future internet

MDPI

Article

kClusterHub: An AutoML-Driven Tool for Effortless Partition-Based Clustering over Varied Data Types †

Konstantinos Gratsos [1], Stefanos Ougiaroglou [1] and Dionisis Margaris [2,*]

[1] Department of Information and Electronic Engineering, School of Engineering, International Hellenic University, Sindos, 57400 Thessaloniki, Greece; kostisgratsos12@gmail.com (K.G.); stoug@ihu.gr (S.O.)
[2] Department of Digital Systems, School of Economics and Technology, University of the Peloponnese, 23100 Sparta, Greece
* Correspondence: margaris@uop.gr
† This paper is an extended version of our paper published in the proceedings of 3rd International Conference on Novel and Intelligent Digital Systems, NIDS 2023, Athens, Greece, 28–29 September 2023.

Abstract: Partition-based clustering is widely applied over diverse domains. Researchers and practitioners from various scientific disciplines engage with partition-based algorithms relying on specialized software or programming libraries. Addressing the need to bridge the knowledge gap associated with these tools, this paper introduces kClusterHub, an AutoML-driven web tool that simplifies the execution of partition-based clustering over numerical, categorical and mixed data types, while facilitating the identification of the optimal number of clusters, using the elbow method. Through automatic feature analysis, kClusterHub selects the most appropriate algorithm from the trio of k-means, k-modes, and k-prototypes. By empowering users to seamlessly upload datasets and select features, kClusterHub selects the algorithm, provides the elbow graph, recommends the optimal number of clusters, executes clustering, and presents the cluster assignment, through tabular representations and exploratory plots. Therefore, kClusterHub reduces the need for specialized software and programming skills, making clustering more accessible to non-experts. For further enhancing its utility, kClusterHub integrates a REST API to support the programmatic execution of cluster analysis. The paper concludes with an evaluation of kClusterHub's usability via the System Usability Scale and CPU performance experiments. The results emerge that kClusterHub is a streamlined, efficient and user-friendly AutoML-inspired tool for cluster analysis.

Keywords: clustering; k-means; k-modes; k-prototypes; elbow method; autoML; web application; web service

Citation: Gratsos, K.; Ougiaroglou, S.; Margaris, D. kClusterHub: An AutoML-Driven Tool for Effortless Partition-Based Clustering over Varied Data Types. *Future Internet* **2023**, *15*, 341. https://doi.org/10.3390/fi15100341

Academic Editor: Michael Sheng

Received: 21 September 2023
Revised: 13 October 2023
Accepted: 16 October 2023
Published: 18 October 2023

1. Introduction

Clustering, a fundamental task in machine learning and data mining [1], involves grouping instances with similar characteristics into clusters. A cluster represents a collection of instances, where each member is more similar to other members of the same cluster than to those outside of it. This technique has found numerous applications across various domains, spanning from market research and search engines to psychology, medicine, biology, and beyond.

While the concept of clustering dates back to the 1930s, with origins in psychology and anthropology [1], computational limitations initially hindered the progress of clustering techniques. However, the advancement of computing power from the late 1950s onward catalyzed the development of the diverse clustering methods that are widely employed today. Given the intricate calculations and processing necessary for cluster identification, computer science plays a pivotal role in enabling these techniques to thrive. Among the plethora of clustering algorithms, partition-based algorithms are popular and frequently employed.

K-means [2] is a partition-based clustering algorithm, targeting the partitioning of instances into k clusters, based on their similarity, leveraging an iterative procedure that assigns instances to the nearest cluster centroid. K-modes [3,4], a variation of k-means, is suitable for categorical data, by optimizing the modes of attribute values within clusters. Unlike traditional numerical-based distance measures, k-modes employs a dissimilarity measure designed for categorical attributes, enhancing its applicability to datasets where attributes lack numerical meanings. Additionally, k-prototypes [3,4], an extension that combines both k-means and k-modes, offers a versatile solution for datasets featuring mixed data types—both categorical and numeric attributes. K-prototypes combines the numerical-based distance calculation of k-means with the categorical-based dissimilarity measure of k-modes to handle hybrid data effectively. This makes k-prototypes a valuable tool for clustering datasets that include attributes of varying types.

The challenge of selecting the appropriate partition-based clustering algorithm for different data types (numeric, categorical, and mixed) is a crucial aspect, especially for non-expert users. This difficulty serves as a motivation behind the present work, as it aims to provide a user-friendly solution which automatically identifies the most suitable algorithm for the given data type.

Furthermore, determining the optimal number of clusters, denoted by k, poses a challenge in k-means clustering and its aforementioned variants. Adjusting the value of k requires prior knowledge of the true number of clusters, which is often unavailable in real-world datasets. Additionally, selecting an inappropriate k may lead to sub-optimal cluster assignments, potentially overlooking meaningful patterns in the data. The elbow method [5] frequently addresses this challenge by identifying the "elbow point" in the plot of explained variance versus the number of clusters. The use of the elbow method by the non-expert user for the automatic determination of the k parameter value constitutes the motive of the present work.

Moreover, K-means clustering, with or without the incorporation of the elbow method, is available in numerous dedicated standalone software solutions (e.g., Matlab, Weka [6], SPSS, Orange [7], etc.) and programming language libraries (e.g., Python scikit-learn [8] and R clustMixType [9]). K-modes and k-prototypes are not available in the aforementioned standalone applications. However, they are available in the scikit-learn library. Utilizing k-means, its variants and the elbow method within these software and programming environments often entail considerations, such as software licensing, downloading, installation, and the need for specialized expertise in these tools and programming skills. This combination of factors might create barriers, including the need for specific knowledge and skills. The absence of freely available web applications or services dedicated to partition-based clustering is notable within the authors' awareness. This observation serves as another motivation for the current work.

This paper extends the previous work by introducing kClusterHub, a web application that builds upon the development of Web-k-means [10], furthering its capabilities to encompass not only k-means but also k-modes and k-prototypes clustering algorithms. kClusterhub deals with the difficulties when performing partition-based clustering by non-experts. The difficulties are summarized as follows: (i) difficulty of the decision of which partition-based clustering algorithm is appropriate for the different data types, (ii) difficulty of the determination of the number of clusters, and (iii) need of programming skills and/or specialized software to perform partition-based clustering.

Therefore, the target users of kClusterHub are researchers and practitioners from diverse domains who may not have expertise in clustering algorithms, programming skills and knowledge on specialized software, yet seek an efficient and accessible tool to derive meaningful insights from their data through partition-based clustering. With kClusterHub, users can seamlessly perform clustering analyses without the need to download, setup and use specialized software (e.g., WEKA [6,11]) or write code and use programming libraries (e.g., Scikit-learn [8]).

The overarching objective of the paper is to create a seamless and user-friendly AutoML-driven environment for non-expert researchers and practitioners seeking to conduct partition-based clustering across numerical, categorical and mixed data. In the spirit of automated machine learning (AutoML) [12], kClusterHub takes a step further in facilitating the clustering tasks. By recognizing the attributes' data types within the uploaded dataset, the tool selects the suitable clustering algorithm from the trio of k-means, k-modes, and k-prototypes. Moreover, kClusterHub is capable of suggesting the optimal number of clusters by utilizing the elbow method. Both aforementioned capabilities are AutoML-inspired. Thus, kClusterHub empowers users to seamlessly upload datasets, select features and execute clustering analyses, via a user-friendly interface. The tool presents the insightful elbow method graph, suggests the optimal number of clusters, and offers a detailed exploratory plot of the cluster assignments that enables users to understand the characteristics of each discovered cluster. In essence, the benefit of using kClusterHub is that it empowers even non-expert users in cluster analysis to effortlessly perform partition-based cluster analyses, making it an useful tool for researchers and practitioners seeking insights from their data.

In addition, kClusterHub offers a REST API that programmers can use to access the capabilities of the tool programmatically from their own code. This allows for the integration of kClusterHub's functionalities into other applications. Therefore, kClusterHub not only enhances the accessibility of clustering analyses to non-experts but also aligns with the direction of automated decision making in machine learning.

Consequently, kClusterhub deals with the difficulties when performing clustering analysis by non-experts. The difficulties are summarized as follows: (i) difficulty of the decision of which clustering algorithm is appropriate for the different data types, (ii) determination of the optimal k value, and (iii) need of programming skills and/or specialized software.

The rest of the paper is organized as follows: Section 2 reviews the related works. Section 3 provides an overview of k-means, k-modes, and k-prototypes as well as the corresponding cluster centroid's initialization methods. The same section reviews the elbow method, which serves as a cornerstone for the optimal number of clusters determination. The kClusterHub app is presented in Section 4. The system's evaluation, encompassing CPU time measurements and usability assessments using the System Usability Scale (SUS) questionnaire, is presented in Section 5. Finally, Section 6 concludes the paper by summarizing the key findings and outlining directions for future work.

2. Related Work

The development and usage of AutoML tools is one of the machine learning and data mining research fields that has attracted attention during the last years. However, they primarily cater to supervised learning tasks, such as classification and regression, rather than clustering. For instance, auto-sklearn [13] is an AutoML tool for Python that focuses on classification and regression tasks. It does not directly support clustering algorithms like k-means, k-modes, or k-prototypes. H_2O.ai [14] provides an AutoML platform that supports various machine learning tasks. While it primarily emphasizes supervised learning, it does offer clustering capabilities.

On the other hand, there are many specialized standalone machine learning software tools and libraries of programming languages that support clustering tasks, partition-based clustering included. However, all of them imply specialized knowledge on these tools and/or programming skills as well as knowledge on the research field of data mining and machine learning. Thus, they cannot be used by practitioners who are non-experts. In addition, the software tools require installation on certain operating systems, and some of them need licenses. Therefore, they are inappropriate for many devices. Moreover, the standalone software tools, libraries of programming languages and web-based platforms (e.g., the non-free Amazon SageMaker Canvas [15]), offer a wide range of machine learning algorithms and extensive capabilities for data preprocessing, modeling, and evaluation. However, they entail a steeper learning curve, especially for non-experts, when

compared to the specialized focus of kClusterHub, which prioritizes simplicity, accessibility, and automation in partition-based clustering tasks, enabling users to analyze their data without the need for expertise or extensive setup.

For example, WEKA [6,11] stands as a widely recognized and extensively used software suite. Developed by the University of Waikato, WEKA provides a comprehensive collection of machine learning and data mining algorithms and data preprocessing techniques, making it a versatile solution for various data analysis tasks. Its graphical interface and open-source nature have contributed to its popularity, allowing researchers and practitioners to efficiently explore, preprocess, and model data for classification, clustering, regression, association rules mining and other machine learning tasks. WEKA must be downloaded and installed on Linux, Windows or Mac OS operating systems as long as Java is installed and configured.

Demsar et al. [7] present Orange, an open-source Python-based framework for data mining and machine learning, which can be used by both researchers and experienced users. Orange empowers users to construct workflows, visualize data, and experiment with various machine learning and data mining techniques (k-means clustering included), while the script-oriented capabilities facilitate the integration of novel methods. Like WEKA, Orange must be downloaded and installed on Linux, Windows or Mac OS operating systems. Also, Orange is built on top of the Python language and relies on Python libraries to provide its functionalities. Therefore, having a Python installation is a prerequisite.

Scikit-learn [8], a widely used machine learning library in Python, offers an extensive set of tools for various tasks in data analysis and modeling. Developed on the foundation of NumPy, SciPy, and matplotlib, scikit-learn provides a user-friendly interface for tasks such as classification, regression, clustering, and more. Its straightforward API and comprehensive documentation make it an essential resource for researchers seeking efficient and accessible implementations of diverse algorithms. Utilizing scikit-learn necessitates having Python installed on your system and a certain level of programming skills. Since scikit-learn is a Python-based library, users should have a working Python environment to leverage its capabilities.

While WEKA, Orange and Scikit-learn provide implementation of the k-means clustering, they do not offer built-in support for k-modes and k-prototypes algorithms. This limitation highlights the importance of considering alternative libraries or tools when working with datasets that require these specialized clustering techniques.

Several R libraries offer implementations of the k-means clustering. The "stats" package, which is included in the base R distribution, provides k-means implementation. Additionally, the "cluster" package [16] introduces more advanced variants of the algorithm which supports various centroids initialization methods. Also, Brock et al. [17] present clValid, a package in R, that includes a set of functions which validates the cluster analysis results. The aforementioned R libraries do not offer directly k-modes and k-prototypes implementations. The clustMixType package [9] offers clustering methods for mixed-type data. While it does not provide a direct implementation of k-modes or k-prototypes, it offers various clustering algorithms suitable for mixed data types. Using the aforementioned libraries require a combination of programming skills, statistical knowledge, and of' course, a familiarity with the R environment.

Moreover, MATLAB offers built-in implementations of the k-means clustering. The k-means algorithm is available as part of MATLAB's Statistics and Machine Learning Toolbox. MATLAB's toolbox focuses on numerical data analysis and does not provide implementation of the k-modes or k-prototypes algorithm. Lin et al. [18] introduce a consensus clustering algorithm package, namely KCC, in MATLAB targeting at solving problem of consensus clustering. Consensus clustering combines multiple clustering solutions obtained from different algorithms or parameter settings to enhance the stability and reliability of the final clustering results. Lin et al. implement partitions using random sampling of data features or clustering numbers, utility function selection from multiple k-means distances, and a knowledge fusion iterative process of basic partitions.

KNIME [19] is an open-source platform for data analytics. KNIME provides a visual programming interface that allows users to create data workflows by connecting various data processing and analysis tasks. It supports a wide range of data manipulation, transformation, and analysis tasks, including data preprocessing, machine learning, and data visualization. KNIME provides extensions that include implementations of k-means, k-modes, and k-prototypes clustering. These algorithms can be utilized within the KNIME visual programming interface. KNIME is a standalone software platform that needs to be installed on a computer with a Windows, macOS, or Linux operating system.

Regarding the k-modes and k-means algorithms, which are not as popular as k-means, we briefly present some related works. Chaouch et al. [20] introduce an audio fingerprint DB structure, based on the MongoDB platform, using the K-modes clustering algorithm. Chadha and Kumar [21] introduce a clustering algorithm, based on the K-modes, that automatically finds the appropriate number of clusters. The algorithm generates clusters, by targeting at maximizing the matching values of the number of attributes within a cluster; however, the time needed for the algorithm to execute is substantially increased when compared to the original K-Modes algorithm. Jiang et al. [22] present two K-modes clustering algorithms, based on the partition entropy-based outlier detection method and on the traditional distance-based outlier detection method, respectively, so as to guarantee that the initial centroids chosen are not outliers. Furthermore, they adopt a new weighted-matching distance measure during the initialization step in order to measure the distance between two items defined by categorical features.

HajKacem et al. [23] present a k-prototypes algorithm parallelization, namely MR-KP, which uses the MapReduce model. This algorithm is able to cluster a large number of mixed data efficiently, as well as satisfactorily scaling along with the increasing number of the datasets. Jia and Song [24] present the weighted k-prototypes clustering algorithm, an extension of k-prototypes, which is based on the hybrid dissimilarity coefficient. The algorithm improves the plain k-prototypes algorithm by automatically selecting the initial cluster centroids, using local neighborhood, average and relative distances. As far as categorical data are concerned, the algorithm uses the type dissimilarity coefficient with entropy weight, while as far as numerical data are concerned, the algorithm uses the quantized numerical dissimilarity coefficient. Kuo and Wang [25] present a clustering-based classification method concerning mixed data, which has both clustering and classification advantages. In order to find the initial centroids and the attribute weights for k-prototypes, the method employs a sine–cosine algorithm. Also, the method's objective function is formulated as a sum-up purity, while it uses the mutation strategy in order to succeed with better performance for sine–cosine algorithms. Sangam and Om [26] introduce a k-prototypes dissimilarity measure, which is able to find the similarity between mixed-type attribute data instances. The weight age Hamming dissimilarity metric introduced is used, while this dissimilarity metric considers both the relative frequency and the distribution of each mode category. Jang et al. [27] present a grid-based k-prototypes algorithm, namely GK-prototypes, that enhances the basic algorithm's performance. As far as categorical attributes are concerned, the algorithm takes into account the maximum distance between a cluster center and a cell, while as far as numeric attributes are concerned, the algorithm takes into account the maximum and minimum distances.

Last but not least, kClusterHub builds upon the development of Web-k-means [10], which is a web tool designed specifically for k-means clustering on numeric data. Both kClusterHub and web-k-means follow the strategy initially introduced by WebApriori [28]. While kClusterHub and web-k-means focus on partition-based clustering, WebApriori serves a distinct purpose, enabling association rule mining through the utilization of the Apriori algorithm. For this purpose, WebApriori provides users with both a user-friendly web interface and a REST API web service.

3. Partition-Based Clustering

3.1. K-Means

K-means clustering [2] is a technique that groups instances based on their similarities. This process involves iteratively assigning each instance to a cluster by considering its distance from the cluster's centroid (the mean of the instances in the cluster). The algorithm then updates the centroids until they stabilize. Its major objective is to minimize the within-cluster sum of squares (WCSS), which represents the sum of squared distances between instances and their assigned cluster centroids. Formally, for a set of instances, $X = \{x_1, x_2, \ldots, x_n\}$, k-means aims to assign the n instances to k clusters ($k \leq n$) $S = \{S_1, S_2, \ldots, S_k\}$ targeting at minimizing the WCSS function:

$$WCSS = \sum_{j=1}^{k} \sum_{x_i \in S_j} ||x_i - \mu_j||^2$$

Here, μ_j represents the mean of cluster S_j, and $||x_i - \mu_j||$ is the selected distance metric between the instance x_i and the corresponding mean.

The k-means algorithm commences by initializing k centroids randomly, which serve as the initial cluster centers. Subsequently, each instance is assigned to its closest centroid, and the centroid is updated to the center of the instances that have been assigned to it. This process repeats iteratively until the centroids converge. Algorithm 1 provides a pseudo-code representation of the k-means clustering technique. Initially, the algorithm takes a dataset $X = x_1, x_2, \ldots, x_n$ and forms k clusters (C). The algorithm initially assigns each instance x_i to the cluster with the closest centroid (mean) c_j, which is termed as the assignment step (lines 4–6). Following the assignment, the algorithm computes new means, by averaging the instances within each cluster (lines 7–9), which can be computed as

$$c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

The assignment step is then repeated, incorporating the updated means. Hence, the means are continually adjusted throughout the process. The algorithm stops when there is no change in means (cluster consolidation) between two consecutive iterations (line 10). Given that each instance is closer to the centroid of its respective cluster, the WCSS function is minimized, signifying successful clustering.

Algorithm 1 K-means clustering.

Require: $X = x_1, x_2, \ldots, x_n$—dataset, k—number of clusters
Ensure: $C = C_1, C_2, \ldots, C_k$—set of clusters
1: Select k initial centroids $c_1, c_2, \ldots, c_k$ randomly from X
2: **repeat**
3: Create k empty clusters $C_1, C_2, \ldots, C_k$
4: **for** $i = 1$ to n **do**
5: Assign x_i to the cluster C_j with the closest centroid c_j
6: **end for**
7: **for** $j = 1$ to k **do**
8: Update centroid c_j as the mean of all points in cluster C_j
9: **end for**
10: **until** no more changes in the assignment of points to clusters

K-means has garnered widespread popularity, giving rise to a plethora of variants. Among these, k-medians clustering [29] is a notable variant. In k-medians clustering, the conventional mean calculation is substituted with the median computation. This alteration has a distinct advantage: heightened resilience to the influence of outliers. Outliers, notorious for their potential to disrupt the computation of means, are effectively

attenuated by the adoption of k-medians. As a result, the robustness of k-medians clustering makes it a promising solution for datasets containing outliers.

3.2. K-Modes

K-modes clustering [3,4] is a variant of k-means that is specifically designed for datasets with categorical attributes, such as customer demographic data, product preferences, survey responses, etc. The k-modes algorithm partitions the data into a specified number of clusters based on the similarity of the categorical instances. Therefore, it allows for the discovery of natural groupings among categorical instances based on their shared attribute values.

Similar to k-means, k-modes starts by selecting initial cluster centroids. These centroids are initially chosen from the dataset. Then, each instance is assigned to the nearest centroid according to a distance metric for categorical data. The most commonly used distance metric for k-modes clustering is the Hamming distance, which calculates the percentage of attributes that differ between two instances. The Hamming distance is suitable for comparing categorical values, as it captures the proportion of attributes that differ between two instances. After assigning all instances to clusters, the centroids are updated. The new centroid of a cluster is determined as the mode (most frequent value) of each attribute within the cluster. Like k-means, the aforementioned steps are repeated iteratively until convergence. In each iteration, the instances are reassigned to the clusters based on the new centroids. The centroids are recalculated based on the instances within each cluster. The algorithm continues iterating until the assignments no longer change, indicating convergence.

Like k-means, the number of clusters is a parameter (k) that needs to be specified by the user before running the k-modes clustering. Similar to k-means, the selection of an appropriate value can be determined by the elbow method.

Let us assume the following dataset of categorical attributes representing people's preferences for colors and animals: {red, dog}, {blue, cat}, {green, dog}, {green, dog}, and {blue, cat}. Let as also assume that k-modes performs clustering with k = 2, and the k-modes starts with the initial centroids: {red, dog} and {blue, cat}. The instance {green, dog} is closer to the centroid {red, dog}, while the instance {blue, cat} is closer to the centroid {blue, cat}. Therefore, after the assignment step, the two clusters are C1 with {red, dog}, {green, dog}, and {green, dog}, and C2, with {blue, cat} and {blue, cat}. In C1, "green" is the most common preference for the color, and "dog" is the most frequent preference for the animal. In C2, "blue" and "cat" are the most common values. Thus, the new centroids (modes) are {green, dog} and {blue, cat}. The assignment step is repeated. The instances {red,dog} and {green, dog} are closer to the centroid {green, dog}. Also, {blue, cat} is closer to {blue, cat}. The two clusters are not modified. Thus, the algorithm stops. The final clusters are C1 with {red, dog}, {green, dog}, and {green, dog}, and C2 with {blue, cat} and {blue, cat}.

3.3. K-Prototypes

K-prototypes clustering [3,4] is also a variant of k-means that accommodates datasets with mixed data types, including both categorical and numeric attributes. While k-means works well for purely numeric data and k-modes works exclusively for categorical data, k-prototypes effectively handles datasets containing mixed data, providing a comprehensive and complete clustering solution that captures the structure of mixed data. In effect, k-prototypes combines the concepts of k-means for numeric attributes and k-modes for categorical attributes. Therefore, it aims to partition instances into clusters so that the dissimilarity for categorical attributes and the WCSS distances for numeric attributes are minimized.

K-prototypes combines the iterative assignment and update steps of both the k-modes and k-means algorithms. Thus, similar to k-modes and k-means, k-prototypes begins by initializing the cluster centroids. These centroids consist of both numeric and categorical

attribute values. For each instance, the algorithm calculates either the Hamming or the Euclidean distance, for categorical and numeric attributes, respectively, from each centroid. Then, the algorithm assigns the instance to the cluster with the nearest centroid. The update step includes the calculation of the new centroid for each cluster. For numeric attributes, k-prototypes computes the mean as in k-means. For categorical attributes, it determines the mode as in k-modes. Then, the algorithm repeats the steps of (a) assignment and (b) update iteratively, until the centroids are stabilized.

3.4. Elbow Method

The elbow method [5] is a well-known method which determines the optimal number of clusters in partition-based algorithms, such as the k-prototypes, k-modes and k-means. While the original application of the elbow method is for k-means, it can be extended to accommodate categorical and mixed data types that are prevalent in the k-prototypes and k-modes clustering.

As far as the k-means clustering is concerned, the elbow method involves plotting the WCSS (calculated using the formula presented in Section 3.1) against the number of clusters (k). As the number of clusters increases, the WCSS tends to decrease, capturing more variance. However, a point is reached where the decrease in WCSS starts to plateau, forming an "elbow" point on the plot. The "elbow" represents the point of diminishing returns, in terms of clustering performance, beyond which adding more clusters does not provide much additional benefit. Thus, the "elbow" point indicates the optimal number of clusters that balances capturing variance and avoiding overfitting and empowers k-means to discover compact and well-separated clusters.

To apply the elbow method to the k-modes and k-prototypes clustering, adjustments are made to the WCSS calculation to account for categorical and mixed data types. In the k-modes clustering, the WCSS is replaced by the Hamming distance, which quantifies the dissimilarity between categorical attribute values and their corresponding cluster modes by calculating the proportion of attributes that differ between two instances. For k-prototypes, which deals with mixed data, the WCSS incorporates the squared Euclidean and the Hamming distance, for numeric and categorical attributes, respectively.

Therefore, the elbow method remains consistent regardless of the clustering algorithm used. The method calculates the appropriate clustering measure (WCSS, Hamming or mixed distance), for each candidate number of clusters, and then plots the measure against the number of clusters. The number of clusters corresponding to the "elbow" point is considered the optimal one.

In Figure 1, the x-axis represents the number of clusters (k), while the y-axis represents the clustering measure (e.g., WCSS). As k increases, the measure decreases, but at $k = 3$, there is a noticeable bend in the plot. This bend represents the "elbow" point, which indicates that adding more clusters beyond this point does not provide significant improvements in clustering performance. In other words, increasing the number of clusters does not significantly decrease the clustering measure. In this case, the optimal number of clusters might be chosen as 3.

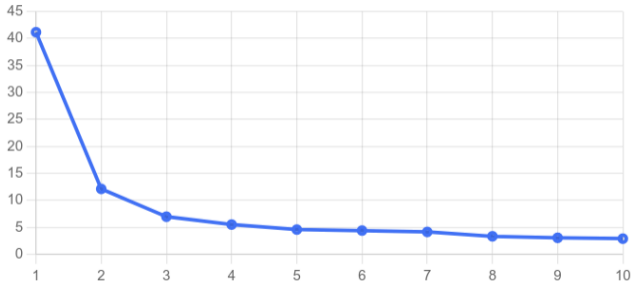
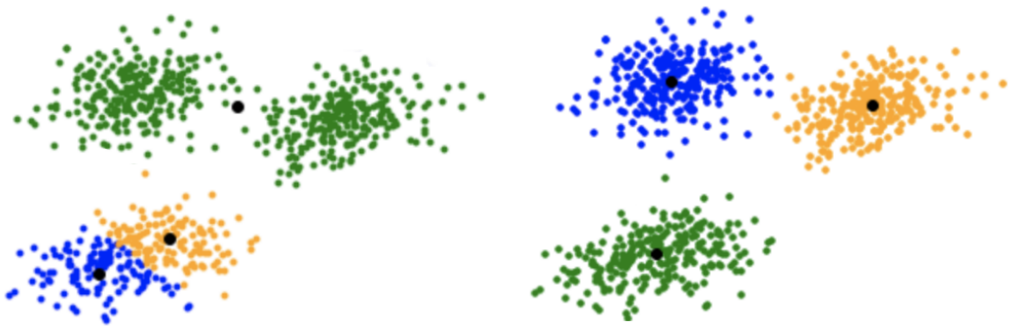


Figure 1. Example of an elbow method plot.

3.5. Initial Centroids Selection

The final clusters discovered by partitional-based algorithms are influenced by the randomly chosen initial centroids. As a consequence, the algorithms will yield varying clusters when applied to the same data using different initial cluster centroids. Therefore, the main challenge is that the choice of initial centroids can significantly impact the final clustering outcome. Poor initial centroids might lead to slow convergence or result in suboptimal cluster assignments. In contrast, efficient initial centroids may result in well-separated clusters. Figure 2 presents the result of a poor and an efficient centroids initialization.



(**a**) Poor initialization (**b**) Efficient initialization

Figure 2. Example of result of a poor and an efficient centroids initialization in k-means clustering.

K-means++ [30] is a smart initialization technique for k-means clustering. It helps to spread the initial centroids across the dataset, ensuring that they are not concentrated in a single area. By choosing centroids that are well distributed, k-means++ aims to provide a good starting point for the subsequent iterative assignment and update steps of k-means. K-means++ does so by probabilistically picking initial centroids that are farthest enough from each other. This strategic approach helps mitigate the common issue of k-means clustering being susceptible to suboptimal solutions due to inadequate initialization.

K-means++ starts by selecting an initial instance at random as the first centroid. Each subsequent centroid is then chosen based on their distance from previous centroids, with a higher probability of selecting instances that are farther apart. More specifically, the algorithm calculates the distance (usually the Euclidean distance) between each instance and its nearest existing centroid. Then, it assigns a probability to each instance, based on the square of its distance to the nearest existing centroid. The farther an instance is located from the existing centroids, the higher its probability of being chosen as the next centroid. K-means++ chooses the next centroid probabilistically, based on the calculated probabilities. The probability ensures that instances which are farther away have a higher chance of being selected as centroids. The algorithms terminates when the desirable number of centroids has been selected.

Similar to k-means++ which is designed to initialize the initial cluster centroids in the k-means clustering, the Cao method [31] is a common initialization technique used for k-modes clustering. Like k-means++, this method aims to select initial centroids that are well distributed across the dataset in terms of their Hamming distances. It helps in finding better starting points for the centroids, which in turn can lead to efficient clustering results.

The Cao method is quite similar to k-means++. However, it uses the Hamming distance. More specifically, the Cao method begins by selecting the first centroid randomly from the dataset. For each instance in the dataset, the method calculates the Hamming distance between the instance and the current centroid. Afterwards, the method calculates the probability for each instance to be selected as the next centroid. This probability is based on the distances calculated in the previous step. Instances that are farther away from the current centroid have a higher probability of being selected as the next centroid. The next centroid is selected probabilistically, based on the calculated probabilities. Instances with

higher probabilities are more likely to be chosen. The procedure is repeated until the desired number of initial centroids (k) is selected.

4. The kClusterHub Application

4.1. Description

As mentioned above, the contribution of this work is as follows:

- The development of kClusterHub, a user-friendly web application that allows users to upload datasets, utilizes the elbow method for finding the optimal number of clusters and uses partition-based clustering on any data types;
- kClusterHub is able to automatically select the appropriate clustering algorithm for the selected attributes of the dataset;
- The cluster assignments are presented in a tabular form and in data plots—they can be downloaded;
- The development of a free REST API which offers all operations to developers who wish to integrate kClusterHub's functionality into their own applications.

Following the architecture of WebApriori [28], kClusterHub includes three components: (i) the modules for the elbow method and partition-based clustering, (ii) a modern and user-friendly web interface and (iii) the back-end and the REST API service. The application is developed using open-source technologies and Git, with its source code publicly available on GitHub (https://github.com/KostisGrf/WebKmeans, accessed on 1 September 2023), while Python is used to code the modules of the first component. kClusterHub uses the k-means implementation, which is available in the scikit-learn library. Also, it uses the implementations of k-modes and k-prototypes which are available in Hammingkmodes library [32]. The Pandas [33] library is utilized for the datasets manipulation, and the kneed library [34] is used to obtain the number of clusters from the elbow graph. PHP is used for the development of the back-end, as well as the REST API. The composer library is used for the package management and PHPMailer for sending confirmation emails. A MySQL database is designed to manage users and their access levels (privileges). For the development of the front-end and the web interface, Javascript with the jQuery library, AJAX, and the Bootstrap framework are used. kClusterHub utilizes the server's file system to store the uploaded datasets. The REST API acts as the interface, where all the technologies can communicate with each other as shown in Figure 3.

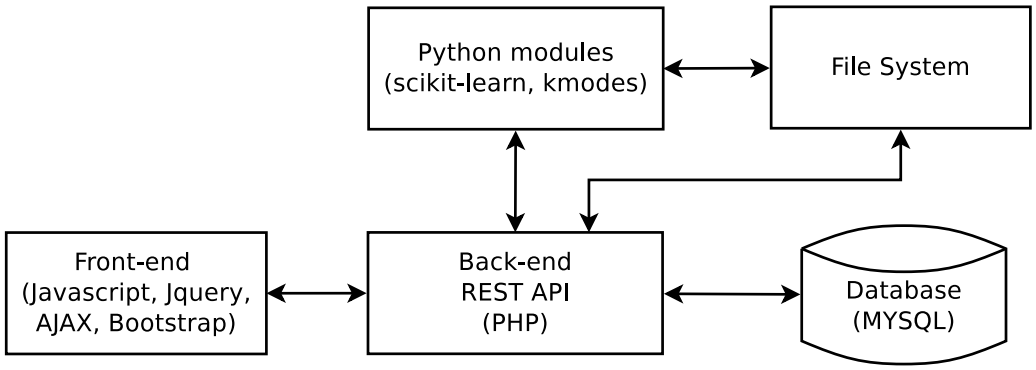


Figure 3. Application architecture.

As presented in Figure 4, in kClusterHub, registered users can upload and select the dataset, choose the attributes that will be used in the clustering, and finally execute the partition-based clustering by utilizing the appropriate centroid initialization method. Partition-based clustering is carried out by k-means, k-modes and k-prototypes, depending on the data types of the dataset's attributes. The application offers an auto mode, which enables kClusterHub to analyze the attributes and automatically choose the appropriate clustering algorithm. Therefore, kClusterHub exhibits a keen understanding of the nature of the dataset's attributes. Furthermore, kClusterHub utilizes the k-means++ initialization

method, when the clustering applied on numerical data, and the Cao initialization method, when clustering is applied on categorical or mixed data. Optimally, kClusterHub displays the "elbow diagram" and recommends the optimal number of clusters. The auto mode for automatic algorithm selection, as well as the automatic suggestion of the k parameter, through the elbow method, constitute the AutoML capabilities, and both enhance the intelligent decision making in the clustering process. The result of the clustering task is the cluster assignment of each instance in an exploratory plot and a tabular form. The elbow graph, the exploratory plot and the dataset with the assigned clusters can be downloaded for further use and processing.

It is worth noting that the k-modes Python library is not as fast as the scikit-learn library. In practice, the elbow method requires running the clustering algorithm multiple times, equal to the maximum number of clusters defined by the user. Our experiments showed that applying the elbow method for k-modes and k-prototypes takes significantly more time compared to k-means. Consequently, when running k-modes or k-prototypes on datasets with more than 1500 instances, kClusterHub employs a strategy of random sampling and applies the elbow method to the sampled data to enhance speed and efficiency.

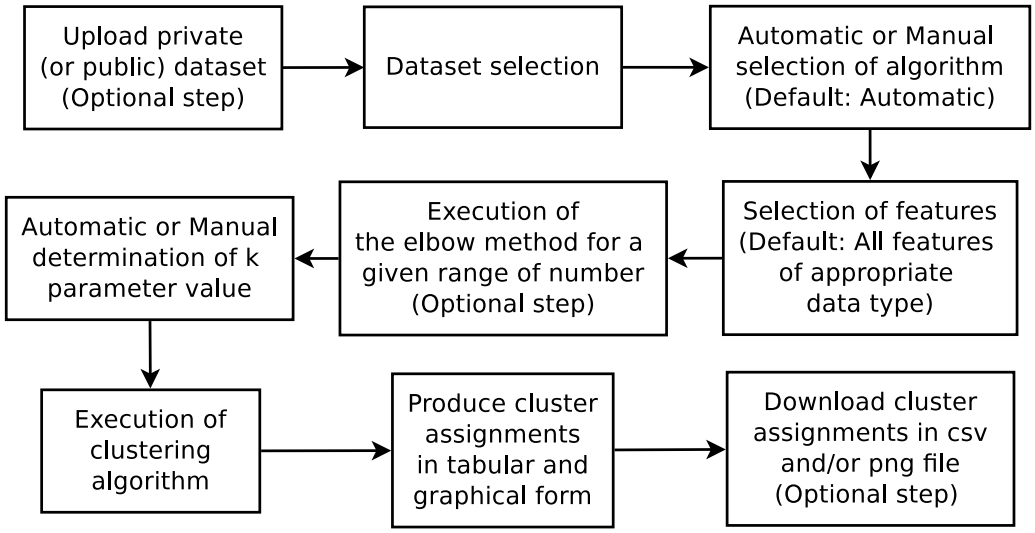


Figure 4. Pipeline diagram of tasks in kClusterHub.

Also, kClusterHub provides a REST API that allows developers to submit their data, execute the elbow method and partition-based clustering and obtain the cluster assignments from their own applications, by incorporating simple API calls into them. The REST API is an advantageous feature of kClusterHub, expanding its usability beyond the web-based interface and allowing for more efficient and automated usage of partition-based clustering.

kClusterHub is a user-centric application that mandates user registration and login as essential components of its functionality. This login system ensures a secure experience for users, enabling them to access the capabilities offered by the platform. The datasets uploaded by the registered users can be designated as either public or private, each carrying distinct accessibility characteristics. Private datasets are confined to the user who initiates the upload, ensuring a restricted scope of access. In contrast, public datasets are open to all registered users of kClusterHub, though exclusive upload privileges are extended solely to users holding advanced status. This facet holds particular significance for educators who aspire to impart datasets to their students for educational purposes.

The application introduces a hierarchy of user roles (see Figure 5), comprising three distinct tiers: (i) simple users, (ii) public dataset creators, and (iii) administrators. Simple users encompass individuals with the ability to upload private datasets and leverage the suite of kClusterHub features. Additionally, they can use public datasets contributed by public dataset creators. The public dataset creators have the same attributes as those of simple users, while augmenting the capability to upload public datasets. This role caters

to those with a proclivity for sharing data and insights. The higher tier of the hierarchy, occupied by administrators, wields the authority to elevate user accounts from the status of simple users to that of public dataset creators. This administrative control facilitates management in kClusterHub.

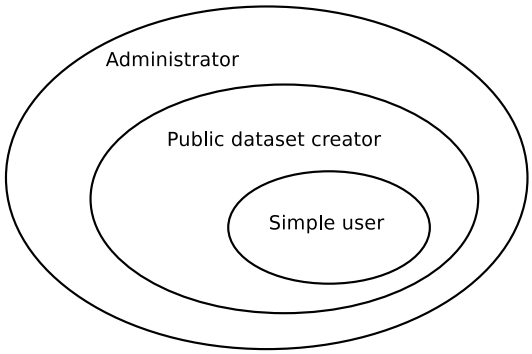


Figure 5. Registered users hierarchy.

4.2. Web Interface

To use kClusterHub, users must first register and confirm their email. Then, they can log in. The main page is divided into three distinct sections. Through the first section, registered users are able to seamlessly upload their private or public datasets in csv, xls, or xlsx format (see Figure 6). All registered users can select an uploaded dataset among the available private and public datasets (see Figure 7) and, of course, can download or delete it (assuming that the user has the necessary permissions). Then, kClusterHub presents the selected dataset in tabular form (see Figure 8). Also, the names of the dataset's attributes are displayed in check boxes, and this feature enables users to select the attributes that will be the basis for their clustering analysis (see Figure 9). The default mode is to consider all the appropriate attributes. However, the users can uncheck the attributes they want to be ignored during the clustering process.

The application offers the k-means, k-modes and k-prototypes clustering for numerical, categorical and mixed data, respectively. Through the first section of the interface, the registered users have the option to either manually select their desired algorithm or opt for the auto mode. The auto mode harnesses the power of automation to determine the most suitable algorithm for the dataset selected. When the dataset comprises only numerical data, k-means with the k-means++ initialization is automatically chosen. Conversely, if only categorical data exist, k-modes is used. K-prototypes is adopted when the dataset has both categorical and mixed data. In cases of k-modes and k-prototypes, the Cao method for centroid initialization is used. On the other hand, if a user selects the desired algorithm (see Figure 10), kClusterHub presents the attributes that can be used, and the user selects among them (see Figure 9). For instance, if the selected dataset has mixed attributes and k-means is selected, the numerical attributes are presented for consideration. Similarly, opting for k-modes exclusively showcases categorical attributes. In essence, ClusterHub's auto mode offers an effortless approach, while the manual selection of an algorithm affords users the control to shape their analysis based on the available attributes.

Subsequently, kClusterHub orchestrates the partition-based clustering process using the appropriate centroid initialization method. The second section of the interface concerns the elbow method. The user may use the elbow method for finding the optimal number of clusters (see Figure 11) by entering the maximum number of clusters, *max*. The application utilizes the selected clustering algorithm *max* times and estimates the clustering metric (WCSS, Hamming or mixed distance) for each time. Then, by clicking "Get elbow chart", kClusterHub presents the elbow diagram which portrays the trade-off between the number of clusters and the clustering metric aiding users in pinpointing the optimal number of clusters. In addition, kClusterHub, based on the elbow graph, suggests the appropriate number of clusters in the range from 2 to *max* value provided.

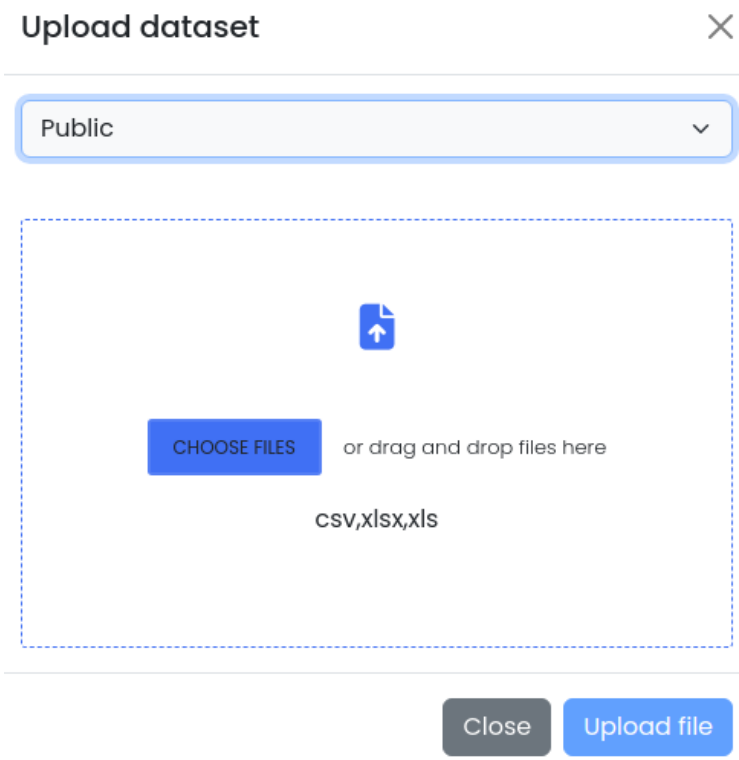


Figure 6. Dataset upload screen.

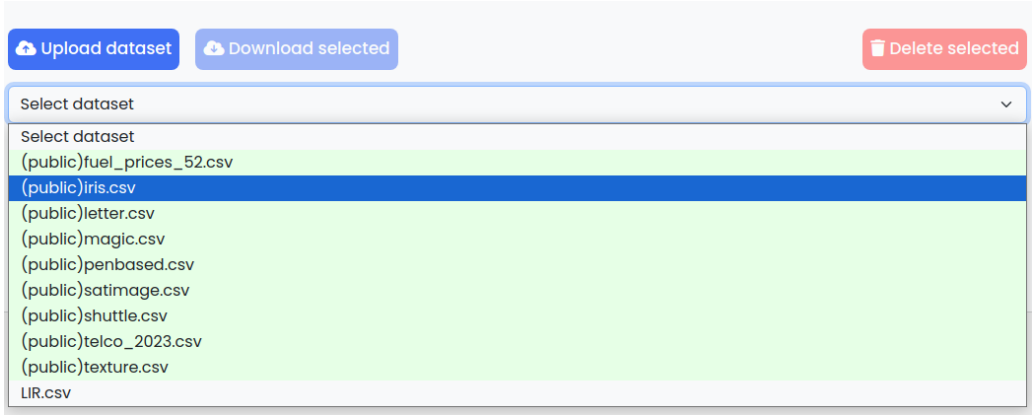
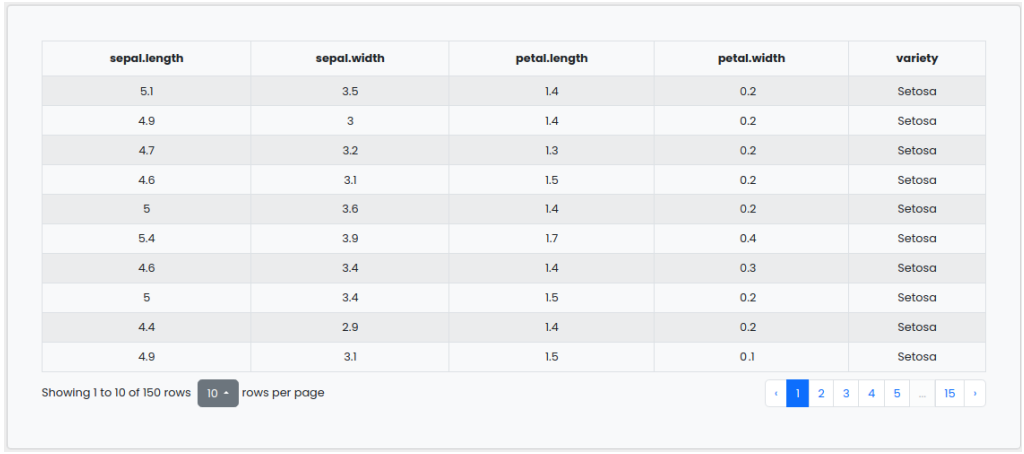


Figure 7. Dataset selection.



sepal.length	sepal.width	petal.length	petal.width	variety
5.1	3.5	1.4	0.2	Setosa
4.9	3	1.4	0.2	Setosa
4.7	3.2	1.3	0.2	Setosa
4.6	3.1	1.5	0.2	Setosa
5	3.6	1.4	0.2	Setosa
5.4	3.9	1.7	0.4	Setosa
4.6	3.4	1.4	0.3	Setosa
5	3.4	1.5	0.2	Setosa
4.4	2.9	1.4	0.2	Setosa
4.9	3.1	1.5	0.1	Setosa

Figure 8. Selected dataset in the tabular form.

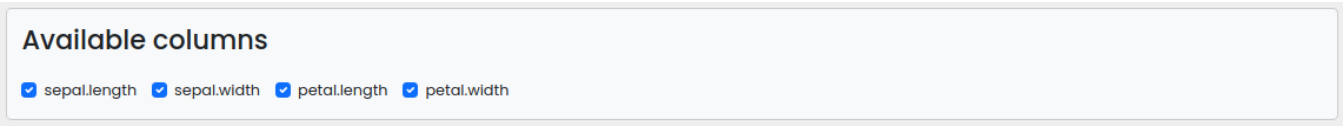


Figure 9. Attributes selection.

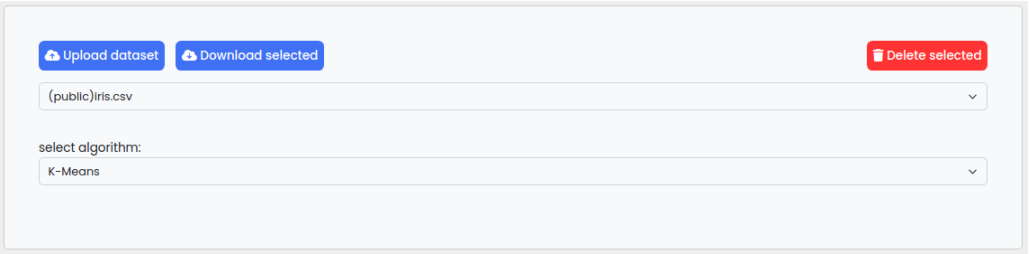


Figure 10. Selection of k-means on mixed data (four numerical attributes and one categorical).

Figure 11. Elbow graph.

The last section of the interface requires users to input their desired number of clusters. This can be either the number suggested from the elbow method or chosen independently by the user. Upon clicking the "Get cluster assignment" button, the selected clustering algorithm runs and a table showcasing the data with the corresponding cluster assignments is presented (see Figure 12). For a visual exploration of the cluster characteristics, users can also opt for the exploratory data plots (see Figure 13). Moreover, users have the option to download the table in CSV format, using the "Download CSV" button. Both the elbow graph and the exploratory data plot are available for download as well.

It is worth mentioning that kClusterHub was deployed in a Apache web server at the Department Information and Electronic Engineering of the International Hellenic University (https://kclusterhub.iee.ihu.gr, accessed on 3 September 2023). It is free and open source, and therefore users can access the source code on Github and deploy it on their own server.

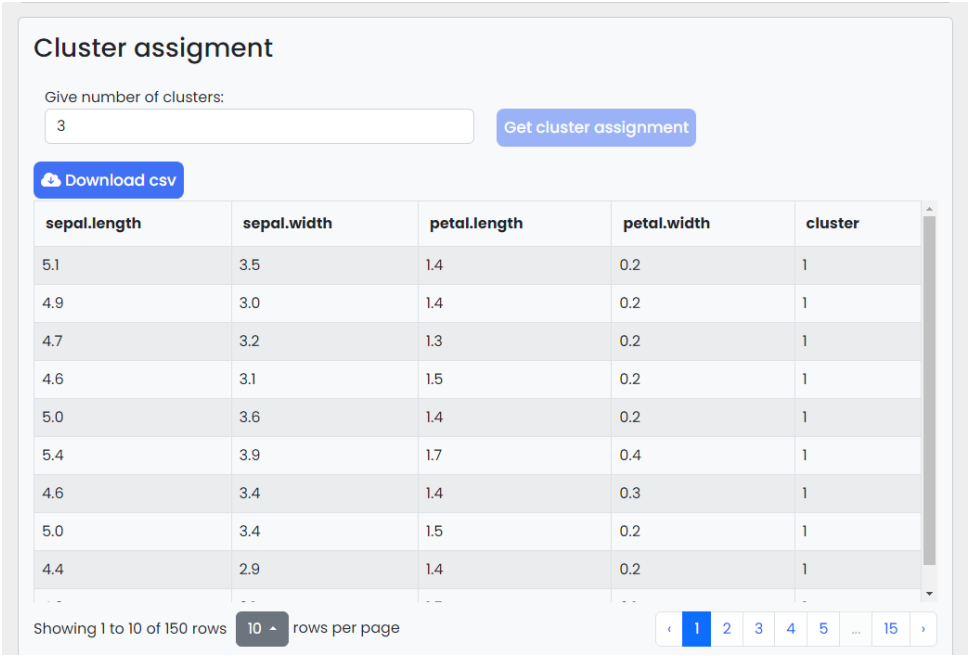
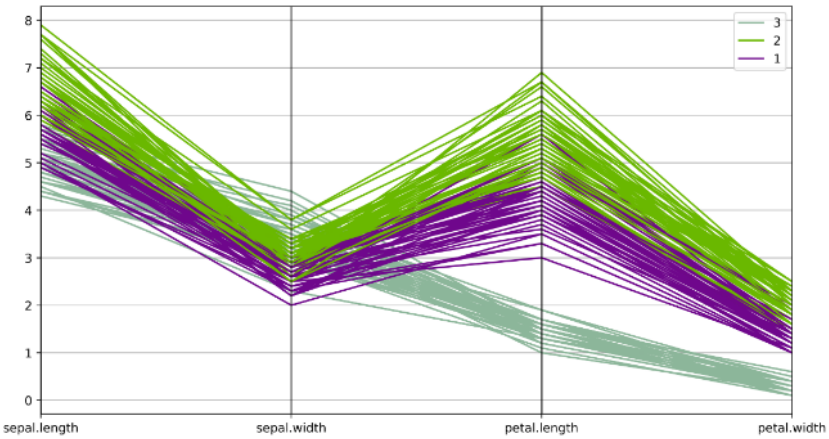
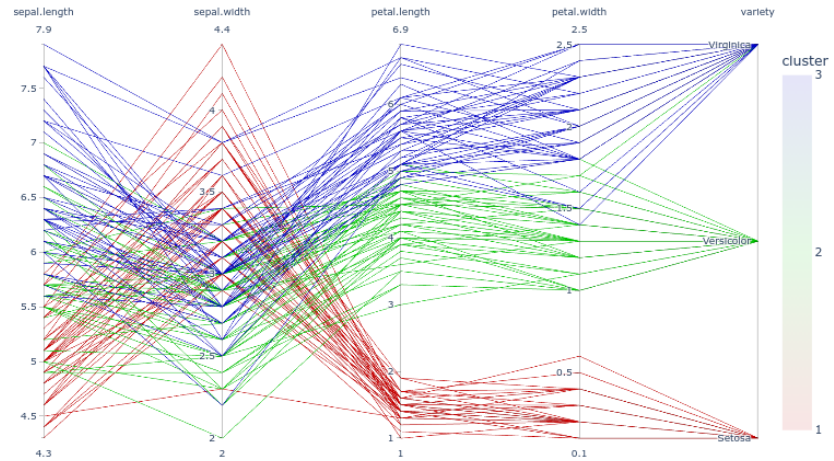


Figure 12. Cluster assignments in tabular form.



(**a**) Dataset with numerical attributes



(**b**) Dataset with attributes of mixed data types

Figure 13. Cluster assignments in exploratory plots.

4.3. Web Service

The kClusterHub web service has been crafted as a REST API, boasting eleven endpoints that expose the kClusterHub's capabilities to external applications via straightforward HTTP requests. To use this service, users/programmers are required to complete a registration process and obtain a personalized API key, which then acts as their ticket to access the full array of functionalities. The integration process is efficient, as every HTTP request is accompanied by the user's API key, ensuring a secure and personalized experience.

Table 1 lists the eleven endpoints that the REST API includes. Among these, four endpoints (Nos. 8, 9, 10 and 11) are for user account management, encompassing essential tasks, such as registration, login, account editing and deletion. Additional four endpoints are allocated for dataset management tasks (Nos. 1, 2, 6 and 7). These tasks concern various operations, including dataset upload, retrieval, deletion, and the ability to access dataset attributes and characteristics.

The remaining triad of endpoints constitutes the major part of the API since they concern the clustering process. One of these key endpoints (No. 3) executes the elbow method, delivering clustering metrics (such as WCSS) and suggesting the optimal number of clusters. Another pivotal endpoint (No. 4) triggers the clustering algorithm, furnishing users with cluster assignments for their data. The last endpoint (No. 5) generates the exploratory plot for cluster assignments. All requests and responses are conducted in JSON format, facilitating data exchange between the web service and external applications.

Table 1. REST API endpoints.

No.	HTTP Method	Endpoint
1	GET	/api/get-datasets.php?apikey={apikey}
2	GET	/api/read-dataset.php?apikey={apikey}&dataset={dataset}
3	POST	/api/elbow.php
4	POST	/api/clusters.php
5	POST	/api/parallel-cords.php
6	POST	/api/upload-dataset.php
7	DELETE	/api/delete-dataset.php
8	POST	/api/register.php
9	POST	/api/login.php
10	POST	/api/edit-profile
11	POST	/api/delete-user.php

A few core examples of endpoint calls, along with their corresponding responses are examined hereupon.

The endpoint with No. 1 retrieves the datasets' names, which the user can use. Below, an example of response of the http GET call is presented.

get-datasets.php?apikey=905a7d7f02b2512fcf3e02b1f6ec52ab

```
JSON response:
{"personal_datasets":["LIR.csv"],
 "public_datasets":["fuel_prices_52.csv","iris.csv",
                    "letter.csv","magic.csv","penbased.csv",
                    "satimage.csv","shuttle.csv",
                    "telco_2023.csv","texture.csv"
                    ]
}
```

The following request concerns the POST endpoint with No 3. It executes the elbow method on the columns 'Age' and 'Salary' of the 'data.csv' dataset. This dataset is private and belongs to the user who possesses the API key: 0a8366a07c0d8fccx48bab2e657f12d0. The elbow method will operate within the range of 1 to 10 clusters. The algorithm value for this request is set to 'auto', enabling the algorithm to be automatically chosen, based on the column types. Alternatively, the algorithm value can also be set to 'k-means', 'k-modes', or 'k-prototypes', allowing the explicit selection of the algorithm. In the case of the example, since both columns are numerical, k-means is used and the corresponding response encompasses the WCSS measurements for various k values, spanning from 1 to 10, along with the recommended optimal number of clusters.

```
JSON request:
{"dataset":"data.csv",
 "dataset-type":"personal",
 "clusters":"10",
 "columns":["Age","Salary"],
 "algotithm":"auto",
 "apikey":"0a8366a07c0d8fccx48bab2e657f12d0"
}
JSON response:
{"metric":["41.2","12.1","6.9","5.5","4.6","4.7","3.8","3.2","3.3","2.7"],
"suggested-k":"3"
}
```

The following request concerns POST endpoint with No 4. It executes k-means on the "sepal.length", "sepal.width", "petal.length", and "petal.width" columns of the "iris.csv" dataset. The dataset is public, and the user is associated with the API key: 905a7d7f02b2512fcf3e02b1f6ec52ab. The k-means algorithm will discover three clusters. The response contains the cluster assignments for each instance, along with the corresponding attribute values and cluster identifiers. Of course, since all selected columns are numeric, k-means would be used even if the auto algorithm was selected.

```
JSON request:
{ "dataset": "iris.csv", "dataset-type": "public",
  "clusters": "3",
  "columns": ["sepal.length", "sepal.width", "petal.length", "petal.width"],
  "algorithm":"k-means"
  "apikey": "905a7d7f02b2512fcf3e02b1f6ec52ab"
}
JSON response:
{  "items": [
    {"sepal.length": "5.1", "sepal.width": "3.5",
     "petal.length": "1.4", "petal.width": "0.2",
     "cluster": "1"},
    {"sepal.length": "4.9", "sepal.width": "3.0",
     "petal.length": "1.4", "petal.width": "0.2",
     "cluster": "2"},
    // ... (remaining instances)
  ]
}
```

The user interface of kClusterHub encompasses a dedicated web page, designed to guide users through the utilization of the eleven endpoints, all of which are accessible with the user's API key. On this informative web page, users are provided with detailed instructions on how to interact with these endpoints effectively. Moreover, the page serves as a valuable resource by showcasing examples of potential HTTP requests complemented by the corresponding

responses they can expect to receive. This interface serves to enable users with the knowledge and practical insights required to utilize the full capabilities of kClusterHub's API.

5. System Evaluation

5.1. CPU Time Measurements

Addressing the resource-intensive nature of partition-based clustering, particularly in terms of CPU and RAM utilization, necessitated the strategic integration of the Pandas library. By leveraging this library, kClusterHub adeptly manages the intricacies associated with processing large datasets. It is important to note that the Python scikit-learn and kmodes libraries were pivotal in facilitating kClusterHub. The experiments were carried out on a virtual machine running Debian GNU/Linux. The virtual machine was equipped with a 6-core CPU operating at 2.5 GHz and 4 GB of RAM. This virtual machine was hosted by a Citrix-hypervisor virtualization server.

To evaluate the efficacy of kCusterHub, an empirical analysis was conducted utilizing twelve diverse datasets sourced from both the keel dataset repository [35] (https://sci2s.ugr.es/keel/datasets.php, accessed on 10 October 2023) and Kaggle (https://www.kaggle.com/datasets, accessed on 10 October 2023). The experimental assessments encompassed the application of the elbow method to identify the optimal number of clusters in the range [1–20], alongside the execution of the partition-based clustering algorithms employing the recommended number of clusters derived from the Elbow method.

Considering the experimental results, as presented in Table 2, it becomes evident that the execution times are linked to the size (rows and columns) of each dataset. It is noteworthy that the elbow method's execution entails a more time-intensive process, compared to the final execution of the partition-based algorithm. This discrepancy can be attributed to the fact that the elbow method necessitates multiple clustering tasks (ranging from k = 1 to k = 20 in our experimentation), ultimately computing the appropriate clustering metric (WCSS, Hamming or mixed distance) for each clustering iteration.

Table 2. CPU time measurements.

Dataset	Size (Kb)	Number of Rows	Number of Numerical Columns	Number of Categorical Columns	Algorithm	Time for Elbow (k = 20)	Suggested k	Time for Clustering
penbased	538	10,992	16	0	k-means	1.06 s	6	0.24 s
letter	716	20,000	16	0	k-means	2.49 s	7	0.35 s
magic	1462	19,020	10	0	k-means	1.67 s	5	0.23 s
texture	1495	5500	40	0	k-means	1.02 s	4	0.45 s
shuttle	1559	57,999	9	0	k-means	3.08 s	5	0.31 s
poker	24,563	1,025,009	10	0	k-means	72 s	6	23.25 s
car	53	1728	0	7	k-modes	36.5 s	5	1.5 s
bank	450	4521	0	10	k-modes	55.6 s	4	8.95 s
nursery	1020	12,960	0	9	k-modes	25.2 s	6	14.8 s
marketing Data	222	2240	7	3	k-prototypes	188.9 s	6	18.5 s
germanCC	80	1000	7	13	k-prototypes	320.6 s	6	11.4 s
abalone	192	4174	7	1	k-prototypes	116.5 s	3	18.8 s

As mentioned before, the k-modes and k-prototypes algorithms, which are part of the k-modes python library, appear to have slower performance compared to scikit-learn's k-means algorithm. This performance disparity is particularly noticeable when working with large datasets. To address this issue, we adopt a tactic of utilizing a smaller sample subset of the data when applying the elbow method with k-modes and k-prototypes. By doing so, we

are able to alleviate the computational burden associated with the elbow method for these algorithms, enabling us to effectively explore a wider range of potential cluster numbers.

An illuminating illustration arises from the Poker dataset, a notably large dataset comprising over a million rows. In this case, the elbow method's execution spans more than a minute, whereas k-means clustering concludes within a mere 24 s. To ameliorate the performance of measurements gleaned from these extensive datasets, a logical consideration lies in migrating kClusterHub to a more potent computational infrastructure. This strategic move holds the potential to significantly enhance the efficiency and processing capabilities of the system, consequently refining the outcomes achieved from handling such voluminous data.

5.2. Usability Testing

The evaluation of kClusterHub's usability employs the widely recognized System Usability Scale (SUS) questionnaire [36] (https://forms.gle/dUSecKe1pgES661z8, accessed on 15 September 2023). More specifically, SUS was used to measure the users' overall satisfaction with kClusterHub, as well as their perception of its effectiveness, efficiency, and ease of use. SUS is a widely used tool for measuring the usability of web applications. It consists of ten questions that participants complete to rate their experience. The questions in the questionnaire are as follows:

1. I think that I would like to use this website frequently;
2. I found the website unnecessarily complex;
3. I thought the website was easy to use;
4. I think that I would need the support of a technical person to be able to use this website;
5. I found the various functions in this website were well integrated;
6. I thought there was too much inconsistency in this website;
7. I would imagine that most people would learn to use this website very quickly;
8. I found the website very cumbersome to use;
9. I felt very confident using the website;
10. I needed to learn a lot of things before I could get going with this website.

Participants rate each question on a 5-point Likert scale, ranging from "strongly disagree" (1) to "strongly agree" (5). Calculating the SUS score involves subtracting 1 from the scores of odd-numbered questions and subtracting their value from 5 for even-numbered questions. The resulting scores are then summed and multiplied by 2.5 to yield a final score between 0 and 100. A SUS score of 80 or above is typically considered excellent.

A total of 37 respondents, primarily computer science undergraduate students enrolled in a data mining course, completed the questionnaire upon our request. The distribution of responses for each rating range is provided in Table 3. The resulting SUS score was 83.58, indicating a high level of user satisfaction with the kClusterHub experience. These results affirm the positive usability of kClusterHub as perceived by its users.

Table 3. Results of SUS questionnaire.

Question	1	2	3	4	5
1.	0	2	4	17	14
2.	23	11	2	0	1
3.	0	0	1	7	29
4.	23	7	3	4	0
5.	0	0	4	11	22
6.	27	7	2	1	0
7.	2	0	4	9	22
8.	16	9	9	2	1
9.	0	0	1	14	22
10.	11	10	11	4	1

6. Conclusions and Future Work

In this paper, we introduced kClusterHub, a versatile and user-friendly web-based application designed to streamline the process of partition-based clustering. Our platform bridges the gap between clustering algorithms and the end users by providing an intuitive interface that enables the easy configuration and execution of clustering tasks. By incorporating algorithms, such as k-means, k-prototypes, and k-modes, along with an intelligent auto-algorithm selection feature and efficient centroid initialization methods, such as k-means++ for numerical data and the Cao method for categorical and mixed data, kClusterHub is able to manage various data types and perform effortless partition-based cluster analysis. The integration of the elbow method for optimal number of clusters autodetermination further enhances the system's decision-making capabilities. The presentation of cluster assignments, in both tabular and exploratory plot formats, facilitates insightful data interpretation. Through empirical evaluations, we showcased the platform's efficiency and efficacy across diverse datasets.

Moreover, kClusterHub extends its functionality beyond the web interface by offering a REST API. This API empowers developers to seamlessly integrate partition-based clustering tasks into their own applications. Through straightforward API calls, developers can effortlessly submit their dataset, trigger the execution of the elbow method and subsequent partition-based clustering, and retrieve the resulting cluster assignments. This seamless integration allows developers to leverage the capabilities of partition-based clustering within their applications, providing them with a versatile tool for data analysis. By bridging the gap between the kClusterHub platform and external applications, the REST API offers ways for incorporating clustering techniques into a wider array of use cases, from custom data analytics workflows to automated decision-making processes.

In addition to assessing the performance of kClusterHub through CPU measurements, the usability of the platform was also evaluated using the SUS questionnaire. The CPU measurements demonstrated the efficiency of kClusterHub's algorithmic implementations. For the usability evaluation, participants, mainly computer science students, were requested to complete the SUS questionnaire. The results revealed a high SUS score, indicating excellent user satisfaction with kClusterHub's web interface and functionality. These findings underscore the effective performance and user-friendly nature of kClusterHub, making it a valuable tool for partition-based cluster analysis.

Our future directions are centered around two key aspects. The first is the integration of density-based clustering (DBSCAN [37]) and hierarchical clustering [38] into kClusterHub. These additions will substantially improve the versatility of the application, enabling users to perform a broader range of data analysis tasks. Also, we plan to optimize the platform's performance by implementing parallelization techniques. By harnessing the power of parallel processing, we aim to significantly expedite the clustering process, particularly for larger datasets. Through these extensions, we aspire to solidify kClusterHub as a complete, efficient tool for effortless data clustering.

Author Contributions: Conceptualization, K.G., S.O. and D.M.; methodology, K.G., S.O. and D.M.; software, K.G. and S.O.; validation, K.G., S.O. and D.M.; formal analysis, K.G., S.O. and D.M.; investigation, K.G., S.O. and D.M.; resources, K.G., S.O. and G.E.; data curation, K.G.; writing—original draft preparation, K.G., S.O. and D.M.; writing—review and editing, K.G., S.O. and D.M.; visualization, K.G., S.O. and D.M.; supervision, S.O. and D.M.; project administration, K.G., S.O. and D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Publicly available datasets were analyzed in this study. These data can be found here: https://sci2s.ugr.es/keel/datasets.php, accessed on 10 October 2023.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Aggarwal, C.C.; Reddy, C.K. *Data Clustering: Algorithms and Applications*, 1st ed.; Chapman & Hall/CRC: Boca Raton, FL, USA, 2013.
2. MacQueen, J.B. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 21 June–18 July 1965 and 27 December 1965–7 January 1966; University of California Press: Berkeley, CA, USA, 1967; Voulme 1, pp. 281–297.
3. Huang, Z. Clustering Large Data Sets with Mixed Numeric and Categorical Values. In Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conference, Singapore, 23–24 February 1997; pp. 21–34.
4. Huang, Z. Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Min. Knowl. Discov.* **1998**, *2*, 283–304. [CrossRef]
5. Kodinariya, T.M.; Makwana, P.R. A review on the Elbow method in clustering. *Int. J. Comput. Appl.* **2013**, *1*, 97–100.
6. Frank, E.; Hall, M.A.; Holmes, G.; Kirkby, R.; Pfahringer, B.; Witten, I.H. Weka: A machine learning workbench for data mining. In *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*; Maimon, O., Rokach, L., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 1305–1314.
7. Demšar, J.; Curk, T.; Erjavec, A.; Gorup, Č.; Hočevar, T.; Milutinovič, M.; Možina, M.; Polajnar, M.; Toplak, M.; Starič, A.; et al. Orange: Data Mining Toolbox in Python. *J. Mach. Learn. Res.* **2013**, *14*, 2349–2353.
8. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
9. Szepannek, G. clustMixType: User-Friendly Clustering of Mixed-Type Data in R. *R J.* **2018**, *10*, 200–208. [CrossRef]
10. Gratsos, K.; Ougiaroglou, S.; Margaris, D. A Web Tool for K-means Clustering. In *Proceedings of the Novel and Intelligent Digital Systems: Proceedings of the 3rd International Conference (NiDS 2023)*; Kabassi, K., Mylonas, P., Caro, J., Eds.; Springer: Cham, Switzerland, 2023; pp. 91–101.
11. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I.H. Weka—A machine learning workbench for data mining. In *Data Mining and Knowledge Discovery Handbook*; Springer: Boston, MA, USA, 2009; pp. 1269–1277.
12. He, X.; Zhao, K.; Chu, X. AutoML: A survey of the state-of-the-art. *Knowl.-Based Syst.* **2021**, *212*, 106622. [CrossRef]
13. Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J.; Blum, M.; Hutter, F. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2015; pp. 2755–2763.
14. LeDell, E.; Poirier, S. H_2O AutoML: Scalable Automatic Machine Learning. In Proceedings of the 7th ICML Workshop on Automated Machine Learning (AutoML), Vienna, Austria, 12–18 July 2020.
15. Amazon Web Services, Amazon SageMaker. 2023. Available online: https://aws.amazon.com/sagemaker/ (accessed on 2 October 2023).
16. Maechler, M.; Rousseeuw, P.; Struyf, A.; Hubert, M.; Hornik, K. *Cluster: Cluster Analysis Basics and Extensions*; R Package Version 2.1.4; R Foundation for Statistical Computing: Vienna, Austria, 2022.
17. Brock, G.; Pihur, V.; Datta, S.; Datta, S. clValid: An R package for cluster validation. *J. Stat. Softw.* **2008**, *25*, 1–22. [CrossRef]
18. Lin, H.; Liu, H.; Wu, J.; Li, H.; Günnemann, S. Algorithm xxxx: KCC: A MATLAB Package for K-means-based Consensus Clustering. *ACM Trans. Math. Softw.* **2023**. [CrossRef]
19. Berthold, M.R.; Cebron, N.; Dill, F.; Gabriel, T.R.; Kötter, T.; Meinl, T.; Ohl, P.; Thiel, K.; Wiswedel, B. KNIME: The Konstanz Information Miner. *Stud. Classif. Data Anal. Knowl. Organ.* **2008**, *1*, 319–326.
20. Chaouch, C.; Sahbudin, M.A.B.; Scarpa, M.; Serrano, S. Audio fingerprint database structure using k-modes clustering. *J. Adv. Res. Dyn. Control Syst.* **2020**, *12*, 1545–1554. [CrossRef]
21. Chadha, A.; Kumar, S. Extension of K-modes algorithm for generating clusters automatically. *Int. J. Inf. Technol. Comput. Sci. (IJITCS)* **2016**, *8*, 51–57. [CrossRef]
22. Jiang, F.; Liu, G.; Du, J.; Sui, Y. Initialization of K-modes clustering using outlier detection techniques. *Inf. Sci.* **2016**, *332*, 167–183. [CrossRef]
23. Kacem, M.A.B.H.; N'cir, C.E.B.; Essoussi, N. MapReduce-based k-prototypes clustering method for big data. In Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Paris, France, 19–22 October 2015; pp. 1–7.
24. Jia, Z.; Song, L. Weighted k-prototypes clustering algorithm based on the hybrid dissimilarity coefficient. *Math. Probl. Eng.* **2020**, *2020*, 5143797 . [CrossRef]
25. Kuo, T.; Wang, K.J. A hybrid k-prototypes clustering approach with improved sine-cosine algorithm for mixed-data classification. *Comput. Ind. Eng.* **2022**, *169*, 108164. [CrossRef]
26. Sangam, R.S.; Om, H. An equi-biased k-prototypes algorithm for clustering mixed-type data. *Sādhanā* **2018**, *43*, 37. [CrossRef]
27. Jang, H.J.; Kim, B.; Kim, J.; Jung, S.Y. An efficient grid-based k-prototypes algorithm for sustainable decision-making on spatial objects. *Sustainability* **2018**, *10*, 2614. [CrossRef]
28. Malliaridis, K.; Ougiaroglou, S.; Dervos, D.A. WebApriori: A Web Application for Association Rules Mining. In *Proceedings of the Intelligent Tutoring Systems, Proceedings of the 16th International Conference, ITS 2020, Athens, Greece, 8–12 June 2020*; Kumar, V., Troussas, C., Eds.; Springer: Cham, Switzerland, 2020; pp. 371–377.
29. Sengupta, J.S.; Auchter, R.F. A *k*-medians clustering algorithm. *Appl. Stat.* **1990**, *39*, 67–79. [CrossRef]

30. Arthur, D.; Vassilvitskii, S. K-Means++: The Advantages of Careful Seeding. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '07, New Orleans, LA, USA, 7–9 January 2007; pp. 1027–1035.
31. Cao, F.; Liang, J.; Bai, L. A data clustering algorithm for mixed data. *Pattern Recognit.* **2009**, *42*, 1855–1864.
32. de Vos, N.J. Kmodes Categorical Clustering Library. 2015–2021. Available online: https://github.com/nicodv/kmodes (accessed on 3 September 2023).
33. McKinney, W. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2010; van der Walt, S., Millman, J., Eds.; pp. 56–61. [CrossRef]
34. Satopaa, V.; Albrecht, J.; Irwin, D.; Raghavan, B. Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior. In Proceedings of the 2011 31st International Conference on Distributed Computing Systems Workshops, Minneapolis, MN, USA, 20–24 June 2011; pp. 166–171.
35. Alcalá-Fdez, J.; Fernández, A.; Luengo, J.; Derrac, J.; García, S.; Sánchez, L.; Herrera, F. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Mult.-Valued Log. Soft Comput.* **2011**, *17*, 255–287.
36. Brooke, J. SUS: A "quick and dirty" usability scale. *Usability Eval. Ind.* **1996**, *189*, 4–7.
37. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, Portland, OR, USA, 2–4 August 1996; AAAI Press: Washington, DC, USA, 1996; pp. 226–231.
38. Zepeda-Mendoza, M.L.; Resendis-Antonio, O. Hierarchical Agglomerative Clustering. In *Encyclopedia of Systems Biology*; Springer: New York, NY, USA, 2013; pp. 886–887. [CrossRef]