancer stages with machine learning over big data healthcare framework. Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 5, pp. 5639–5649.

[13] H. Jain, G. Yadav, R. Manoov (2021) Churn Prediction and Retention in Banking, Telecom and IT Sectors Using Machine Learning Techniques. In Advances in Machine Learning and Computational Intelligence, Singapore, pp. 137–156.

[14] Iatrellis, Omiros, et al. (2021) A two-phase machine learning approach for predicting student outcomes. Education and Information Technologies, vol. 26, no. 1, pp. 69–88.

[15] Aljohani, A. (2023) Predictive analytics and machine learning for real-time supply chain risk mitigation and agility. Sustainability, 15(20), 15088.

[16] Kumar, V., Chhabra, et al. (2014) Performance evaluation of distance metrics in the clustering algorithms. INFOCOMP Journal of Computer Science, 13(1), 38-52.

[17] Hasan, M. R., & Ferdous, J. (2024) Dominance of AI and Machine Learning Techniques in Hybrid Movie Recommendation System Applying Text-to-number Conversion and Cosine Similarity Approaches. Journal of Computer Science and Technology Studies, 6(1), 94-102.

[18] Apat, Shraban Kumar, et al. (2023) An artificial intelligence-based crop recommendation system using machine learning. Journal of Scientific & Industrial Research (JSIR), 82(05), 558-567.

[19] Řezanková, H. A. N. A. (2018). Different approaches to the silhouette coefficient calculation in cluster evaluation. In 21st

International Scientific Conference AMSE Applications of Mathematics and Statistics in Economics, pp. 1-10.

[20] J. Bergstra, R. Bardenet, Y. Bengio, and B. K´egl (2011) Algorithms for hyperparameter optimization. In Proceedings of the 24th International Conference on Neural Information Processing Systems, pages 2546–2554.

[21] Fernández-de-Las-Peñas, César, et al. (2023) Clustering analysis reveals different profiles associating long-term post-COVID symptoms, COVID-19 symptoms at hospital admission and previous medical co-morbidities in previously hospitalized COVID-19 survivors. Infection 51.1, 61-69.

[22] Song, Jiyoun, et al. (2023) The identification of clusters of risk factors and their association with hospitalizations or emergency department visits in home health care. Journal of advanced nursing 79.2, 593-604.

[23] Meng, Lu, et al. (2023) Cluster analysis of adults unvaccinated for COVID-19 based on behavioral and social factors, National Immunization Survey-Adult COVID Module, United States. Preventive Medicine 167, 107415.

[24] Park, Sanghyun, Seungmo Kim, and Weisheng Chiu (2023) Segmenting sport fans by eFANgelism: a cluster analysis of South Korean soccer fans. Managing Sport and Leisure 28.2, 182-196.

[25] Elgammal, Islam, Ghada Talat Alhothali, and Annarita Sorrentino (2023) Segmenting Umrah performers based on outcomes behaviors: a cluster analysis perspective. Journal of Islamic Marketing 14.3, 871-891.

[26] Rajput, Lucky, and Shailendra Narayan Singh (2023) Customer Segmentation of E-commerce data using K-means Clustering Algorithm. In 2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence). IEEE.

[27] Li, Yue, et al. (2023) Customer segmentation using K-means clustering and the hybrid particle swarm optimization algorithm. The Computer Journal 66.4, 941-962.

[28] Kumar, Amit (2023) Customer Segmentation of Shopping Mall Users Using K-Means Clustering. Advancing SMEs Toward E-Commerce Policies for Sustainability. IGI Global, 248-270.

[29] Das, Debasmita, Parthajit Kayal, and Moinak Maiti (2023) A K-means clustering model for analyzing the Bitcoin extreme value returns. Decision Analytics Journal 6, 100152.

[30] Balcilar, Mehmet, Ahmed H. Elsayed, and Shawkat Hammoudeh (2023) Financial connectedness and risk transmission among MENA countries: Evidence from connectedness network and clustering analysis. Journal of International Financial Markets, Institutions and Money 82, 101656.

[31] Li, Chang, et al. (2023) Analysis and Categorization of Stock Price Factors via a Novel Framework based on Computer Science Technology. World Journal of Technology and Scientific Research 12.2023, 361-366.

[32] Stevens, T. M., et al. (2023) Teacher profiles in higher education: the move to online education during the COVID-19 crisis. Learning Environments Research, 1-26.

[33] Ifenthaler, Dirk, Clara Schumacher, and Jakub Kuzilek (2023) Investigating students' use of self-assessments in higher education using learning analytics. Journal of Computer Assisted Learning 39.1, 255-268.

[34] M.-A. Zöller, M. F. Huber (2021) Benchmark and Survey of Automated Machine Learning Frameworks. Journal of Artificial Intelligence Research, vol. 70, pp. 409–472.

[35] Thornton C, et al. (2013) Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[36] G. Holmes, A. Donkin, I. H. Witten (1994) WEKA: a machine learning workbench. In Proceedings of ANZIIS ’94 - Australian New Zealnd Intelligent Information Systems Conference, pp. 357–361.

[37] F. Hutter, H. H. Hoos, K. Leyton-Brown (2011) Sequential Model-Based Optimization for General Algorithm Configuration. In Learning and Intelligent Optimization, Berlin, Heidelberg, pp. 507–523.

[38] Kotthoff, C. Thornton, et al. (2019) Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA. Springer International Publishing, pp. 81–95.

[39] Feurer, Matthias, et al. (2015) Efficient and Robust Automated Machine Learning. In Advances in Neural Information Processing Systems, vol. 28.

[40] J. Vanschoren (2019) Meta-Learning. In Automated Machine Learning, Springer International Publishing, pp. 35–61.

[41] Vanschoren, Joaquin, et al. (2014) OpenML: networked science in machine learning. ACM

SIGKDD Explor. Newsl., vol. 15, no. 2, pp. 49–60.

[42] Feurer, Matthias, Jost Springenberg, and Frank Hutter (2015) Initializing Bayesian Hyperparameter Optimization via Meta-Learning. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29, no. 1, Art. no. 1.

[43] Guyon I, et al. (2015) Design of the 2015 ChaLearn AutoML challenge. International Joint Conference on Neural Networks (IJCNN).

[44] Olson, Randal S., et al. (2016) Automating Biomedical Data Science Through Tree-Based Pipeline Optimization. In Applications of Evolutionary Computation, pp. 123–137.

[45] Banzhaf, Wolfgang, et al. (1998) Genetic programming: an introduction: on the automatic evolution of computer programs and its applications. Vol. 1. San Francisco: Morgan Kaufmann Publishers Inc.

[46] E. LeDell, S. Poirier (2020) H2O AutoML: Scalable Automatic Machine Learning. Proceedings of the AutoML Workshop ICML, vol. 2020, p. 16.

[47] Vakhrushev, Anton, et al. (2022) LightAutoML: AutoML Solution for a Large Financial Services Ecosystem. arXiv preprint arXiv:2109.01528.

[48] Garouani, Moncef, et al. (2022) Using meta-learning for automated algorithms selection and configuration: an experimental framework for industrial big data. Journal of Big Data, vol. 9, no. 1, p. 57.

[49] T. Swearingen, W. Drevo, B. Cyphers, et al. (2017) ATM: a distributed, collaborative, scalable system for automated machine learning. In Proceedings of IEEE International Conference on Big Data, pp. 151–162.

[50] Mohr, Felix, Marcel Wever, and Eyke Hüllermeier (2018) ML-Plan: Automated machine learning via hierarchical planning. Machine Learning, vol. 107, no. 8, pp. 1495–1515.

[51] Ghallab, Malik, Dana Nau, and Paolo Traverso (2004) Automated Planning: Theory and Practice. Elsevier.

[52] I. Drori et al. (2021) AlphaD3M: Machine Learning Pipeline Synthesis. arXiv preprint arXiv:2111.02508.

[53] Maher, Mohamed Mohamed Maher Zenhom Abdelrahman, and Sherif Sakr (2019) SmartML: A Meta Learning-Based Framework for Automated Selection and Hyperparameter Tuning for Machine Learning Algorithms. In EDBT: 22nd International Conference on Extending Database Technology.

[54] S. Sakr, A. Y. Zomaya, Eds (2019) Encyclopedia of Big Data Technologies. Springer International Publishing.

[55] Pedregosa, Fabian, et al. (2011) Scikit-learn: Machine Learning in Python. the Journal of machine Learning research, vol. 12, no. 85, pp. 2825–2830.

[56] Hutter, Frank, Lars Kotthoff, and Joaquin Vanschoren (2019) Automated Machine Learning: Methods, Systems, Challenges. Springer Nature.

[57] Marcilio C.P. De Souto et al. (2008) Ranking and selecting clustering algorithms using a meta-learning approach. In Proceedings of the International Joint Conference on Neural Networks.

[58] Daniel G Ferrari and Leandro Nunes de Castro (2012) Clustering algorithm recommendation: a meta-learning approach. In International Conference on Swarm, Evolutionary, and Memetic Computing. Springer, 143–150.

[59] Daniel G Ferrari and Leandro Nune de Castro (2015) Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods. Information Sciences 301, 181–194.

[60] André C.A. Nascimento et al. (2009) Mining rules for the automatic selection process of clustering methods applied to cancer gene expression data. In Artificial Neural Networks – ICANN 2009, Vol. 5769 LNCS. Springer Berlin Heidelberg.

[61] Pimentel, Bruno Almeida, and Andre CPLF De Carvalho (2019) A new data characterization for selecting clustering algorithms using meta-learning. Information Sciences 477, 203-219.

[62] J A Sáez and E Corchado (2019) A Meta-Learning Recommendation System for Characterizing Unsupervised Problems: on Using Quality Indices to Describe Data Conformations. IEEE Access 7, 63247-63263.

[63] Rodrigo G F Soares, Teresa B Ludermir, and Francisco AT De Carvalho (2009) An analysis of meta-learning techniques for ranking clustering algorithms applied to artificial data. In International Conference on Artificial Neural Networks.

[64] Caliński, Tadeusz, and Jerzy Harabasz (1974) A Dendrite Method for Cluster Analysis. Communications in Statistics-theory and Methods 3.1, 1-27.

[65] David L. Davies and Donald W. Bouldin (1979) A Cluster Separation Measure. IEEE

Transactions on Pattern Analysis and Machine Intelligence 2, 224-227.

[66] Peter J. Rousseeuw (1987) Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics 20, 53-65.

[67] Manuel Fritz, Michael Behringer, and Holger Schwarz (2020) LOG-Means: Efficiently Estimating the Number of Clusters in Large Datasets. Proceedings of the VLDB Endowment 13.12, 2118-2131.

[68] Greg Hamerly and Charles Elkan (2003) Learning the K in K-Means. In Neural Information Processing Systems 17.

[69] Leland McInnes and John Healy (2017) Accelerated Hierarchical Density Based Clustering. In 2017 IEEE international conference on data mining workshops (ICDMW).

[70] Dan Pelleg and Andrew W Moore (2000) X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In Proceedings of the Seventeenth International Conference on Machine Learning, Vol. 1.

[71] Dennis Tschechlov, Manuel Fritz, and Holger Schwarz (2021) AutoML4Clust: Efficient AutoML for Clustering Analyses. In EDBT.

[72] Yannis Poulakis, Christos Doulkeridis, and Dimosthenis Kyriazis (2020) AutoClust: A Framework for Automated Clustering Based on Cluster Validity Indices. In 2020 IEEE International Conference on Data Mining (ICDM).

[73] Yue Liu, Shuang Li, and Wenjie Tian (2021) AutoCluster: Meta-learning Based Ensemble Method for Automated Unsupervised Clustering. In Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer International Publishing.

[74] Bernhard Pfahringer, Hilan Bensusan, and Christophe G. Giraud-Carrier (2000) Meta-Learning by Landmarking Various Learning Algorithms. In ICML, 743-750.

[75] Treder-Tschechlov, Dennis, et al. (2023) ML2DAC: Meta-Learning to Democratize AutoML for Clustering Analysis. Proceedings of the ACM on Management of Data 1.2, 1-26.

[76] James MacQueen et al. (1967) Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Vol. 1. 14. Oakland, CA, USA. pp. 281–297.

[77] Sculley D, editor (2010) Web-scale k-means clustering. Proceedings of the 19th international conference on World wide web; 2010 Apr 26–30; Raleigh, North Carolina, USA: Association for Computing Machinery.

[78] Frey BJ, Dueck D. (2007) Clustering by passing messages between data points. Science. 315.5814, 972-976.

[79] Dorin Comaniciu and Peter Meer (2002) Mean shift: A robust approach toward feature space analysis. IEEE Transactions on pattern analysis and machine intelligence 24.5, pp. 603–619.

[80] Andrew Y Ng, Michael I Jordan, and Yair Weiss (2002) On spectral clustering: Analysis and an algorithm. Advances in neural Information processing systems. pp. 849–856.

[81] Fionn Murtagh and Pierre Legendre (2014) Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion? In Journal of classification 31.3, pp. 274–295.

[82] Martin Ester et al. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd. Vol. 96. 34, pp. 226–231.

[83] Mihael Ankerst et al. (1999) OPTICS: ordering points to identify the clustering structure. ACM Sigmod record 28.2, pp. 9–60.

[84] Carl Edward Rasmussen (2000) The infinite Gaussian mixture model. Advances in neural Information processing systems. pp. 554–560.

[85] Tian Zhang, Raghu Ramakrishnan, and Miron Livny (1996) BIRCH: an efficient data clustering method for very large databases. In ACM Sigmod Record 25.2, pp. 103–114.

[86] Hubert, Lawrence, and Phipps Arabie (1985) Comparing partitions. Journal of classification 2, 193-218.

Table 2: The List of Algorithms Used By SOL-AutoClust.

Data preprocessing	Feature selection	Feature engineering	Clustering algorithm
Ordinal Encoder One Hot Encoder Log Transformer Imputer – Mean Imputer – Median Imputer – Simple (Mode) Imputer – KNN Min Max - Scaler	RFE-RandomForest Remove Collinearity Variance Threshold	PCA Polynomial Features	K-means [76] Bisecting K-means [77] Affinity Propagation [78] Mean-shift [79] Spectral [80] Agglomerative [81] DBScan [82] Optics [83] Gaussian Mixtures [84] Birch [85]

Table 3: Dataset Characteristics Mapping Configuration.

	Scale (No. of records)				Dimensionality (No. of columns)		
	Small	Medium	Large	VLarge	Small	Medium	Large
Minimum value	2	10001	100001	1000001	2	21	101
Maximum value	10000	100000	1000000	10000000	20	100	1000

Table 4: Characteristic of Clustering Algorithms (n = the number of objects to be clustered, k = the number of clusters, d = the number of dimensions, i = the number of iterations to convergence).

Clustering Algorithm	Scale size				Dimensionality			Handling Noise?	Scalability	Time Complexity
	Small	Medium	Large	VLarge	Small	Medium	Large			
K-means	✓	✓	✓	✓	✓	✓			High	O(n k d i)
Mini Batch		✓	✓	✓	✓	✓	✓		Medium	< O(n k d i)
DBScan	✓	✓	✓		✓	✓		✓	Medium	O(n log n)
Optics	✓	✓			✓	✓		✓	Medium	O(n2)
Affinity	✓	✓			✓	✓	✓		Low	O(n2i)
Mean-shift	✓	✓			✓	✓			Low	O(n2)
Spectral		✓	✓		✓	✓	✓		Low	O(n)
Agglomerativ	✓	✓	✓		✓	✓			Low	O(n2)
Birch	✓	✓	✓		✓	✓	✓	✓	High	O(n)
Gaussian	✓	✓	✓		✓	✓		✓	Low	O(k i n2)

Table 7: Characteristics of the 16 Datasets Used for The Evaluation.

Dataset	Cardinality	Degree	Outlier flag	URL
Telco-Customer-Churn	7043	21	No	https://www.kaggle.com/blastchar/telco-customer-churn
Anomaly-Detection	134229	8	Yes	https://www.kaggle.com/jorekai/anomaly-detection-falling-people-events
Covid-19	5644	106	Yes	https://www.kaggle.com/einsteindata4u/covid19
Malware-Detection	5210	70	No	https://www.kaggle.com/saurabhshahane/classification-of-malwares
Abalone	4168	9	Yes	https://archive.ics.uci.edu/ml/datasets/abalone
Oil-Spill	937	50	Yes	https://www.kaggle.com/paulh2718/oil-spill
Wine-Quality	1599	12	Yes	https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009
Cell-Samples	699	10	No	https://www.kaggle.com/datasets/ammaraahmad/top-10-machine-learning-datasets?select=cell_samples.csv
Cars-Clus	117	16	Yes	https://www.kaggle.com/datasets/ammaraahmad/top-10-machine-learning-datasets?select=cars_clus.csv
CO2-Emission	935	12	No	https://www.kaggle.com/datasets/ammaraahmad/top-10-machine-learning-datasets?select=CO2_emission.csv
Drug	200	6	No	https://www.kaggle.com/datasets/ammaraahmad/top-10-machine-learning-datasets?select=drug.csv
Air quality health impact	5811	15	Yes	https://www.kaggle.com/datasets/rabieelkharoua/air-quality-and-health-impact-dataset
Marketing Campaign	2240	29	Yes	https://www.kaggle.com/datasets/rodsaldanha/arketing-campaign
Commerce Shipping	10999	12	No	https://www.kaggle.com/datasets/prachi13/customer-analytics
Advanced IoT Agriculture	30000	14	Yes	https://www.kaggle.com/datasets/wisam1985/advanced-iot-agriculture-2024
Diabetes prediction	100000	9	Yes	https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset

Table 8: Clustering Auto-ML Frameworks' Performance Comparison (SOL-Auto-Clust vs Auto-Clust and Auto-Cluster).

Dataset	Auto-Clust		Auto-Cluster		SOL-Auto-Clust	
	ARI	SC	ARI	SC	ARI	SC
Telco-Customer-Churn	0.02	-0.08	0.012	0.006	0.98	0.34
Anomaly-Detection	0.1	0.005	0.03	0.07	0.68	0.54
Covid-19	-0.0008	0.001	0.001	0.06	0.50	0.61
Malware-Detection	0.04	-0.5	0.0002	-0.8	0.51	0.30
Abalone	0.02	0.09	0.14	0.03	0.51	0.56
Oil-Spill	0.002	0.03	0.25	0.004	0.91	0.98
Wine-Quality	0.0007	0.04	0.006	0.27	0.92	0.26
Cell-Samples	0.34	0.21	0.31	0.05	0.93	0.53
Cars-Clus	0.06	0.05	0.18	-0.0042	0.81	0.50
CO2-Emission	0.003	0.001	-0.016	0.08	0.92	0.58
Drug	0.04	-0.32	-0.007	-0.19	0.90	0.38
Air quality health impact	0.006	0.04	0.08	0.001	0.79	0.62
Marketing Campaign	0.007	0.03	0.16	0.007	0.63	0.24
Commerce Shipping	0.008	-0.05	0.014	-0.003	0.56	0.54
Advanced IoT Agriculture	-0.13	-0.41	-0.02	0.006	0.48	0.42
Diabetes prediction	0.2	0.004	0.09	0.14	0.81	0.37

Table 9: The Recommended Clustering Algorithms and the Number of Clusters (SOL-Auto-Clust vs Auto-Clust and Auto-Cluster).

Dataset	Auto-Clust		Auto-Cluster		SOL-Auto-Clust	
	Alg	No. of clust	Alg	No. of clust	Alg	No. of clust
Telco-Customer-Churn	KMeans	5	KMeans	5	KMeans	4
Anomaly-Detection	Agglomerative	4	DBScan	2	BIRCH	2
Covid-19	MeanShift	3	KMeans	4	BIRCH	2
Malware-Detection	KMeans	7	Agglomerative	5	Affinity Propagation	5
Abalone	MeanShift	2	KMeans	6	BIRCH	2
Oil-Spill	KMeans	2	Affinity Propagation	2	BIRCH	2
Wine-Quality	BIRCH	2	Agglomerative	2	OPTICS	2
Cell-Samples	Agglomerative	2	KMeans	2	Agglomerative	2
Cars-Clus	MeanShift	3	MeanShift	3	BIRCH	4
CO2-Emission	Agglomerative	3	Affinity Propagation	3	Affinity Propagation	2
Drug	MeanShift	5	KMeans	7	Affinity Propagation	2
Air quality health impact	MeanShift	2	KMeans	11	BIRCH	4
Marketing Campaign	KMeans	3	Affinity Propagation	2	BIRCH	2
Commerce Shipping	KMeans	4	KMeans	4	KMeans	4
Advanced IoT Agriculture	MeanShift	2	Affinity Propagation	2	DBScan	2
Diabetes prediction	Agglomerative	4	DBScan	2	BIRCH	2

Table 10: ML2DAC Performance Through 4 Experiments.

Dataset	ML2DAC (Exp. 1)		ML2DAC (Exp. 2)		ML2DAC (Exp. 3)		ML2DAC (Exp. 4)	
	ARI	SC	ARI	SC	ARI	SC	ARI	SC
Telco-Customer-Churn	0.02	-0.09	0.02	-0.095	0.02	-0.095	0.02	-0.095
Anomaly-Detection	0.07	0.09	-0.17	0.149	0.0065	0.036	0.0065	0.036
Covid-19	-0.005	-0.49	-0.005	-0.49	-0.005	-0.49	-0.005	-0.49
Malware-Detection	0.04	-0.45	0.02	-0.44	-0.657	0.996	0.04	-0.53
Abalone	0.006	0.42	0.006	0.42	-0.087	0.135	-0.005	0.435
Oil-Spill	0.0001	0.35	0.016	-0.42	0.0003	0.35	-0.93	-0.41
Wine-Quality	0.02	0.18	0.02	0.18	0.02	0.18	0.02	0.18
Cell-Samples	0.64	0.45	0.82	0.548	0.88	0.53	0.82	0.548
Cars-Clus	0.0	0.17	0.0	0.1	0.0	0.1	0.0	0.17
CO2-Emission	0.02	-0.096	0.016	-0.08	0.019	-0.09	0.02	-0.118
Drug	0.05	-0.29	0.039	-0.34	0.039	-0.34	0.05	-0.29
Air quality health impact	0.6	0.0127	0.6	0.0127	0.042	-0.011	0.6	0.0127
Marketing Campaign	0.12	0.398	0.19	0.175	0.19	0.175	0.12	0.398
Commerce Shipping	0.008	-0.07	0.02	-0.095	0.008	-0.07	0.008	-0.07
Advanced IoT Agriculture	0.18	0.053	0.35	0.24	0.18	0.053	0.35	0.24
Diabetes prediction	-0.0023	0.018	-0.0023	0.018	0.0084	0.054	-0.0023	0.018

Journal of Theoretical and Applied Information Technology
31st May 2025. Vol.103. No.10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 11: The Recommended Clustering Algorithms and the Number of Clusters by ML2DAC Through 4 Experiments.

Dataset	ML2DAC (Exp. 1)		ML2DAC (Exp. 2)		ML2DAC (Exp. 3)		ML2DAC (Exp. 4)	
	Alg	No. of clust	Alg	No. of clust	Alg	No. of clust	Alg	No. of clust
Telco-Customer-Churn	KMeans	11	KMeans	11	KMeans	11	KMeans	11
Anomaly-Detection	BIRCH	50	BIRCH	84	ward	51	ward	51
Covid-19	ward	10	ward	10	ward	10	ward	10
Malware-Detection	ward	95	BIRCH	2	ward	2	KMeans	80
Abalone	KMeans	9	KMeans	9	BIRCH	17	DBScan	2
Oil-Spill	KMeans	2	GMM	11	GMM	2	BIRCH	17
Wine-Quality	GMM	2	GMM	2	GMM	2	GMM	2
Cell-Samples	DBScan	2	KMeans	2	ward	2	KMeans	2
Cars-Clus	KMeans	3	GMM	2	GMM	2	KMeans	2
CO2-Emission	KMeans	149	KMeans	200	KMeans	185	GMM	121
Drug	ward	58	BIRCH	117	BIRCH	117	ward	58
Air quality health impact	ward	2	ward	2	BIRCH	10	ward	2
Marketing Campaign	GMM	2	ward	2	ward	2	GMM	2
Commerce Shipping	KMeans	13	ward	2	KMeans	13	KMeans	13
Advanced IoT Agriculture	BIRCH	24	ward	2	BIRCH	24	ward	2
Diabetes prediction	ward	47	ward	47	KMeans	78	ward	47

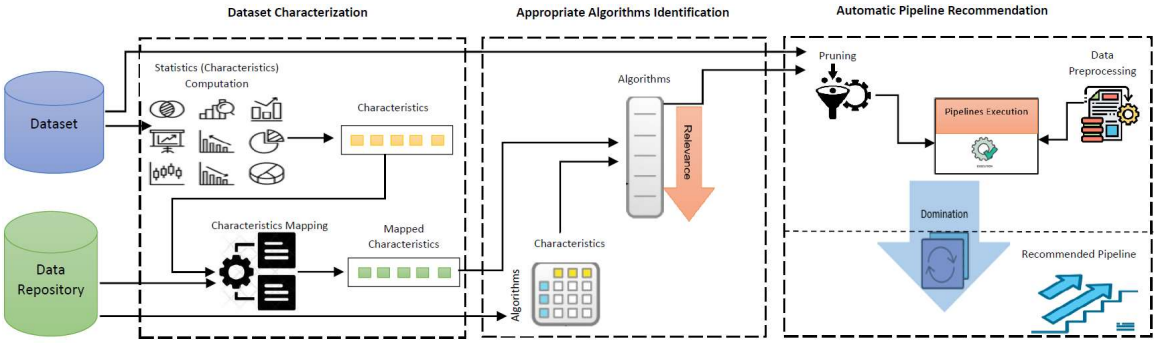

Figure 2: Online Clustering Recommendation Pipeline

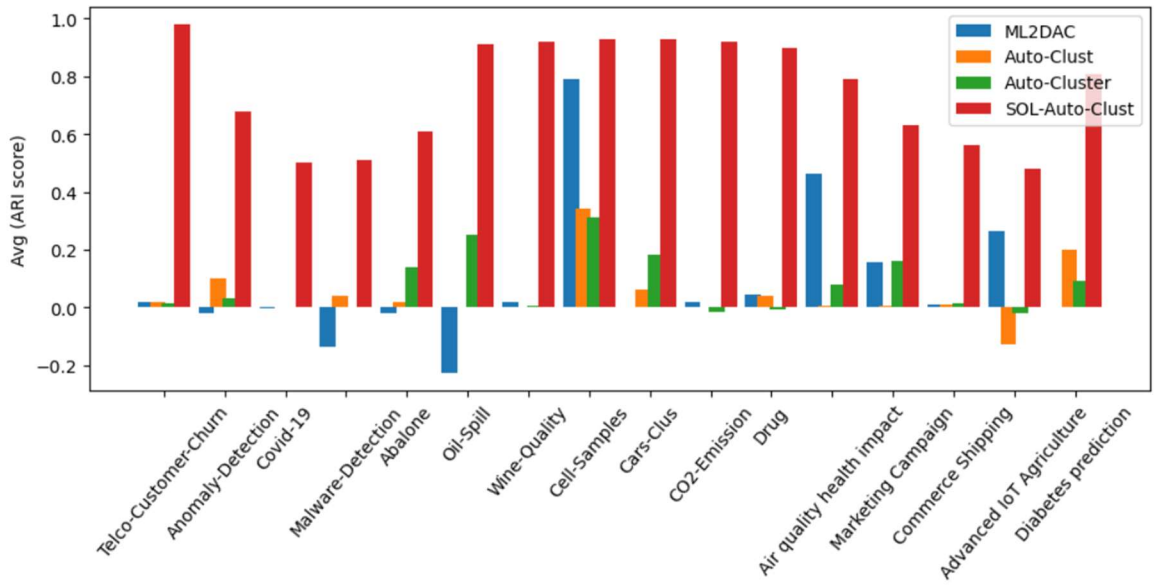

Figure 4: Performance Evaluation (ARI) SOL-Auto-Clust vs Existing Frameworks

Journal of Theoretical and Applied Information Technology
31st May 2025. Vol.103. No.10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

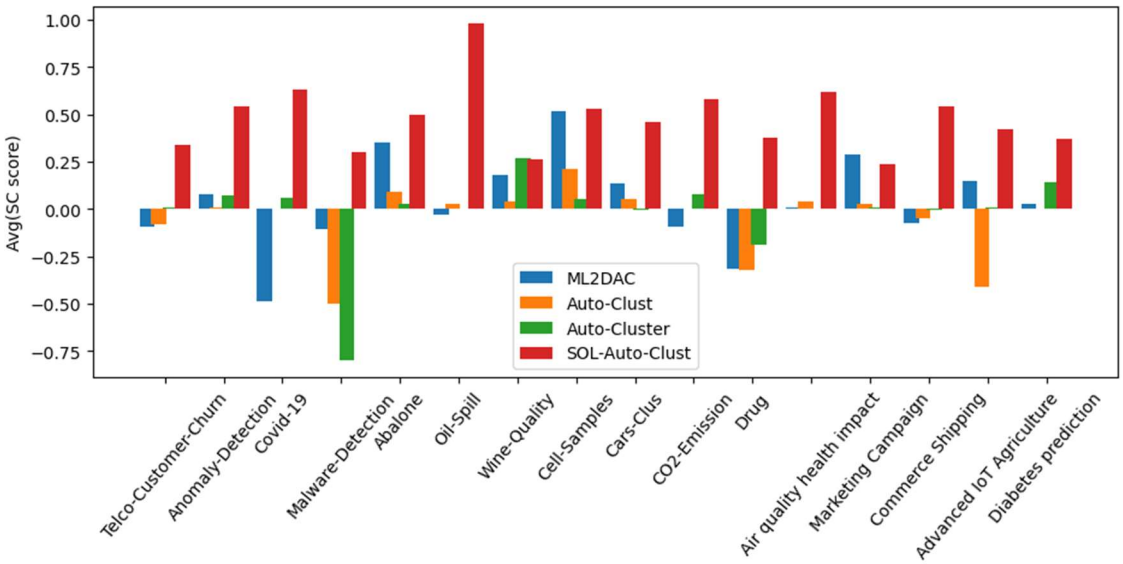

Figure 5: Performance Evaluation (SC) SOL-Auto-Clust vs Existing Frameworks

Algorithm 1 Clustering algorithms pruning

Input: Dataset D, The set of relevant clustering algorithms S_R, Clustering algorithms' characteristics C

Output: A set of the dominant clustering algorithms S_D

1: $dom_{alg} \leftarrow S_R[0]$ ▷ //set te first algorithm in S_R as the dominant algorithm
2: *Append* dom_{alg} *to* S_D
3: **for** $R_{alg} \subset S_R$ **do**
4: **for** $D_{alg} \subset S_D$ **do**
5: **if** $D_{alg}[scalability] > R_{alg}[scalability] \wedge D_{alg}[complixity] < R_{alg}[complixity] \wedge D_{alg}[hyperparameters] < R_{alg}[hyperparameters]$ **then**
6: continue
7: **else if** $D_{alg}[scalability] < R_{alg}[scalability] \wedge D_{alg}[complixity] > R_{alg}[complixity] \wedge D_{alg}[hyperparameters] > R_{alg}[hyperparameters]$ **then**
8: *Append* R_{alg} *to* S_D
9: *Remove* D_{alg} *from* S_D
10: **else**
11: *Append* R_{alg} *to* S_D
12: **end if**
13: **end for**
14: **end for**
15: $S_D \leftarrow$ set (S_D) ▷//*Toremoveduplicates*