Ensemble Clustering based on Meta-Learning and Hyperparameter Optimization

Dennis Treder-Tschechlov
University of Stuttgart
Stuttgart, Germany
Dennis.Tschechlov@ipvs.uni-stuttgart.de

Manuel Fritz
University of Stuttgart
Stuttgart, Germany
Manuel.Fritz@ipvs.uni-stuttgart.de

Holger Schwarz
University of Stuttgart
Stuttgart, Germany
Holger.Schwarz@ipvs.uni-stuttgart.de

Bernhard Mitschang
University of Stuttgart
Stuttgart, Germany
Bernhard.Mitschang@ipvs.uni-stuttgart.de

ABSTRACT

Efficient clustering algorithms, such as k-Means, are often used in practice because they scale well for large datasets. However, they are only able to detect simple data characteristics. Ensemble clustering can overcome this limitation by combining multiple results of efficient algorithms. However, analysts face several challenges when applying ensemble clustering, i. e., analysts struggle to (a) efficiently generate an ensemble and (b) combine the ensemble using a suitable consensus function with a corresponding hyperparameter setting. In this paper, we propose EffEns, an efficient ensemble clustering approach to address these challenges. Our approach relies on meta-learning to learn about dataset characteristics and the correlation between generated base clusterings and the performance of consensus functions. We apply the learned knowledge to generate appropriate ensembles and select a suitable consensus function to combine their results. Further, we use a state-of-the-art optimization technique to tune the hyperparameters of the selected consensus function. Our comprehensive evaluation on synthetic and real-world datasets demonstrates that EffEns significantly outperforms state-of-the-art approaches w.r.t. accuracy and runtime.

PVLDB Reference Format:
Dennis Treder-Tschechlov, Manuel Fritz, Holger Schwarz, and Bernhard Mitschang. Ensemble Clustering based on Meta-Learning and Hyperparameter Optimization. PVLDB, 17(11): 2880 - 2892, 2024.
doi:10.14778/3681954.3681970

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/tschechlovdev/EffEns.

1 INTRODUCTION

Efficient clustering algorithms, e. g., k-center algorithms such as k-Means, are often used in practice because they scale well for large datasets [40, 66]. However, they are only able to detect simple data characteristics and are not able to detect complex ones (e. g., circle-like clusters as shown in Figure 1). In contrast, complex clustering algorithms, e. g., density-based algorithms, are able to detect complex characteristics, but they are not feasible for large datasets due to high runtime complexities [30, 36]. To detect complex data characteristics efficiently, we aim for an ensemble clustering [57] approach by combining multiple results of efficient algorithms.

Ensemble clustering consists of two steps (cf. Figure 1): (1) Ensemble generation, where a large set of base clusterings $\mathcal{E}$ is generated from the given dataset $\mathcal{D}$, and (2) consensus clustering, where a consensus function is used to combine the base clusterings of the ensemble $\mathcal{E}$ into a more accurate final clustering result.

However, ensemble clustering methods are often not established in practice as only experts know how to apply and benefit from them. Therefore, popular machine learning libraries such as scikit-learn [51] do not provide any ensemble clustering methods. Thus, especially novice analysts do not know how to generate an ensemble and how to select a suitable consensus function to achieve an accurate final clustering result, e. g., to detect complex data characteristics such as circle-like clusters. The reason is that analysts face several challenges when applying ensemble clustering.

To obtain an accurate final clustering result, ensemble generation should produce diverse and accurate base clusterings [10, 40, 57, 64]. To achieve diversity, different clustering configurations, i. e., clustering algorithms and hyperparameter settings, have to be executed on $\mathcal{D}$. Yet, it is not clear how to select configurations from the configuration space, i. e., the search space of all possible clustering configurations, to obtain diverse and accurate clustering results. Exploring the whole configuration space is infeasible. Thus, it is not clear how to **generate base clusterings efficiently** (cf. challenge **C1** in Figure 1).

From generated base clusterings, literature typically selects a more rigorous subset – the ensemble $\mathcal{E}$ – based on specific criteria that reflect a trade-off between diversity and accuracy [1, 8, 23, 32, 48, 67]. However, these approaches require the size of the ensemble m as input, i. e., the number of base clusterings in $\mathcal{E}$. Yet, larger ensemble sizes do not guarantee a higher accuracy; on the contrary, larger ensemble sizes often decrease the accuracy of the final clustering result [23, 38]. Therefore, the size of the ensemble is dependent on the characteristics of the data and it is unclear how to **select the size of the ensemble** (cf. **C2** in Figure 1) in advance.

Regarding the consensus step, literature comprises many different consensus functions (e. g., [5, 10, 18, 31, 37, 38, 50, 57–59, 64]).

Proceedings of the VLDB Endowment, Vol. 17, No. 11 ISSN 2150-8097.
doi:10.14778/3681954.3681970

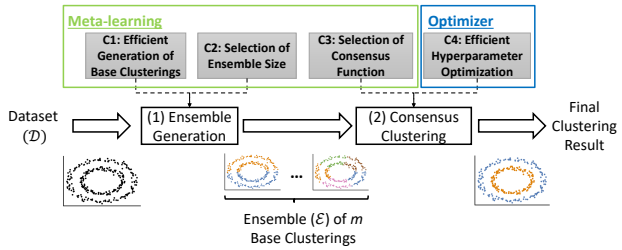


Figure 1: Overview of general steps in ensemble clustering and the challenges (C1 - C4) to achieve accurate results. We use meta-learning to address the challenges C1 - C3 and an optimizer to address challenge C4.

Hence, it is not clear how to **select a consensus function** for a given $\mathcal{D}$ (cf. **C3** in Figure 1).

Furthermore, consensus functions have hyperparameters, e. g., the number of clusters in the final clustering result, which influence the consensus clustering result. Therefore, the **hyperparameters have to be optimized efficiently** (cf. **C4** in Figure 1).

Some of the existing literature on ensemble clustering addresses challenge C1, but none of them focuses on C2 - C4 (cf. Table 1). Existing AutoML for clustering systems use hyperparameter optimizers that can be adapted to optimize hyperparameters of consensus functions (C4). Yet, they are not able to address C1 - C3 (cf. Table 1). Thus, existing approaches only address either challenge C1 or C4.

In this paper, we propose the efficient ensemble clustering approach ***EffEns***. EffEns is the first end-to-end ensemble clustering approach that addresses the challenges C1 - C4 together and is still able to detect complex data characteristics efficiently. Our approach uses meta-learning, which is often referred to as "learning to learn" [13], to address C1 - C3 and adapts existing optimizers to address C4 (cf. Figure 1). Our contributions include the following:

- We use meta-learning to learn which base clustering results are required for a clustering ensemble and which consensus function is suitable w.r.t. the ensemble and the data characteristics. To this end, we learn two classification models based on data characteristics: First, an ensemble generation model, which can directly predict the ensemble without requiring the ensemble size as input, and second, a consensus function model, which predicts a suitable consensus function. Thus, we can address C1 - C3.
- We adapt an existing hyperparameter optimizer to tune the hyperparameters of the selected consensus function (C4).
- In our comprehensive evaluation, we show that EffEns outperforms existing state-of-the-art baselines w.r.t. clustering accuracy and runtime on synthetic and real-world data.

The remainder of this paper is structured as follows: Section 2 formulates the challenges that we address in this paper. We give an overview of related works and their limitations in Section 3. Section 4 presents the intuition of our approach. We describe the details of the learning phase in Section 5, while we detail on the application phase in Section 6. In Section 7, we discuss the results of our evaluation and conclude this paper in Section 8.

2 CHALLENGES AND PROBLEM STATEMENT

Let $\mathcal{D} = \{x_1, ..., x_n\} \in \mathbb{R}^{n \times f}$ be an unseen dataset with n instances and each instance is an f-dimensional feature vector. We define a clustering ensemble as $\mathcal{E} = \{y_1, ..., y_m\}$, while y_i describes a single base clustering result. To generate the base clusterings, different clustering configurations have to be executed. Therefore, we assume a clustering configuration space $\mathcal{CS} = \mathcal{A} \times \mathcal{H}$ that comprises a set of clustering algorithms $\mathcal{A}$ and their hyperparameter values $\mathcal{H}$. We denote a configuration from the configuration space as $c_i = (a, h) \in \mathcal{CS}$. A base clustering is the result of executing c_i on $\mathcal{D}$, i. e., a label-vector $y_i \in \mathbb{N}^n$ that comprises for each data instance of $\mathcal{D}$ the corresponding cluster label. The performance of a base clustering is typically assessed with an internal clustering validity index (*CVI*) [33, 34, 44] that measures the internal structure of a clustering result. A *CVI* is a function $CVI : \mathbb{R}^{n \times f} \times \mathbb{N}^n \rightarrow \mathbb{R}$. It takes as inputs $\mathcal{D}$ and y_i, while the output is a real value. We assume that lower *CVI* values indicate better clustering results. Otherwise, we can multiply its values with -1 to obtain a minimization objective.

A consensus function $CF : \mathbb{N}^{n \times m} \rightarrow \mathbb{N}^n$ is used to combine the clustering results. That is, the consensus function has as input a clustering ensemble $\mathcal{E}$ of size m and combines the result into a single consensus clustering result. Literature comprises many different consensus functions that also have additional hyperparameters. Therefore, we assume a set of consensus functions $\mathcal{CF}$, while each $cf \in \mathcal{CF}$ has its search space of hyperparameter values $\mathcal{H}_{cf}$. Our general goal is to obtain a valuable ensemble clustering result in $O(n)$. Thus, both, the ensemble generation and the consensus step have to be in $O(n)$. For the generation, we can only consider clustering algorithms in $\mathcal{CS}$ with a linear runtime, such as k-center clustering algorithms [4, 40, 46]. Similarly, we only include consensus functions with a linear runtime in $\mathcal{CF}$. Based on these, we address the following challenges regarding ensemble generation and consensus clustering:

C1: Efficient Generation of Base Clusterings. The first challenge is to generate a set of base clusterings $\mathcal{B} \subset \mathcal{B}_{CS}$. Here, $\mathcal{B}_{CS}$ denotes the full set of all possible base clusterings that would be produced when executing all clustering configurations from the configuration space $\mathcal{CS}$. Furthermore, $\mathcal{B}$ should contain diverse and accurate base clusterings. However, even when only considering k-center algorithms, exploring the whole $\mathcal{CS}$ would still be too time-consuming or even infeasible [27–29, 62]. Thus, we aim for a generation strategy that does not require exploring the whole $\mathcal{CS}$, while still generating diverse and accurate clustering results.

C2: Ensemble Selection. From the set of base clustering $\mathcal{B}$, a subset $\mathcal{E}^*_m \subseteq \mathcal{B}$ of m clustering results has to be selected as ensemble that can be used for the consensus step. Literature comprises several methods to achieve such a subset with decent diversity and accuracy at the same time. These existing approaches require the ensemble size m as input. However, selecting m is not trivial as it has a major influence on the consensus step and is dependent on the data characteristics as well as the consensus function. Therefore, we aim for an approach to select the ensemble without specifying m for a new dataset.

C3: Selection of Consensus Function. We aim to select a consensus function $cf \in \mathcal{CF}$ that can achieve accurate clustering results. As literature comprises different consensus functions and

Table 1: Overview of related work for AutoML and ensemble clustering approaches.

Approaches		Complex Data Characteristics	C1: Efficient Ensemble Generation	C2: Selection of Ensemble Size	C3: Selection of Consensus Function	C4: Efficient Hyperparameter Optimization
Ensemble Clustering	Generation of Base Clusterings [10, 26, 58, 60, 64], Ensemble Selection [1, 8, 23, 32, 48, 67]	X	✓	X	X	X
	Consensus Functions [6, 10, 12, 37, 38, 50, 57–59, 64]	✓†	X	X	X	X
AutoML Approaches	AutoML4Clust (AML4C) [62]	X	✓	X	X	✓
	AutoClust [53], AutoCluster [45], CSMartML [20], TPE-AutoClust [21]	✓	X	X	X	✓

† Depending on concrete consensus function as each is able to detect different data characteristics and they have different runtime complexities.

each is better suited for different data characteristics, the selection should take the data characteristics of $\mathcal{D}$ into account. Further, as consensus functions combine the ensemble in different ways, the selection has to be based on the ensemble $\mathcal{E}_m^*$ as well.

C4: Efficient Hyperparameter Optimization. The hyperparameters of consensus functions have a major influence on the final clustering accuracy. For instance, many consensus functions have as hyperparameter the final number of clusters [6, 23, 50, 57, 58]. However, the search space of hyperparameter values for a consensus function $\mathcal{H}_{cf}$ may be large [27, 28, 62] and thus it is crucial to select the hyperparameters efficiently.

In our paper, we address the challenges C1 - C4 by focusing on the following equation:

$$y^* = \underset{cf \in \mathcal{CF},\ h \in \mathcal{H}_{cf},\ \mathcal{E}_m^* \subset \mathcal{B}_{CS}}{arg\,min} CVI(\mathcal{D}, cf(\mathcal{E}_m^*, h)) \quad (1)$$

such that $\forall y_i \in \mathcal{E}_m^* : CVI(\mathcal{D}, y^*) \leq CVI(\mathcal{D}, y_i)$, i. e., the consensus clustering result is more accurate than each individual base clustering in the ensemble.

3 RELATED WORK

We identified two relevant groups of related work: (1) existing approaches to ensemble clustering, which combine multiple configurations to achieve better clustering results, and (2) AutoML approaches for clustering, which find a single best configuration. Table 1 summarizes our key findings regarding related work.

(1) Ensemble Clustering: We discuss approaches for generating base clusterings, ensemble selection, and consensus clustering.

To obtain a large set of base clusterings with diverse clustering results, existing works execute various clustering configurations [1, 10, 32, 64]. Using clustering algorithms with high computational complexity, i. e., $O(n^2)$ or even higher, is not feasible for large datasets [10, 27, 29, 40, 62]. The most often used method in literature is to run k-Means with an exhaustive search [23, 42, 57], i. e., to execute all values for k in a defined search space. This has the benefit of relying on an efficient clustering algorithm and still creating diversity in the ensembles as the results vary in their number of clusters. However, it is not clear in advance how to define such a search space for k. On the one hand, large search spaces may lead to diverse clustering results, but executing all k values in a large search space is, especially for large datasets, infeasible in practice [27, 29, 62]. A small search space may lead to too similar results, so that the consensus is not able to increase the accuracy [57].

From the generated set of base clusterings, literature selects a subset that exhibits a trade-off between diversity and accuracy. This is also known as overproduce and select [23, 32, 42, 48], i. e., all k values in a pre-defined search space are executed to generate the base clusterings and a subset of the base clusterings is selected as the ensemble. Nevertheless, these approaches require the size of the ensemble, i. e., the number of base clusterings, as an input parameter. This ensemble size highly influences the selected ensemble and thus the final consensus clustering result as well. Furthermore, existing approaches on the generation of base clustering and ensemble selection use a manually selected consensus function with its hyperparameter settings. Thus, they do not address C3 and C4.

Literature comprises many different consensus functions that combine an ensemble into a final clustering result [5, 10, 18, 31, 37, 38, 50, 57–59, 64]. These consensus functions vary in how they represent the ensemble, e. g., as label or co-occurence matrix [64], and how they combine the ensemble, e. g., using voting or clustering the ensemble representation [10]. The consensus functions also have hyperparameters, which often include the final number of clusters to produce in the consensus clustering. The consensus functions differ in their runtime complexity, i. e., some have a linear runtime complexity, while others have a quadratic or higher runtime complexity [57, 64]. Furthermore, literature has shown that different consensus functions are better suited to detect different data characteristics [40, 57, 64]. Nevertheless, literature does not comprise an approach that is able to select an appropriate consensus function with different hyperparameter settings for different datasets yet.

(2) AutoML Approaches for Clustering: Approaches of this category address the combined algorithm selection and hyperparameter optimization (CASH) problem for clustering analyses but differ in the way they solve it.

Tschechlov et al. introduce AutoML4Clust (AML4C) [62], an AutoML approach to tackle the CASH problem for clustering that explores the clustering configuration space $\mathcal{CS} = \mathcal{A} \times \mathcal{H}$. As AML4C focuses on k-center algorithms, it can be applied efficiently to large datasets but is also limited to certain data characteristics.

AutoClust [53], AutoCluster [45], and CSmartML [20] use meta-learning to reduce the configuration space to a single algorithm and subsequently optimize its hyperparameters, e. g., using Bayesian Optimization [56]. They rely on meta-features to find datasets with similar characteristics and select a single algorithm that performed best based on similar datasets. However, they may select a complex clustering algorithm and thus, face long runtimes or the selected algorithm is not feasible for large datasets. If they select an efficient algorithm, they can only detect certain data characteristics.

Summary: Summarizing related work (cf. Table 1), existing approaches that can detect complex data characteristics are only able

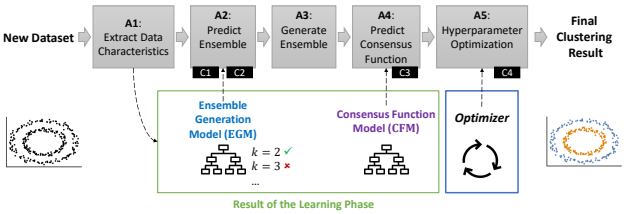


Figure 2: Intuition of the application phase, i. e., how we apply ensemble clustering with steps A1 - A5 on a new dataset.

to address either C1 or C4. Ensemble clustering approaches can generate the ensemble efficiently using efficient clustering algorithms. However, none of them is able to address C2 - C4. AutoML for clustering approaches have to rely on complex clustering algorithms to detect complex data characteristics. Thus, they are not feasible for large datasets. While their hyperparameter optimizers can be used to select the hyperparameters of consensus functions (C4), they do not address C1 - C3 as they do not use ensemble clustering at all.

4 EFFENS: EFFICIENT ENSEMBLE CLUSTERING

In this section, we illustrate the intuition of EffEns. To this end, we first describe its application phase, i. e., how we apply ensemble clustering to new unseen datasets (Section 4.1). This application phase relies on two classification models. In Section 4.2, we describe how we use the learning phase to learn these models.

4.1 Intuition of the Application Phase

Figure 2 shows the general steps of how we cluster new unseen datasets. In step A1, we extract the data characteristics of a new dataset as both generating the ensemble and selecting the consensus function highly depend on these data characteristics. In step A2, we apply an **ensemble generation model (*EGM*)**, which is a classification model that, based on the data characteristics, predicts the clustering configurations to be used for generating the clustering ensemble. Thus, we are able to address the challenges C1 and C2 both together as we directly predict the clustering ensemble and do not have to use an overproduce-and-select approach. Hence, EGM implicitly selects the ensemble size m and in contrast to related work, we do not require m as input for unseen datasets. Step A3 generates the clustering ensemble by executing the predicted *EGM* configurations. We rely on k-Means as a clustering algorithm due to its efficiency and hyperparameter k for diversity. However, our approach is not limited to specific clustering algorithms, i. e., it is possible to use different clustering algorithms and their hyperparameters as well. In step A4, we select a consensus function based on a **consensus function model (*CFM*)**. Similar to *EGM*, this model is trained on data characteristics and predicts a consensus function for the new dataset. Thus, we are able to address challenge C3. Step A5 optimizes the hyperparameters of the selected consensus function. For this, we use efficient **hyperparameter optimization techniques** that are commonly used in existing AutoML systems [25, 45, 53, 62] and therefore address challenge C4. As result, we obtain a final clustering result.

Nevertheless, the main question remains: how do we obtain the models *EGM* and *CFM*? In the following, we describe the intuition of our prior learning phase that produces these models.

4.2 Intuition of the Learning Phase

The learning phase of our approach is executed only once before, resulting in the ensemble generation model (*EGM*) and the consensus function model (*CFM*). Figure 3 gives an intuition of how we obtain our models. First, we assume that we have access to a dataset repository $\mathcal{DR}$ with several datasets that exhibit varying characteristics. In our evaluation (cf. Section 7), we use synthetically generated data as this makes it easy to obtain data with varying characteristics in a controlled manner. Yet, such datasets could also be taken from existing machine learning repositories such as OpenML [63] or the UCI Machine Learning Repository [19]. For these datasets, ground-truth clustering results are available and thus we can use them in our learning phase. Nevertheless, a few questions arise on how to acquire the relevant training data and the relevant classification labels for the two models. In particular, we have to answer the following questions (cf. Figure 3):

Q1: How to measure data characteristics? Our models should predict the ensemble and the consensus function based on data characteristics. To measure data characteristics, we extract so-called meta-features [54]. These *Meta-Features* (cf. Figure 3) numerically describe different characteristics of a dataset, e. g., the number of instances (n) or standard deviation of feature values (std).

Q2: How to measure the performance of clustering ensembles and consensus functions? After applying consensus clustering, we obtain a final clustering result. Yet, the performance of the consensus step is dependent on the clustering ensemble and thus we have to measure the performance of both simultaneously. To this end, we assume a repository *Evaluated Ensembles* (cf. Figure 3) that contains different settings of clustering ensembles and consensus functions. As mentioned previously, we know the ground-truth clustering, i. e., the clustering label for each data instance, in each already clustered dataset. Hence, we can use an external clustering validity index to compare the clustering result with ground-truth clustering [34]. Figure 3 shows the Adjusted Mutual Information (AMI) [55, 65] being used for that purpose. Based on the AMI, we can decide for each dataset which setting of clustering ensemble and consensus function leads to the best final clustering result.

Q3: How to select the consensus function? As we know which consensus function performs best on a dataset, we can train a multi-class single-label classification model that predicts for each dataset a suitable consensus function. We dub this model the consensus function model (CFM). As the consensus function is dependent on data characteristics, we use meta-features as input data and the selected consensus function as the target label (cf. *Training Data for CFM* in Figure 3). For a new dataset, we have to extract the same meta-features and can then predict a suitable consensus function.

Q4: How to generate the clustering ensemble? Similar to CFM, we also use the meta-features as input for our ensemble generation model (EGM). EGM is a binary multi-label classification

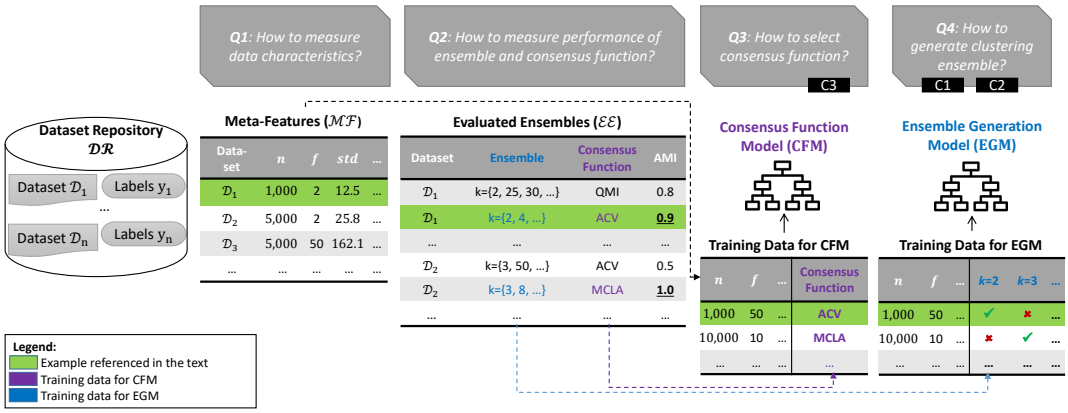


Figure 3: Intuition of the learning phase.

model [35] that predicts for each clustering configuration whether its result should be used in the ensemble or not. For example, EGM predicts whether the result of k-Means with $k = 2$, $k = 3$, etc should be in the clustering ensemble. For that purpose, we transform the ensemble from *Evaluated Ensembles* via one-hot encoding and use the encoded clustering configurations as binary multi-label target columns (cf. *Training Data for EGM* in Figure 3).

In Section 5, we discuss how we obtain the training data systematically and how we derive the models EGM and CFM.

5 LEARNING PHASE

In this section, we describe the steps of the learning phase in more detail, in particular, how we obtain training data for our CFM and EGM models. As we execute the learning phase once, we may also tolerate a longer runtime than in the application phase. Once we trained the models, we can apply them to any unseen dataset.

As shown in Figure 3, our learning phase relies on a dataset repository, which contains multiple datasets. We define it as $\mathcal{DR} = \{(\mathcal{D}_1, y_1^{GT}), ..., (\mathcal{D}_o, y_o^{GT})\}$, where each dataset $\mathcal{D} \in \mathcal{DR}$ may have varying numbers of instances and features. To learn the performance of the ensemble w.r.t. the subsequent consensus step, we require information about ground-truth clustering. Therefore, the datasets in the dataset repository also contain the ground-truth clustering, i. e., the clustering label for each data instance. For each dataset $\mathcal{D}$, we denote its ground-truth clustering labels as y^{GT}.

Algorithm 1 shows the five steps L1 - L5 of our learning phase. In the following, we explain the details of each of these steps.

5.1 Extract Meta-Features

In this step L1 (line 2 in Algorithm 1), we learn the characteristics of different datasets such that we can compute the similarity of datasets based on these characteristics. Therefore, we compute for each dataset meta-features $\{mf_1, ..., mf_r\}$ that capture these characteristics and store the results in the set $\mathcal{MF} = \{(\mathcal{D}, mf_1(D), ..., mf_r(\mathcal{D}))|\mathcal{D} \in \mathcal{DR}\}$ (line 10). Each meta-feature mf_i describes a function $mf_i : \mathcal{DR} \rightarrow \mathbb{R}$, i. e., it assigns to a dataset $\mathcal{D} \in \mathcal{DR}$ a real value. The meta-features can be general meta-features such as the number of instances or features [54], statistical ones as the mean or standard deviation [24], based on information theory [54], or even more complex ones such as landmarking [45], where a clustering algorithm is applied to the dataset and the results are used as meta-features. The choice of the meta-feature set is crucial for a precise representation of data characteristics.

In a previous work [61], we evaluated different meta-feature sets, including meta-feature sets from existing AutoML for clustering systems. A combination of statistical and general meta-feature sets achieved the best results [61]. Therefore, we adopt this set of 40 meta-features that include amongst others the number of instances n, features f, and the standard deviation std (cf. Figure 3).

5.2 Generate Base Clusterings

In step L2, we generate the base clusterings (lines 12 - 22). For each dataset $\mathcal{D}$ from $\mathcal{DR}$ and each k-value from a given range of k-values $\mathcal{H}$, we execute k-Means on $\mathcal{D}$ (line 16). The result is a label-vector y that assigns a clustering label to each data instance from $\mathcal{D}$. In line 17, we calculate the AMI values ami of each clustering result w.r.t. to the ground-truth y^{GT}. We require this information for step L3 to select ensembles with high accuracy. We store the information $(\mathcal{D}, y, ami, k)$ in the set $\mathcal{B}$, so that we know the clustering result y_i, the accuracy of each clustering result ami, on which dataset $\mathcal{D}$ we have obtained it, and which clustering configuration k we have used to obtain the result (line 18). The result of this step is the set of base clusterings $\mathcal{B}$ (line 21).

5.3 Evaluate Ensembles

In step L3, we evaluate the performance of different ensemble subsets of the set of base clusterings w.r.t. different consensus functions. The procedure EVALUATE_ENSEMBLES(...) shows the details of this step. The inputs for this procedure are the dataset repository $\mathcal{DR}$, the set of base clusterings $\mathcal{B}$, and a set of ensemble sizes $\mathcal{M}$ that defines the different ensemble subset sizes that we evaluate. In our evaluation, we use $\mathcal{M} = \{5, 10, 15, ..., 50\}$. Note that we require $\mathcal{M}$ only in the learning phase, while our EGM model implicitly selects the ensemble size in the application phase for new unseen datasets. First, we iterate over the datasets and their ground-truth labels $(\mathcal{D}, y^{GT}) \in \mathcal{DR}$ (line 25), the different ensemble sizes $m \in \mathcal{M}$ (line 26), and different strategies to select the ensemble subsets (line 27). To this end, we use the two most-prominent approaches from literature [23, 32, 48]:

Algorithm 1 Algorithm for the learning phase.

Input: $\mathcal{DR}$: Dataset repository, $\mathcal{H}$: Hyperparameter search space for the ensemble generation, $\mathcal{M}$: Set of possible ensemble size values, $\mathcal{CF}$: Consensus functions
Output: CFM: Consensus function model, EGM: Ensemble Generation Model.

```
1: procedure Learning_Phase(DR, H, M, CF)
2:     MF ← Extract_Meta_Features(DR);                          ▷ L1
3:     B ← Generate_Base_Clusterings(DR, H);                    ▷ L2
4:     EE ← Evaluate_Ensembles(DR, B, M, CF);                   ▷ L3
5:     CF_best, E_best ← Select_Best_CF_And_Ensemble(EE);        ▷ L4
6:     CFM, EGM ← Train_CFM_And_EGM(MF, CF_best, E_best);       ▷ L5
7:     return CFM, EGM;
8: end procedure

9: procedure Extract_Meta_Features(DR)                          ▷ L1 Details
10:    return {(D, mf_1(D), mf_2(D), ..., mf_r(D)) | D ∈ DR};
11: end procedure

12: procedure Generate_Base_Clusterings(DR, H)                  ▷ L2 Details
13:    B ← ∅;
14:    for (D, y^GT) ∈ DR do
15:        for k ∈ H do
16:            y ← Execute_KMeans(D, k);
17:            ami ← AMI(y, y^GT);
18:            B ← B ∪ (D, y, ami, k);
19:        end for
20:    end for
21:    return B;
22: end procedure

23: procedure Evaluate_Ensembles(DR, B, M, CF)                  ▷ L3 Details
24:    EE ← ∅
25:    for (D, y^GT) ∈ DR do
26:        for m ∈ M do
27:            for Ens_Select ∈ {Quality_Select, Cluster_and_Select} do
28:                B_D ← Get_B_For_D(B, D);
29:                E_m ← Ens_Select(B_D, m);
                   ▷ Evaluate consensus functions
30:                for cf ∈ CF do
31:                    k* ← Unique_Labels(y^GT);
32:                    y ← cf(E_m, k*);
                       ▷ Evaluate with ground-truth y^GT
33:                    ami ← AMI(y, y^GT);
34:                    EE ← EE ∪ {(D, cf, ami, E_m)};
35:                end for
36:            end for
37:        end for
38:    end for
39:    return EE;
40: end procedure

41: procedure Select_Best_CF_and_Ensemble(EE)                   ▷ L4 Details
42:    E_best ← ∅; CF_best ← ∅;
43:    for D ∈ DR do
44:        E_D, cf_D ← select ensemble E_D and cf with best AMI value;
45:        E_best ← E_best ∪ {(D, E_D)};
46:        CF_best ← CF_best ∪ {(D, cf_D)};
47:    end for
48:    return E_best, CF_best;
49: end procedure

50: procedure Train_CF_And_EGM(MF, E_best, CF_best)             ▷ L5 Details
51:    CFM ← Train classification model on MF with labels CF_best;
52:    E_best ← One_Hot_Encoding(E_best);
53:    EGM ← Train classification model on MF with labels E_best;
54:    return CFM, EGM;
55: end procedure
```

(i) Quality_Select, i. e., we select the m clustering results from the ensemble with the highest accuracy.

(ii) Cluster_and_Select, i. e., we cluster all base clusterings in $\mathcal{B}$ by their similarity and select from each cluster the base clustering with the highest accuracy. For more details of this selection strategy, we refer to the work of Fern and Lin [23].

In line 28, we retrieve the set of base clusterings $\mathcal{B}_{\mathcal{D}}$ that we generated for dataset $\mathcal{D}$. Subsequently, we apply the ensemble selection strategy to obtain the selected ensemble $\mathcal{E}_m$ with size m (line 29). In lines 30 - 36, we use the selected ensemble to evaluate different consensus functions. As we know the ground-truth clustering labels, we also know the actual number of clusters k^* of each dataset (line 31). Then, we execute cf with k^* as hyperparameter on the selected ensemble $\mathcal{E}_m$ to obtain the clustering labels y (line 32). To measure the performance of the consensus configuration, we calculate the AMI value using the consensus clustering labels y and the ground-truth labels y^{GT} (line 33). In line 34, we store the information $(\mathcal{D}, cf, ami, \mathcal{E}_m)$ in the set $\mathcal{EE}$. Finally, we return the set of evaluated ensembles $\mathcal{EE}$ (line 39).

Figure 3 shows examples of entries in $\mathcal{EE}$. For instance, on dataset $\mathcal{D}_1$ (green color in Figure 3) with 1,000 instances, 2 features and a standard deviation of 12.5, we select the clustering results of k-Means with $k = 2, 4, \ldots$ from the set of base clustering $\mathcal{B}$ to form our ensemble. Then, we executed the ACV [6] consensus function to combine the ensemble into a single clustering result and achieved an AMI score of 90% (the best score for $\mathcal{D}_1$ in $\mathcal{EE}$).

5.4 Select Best CF and Ensemble

In step L4, we select the best-performing combination of consensus function (CF) and ensemble. First, we iterate over each dataset (line 43). Then, we actually select the best ensemble and the best consensus function for the dataset $\mathcal{D}$ (line 44). As selection criteria, we use the AMI values, i. e., we select for each dataset the ensemble and the consensus function with the best AMI values. Then, we store these best ensembles and the best consensus functions (lines 45 and 46). Finally, we return the set of best ensembles $\mathcal{E}_{best}$ and best consensus function $\mathcal{CF}_{best}$ (line 48).

To illustrate this step: on dataset $\mathcal{D}_1$ (highlighted in green in Figure 3), combining the ensemble of clustering results from k-Means with $k = 2, 4, \ldots$ using the consensus function ACV achieved the best AMI score of 90%. Thus, one entry in *Training Data for CFM* (cf. Figure 3), i. e., $\mathcal{CF}_{best}$ in Algorithm 1, has the meta-feature values of $\mathcal{D}_1$ as features and the consensus function ACV as target class label. In a similar way, for *Training Data for EGM*, we have the meta-feature values of $\mathcal{D}_1$ as features and labels that indicate for each k-value whether it is in the best ensemble or not. So, in this example, we have $k = 2$, $k = 4$, etc in the ensemble for $\mathcal{D}_1$.

5.5 Train CFM and EGM

In the last step, we use the collected data to train our CFM and EGM models (line 6). That is, we use for both models the meta-features as input data. For CFM, we use the selected consensus functions for the respective dataset as target column to train a multi-class classification model (line 51). The set of best ensembles $\mathcal{E}_{best}$ however may contain different k-values. Therefore, we first have to transform the ensemble in a way that we can use it as multi-label classification target. To this end, in line 52, we apply one-hot-encoding on $\mathcal{E}_{best}$ so that we have one column for each k-value and the values of the rows are either 1 (k-value is in the ensemble) or 0 (not in the ensemble). Subsequently, we can train

Algorithm 2 Algorithm for the application phase.

Input: $\mathcal{D}_{new}$: New unseen dataset, b: Budget, CVI: Clustering validity index, $\mathcal{H}_{cf}$: Hyperparameter search space, CFM: Consensus function model trained in our learning phase, EGM: Ensemble generation model trained in our learning phase.
Output: y^*: Final clustering result.

1: **procedure** APPLICATION_PHASE($\mathcal{D}_{new}$, b, CVI, $\mathcal{H}$, CFM, EGM)
▷ Initialization
2: $cvi^* \leftarrow \infty$; $y^* \leftarrow \emptyset$; $\mathcal{E} \leftarrow \emptyset$
▷ **A1: Extract Meta-Features**
3: $\mathcal{MF}_{new} \leftarrow$EXTRACT_META_FEATURES($\mathcal{DR}$)
▷ **A2: Predict Clustering Ensemble**
4: $\mathcal{K} \leftarrow EGM$.PREDICT_ENSEMBLE($\mathcal{MF}_{new}$)
▷ **A3: Generate Clustering Ensemble**
5: **for** $k \in \mathcal{K}$ **do**
6: $y \leftarrow$ KMEANS($\mathcal{D}_{new}, k$)
7: $\mathcal{E} \leftarrow \mathcal{E} \cup \{y\}$
8: **end for**
▷ **A4: Predict Consensus Function**
9: $cf \leftarrow CFM$.PREDICT_CONSENSUS_FUNCTION($\mathcal{MF}_{new}$)
▷ **A5: Optimize Hyperparameters**
10: $O \leftarrow$ INIT_OPTIMIZER($\{cf\}, \mathcal{H}_{cf}$)
11: **for** $i = 1, ..., b$ **do**
12: $cc_i \leftarrow O$.SELECT_CONFIGURATION()
13: $y_i \leftarrow cc_i(\mathcal{E})$
14: $cvi_i \leftarrow$ CVI($\mathcal{D}, y_i$)
15: **if** $cvi_i < cvi^*$ **then**
16: $cvi^* \leftarrow cvi_i$
17: $y^* \leftarrow y_i$
18: **end if**
19: O.UPDATE(cc_i, cvi_i)
20: **end for**
21: **return** y^*
22: **end procedure**

the EGM model (line 53). For any new dataset, we can then use its meta-features as input to EGM and CFM to predict the consensus function and the clustering ensemble.

For CFM, we could use any kind of classification model. However, as we have only a limited amount of training data (we have as much training data as the number of datasets in $\mathcal{DR}$), we use a robust classification model, e. g., Random Forest [14]. For EGM, we have only limited training data as well and also require a model that supports multi-label classification. Random Forest can also be applied to multi-label classification problems by using so-called label power set transformation [11, 35], which transforms the multi-label problem into a multi-class problem using power sets. Thereby, we can also learn correlations between the classification labels, e. g., $k = 2$ is often used in an ensemble, but only in combination with $k = 50$. In line 54, we return both models, which are also the final results of the overall learning phase.

6 APPLICATION PHASE

In this section, we detail on the steps of the application phase of our approach. Figure 2 shows its main steps. Similar to existing AutoML for clustering approaches that also rely on optimization techniques, our approach has as inputs: (i) $\mathcal{D}_{new} \subset \mathbb{R}^n$, a new dataset with f features that is intended to be clustered, (ii) CVI, which is an internal clustering validity index to evaluate clustering results, (iii) b, which is a budget and defines the number of consensus configurations to execute, and (iv) $\mathcal{H}_{cf}$, which is the search space of hyperparameter values for consensus clustering. Furthermore, we also use the results of our learning phase, which are the consensus function model CFM and the ensemble generation model EGM.

In the following, we describe the steps of our application phase in more detail on the basis of Algorithm 2. We start by initializing the variables cvi^*, which tracks the best evaluated CVI value, y^*, which tracks the best clustering result that obtains the best CVI value after consensus clustering, and $\mathcal{E}$, which tracks the ensemble that we use for the consensus step (line 2). We then perform the following five steps:

(A1) Extract Meta-Features: We measure data characteristics of $\mathcal{D}_{new}$ by extracting its meta-feature values (line 3). The extracted meta-feature values serve as inputs for our classification models. Note that we extract the same set of meta-features as in the learning phase; otherwise, we cannot apply our learned models.

(A2) Predict Clustering Ensemble: We apply our EGM model to the meta-features of the new dataset to obtain a set of clustering configurations (line 4). In our work, we focus on k-Means and the hyperparameter k. Thus, we predict a set of k-values.

(A3) Generate Clustering Ensemble: In lines 5 - 8, we generate the clustering ensemble. For each k value that we predicted in step (A2), we execute the k-Means algorithm with the corresponding k value to obtain one clustering result (line 6). In line 7, we add this clustering result to the ensemble. The result is the ensemble $\mathcal{E}$.

(A4) Predict Consensus Function: We apply our CFM model to the meta-features of the new dataset to predict a suitable consensus function based on the data characteristics (line 9).

(A5) Optimize Hyperparameters: In lines 10 -21, we optimize the hyperparameters of the selected consensus function. To this end, we initialize the optimizer O with the consensus function cf and the search space of its hyperparameter values $\mathcal{H}_{cf}$ that we want to optimize (line 10). We use Bayesian optimization (BO) as an optimizer as it trades off exploration and exploitation, i. e., we exploit well-performing configurations on the one side, but also explore new regions of configurations on the other side. BO is used in many existing AutoML systems for supervised learning tasks [25], but also in AutoML systems for clustering analyses [53, 62]. Subsequently, we execute b loops of the optimization procedure (lines 11 - 18). In each loop, we first select a consensus configuration cc_i, i. e., the values for the hyperparameters of the consensus function, using the optimizer O (line 12). Then, we execute the consensus configuration on the ensemble $\mathcal{E}$ to obtain a single clustering result y_i (line 13). Further, we evaluate the result with a cluster validity index CVI, e. g., Calinski-Harabasz [15] or Davies Bouldin [17], and obtain the corresponding value cvi_i (line 14). In lines 15 - 18, if the current CVI value cvi_i is lower than cvi^*, we change the current best CVI value and also update the current best clustering result y^*. Subsequently, we update the optimizer with the consensus configuration cc_i and its corresponding CVI value cvi_i (line 19). Finally, we return the best consensus clustering result y^* that we have obtained during the b optimizer loops (line 21).

7 EVALUATION

In this section, we evaluate our approach with different experiments. To this end, we first describe the setup of our experiments (cf. Section 7.1). Subsequently, we compare our approach against state-of-the-art baselines on synthetic datasets (cf. Section 7.2). Finally, we show the feasibility of our approach on real-world benchmark data for clustering analysis (cf. Section 7.3).

7.1 Setup

Hard- and Software: Our implementation is based on Python 3.9 and is available as an anonymous GitHub repository[1]. We implemented the optimizer using SMAC [39, 43], a sequential model-based optimizer. A large part of the implementation is based on the scikit-learn library [51]. We used the PyMFE [2] package to extract in total all 40 meta-features of the groups general and statistical[2]. We conducted our experiments on a VM with Ubuntu 20.04, a 2.6 GHz processor, 16 CPUs, and 32 GB RAM.

Synthetic Datasets: We used data generators from scikit-learn[3] to generate 78 synthetic datasets. These include data with four different characteristics: (i) Gaussian datasets, where all clusters have the same standard deviation, (ii) Varied Gaussian, where the clusters have varying standard deviations, (iii) Circles, where the clusters follow a circle-like structure, and (iv) Half-Moons, where the clusters have the structure of Half-Moons. For all of them, we vary the number of instances $n \in \{1{,}000; 10{,}000; 50{,}000\}$. For Gaussian and for Varied Gaussian, we vary the number of features $f \in \{50; 75; 100\}$ and clusters $k \in \{10; 30; 50\}$. Regarding Circles and Half-Moons, the implementation of scikit-learn supports only $f = k = 2$, but therefore we vary the ratio of noise $r \in \{0; 0.01; 0.05; 0.1\}$.

Consensus Functions: As mentioned in Section 2, we focus on consensus functions with linear runtime complexity. We have identified which consensus functions meet this requirement and can be used with a custom ensemble generation method. Therefore, we use the following five consensus functions: Mixture Models (MM) [59], Meta-Clustering Algorithm (MCLA) [57], Adaptive cVote (ACV) [6], Adaptive bVote (ABV) [7], and Quadratic Mutual Information (QMI) [58]. They all have as hyperparameters the final number of clusters in the consensus clustering and we use $\{2, ..., 100\}$ as hyperparameter space.

Learning Phase: In our learning phase, we generate different clustering ensembles. To this end, we focus on the k-Means algorithm and use $\{2, ..., 100\}$ as hyperparameter space for the hyperparameter k of k-Means. We also evaluated other k-center clustering algorithms such as GMM [9] for the generation but did not find any improvements in the consensus clustering. Regarding the ensemble selection, we evaluate two strategies, Quality_Select and Cluster_and_Select, as described in Section 5. We exploit for each strategy the ensemble sizes $\mathcal{M} = \{5, 10, 15, ..., 50\}$.

Application Phase: We split our 78 synthetic datasets into 54 (70%) training and 24 (30%) test datasets. We ensure that we have the same distribution of data characteristics in both, training and test datasets. Thus, we use the knowledge from the 54 training datasets to train our EGM and CFM models and apply them to the 24 test datasets. We use Random Forest [14] as it can handle multiple classes and is robust regarding small training data. More details can be found in our repository. In the application phase, we use different cluster validity indices (CVIs) and report the results for the best CVI. We execute $b = 70$ optimizer loops of Bayesian optimization. To evaluate the accuracy, we use the adjusted mutual information (AMI) index, which is adjusted by chance [65]. Note that we have also evaluated the Adjusted Rand Index (ARI) but we do not present its results as they are largely identical to the results using AMI. The ARI results can be found in our GitHub repository. We perform 3 runs of our approach and report average results if not stated otherwise.

[1]Prototypical implementation: https://github.com/tschechlovdev/EffEns

[2]Description of Meta-Features: https://pymfe.readthedocs.io/en/latest/auto_pages/meta_features_description.html

[3]Cf. data generators 'make_blobs' for (varied) Gaussian, 'make_moons' and 'make_circles': https://scikit-learn.org/stable/datasets/sample_generators

7.2 Comparison on Synthetic Data

In this section, we compare our approach against state-of-the-art baselines on the synthetically generated datasets.

7.2.1 State-of-the-art Baselines. First, we compare our approach against state-of-the-art AutoML systems for clustering analyses that return the results of a single clustering algorithm. These AutoML systems also use more complex clustering algorithms such as density-based [22], hierarchy-based [41], or spectral clustering algorithms [36, 49]. For the comparison, we use the following six automated clustering approaches, including four approaches based on existing AutoML systems, one automatic density-based approach, and one automated ensemble clustering approach:

AML4C^k [62]: AutoML4Clust (AML4C), which is an AutoML system for clustering that uses Bayesian Optimization to tackle the combined search space of clustering algorithms and hyperparameters. It relies only on k-center clustering algorithms k-Means, Gaussian Mixture Models, and MiniBatch k-Means.

AML4C^A [61, 62]: Treder-Tschechlov et al. extend AML4C to include nine available clustering algorithms from scikit-learn, which include amongst others density-based, hierarchy-based and spectral clustering algorithms.

AutoClust [53]: AutoClust is an AutoML for clustering system that first uses meta-learning to select a clustering algorithm for a new dataset and subsequently, optimizes its hyperparameters using Bayesian Optimization. For a new dataset, Poulakis et al. first apply the Meanshift [16] algorithm and then calculate the scores of multiple cluster validity indices. Based on these meta-features, they search for the 10 most-similar datasets from the training phase and select the algorithm that performed best on most of these similar datasets. We implemented AutoClust ourselves as the authors do not provide a publicly available implementation.

AS→HPO (inspired by [20, 45, 53, 61]): The evaluation of Treder-Tschechlov et al. unveils that a combination of statistical and general meta-features performs best [61]. Thus, we adapt the AutoClust baseline to use this set of meta-features as another baseline.

OPTICS [3]: We also use an automatic density-based approach as a baseline, which is known to detect characteristics such as Circles or Half-Moons that can not be detected by k-center algorithms. Since it is a single algorithm, it does not depend on an optimizer and therefore, does not have such a large search space as AML4C^A. We use the implementation from scikit-learn.

AEC: As literature does not comprise an end-to-end automated ensemble clustering approach (cf. Section 3), we also use a simple, but novel baseline that we call the automatic ensemble clustering baseline. For this baseline, we first generate an exhaustive ensemble, i. e., running k-Means with $k = 2, ..., 100$ as this is typically done by literature to generate a clustering ensemble [23, 57, 58, 60]. For ensemble selection, we use the cluster-and-select (CAS) method as this is often used in literature and makes a trade-off between accuracy and diversity [23, 32, 42, 64]. Subsequently, we use Bayesian

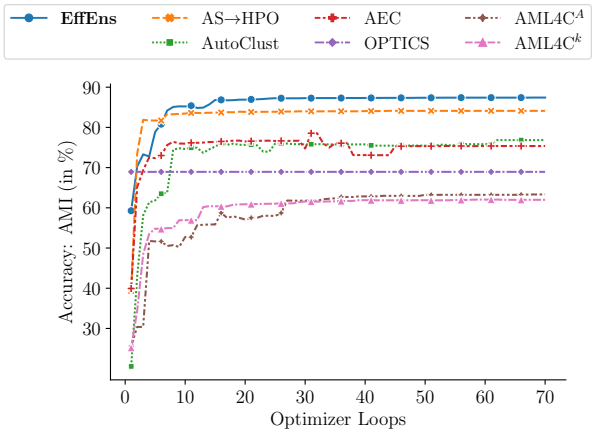


Figure 4: Averaged clustering accuracy results w.r.t AMI over all datasets for our approach EffEns and the baselines.

Optimization to optimize (i) the ensemble size, (ii) the consensus function, and (iii) its hyperparameters simultaneously. That is, the search space of the optimizer is $\mathcal{CF} \times \mathcal{M} \times \mathcal{H}_{\mathcal{CF}}$.

7.2.2 Clustering Accuracy Comparison. The results regarding accuracy are shown in Figure 4. We make the following observations:

(1) EffEns achieves the highest accuracy results compared to the state-of-the-art baselines. It requires only around 15 optimizer loops to achieve these results. The main reason is that EffEns has a small search space in comparison to most of the baselines. In our approach, the optimizer only has to optimize the final number of clusters in the consensus, while the baselines have multiple clustering algorithms and/or they have more complex hyperparameters in the search space. EffEns achieves the highest accuracy, i. e., AMI values, on 20 of the 24 datasets. Taking the highest AMI value as rank 1, then EffEns has an average rank over all 24 datasets of 1.4. In contrast, the best baseline has an average rank of 2.6.

(2) The best baseline is AS→HPO. It is able to reduce the search space and select suitable clustering algorithms depending on data characteristics. While it has a smaller search space compared to baselines using all clustering algorithms, it is still able to find accurate results for most datasets within the budget. However, it requires to execute complex clustering algorithms that have high runtime complexities. Interestingly, although EffEns only uses k-Means for the ensemble generation and consensus functions with linear runtime, it still achieves more accurate results than AS→HPO.

(3) AML4C^A and AML4C^k are the baselines with the worst accuracy results. The reason for AML4C^A is its large configuration space, in particular, many of the more complex clustering algorithms run into the timeout on large datasets. As the results for AS→HPO show, there are single clustering results that are much better, but AML4C^A is not able to find them within the budget. As we show in the next subsection, AML4C^k is only able to detect certain (Gaussian) cluster characteristics. OPTICS is able to achieve more accurate results than AML4C^A and AML4C^k as it is a density-based clustering algorithm. Thus, it can also detect more complex cluster characteristics than AML4C^k and does not have such a large search space as AML4C^A.

(4) AEC is also able to achieve more accurate results than a more complex density-based algorithm such as OPTICS. However, AEC is not able to achieve such accurate results as our approach. The main reason is that AEC is not able to generate the ensemble effectively with respect to the consensus functions. That is, AEC relies on the CAS ensemble selection method, while EffEns generates the ensemble based on the data characteristics and also considers which ensemble performs well for a consensus function.

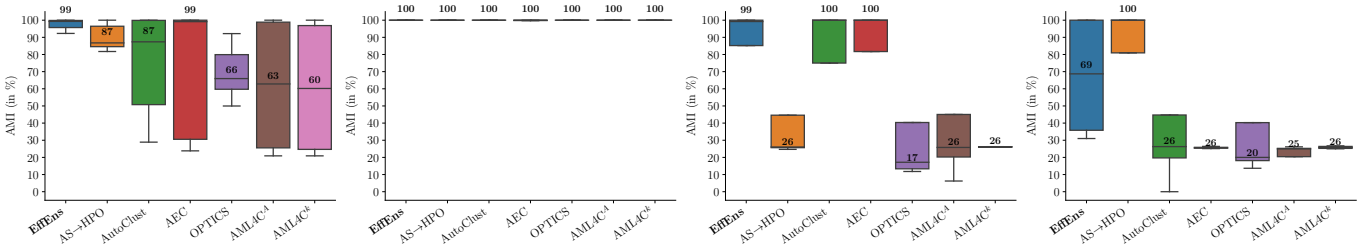
7.2.3 Accuracy Comparison for Different Data Characteristics. Figure 5 shows the accuracy comparison regarding the different data characteristics. We observe:

(1) EffEns achieves the highest accuracy w.r.t. AMI on Varied Gaussian, Gaussian, and Half-Moons, i. e., at least 99% accuracy. In particular, EffEns can detect the Half-Moons characteristic very well by only relying on k-Means for the ensemble generation and consensus functions with linear runtime complexity. Thereby, it even outperforms baselines with more complex algorithms on the Half-Moons data. We found that one reason is that EffEns is more robust regarding noise. For the highest noise ratio of 10%, we achieve an accuracy of 65%, while the best baseline only achieves 20%. Thus, ensembles seem to be very effective for cases with complex characteristics, e. g., Half-Moons data with high noise ratios (cf. Figure 5c) or with varying Gaussian distributions in the data (cf. Figure 5a).

(2) The most accurate baselines on the different data characteristics are AS→HPO and AEC. Although AS→HPO uses even more complex clustering algorithms, it achieves more accurate results than EffEns only on the Circles data. Further, EffEns outperforms AS→HPO on the Varied Gaussian and Half-Moons data. AEC achieves similar results to EffEns, except for the Circles data, where EffEns outperforms AEC because it can generate the ensemble more effectively based on the data characteristics. Further, as we show in the next subsection, EffEns is much more efficient as AEC, because it can directly generate the ensemble and does not require to select the ensemble in each optimizer loop. Thus, EffEns is more robust regarding different data characteristics than existing state-of-the-art baselines. It can be more effective than more complex clustering algorithms or an automated ensemble clustering approach, especially on complex data characteristics. Nevertheless, the results also show that there is room for improvement, in particular regarding the Circles data. For these datasets, we have datasets with high noise ratios in the test set. Complex clustering algorithms are able to handle such noisy datasets slightly better than ensembles of efficient clustering algorithms. Future work could explore how diversity can be explicitly used in the ensemble generation to further improve accuracy in such cases, e. g., using random subsets of the data [52].

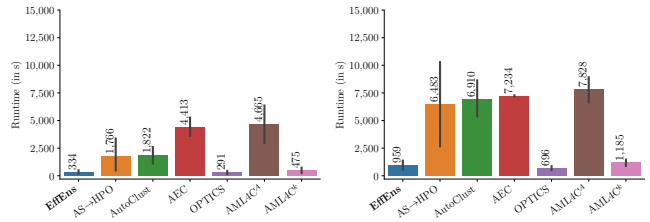
7.2.4 Runtime. We compare the runtime when applying EffEns and the baselines to new datasets. Figure 6 shows the average runtimes on all datasets and on the large datasets. We observe:

(1) In Figure 6, EffEns is shown with an average runtime of 334 seconds across all datasets (cf. Figure 6a) and 959 seconds on average considering only the large datasets (cf. Figure 6b). EffEns is faster than all of the baselines except for OPTICS, which however has worse accuracy results (cf. Figure 4). In comparison to



(a) Varied Gaussian (b) Gaussian (c) Half-Moons (d) Circles

Figure 5: Clustering accuracy, i. e., AMI values, for each data characteristic separately. Black values present median values.



(a) All datasets (b) Large datasets

Figure 6: Average runtime results on (a) all datasets and (b) only the large datasets (n = 50,000) for executing 70 loops. Grey vertical lines show the standard deviation w.r.t. runtime.

AS→HPO, the baseline with the highest average accuracy results, EffEns achieves speedups of more than 6x on large datasets (cf. Figure 6b). In particular, AS→HPO requires 1.8 hours, while EffEns requires only 16 minutes. The reason for the higher runtime of AS→HPO is that it can select a more complex clustering algorithm, e. g., DBSCAN, with quadratic or higher runtime complexity. This shows that using ensembles relying on k-Means and efficient consensus functions is more efficient than using complex clustering algorithms, while achieving even higher accuracy (cf. Section 7.2). Compared to AML4C^A, EffEns has speedups of 13.9x and 8x (cf. Figure 6a and 6b). Executing only 15 loops already leads to accurate clustering results (cf. Section 7.2.2) with an average runtime of 68 seconds for all datasets (190 seconds for large datasets), i. e., achieving even more speedups.

(2) Compared to AEC, EffEns achieves speedups of 13x and 7.5x (cf. Figure 6a and 6b). Although AEC also relies on k-Means for the ensemble generation and consensus functions with linear runtime, it has to execute the ensemble selection, i. e., cluster-and-select (CAS), in each optimizer loop, so that it can optimize the ensemble size. CAS has to compute all pairwise similarities of the base clusterings and executes a clustering algorithm on it (cf. line 60 in Algorithm 1), which is the main bottleneck of AEC.

7.2.5 Scalability. In the following, we investigate the scalability of EffEns in contrast to the baselines. To generate large-scale datasets, we vary the number of instances from $n = 1,000$ by a factor of 10 up to $n = 1,000,000$. We generated data with Varied Gaussian characteristics, so that we can generate high-dimensional datasets with $f = 100$ features. This allows for more meaningful observations on

Table 2: Scalability results for Varied Gaussian datasets by varying the number of instances n. We execute 50 loops and set a timeout of 6 hours.

	Runtime (in s) for ...			
Approach	$n = 1,000$	$n = 10,000$	$n = 100,000$	$n = 1,000,000$
EffEns	6	**37**	**93**	**971**
AML4C^A	**3**	174	13,402	21,600$^{\dagger 12}$
AML4C^k	12	39	3,058	21,600$^{\dagger 14}$
AEC	1,180	2,995	21,600$^{\dagger 38}$	21,600$^{\dagger 9}$
AS→HPO	7	622	13,021	21,600$^{\dagger 22}$
OPTICS	**3**	77	3,321	194,044
AutoClust	7	909	21,600$^{\dagger 8}$	21,600$^{\dagger 5}$

† Shows at which optimizer loop an approach reached the timeout of 6 hours.

the scalability of the approaches. Nevertheless, we observed similar trends on data with Circles and Half-Moons characteristics. For approaches that use an optimizer, we execute 50 optimizer loops and abort earlier if they require more than 6 hours (21,600 seconds).

The results in Table 2 show that the runtime of EffEns increases linearly for increasing values of n, thus demonstrating its linear runtime complexity. In particular, for large datasets ($n \geq 10,000$) we achieve significant speedups compared to the baselines. For the largest dataset, i. e., $n = 1,000,000$, we have a runtime of less than 1,000 seconds (< 17 minutes), while even the fastest baseline runs into the timeout of 21,600 seconds (360 minutes), i. e., EffEns achieves a speedup of more than 20x compared to the fastest baseline. However, the fastest baseline only executes 22 optimizer loops. Therefore, the speedups would be even more severe if the baselines executed all 50 loops. In particular, EffEns is the only approach that executes all 50 optimizer loops for the largest dataset. In contrast to OPTICS, the baseline with the highest runtime of 194,044 seconds for $n = 1,000,000$, EffEns achieves a speedup of more than 194x. Overall, EffEns is more scalable and much faster than baseline approaches on large datasets.

Note that EffEns has very high accuracy values even for the large datasets (AMI ≈ 99%), while especially approaches with fewer optimizer loops, e. g., AEC and AutoClust, have significantly lower accuracy values for large datasets.

7.2.6 Ablation Study. In this subsection, we verify the effectiveness of our three main components: The ensemble generation model

Table 3: Results of the ablation study, i. e., accuracy achieved when activating (✓) / deactivating (X) selected components.

Approach	EGM	CFM	Optimizer	AMI (in %)
ALL	✓	✓	✓	**88.8**
No EGM	X	✓	✓	61.9
No CFM (ABV)	✓	X	✓	67.3
No Optimizer	✓	✓	X	65.0
Baseline: **Best Base Clustering**	✓	X	X	62.6

Table 4: Overview of real-world datasets.

Dataset	n	f	k^*
Iris	150	4	3
Ecoli	336	8	8
Dermatology	366	33	6
Wdbc	569	31	2
Banknote	1,372	4	2
Pendigits	10,992	16	10
USPS	11,000	256	10
Letter	20,000	16	26
F-MNIST-{10; ...; 70}k	{10k; 20k; ..., 70k}	784	10

*We use the classification labels as cluster labels.

(EGM), the consensus function model (CFM), and the optimizer. We study the effects when not using one of our components, but instead a common technique for clustering or ensemble clustering. To this end, we compare the following five ablations of our approach:

(1) ALL: We use all three components of our approach, i. e., EGM, CFM, and the optimizer. **(2) No EGM:** We only apply CFM and the optimizer. For the generation of the base clusterings, we do an exhaustive generation, i. e., we execute k-Means for $k = 2, 3, ..., 100$. For the ensemble selection, we use the quality-based technique described in Section 5. However, we still have to select the ensemble size m for the selection strategy. Thus, we extend the consensus search space of the optimizer to also include m. **(3) No CFM (ABV):** We do not use CFM, but only EGM and the optimizer. Instead of selecting a consensus function, we apply the same consensus function for each dataset. We report the results for the ABV consensus function ABV as it achieved the best average results in our experiments. **(4) No Optimizer:** We only use CFM and EGM, i.e., we do not use the optimizer, but instead use the k value from the best base clustering result of the generated ensemble. **(5) Best Base Clustering:** This constitutes a simple baseline, where we only apply EGM to generate the ensemble and subsequently, return the best single clustering result.

The accuracy results of our ablation study on synthetic datasets are shown in Table 3. We make the following observations:

First, using all three components achieves the best result of 88.8%, which can also be seen in Figure 4. This shows that all of our components are crucial to obtain valuable ensemble clustering results. The results also show that leaving out only one of our components already leads to an accuracy loss of over 20%-points. Furthermore, we achieve an improvement of 26.2%-points over the best base clustering result from the generated ensemble.

Second, not using EGM has the worst accuracy of 61.9%. In this case, we optimize the ensemble size with the optimizer and thus have a very large search space ($m \times k$) which makes it more difficult to achieve valuable results in a short time frame. In contrast, our EGM model predicts the ensemble dependent on the data characteristics and on the selected consensus function for the dataset.

Third, not using CFM only achieves an accuracy of 67.3%, which is 21.5%-points lower than when all components are used. The main reason is that the consensus functions can detect one or two of the data characteristics very well, e. g., Gaussian and varied Gaussian, but fail to detect other characteristics such as Circles or Half-Moons.

Fourth, not using an optimizer achieves an accuracy of 65% when selecting the best k value from the generated clustering ensemble to set the final number of clusters in the consensus clustering.

7.3 Results on Real-world Data

In this section, we evaluate our approach on unseen real-world data, i. e., while trained on synthetic datasets, we apply EffEns to real-world data that comprise varying distributions compared to the synthetic datasets. To this end, we use 9 datasets that are often used in literature as benchmark datasets to test the effectiveness of novel clustering or ensemble algorithms [36, 38, 47, 53]. They are taken from the UCI ML Repository [19] or OpenML [63]. Table 4 shows how the 9 datasets vary regarding n, f, and k. Note that these datasets are actually designed for classification tasks and thus, we can use the class labels as cluster labels to evaluate clustering accuracy. For the dermatology dataset, we remove the feature *age* as it is incomplete. For the Fashion-MNIST dataset (*F-MNIST*), we take random subsamples of size 10k, 20k, ..., 70k (original dataset), while preserving the same distribution of clusters in the data. Thus, we obtain a total of 15 real-world benchmark datasets.

7.3.1 Learning Phase. In the following, we report the runtime that the meta-learning approaches require. For the offline learning phase, we use all 78 synthetic datasets (as described in Section 7.1) for the meta-learning approaches of EffEns, AutoClust, and AS→HPO . The learning phase of EffEns using all synthetic datasets took 35 hours (≈ 1.5 days). In contrast, AS→HPO took 62 hours (≈ 2.6 days) and AutoClust took overall 84 hours (≈ 3.5 days). Thus, our learning phase is two days faster than AutoClust and one day faster than AS→HPO. Note that the baseline approaches use an optimizer to traverse the search space and still have a higher runtime than EffEns for the learning phase. When using larger datasets in the learning phase, such as in Section 7.2.5, the overhead for the learning phase of AutoClust and AS→HPO would be even more severe.

7.3.2 Accuracy Comparison. Table 5 shows the results of EffEns and the baselines on the real-world datasets. As on synthetic data, EffEns achieves the most accurate on real-world data. On average, EffEns achieves 55.4%, which is an improvement of 9.3 (AML4C^A), 10.4 (AML4C^k), 14.5 (AEC), 37.5 (AS→ HPO), and 44.1%-points (OPTICS). EffEns also has the highest minimum (35.9%) and maximum

Table 5: Results on real-world data in comparison to state-of-the-art baselines.

Approach	AMI (in %)			Runtime (s)	
	Min.	Avg. (±std)	Max.	Avg.	Max.
EffEns	**35.9**	**55.4** (±12.8)	**77.1**	**1,120**	**4,155**
AML4C^A	1.7	46.1 (±18.6)	73.2	10,675	21,600
AML4C^k	18.8	45.0 (±18.3)	73.2	7,173	21,600
AEC	16.6	42.9 (±16.6)	73.2	3,642	7,170
AS→HPO	0.0	19.8 (±26.2)	65.6	9,383	21,600
OPTICS	1.3	11.3 (±12.2)	37.7	1,295	6,520
AutoClust	0.0	12.3 (±21.9)	63.0	5,778	21,600

AMI values (77.1%). In particular, the minimum of 35.9% is 17%-points higher than for the best baseline (18.8% for AML4C^A). Thus, EffEns can achieve significant improvements on datasets where the baseline approaches achieve low accuracy values. Regarding the standard deviation, EffEns has the second-lowest one with 12.8%. OPTICS is the only baseline with a lower standard deviation, but it also has low minimum and maximum accuracy values (cf. Table 5).

On real-world datasets, k-center algorithms obtain decently accurate results. Therefore, AML4C^A and AML4C^k are the baselines with the highest accuracy values on average. AS→HPO performs worse on real-world data than on synthetic data. It is not able to select a suitable algorithm for the unseen real-world data and therefore achieves minimum AMI values of 0% (cf. Table 5).

EffEns achieves the most accurate results on 10 of the 15 datasets and has an average rank of 1.6. In contrast, the best baselines, AML4C^k and AML4C^A, have an average rank of 2.7. Hence, EffEns outperforms the baselines on most of the datasets.

7.3.3 Runtime Comparison. On the real-world datasets, our approach is the fastest w.r.t. to the average and maximum runtimes (cf. Table 5). The F-MNIST datasets have large feature sets comprising 784 features. This affects the runtime of most baseline approaches. Therefore, the baseline approaches AML4C^k, AML4C^A, AutoClust, and AS→HPO run into the timeout of 21,600 seconds for the largest dataset (cf. Table 5). For EffEns, the large feature set is not that severe, because the consensus functions are applied on the clustering results. Therefore, the runtimes of consensus functions depend on the size of the ensemble and not on the number of features.

7.3.4 Complexity Analysis. We focus the complexity analysis on the number of instances n in a dataset and on the application phase, as this is crucial to obtain valuable clustering results in a short time-frame. Our application phase has five subsequent steps and therefore, the runtime complexity is the same as for the step with the highest complexity. For extracting the meta-features (cf. A1 in Algorithm 2), the runtime complexity is $O(n)$ as we only use general and statistical meta-features [54]. Predicting the ensemble (A2) and the consensus function (A4) using EGM and CFM is not dependent on n and can thus be done in $O(1)$. For the ensemble generation (A3), we rely on k-Means as clustering algorithm to generate the ensemble. In the worst case, we have to execute k-Means for the whole hyperparameter space $\mathcal{H}$, i. e., $|\mathcal{H}|$ times. As k-Means has a linear runtime complexity and $|\mathcal{H}| << n$, the complexity for A3 is $O(n * |\mathcal{H}|) = O(n)$. For A5, the runtime complexity for the consensus function is $O(n)$ as we only use linear consensus functions and executing the function b times also results in $O(n)$. Hence, the overall runtime complexity for the application phase is $O(n) + O(1) + O(n) + O(1) + O(n) = O(n)$.

7.4 Evaluation Summary

In the following, we summarize the main findings (F1 - F5) of our comprehensive evaluation:

F1: On synthetic datasets, EffEns obtains more accurate results than state-of-the-art baselines that use even more complex clustering algorithms (cf. Section 7.2.2).

F2: EffEns achieves the most robust results regarding different data characteristics (cf. Section 7.2.3). Although it relies on efficient clustering algorithms and consensus functions, it can still consider complex data characteristics such as Half-Moons.

F3: In the application phase, EffEns achieves significant speedups compared to state-of-the-art baselines (cf. Section 7.2.4). For datasets with up to 1 million instances, it achieves speedups of 20x to 194x (cf. Section 7.2.5). The learning phase is also much faster (≈ twice as fast) than that of baseline approaches (cf. Section 7.3.1).

F4: We show that all three components (EGM, CFM, and the optimizer) are crucial to obtain valuable ensemble clustering results (cf. Section 7.2.6). Leaving out only one of these components already leads to an accuracy loss of over 20%-points.

F5: The results on real-world data confirm our observations on synthetic data regarding accuracy and runtime (cf. Section 7.3). This demonstrates the practical feasibility of our approach as we can easily generate data synthetically to train our approach and still achieve accurate results on real-world data.

Due to F1 - F5, EffEns can be a strong fit especially for novice analysts to achieve valuable clustering results in a short time frame. Even experienced analysts can benefit from it. For instance, they can adapt the recommendations of the models, e. g., the ensemble from EGM, based on their domain knowledge and experience.

8 CONCLUSION

In this paper, we propose the novel efficient ensemble clustering approach ***EffEns***. We rely on meta-learning to learn the correlation of clustering ensembles and consensus functions, depending on the data characteristics. By relying on efficient algorithms for the ensemble generation and on efficient consensus functions, our approach is able to automatically apply ensemble clustering in an efficient way. In our comprehensive evaluation, we unveil that our approach significantly outperforms state-of-the-art approaches regarding clustering accuracy and runtime. Therefore, novice analysts can achieve valuable clustering results even on large datasets.

Future work will investigate how data pre-processing steps can be used to increase the accuracy of the ensemble clustering results, while still preserving efficiency.

ACKNOWLEDGMENTS

Parts of this research were performed in the project 'VALID-PARTITION' as part of the Software Campus program, which is funded by the German Federal Ministry of Education and Research (BMBF) under Grant No.: 01IS17051.

REFERENCES

[1] Ebrahim Akbari et al. 2015. Hierarchical cluster ensemble selection. *Engineering Applications of Artificial Intelligence* (2015).

[2] Edesio Alcobaça et al. 2020. MFE: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research* 21, 111 (2020), 1–5.

[3] Mihael Ankerst et al. 1999. OPTICS: Ordering Points to Identify the Clustering Structure. In *ACM SIGMOD*.

[4] D. Arthur and Vassilvitskii. 2007. k-means++: The Advantages of Careful Seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics Philadelphia, 1027–1035.

[5] Hanan Ayad and Mohamed Kamel. 2003. Finding Natural Clusters Using Multi-Clusterer Combiner Based on Shared Nearest Neighbors. In *MCS*.

[6] Hanan G. Ayad and Mohamed S. Kamel. 2008. Cumulative Voting Consensus Method for Partitions with Variable Number of Clusters. *IEEE TPAMI* (2008).

[7] Hanan G. Ayad and Mohamed S. Kamel. 2010. On voting-based consensus of cluster ensembles. *Pattern Recognition* 43, 5 (2010), 1943–1953.

[8] Javad Azimi and Xiaoli Fern. 2009. Adaptive Cluster Ensemble Selection. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*. Morgan Kaufmann Publishers Inc., 992–997.

[9] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[10] Tossapon Boongoen and Natthakan Iam-On. 2018. Cluster ensembles: A survey of approaches with recent extensions and applications. *Computer Science Review* (2018).

[11] Matthew R. Boutell et al. 2004. Learning multi-label scene classification. *Pattern Recognition* 37, 9 (2004), 1757–1771. https://doi.org/10.1016/j.patcog.2004.03.009

[12] Paul S. Bradley and Usama M. Fayyad. 1998. Refining Initial Points for K-Means Clustering. In *ICML*.

[13] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. 2008. *Metalearning: Applications to data mining*. (1 ed.).

[14] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[15] T. Caliński and J. Harabasz. 1974. A Dendrite Method For Cluster Analysis. *Communications in Statistics* 1 (1974).

[16] Dorin Comaniciu and Peter Meer. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI* 24 (2002), 603–619.

[17] David L. Davies and Donald W. Bouldin. 1979. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2 (1979), 224–227.

[18] Carlotta Domeniconi and Muna Al-Razgan. 2009. Weighted cluster ensembles. *ACM TKDD* (2009).

[19] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[20] Radwa ElShawi, Hudson Lekunze, and Sherif Sakr. 2021. cSmartML: A Meta Learning-Based Framework for Automated Selection and Hyperparameter Tuning for Clustering. In *2021 IEEE International Conference on Big Data*.

[21] Radwa ElShawi and Sherif Sakr. 2022. TPE-AutoClust: A Tree-based Pipline Ensemble Framework for Automated Clustering. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*. 1144–1153.

[22] Martin Ester et al. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *ACM SIGKDD*.

[23] Xiaoli Z. Fern and Wei Lin. 2008. Cluster Ensemble Selection. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 1, 3 (nov 2008), 128–141.

[24] 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 (2015), 181–194. https://doi.org/10.1016/j.ins.2014.12.044

[25] Matthias Feurer et al. 2015. Efficient and robust automated machine learning. In *Advances in neural information processing systems*. 2962–2970.

[26] Ana L.N. Fred and Anil K. Jain. 2005. Combining multiple clusterings using evidence accumulation. *IEEE PAMI* (2005).

[27] Manuel Fritz et al. 2021. Efficient exploratory clustering analyses in large-scale exploration processes. *The VLDB Journal* 31, 4 (nov 2021), 711–732.

[28] Manuel Fritz, Dennis Tschechlov, and Holger Schwarz. 2020. Learning from Past Observations: Meta-Learning for Efficient Clustering Analyses. In *Big Data Analytics and Knowledge Discovery*.

[29] Manuel Fritz Michael Behringer Holger Schwarz. 2020. LOG-Means: Efficiently Estimating the number of Clusters in Large Datasets. *PVLDB* (2020).

[30] Junhao Gan and Yufei Tao. 2015. DBSCAN revisited: Mis-claim, un-fixability, and approximation. In *ACM SIGMOD*.

[31] Andrey Goder and Vladimir Filkov. 2008. Consensus Clustering Algorithms: Comparison and Refinement. In *ALENEX*.

[32] Keyvan Golalipour et al. 2021. From clustering to clustering ensemble selection: A review. *Engineering Applications of Artificial Intelligence* 104 (2021), 104388.

[33] Ibai Gurrutxaga et al. 2010. SEP/COP: An efficient method to find the best partition in hierarchical clustering based on a new cluster validity index. *Pattern Recognition* (2010).

[34] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. 2001. On Clustering Validation Techniques. *Journal of Intelligent Information Systems* (2001).

[35] Francisco Herrera et al. 2016. *Multilabel classification*. Springer.

[36] Ellen Hohma et al. 2022. SCAR: Spectral Clustering Accelerated and Robustified. *PVLDB* (2022).

[37] Prodip Hore, Lawrence O. Hall, and Dmitry B. Goldgof. 2009. A scalable framework for cluster ensembles. *Pattern Recognition* (2009).

[38] Dong Huang, Chang-Dong Wang, and Jian-Huang Lai. 2018. Locally Weighted Ensemble Clustering. *IEEE Transactions on Cybernetics* 48, 5 (2018), 1460–1473. https://doi.org/10.1109/TCYB.2017.2702343

[39] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*.

[40] Anil K Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern recognition letters* 31, 8 (2010), 651–666.

[41] Anil K. Jain and Richard C. Dubes. 1988. *Algorithms for clustering data*. Prentice Hall. 1–320 pages.

[42] Ludmila I Kuncheva, Stefan Todorov Hadjitodorov, and Ludmila P Todorova. 2006. Experimental comparison of cluster ensemble methods. In *2006 9th International Conference on Information Fusion*. IEEE, 1–7.

[43] Marius Lindauer et al. 2022. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. *Journal of Machine Learning Research* (2022).

[44] Yanchi Liu et al. 2010. Understanding of internal clustering validation measures. In *Proceedings - IEEE International Conference on Data Mining, ICDM*.

[45] Yue Liu, Shuang Li, and Wenjie Tian. 2021. AutoCluster: Meta-learning Based Ensemble Method for Automated Unsupervised Clustering. In *PAKDD*.

[46] Stuart P Lloyd. 1982. *Least Squares Quantization in PCM*. Technical Report 2.

[47] Lukas Miklautz et al. 2022. Deep Clustering With Consensus Representations. *2022 IEEE International Conference on Data Mining (ICDM)* (2022), 1119–1124.

[48] Murilo Coelho Naldi, ACPLF Carvalho, and Ricardo JGB Campello. 2013. Cluster ensemble selection based on relative validity indexes. *Data Mining and Knowledge Discovery* 27 (2013), 259–289.

[49] Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems* 14 (2001).

[50] Nam Nguyen and Rich Caruana. 2007. Consensus Clusterings. In *IEEE ICDM*.

[51] F Pedregosa et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[52] Robi Polikar and others. 2010. Learn++.MF: A Random Subspace Approach for the Missing Feature Problem. *Pattern Recognition* (2010).

[53] Yannis Poulakis, Christos Doulkeridis, and Dimosthenis Kyriazis. 2020. AutoClust: A Framework for Automated Clustering Based on Cluster Validity Indices. In *IEEE ICDM*.

[54] Adriano Rivolli, Lu´P F Garcia, Carlos Soares, Joaquin Vanschoren, and André CPLF de Carvalho. 2018. Towards reproducible empirical research in meta-learning. *arXiv preprint arXiv:1808.10406* (2018).

[55] Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. 2016. Adjusting for chance clustering comparison measures. *Journal of Machine Learning Research* 17, 134 (2016), 1–32.

[56] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *NeurIPS*.

[57] Alexander Strehl and Joydeep Ghosh. 2003. Cluster Ensembles - a Knowledge Reuse Framework for Combining Multiple Partitions. *J. Mach. Learn. Res.* (2003).

[58] A. Topchy, A.K. Jain, and W. Punch. 2005. Clustering ensembles: models of consensus and weak partitions. *IEEE TPAMI* (2005).

[59] Alexander Topchy, Anil K. Jain, and William Punch. 2004. A Mixture Model for Clustering Ensembles. In *SIAM SDM*.

[60] A.P. Topchy, M.H.C. Law, A.K. Jain, and A.L. Fred. 2004. Analysis of Consensus Partition in Cluster Ensemble. In *IEEE ICDM*.

[61] Dennis Treder-Tschechlov et al. 2023. ML2DAC: Meta-Learning to Democratize AutoML for Clustering Analysis. *Proc. ACM Manag. Data* (2023).

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

[63] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2014. OpenML: networked science in machine learning. *SIGKDD Explor. Newsl.* 15, 2 (jun 2014).

[64] Sandro Vega-Pons and José Ruiz-Shulcloper. 2011. A survey of clustering ensemble algorithms. *Int. J. Pattern Recognit. Artif. Intell.* 3 (5 2011).

[65] Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research* (2010).

[66] Xindong Wu et al. 2008. Top 10 algorithms in data mining. *Knowledge and information systems* (2008).

[67] Xingwang Zhao, Jiye Liang, and Chuangyin Dang. 2017. Clustering ensemble selection for categorical data based on internal validity indices. *Pattern Recognition* (2017).