VISPEK: a Visual Interactive System for Progressive Ensemble K-Means Clustering

Marco Angelini*
Link Campus University
Rome, Italy
m.angelini@unilink.it

Graziano Blasilli*
Sapienza University of Rome
Rome, Italy
graziano.blasilli@uniroma1.it

Giorgio Cazzetta*
Sapienza University of Rome
Rome, Italy
cazzetta@diag.uniroma1.it

Simone Lenti*
Sapienza University of Rome
Rome, Italy
lenti@diag.uniroma1.it

Alessia Palleschi*
Sapienza University of Rome
Rome, Italy
palleschi@diag.uniroma1.it

Giuseppe Santucci*
Sapienza University of Rome
Rome, Italy
santucci@diag.uniroma1.it

Abstract

K-means clustering is widely used, but its iterative and initialization-sensitive nature makes results hard to interpret, compare, and debug, especially when multiple runs are needed to obtain a reliable solution. This paper presents VISPEK, a visual interactive system for progressive ensemble k-means clustering that helps users understand how clustering results evolve and how agreement emerges across runs. By combining Progressive Visual Analytics with an ensemble-based strategy, VISPEK exposes intermediate results, stability and quality metrics, and similarities among runs, enabling users to inspect, explain, and steer the clustering process before convergence. In this way, VISPEK supports the interpretation of consensus formation, highlights uncertainty, and helps users identify promising results early. We validate the approach through two usage scenarios and an expert study with data science and machine learning experts, showing that VISPEK improves analysis transparency while reducing time and computational effort.

CCS Concepts

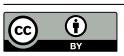
• **Human-centered computing** → **Visual analytics**; • **Computing methodologies** → **Machine learning**.

Keywords

Clustering, Ensemble Clustering, K-Means, Progressive Visual Analytics, Progressive Data Analysis and Visualization.

ACM Reference Format:
Marco Angelini, Graziano Blasilli, Giorgio Cazzetta, Simone Lenti, Alessia Palleschi, and Giuseppe Santucci. 2026. VISPEK: a Visual Interactive System for Progressive Ensemble K-Means Clustering. In *Proceedings of the 2026 International Conference on Advanced Visual Interfaces (AVI '26), June 08–12, 2026, Venice, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3811427.3811436

*The authors contributed equally to this research.



AVI '26, Venice, Italy

ACM ISBN 979-8-4007-2342-1/26/06
https://doi.org/10.1145/3811427.3811436

1 Introduction

Clustering is a widely used unsupervised method for exploring datasets with little prior knowledge. Achieving high-quality results, however, often requires iterative tuning, repeated runs, and high computational cost. In *k-means* [32], these issues are compounded by sensitivity to initialization: different runs can yield different outcomes, making results harder to interpret and compare. In addition, clustering quality is typically assessed only after execution, which limits timely intervention and slows exploratory analysis.

To address these limitations, Interactive Machine Learning (IML) introduced a human-in-the-loop approach [34], enabling users to guide machine learning processes through feedback and interaction. In parallel, Progressive Visual Analytics (PVA) [5] has shown how partial results can support earlier insight and more effective steering during long-running computations. In this paper, we present *VISPEK*, a visual interactive system for progressive ensemble *k-means* clustering. By leveraging PVA principles, *VISPEK* exposes intermediate clustering results during execution, rather than only at the end of the process. This allows users to understand how clustering evolves over time, how agreement emerges across runs, and how stable and reliable intermediate results are. *VISPEK* relies on the following strategies:

(1) to improve the quality of the clustering, it uses an ensemble-based approach, in which each intermediate result is obtained from an ensemble of partitions, guaranteeing a better quality than classic *k-means* at every iteration and mitigating this process;
(2) to produce a progression of results allowing human-in-the-loop clustering, it generates a sequence of ensembles in a parallel fashion, leveraging on r *k-means++* parallel processes and presenting the user with a representative clustering for each ensemble;
(3) to inform the user that it computes a set of stability and quality metrics that can be used to analyze the progression and manually steer it effectively at every iteration;
(4) to speed up the clustering of large datasets, it provides a convergence threshold that highlights the first partial result with a quality comparable with the one of the final result; such a threshold can be used to automatically trigger the early termination of *VISPEK*, speeding up repetitive and iterative analyses like, e.g., selecting the right k using the Elbow Method [41].

To further support repetitive tasks such as selecting the right k with the Elbow Method [41], *VISPEK* also provides early-termination mechanisms that identify partial results with quality comparable to the final ones, reducing time and computational cost. Overall, *VISPEK* turns clustering from a largely opaque post hoc activity into a more transparent, explainable, and interactive process.

2 Background and Motivation

The *k-means* is the most popular clustering algorithm [23]. It generates k non-overlapping subsets (clusters) from a dataset of n points in a $\mathcal{R}^d$ space. The objective function of *k-means* is to minimize a clustering metric called *inertia*, i.e., the sum of squared distances between each data point and its assigned centroid. The algorithm is iterative and works in four steps: 1) Initialization. Select k initial centroids randomly or use a more careful method, e.g., *k-means++* [6]. 2) Assign each data point to the closest centroid. 3) Recalculate the centroids as the mean of the data points of each cluster. 4) Repeat Steps 2 and 3 until convergence, i.e., when centroids no longer change significantly or when a maximum number of iterations has been reached. At each iteration, which costs $O(ndk)$, *k-means* generates a clustering that is refined in the next iteration, thereby decreasing inertia over time. Since *k-means* suffers from local minima, it can produce suboptimal solutions depending on the initial centroids (initialization). Different approaches for choosing a better initialization have been proposed [23], among them, *k-means++* [6] has gained popularity for its simplicity, speed, and quality. Although *k-means++* improves the clustering results, it still suffers from local minima. To illustrate this problem, we empirically evaluated the variability of *k-means++* results, even though it considerably improves on the classic *k-means* with random initialization.

Datasets. The evaluations in this paper were conducted using 21 datasets. Eight of them are *k-means* benchmark datasets (*S1, S2, S3, S4*) [20], (*A1, A2, A3*) [27], and *Unbalanced* [37]. The remaining ones are real datasets from the UCI Machine Learning Repository [14] (*Balance Scale, Liver-disorders, Contraceptive Method Choice, Diabetes, Glass, Heart Statlog, Ionosphere, Iris, Segmentation, Sonar, SPECTF Heart, Vehicles, Wine*). The Appendix Section 1 provides additional details about the datasets.

Variability of k-means++. To measure the variability of the *k-means++* results, we ran the algorithm on the 21 datasets. Given that inertia and, in general, all clustering validity indexes [25] (e.g., *Davies–Bouldin Index*, *Dunn Index*, *Silhouette*, etc.) heavily rely on the chosen parameter k, so we opted to assess each dataset using ten distinct values. These values were selected as equispaced points within the range $[2, \sqrt{n}]$, where n represents the size of the dataset. For each pair $\langle dataset,\ k\rangle$, we computed 100 runs of *k-means++* with different initializations. The resulting clusterings had varying inertia values. The optimal inertia (i.e., the minimum) was used as a baseline for evaluating the variability. All the other results were expressed as percentage differences relative to the best inertia. Then, it was possible to aggregate the variation for each dataset and all k values. Considering all datasets, k values, and 100 runs per pair, the resulting inertia varied by up to 160% from the optimal obtained in our experiments. The Appendix, Section 2, reports the full results obtained. The same experiments were conducted with *k-means* using random initializations, resulting in a variation in inertia of up to 3127%. To mitigate the risk of falling into suboptimal solutions, it is common practice to generate an ensemble of partitions running a *k-means* (or *k-means++*) algorithm multiple times with different initializations and then choose the best result based on the lowest inertia [6, 35]. In fact, by adopting this strategy, we increase the chances of finding a better clustering solution.

3 Related Work

This paper touches on three main areas of research: ensemble methods for controlling clustering quality, the progressive approach applied to computation to enable human-in-the-loop analysis, and interactive visual analysis of clustering results.

Ensemble clustering. Ensemble clustering combines multiple clustering results (partitions) into a single consolidated solution [40]. It generally involves two steps: generating diverse partitions and applying a consensus strategy to obtain the final clustering. Diversity can be achieved through multiple runs with different initializations or parameters, different algorithms, or different data subspaces [9]. The goal of clustering ensembles is to improve robustness and solution quality compared to individual clustering runs [1], and they have been successfully applied in several domains, including bioinformatics and image analysis. Most existing approaches rely on heterogeneous ensembles and produce results only after all runs have completed, which can be computationally expensive and limit user insight during execution [44]. In contrast, we adopt a homogeneous ensemble based on k-means and its variants and make the ensemble process progressive, following the Progressive Data Analysis and Visualization (PDAV) paradigm [16]. This allows intermediate results to be produced and inspected during execution, improving efficiency and supporting earlier insight with limited impact on clustering quality.

Progressive data analysis and visualization. Addressing scalability issues and enabling fluent exploration of clustering results have long been studied in the literature. Existing solutions span multiple levels, including computation, visualization, and interaction. Within this context, Progressive Data Analysis and Visualization (PDAV) has emerged as a research discipline that investigates methods for producing and interacting with partial, intermediate results during data analysis [16, 43]. Progressive Visual Analytics (PVA) is a prominent instantiation of this paradigm, focusing on integrating progressive computation with interactive visual interfaces to support early insight generation and user steering during long-running analyses [39]. Mühlbacher *et al.*[34] hinted at the possibility of applying PVA techniques to data clustering; our proposal starts from those hints to construct a PVA-based ensemble clustering (PEC) and, on top of it, a visual analytics environment. To the best of the authors' knowledge, no proposal for using progressive data analysis in ensemble clustering exists. The only proposal explicitly targeting the progressive implementation of clustering is the work by Fekete and Primet [17]: in this work, an algorithmic solution based on *mini-batch k-means* is provided, allowing the user to see incremental results. This approach, however, suffers from all the problems of a single run of *k-means* i.e., sensitivity to the choice of initial seeds and to the estimation of the quality of the final clustering result. To overcome these problems, our approach builds on the idea of

exploiting PVA but differs by using ensemble clustering techniques to gain greater control over the quality of the resulting clustering. Moreover, we provide a fully featured visual analytics environment for interactive exploration and steering of PEC results.

Interactive analysis of clustering. Applying Progressive Visual Analytics (PVA) to ensemble clustering enables user involvement during the clustering process, improving decision-making through early feedback and steering. A substantial body of work in Visualization and Visual Analytics has explored interactive clustering analysis, ranging from domain-specific systems (e.g., spatial, bioinformatics, and document clustering) to general-purpose visual analytic tools for cluster exploration and refinement. Representative systems support operations such as cluster comparison, merging, sub-clustering, and parameter tuning [12, 13, 29, 31]. Other approaches integrate prior user knowledge or guide users in selecting appropriate algorithms and parameters, sometimes leveraging ensemble-based strategies [36]. Pister *et al.*[36] explicitly acknowledge PVA as a way to overcome computation limitations encountered in their work. We propose a solution in this paper that fits this gap. Fiol-Gonzalez *et al.* [19] introduce a visual analytic system to study results from ensemble clustering. However, they address classic monolithic ensemble clustering and do not propose mechanisms for progressive computation and related visualizations. However, most of these solutions rely on classic monolithic computation, producing clustering results only after all configurations have been computed. As a consequence, they scale poorly as the number of parameter combinations or dataset size increases, forcing users to wait for lengthy precomputations before analysis can begin. Our work addresses this limitation by supporting progressive computation and interaction, enabling users to explore and guide clustering results as they are generated.

Lying at the intersection of those three areas, our proposal focuses on making available the clustering results in a progressive way, while respecting the interaction time constraints associated with the *unit task completion* activity [2, 42, 46]. While addressing identified challenges [8] in interactive clustering, such as improving parameter selection, providing better guidance during the process, enabling machine initiative through the introduced optimization to *k-means*, and ensuring comparability of results by keeping track of the process followed by the analyst. The use of ensemble clustering allows for higher-quality, more stable results, while progression ensures interactive exploration of clustering algorithm results by producing meaningful incremental results. The user is allowed to explore the results of a clustering ensemble long before its natural termination, fostering exploratory activities, e.g., by early-terminating a not useful analysis or early-detecting a bad clustering configuration, making informed decisions sooner. All those functionalities are implemented in the proposed visual analytics environment and have the benefits of reducing the needed time for analyzing clustering results, controlling the quality of the produced partial results, and allowing steering of the process, all of it while keeping intact the exploration of clustering features that other monolithic workflows provide to the analyst.

4 VISPEK

The cluster analysis procedure usually involves a series of trials and repetitions of five steps [24, 45]. The *Features selection* step is the selection of distinguishing features from the original ones or their transformation (extraction) into novel features. *Algorithm design* is the selection and parametrization of the clustering algorithm, while *Clusters computation* executes the clustering algorithm to obtain the partition. During *Cluster validation*, effective evaluation standards and criteria should support tasks such as determining the number of clusters and assessing the meaningfulness of the results. There are three main categories of testing criteria: external indices, based on prior information about the data; internal indices, based on analysis of the clustering structure of the original data; and relative indices, based on comparisons of different clustering results. Finally, in *Results interpretation*, experts in the data domain interpret the data partition to obtain meaningful insights and evaluate the extracted knowledge. These steps can be traced back to those in Dudley and Kristensson's workflow [15] for Interactive Machine Learning (IML), i.e., *Features selection*, *Model selection*, *Model steering*, *Termination assessment*, *Quality assessment*, and *Transfer*.

This paper explores the idea of accelerating the internal loop *(algorithm design - clusters computations - clusters validation)* by parallelizing r runs of *k-means*. The conceptual architecture is shown in Fig. 1. At every iteration j, the runs $(p_1, \ldots, p_r)$ produce an ensemble E_j of partitions. The intermediate result c_j is the best partition of the ensemble E_j. The number of intermediate results $(c_1, \ldots, c_f)$ equals the number of iterations in the longest run, and completed runs contribute their last result to the ensemble.

Generating intermediate results via an ensemble is more reliable than relying on a single run (Sec. 2). It enables the use of relative indices alongside internal indices during evaluation. To make this solution actionable, we explored how to inform the evaluation of intermediate results and the monitoring *(quality assessment)*, steering *(model steering)*, and termination *(termination assessment)* of the progression. To support progression management, we have selected a set of state-of-the-art PVA requirements [5, 7, 18] for both computational and visual aspects. The requirements (**Rxy**) were mapped to the three clustering steps that support the generation of partial results. Appendix Section 3 provides further details on the requirements. These were combined with the six key solution principles proposed by Dudley and Kristensson [15] to design the visual analytics system. Sec. 4.1 describes the metrics and means that support the progression while Sec. 4.2 depicts the system.

4.1 Supporting the progression

The proposed solution can generate structure-preserving intermediate results (**RM24**) by construction, while the convergence properties of *k-means* and the selection of the optimal partition at each iteration yield increasingly meaningful partial results (**RS14**). However, it is necessary to determine supporting metrics and methods to evaluate the progression and, if necessary, control it.

Metrics. We have identified three groups of metrics that enable the user to assess the process's progress, quality, and stability [4]. We chose the inertia for the progress due to its intrinsic link with *k-means*. It is the metric the algorithm attempts to minimize, and in the classic version, it is calculated at each iteration, requiring no additional calculations. We used standard clustering quality metrics [10, 25], such as the Calinski-Harabasz Score, the Davies–Bouldin Index, the Dunn Index, and the Simplified Silhouette. To assess

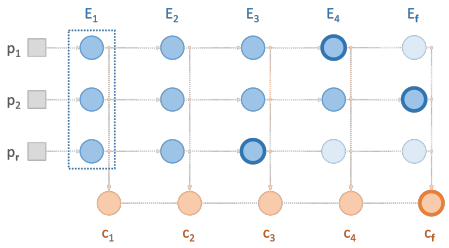


Figure 1: *VISPEK* computes a sequence of partial results c_j using the ensembles E_j generated by $r = 3$ parallel runs at each iteration j, while standard ensemble clustering waits for all runs to finish before producing c_f. The number of iterations per run is variable, so the process duration is determined by the longest run (p_2); completed shorter runs (p_1 and p_r, blue borders) contribute their final result to successive ensembles.

stability, we defined the *Local Stability* and *Global Stability* metrics, which were computed for individual data entries and the entire dataset, respectively. These metrics, for a given iteration j, analyze a window of w previous partial results and return a value in $[0, 1]$ where 1 means the result was completely stable during the last w iterations and 0 means completely unstable. Differences among partial results are weighted, with greater importance given to the latter. Let $L_j(x)$ be the cluster assigned to the data entry x at iteration j. The Local Stability (S) is defined as follows: $S(x, j, w) = \sum_{i=1}^{w} [L_j(x) == L_{j-i}(x)] \cdot \log(i + 1)$. The Global Stability is the average of all the Local Stability values.

Early Termination. A key feature of *VISPEK* is its ability to support the decision of stopping the progression before its termination by looking at values of inertia collected during the progression (**RM28**). When two consecutive iterations show a decrease of inertia ($\Delta_i = [1 - (inertia_i / inertia_{i+1})]$) below a certain small threshold $\mathcal{T}$, the result can be considered stable and the early termination can occur. In our experiments on the 21 datasets, two threshold values emerged: $\mathcal{T}_{Fast} = 2.0244\text{e}{-3}$ (faster) and $\mathcal{T}_{Slow} = 2.4399\text{e}{-4}$ (more conservative); see Appendix Section 4 for derivation details.

Early Termination Validation. We conducted an experimental validation using 21 datasets (Sec. 2) to assess the effectiveness of early termination. First, for each dataset, we chose a set of ten k values as equispaced points within the range $[2, \sqrt{n}]$. Then, for each pair $\langle dataset, k \rangle$, we compute 100 ensembles with $r = 16$ and different initializations. For each ensemble, the results obtained at the two levels of early termination were compared with the final result obtained at the normal end of the ensemble computation. For doing that, we identified two aspects to consider: the *computation gain* and the *inertia error*. The *computation gain* quantifies the percentage of computations (i.e., *k-means* iterations) saved if the progress was early terminated. The *inertia error* expresses how much the inertia obtained at the early termination differs from the inertia value obtained at the end of the progression, which is the best inertia obtainable for the current ensemble. The *inertia error* at iteration i is $E_i = [(inertia_i / inertia_z) - 1]$, when the total iterations are z. It is important to note that both metrics can be computed a posteriori only when we have the final result of the progression. Appendix Section 4 presents detailed results. For the *computation gain*, we obtained median values of 45% and 26% for the fast and slow early termination, respectively. This result confirms the effectiveness of early termination in reducing computations. For the *inertia error*, we obtained medians of 0.01% and 0.002% with fast and slow early termination, respectively. Both early termination approaches exhibit limited *inertia error*, with the slow early termination having a first-quartile value of 0. This confirms the effectiveness of early termination in limiting the error.

4.2 Visual solution

Relying on the identified requirements and the means described in the previous section, we designed and implemented a progressive visual analytics system (Fig. 2). The **Configuration Pane** (Fig. 2A) is responsible for configuring and managing execution (**RT36**). After selecting the dataset, the operator chooses the initialization method, the number of parallel runs, and the termination condition (fast, slow, or complete). Two modes are available: the first specifies the number of clusters to launch in a single execution, while the second defines a range of k values to run multiple executions, each with a different k. We describe the former first and then the additional features of the latter. Once configured, the operator can play, pause, stop, and restart the computation (**RH3**). The **Activities Pane** (Fig. 2B) lists all launched computations. Each entry shows a progress bar whose color encodes r, overlaid with early termination markers (silver for fast, gold for slow), and a quality indicator for the current result. Clicking an entry loads it into the main views.

Three main views support inspection, comparison, and steering, each using a distinct visual representation (**RF12**). The **Clusters View** (Fig. 2C) displays the current clustering c_j on a 2D projection (PCA, t-SNE, or UMAP), updated at each iteration. Color assignment is kept stable across iterations and runs by solving a minimum-cost centroid pairing (**RS17,RB44**). To provide feedback on uncertainty (**RF9**), the points are divided into three bins according to their stability: unstable points are shown in light gray, mid-stable in dark gray, and stable points in cluster color, making border regions immediately visible. The **Comparison View** (Fig. 2D) shows the ensemble E_j that produced the current result (**RM26**). A similarity matrix (AMI or ARI [22, 40]) summarizes agreement among all run results; the row and column labels of the best run are highlighted in pink. Clicking a run label loads its partition into an adjacent scatterplot, where point color encodes agreement with c_j, and the quality metrics table is updated with color-coded percentage differences to ease comparison. The **Progression View** (Fig. 2E) tracks the full sequence $[c_1, \ldots, c_j]$ (**RT41**). The *Metrics Progression Pane* shows three vertically aligned line charts for inertia, a selectable quality metric, and Global Stability; vertical silver and gold lines mark the fast and slow early termination points (**RS18**). The *Runs Progression Pane* uses a matrix layout (runs × iterations) with color-coded inertia and quality values; early unstable iterations are grayed out to discourage premature decisions. At each iteration, the best result is highlighted, and results within a user-adjustable similarity threshold are bordered in yellow as alternatives to explore. The analyst can terminate underperforming runs directly from this pane (**RM30**), redistributing computation to the remaining ones.

In the second mode of use (Fig. 3), the analyst selects a range of k values to explore with multiple runs of the clustering process. The system presents a line chart showing the inertia values as k increases, enabling, for example, the application of the Elbow Method. To assist the analyst in identifying the knee, the system

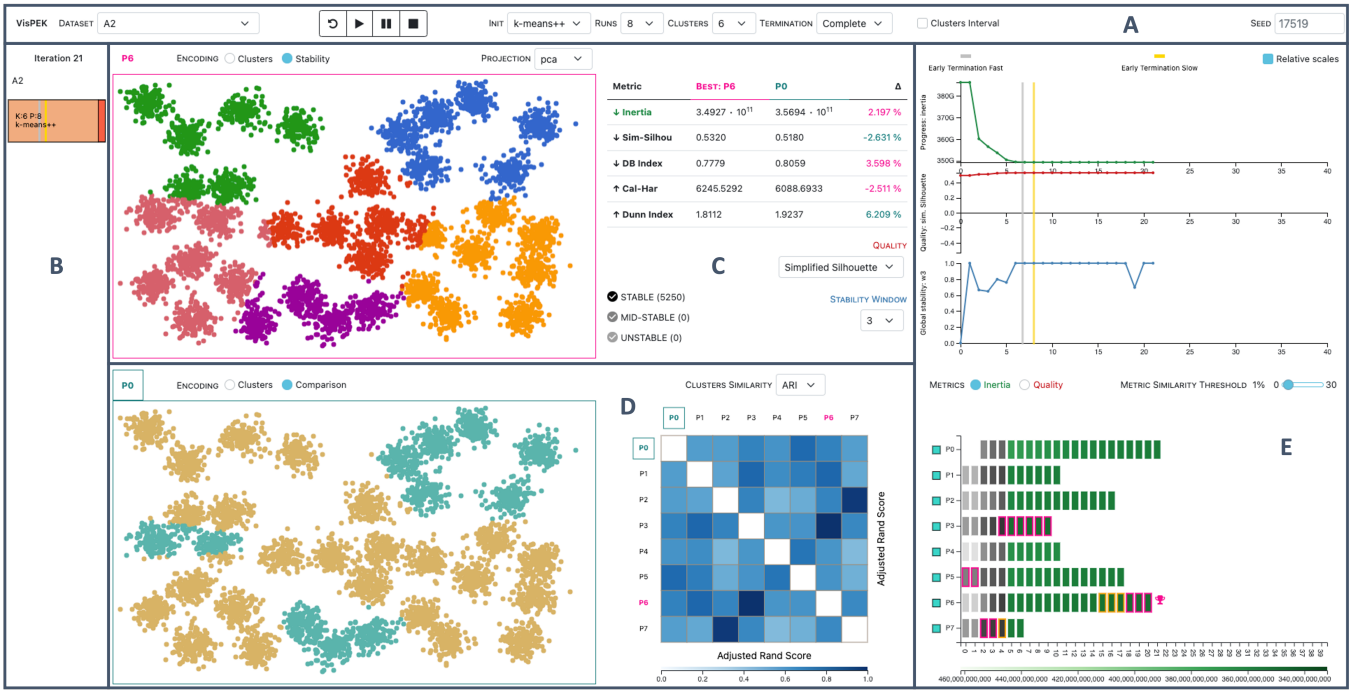


Figure 2: Overview of the *VISPEK* system. The *Configuration Pane (A)* is in charge of configuring and managing the executions. The *Activities Pane (B)* shows all the computations that have been launched. The *Clusters View (C)* allows the inspection of the clusters, showing them on a scatterplot with a 2D projection. The *Comparison View (D)* presents the ensemble that produced the current result by providing an overview of the results' homogeneity and showing the similarity matrix among the run results that compose the ensemble. The *Progression View (E)* shows information related to the progression with the *Metrics Progression Pane* (top) that shows the trends of different metrics, and the *Runs Progression Pane* (bottom) that shows the progression of all the runs composing the ensemble.

applies the automatic method proposed by Satopaa *et al.*[38] to compute it and highlights it on the line chart. At the bottom, a sequence of small multiples shows the scatterplots of the partitions calculated for each k. When a scatterplot is clicked, it appears in the main view, along with metric values and similarity metrics for the runs associated with that result.

Overall, *VISPEK* supports a spectrum of analysis strategies, from rapid, low-accuracy exploration to high-accuracy confirmatory analysis, while keeping users engaged as results unfold progressively, comparable to classic VA systems (e.g., [12, 29]).

5 Validation

VISPEK has been validated with two usage scenarios (Sec. 5.1 and Sec. 5.2) and a case study (Sec. 5.3) involving clustering experts.

The validations rely on the *"Spotify song tracks"* dataset [33], a real dataset containing audio features for over 1.2 million songs obtained through the Spotify API. As a preliminary step, we preprocessed the data by removing incomplete entries and keeping only numerical features. As a result, we got 14 numerical features characterizing the songs: *explicit*, *danceability*, *energy*, *key*, *loudness*, *mode*, *speechiness*, *acousticness*, *instrumentalness*, *liveness*, *valence*, *tempo*, *duration*, and *year*.

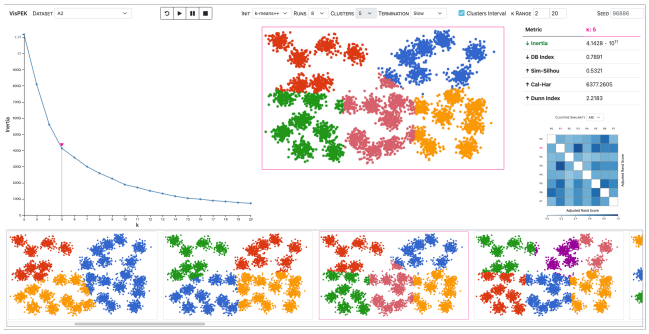


Figure 3: In the second usage mode of *VISPEK*, it displays the inertia across a range of increasing k values, facilitating the application of the Elbow Method. The system also applies an automatic method to identify the knee of the curve to assist the analyst. Furthermore, the system presents scatterplots showing the clustering results for various k values.

5.1 Usage scenario: Elbow method

Determining the correct number of clusters in a dataset is nontrivial, and several methods have been proposed in the literature [28]. For *k-means*, a common approach is the Elbow Method [41]. It consists

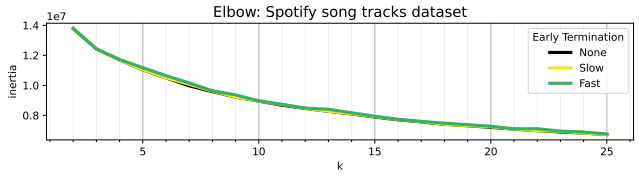


Figure 4: Elbow Method on the Spotify Dataset. The inertia curves for no early termination (black), slow (yellow), and fast (green) termination yield almost identical shapes, allowing the analyst to reach the same conclusion in less time.

of applying the clustering algorithm with increasing values of k (e.g., 2, 3, 4, ...) and plotting the resulting inertia as a function of k. The optimal number of clusters can be identified by looking at the shape of the curve rather than the actual inertia values to spot the k that creates an elbow. The elbow method is time-consuming since it executes the clustering algorithm a dozen times. In this scenario, *VISPEK* can reduce execution time by enabling early termination of the ensemble clustering processes at each k value, thereby speeding up the elbow computation. To evaluate the capability of *VISPEK* to speed up the time-consuming Elbow Method, we compared the shape of the inertia curves generated with i) no early termination condition (i.e., final curve), ii) slow early termination, and iii) fast early termination. To formally quantify the shape similarity between the final curve and the two early termination ones, we used the Pearson correlation of their derivatives [21]. The derivative is used to remove the descending trend that influences the correlation.

As shown by Fig. 4, through *VISPEK* we computed the elbow on the *Spotify Dataset* with $r = 16$ and the two early termination conditions. We obtained three curves: the final curve (no early termination) in 55 min, the slow curve in 20 min, and the fast curve in 13 min. The correlation of the slow curve with the final one is $\rho_{Slow} = 0.9984$, while the correlation of the fast curve with the final one is $\rho_{Fast} = 0.9952$. These values led us to assume that the fast and slow elbow curves perform similarly to the final elbow curve, with the advantage of speeding up the process. In fact, with the fast early termination condition, the elbow is computed in almost 20% of the computation time of the final curve.

We then systematically applied the same correlation calculations to the 21 datasets presented in Sec. 2. For each dataset, we used k values ranging in $[2, \sqrt{n}]$. The mean correlation of the slow curve with the final one is $\mu\rho_{Slow} = 0.99908$ with standard deviation $\sigma\rho_{Slow} = 0.00241$. The mean correlation of the fast curve with the final one is $\mu\rho_{Fast} = 0.99879$ with standard deviation $\sigma\rho_{Fast} = 0.00294$. The appendix provides additional details about the distributions. These results allow us to conclude that *VISPEK*, by leveraging early termination, enables us to perform the elbow analysis faster yet with equal accuracy.

5.2 Usage scenario: Song tracks clustering

This section demonstrates the workflow of clustering analysis using *VISPEK* on the *Spotify Dataset*. The analyst aims to find the most suitable clustering for the dataset to derive song genres.

Step #1: Preliminary elbow exploration. The analyst starts by performing a quick elbow method on the dataset to obtain candidate values of k to explore further and confirm. For this reason, she

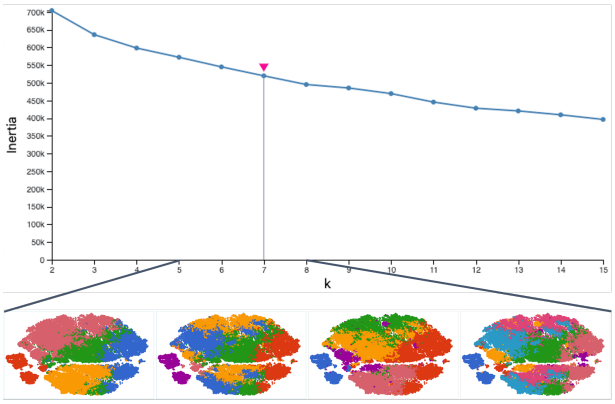
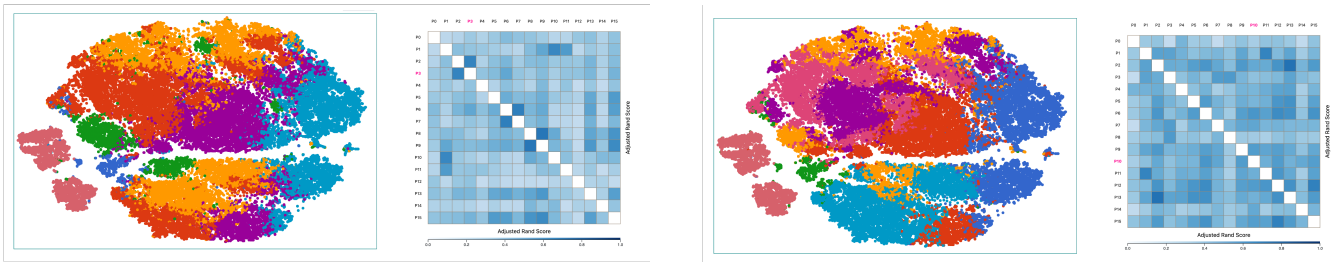


Figure 5: Elbow method on the Spotify Dataset ($r = 8$, fast early termination). Clusterings for $k \in [5, 8]$ are shown below the curve. The analyst identifies this interval as a good candidate range to continue the analysis (Step #3).

computes the elbow with a small $r = 4$ and the fast early termination, setting k in $[2, 20]$. This parametrization allows for a quick initial exploration and yields an inertia curve similar to the green curve in Fig. 4. The analyst spots possible elbow points between $k = 3$ and $k = 5$. At the same time, she noticed that for $k > 15$, the inertia seems relatively flat, suggesting that those k values may not be worth exploring. The computation of the fast elbow takes 14 s, whereas without early termination it would take 30 s.

Step #2: Initial interactive exploration. The analyst begins a quick exploratory analysis. For this reason, she executes a progressive ensemble clustering computation with $k = 3$ (spotted from Step #1) and a low number of runs $r = 4$. The analyst reviews the quality metrics in the Metrics Progression Pane and finds they are fairly stable. Conversely, the Comparison View, via the similarity matrix, shows that the runs are highly dissimilar (on average, $ARI < 0.3$) and yield different clusterings. In addition, the analyst notices that Local Stability indicates many unstable points in the Clusters View, particularly along the separation surfaces between clusters. Overall, the result is unsatisfactory, and the analysis is continued to explore with a different k. She then tests the $k = 5$ case. The fast early termination occurs after 12 s, but the analyst wants to stop the process at the slow early termination (14 s) to obtain a more accurate result. The clustering without early termination would last 60 s. She starts analyzing the quality metrics by comparing them with those obtained at $k = 3$. Quality metrics remained almost the same for lower k, but the similarity matrix shows that the runs now produced more comparable results. At the same time, the point's stability has increased. This gives the analyst the hint that a higher value of k could be the right one. At the end of this explorative phase, the analyst saved almost 80% of the time (22 s with early terminations against almost 120 s to obtain the same results without early termination.

Step #3: Higher-quality elbow exploration. Having gained a clue about raising the k value, the analyst chooses to use the elbow method k in $[2, 15]$ with twice as many runs ($r = 8$) to improve the



(a) $k = 7$ **(b)** $k = 8$

Figure 6: Comparison of clustering results on the Spotify Dataset (Step #5) with $r = 16$ and different values of k. (A) With $k = 7$, the similarity matrix reveals low agreement among runs, indicating suboptimal clustering. (B) In contrast, $k = 8$ shows greater agreement between runs and more compact clusters, suggesting that $k = 8$ is the desired number of clusters for this dataset.

accuracy of the result. The analyst still uses the fast early termination, since she knows it is as effective as the normal elbow but faster (see Sec. 4.1). The fast elbow is computed in almost 30 s, while the normal elbow would last almost 70 s. Inspecting the resulting curve, the analyst understands that k cannot be greater than 10 and that the interval to explore further is $[5, 8]$, as shown in Fig. 5.

Step #4: Increasing the accuracy. Having identified an interesting interval for k, the analyst increases the number of runs to $r = 8$ to obtain more accurate metrics and point stability values than in the first exploratory phase. The analysis proceeds from $k = 5$ to $k = 8$ in a similar way: launching the progressive clustering ensemble computation, examining the metrics, terminating unpromising runs to save resources, and identifying interesting alternatives with respect to labeling quality. Additionally, the analyst stops after fast early termination of both the computations with $k = 7$ and $k = 8$, identifying significant improvements in the quality metrics and stability. In particular, for $k = 8$, both global and local stability show the best performances, clusters are well separated, and all the clustering quality metrics show comparable (*Davies–Bouldin Index*, *Dunn Index*) or better results (*Calinski-Harabasz Score*) than the other explored k already at the fast early termination. By comparing all previously computed results in the Activity Pane, the analyst restricts the candidates to $k = 7$ and $k = 8$, discarding the previous hypothesis of $k = 5$ and excluding $k = 6$. In this phase the time saving, using the different means to steer the progression (early terminate progressive ensemble clustering computations, terminate runs, explore different partial results from the runs by looking at labeling quality) is even higher, resulting on average 20 s for all the 4 cases using fast early termination versus almost 105 s without early termination.

Step #5: Confirmatory analysis. In this step, the analyst performs a more accurate analysis for the candidates $k = 7$ and $k = 8$ using a large number of runs, $r = 16$ (Fig. 6). With this new parametrization, while without early termination the analyst should wait for almost 400 s or 60 s for the slow early termination, the fast early termination elapses almost 35 s. The ensemble computations with $k = 7$ and $k = 8$ produce quite similar results in terms of quality metrics; for example, *Davies–Bouldin Index* is 1.6 versus 1.8, and *Dunn Index* is 0.56 versus 0.55. The ensemble with $k = 7$ shows low agreement among runs, resulting in low Average Rand Index values; see Fig. 6a. Conversely, $k = 8$ shows greater agreement among the runs and fewer unstable points (see Fig. 6b), suggesting to the analyst that $k = 8$ could be the desired result. The analyst, to confirm this finding, launches another computation with $k = 8$ and a different initialization, waiting for the slow early termination. The new analysis shows slightly better quality metrics for $k = 8$, allowing the analyst to conclude that $k = 8$ is a good number of clusters.

In total, *VISPEK* with fast early termination allows the user to perform all computations for the presented analysis in almost 4 minutes. The same computations without early termination would take a total of 30 minutes. In this way, *VISPEK* provides a gain of almost 85% in computation time while preserving result quality, thereby strongly supporting the user in interactive clustering analyses. Given an average time of results analysis of 3 minutes per step, it means that this workflow can be completed in approximately 20 minutes using *VISPEK* and in 45 minutes using a monolithic approach, halving the time needed at comparable quality and keeping the user engaged in the exploration and hypotheses formulation and verification tasks.

5.3 Case study with domain experts

The case study was designed to address three research questions: *RQ1* — Does *VISPEK* reduce the time needed to conduct a clustering analysis compared to a monolithic approach? *RQ2* — Do the progressive and interactive features support users in making informed clustering decisions during the process? *RQ3* — How usable is the system for domain experts unfamiliar with Progressive Visual Analytics? Qualitative data were collected via think-aloud observations during the interactive and test sessions and through open-ended questions in the final questionnaire; responses were analyzed by grouping observations thematically around the three RQs. The case study involved four experts ($\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$, $\mathcal{E}_4$), aged 26-43. They are all Data Scientists and Machine Learning specialists who regularly deal with clustering. In particular, $\mathcal{E}_1$ works for *DXC Technology* and has a strong background in Data Science and clustering; $\mathcal{E}_2$ is a Professor of Data Science and Data Mining for Master's and Ph.D. courses; $\mathcal{E}_3$ is a researcher with a Ph.D. in Data Science and a strong background in clustering for genomics; and $\mathcal{E}_4$ is a Ph.D. student in computer vision. All of them declared that they used the scientific distribution of Python as the primary tool for conducting cluster analysis, with $\mathcal{E}_3$ also using R and $\mathcal{E}_1$ using Tableau.

Methodology. We evaluated the system with each expert separately in a two-hour online session structured as follows: VISPEK *presentation*, introducing the goals, design, and main features of the system (20 minutes); VISPEK *demo*, showing how to use the system on the A1 dataset (20 minutes); *Interactive session*, in which experts first explored the S1 dataset to familiarize themselves with the interface and then tested other available datasets (30 minutes); *Test session with the Spotify Dataset*, where experts conducted a clustering analysis, produced a final result, and reported their confidence in it under simulated real-task conditions (40 minutes); *Questionnaire*, completed at the end of the session to collect feedback on the experience with *VISPEK* (10 minutes).

5.3.1 Results. Experts could ask questions and receive assistance at any time. None of the experts was an expert in progressive data analysis. They claim to know *k-means* and other main clustering techniques, such as DBSCAN and hierarchical clustering, well.

RQ1 — Efficiency. All experts demonstrated strong proficiency in steering the process, terminating runs, and exploring partial results. On average, experts executed 8 progressive ensemble clustering computations ($\mathcal{E}_2$ executed 23). $\mathcal{E}_1$ reported progressive results as the main advantage over classic clustering, enabling strong interactivity during analysis. $\mathcal{E}_4$ noted that the system's speed and real-time unfolding of results are important functionalities. Experts found a suitable number of clusters for the Spotify Dataset in the range $[6, 8]$, consistent with the results of Sec. 5.2, and all declared high confidence in their results.

RQ2 — Decision support. The progressive views and the Activities Pane were unanimously appreciated for enabling comparison across partial results and steering the analysis. The similarity matrix was the most debated component: $\mathcal{E}_1$ found it useful for understanding the final result ("I found the matrix useful to understand the final result") and appreciated multiple accuracy levels ("The fast computation in real-time gives an initial intuition"), while $\mathcal{E}_2$ considered it less useful during analysis, though he was also the expert who executed the most computations, suggesting a preference for iterative re-running over ensemble inspection. This is the only component to receive divergent opinions; all other views were unanimously judged useful. $\mathcal{E}_3$ suggested adding a "step" functionality to the elbow method for large k-ranges, a direction for future work.

RQ3 — Usability. Experts completed a System Usability Scale (SUS) questionnaire [11]. Scores ranged from 65 to 75 (median 70.0, mean 69.37, std 5.0), indicating good perceived usability with room for improvement. $\mathcal{E}_1$ rated the system "Perfect"; the remaining experts rated it "Mostly Good". All declared they would use *VISPEK* in their clustering activities. Detailed SUS results are in the appendix.

6 Discussion and limitations

The validation results allow us to assess whether *VISPEK* achieves its stated goals. Regarding efficiency, the 85% computation gain demonstrated in the Spotify scenario (Sec. 5.2) and the experts' behavior—averaging 8 ensemble runs per session, with the ability to terminate unpromising runs early—confirm that *VISPEK* substantially reduces the computational cost of iterative clustering analysis. Regarding interpretability of result formation, the unanimous expert appreciation of the partial result views and Local Stability encoding confirms that progressive unfolding supports understanding of how clustering stabilizes, a capability absent in monolithic tools. Regarding interaction design, the divergence of opinion on the similarity matrix points to a concrete design lesson: its utility appears to depend on the analyst's preferred strategy—ensemble inspection versus iterative re-running. Future versions could make this view less prominent by default, activating it on demand, and could investigate whether adaptive interface configurations based on user behavior better serve different analytical profiles. These findings, together with $\mathcal{E}_3$'s suggestion of a step-based elbow exploration for large k-ranges, provide actionable design implications for future interactive machine learning tools of this type.

VISPEK inherits the pros and cons of *k-means*. In particular, it can face challenges when dealing with huge datasets due to the complexity of the *k-means*. As the dataset size increases, the computational cost rises, potentially leading to performance issues and extended processing times. Mini-Batch *k-means* has been proposed as a variation of the traditional *k-means* by processing small, random subsets, or "mini-batches", of the dataset in each iteration rather than using the entire dataset at once. Integrating Mini-Batch *k-means* is a promising direction, though it poses challenges for inertia computation and the timing of full-dataset labeling. A more sophisticated architecture of *VISPEK* could use Mini-Batch adaptively, triggering its use based on data size and user requirements.

In *VISPEK*, the best results are selected based on the best inertia. Other decision strategies for cluster ensembles exist, for example, consensus-based strategies such as the graph- and hypergraph-based methods presented by Strehl and Ghosh (CSPA, HGPA, and MCLA) [40]. These strategies could improve the results produced by *VISPEK*, even if they pose challenges due to their high computational complexity. A possible future research direction is to integrate some of those consensus functions into *VISPEK*. The explanation of decision strategies, integrated with visual analysis and aligned with eXplainable Artificial Intelligence principles, represents a third area of improvement for *VISPEK*, as shown by recent works [26, 30].

7 Conclusions

In this paper, we have presented *VISPEK*, a Progressive Visual Analytics solution that relies on a novel progressive ensemble clustering method based on *k-means*. The design of *VISPEK* aligns with state-of-the-art PVA requirements and provides means to observe the progression of results and to steer the whole process.
The analytic engine of *VISPEK* has been implemented and released as a Python module, ***pek*** [3], to facilitate adoption and use of our solution by researchers and practitioners.

Additional materials, source code, and documentation can be found at *https://github.com/aware-diag-sapienza/vispek*.

References

[1] Tahani Alqurashi and Wenjia Wang. 2019. Clustering ensemble method. *International Journal of Machine Learning and Cybernetics* 10, 6 (2019), 1227–1246.
[2] Robert Amar, James Eagan, and John Stasko. 2005. Low-level components of analytic activity in information visualization. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* IEEE, 111–117.
[3] Marco Angelini, Graziano Blasilli, Giorgio Cazzetta, Simone Lenti, Alessia Palleschi, and Giuseppe Santucci. 2026. PEK – Python Module. Available at https://pypi.org/project/pek/.
[4] Marco Angelini, Thorsten May, Giuseppe Santucci, and Hans-Jörg Schulz. 2019. On Quality Indicators for Progressive Visual Analytics. doi:10.2312/eurova.20191120

[5] Marco Angelini, Giuseppe Santucci, Heidrun Schumann, and Hans-Jorg Schulz. 2018. A Review and Characterization of Progressive Visual Analytics. *Informatics* 5, 3 (2018), 31. doi:10.3390/informatics5030031

[6] David Arthur and Sergei Vassilvitskii. 2007. K-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (New Orleans, Louisiana) *(SODA '07)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1027–1035. http://dl.acm.org/citation.cfm?id=1283383.1283494

[7] Sriram Karthik Badam, Niklas Elmqvist, and Jean-Daniel Fekete. 2017. Steering the Craft: UI Elements and Visualizations for Supporting Progressive Visual Analytics. *Computer Graphics Forum* 36, 3 (2017), 491–502. doi:10.1111/cgf.13205

[8] Juhee Bae, Tove Helldin, Maria Riveiro, Sławomir Nowaczyk, Mohamed-Rafik Bouguelia, and Göran Falkman. 2020. Interactive Clustering: A Comprehensive Review. *ACM Comput. Surv.* 53, 1, Article 1 (Feb. 2020), 39 pages. doi:10.1145/3340960

[9] Liang Bai, Jiye Liang, and Fuyuan Cao. 2020. A multiple k-means clustering ensemble algorithm to find nonlinearly separable clusters. *Information Fusion* 61 (2020), 36–47. doi:10.1016/j.inffus.2020.03.009

[10] Graziano Blasilli, Daniel Kerrigan, Enrico Bertini, and Giuseppe Santucci. 2024. Towards a Visual Perception-Based Analysis of Clustering Quality Metrics. In *2024 IEEE Visualization in Data Science (VDS)*. 15–24. doi:10.1109/VDS63897.2024.00007

[11] John Brooke. 1996. SUS: a "quick" and dirty usability scale. *Usability evaluation in industry* (1996), 189–194.

[12] M. Cavallo and C. Demiralp. 2019. Clustrophile 2: Guided Visual Clustering Analysis. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan 2019), 267–276. doi:10.1109/TVCG.2018.2864477

[13] J. Choo, H. Lee, J. Kihm, and H. Park. 2010. iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In *2010 IEEE Symposium on Visual Analytics Science and Technology*. 27–34. doi:10.1109/VAST.2010.5652443

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

[15] John J. Dudley and Per Ola Kristensson. 2018. A Review of User Interface Design for Interactive Machine Learning. *ACM Trans. Interact. Intell. Syst.* 8, 2, Article 8 (June 2018), 37 pages. doi:10.1145/3185517

[16] Jean-Daniel Fekete, Michael Sedlmair, and coeditors (Eds.). 2024. *Progressive Data Analysis and Visualization*. Eurographics Association. https://diglib.eg.org/items/c146a716-7b3f-469a-95a2-a354d349389e Book on the paradigm of progressive data analysis and visualization.

[17] Jean-Daniel Fekete and Romain Primet. 2016. Progressive analytics: A computation paradigm for exploratory data analysis. *arXiv preprint arXiv:1607.05162* (2016).

[18] Matteo Filosa, Graziano Blasilli, Emilio Martino, and Marco Angelini. 2026. ProVega: A Grammar to Ease the Prototyping, Creation, and Reproducibility of Progressive Data Analysis and Visualization Solutions. arXiv:2604.02096 [cs.HC] doi:10.48550/arXiv.2604.02096

[19] Sonia Fiol-González., Cassio Almeida., Ariane Rodrigues., Simone Barbosa., and Hélio Lopes. 2019. Visual Exploration Tools for Ensemble Clustering Analysis. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: IVAPP,*. INSTICC, SciTePress, 259–266. doi:10.5220/0007366302590266

[20] P. Fränti and O. Virmajoki. 2006. Iterative shrinking method for clustering problems. *Pattern Recognition* 39, 5 (2006), 761–765. doi:10.1016/j.patcog.2005.09.012

[21] EE Holmes, MD Scheuerell, and EJ Ward. 2020. Applied time series analysis for fisheries and environmental data. *Seattle: Northwest Fisheries Science Center* (2020).

[22] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of Classification* 2, 1 (01 Dec 1985), 193–218. doi:10.1007/BF01908075

[23] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. 2023. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences* 622 (2023), 178–210. doi:10.1016/j.ins.2022.11.139

[24] A. K. Jain, M. N. Murty, and P. J. Flynn. 1999. Data Clustering: A Review. *Comput. Surveys* 31, 3 (Sept. 1999), 264–323. doi:10.1145/331499.331504

[25] Adán José-García and Wilfrido Gómez-Flores. 2021. A survey of cluster validity indices for automatic data clustering using differential evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Lille, France) *(GECCO '21)*. Association for Computing Machinery, New York, NY, USA, 314–322. doi:10.1145/3449639.3459341

[26] Md Abdul Kadir, Abdulrahman Mohamed Selim, Michael Barz, and Daniel Sonntag. 2023. A User Interface for Explaining Machine Learning Model Explanations. In *Companion Proceedings of the 28th International Conference on Intelligent User Interfaces* (Sydney, NSW, Australia) *(IUI '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 59–63. doi:10.1145/3581754.3584131

[27] I. Kärkkäinen and P. Fränti. 2002. *Dynamic local search algorithm for the clustering problem.* Technical Report A-2002-6. Department of Computer Science, University of Joensuu, Joensuu, Finland.

[28] Trupti Kodinariya and P.R. Makwana. 2013. Review on Determining of Cluster in K-means Clustering. *International Journal of Advance Research in Computer Science and Management Studies* 1 (01 2013), 90–95.

[29] B. C. Kwon, B. Eysenbach, J. Verma, K. Ng, C. De Filippi, W. F. Stewart, and A. Perer. 2018. Clustervision: Visual Supervision of Unsupervised Clustering. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan 2018), 142–151. doi:10.1109/TVCG.2017.2745085

[30] B. La Rosa, G. Blasilli, R. Bourqui, D. Auber, G. Santucci, R. Capobianco, E. Bertini, R. Giot, and M. Angelini. 2023. State of the Art of Visual Analytics for eXplainable Deep Learning. *Computer Graphics Forum* 42, 1 (2023), 319–355. doi:10.1111/cgf.14733

[31] Hanseung Lee, Jaeyeon Kihm, Jaegul Choo, John Stasko, and Haesun Park. 2012. iVisClustering: An Interactive Visual Document Clustering via Topic Modeling. *Computer Graphics Forum* (2012). doi:10.1111/j.1467-8659.2012.03108.x

[32] S. Lloyd. 2006. Least squares quantization in PCM. *IEEE Trans. Inf. Theor.* 28, 2 (Sept. 2006), 129–137. doi:10.1109/TIT.1982.1056489

[33] Spotify MusicBrainz. 2020. Spotify 1.2M+ Songs. Audio features of over 1.2 million songs obtained with the Spotify API. https://www.kaggle.com/rodolfofigueroa/spotify-12m-songs.

[34] T. Mühlbacher, H. Piringer, S. Gratzl, M. Sedlmair, and M. Streit. 2014. Opening the Black Box: Strategies for Increased User Involvement in Existing Algorithm Implementations. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (Dec 2014), 1643–1652. doi:10.1109/TVCG.2014.2346578

[35] 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, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[36] A. Pister, P. Buono, J. D. Fekete, C. Plaisant, and P. Valdivia. 2021. Integrating Prior Knowledge in Mixed-Initiative Social Network Clustering. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 1775–1785. doi:10.1109/TVCG.2020.3030347

[37] M. Rezaei and P. Fränti. 2016. Set-matching methods for external cluster validity. *IEEE Trans. on Knowledge and Data Engineering* 28, 8 (2016), 2173–2186.

[38] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops*. 166–171. doi:10.1109/ICDCSW.2011.20

[39] C. D. Stolper, A. Perer, and D. Gotz. 2014. Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (Dec 2014), 1653–1662. doi:10.1109/TVCG.2014.2346574

[40] Alexander Strehl and Joydeep Ghosh. 2003. Cluster Ensembles — a Knowledge Reuse Framework for Combining Multiple Partitions. *J. Mach. Learn. Res.* 3 (March 2003), 583–617. doi:10.1162/153244303321897735

[41] Robert L. Thorndike. 1953. Who Belongs in the Family? *Psychometrika* 18, 4 (Dec. 1953), 267–276. doi:10.1007/BF02289263

[42] C. Turkay, E. Kaya, S. Balcisoy, and H. Hauser. 2017. Designing Progressive and Interactive Analytics Processes for High-Dimensional Data Analysis. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan 2017), 131–140. doi:10.1109/TVCG.2016.2598470

[43] Alex Ulmer, Marco Angelini, Jean-Daniel Fekete, Jörn Kohlhammer, and Thorsten May. 2024. A Survey on Progressive Visualization. *IEEE Transactions on Visualization and Computer Graphics* 30, 9 (2024), 6447–6467. doi:10.1109/TVCG.2023.3346641

[44] Sandro Vega-Pons and José Ruiz-Shulcloper. 2011. A SURVEY OF CLUSTERING ENSEMBLE ALGORITHMS. *International Journal of Pattern Recognition and Artificial Intelligence* 25, 03 (2011), 337–372. doi:10.1142/S0218001411008683

[45] Dongkuan Xu and Yingjie Tian. 2015. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science* 2, 2 (June 2015), 165–193. doi:10.1007/s40745-015-0040-1

[46] Ji Soo Yi, Youn ah Kang, and John Stasko. 2007. Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics* 13, 6 (2007), 1224–1231.