Towards Automated Data Preprocessing for Clustering: Challenges and Future Directions

Leonard Labes[1]

[1]Institute for Parallel and Distributed Systems, University of Stuttgart, Universitätsstr. 38, Stuttgart, Germany

Abstract

This paper addresses the challenge of automating data preprocessing for clustering within unsupervised machine learning. A structured conceptual process is proposed, including imperfection diagnosis, candidate pipeline generation, quality estimation, selection and refinement, and result presentation. Through systematic identification and discussion, significant challenges such as diverse preprocessing techniques, combinatorial search space explosion, the selection of appropriate evaluation metrics, effective refinement strategies, and the need for visualization and explainability are highlighted. An assessment of existing approaches reveals research gaps, showing that current solutions only partially address these challenges. This work outlines promising directions for future research, aiming towards comprehensive, robust, efficient, and explainable automated preprocessing to enhance the effectiveness of clustering analyses.

Keywords

Data Preprocessing, Automated Preprocessing, Clustering, Pipeline Optimization, AutoML

1. Introduction

Clustering is a widely used technique for discovering patterns in unlabeled data, thereby supporting decision-making in various fields, including biology, computer vision, marketing, and many others [1]. In recent decades, researchers have proposed a wide range of clustering algorithms, each addressing different aspects of similarity, cluster shapes, scalability, or improving clustering performance in specific domains [1, 2]. These algorithms are typically developed and evaluated on synthetic or selected benchmark datasets that have been cleaned or fitted to the algorithm's assumptions. In contrast, real-world data often shows characteristics that standard algorithms cannot handle directly, or the search for a suitable algorithm is non-trivial. For example, most clustering algorithms cannot be applied when missing values are present. Furthermore, feature scales that differ by orders of magnitude, variables of different types (e.g., numerical and categorical), or high-dimensional spaces can significantly reduce overall cluster quality [3, 4].

To address these characteristics and obtain good clustering results, the data must first be preprocessed (e.g., imputing missing values), an important part in the KDD-Process [5] and a step that has repeatedly been shown to improve clustering performance [3, 4, 6]. Depending on the experience and domain knowledge of data analysts, identifying relevant data characteristics and defining a suitable pipeline for preprocessing can be a challenging task that may result in a time-consuming process. In the context of clustering analyses, analysts require guidance in this process.

In recent years, such guidance has been studied as AutoML [7, 8, 9, 10]. In these approaches, the goal is to support data analysts in finding appropriate clustering algorithms and their parameters. In a similar way, automating data preprocessing in analytics pipelines would support data analysts in their decision-making and reduce the trial-and-error search they typically face. However, current AutoML approaches for unsupervised learning often neglect data preprocessing and do not address the additional or different challenges that occur in this context.

36th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), September 29 - October 01, 2025, Regensburg, Germany

leonard.labes@ipvs.uni-stuttgart.de (L. Labes)

https://www.ipvs.uni-stuttgart.de/institute/team/Labes/ (L. Labes)

0000-0001-8672-8972 (L. Labes)

© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2025-12-03

CEUR-WS.org/Vol-4118/paper10.pdf

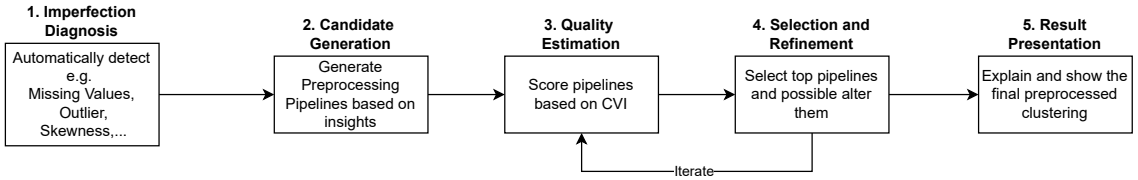

Figure 1: Abstract Process describing the Different Steps of Automated Preprocessing for Clustering

While an automated approach for preprocessing may identify some characteristics, such as missing values, relatively easily, more complex decisions, such as determining a suitable target dimension for a dimensionality reduction algorithm, are difficult to define automatically. Hence, identifying the underlying data characteristics is a primary challenge for an automated approach.

Even with the characteristics identified, an automated approach needs to include one or several appropriate methods to clean the dataset. Often, there are multiple methods available for the same purpose, e.g., multiple dimensionality reduction methods. These methods have parameters and may be interdependent. So, defining a search space and selecting a suitable set of preprocessing steps, as well as their order and parameters, is a challenging task.

A further challenge is based on the fact that clustering is inherently unsupervised. Therefore, no ground truth is available, and internal Cluster Validity Indices (CVIs) are needed to evaluate the clustering results. However, no single metric is always the right choice [7], making it challenging to determine whether a preprocessing pipeline actually improves the quality of the clustering.

Lastly, practical constraints such as runtime and scalability need to be factored in. High-quality clustering results are not always the only part to consider. They must be balanced with efficiency, especially when working with large or complex datasets. All of these uncertainties contribute to making the automation of the preprocessing phase challenging for unsupervised learning.

To make important challenges of automated preprocessing for unsupervised clustering explicit, this paper (i) formulates a five-stage conceptual process covering imperfection diagnosis through result presentation, (ii) distills five overarching research challenges from this process, (iii) compares existing work against these challenges, and (iv) sketches directions for future solutions.

The remainder of this paper is structured as follows. Section 2 describes the proposed process, outlines what an automated approach needs to handle, and identifies challenges for actual implementations. In Section 3, existing solutions are assessed with respect to these challenges to illustrate the state of the art in this field. Section 4 outlines future directions and describes general techniques that can be used to further develop automated preprocessing for clustering. Finally, Section 5 concludes this work.

2. Conceptual Process for Automated Preprocessing

The proposed process is divided into five sequential steps, as shown in Figure 1. Starting with the diagnosis of data imperfections, we proceed to generating candidates for preprocessing pipelines, followed by quality estimation, selection, and refinement stages, and finally presenting the results. Compared to the Figure 1, existing work can repeat steps differently or several times and merge or split them. However, this process provides a mental model for better defining the challenges of an automated approach.

2.1. Step 1: Imperfection Diagnosis

The initial step is to identify the relevant data characteristics. This step should automatically inspect the raw data and report any existing problems. Examples that could be covered include missing-value handling, outlier detection, feature scaling, duplicate-feature removal, and transformations. It is essential to identify such potential issues, as without this knowledge, the subsequent steps of the process become significantly more challenging. The complexity of this step depends on the data characteristics

that need to be identified and can therefore range from a straightforward detection mechanism to complex data analysis.

Challenges: The primary challenge for the identification of all data characteristics is that each characteristic requires a different detection method. Simple general rules can detect certain characteristics in all kinds of data, but more advanced methods often depend on the domain, structure, or distribution of the data. Examples of rule checks include the detection of missing values, whereas detecting outliers is a more complex task that may involve statistical measures.

Furthermore, the features to treat and those not to treat may be problem-dependent, as there may be cases where outliers should be retained, e.g., if they are interesting for machine learning tasks. In other cases, outliers reduce the clustering quality and should therefore be removed. Lastly, there are many options to detect characteristics. For outlier detection alone, various methods exist, and the results produced vary depending on the method and parameter choices [11]. Therefore, the diversity of available methods and parameter settings yields a large search space of detection techniques, which is difficult to constrain given the challenges mentioned above.

2.2. Step 2: Candidate Generation

The goal of this step is to provide a first candidate set of preprocessing pipelines that could address the characteristics identified in Step 1. For example, if missing values are present and possible outliers are detected, the set of candidates includes multiple strategies for imputing missing values or removing outliers. Such candidates can range from a single method that resolves one problem, like removing outliers, to a multi-step pipeline that addresses several characteristics consecutively, e.g., first imputing missing values and then scaling the features.

Challenges: The main challenge in this step is to balance the pipelines in terms of diversity, addressing characteristics, considering efficiency, and applicability, thus shaping the search space. To make this space manageable, constraints, such as limits on pipeline length, definitions on how often a method may recur, and explicit exclusions of incompatible method combinations must be introduced. With these constraints defined, a candidate set that is broad enough to cover every relevant characteristic, small enough to keep the downstream evaluation efficient, and diverse enough to leave multiple options for later improvement instead of leading the optimization into a single local optimum of the space is needed. Finding the right balance among breadth, compactness, and diversity while considering the defined constraints remains an open question.

2.3. Step 3: Quality Estimation

In Step 3, the candidates are evaluated, i.e., the improvement of the cluster quality is assessed. Because clustering tasks have no ground truth, internal CVIs are used to validate the clustering result [12].

Challenges: The core challenge here is to assess the quality using internal CVIs. Internal CVIs measure cluster quality differently, and when choosing one, such as the silhouette score [13] or the Davies-Bouldin Index [14], the risk exists that the preprocessing is optimized towards this heuristic metric rather than the actual clustering quality. At the same time, an exhaustive evaluation with multiple CVIs might be cost-intensive.

2.4. Step 4: Selection and Refinement

Next, the best-performing pipelines need to be selected, considering the results from the previous quality estimation step. While the focus of Step 3 is only on quality, this step may also consider different aspects, such as the efficiency of the pipeline. Depending on the specific approach, the best performing pipelines are then either refined and the process goes back to Step 3 for further quality assessment or the refinement part is finished and the process continues with Step 5 (cf. Figure 1). The goal of the refinement is to improve the already promising pipelines. The details of how this is done depend on the specific approach, but it typically involves generating new pipelines by modifying the existing

Step	C1 Diverse Techniques	C2 Search Space Explosion	C3 Evaluation Metric	C4 Refinement Strategy	C5 Visualization & Explainability
1: Imperfection Detection	✓	✓			
2: Candidate Generation	✓	✓			
3: Quality Estimation			✓		
4: Refinement	✓	✓		✓	
5: Presentation					✓

Table 1
Mapping Between Steps and Derived Challenges

ones. Examples of this include adding, removing, or replacing methods, changing parameter values, or altering the sequence of the pipeline.

Challenges: The difficulty in this step is steering the search towards improved clustering quality without refining a vast number of candidates. Intuitively, pipelines that indicate promising improvements from the previous step should be considered for further refinement. However, an approach should still consider new or more drastic changes to avoid being trapped in a local optimum. This case is typically defined as an exploration versus exploitation dilemma [15] and is challenging to balance. Multi-objective trade-offs, such as balancing quality and runtime, further complicate the selection. Deciding how to weigh runtime and quality is far from clear, and it may be use case dependent. Consequently, the refinement stage must explore an even larger search space of possible pipeline configurations than the initial candidate generation step (Step 2). To let the search space not become too large, it is critical to decide how many best-performing pipelines should be considered in this step.

2.5. Step 5: Present the Results

This step concludes the process and consists mainly of two parts. The first part presents the user with the final result of the clustering, which can range from displaying the achieved improvement and final scores to a visualization of the identified clusters. Secondly, explainability is essential. If such an automated preprocessing approach is applied in the real world, it is not only interesting to see what scores were achieved, but also how they were achieved. The explainability should include the identified characteristics from Step 1 and the applied transformations during preprocessing.

Challenges: This step faces two challenges, first, visualizing the final result to a user. Visualization of results is a general challenge in machine learning [16], but it becomes more complicated when multiple quality measures are considered. Showing only one CVI or an oversimplified version of the data may hide important information. Secondly, the explainability is essential. Telling the user why and how the improvement has been achieved in a comprehensive but understandable way is not straightforward.

2.6. Summary and Derived Challenges

From the challenges described for each step, five main challenges were identified to summarize the process and ease comparison with existing work. Since some challenges can occur in more than one step, Table 1 provides an overview of which challenges are relevant for which step. An automated preprocessing solution for clustering should solve:

C1 Diverse Characteristic Detection and Preprocessing Techniques (Steps 1,2, and 4): Automated preprocessing must detect a wide range of data characteristics (e.g., missing values, outliers, skewness) using different methods for each type of characteristic. Each issue may require specialized detection logic, and the domain or task context can influence whether an imperfection should be corrected or left in place. This heterogeneity makes a comprehensive diagnosis difficult. The same holds for preprocessing. A diverse set of techniques must be available to clean the detected characteristics.

C2 Search Space Explosion (Step 1, 2, and 4): There are many possible methods and parameter settings for detecting data issues, leading to an enormous combinatorial search space. For instance, dozens of outlier-detection algorithms exist, and each may yield different results. These options create a huge search space of detection techniques.

Exhaustively defining and exploring this space is infeasible and complicates the diagnosis step. Furthermore, generating and refining candidate preprocessing pipelines requires a balance between diversity and tractability. The candidate set must cover all identified data issues in multiple ways, yet remain small enough to evaluate later. In practice, generating many diverse pipelines can improve the chances of finding a good solution, but it also risks a combinatorial explosion and high computational cost. To illustrate this better, consider a search space with only three methods, each with two parameters, and each parameter can have five discrete values, therefore having $3 \times 5^2 = 75$ choices per step. With repeats allowed and the maximal length of a pipeline l is three, we would have $\sum_l^3 75^l = 427,575$ possible candidates, and usually there are more methods, more parameters, and more parameter values.

C3 Evaluation Metric (Step 3): Without ground truth (especially in clustering tasks), quality must be assessed by internal CVIs. Relying on a single index (e.g., Silhouette Score [13]) risks optimizing preprocessing towards that surrogate, rather than true clustering quality. Evaluating multiple CVIs can mitigate this bias, but it is computationally expensive. Balancing metric choice and evaluation cost is, therefore, a challenge.

C4 Refinement Search Strategy (Step 4): After selecting promising pipelines, the approach must decide how to refine them. Creating a detailed approach that covers many possibilities is very challenging. Furthermore, there is an exploration-exploitation dilemma[15]: one can either make small (exploitative) tweaks to top pipelines or try entirely new pipeline variations. Finding the right balance is difficult. At the same time, multi-objective trade-offs (improving quality vs. keeping runtime low) further complicate the selection of which pipelines to refine.

C5 Visualization and Explainability (Step 5): The final preprocessing results must be conveyed effectively to users, often involving multiple quality measures. Visualizing improvements and clusterings is non-trivial, especially when showing more than one metric. An oversimplified view (e.g., a single CVI) may hide important details. Designing visual summaries that capture multi-criteria results without overwhelming the user is a significant challenge. Additionally, the user does not only need to see if there is an improvement, but also why. This should include the answers to the questions of why a specific preprocessing technique was selected and how it has improved the clustering quality.

3. Systematic Comparison of Related Work

Section 2 describes the automation of steps in the data analysis process. Existing research examines various aspects of the process from different angles, addressing individual points in distinct ways. For a better overview, this section is grouped into the following parts: AutoML was initially developed for supervised learning but has now also been extended to unsupervised learning, thereby demonstrating automated processes in the context of unsupervised learning (Section 3.1). There is also research that has already dealt with automated preprocessing, but not for unsupervised learning (Section 3.2). Some of these research proposals have also been integrated into widely available libraries, thus serving as the ”default” automation toolkits (Section 3.3). In addition, there are tools that provide information on transformation steps, focusing on visualization and explainability (Section 3.4). In the following, the approaches in each of these categories are briefly described and the extent to which they address the challenges outlined in Section 2.6 is evaluated. Section 3 shows an overview of this evaluation.

3.1. Unsupervised AutoML Approaches

Several approaches address automated clustering [7, 9, 17]. While these approaches have a strong model focus rather than a preprocessing focus, they provide solutions for automatically evaluating clustering results and how optimization in the context of unsupervised machine learning can be accomplished.

Paper	C1 Diverse Techniques	C2 Search Space Explosion	C3 Evaluation Metric	C4 Refinement Strategy	C5 Visualization & Explainability
ML2DAC [7]	○	○	●	○	○
AutoClust [9]	○	○	◑	○	○
TPE-AutoClust [17]	◑	◑	◑	◑	○
Saga [18]	◑	●	○	●	○
CtxPipe [19]	●	◑	○	●	○
LLMClean [20]	◑	○	○	○	◑
Auto-Weka [21, 22]	○	◑	○	○	○
Auto-Sklearn [23, 24]	○	◑	○	○	○
AutoGluon [25]	○	◑	○	○	○
TPOT [26]	◑	◑	○	◑	○
Wrangler [27]	○	○	○	○	◑
Vizier [28]	○	○	○	○	◑

Table 2
Assessment of Related Work with Respect to the Identified Challenges (○ not addressed, ◑ limited addressed, ● completely addressed)

ML2DAC addresses the Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem. It uses a knowledge base, meta-learning, and a trained classifier to suggest clustering algorithms, their hyperparameters, and a suitable CVI. After that, the selection is then further optimized with a Bayesian optimization. While showing how a dataset can be automatically clustered and evaluated, they leave the preprocessing to the user.

From a broad perspective, AutoClust [9] applies a similar approach to ML2DAC [7]. It uses meta-learning and Bayesian optimization to identify a suitable clustering algorithm and promising parameter values. However, instead of a single CVI, 10 are used. A Multilayer Perceptron Regressor (MLP) is trained and then utilized to predict an external CVI based on these 10 internal CVIs to overcome the absence of ground truth, but still have a single metric that is similar to a ground truth evaluation. This predicted score is then used for the optimization. Also, as with ML2DAC, no preprocessing is performed during optimization.

TPE-AutoClust [17] utilizes a knowledge base and meta-learning as well, but chooses an evolutionary algorithm to optimize the suggested pipelines and further applies a majority vote of three CVIs to evaluate the pipelines. In contrast to the other approaches, TPE-Autoclust applies preprocessing during the optimization process. However, the considered preprocessing is limited to four functions from the areas of missing value imputation, dimensionality reduction, and scaling. For these four functions, a limited number of parameters is considered.

Most research in AutoML for clustering focuses on the clustering algorithms, their parameter, and the evaluation metric [7, 9], and only partially considers preprocessing [17]. Therefore, ML2Dac [7] and AutoClust [9] do not cover **C1, C2, C4** and **C5** from a preprocessing perspective, but showcase how unsupervised learning can be evaluated, thus addressing **C3**. ML2DAC [7] completely addresses **C3** efficiently because it picks one CVI based on meta-learning and then only uses this CVI during optimization. AutoClust [29] only partially addresses **C3** because its strategy includes calculating multiple CVIs during the optimization process and therefore neglects efficiency [7]. TPE-AutoClust is the only approach that includes preprocessing, but only in a limited way. The restriction to only a few methods and few parameters per method addresses **C1** only in a limited way, and the question of whether this approach can be applied to a larger, more diverse search space remains unanswered, thus only partially addressing **C2** and **C4**. Additionally, their choice of the majority voting of CVI also includes a limited number of CVIs, and always considering every CVI may not be the best way of evaluating clustering [7, 9], therefore only partially addressing **C3**. Since none of the solutions focus on explainability, none addresses **C5**.

3.2. Data Preprocessing for Supervised Machine Learning

Besides unsupervised AutoML, some approaches solely focus on supervised learning or data cleaning in general, but address preprocessing more specifically than the unsupervised approaches.

Saga [18] is a general-purpose data cleaning framework. It identifies suitable data preprocessing pipelines for a given dataset through a two-step optimization process. First, an evolutionary algorithm is used to find methods assembled into a pipeline. In a second step, the Hyperband algorithm [30] is used to further fine-tune the parameters of the pipelines identified in the first step. Saga can apply several preprocessing functions from the area of outlier removal, missing value imputation, encoding, and string handling [18].

CtxPipe [19] additionally considers the context of the data. It uses pretrained embedding models to capture the context or domain-specific information of the data. Then it uses a deep reinforcement learning approach to construct a pipeline based on embeddings of statistical features and the extracted context. CtxPipe can apply data preprocessing methods for missing value imputation, encoding, scaling, and feature selection, but does not consider the parameters of these functions.

LLMClean [20] technically does not perform preprocessing but uses Large Language Models (LLMs) to construct a context model with Ontology Functional Dependencies (OFDs) to detect errors and repair options in IoT data, but includes a generalization for all types of data. It can detect missing values, look up additional information such as minimum and maximum values from external sources, and check for violations of functional dependencies.

Saga and CtxPipe do not explicitly detect specific characteristics, but discover many characteristics during their optimization. Several techniques of Saga require labels, thus only partially addressing **C1**, while CtxPipe only contains methods that are also applicable to unsupervised learning, therefore addressing **C1** completely. LLMClean focuses more on violations within the data and not classical preprocessing, thus it also addresses **C1** only in a limited way. Due to the design of Saga, **C2** is addressed thoroughly. CtxPipe limits the search space since it does not optimize parameters, resulting in a limited addressing of **C2**. Furthermore, LLMClean provides no concrete pipeline optimization, which means it does not cover **C2**. **C3** is not addressed at all, because all approaches are designed and evaluated with supervised machine learning algorithms. Saga and CtxPipe provide extensive tuning of the pipelines and therefore fully address **C4**. Due to its design, LLMClean does not address **C4**. The focus of Saga and CtxPipe is not on visualization or explainability, as both primarily provide methods for finding and optimizing preprocessing pipelines. Thus, they do not address **C5**. LLMClean provides the user with a context model, which displays derived rules, but the process by which these rules are obtained is not explainable. Additionally, there is no visualization, so **C5** is only addressed in a limited way.

3.3. Libraries

Several libraries have emerged from research and focus on AutoML, including preprocessing to a certain extent. Auto-Weka [21, 22], Auto-Sklearn [23, 24], TPOT [26], and AutoGluon [25] are considered here as prominent examples to evaluate how well-established libraries optimize machine learning pipelines.

Auto-Weka [21, 22] is one of the first libraries in the context of AutoML. It utilizes Bayesian optimization to optimize the selection of machine learning algorithms, along with their parameters, for supervised machine learning tasks. The only preprocessing step Auto-Weka considers during its optimization is automatic feature selection.

Auto-Sklearn [23, 24] utilizes Bayesian optimization as well and is developed on top of the well-known scikit-learn project [31] using its functions to optimize classification or regression tasks. Auto-Sklearn considers preprocessing in two phases: first, data preprocessing, such as missing value imputation or scaling. Secondly, feature preprocessing, which is primarily a dimensionality reduction method.

In contrast to Auto-Sklearn and Auto-Weka, AutoGluon [25] uses ensemble learning in combination with supervised models to provide a good machine learning result. It includes preprocessing steps such as imputing missing values, encoding, and text expansions. However, this is not applied in the search space, but as a fixed step before starting the ensemble learning.

Similar to the other libraries, TPOT [26] only supports supervised tasks. It uses a genetic tree-based optimization strategy to evolve a population of different pipelines. In contrast to the others, TPOT explicitly includes preprocessing as part of its evolutionary optimization. Additionally, various preprocessing methods from the area of normalization, dimensionality reduction, and feature selection are included, automatically selected, and their parameters are tuned during the optimization process.

Most libraries (Auto-Weka, Auto-Sklearn, AutoGluon) apply different strategies to automate the machine learning tasks, yet primarily focus on the downstream tasks instead of preprocessing [21, 22, 23, 24, 25]. Although they do not address the challenges **C1** and **C3-C5** concerning preprocessing, they may show directions on how large search spaces can be handled, therefore partially contributing to **C2**. TPTO [26] is the only library that considers preprocessing as part of the optimization problem. TPOT includes various preprocessing techniques. However, real-world data may have common characteristics like outlier removal that TPOT does not cover. Additionally, some preprocessing methods are only applicable for supervised machine learning, therefore only partially addressing **C1**. The same holds true for challenges **C2 and C4**, TPOT showcases how this can be accomplished, but for a limited number of characteristics. TPOT does not allow unsupervised evaluation and does not provide a dedicated explanation of why the methods have been applied and what the effect is. Therefore, not addressing **C3** and **C5**.

3.4. Semi-Automated Visualization Tools

There is limited research on automated tools that also prioritize explainability or result visualization. However, several tools exist that provide an interactive way to apply and suggest transformations, covering both these topics.

Wrangler [27] supports users by visualizing the data in a spreadsheet style. It combines user input suggestions of transform types to generate multiple options for how the data can be transformed, and shows the effect. It supports reformatting, splitting or combining columns, imputing missing values, and detecting type anomalies.

Another option is Vizier [28], which combines spreadsheet-style visualization with notebooks for code execution and plots to provide the user multiple options to alter and view the data. It primarily enables the user to apply data preprocessing in an easy-to-understand manner, while also incorporating the work of Yang et al. [32] to perform schema matching, automatically impute missing values, or check constraints. Additionally, Vizier keeps track of every transformation done by the user.

The mentioned work does not focus on automating preprocessing for clustering, but showcases how better explainability and visualization of data could be accomplished. Therefore, **C1-4** are not addressed. **C5** is addressed, but still needs changes to fit the described process of Section 2. Therefore, the solutions are considered as only partially addressing **C5**.

3.5. Summary

Section 3 reviews existing research related to automated data preprocessing for clustering, categorizing it into distinct categories. Each category addresses certain aspects of the challenges outlined in Section 2, yet none comprehensively covers all necessary preprocessing steps and optimization requirements. The detailed evaluation is summarized in Section 3. This highlights research gaps and emphasizes the need for robust solutions that entirely automate preprocessing in an unsupervised clustering scenario.

4. Future Directions of Automated Preprocessing for Clustering

Section 3 showcases approaches, none of which are directly transferable to the described process (cf. Section 2) and automated preprocessing for clustering. However, they hint at directions or techniques that could be applied or modified.

Detecting data characteristics remains challenging. Some approaches utilize rule-based detection or implicit detection through candidate generation or refinement. However, these approaches may not be

easily extended to a large search space, or this may imply a long runtime. Inspired by other AutoML approaches [7, 9, 17], meta-learning is an interesting area since it can utilize knowledge from the past [33], e.g., efficiently detecting characteristics based on previously seen datasets. To apply meta-learning, well-selected meta-features are required, but it is unclear which meta-features should be considered to identify a wide range of characteristics. Another step, especially when incorporating other information such as domain knowledge, would be to use an embedding that represents the data [19]. However, creating such an embedding efficiently is still challenging, and it may lack explainability.

Handling the vast search space (**C2**) and efficiently refining it (**C4**) is a topic that others have considered for different problems (e.g., AutoML for Clustering) [17, 18, 19, 21, 22, 23, 24, 25, 26]. The current approaches show that many optimization techniques exist and should be used instead of an exhaustive search. Various methods like Bayesian optimization [21, 22, 23, 24], genetic algorithms [17, 18, 26], reinforcement learning [18, 19], or even LLMs [20] could be applied, while genetic or evolutionary algorithms seem to be a preferred choice when automating preprocessing, mainly because of their flexibility. Again, this still requires a detailed analysis of how such methods can be employed, and also seems promising in combination with meta-learning to restrict the search space during optimization and start with already promising candidates. LLMs are a widely discussed topic, but they struggle to handle big tabular data [34] and tend to generalize when provided with a dataset. It seems more likely that a good representation of the data and preprocessing pipelines should be developed before providing it to an LLM, or that their use case primarily focuses on the context area and processing other information than the raw data (e.g., domain knowledge). In order to utilize LLMs, a good encoding strategy is still required, but it is not yet available for unsupervised preprocessing.

The evaluation with different approaches, such as utilizing internal CVIs [17, 7] or trying to predict external CVI [9], is considered in the area of AutoML for clustering. Yet, an efficient way to transfer this to preprocessing without drastically increasing the runtime remains an open challenge. An interesting direction to evaluate preprocessing pipelines for clustering would be to use a model that predicts external CVIs or suggests internal CVIs. However, creating such a model is challenging because the data, their characteristics, and consequently the recommendations based on these characteristics change during preprocessing. Simply incorporating a CVI can lead to optimization towards this CVI instead of a promising preprocessing pipeline.

Lastly, many tools demonstrate how visualization and explainability can be achieved [27, 28]. However, they are mostly user-focused or require human input. While helpful information is available in some approaches that automate preprocessing [17, 26, 18], it is not effectively utilized. Existing visualization strategies and concepts can be built upon a solution. This could range from simple result visualization to more extensive analysis, showing what was detected, how a pipeline was derived from an optimization process, and what data changes it results in.

To summarize, while there are many interesting parts and pieces, none of them already fit perfectly for automated preprocessing for clustering and need further research and new or adapted ideas to create a well-working solution.

5. Conclusion

In conclusion, this work addresses the automation of data preprocessing for clustering, highlighting the inherent complexities within unsupervised machine learning scenarios. By proposing a structured conceptual process consisting of imperfection diagnosis, candidate pipeline generation, quality estimation, selection, refinement, and result presentation, the paper systematically outlines the steps and associated challenges involved.

Key identified challenges include the need for diverse preprocessing techniques, the combinatorial explosion of the search space, the selection of appropriate evaluation metrics, refinement strategies, and the requirement for effective visualization and explainability. An assessment of existing approaches reveals that while current research partially addresses these challenges, no existing solution comprehensively addresses all of them.

The clear definition of these challenges and the evaluation of current research not only reveal research gaps but also outline promising future research directions. This contributes to the development of robust, efficient, and explainable automated preprocessing solutions, thereby enhancing the effectiveness of clustering analyses in practice.

Declaration on Generative AI

During the preparation of this work, the author used Grammarly and Writefull to correct grammar and spelling and to apply rephrasing. After using these services, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] A. K. Jain, Data clustering: 50 years beyond k-means, Pattern recognition letters 31 (2010) 651–666.

[2] A. E. Ezugwu, A. M. Ikotun, O. O. Oyelade, L. Abualigah, J. O. Agushaka, C. I. Eke, A. A. Akinyelu, 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 (2022) 104743.

[3] V. V. Baligodugula, F. Amsaad, Unsupervised learning: Comparative analysis of clustering techniques on high-dimensional data, arXiv preprint arXiv:2503.23215 (2025).

[4] I. B. Mohamad, D. Usman, Research article standardization and its effects on k-means clustering algorithm, Res J Appl Sci Eng Technol 6 (2013) 3299–3303.

[5] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery in databases, AI magazine 17 (1996) 37–37.

[6] C. Wongoutong, The impact of neglecting feature scaling in k-means clustering, PloS one 19 (2024) e0310839.

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

[8] Y. Liu, S. Li, W. Tian, Autocluster: Meta-learning based ensemble method for automated unsupervised clustering, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2021, pp. 246–258.

[9] Y. Poulakis, C. Doulkeridis, D. Kyriazis, Autoclust: A framework for automated clustering based on cluster validity indices, in: 2020 IEEE International Conference on Data Mining (ICDM), IEEE, 2020, pp. 1220–1225.

[10] D. Tschechlov, M. Fritz, H. Schwarz, Automl4clust: Efficientautoml forclustering analyses, cit. on (2021) 14.

[11] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, M. E. Houle, On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study, Data mining and knowledge discovery 30 (2016) 891–927.

[12] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu, Understanding of internal clustering validation measures, in: 2010 IEEE international conference on data mining, Ieee, 2010, pp. 911–916.

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

[14] D. L. Davies, D. W. Bouldin, A cluster separation measure, IEEE transactions on pattern analysis and machine intelligence (2009) 224–227.

[15] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM computing surveys (CSUR) 35 (2003) 268–308.

[16] A. Chatzimparmpas, R. M. Martins, I. Jusufi, A. Kerren, A survey of surveys on the use of visualization for interpreting machine learning models, Information Visualization 19 (2020) 207–233.

[17] R. ElShawi, S. Sakr, Tpe-autoclust: A tree-based pipline ensemble framework for automated

clustering, in: 2022 IEEE International Conference on Data Mining Workshops (ICDMW), IEEE, 2022, pp. 1144–1153.
[18] S. Siddiqi, R. Kern, M. Boehm, Saga: a scalable framework for optimizing data cleaning pipelines for machine learning applications, Proceedings of the ACM on Management of Data 1 (2023) 1–26.
[19] H. Gao, S. Cai, T. T. A. Dinh, Z. Huang, B. C. Ooi, Ctxpipe: Context-aware data preparation pipeline construction for machine learning, Proceedings of the ACM on Management of Data 2 (2024) 1–27.
[20] F. Biester, M. Abdelaal, D. Del Gaudio, Llmclean: Context-aware tabular data cleaning via llm-generated ofds, in: European Conference on Advances in Databases and Information Systems, Springer, 2024, pp. 68–78.
[21] C. Thornton, F. Hutter, H. H. Hoos, K. Leyton-Brown, 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, 2013, pp. 847–855.
[22] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, K. Leyton-Brown, Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka, Journal of Machine Learning Research 18 (2017) 1–5.
[23] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, in: Advances in Neural Information Processing Systems 28 (2015), 2015, pp. 2962–2970.
[24] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, F. Hutter, Auto-sklearn 2.0: Hands-free automl via meta-learning, arXiv:2007.04074 [cs.LG] (2020).
[25] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, A. Smola, Autogluon-tabular: Robust and accurate automl for structured data, arXiv preprint arXiv:2003.06505 (2020).
[26] R. S. Olson, J. H. Moore, Tpot: A tree-based pipeline optimization tool for automating machine learning, in: Workshop on automatic machine learning, PMLR, 2016, pp. 66–74.
[27] S. Kandel, A. Paepcke, J. Hellerstein, J. Heer, Wrangler: Interactive visual specification of data transformation scripts, in: Proceedings of the sigchi conference on human factors in computing systems, 2011, pp. 3363–3372.
[28] M. Brachmann, C. Bautista, S. Castelo, S. Feng, J. Freire, B. Glavic, O. Kennedy, H. Müeller, R. Rampin, W. Spoth, et al., Data debugging and exploration with vizier, in: Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 1877–1880.
[29] Y. Poulakis, C. Doulkeridis, D. Kyriazis, A survey on automl methods and systems for clustering, ACM Transactions on Knowledge Discovery from Data 18 (2024) 1–30.
[30] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: A novel bandit-based approach to hyperparameter optimization, Journal of Machine Learning Research 18 (2018) 1–52.
[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.
[32] Y. Yang, N. Meneghetti, R. Fehling, Z. H. Liu, O. Kennedy, Lenses: An on-demand approach to etl, Proceedings of the VLDB Endowment 8 (2015) 1578–1589.
[33] P. Brazdil, J. N. Van Rijn, C. Soares, J. Vanschoren, Metalearning: applications to automated machine learning and data mining, Springer Nature, 2022.
[34] Y. Sui, M. Zhou, M. Zhou, S. Han, D. Zhang, Table meets llm: Can large language models understand structured table data? a benchmark and empirical study, in: Proceedings of the 17th ACM International Conference on Web Search and Data Mining, 2024, pp. 645–654.