Information Sciences 663 (2024) 120282

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A comprehensive framework for explainable cluster analysis

Miguel Alvarez-Garcia [a,*], Raquel Ibar-Alonso [b], Mar Arenas-Parra [a]

[a] Department of Quantitative Economics, University of Oviedo, Oviedo, Spain
[b] Department of Applied Economics I, Rey Juan Carlos University, Madrid, Spain

ARTICLE INFO

Dataset link: https://doi.org/10.17632/hwj6bfb899.1

Dataset link: https://github.com/malgar/clust-learn

Keywords:
Explainable cluster analysis
Knowledge extraction
Data preprocessing
Dimensionality reduction
Classification
Mixed-type data

ABSTRACT

Machine learning has proven to be a powerful tool for knowledge extraction from large data sets across different domains. Data quality and results interpretability are essential when applying machine learning to inform decision-making processes. This is especially true for clustering methods, which are frequently employed for extracting knowledge from large data sets, due to their unsupervised nature. Although there are significant recent developments in explainable artificial intelligence (XAI) applied to unsupervised problems, they focus primarily on cluster interpretability and often overlook data quality challenges. Moreover, these developments are typically designed to use specific clustering algorithms, limiting their adaptability to incorporate alternative techniques. We propose a novel and comprehensive four-step sequential framework for explainable cluster analysis on high-dimensional mixed-type data to address these limitations. The framework encompasses data preprocessing, dimensionality reduction, clustering, and classification to ensure robust and explainable results. The proposed methodology has also been implemented in an open-source Python package called Clust-learn, designed to be accessible and customizable for researchers and practitioners. The framework has been validated by applying a case study focusing on large-scale assessments in education, effectively illustrating the strength and usefulness of the methodology in extracting and synthesizing knowledge from complex real-world data.

1. Introduction

Extracting information and knowledge from large data sets has been a hot topic in academia and industry for the past decade. This interest is explained by the explosion of data availability and low-cost computing, which in turn have made big data and machine learning (ML) ubiquitous in fields as different as engineering and social sciences because of their potential to support better and faster decision-making [1], [2]. However, the results from statistical and ML models are only as good as the data they are trained on and their ease of interpretation.

Although data quality is essential for obtaining robust and reliable results, real-world data is often incomplete, noisy, or inconsistent [3]. Therefore, data preprocessing is crucial before performing any analytical study. In addition, the results obtained from ML models should be easy to interpret so that knowledge can be easily extracted from them and decision-making processes can be improved. This is why explainable artificial intelligence (XAI) has gained momentum in the last few years [4], [5].

Among the various ML techniques suitable for knowledge discovery, clustering methods are widely used when working with large volumes of data [1]. Clustering is a fundamental task in data mining that groups observations according to their similarities,

* Corresponding author.
E-mail addresses: uo291291@uniovi.es, malvarez.statistics@gmail.com (M. Alvarez-Garcia).

https://doi.org/10.1016/j.ins.2024.120282
Received 8 July 2023; Received in revised form 18 January 2024; Accepted 31 January 2024

Available online 5 February 2024

synthesizing the information in the data and thus making it more accessible. Nonetheless, clustering results also depend on the data quality, and their interpretability can be particularly challenging due to the unsupervised nature of the problem [1], [6].

While XAI is more commonly applied to supervised ML, critical recent developments have emerged in its application to unsupervised problems. These developments often involve converting the clustering problem into a classification task by utilizing the labels previously obtained in the clustering process [7]. Decision trees are commonly used to interpret clusters by extracting the decision rules that predict cluster labels. For example, in [8], authors propose a model of explainable k-means and k-median based on decision trees with k leaves, shedding some light on the influence of parameters like the dimension of the data or the number of outliers on the computational complexity of explainable clustering. Similarly, [6] presents an improved tree-based method with a clustering technique based on a mechanism that constructs multiple decision trees, known as eUD3.5. This approach considers both separation and compactness when evaluating a feature split.

Alternatively, a novel strategy proposed in [7] introduces an explanation mechanism based on multidimensional bounding boxes. This method allows for representing clusters with arbitrary shapes and combines local and global explanations. The authors of [9] introduce EXPLAIN-IT, a methodology that uses an innovative XAI approach to comprehend the factors contributing to specific decisions made by supervised learning models. Support vector machines (SVM) are used as classifiers for unsupervised ML, and lime [9] is employed for explainability. Six different supervised algorithms are compared in [10] to classify cluster labels, obtaining the best predictive performance for ensemble algorithms.

Most of the methods above primarily focus on the explainability task and are designed to work with particular clustering and classification algorithms. Moreover, data preprocessing is outside their scope, and challenges such as outlier detection, high-dimensionality, or handling mixed-type data are not explicitly addressed. However, data preprocessing is essential before running any cluster analysis [3].

This work aims to present a comprehensive methodological framework for conducting explainable cluster analysis, which effectively addresses the data quality and explainability challenges described above for large and high-dimensional mixed-type data. This framework serves as a guide for researchers and practitioners, offering an end-to-end sequential workflow consisting of four main steps:

- Data preprocessing to ensure the quality of the data.
- Dimensionality reduction to help interpretability by reducing the number of variables to a manageable level for human comprehension [11] and to avoid the curse of dimensionality.
- Clustering to group observations into similar groups and extract relevant patterns.
- Classification of the previously computed clusters to facilitate cluster explainability and classify future observations.

For decades, these four topics have been extensively studied; however, to the best of our knowledge, there is currently a lack of a comprehensive and guided framework covering all of them, allowing for combined quantitative and qualitative data and with a focus on explainability. Instead, specialized methods addressing specific tasks within the proposed framework can be found.

1.1. Novelty and contributions

The main contributions of this work are summarized as follows:

- We introduce a novel framework for explainable cluster analysis, addressing existing challenges related to data quality, explainability, large datasets, and mixed-type data using an end-to-end methodology.
- We implement this methodology in an open-source Python package, Clust-learn, for broad accessibility to researchers and practitioners.
- We validate the methodology through a holistic application to a real-world large-scale assessment database in the field of education.

Additionally, each of the four components of the methodology has specific contributions:

- We design a novel data imputation method that effectively handles mixed-type data, leverages strong variable relationships, and addresses high-dimensionality.
- We enhance existing dimensionality reduction frameworks for mixed-type data by applying regularization, which facilitates interpretability.
- We build a flexible process for comparing multiple clustering algorithms, acknowledging the absence of a universal algorithm and enabling optimal algorithm selection.
- We create a pipeline for classification that includes feature selection, hyperparameter optimization, and local and global feature importances using XAI methods and cross-validation. The pipeline also supports multiple tree-based classification algorithms.

2. Methodology overview

This section provides an overview of the methodology, which consists of four components: data preprocessing, dimensionality reduction, clustering, and classification. These components are designed to be used sequentially to ensure robust and explainable

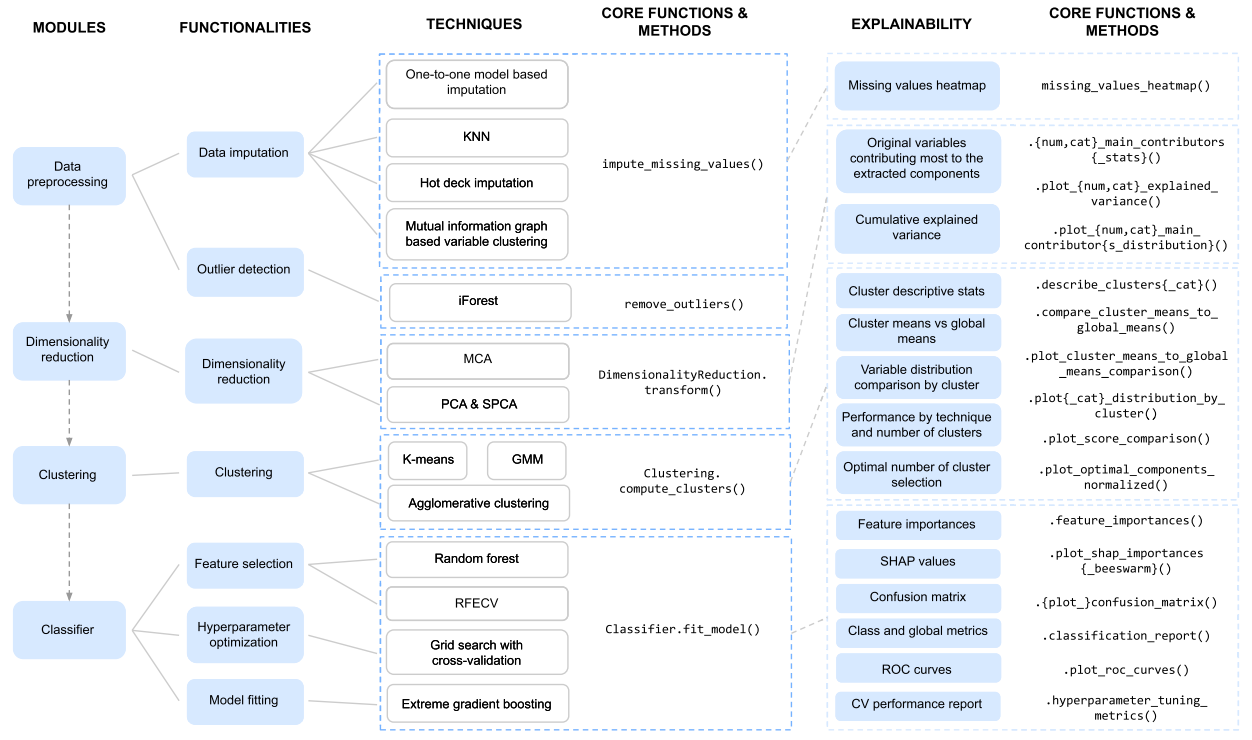

Fig. 1. Clust-learn package layout.

results. However, they can also be employed independently to suit different use cases. Sections 3 through 6 describe the background and methods used in each of the four components, together with their implementation and customization options in the Python package Clust-learn.

The data preprocessing component covers missing data imputation and outlier detection. To address missing data, we propose a method that first identifies strongly related variable pairs for one-to-one model-based imputation. Subsequently, we employ graph theory to identify clusters of variables with strong relationships, allowing us to perform imputation within each cluster to mitigate the challenges caused by high-dimensionality. Missing values are imputed using linear regression and hot deck imputation. As for outlier detection, we use the Isolation Forest algorithm [12], which is well-suited for identifying multivariate outliers.

A different approach is employed for numerical and categorical variables for dimensionality reduction, followed by their subsequent integration. Sparse principal component analysis (SPCA) is applied to numerical variables to leverage sparse connections with the original variables, thus facilitating interpretability. Categorical variables, on the other hand, are treated using multiple correspondence analysis (MCA).

The clustering component enables the comparison of multiple clustering algorithms by using different validity metrics and the elbow method to identify the optimal number of clusters.

In the classification component, a pipeline is employed, starting with a random train-test split, followed by feature selection and hyperparameter optimization using cross-validation. Lastly, a classification model is trained and evaluated. This pipeline supports a wide range of algorithms and performance metrics.

The proposed methodology is implemented in the Python package Clust-learn, which encapsulates complex methodological procedures using methods and functions that are easy to use. It is organized into four modules, one for each component of the methodology: `data_preprocessing`, `dimensionality_reduction`, `clustering`, and `classifier`. Clust-learn is implemented in Python because it has been the most popular programming language in 2022 and 2023, with a steady upward trend since 2018, according to the TIOBE index [13].

Fig. 1 shows the package layout with the functionalities covered by each module, the techniques used, the explainability strategies available, and the main functions and class methods implementing these techniques and explainability strategies. The package also offers a rich set of tabular and graphical descriptive statistics to help interpretability in each step of the methodology.

The key contributions of Clust-learn lie in its emphasis on unsupervised learning, a strong focus on data quality and interpretability, and a guided approach for users. Compared to automated machine learning (AutoML) [14] packages such as tpot [15] and auto-sklearn [16], which facilitate the comparison of a plethora of supervised models through a systematic approach to model selection, Clust-learn distinguishes itself by incorporating data preprocessing and unsupervised learning, and by being structured to prioritize interpretability. Python packages focused on data preprocessing, such as pandas-profiling [17] and missingno [18], tend to treat numerical and categorical variables separately and do not fully leverage their relationships. PyCaret [19], a Python package with a broader scope compared to the previous ones, serves as an open-source, low-code Python library designed to make ML more

accessible to a broader audience. PyCaret offers rich functionality for data preprocessing, both supervised and unsupervised learning, and time series analysis. Clust-learn, on the other hand, distinguishes itself for prioritizing the interpretability of results and providing a guided framework for explainable cluster analysis. It offers a more targeted set of techniques compared to PyCaret. Finally, SHAP (SHapely Additive exPlanations) is the state-of-the-art method for explainability of ML models [20] and it is incorporated in Clust-learn through the Python package shap [21].

3. Data preprocessing

The proposed methodology follows the framework introduced in [3] for data preprocessing, which identifies five main tasks to be performed before running any cluster analysis: handling missing values, outlier detection, dimensionality reduction, data scaling, and mixed-type data handling. In our methodology, the data preprocessing module in this section covers tasks 1 and 2. Task 3 is handled in a separate module in Section 4 due to its relevance. Tasks 4 and 5 are considered at different stages throughout the methodology.

3.1. Imputation of missing values

The presence of missing values is common in scientific and research studies, and their imputation is essential when working with real-world data as missing information can significantly influence the results obtained [22], [23]. There are multiple imputation methods in the literature, and their suitability depends on the types of variables and the type and amount of missing values [22]. We propose a method that combines data deletion, model-based one-to-one imputation, k-nearest neighbors (KNN), and hot deck imputation (see Fig. 2).

Hot deck imputation is one of the most widely used methods for data imputation [22]. For each observation with some missing value (recipient), it identifies a set of observations with a value informed for the corresponding variable (donors) and imputes a value based on those of the donors. This method has numerous versions depending on the donor selection technique and the criterion for selecting the value to be imputed [22]. In our methodology, we use hot deck imputation in two of the steps with two different donor selection techniques, as described below.

In the first step of the data imputation method, we remove all variables with a percentage of missing values above a predefined threshold. Next, in the second step, we use model-based imputation for strongly related pairs of variables. The relationship between pairs of variables is measured using Pearson's correlation coefficient for numerical variables, partial *eta* squared, η_p^2, for mixed-type variables, and mutual information for categorical pairs. A pair of variables is considered to be strongly related if the corresponding metric score is above a predefined threshold. If a variable has a strong relationship with more than one variable, the pair that allows the largest imputations is selected.

Once these pairs of dependent (Y, variable with missing values) and independent variable (X, variable used as predictor) are calculated, we impute values differentiating three cases:

Case 1. If variables X and Y are numerical, linear regression is used. The regressor is fitted using all observations with known values for both variables.

Case 2. If both variables are categorical, we use hot deck imputation. For every recipient (observation with a missing value for Y and known $X = x_i$), the value to be imputed is selected at random from the discrete empirical distribution given by

$$P(Y = y_j | X = x_i) = \hat{f}_{x_i}(y_j), \tag{1}$$

where $\hat{f}_{x_i}(y_j)$ is the relative frequency of y_j when $X = x_i$.

Case 3. For mixed-type variables, the numerical one is discretized into quantiles and both variables are treated as categorical (see Case 2).

The third step consists of removing all records with a percentage of variables with missing values above a predefined threshold. This step aims to ensure that in step four, the recipient and donors are sufficiently similar when using KNN and that a large number of missing values does not dominate the relationship.

Finally, the remaining missing values are imputed in the fourth step using the hot deck method with KNN to search for donors. In particular, we use scikit-learn's KNNImputer algorithm [24], [25] that allows for custom distance metrics and value selection criteria. For distance computation, we use the Euclidean distance with missing values proposed in [26] given by

$$d(x, y) = \sqrt{w \sum_{\substack{i \in I \\ x_i, y_i \neq nan}} (x_i - y_i)^2}, \tag{2}$$

where I is the set of variable indices and $w = \frac{|I|}{|\{i \in I : x_i, y_i \neq nan\}|}$.

The value k in KNN is configurable, but using a value greater than 1 is generally recommended, as it usually ensures a better quality of the imputation [22]. In addition, all variables are normalized to the 0-1 range to avoid dominance driven by scale differences. Among the nearest neighbors, a value is selected at random, giving the same probability to all neighbors.

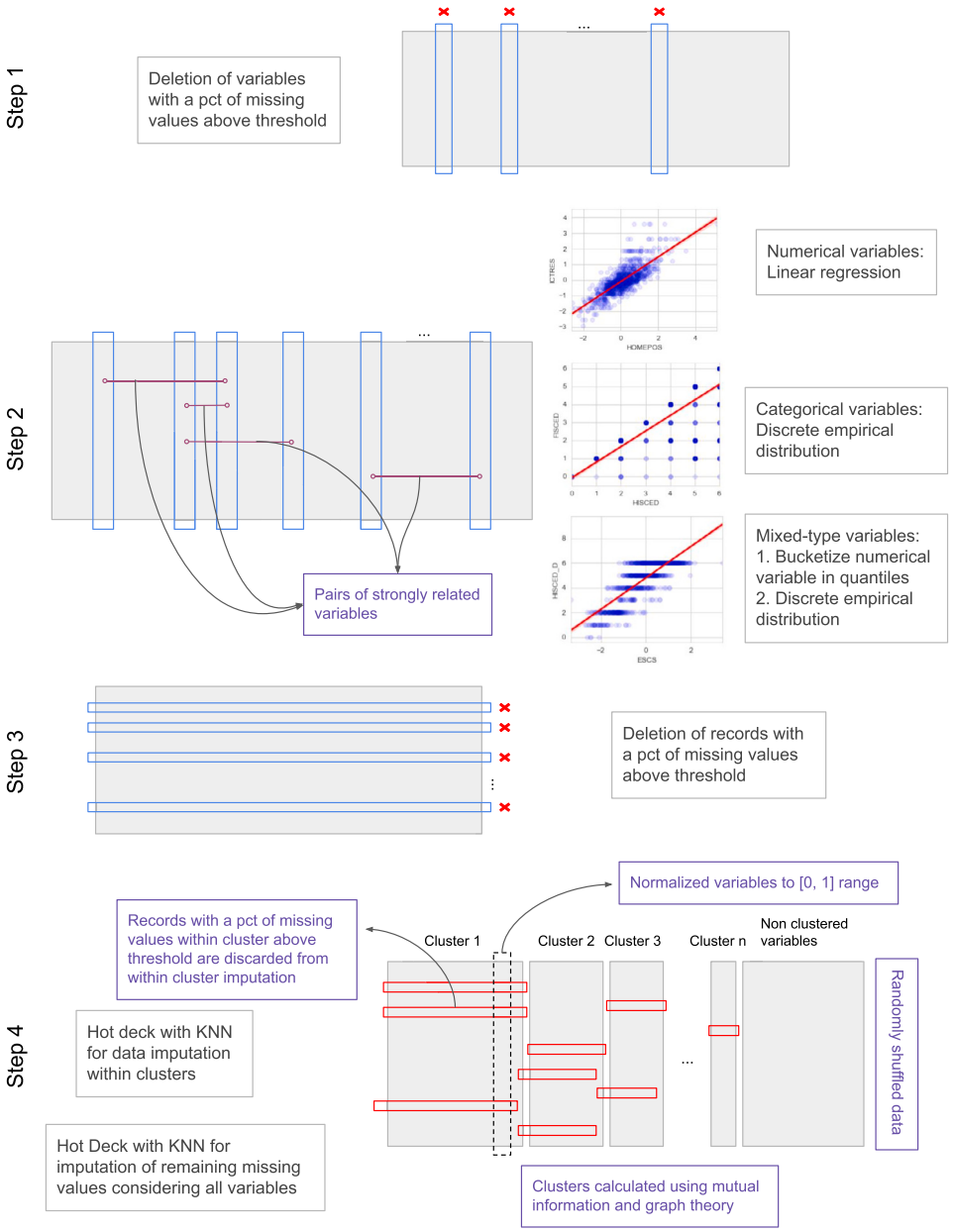

Fig. 2. Data imputation methodology.

The curse of dimensionality affects the Euclidean distance since the relative distance between the farthest and the nearest point converges to zero as the number of dimensions increases [27]. This problem also extends to the Euclidean distance with missing values, given its reliance on the standard Euclidean distance (see Eq. (2)). To overcome this drawback, different alternatives can be found in the literature, such as those proposed in [22], [23]. Based on these studies, our methodology uses mutual information scores between pairs of variables to identify clusters of variables with similar behavior and impute values within each cluster, thus reducing dimensionality. The clusters of variables are computed using graph theory. In particular, variables are modeled as the graph nodes (V) and the pairs of variables with a mutual information score above a predefined threshold as edges (E). The connected components of the undirected graph $G = (V, E)$ constitute the clusters. Only clusters with a minimum number of elements are considered.

Observations with a percentage of missing values within a cluster above a predefined threshold are discarded from this imputation to avoid recipient-donor similarities based on very few variables. For these cases and for other variables that do not belong to any cluster, missing values are imputed considering all the variables of the study together.

It is important to note that the KNNImputer algorithm imputes values in the order of the given data. Before the fourth step, a random shuffle is performed to avoid possible biases caused by the original order of the data.

3.2. Outlier detection

For outlier detection, we use the scikit-learn's implementation of the Isolation Forest algorithm (iForest) [12], a multivariate anomaly detection algorithm consisting of an ensemble of iTrees, binary trees where both variables and splits are randomly selected for data partitioning. This algorithm is well suited for high-dimensional and mixed-type data [12].

Due to their susceptibility to isolation, outliers are isolated closer to the tree's root node. This is the key feature of the algorithm for identifying outliers. The algorithm implementation has two other sources of randomness: the selection of samples with optional bootstrapping and the selection of features for each iTree.

3.3. Data preprocessing with Clust-learn

The `data_preprocessing` module is organized into a set of functions. Its key functionalities are data imputation and outlier detection. The former is performed by means of the function

```
impute_missing_values(df, num_vars, cat_vars, num_pair_kws = None,
   mixed_pair_kws = None, cat_pair_kws = None, graph_thres = 0.05,
   k = 8, max_missing_thres = 0.33)
```

that implements all steps in the proposed method for a given data set (`df`) with numerical (`num_vars`) and categorical variables (`cat_vars`). Some customization parameters are allowed, such as the arguments to pass to compute imputation pairs for one-to-one model-based imputation (`num_pair_kws`, `mixed_pair_kws`, and `cat_pair_kws`), the threshold for determining whether two variables are connected when computing variable clusters (`graph_thres`), the number of neighbors for KNN (`k`), and the maximum proportion of missing values per observation allowed at step 3 of the imputation method (`max_missing_thres`). In addition to these parameters, separate functions are provided for each step of the data imputation methodology should the user want to tailor it further.

Outliers are identified and removed using the function `remove_outliers(df, variables, iforest_kws = None)` that allows iForest customization through `iforest_kws` for a selection of variables (`variables`).

In addition to the prior functions, the module provides the function `missing_values_heatmap()` for visualizing the presence of missing values on a heat map (Fig. 3a), and the function

```
plot_imputation_distribution_assessment(df_prior, df_posterior,
   imputed_vars, sample_frac = 1.0, prior_kws = None,
   posterior_kws = None, output_path = None, savefig_kws = None)
```

for visualizing a comparison between the kernel distribution of a set of variables (`imputed_vars`) before and after imputation through kernel density estimation plots (Fig. 3b).

4. Dimensionality reduction

For dimensionality reduction, a procedure based on factor analysis of mixed data (FAMD) [28] and multiple factor analysis (MFA) [28] is applied. In particular, numerical and categorical variables are treated separately, as proposed in [28], and the results are later combined in a single table.

For numerical variables, we use SPCA, a principal component analysis (PCA) variant where each extracted component is a linear combination of a reduced number of original variables, facilitating explainability. We use scikit-learn's implementation, which is based on the one proposed in [29] wherein the problem is formulated as a PCA with L1 regularization on the components

$$\begin{aligned} &\min_{\substack{U\in\mathbb{R}^{m\times r}\\ V\in\mathbb{R}^{n\times r}}} \frac{1}{2}\|X - UV^{\top}\|_F^2 + \lambda\|V\|_{1,1} \\ &\text{subject to } \|U^k\|_2 \le 1, \forall k = 1,\dots,r, \end{aligned} \tag{3}$$

where $X \in \mathbb{R}^{m\times n}$ is the matrix with m observations and n original variables, $U \in \mathbb{R}^{m\times r}$ is the transformed matrix with r derived variables, $V \in \mathbb{R}^{n\times r}$ is the coefficient matrix, and λ is the regularization parameter. $\|\cdot\|_F$ and $\|\cdot\|_{1,1}$ are the entry-wise matrix norms for $p = 2$, $q = 1$ (the so-called Frobenius norm) and $p = 1$, $q = 1$, respectively, where

$$\|A\|_{p,q} = \left(\sum_{j=1}^{n}\left(\sum_{i=1}^{m}\left|a_{ij}\right|^p\right)^{\frac{q}{p}}\right)^{\frac{1}{q}}. \tag{4}$$

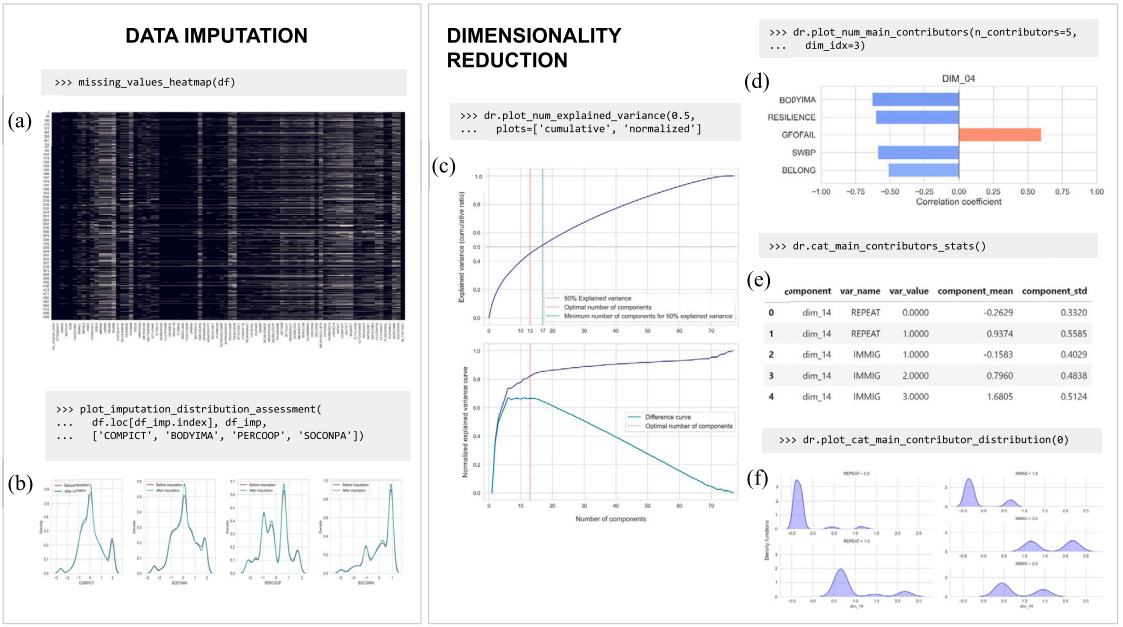

Fig. 3. Selection of tabular and graphical descriptive statistics for data preprocessing and dimensionality reduction modules available in Clust-learn.

The original variables are standardized (centered and divided by their standard deviation) before applying SPCA to avoid any dominance driven by scale differences.

For categorical variables, we use MCA, a variant of correspondence analysis (CA) for multiple categorical variables. In particular, the method implemented in [30] is used as described below.

Let n be the number of categorical variables and p_j the number of categories of variable j, with $j \in \{1, \dots, n\}$ and $\sum_{j=1}^{n} p_j = p$. Let m be the number of observations and X the $m \times p$ correspondence matrix of relative frequencies. Let us denote v_r and v_c as the vectors of row and column totals, respectively, and $D_{v_r} = \text{diag}(v_r)$ and $D_{v_c} = \text{diag}(v_c)$. The following singular value decomposition (SVD) factorization provides the factor scores

$$D_{v_r}^{\frac{-1}{2}} \left(X - v_r v_c^{\top}\right) D_{v_c}^{\frac{-1}{2}} = U \Sigma V^{\top}, \tag{5}$$

where Σ is the diagonal matrix of the singular values and $\Lambda = \Sigma^2$ the matrix of eigenvalues.

The contribution of column l to the extracted factor k is obtained as

$$w_{lk} = \frac{f_{lk}^2}{\lambda_k}; l, k = 1, \dots, p, \tag{6}$$

where $F = (f_{lk})_{lk} = D_{v_c}^{\frac{-1}{2}} V \Sigma$.

The one-hot encoding applied to obtain the correspondence matrix distorts the original dimension of the data, which causes total inertia to be artificially inflated. Therefore, the inertia explained by the first extracted factors is underestimated. For eigenvalue correction, the Benzecri [31] formula is applied, wherein the corrected eigenvalues are defined as

$$\bar{\lambda}_k = \begin{cases} \left[\left(\frac{n}{n-1}\right)\left(\lambda_k - \frac{1}{n}\right)\right]^2 & \text{if } \lambda_k > \frac{1}{n} \\ 0 & \text{if } \lambda_k \leq \frac{1}{n} \end{cases}. \tag{7}$$

Total inertia is then replaced for the alternative definition proposed by Greenacre [32]

$$\bar{I} = \frac{n}{n-1}\left(\sum_{k=1}^{p} \lambda_k^2 - \frac{p-n}{n^2}\right), \tag{8}$$

and, therefore, the percentage of inertia explained by the factor k is calculated as

$$\frac{\bar{\lambda}_k}{\bar{I}}, \text{ for all } k = 1, \dots, p. \tag{9}$$

The optimal number of components to be extracted from numerical and categorical variables is calculated using the elbow method on the cumulative explained variance and inertia. The elbow method has proven to be powerful because it is easy to understand

and assess through visualization [33]. We use the method proposed and implemented in [34]. The basis of this algorithm lies in the concept of curvature of a continuous function. Since the cumulative explained variance and inertia are discrete functions, a smoothing spline is first applied to obtain a continuous function that preserves the shape of the original data. Secondly, the data are normalized, and a new curve $(x, y - x)$ is calculated. The local maxima of the latter curve are the candidate elbow points. Finally, a threshold based on the difference of consecutive x-values is used for identifying the elbow points. For a thorough description of the algorithm, see [34].

To explain the extracted components, a threshold is set on Pearson's correlation coefficient to determine the original numerical variables explained by each principal component. Similarly, a threshold is set on the partial *eta* square for categorical variables to determine the original variables explained by each derived variable.

4.1. Dimensionality reduction with Clust-learn

All the functionality of the `dimensionality_reduction` module is encapsulated in the `DimensionalityReduction` class so that the original data, the instances of the models used, and any other relevant information is self-maintained and always accessible. Table A1 shows the class parameters and attributes. It is instantiated as follows

```
dr = DimensionalityReduction(df, num_vars = None,
   cat_vars = None, num_algorithm ='pca', cat_algorithm ='mca',
   num_kwargs = None, cat_kwargs = None)
```

Users must enter the list of numerical (`num_vars`) and categorical variables (`cat_vars`) along with the technique applied to each subset (`num_algorithm` and `cat_algorithm`, respectively). For now, PCA and SPCA are supported for numerical variables and MCA for categorical variables.

Dimensionality reduction is performed using the class method

```
dr.transform(n_components, min_explained_variance_ratio = 0.5)
```

This method implements the proposed methodology, transforming the original data into a lower dimensional space handling numerical, categorical, and mixed-type data. The user can set either the number of components to be extracted (`n_components`), the minimum variance to be explained by them (`min_explained_variance_ratio`), or they can set both parameters to `None` if they would like to obtain the optimal number of components using the elbow method on the cumulative explained variance and inertia.

The class also provides a set of methods to explain the extracted components. The methods

```
dr.{num,cat}_main_contributors(thres, n_contributors = None,
   dim_idx = None, component_description = None,
   col_description = None, output_path = None)
```

obtain the original numerical (`num`) or categorical (`cat`) variables explained by the extracted component `dim_idx` (if set to `None`, this is computed for all extracted components). The number of explained original variables can be controlled using the parameter `thres` that sets a threshold on Pearson's correlation coefficient or partial *eta* squared, respectively, or by simply setting the number through `n_contributors`. In the latter case, the method returns the `n_contributors` original variables most strongly related to the component `dim_idx`.

The method

```
dr.cat_main_contributors_stats(thres = 0.14,
   n_contributors = None, dim_idx = None, output_path = None)
```

computes for every original categorical variable's value, the mean and standard deviation of the extracted components with a partial *eta* square above `thres` (Fig. 3e). If the user wants to obtain these statistics for a number of extracted components with the highest *eta* square, they can do so through the parameter `n_contributors`.

For visualizations, the methods

```
dr.plot_{num,cat}_explained_variance(thres, plots ='all',
   output_path = None, savefig_kws = None)
```

plot the explained variance curve for numerical (Fig. 3c) and categorical variables, respectively. The available plots under the parameter `plots` are `cumulative`, `ratio`, and `normalized`. Any subset from these may be selected.

The class method

```
dr.plot_num_main_contributors(thres = 0.5, n_contributors = 5,
   dim_idx = None, output_path = None, savefig_kws = None)
```

generates a bar plot showing the Pearson correlation coefficient of the extracted `dim_idx` component with the most correlated original numerical variables (Fig. 3d). The number of original variables to be displayed can be controlled using the parameter `thres` to set a minimum on the absolute value of the correlation coefficient, or by simply setting a number using `n_contributors`.

For categorical variables, the class method

```
dr.plot_cat_main_contributor_distribution(thres = 0.14,
   n_contributors = None, dim_idx = None, output_path = None,
   savefig_kws = None)
```

plots for every value of each original categorical variable, a kernel density estimation plot of the extracted components that have a partial *eta* square above a predefined threshold (`thres`) (see Fig. 3f). If the user wants to obtain these plots for a number of components with the highest *eta* square, they can do so through the parameter `n_contributors`.

5. Clustering

Cluster analysis substantially depends on the particular field of study, and it has been consistently proved that no universal clustering approach exists [35]. In recent years, numerous novel clustering algorithms have been proposed and implemented, emphasizing the need for enhanced, flexible, and efficient methods to accommodate different data types and fields of study [1]. To address this requirement, our methodology is designed to accept any clustering algorithm that adheres to the scikit-learn standards. The proposed approach enables us to compare multiple algorithms based on a selection of cluster validity metrics, thus allowing us to identify the optimal technique. These metrics are presented later in this same section.

As the default choice, we employ k-means [36] with k-means++ [37] for centroid initialization, given its wide adoption [1], thereby ensuring that our methodology remains accessible to a broad community, particularly for beginners.

For data other than the components extracted from dimensionality reduction, variable normalization is performed to ensure that no priority is given to any of them due to differences in scale. This option can be attractive when the clustering method selected already deals with high-dimensionality and the previous step of the methodology can be skipped.

Finding the relevant number of clusters is a fundamental problem in clustering [1]. The elbow method [34] applied to one of the four unsupervised validity metrics available in scikit-learn is used to find this number. Note these are the same metrics we use for best clustering algorithm selection.

Hereafter, we denote n as the number of observations, $X = \{x_i\}_{i=1,\dots,n}$ as the observations, q as the number of clusters, c_k as the centroid of cluster k, n_k as the number of observations of cluster k, and X_k as the observations in cluster k, with k in $\{1, \dots, q\}$.

Within cluster sum of squares (WSS). Also known as the sum of squared distances, WSS (10) measures the sum of the square distances of every observation to its corresponding cluster.

$$WSS = \sum_{k=1}^{q} \sum_{x_i \in X_k} (x_i - c_k)^2 \tag{10}$$

Davies-Bouldin index. This index compares the mean intra-cluster distance of any pair of clusters to the distance between their centroids [38]. It is calculated using Equation (11).

$$DB = \frac{1}{q} \sum_{k=1}^{q} \max_{k \neq l} \left\{ \frac{\delta_k + \delta_l}{d_{kl}} \right\} \tag{11}$$

where

- $\delta_k = \frac{1}{n_k} \sum_{x_i \in X_k} d(x_i, c_k)$ is the mean intra-cluster distance, and
- $d_{kl} = d(c_k, c_l)$ is the distance between the cluster centroids.

The lower the Davies-Bouldin index, the better the clustering, being zero the minimum possible value.

Silhouette index. The silhouette index [39] in Equation (12) compares, for every observation, the average distance to the other elements of the same cluster with the minimum mean distance to all observations in each cluster.

$$Silhouette = \frac{1}{n} \sum_{i=1}^{n} s_i \tag{12}$$

where

- $s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}$,
- $a_i = \frac{1}{n_{k_0} - 1} \sum_{j \neq i; x_j \in X_{k_0}} d(x_i, x_j)$ with k_0 such that $x_i \in X_{k_0}$, is the mean distance between of one observation to all other observations in the same cluster, and

- $b_i = \min_{k_1 \neq k_0} \left\{ \frac{1}{n_{k_1}} \sum_{x_j \in X_{k_1}} d(x_i, x_j) \right\}$ is the smallest mean distance to all elements in each of the other clusters.

The silhouette index can take values between -1 and 1, where 1 is the best possible value.

Calinski and Harabasz index. This index (13) compares the cluster centroid closeness to the mean intra-cluster distance [40].

$$CH = \frac{tr(B_q)}{tr(W_q)} \cdot \frac{n-q}{q-1} \tag{13}$$

where

- $tr(\cdot)$ is the trace of a matrix function,
- $B_q = \sum_{k=1}^{q} n_k (c_k - \bar{x})(c_k - \bar{x})^\top$ with $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ is the inter-cluster dispersion matrix, and
- $W_q = \sum_{k=1}^{q} \sum_{x_i \in X_k} (x_i - c_k)(x_i - c_k)^\top$ is the intra-cluster matrix.

Under Calinski and Harabasz index, the best clustering is that which maximizes its value.

The last and most important step in cluster analysis is understanding the results to extract knowledge from the data and label the clusters for further practical use [6]. This goes beyond assessing the validity of the obtained clusters and requires putting the results into the context of the study. Therefore, a good presentation of the clustering output is essential for interpretability [1]. We propose using tabular and graphical descriptive statistics using internal and external variables, i.e., variables that were and were not used to compute the clusters. The complete list of available tables and visualizations and their corresponding description, can be found in Section 5.1.

5.1. Clustering with Clust-learn

The `Clustering` class encapsulates this module's functionality and stores the data, the instances of the algorithms used, and other relevant information so it is always accessible. See Table A2 for detailed information on class parameters and attributes. The class is instantiated as follows

```
cl = Clustering(df, algorithms = kmeans, normalize = False)
```

The methodology presented for clustering is implemented via the class method

```
cl.compute_clusters(n_clusters = None, metric ='inertia',
   max_clusters = 10, prefix = None)
```

which returns an array of labels, one for each observation. If more than one algorithm is passed in the class constructor, they are compared using the metric passed (`metric`) and the best performing one is selected. The number of clusters to be calculated can be selected manually or optimized using the elbow method with the parameter `n_clusters`.

In addition to the computation of clusters, the `Clustering` class provides a set of methods to help interpret clusters through summary tables and visualizations. The class method

```
cl.describe_clusters(df_ext = None, variables = None,
   cluster_filter = None, statistics = ['mean','median','std'],
   output_path = None)
```

describes clusters in terms of internal variables (those used for clustering) or external continuous variables (`df_ext`). The `variables` and `cluster_filter` parameters allow the user to select a subset of variables or clusters from which to obtain the descriptive statistics passed to the parameter `statistics`.

For external categorical variables, frequency distribution tables can be computed through the class method

```
cl.describe_clusters_cat(cat_array, cat_name, order = None,
   normalize = False, use_weights=False, output_path = None)
```

To help identify which variables most strongly characterize each cluster, the class method

```
cl.compare_cluster_means_to_global_means(df_original = None,
   output_path = None)
```

computes the relative difference between intra-cluster means and global means. The parameter `df_original` allows the user to compute these differences using the original variables in case clusters are computed on a lower dimensional space. For a heat map visualization of this analysis (see Fig. 4c), use the class method

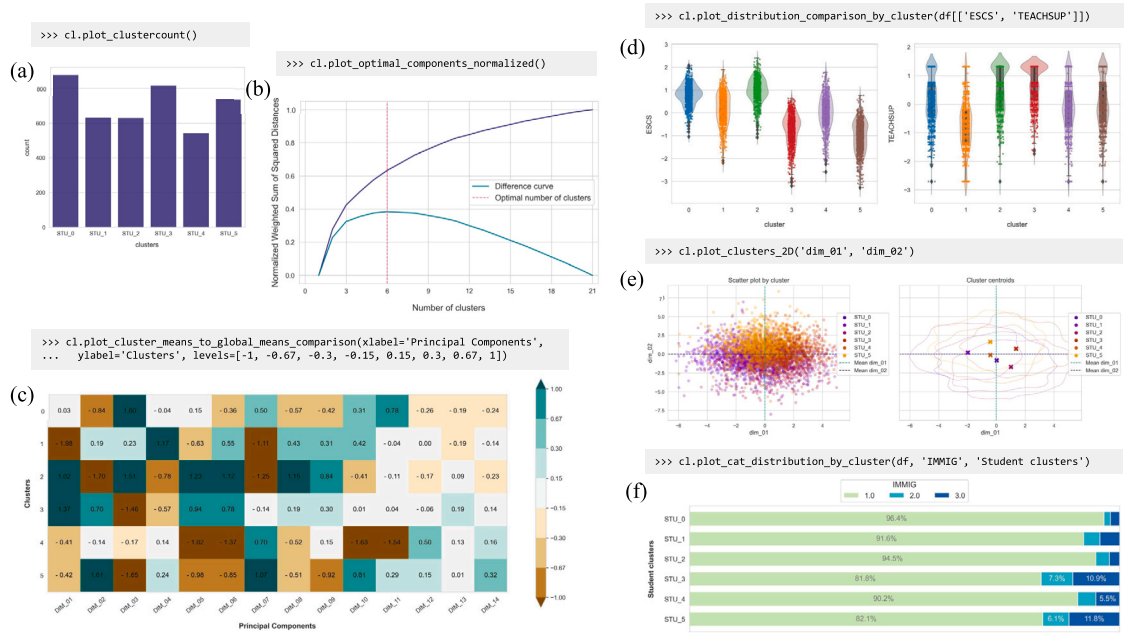

Fig. 4. Selection of tabular and graphical descriptive statistics for the clustering module available in Clust-learn.

```
cl.plot_cluster_means_to_global_means_comparison(
   use_weights = False, df_original = None, xlabel = None,
   ylabel = None, levels = [-0.5, -0.3, -0.1, 0.1, 0.3, 0.5],
   use_weights = False, df_original = None, xlabel = None,
   output_path = None, savefig_kws = None)
```

Other available visualizations include: count plots per cluster through `plot_clustercount()` (Fig. 4a); a comparison of the distribution of internal or external continuous variables by cluster through a composite visualization of violin plots, box plots and point density plots with the method `plot_distribution_by_cluster()` (Fig. 4d); a comparison of the distribution of categorical variables with normalized stacked bar plots using `plot_cat_distribution_by_cluster()` (Fig. 4f); and 2-dimensional scatter plots and kernel density estimation plots for detailed cluster analyses through the visualization of pairs of internal variables by cluster using `plot_clusters_2D()` (Fig. 4e).

The user can also run cluster validation analyses based on the selected performance metric. In particular, they can compare different clustering algorithms and evaluate the selection of the optimal number of clusters with the class methods `plot_score_comparison()` and `plot_optimal_components_normalized()` (Fig. 4b), respectively.

Finally, analysis of variance (ANOVA) and *chi* squared tests are available through the class methods `anova_tests()` and `chi2_test()` to test the independence of clusters and numerical and categorical variables, respectively.

6. Classifier

The classification module has two objectives in the proposed methodology: to support a better understanding of the clusters calculated in the previous module and to allow future classification of new observations that were not part of the original study. In addition, this module can be used independently to classify other labeled data.

The module supports mixed-type data and binary and multiclass classification. For explainability, we rely on feature selection, essential when working with high-dimensional data [41], and Tree SHAP values [42].

SHAP [21] is an additive feature attribution method that calculates the unique combination of predictor contributions for every model output. These contributions are the so-called SHAP values. They are the approximate Shapley values [43], a concept from game theory, computed using weighted linear regression [21]. SHAP values represent the importance of each feature in each prediction. In addition to these local importances, a model's global feature importances can be obtained by averaging the absolute value of the local SHAP values. SHAP is a model-agnostic method with a novel version, Tree SHAP [42], designed to work efficiently with tree-based models and provide exact solutions. Since all the models used throughout the methodology are tree-based, we selected this last version.

We follow four main steps for building a classification model:

Step 1. Data is randomly **split into train and test** subsets. Steps 2 and 3, and model training in step 4 are run using only the train split.

Step 2. Feature selection is performed in three steps. The first two steps remove highly related features that add redundant information. The relationship between variables is measured using Pearson's correlation coefficient, partial *eta* square, and mutual information, as it is done consistently throughout the methodology, depending on the types of variables. First, if some features are to be kept because of their relevance to the study, all other features with a strong relationship to the former are removed. Secondly, a random forest classifier [44] is iteratively trained. At each iteration, the feature with the highest SHAP value not previously visited is selected and all other highly related features are removed. The last step runs recursive feature elimination with cross-validation (RFECV) [45] as implemented by scikit-learn.

Step 3. Hyperparameter optimization is performed through exhaustive grid search with cross-validation for the selected hyperparamters.

Step 4. Model training, evaluation, and interpretation. By default, we use xgboost [46] with the area under the curve (AUC) as performance metric for model fitting. For a complete model evaluation, we use the confusion matrix, accuracy, precision, recall, f-1 score metric, and the receiver operating characteristic (ROC) curve in addition to the AUC. We rely on SHAP values for results interpretability for local and global importances.

By default, we use scikit-learn's implementation of random forests for feature selection because it is one of the most successful algorithms in machine learning and it has shown impressive performance with default hyperparameters [47], and xgboost for the final classification model because it is among the top performers in Kaggle competitions [47]. Hyperparameter tuning is applied on xgboost because its performance can improve significantly with the appropriate ones [47]. The methodology is adaptable and allows any tree-based classification algorithm.

6.1. Classification with Clust-learn

The functionality of the `classifier` module is encapsulated in the `Classifier` class, which is also responsible for storing the original data, the instances of the models used, and any other relevant information. See Table A3 for detailed information on class parameters and attributes. The class is instantiated as follows

```
classifier = Classifier(df, predictor_cols, target,
   num_cols = None, cat_cols = None)
```

The methodology for classification presented above is implemented via the method

```
classifier.train_model(model = None, feature_selection = True,
   features_to_keep = [], feature_selection_model = None,
   hyperparameter_tuning = False, param_grid = None,
   train_size = 0.8)
```

By default, xgboost is used for the classifier (`model`) and random forests for feature selection (`feature_selection_model`); however, any other tree-based classification algorithm that implements the `fit`, `predict`, and `set_params` methods can be used. Feature selection is optional and can be disabled through the `feature_selection` argument. To force one feature to get selected, it must be set via `features_to_keep`. The default train-test split size is 80-20, but this can be changed with `train_size`. To enable hyperparameter optimization with cross-validation, use the boolean argument `hyperparameter_tuning`, and `param_grid` to set the parameter grid.

The `Classifier` class also provides tabular and graphical methods for explainability and performance assessment. For the former, the method `feature_importances()` computes feature importance as the combined average of the absolute values of the SHAP values for all classes. A bar plot visualization of these importances can be generated with the method `plot_shap_importances()` (Fig. 5a).

To visualize feature importances for a specific class and understand how different predictor values affect the classification of observations to that particular class (Fig. 5b), use

```
classifier.plot_shap_importances_beeswarm(class_id, n_top = 10,
   output_path = None, savefig_kws = None)
```

The goodness-of-fit of the classification model can be evaluated through the multi-class confusion matrix and the scikit-learn classification report. For the former, use the method

```
classifier.confusion_matrix(test = True, sum_stats = True,
   output_path = None)
```

Use the argument `test` to compute the confusion matrix on the test or training set, and `sum_stats` to add precision, recall, and global accuracy to the matrix. For a visualization of the confusion matrix, use the method `plot_confusion_matrix()` (Fig. 5d).

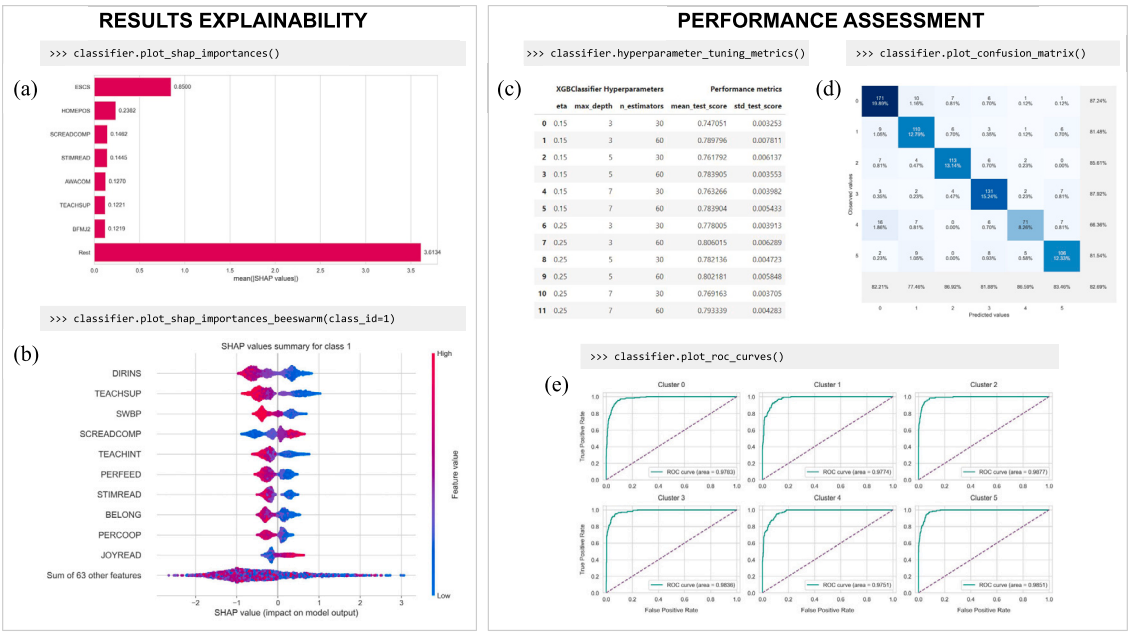

Fig. 5. Selection of tabular and graphical descriptive statistics for the classification module available in Clust-learn.

A classification report containing intraclass precision, recall, and f1-score metrics, along with the global accuracy, macro average, and weighted average of the three intraclass metrics, can be generated with the method `classification_report()`. For visualizing the receiver operating characteristic (ROC) curves by class, use the method `plot_roc_curves()` (Fig. 5e).

Finally, hyperparameter optimization can be assessed with a report containing the average and standard deviation performance of the cross-validation runs for every hyperparameter combination. This report is generated with the class method `hyperparameter_tuning_metrics()` (Fig. 5c).

7. Case study

The proposed methodology has undergone a detailed empirical study using a real-world database, enabling the validation of the methodological framework. In particular, we performed an end-to-end analysis of student data collected for the OECD's Programme for International Student Assessment (PISA). Specifically, we used a sample of 5,000 students randomly selected from the 35,943 Spanish students who took the PISA 2018 survey [48]. A total of 80 variables were selected from the different survey questionnaires (see Table B1). We selected a combination of numerical and categorical variables to illustrate how the methodology handles mixed-type data.

We started by performing data preprocessing. The data set contained a 12.01% rate of missing values, distributed unevenly across variables, ranging from as low as 0% to as high as 38.44%. We ran missing value imputation using the default options, obtaining ten pairs of variables for model-based one-to-one imputation, three clusters of variables for lower-dimension hot deck imputation (see Fig. 6), and the remainder of missing values were imputed using hot deck with all variables. After this process, we retained 4,556 observations, removing 444 where more than 1/3 of their variables' values were missing. Additionally, as part of the data preprocessing, we removed outliers using the default options, retaining 4,241 observations for the remainder of our analysis.

Once the data was preprocessed, we proceeded to apply dimensionality reduction. We used SPCA for numerical variables and MCA for categorical variables (default option) and selected the optimal number of components according to the elbow method. We obtained 13 extracted components from numerical variables that explained 42.6% of their total variance and one from categorical variables that explained 95.7% of their adjusted total inertia. As a result, we effectively reduced the original pool of 80 variables to 14 derived variables (see Table 1). Their names were established by considering the original numerical variables that presented a correlation larger than 0.5 in absolute value with the derived ones (see Table C1) and the original categorical variables with a partial *eta* squared greater than 0.14 (see Table C2).

Using the set of 14 newly derived variables, we conducted cluster analysis. We compared the performance of k-means, agglomerative clustering, and Gaussian mixture models using WSS as validity metric. We also determined the optimal number of clusters within the range of 1 to 21 using this same metric, obtaining that the optimal combination was k-means for $k = 6$ (see Fig. 7) with clusters of size 878 (20.70%), 633 (14.93%), 631 (14.88%), 817 (19.26%), 543 (12.80%), and 739 (17.43%).

We provide an initial description of the clusters by examining the average values of each derived variable used in the clustering process, comparing the internal cluster mean to the global mean. Since the derived variables were standardized with mean 0 and varying variances due to the dimensionality reduction process, we focus on comparing these means without further normalization

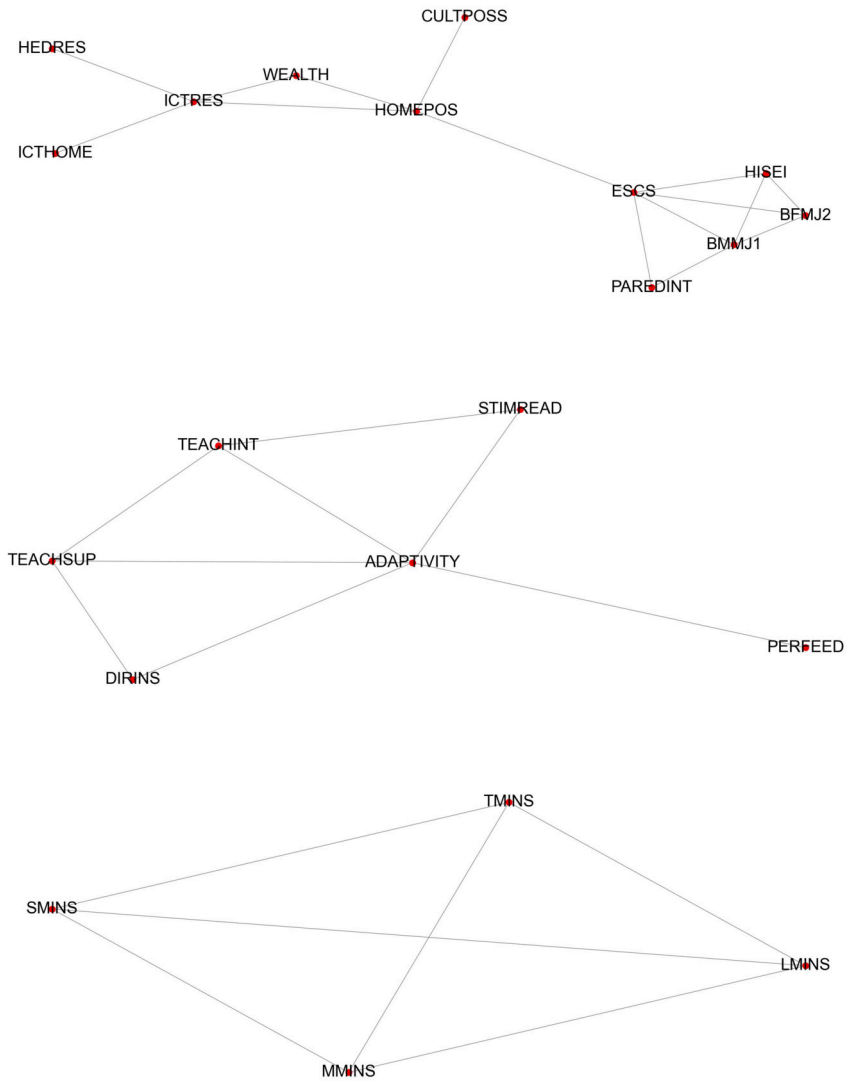

Fig. 6. Clusters of variables obtained for imputing missing values. These clusters correspond to the connected components of an undirected graph, where variables serve as nodes, and edges represent pairs of variables with a mutual information score above some threshold.

Table 1
Variables derived from the dimensionality reduction process.

	Name
1	Students' perception of teacher engagement
2	Student home resource advantage
3	Economic, social and cultural status
4	Psychological well-being
5	Work mastery and parents' emotional support
6	Global cultural attitudes
7	Reading confidence and competence
8	ICT empowerment
9	Financial confidence and competence
10	ICT use for learning
11	Meta-cognition and discriminating school climate
12	School changes
13	Learning time
14	Repetition and immigration index

(see Fig. 8). The first cluster (cluster with label 0) is characterized by a significantly high economic, social, and cultural status (1.60), home resource advantage (-0.84), and meta-cognition (0.78), compared to all students. Note that some of the derived variables correlate negatively with the original ones; this is why the first cluster is considered to have a high home resource advantage with

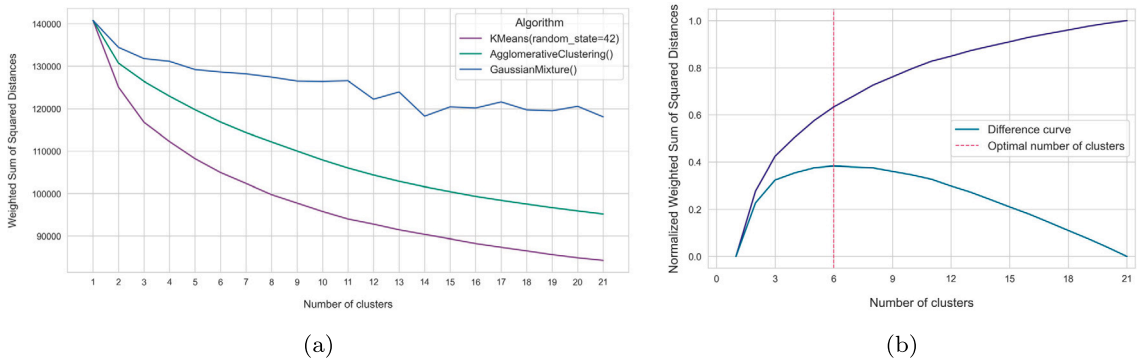

Fig. 7. Comparison of cluster performance depending on k (a) and computation of the optimal k (b).

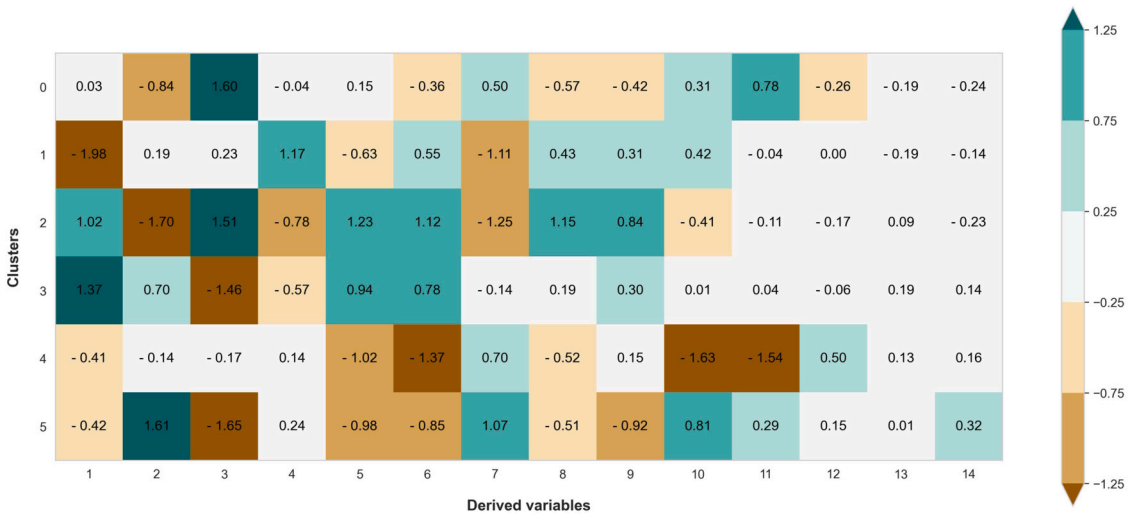

Fig. 8. Heat map of the cluster internal means where global means are 0 for all variables.

an intracluster mean of -0.84 (see correlation detail in Table C1). Students in the second cluster have, on average, a low perception of teacher engagement (-1.98), low psychological well-being (1.17), and a high reading confidence and competence (-1.11).

The third cluster deviates positively from all students considered for the study in all indices that reflect an advantage for academic achievement. These students have high home resource advantage (-1.70), economic, social, and cultural status (1.51), reading confidence and competence (-1.25), work mastery and parents' emotional support (1.23), ICT empowerment (1.15), global cultural attitudes (1.12), perception of teacher engagement (1.02), financial confidence and competence (0.84), and psychological well-being (-0.78). The fourth cluster (cluster with label 3) is characterized by a low economic, social, and cultural status (-1.46), high perception of teacher engagement (1.37), high work mastery and parents' emotional support (0.94), and high global cultural attitudes (0.78).

Students in the fifth cluster have, on average, high use of ICT for learning (-1.63), low meta-cognition and high discriminating school climate (-1.54), low global cultural attitudes (-1.37), and low work mastery and parents' emotional support (-1.02). Lastly, the sixth cluster stands in contrast to the third one as it shows negative deviations across various indices that measure the advantage for academic achievement. These students have low economic, social, and cultural status (-1.65), home resource advantage (1.61), reading confidence and competence (1.07), work mastery and parents' emotional support (-0.98), financial confidence and competence (-0.92), global cultural attitudes (-0.85), and ICT use for learning (0.81).

To provide further insights into the obtained clusters, we trained a classification model using the original variables as predictors and the cluster labels as the target variable. This was accomplished by constructing a four-step pipeline consisting of:

- Train-test random split: The data was divided into a training set (80%) and a testing set (20%).
- Feature selection using the default methodology options. The original variable ESCS (index of economic, social and cultural status) was set to be kept in the feature selection process because of its relevance in education achievement [49].
- Hyperparameter optimization was performed through exhaustive grid search with cross-validation with different values for the number of estimators (30 and 60), *eta* (0.15 and 0.25) and maximum tree depth (3, 5, and 7).

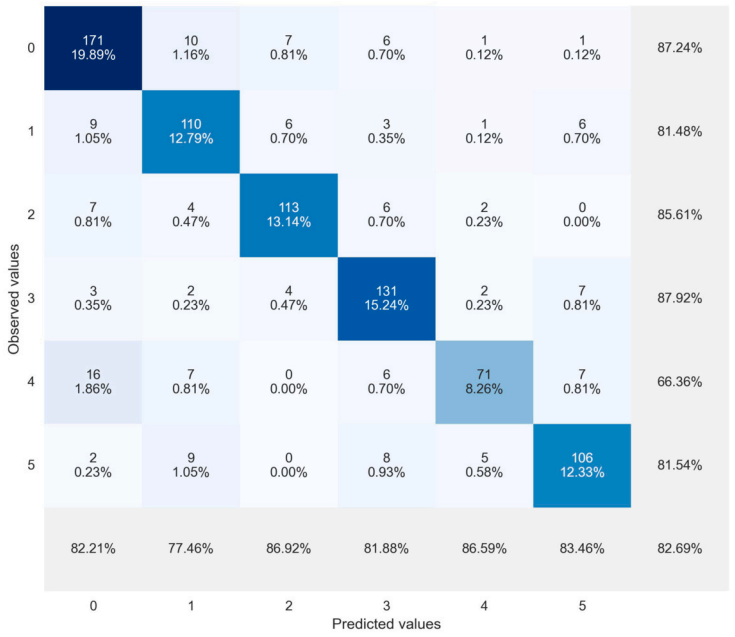

Fig. 9. Confusion matrix for the testing set.

Table 2
SHAP values of the top five predictors.

Variable name	SHAP importance
ESCS	0.850021
HOMEPOS	0.238159
SCREADCOMP	0.146157
STIMREAD	0.144455
AWACOM	0.126995

- A final classification model was trained using the features and hyperparameters from previous steps. We used the xgboost algorithm and AUC as the performance metric.

By applying this pipeline, a subset of 73 variables was selected from the original set of 80. The optimal hyperparameter configuration was achieved with 60 estimators, an *eta* of 0.25, and a maximum tree depth of 3. This configuration yielded an AUC larger than 0.97 for all clusters. In addition, a global accuracy of 0.8269 was achieved with precision values ranging from 0.7746 for cluster 1 to 0.8692 for cluster 2, and recall values ranging from 0.6636 for cluster 4 to 0.8792 for cluster 3 (see Fig. 9) – all these metric values were measured on the testing set.

A key result derived from this pipeline is the identification of the variables exhibiting the highest predictive power for classifying students. These were ESCS (economic, social, and cultural status), HOMEPOS (home possessions), SCREADCOMP (self-concept of reading), STIMREAD (teacher's stimulation of reading engagement perceived by student), and AWACOM (awareness of intercultural communication) as shown in Table 2.

Lastly, we extracted local SHAP values to understand which variables best characterized each cluster. This valuable information was used to assign meaningful cluster names within the educational context of our case study (see Table 3). These names were derived from analyzing the top five variables with the highest predictive power within each cluster and considering the effect that different values of these variables have on the SHAP values (Fig. 10).

This last analysis complements the previous comparison between intracluster means and global means by identifying the variables that characterize each cluster and determining which variables effectively differentiate between different clusters. Local SHAP values help us understand the features that make each cluster unique. For instance, cluster 0 is characterized by having higher levels of meta-cognition and a lower prevalence of a discriminating school climate compared to the overall student sample. However, these characteristics do not contribute significantly to the differentiation of this cluster from others (see Fig. 10a). On the contrary, the variable with the third highest predictive power for cluster 5 is the student's information about careers (INFOCAR) (see Fig. 10f), which shows a low correlation with the derived variables used for clustering.

Table 3
Cluster names.

Cluster label	Name
1	Advantaged tech-dependent students
2	Students with low teacher engagement
3	Advantaged global achievers
4	Resilient students
5	Students in a discriminating school climate
6	Disadvantaged students

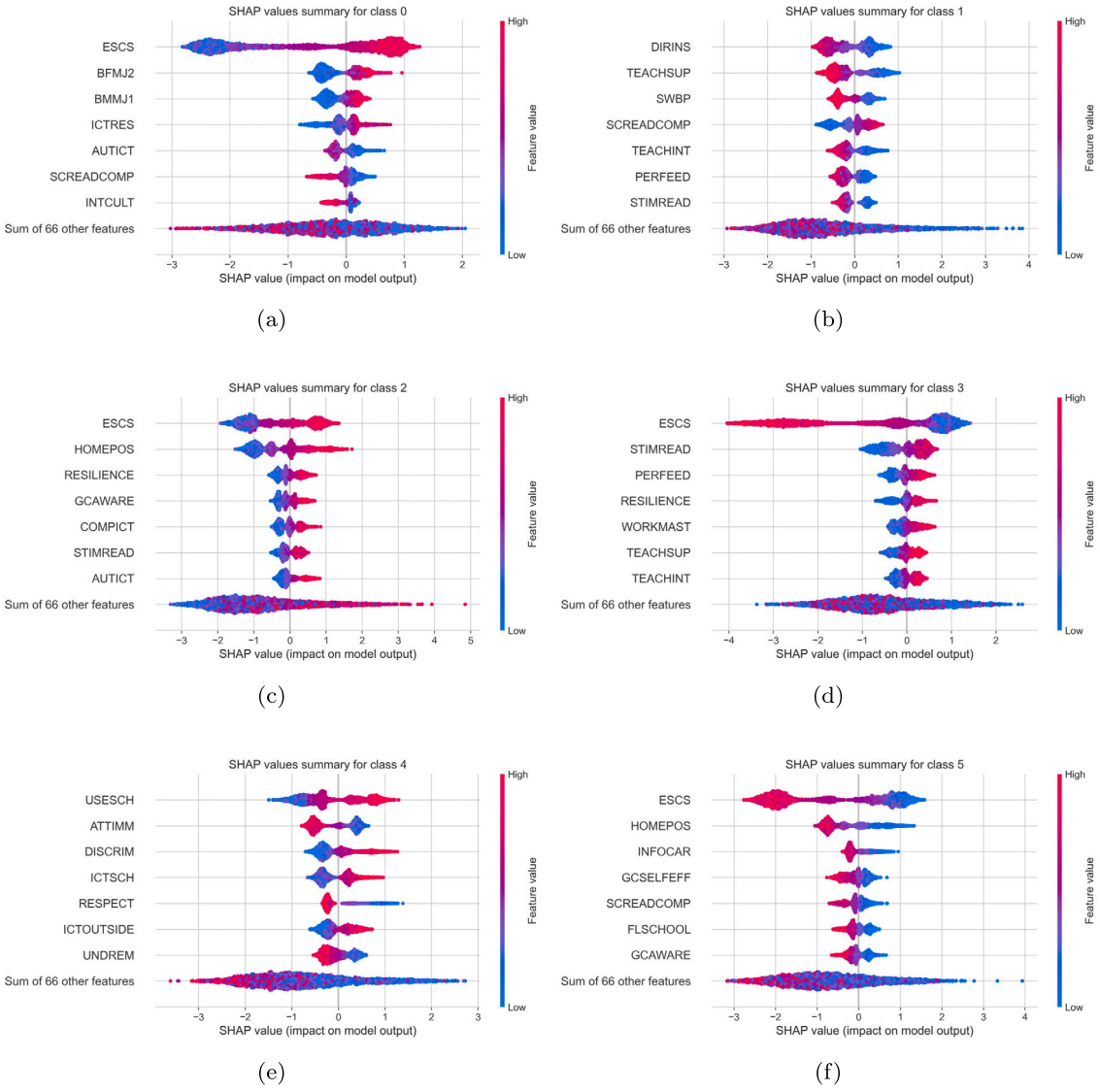

Fig. 10. Local SHAP values summary for all clusters.

8. Conclusion

This paper proposes an integral methodology for explainable cluster analysis that addresses the challenges of data quality and results interpretability. The framework consists of four components applied sequentially, ensuring high-quality and explainable results. Often overlooked in existing methodologies, the data preprocessing component introduces a novel data imputation method for high-dimensional mixed-type data. By leveraging strong relationships between variables and graph theory, clusters of variables are identified for lower-dimensional imputation. The dimensionality reduction component enhances existing methods with regularization techniques, improving the interpretability of derived variables. The clustering component integrates multiple clustering algorithms,

addressing a significant gap in existing explainable cluster analysis methods, often limiting the choice to a single algorithm. The final component is a pipeline for classification models to enhance cluster interpretability with feature selection and SHAP values, supporting multiple tree-based algorithms.

The proposed methodology has been implemented in the open-source Python package Clust-learn, which is designed to be user-friendly and customizable, thus ensuring accessibility across various domains. The package emphasizes visualizations, aiding in the interpretation of results.

We empirically evaluated the proposed methodology using a large-scale assessment database in the field of education that combined numerical and categorical variables with a significant rate of missing values. We used SPCA and MCA for dimensionality reduction, obtaining 14 derived variables that capture various aspects of students' environmental conditions. Clustering analysis (k-means, agglomerative clustering, and Gaussian mixture models) revealed the best performance for k-means with six clusters. Intracluster means were compared against global means, and an xgboost classifier was trained to identify the most predictive variables for each cluster. This allowed for the interpretation and naming of the clusters, providing meaningful insights into the characteristics of students in each cluster.

Future work will focus on expanding the capabilities of the framework. This includes extending support for a broader range of algorithms for dimensionality reduction. Incorporating geospatial data analysis with regional clustering and specialized imputation techniques leveraging spatial relationships is also planned. Additionally, weighted clustering will be implemented to handle scenarios where observations have different weights.

CRediT authorship contribution statement

Miguel Alvarez-Garcia: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Raquel Ibar-Alonso:** Conceptualization, Project administration, Supervision, Writing – review & editing. **Mar Arenas-Parra:** Conceptualization, Funding acquisition, Project administration, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data used for this work is publicly available and can be found at https://doi.org/10.17632/hwj6bfb899.1 [50]. Clust-learn code can be found at https://github.com/malgar/clust-learn.

Clust-learn (Original Data) (GitHub)

PISA 2018 - Sample of Spanish student data (Original Data) (Mendeley Data)

Acknowledgements

This work was partially supported by Fundación para el Fomento en Asturias de la Investigación Científica Aplicada y la Tecnología (FICYT), Project AYUD/2021/50878.

Appendix A. Clust-learn class parameters and attributes

Table A1
`DimensionalityReduction` class parameters and attributes.

Parameter	Type	Description
`df`	`pandas.DataFrame`	Data table containing the data with the original variables.
`num_vars`	`string` or array-like	Numerical variable name(s).
`cat_vars`	`string` or array-like	Categorical variable name(s).
`num_algorithm`	`string`	Algorithm to be used for dimensionality reduction of numerical variables. By default, PCA is used. SPCA is also supported.
`cat_algorithm`	`string`	Algorithm to be used for dimensionality reduction of categorical variables. By default, MCA. For now, only MCA is supported.
`num_kwargs`	`dictionary`	Additional keyword arguments to pass to the model used for numerical variables.
`cat_kwargs`	`dictionary`	Additional keyword arguments to pass to the model used for categorical variables.

Table A1 (*continued*)

Attribute	Type	Description
n_components_	int	Final number of extracted components.
min_explained _variance_ratio_	float	Minimum explained variance ratio. By default, 0.5.
num_trans_	pandas.DataFrame	Extracted components from numerical variables.
cat_trans_	pandas.DataFrame	Extracted components from categorical variables.
num_components_	list	List of names assigned to the extracted components from numerical variables.
cat_components_	list	List of names assigned to the extracted components from categorical variables.
pca_	sklearn .decomposition.PCA	PCA instance used to speed up some computations and for comparison purposes.

Table A2
`Clustering` class parameters and attributes.

Parameter	Type	Description
df	pandas.DataFrame	Data frame containing the data to be clustered.
algorithms	instance or list of instances	Algorithm instances to be used for clustering. They must implement the `fit` and `set_params` methods.
normalize	bool	Whether to apply data normalization for fair comparisons between variables. In case dimensionality reduction is applied beforehand, normalization should not be applied.
Attribute	**Type**	**Description**
dimensions_	list	List of columns of the input data frame.
instances_	dictionary	Pairs of algorithm name and its instance.
metric_	string	The cluster validation metric to be used: [`'inertia'`, `'davies_bouldin_score'`, `'silhouette_score'`, `'calinski_harabasz_score'`].
optimal_config_	tuple	Tuple with the optimal configuration for clustering containing the algorithm name, number of clusters, and value of the validation metric.
scores_	dictionary	Pairs of algorithm name and a list of values of the chosen validation metric for a cluster range.

Table A3
`Classifier` class parameters and attributes.

Parameter	Type	Description
df	pandas.DataFrame	Data frame containing the data.
predictor_cols	list of string	List of columns to use as predictors.
target	numpy.array or list	Values of the target variable.
num_cols	list	List of numerical columns from predictor_cols.
cat_cols	list	List of categorical columns from predictor_cols.
Attribute	**Type**	**Description**
filtered_features_	list	List of columns of the input data frame.
model_	Instance of TransformerMixin and BaseEstimator from sklearn.base	Trained classifier.
X_train_	numpy.array	Train split of predictors.
X_test_	numpy.array	Test split of predictors.
y_train_	numpy.array	Train split of target.
y_test_	numpy.array	Test split of target.
grid_result_	sklearn.model_selection .GridSearchCV	Instance of fitted estimator for hyperparameter tuning.

Appendix B. Variables used in the case study

Table B1
Variables used in the case study.

Variable name	Type	Description
ADAPTIVITY	Num	Adaptation of instruction (WLE)
AGE	Num	Age
ATTIMM	Num	Student's attitudes towards immigrants (WLE)
ATTLNACT	Num	Attitude towards school: learning activities (WLE)
AUTICT	Num	Perceived autonomy related to ICT use (WLE)
AWACOM	Num	Awareness of intercultural communication (WLE)

(*continued on next page*)

Table B1 (*continued*)

Variable name	Type	Description
BEINGBULLIED	Num	Student's experience of being bullied (WLE)
BELONG	Num	Subjective well-being: Sense of belonging to school (WLE)
BFMJ2	Num	ISEI of father
BMMJ1	Num	ISEI of mother
BODYIMA	Num	Body image (WLE)
BSMJ	Num	Students expected occupational status (SEI)
CHANGE	Num	Number of changes in educational biography (Sum)
COGFLEX	Num	Cognitive flexibility/adaptability (WLE)
COMPETE	Num	Competitiveness (WLE)
COMPICT	Num	Perceived ICT competence (WLE)
CULTPOSS	Num	Cultural possessions at home (WLE)
DIRINS	Num	Teacher-directed instruction (WLE)
DISCLIMA	Num	Disciplinary climate in test language lessons (WLE)
DISCRIM	Num	Discriminating school climate (WLE)
DURECEC	Num	Duration in early childhood education and care
EMOSUPS	Num	Parents' emotional support perceived by student (WLE)
ENTUSE	Num	ICT use outside of school (leisure) (WLE)
ESCS	Num	Index of economic, social and cultural status
EUDMO	Num	Eudaemonia: meaning in life (WLE)
FCFMLRTY	Num	Familiarity with concepts of finance (Sum)
FLCONFIN	Num	Confidence about financial matters (WLE)
FLCONICT	Num	Confidence about financial matters using digital devices (WLE)
FLFAMILY	Num	Parental involvement in matters of Financial Literacy (WLE)
FLSCHOOL	Num	Financial education in school lessons (WLE)
GCAWARE	Num	Student's awareness of global issues (WLE)
GCSELFEFF	Num	Self-efficacy regarding global issues (WLE)
GFOFAIL	Num	General fear of failure (WLE)
GLOBMIND	Num	Global-mindedness (WLE)
HEDRES	Num	Home educational resources (WLE)
HISEI	Num	Index highest parental occupational status
HOMEPOS	Num	Home possessions (WLE)
HOMESCH	Num	Use of ICT outside of school (for school work activities) (WLE)
ICTCLASS	Num	Subject-related ICT use during lessons (WLE)
ICTHOME	Num	ICT available at home
ICTOUTSIDE	Num	Subject-related ICT use outside of lessons (WLE)
ICTRES	Num	ICT resources (WLE)
ICTSCH	Num	ICT available at school
IMMIG	Cat	Index Immigration status
INFOCAR	Num	Information about careers (WLE)
INFOJOB1	Num	Information about the labor market provided by the school (WLE)
INFOJOB2	Num	Information about the labor market provided outside of school (WLE)
INTCULT	Num	Student's interest in learning about other cultures (WLE)
INTICT	Num	Interest in ICT (WLE)
JOYREAD	Num	Joy/Like reading (WLE)
LMINS	Num	Learning time (minutes per week) - test language
MASTGOAL	Num	Mastery goal orientation (WLE)
METASPAM	Num	Meta-cognition: assess credibility
METASUM	Num	Meta-cognition: summarizing
MMINS	Num	Learning time (minutes per week) - Mathematics
PAREDINT	Num	Index highest parental education (international years of schooling scale)
PERCOMP	Num	Perception of competitiveness at school (WLE)
PERCOOP	Num	Perception of cooperation at school (WLE)
PERFEED	Num	Perceived feedback (WLE)
PERSPECT	Num	Perspective-taking (WLE)
REPEAT	Cat	Grade Repetition
RESILIENCE	Num	Resilience (WLE)
RESPECT	Num	Respect for people from other cultures (WLE)
SCCHANGE	Num	Number of school changes
SCREADCOMP	Num	Self-concept of reading: Perception of competence (WLE)
SCREADDIFF	Num	Self-concept of reading: Perception of difficulty (WLE)
SMINS	Num	Learning time (minutes per week) - <science>
SOCONPA	Num	Social Connections: Parents (WLE)
SOIAICT	Num	ICT as a topic in social interaction (WLE)
ST004D01T	Cat	Student (Standardized) Gender
STIMREAD	Num	Teacher's stimulation of reading engagement perceived by student (WLE)
STUBMI	Num	Body mass index of student
SWBP	Num	Subjective well-being: Positive affect (WLE)
TEACHINT	Num	Perceived teacher's interest (WLE)
TEACHSUP	Num	Teacher support in test language lessons (WLE)

Table B1 (*continued*)

Variable name	Type	Description
TMINS	Num	Learning time (minutes per week) - in total
UNDREM	Num	Meta-cognition: understanding and remembering
USESCH	Num	Use of ICT at school in general (WLE)
WEALTH	Num	Family wealth (WLE)
WORKMAST	Num	Work mastery (WLE)

Appendix C. Dimensionality reduction derived variables

Table C1
Principal components extracted from numerical variables and the original variables with a correlation larger than 0.5 in absolute value.

	Name	Original variable	ρ
1	Students' perception of teacher engagement	TEACHSUP	0.7761
		ADAPTIVITY	0.7694
		STIMREAD	0.7564
		DIRINS	0.7514
		TEACHINT	0.7427
		PERFEED	0.6616
2	Student home resource advantage	HOMEPOS	-0.9469
		ICTRES	-0.8636
		WEALTH	-0.8553
		ESCS	-0.6562
		ICTHOME	-0.6505
		HEDRES	-0.5922
		CULTPOSS	-0.5480
3	Economic, social and cultural status	HISEI	0.9419
		ESCS	0.9196
		BMMJ1	0.8337
		BFMJ2	0.8018
		PAREDINT	0.7224
4	Psychological well-being	BODYIMA	-0.6302
		RESILIENCE	-0.6025
		GFOFAIL	0.5983
		SWBP	-0.5889
		BELONG	-0.5124
5	Work mastery and parents' emotional support	MASTGOAL	0.6489
		WORKMAST	0.6473
		EMOSUPS	0.5797
		ATTLNACT	0.5496
6	Global cultural attitudes	INTCULT	0.6878
		RESPECT	0.6494
		PERSPECT	0.6398
		GLOBMIND	0.5835
		ATTIMM	0.5688
7	Reading confidence and competence	SCREADCOMP	-0.7712
		SCREADDIFF	0.6611
		JOYREAD	-0.6125
		GCSELFEFF	-0.5297
8	ICT empowerment	AUTICT	0.7729
		COMPICT	0.7634
		INTICT	0.6395
		SOIAICT	0.5883
9	Financial confidence and competence	FCFMLRTY	0.6761
		FLSCHOOL	0.6735
		FLCONFIN	0.5731
		FLCONICT	0.5583
10	ICT use for learning	USESCH	-0.7028
		HOMESCH	-0.6327
		ICTOUTSIDE	-0.6080
		ICTCLASS	-0.5706
11	Meta-cognition and discriminating school climate	METASUM	0.5732
		DISCRIM	-0.5638
12	School changes	SCCHANGE	0.9482
		CHANGE	0.9480
13	Learning time	MMINS	0.8516
		LMINS	0.8305

Table C2
Components extracted from categorical variables and the original variables with a η_p^2 larger than 0.14.

	Name	Original variable	η_p^2
14	Repetition and immigration index	REPEAT	0.6150
		IMMIG	0.5717

References

[1] A.E. Ezugwu, A.M. Ikotun, O.O. Oyelade, L. Abualigah, J.O. Agushaka, C.I. Eke, A.A. Akinyelu, A comprehensive survey of clustering algorithms: state-of-the-art machine learning applications, taxonomy, challenges, and future research prospects, Eng. Appl. Artif. Intell. 110 (2022) 104743, https://doi.org/10.1016/j.engappai.2022.104743.

[2] M. Injadat, A. Moubayed, A.B. Nassif, A. Shami, Machine learning towards intelligent systems: applications, challenges, and opportunities, Artif. Intell. Rev. 54 (5) (2021) 3299–3348, https://doi.org/10.1007/s10462-020-09948-w.

[3] K. Kirchner, J. Zec, B. Delibasic, Facilitating data preprocessing by a generic framework: a proposal for clustering, Artif. Intell. Rev. 45 (3) (2016) 271–297, https://doi.org/10.1007/s10462-015-9446-6.

[4] W. Ding, M. Abdel-Basset, H. Hawash, A.M. Ali, Explainability of artificial intelligence methods, applications and challenges: a comprehensive survey, Inf. Sci. 615 (2022) 238–292, https://doi.org/10.1016/j.ins.2022.10.013.

[5] D. Li, Y. Liu, J. Huang, Z. Wang, A trustworthy view on explainable artificial intelligence method evaluation, Computer 56 (4) (2023) 50–60, https://doi.org/10.1109/MC.2022.3233806.

[6] O. Loyola-Gonzalez, A.E. Gutierrez-Rodriguez, M.A. Medina-Perez, R. Monroy, J.F. Martinez-Trinidad, J.A. Carrasco-Ochoa, M. Garcia-Borroto, An explainable artificial intelligence model for clustering numerical databases, IEEE Access 8 (2020) 52370–52384, https://doi.org/10.1109/ACCESS.2020.2980581.

[7] S. Bobek, M. Kuk, M. Szelazek, G.J. Nalepa, Enhancing cluster analysis with explainable ai and multidimensional cluster prototypes, IEEE Access 10 (2022) 101556–101574, https://doi.org/10.1109/ACCESS.2022.3208957.

[8] S. Bandyapadhyay, F.V. Fomin, P.A. Golovach, W. Lochet, N. Purohit, K. Simonov, How to find a good explanation for clustering?, Artif. Intell. 322 (2023) 103948, https://doi.org/10.1016/j.artint.2023.103948.

[9] A. Morichetta, P. Casas, M. Mellia, Explain-it: towards explainable ai for unsupervised network traffic analysis, in: Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks, Big-DAMA'19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 22–28.

[10] G. Feng, M. Fan, Research on learning behavior patterns from the perspective of educational data mining: evaluation, prediction and visualization, Expert Syst. Appl. 237 (2024) 121555, https://doi.org/10.1016/j.eswa.2023.121555.

[11] G.S. Halford, R. Baker, J.E. McCredden, J.D. Bain, How many variables can humans process?, Psychol. Sci. 16 (1) (2005) 70–76, https://doi.org/10.1111/j.0956-7976.2005.00782.

[12] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, ACM Trans. Knowl. Discov. Data 6 (1) (2012) 3, https://doi.org/10.1145/2133360.2133363.

[13] I. Tiobe, Tiobe index for January 2024, 2024.

[14] I. Triguero, D. Molina, J. Poyatos, J. Del Ser, F. Herrera, General purpose artificial intelligence systems (gpais): properties, definition, taxonomy, societal implications and responsible governance, Inf. Fusion 103 (2024) 102135, https://doi.org/10.1016/j.inffus.2023.102135.

[15] T.T. Le, W. Fu, J.H. Moore, Scaling tree-based automated machine learning to biomedical big data with a feature set selector, Bioinformatics 36 (1) (2020) 250–256, https://doi.org/10.1093/bioinformatics/btz470.

[16] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, in: Advances in Neural Information Processing Systems, vol. 28, 2015, pp. 2962–2970.

[17] S. Brugman, Pandas-profiling: exploratory data analysis for python, pandas-profiling version 3.6.2, https://pandas-profiling.ydata.ai/docs/master/index.html, 2019.

[18] A. Bilogur, Missingno: a missing data visualization suite, J. Open Sour. Softw. 3 (22) (2018) 547, https://doi.org/10.21105/joss.00547.

[19] M. Ali, PyCaret: an open source, low-code machine learning library in python, pyCaret version 1.0.0, https://www.pycaret.org, April 2020.

[20] W.E. Marcilio, D.M. Eler, From explanations to feature selection: assessing shap values as feature selection mechanism, in: 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 2020, pp. 340–347.

[21] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc., 2017, https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[22] R.R. Andridge, K.J. Thompson, Adapting nearest neighbor for multiple imputation: advantages, challenges, and drawbacks, J. Surv. Stat. Methodol. 11 (1) (2023) 213–233, https://doi.org/10.1093/jssam/smab058.

[23] E. Tavazzi, S. Daberdaku, R. Vasta, A. Calvo, A. Chio, B.D. Camillo, Exploiting mutual information for the imputation of static and dynamic mixed-type clinical data with an adaptive k-nearest neighbours approach, BMC Med. Inform. Decis. Mak. 20 (2020) 174, https://doi.org/10.1186/s12911-020-01166-2.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[25] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, R.B. Altman, Missing value estimation methods for dna microarrays, Bioinformatics 17 (6) (2001) 520–525, https://doi.org/10.1093/bioinformatics/17.6.520.

[26] J.K. Dixon, Pattern-recognition with partly missing data, IEEE Trans. Syst. Man Cybern. 9 (10) (1979) 617–621, https://doi.org/10.1109/TSMC.1979.43c10090.

[27] S. Xia, Z. Xiong, Y. Luo, G. Zhang, et al., Effectiveness of the Euclidean distance in high dimensional spaces, Optik 126 (24) (2015) 5614–5619, https://doi.org/10.1016/j.ijleo.2015.09.093.

[28] L.H. Nguyen, S. Holmes, Ten quick tips for effective dimensionality reduction, PLoS Comput. Biol. 15 (6) (2019) e1006907, https://doi.org/10.1371/journal.pcbi.1006907.

[29] J. Mairal, F. Bach, J. Ponce, G. Sapiro, Online learning for matrix factorization and sparse coding, J. Mach. Learn. Res. 11 (2010) 19–60.

[30] M. Halford, Prince, Version 0.7.1, https://pypi.org/project/prince/, Apr. 2022.

[31] J.-P. Benzecri, Sur le calcul des taux d'inertie dans l'analyse d'un questionnaire, addendum et erratum a [bin. mult.], Cah. Anal. Donnees 4 (3) (1979) 377–378.

[32] M. Greenacre, Correspondence Analysis in Practice, Academic Press, London, 1993.

[33] F. Liu, Y. Deng, Determine the number of unknown targets in open world based on elbow method, IEEE Trans. Fuzzy Syst. 29 (5) (2021) 986–995, https://doi.org/10.1109/TFUZZ.2020.2966182.

[34] V. Satopaa, J. Albrecht, D. Irwin, B. Raghavan, Finding a "kneedle" in a haystack: detecting knee points in system behavior, in: 2011 31st International Conference on Distributed Computing Systems Workshops, 2011, pp. 166–171.

[35] C. Hennig, What are the true clusters?, Pattern Recognit. Lett. 64 (2015) 53–62, https://doi.org/10.1016/j.patrec.2015.04.009.

[36] J.A. Hartigan, M.A. Wong, A k-means clustering algorithm, JSTOR: Appl. Stat. 28 (1) (1979) 100–108.

[37] D. Arthur, S. Vassilvitskii, K-means plus plus: the advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 1027–1035.
[38] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-1 (2) (1979) 224–227, https://doi.org/10.1109/TPAMI.1979.4766909.
[39] P.J. Rousseeuw, Silhouettes - a graphical aid to the interpretation and validation of cluster-analysis, J. Comput. Appl. Math. 20 (1987) 53–65, https://doi.org/10.1016/0377-0427(87)90125-7.
[40] T. Calinski, J. Harabasz, A dendrite method for cluster analysis, Commun. Stat., Simul. Comput. 3 (1) (1974) 1–27.
[41] J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection: a data perspective, ACM Comput. Surv. 50 (6) (2017), https://doi.org/10.1145/3136625.
[42] S.M. Lundberg, G. Erion, H. Chen, A. DeGrave, J.M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, S.-I. Lee, From local explanations to global understanding with explainable ai for trees, Nat. Mach. Intell. 2 (1) (2020) 56–67, https://doi.org/10.1038/s42256-019-0138-9.
[43] L.S. Shapley, A value for n-person games, Princeton University Press, Princeton, 1953, pp. 307–318.
[44] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32, https://doi.org/10.1023/A:1010933404324.
[45] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Mach. Learn. 46 (1) (2002) 389–422, https://doi.org/10.1023/A:1012487302797.
[46] T. Chen, C. Guestrin, Xgboost: a scalable tree boosting system, in: KDD'16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[47] C. Bentejac, A. Csorgo, G. Martinez-Munoz, A comparative analysis of gradient boosting algorithms, Artif. Intell. Rev. 54 (3) (2021) 1937–1967, https://doi.org/10.1007/s10462-020-09896-5.
[48] OECD, PISA 2018 Technical Report, 2020, https://www.oecd.org/pisa/data/pisa2018technicalreport/.
[49] J.S. Coleman, E.Q. Campbell, C.J. Hobson, J. McPartland, A.M. Mood, F.D. Weinfeld, R.L. York, Equality of Educational Opportunity Washington, US Government Printing Office, DC, 1966, pp. 1–32.
[50] M. Alvarez-Garcia, R. Ibar-Alonso, M. Arenas-Parra, Pisa 2018 - sample of Spanish student data, Mendeley Data V1, https://doi.org/10.17632/hwj6bfb899.1, 2023.