Check for updates

SOFTWARE TOOL ARTICLE

selfX: An open-source R package with autonomic abilities for data mining

[version 1; peer review: 2 not approved]

M Galende [1,2], A Corral [1], A Reñones [1]

[1]CARTIF Centro Tecnológico, Parque Tecnológico de Boecillo, parc. 205, Boecillo, Valladolid, 47151, Spain
[2]Department of System Engineering and Automatic Control, School of Inustrial Engineeringd, Universidad de Valladolid, Valladolid, 47011, Spain

v1 **First published:** 29 Jul 2024, **4**:156
https://doi.org/10.12688/openreseurope.18070.1
Latest published: 29 Jul 2024, **4**:156
https://doi.org/10.12688/openreseurope.18070.1

Abstract

This article describes the key characteristics of the functionality implemented in the “selfX” R software package, developed under the European s-X-AIPI project as an open-source software and offered to the R community for use and enhancement.

Specifically this package offers a range of more autonomic functions for both data exploration and model training activities, such as unsupervised and supervised feature selection, unsupervised anomaly detection and/or predictive algorithm selection. Moreover, it includes complementary functions for data cleaning, statistical tests and outlier and missing values treatment.

Plain language summary

The article presents "selfX," an open-source software tool designed to make data analysis and model creation easier. In simple terms, this tool is to make the process of working with data more automatic and accessible for both experts and beginners. This is especially useful for people working on artificial intelligence projects who need to maintain and update their models with new data efficiently.

"selfX" is open-source, meaning it is available for anyone to use and improve, contributing to the community of data scientists and developers.

Keywords

selfX, Data Exploration, Feature Selection, Model Training, Anomaly Detection, Artificial Intelligence

Open Peer Review

Approval Status ✗ ✗

	1	2
version 1 29 Jul 2024	✗ view	✗ view

1. **Andrzej Dudek**, Uniwersytet Ekonomiczny we Wrocławiu, Wrocław, Poland

2. **Caspar J. Van Lissa**, Tilburg University, Tilburg, The Netherlands

Any reports and responses or comments on the article can be found at the end of the article.

This article is included in the Horizon Europe gateway.

This article is included in the Self-Optimized AI for Sustainable & Agile Manufacturing in European Process Industries collection.

Corresponding author: M Galende (margal.cartif@gmail.com)

Author roles: Galende M: Conceptualization, Investigation, Methodology, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Corral A**: Software, Validation; **Reñones A**: Funding Acquisition, Project Administration, Supervision

Competing interests: No competing interests were disclosed.

Grant information: This work has been supported by European Union's Horizon Europe research and innovation programme under grant agreement N° 101058715 (self-X Artificial Intelligence for European Process Industry digital transformation [s-X-AIPI]).Co-funded by the European Union and Made In Europe. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or HaDEA. Neither the European Union nor the granting authority can be held responsible for them.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2024 Galende M *et al*. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Galende M, Corral A and Reñones A. **selfX: An open-source R package with autonomic abilities for data mining [version 1; peer review: 2 not approved]** Open Research Europe 2024, **4**:156 https://doi.org/10.12688/openreseurope.18070.1

First published: 29 Jul 2024, **4**:156 https://doi.org/10.12688/openreseurope.18070.1

1. Introduction

This document compiles a detailed description of all the functionalities included in the "selfX" R package developed within the "self-X Artificial Intelligence for European Process Industry digital transformation (s-X-AIPI)" project[1]. This R package is implemented as open-source software (OOS) and offered to the R community for use and improvement.

s-X-AIPI project is driven by autonomic computing, which is inspired by the human body's autonomic nervous system, which enables systems to self-manage and adapt with minimal human intervention. This concept underpins the self-X abilities in the "selfX" R package, facilitating automated creation of data-based AI applications. As part of the s-X-AIPI project, various AI applications have been developed for industrial scenarios, such as the manufacturing of asphalt mixes. These AI applications are also driven by the autonomic computing principles, and exhibit self-x principles, leading to the creation of several AI procedures and tools, like those documented here.

When a data scientist is faced with the task of developing a data-based AI application, there are a set of fundamental tasks that need to be addressed, including Feature Selection as major task in Data Exploration and/or Algorithm Selection and Anomaly Detection in Model Training (Figure 1). In addition, other tasks such as Data Cleaning, Statistical Tests or Outliers Treatment are also necessary to prepare the data.

Ensuring that these tasks can be carried out in the most autonomic way possible is the final objective of the functions included in this R package, whose end user profile is twofold:

1. Former, data scientists who will allow to perform more autonomous maintenance of AI applications, so that the re-training of the AI systems faced with new data be as autonomic as possible.
2. Latter, non-expert AI developers who will allow to perform typical activities of an AI pipeline in an autonomic way.

Following fundamental aspects to be covered within any data mining project are covered within this R package:

- **self-X autonomic data preparation**: "selfX" R package includes the implementation of several functions to perform data preparation activities in a more autonomic way (Section Data Preparation):
    - Data Cleaning, focus on (1) search and remove of variables with constant values and (2) search and remove of examples with missing values.
    - Statistical Test, both normality and correlation hypothesis tests.
    - Outliers and Missing Values Treatment based on ad-hoc expert rules.
- **self-X autonomic supervised and unsupervised feature selection**: "selfX" R package includes the implementation of several functions to perform feature selection (application of various methodologies and combination of results for decision making) within data exploration activities in a more autonomic way (Sections self-X autonomic unsupervised and supervised feature selection, Decision-making process and Feature Selection).
- **self-X autonomic unsupervised anomaly detection**: "selfX" R package includes the implementation of an ad-hoc procedure based on outlier analysis and CUSUM RLS filter to detect anomalies in the data in a more autonomic way (Sections self-X autonomic unsupervised anomaly detection, Decision-making process and Unsupervised Anomaly Detection).
- **self-X autonomic predictive algorithm selection**: "selfX" R package includes the implementation of several functions to perform predictive algorithm selection (application of various algorithms and selection of the most promising one) within model training activities in a more autonomic way (Sections self-X autonomic predictive algorithm selection, Decision-making process and Predictive Algorithm Selection).

[1] This project receives funding in the European Commission's Horizon Europe Research Programme under Grant Agreement Number 101058715. https://cordis.europa.eu/project/id/101058715, https://s-x-aipi-project.eu/

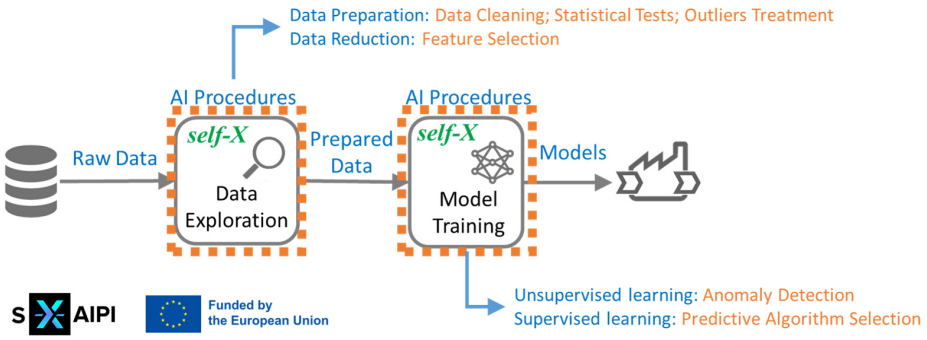


Figure 1. AI procedures included in the "selfX" R package.

Section self-X autonomic procedures: theoretical background summarizes the theoretical part of the models and methods used within autonomic feature and algorithm selection, specifically well-known techniques as Principal Components Analysis (PCA), Factor Analysis (FA), RELIEF, Outliers Analysis, Cumulative Sum (CUSUM) Recursive Least Squares (RLS) filter, Regression, Neural Networks, Support Vector Machines (SVM), Random Forest and Fuzzy Rule-Based Systems. Moreover, the decision-making process based on OWA operator and used in all the selection functions is presented. Section self-X methodology & implementation introduces the methodology and present the implementations of all functions included in the package, along with usage examples. Finally, section Conclusions to summarize the main findings and contributions.

2. self-X autonomic procedures: theoretical background

There are several methodologies proposed for the development of AI applications, and all of them include a "Data Exploration" phase in which the cleaning of the data, the treatment of outlier data, the analysis of correlations and feature selection (among others) are carried out. Furthermore, algorithm selection in "Model Training" is also crucial to obtain accurate predictions and valuable insights to perform informed decision-making.

In this section theoretical principles related with main models and methods used within "Feature Selection", "Anomaly Detection" and "Algorithm Selection" tasks are explained, as well as decision-making process proposed that guides the final selection and that is applied in multiple steps of the proposed methodologies. The rest of the functions included in this package, related to data cleaning, statistical tests or outliers treatment, are presented directly in the following section.

2.1. self-X autonomic unsupervised and supervised feature selection

Feature Selection in Data Mining is a complementary technique that also helps to reduce the dimensionality of the data on which to subsequently apply the modelling algorithms, i.e., to enhance the quality and efficiency of clustering and/or predictive models, enabling them to focus on the most relevant and significant features for the given problem. In this way, the quality of the results is improved, their visualization and interpretation is facilitated, and the problems associated with high-dimensional data sets are alleviated.

With the aim of making feature selection more autonomic, an ad-hoc procedure was defined as part of the Asphalt Use Case within the s-X-AIPI project and subsequently adapted for general use (Figure 2). The approach can be summarized as follows:

1. Initially, well-known techniques (PCA/FA for unsupervised and RELIEF for supervised approaches) are used to select the most important features under different criteria.

2. Subsequently, an OWA operator is applied for decision making to select the three most relevant features.

The main theoretical foundations of PCA, FA and RELIEF are explained below, as well as how to evaluate the importance of the variables based on the results obtained with these methods.

2.1.1. PCA: Principal Component Analysis

PCA is a feature extraction method that can be used for feature selection[1,2]. PCA is a statistical feature extraction technique frequently used for dimensionality reduction in unsupervised classification problems[1]. It is capable of generating a new space of reduced dimensions with extracted features, and which therefore usually lacks physical meaning. The transformation performed by PCA is linear, although there are versions of the method that implement non-linear transformations, as is the case with Kernel PCA. The PCA extractor bases its behaviour on obtaining the principal components from the covariance matrix, so that these components correspond to the eigenvalues of the covariance matrix, ordered from highest to lowest eigenvalue. The first principal component obtained will correspond to the predominant direction of distribution of the data. The second principal

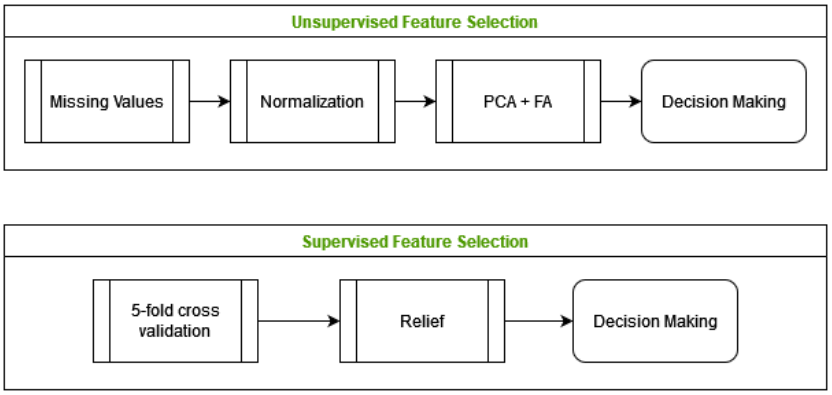


Figure 2. Feature selection: general methodology.

component will correspond to the second most predominant direction of the data, and so on. According to all this, the behaviour of the PCA is based on two general ideas:

1. Transform the input data into a lower dimension.
2. Preserve the maximum variance of original data.

In the end, principal components obtained are a linear combination of the original features of the problem. By applying the matrix of principal components to the original matrix, the new matrix of reduced features is obtained to work with.

From the data obtained in the feature extraction it is possible to propose a selection of them. The principal components are linear combinations of all available features. However, these features do not have to be of the same importance for the formation of the components. Some of these features will be critical and will be assigned greater weight in the combination. Other features will be redundant, irrelevant, or insignificant, and will be assigned a lower evaluation in the combination[1]. In addition, the most important features will be associated with the first principal components obtained, while the least important will be associated with the last principal components. And all this thanks to the ability of PCA to discriminate directions with the greatest variance in a given data set[2].

Among all the main components obtained, it is necessary to determine the appropriate number of extracted features to work with. According to 1, a rule that is usually applied to determine the dimension of the new space obtained is to consider those eigenvalues that provide at least 90% of explained variance or whose value is higher than the average of eigenvalues.

2.1.2. FA: Factor Analysis

FA is a multivariate method that aims to express a set of initial observable variables *n* as a linear combination of new hypothetical or latent variables, called factors, plus an error expression[3]. FA is similar to PCA[4], although they are not identical. The main difference is that FA uses regression modelling techniques while PCA is a descriptive statistical technique.

FA is a feature extraction method that can be used for dimensionality reduction or feature selection. Intuitively, the idea is based on transforming an n-dimensional space into another with a lower dimension in which the loss of information is as small as possible. To do this, the eigenvectors that define the new base with dimension *p* less than *n* must be selected. Each of these eigenvectors represents a feature, which could be contained in the initial feature space or be a combination thereof.

In order to use FA to perform feature selection, an exploratory analysis will be used in which the following parameters must be defined:

- The number of latent factors. For this there are a series of techniques among which Kaiser Rule, Horn's Parallel Analysis (PA) or the Minimum Partial Average (*Velicer* MAP) can be highlighted[5].
- The type of factorization. There are three fundamental methods: Principal Component Analysis (PCA), Principal Factor Analysis (PAF), or Maximum *Neighborhood* (ML)[6].
- The factorial rotation method. The methods to use can be divided into orthogonal rotations such as *Varimax* or *Quartimax* or oblique rotations such as *Oblimin* or *Promax*[7].

At the end of the process, a matrix of loadings is obtained on which to work to determine which of the initial features are most important. An initial interpretation of the results can be derived from the weights of the loadings in the factorial matrix. A high loading value for a variable within a factor suggests that the variable accounts for a significant portion of the variance explained by that factor. Following this idea, one way to determine the importance of the features is to operate on the loads to obtain the *communality* value of them. Based on this value of *communality*, an importance value of the feature can be established.

2.1.3. RELIEF

RELIEF is a feature selection technique used in preparing data for machine learning tasks[8,9]. Its main goal is to identify the features that are most distinctive between instances of different classes, so it is particularly useful for classification tasks.

Its operation is based on evaluating the importance of features according to how well they differentiate between instances that are close to each other. For each instance, the algorithm finds its nearest neighbour of the same class (the "*near neighbour*") and its nearest neighbour of a different class (the "*far neighbour*"). RELIEF then adjusts the importance score of each feature based on how much that feature differs between the current instance, its "*near neighbour*", and its "*far neighbour*".

The method is designed to handle different types of data, including binary, nominal, and continuous. One of the big advantages of RELIEF is that it not only considers features individually, but also takes into account interactions between features. This makes it very effective for problems where features have dependencies on each other.

2.1.4. Evaluation of the importance of variables

The methods applied to assign an importance value to the variables and thus establish a ranking between them are explained below:

Evaluation from PCA: The first step is to apply PCA on the covariance matrix and once the eigenvalues are obtained, evaluate the importance of the initial features based on the principal components considered. In this specific approach the following number of main components to carry out the evaluation have been considered:

1. the first principal component.
2. all those principal components that are above the mean eigenvalues.
3. all principal components that contain 90% of the explained variance.

The result is a list of the initial features, an evaluation of their importance and the ranking assigned to each of them.

Evaluation from FA: Once FA is applied, a loading matrix is obtained from which the *communality* can be calculated and the importance of the features can be evaluated. As with PCA, the result is a list of the initial features, an evaluation of their importance and the assigned ranking considering all factors. The result is a list of the initial features and their assigned weights.

Evaluation from RELIEF: RELIEF is applied following a 5-fold cross-validation strategy to assign an individual evaluation measure to each feature. Then, different direct search algorithms (cut-off method) are applied to select which are the features with the greatest evaluations. Specifically, the following methods are applied, which allow different aspects of the features to be highlighted, such as their ability to differentiate between classes or their robustness against noise and outliers:

1. Select K best: selects the $k = 3$ features with the greatest evaluations.
2. Select percentile: selects *percentage* = 80% of the total number of available features.
3. Select threshold: selects the features whose evaluation is over and under *threshold* = 0.1.
4. Select threshold range: selects the features whose evaluation is over a $threshold = (((min - max) * p.threshold) + max)$, $p.threshold = 0.8$
5. Select difference: selects features (in descending order from the best evaluation measure to the lowest) until evaluation difference is over $d.threshold = 0.2$.
6. Select slope: selects features (in descending order from the best evaluation measure to the lowest) until the slope to the next feature is over $s.threshold = 1.5$.

Finally, the importance value of each feature is assigned based on the number of times it has been selected.

2.2. self-X autonomic unsupervised anomaly detection

Anomaly Detection in an unsupervised way is an important task because it helps improve process efficiency, prevent errors, and optimize the performance of data-driven applications.

With the aim of making unsupervised anomaly detection more autonomic, an ad-hoc procedure was defined as part of the Asphalt Use Case within the s-X-AIPI project and subsequently adapted for general use (Figure 3). The approach can be summarized as follows:

1. Several well-known techniques (outliers analysis and CUSUM RLS filter) are applied to detect outliers and abrupt changes (so, anomalies) in the dataset.
2. OWA operator is applied for decision making to validate anomalies.

The main theoretical foundations of outlier analysis and CUSUM RLS filter are explained below.

2.2.1. Outlier analysis

This analysis is based on statistical principles that identifies as outliers those values such that: $value < Q_1 - threshold * (Q_3 - Q_1)$ or $value > Q_3 + threshold * (Q_3 - Q_1)$, where Q_1, Q_3 are the first and third quartiles respectively, and *threshold* could be 3 to detect extreme outliers or 1.5 to detect slight ones[10]. Outliers are those values that must be detected and analyzed to determine whether or not they are anomalies.

2.2.2. CUSUM RLS filter: Cumulative Sum Recursive Least Squares filter

CUSUM RLS filter[11] is a change detector method based on Cumulative Sum (CUSUM) test and Recursive Least Squares (RLS) estimation. This filter assumes that the values of the variables change slowly over time, but that sometimes more abrupt changes occur. The latter are the ones that must be detected and analysed whether or not they are anomalies.

As shown in the formulation of the CUSUM RLS filter (Table 1), the procedure is based on:

1. Calculate the estimated value of a variable for the next step (t) using RLS method (based on a moving

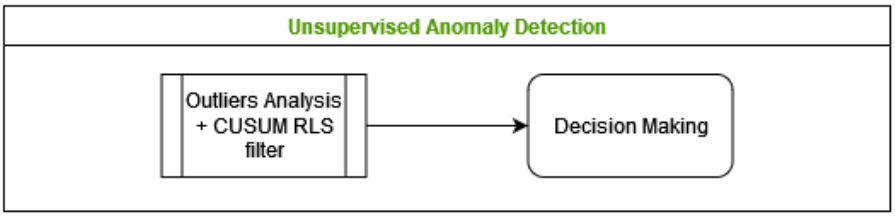


Figure 3. Anomaly Detection: general methodology.

Table 1. CUSUM RLS filter.

$\hat{\theta}_t$	=	$\lambda * \hat{\theta}_{t-1} + (1-\lambda) * y_t$
ϵ_t	=	$y_t - \hat{\theta}_{t-1}$
s_t^1	=	ϵ_t
s_t^2	=	$-\epsilon_t$
g_t^1	=	$\max(g_{t-1}^1 + s_t^1 - v, 0)$
g_t^2	=	$\max(g_{t-1}^2 + s_t^2 - v, 0)$
"Alarm" if $g_t^1 > h$ or $g_t^2 > h$		
After an "Alarm", reset $g_t^1 = 0$, $g_t^2 = 0$, and $\hat{\theta}_t = y_t$		

average of forgetting factor λ) and the error between the estimated value on the previous step ($t-1$) and the real value (t).

2. Apply the CUSUM test to the error to determine whether an abrupt change is detected.

In summary, the method learns normal behaviour and when abrupt changes are detected the "Alarm" is activated.

Setting Parameters: There are three design parameters that must be adjusted:

- Forgetting factor (λ) and drift (v): used to avoid false "Alarms".
    - λ must be close to unity so that the parameters obtained are calculated from a large number of previous samples, so their variation will be smooth. According to 12 if N is the number of previous samples to take into account, then $\lambda = \frac{N}{N+1}$
    - v must be one half of the expected change.
- Threshold (h): to control the desired delay for detection.

Additional improvement: Besides the previous procedure, and with the aim of improving the reliability in the generation of the "Alarms", it can establish a consecutive repetition of abrupt changes before declaring its occurrence. To implement it, the formulation must keep track of the number of consecutive abrupt changes on each variable and triggers an "Alarm" only if the maximum number of consecutive changes is exceeded. This value (*MaxChanges*) must also be adjusted individually.

2.2.3. Detection of anomalies

The previous methods can be used to establish if the values of a set of variables at a given instant of time are or not an anomaly:

Anomalies detection from outlier analysis: this analysis can be applied in a multivariate analysis so that:

1. The outlier analysis is applied to each variable individually.
2. Each variable requires its own Q_1, Q_3, but *threshold* must be the same for all of them.
3. A set of values is considered anomalous if any of them are outliers individually.

The result is a logical value *TRUE*/*FALSE* indicating if there is or not an outlier, so an anomaly is detected.

Anomalies detection from CUSUM RLS filter: Also the filter can be used to analyse the values that multiple variables take at the same instant of time. In that cases:

1. The previous procedures are applied to each of the values of each variable.
2. Since each variable require its own parameter set (N, v, h, *MaxChanges*), it is necessary to adjust these values to the behaviour of each variable individually.
3. A set of values generates an "Alarm" if any of them generates one and then an anomaly is detected.

Besides, in order to apply an unsupervised self-X version of the filter, it is necessary to make a prior adjustment of the parameters. This parameterization has been established based on expert knowledge and is:

- The number of previous samples to take into account to make estimations N is set to 10 by default, since manufacturing systems are usually slow systems. In general, low values of N are established for fast systems, and high values for slow systems. The aim is to achieve an estimation $\hat{\theta}_t$ that does not generate an alarm when no outlier occurs.
- As drift must be one half of the expected change its default value can be adjusted from the actual data as: $v = \frac{threshold * (Q_1 - Q_3)}{2}$, being Q_1 and Q_3 the first and third quartiles respectively, and *threshold* = 3 to detect extreme changes or *threshold* = 1.5 to detect slight ones.
- Following previous ideas, the procedure to calculate default value for h, based on accumulated errors of the filter without reset, is:
    - Calculate $g1$ and $g2$ for all t applying CUSUM RLS filter without reset.
    - Calculate Q_1 and Q_3 for both $g1$ and $g2$ ignoring zeros.
    - Establish limits $h1 = threshold * (Q_3(g1) - Q_1(g1))$ and $h2 = threshold * (Q_3(g2) - Q_1(g2))$. As previosuly, *threshold* = 3 for extreme changes and *threshold* = 1.5 for slight ones.

- Select $h = min(h1, h2)$

- Finally, the number to consecutive changes *MaxChanges* is set to 3 in all cases.

The result is a logical value *TRUE*/*FALSE* that indicates whether an "Alarm" is triggered by the variables, so an anomaly is detected.

2.3. self-X autonomic predictive algorithm selection

Algorithm Selection during Model Training phase of predictive models is crucial to achieve accurate, efficient, interpretable and generalizable results.

With the aim of making predictive algorithm selection more autonomic, an ad-hoc procedure was defined as part of the Asphalt Use Case within the s-X-AIPI project and subsequently adapted for general use (Figure 4). The approach can be summarized as follows:

1. Well-known algorithms (LM, NN, SVM, RF, FRBS approaches) are used to obtain predictive models under different configurations.

2. Several accurate metrics (MSE, RSE, RMSE, MAPE, MAE and R^2) are selected to evaluate the performance of each model.

3. OWA operators are applied successively for decision making to select the most relevant predictive model.

The main theoretical foundations of linear models (LM), neural networks (NN), support vector machines (SVM), random forest (RF) and fuzzy rule-based systems (FRBS) are explained below, as well as how to select the best models based on the performance obtained.

2.3.1. LM: Linear Models

A linear regression model is a statistical technique used to predict the value of a dependent variable (output) based on the value of one or more independent variables (inputs). This model assumes a linear relationship between the variables, represented by the equation $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$, where y is the output, $x_1, x_2, \ldots, x_n$ are the inputs, β_0 is the intercept or constant term, $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients representing the change in y per unit change in each x, and ϵ is the error term that captures deviations not explained by the model.

Linear regression is widely used across various disciplines to model and understand relationships between variables and to make predictions based on observed data.

2.3.2. NN: Neural Networks

A NN is computational model inspired by the structure and function of the human brain[13]. It consists of interconnected nodes, called neurons, organized in layers: an input layer, one or more hidden layers, and an output layer. Each connection between neurons has an associated weight that adjusts as learning proceeds, which helps the network learn from data.

NNs are widely used for tasks such as classification, regression, and pattern recognition, and are particularly powerful for handling complex, non-linear relationships. They are the foundation of deep learning, a subset of machine learning that involves large neural networks with many layers, known as deep neural networks.

2.3.3. SVM: Support Vector Machines

SVM are a set of supervised learning algorithms developed by Vladimir Vapnik and his team at AT&T Bell Labs[14], that are commonly used for classification and regression tasks. Its objective is to find the optimal hyperplane that best separates

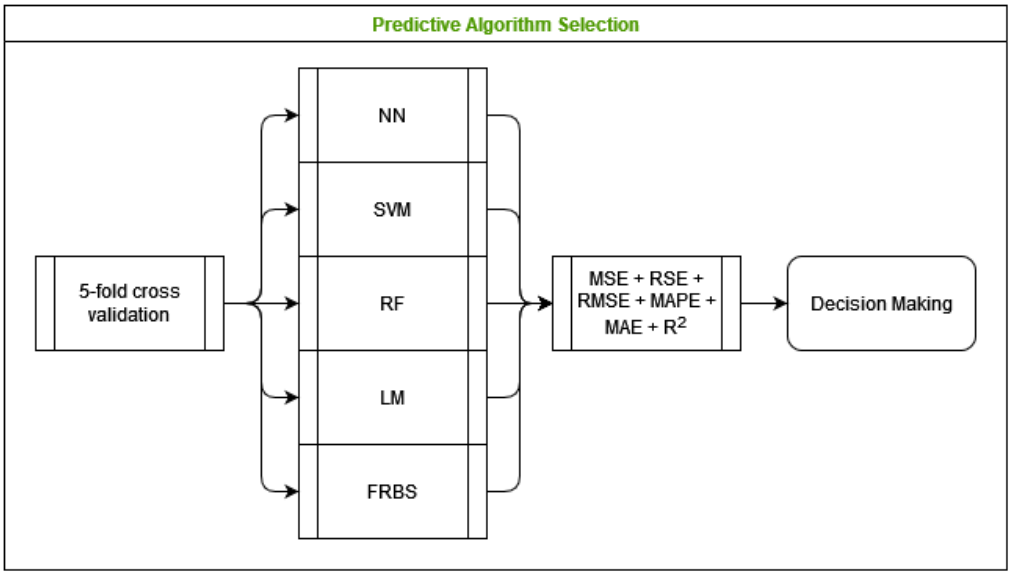


Figure 4. Algorithm selection: general methodology.

data into different classes in a multidimensional space. The goal is to maximize the margin between the closest points of the classes, known as support vectors. SVMs can handle linear and non-linear data by using appropriate kernel functions to transform the input features into higher dimensions, making it easier to find a separating hyperplane.

SVMs are known for their effectiveness in high-dimensional spaces and their ability to be versatile with different kernel functions, making them powerful tools for various pattern recognition and classification problems.

2.3.4. RF: Random Forest

RF is an ensemble learning method used for classification, regression, and other tasks that operates by constructing a multitude of decision trees during training[15]. For classification tasks, the output of the RF is the mode of the classes predicted by individual trees; for regression tasks, it is the mean prediction of the individual trees. This method improves predictive accuracy and controls overfitting by averaging multiple decision trees, which reduces variance. Each tree is trained on a different subset of the data (using a technique called bootstrap sampling) and a random subset of features, ensuring diversity among the trees.

Random Forests are known for their robustness, ability to handle large datasets with higher dimensionality, and effective performance without extensive hyper-parameter tuning.

2.3.5. FRBS: Fuzzy Rule-Based Systems

FRBS are systems that use fuzzy logic to reason about data, so that conditions and conclusions can handle degrees of truth rather than being strictly true or false[16]. Technically these systems can be seen as a Knowledge Base that includes a Rule Base with information described by "if-then" fuzzy rules and a Data Base with the correspondence of the fuzzy values. Besides, a Fuzzification System that converts crisp inputs into fuzzy sets using predefined membership functions, an Inference Engine that uses the fuzzy rules to process the fuzzy input sets and generate fuzzy outputs, and a Defuzzification Interface that converts the fuzzy outputs back into crisp values to provide actionable decisions or predictions[17].

FRBS are able to model complex, uncertain, or imprecise information more naturally, making them well-suited for applications where human reasoning and decision-making are involved.

2.3.6. Evaluation of the accuracy of the models

A system is more precise the closer its behaviour is to the real system it models. The most used measure to evaluate the precision of a regression model is estimation error. Some of the most used formulas are those shown in Table 2[18], where N is the number of examples, Y_i the actual output and Y'_i the estimated output. To improve the accuracy of a model it is necessary to minimize *MSE*, *RSE*, *RMSE*, *MAPE* or *MAE* and to maximize *R*2.

Table 2. Accuracy metrics in regression problems.

Metrics	Formulas
Mean Squared Error	$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - Y'_i)^2$
Relative Squared Error	$RSE = \frac{\sum_{i=1}^{N}(Y_i - Y'_i)^2}{\sum_{i=1}^{N}(Y_i - \overline{Y}_i)^2}, \overline{Y}_i = \frac{1}{N}\sum_{i=1}^{N} Y_i$
Root Mean Squared Error	$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - Y'_i)^2}$
Mean Absolute Percentage Error	$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left\|\frac{Y_i - Y'_i}{Y_i}\right\|$
Mean Absolute Error	$MAE = \frac{1}{N}\sum_{i=1}^{N}\lvert Y_i - Y'_i\rvert$
R Square	$R^2 = 1 - \frac{\sum_{i=1}^{N}(Y_i - Y'_i)^2}{\sum_{i=1}^{N}(Y_i - \overline{Y}_i)}, \overline{Y}_i = \frac{1}{N}\sum_{i=1}^{N} Y_i$

2.4. Decision-making process

In various stages of the self-X methods, a decision-making process must be applied to determine at each moment which is the best option out of several possible ones. Specifically within this package, decision-making is based on OWA operators and RIM quantifiers.

2.4.1. OWA operator

The OWA (*Ordered Weighted Averaging*) operator is an aggregation operator defined by 19 that is used to obtain a single representative value from a set of data according to predetermined optimism/pessimism parameters. That is, it allows partial solutions to be added to obtain a global solution according to a pre-established degree of optimism or pessimism.

Mathematically, an OWA operator of dimension n is a function F: $R^n \rightarrow R$ that has an associated vector of n weights

$$W = (w_1, \ldots, w_n)$$

such that $w_i \in [0,1]$, $1 \leq i \leq n$ and $\sum_{i=1}^{n} w_i = w_1 + \cdots + w_n = 1$, and it is defined to aggregate list of values a_1,..., a_n according to the expresion

$$F_W(a_1, \ldots, a_n) = \sum_{i=1}^{n} (w_i b_i)$$

being b_i being the ith largest value in the set a_1,..., a_n.

A fundamental aspect of OWA operators is the reordering step. An element a_j is not associated with a particular weight w_i, but rather a weight is associated with a particular ordered position i of the elements. Moreover, the final value of the aggregation is different depending on the values assigned to the weight vector W.

Besides, associated with the weight vector, the measures of *orness* (optimism) and *andness* (pessimism) are defined analogously to operators *or* and *and* in classical theory, so that:

$$orness(W) = \frac{1}{n-1} \sum_{i=1}^{n} \left((n-i) * w_i \right), \; orness(W) \in [0,1]$$

$$andness(W) = 1 - orness(W), \; andness(W) \in [0,1]$$

2.4.2. RIM quantifier

In 20 several approaches to obtain the OWA operator weights (W) through fuzzy linguistic RIM (*Regular Increasing Monotone*) quantifiers Q are presented. The quantifier Q determines the construction strategy of the weight vector, allowing the importance of the different aggregation components to be increased or decreased according to the semantics associated with the operator. Moreover, for a given value $x \in [0,1]$, $Q(x)$ is the degree to which x satisfies the fuzzy concept represented by the quantifier Q.

Mathematically, a RIM quantifier is defined as a function Q:$[0,1] \rightarrow [0,1]$ where $Q(0) = 0$, $Q(1) = 1$ and $Q(x) \geq Q(y)$ if $x > y$ and obtaining the weights (W) is done through:

$$w_i = Q\left(\frac{i}{n}\right) - Q\left(\frac{i-1}{n}\right)$$

In 21 different RIM quantifiers have been summarized. Specifically within the self-X methods presented previously, the RIM quantifier family with power function have been considered in decision making processes, being $Q_\alpha(r) = r^\alpha$, $\alpha \in (0,1)$ and $orness(Q_\alpha) = \frac{1}{\alpha + 1}$. For this family of quantifiers, the smaller α the more 'orlike' quantifier[22].

3. self-X methods & implementation

This section presents the main procedures associated to the creation of a self-X AI application, which are the preparation of the data, the feature selection, the detection of anomalies and how to select the appropriate predictive algorithm. The section also includes the instructions about how to install and use the "selfX" R software package.

3.1. Data preparation

3.1.1. Data cleaning

Performing correct data cleaning is essential when carrying out activities related to the extraction of knowledge from data. Specifically, it is necessary to eliminate variables with zero or little variability applied following procedure (Algorithm 1):

1. Find and remove samples (rows) where all values are missing values.
2. Find and remove variables (columns) where all values are missing values.
3. Find and remove NUMERIC, CATEGORICAL or LOGICAL variables (columns) with only one value (without taking into account missing values)
4. A dataset containing clean data (no constant values) for data mining activities is obtained and returned.

Algorithm 1. self-X initial data cleaning

Require: dataset containing numerical, categorical or logical variables for data mining activities
for each sample *s* **do**
  if all values are missing values **then**
    remove row *s*
  end if
end for
for each variable *v* **do**
  if all values are missing values **then**
    remove variable *v*
  else if only one value (without taking into account missing values) **then**
    remove variable *v*
  end if
end for
dataset containing clean data (no constant values) for data mining activities

3.1.2. Statistical test

As a complementary task to the previous one, it is necessary to perform statistical tests on the variables to know whether or not they follow a normal distribution and/or if there are correlated variables that cannot be taken into account in the following activities. Specifically:

- The Shapiro-Wilk test is performed to test the normality of numerical variables, only if there are at least 3 and at most 5000 non-missing samples (Algorithm 2). If the p – obtained is equal to or greater than 0.05 then it is considered that the distribution of the variable is not significantly different from normal distribution and normality can be assumed; otherwise, variable does not follow a normal distribution.

Algorithm 2. self-X statistical test: normality of the variables

Require: dataset containing numerical variables with a minimum of 3 and a maximum of 5000 non-missing samples
for each NUMERICAL variable *v* **do**
  perform Shapiro-Wilk Normality test on *v*
  if *p – value* ≥ 0.05 **then**
    normality can be assumed for variable *v*
  else
    variable *v* does not follow a normal distribution
  end if
end for
list containing "normal" and "non-normal" variables

- The Kendall rank correlation coefficient is computed to measure the correlation between 2 numerical variables, only if there are at least 3 samples in each one. (Algorithm 3). If the Kendall rank correlation coefficient is greater than 0.8 or less than –0.8, and p-value obtained is equal to or greater than 0.05 then a significant correlation between the variables have been detected; otherwise, if the correlation coefficient is greater than 0.8 or less than –0.8 but p-value obtained is less than 0.05 then correlation is non-significant. In any other case, no correlation is detected.

Algorithm 3. self-X statistical test: correlation between variables

Require: dataset containing numerical variables with a minimum of 3 samples
for each NUMERICAL variable *v*1 **do**
  for each NUMERICAL variable *v*2 ! = *v*1 **do**
  obtain Kendall's τ coefficient between *v*1 and *v*2
    if |*correlation*(*v*2,*v*1)| > 0.8 **then**
      *v*1 and *v*2 are correlated variables
      if *p – value* ≥ 0.05 **then**
        significant correlation detected between *v*1 and *v*2
      else
        non-significant correlation between *v*1 and *v*2
      end if
    end if
  end for
end for
list containing SIGNIFICANT correlations between numerical variables

Besides, different graphs can be generated to normality and correlations in a more graphic way.

3.1.3. Outlier and missing values treatment

Additionally, locating and treating outliers and missing values is an essential task when data is pre-processed. An univariate procedure for numerical variables based on ad-hoc expert rules could be (Algorithm 4):

1. Detect outliers within each of the numerical variables:
   a. Outliers will be those values that $outlier < Q_1 - threshold * (Q_3 - Q_1)$ or $outlier > Q_3 + threshold * (Q_3 - Q_1)$ where Q_1 and Q_3 are the first and third quartiles respectively, and *threshold* is set equal to 3 (by default) to detect extreme outliers or to 1.5 to detect slight outliers.
2. Apply treatment rules for outliers and missing values based on expert knowledge.
   a. If a sample only contains outliers or missing values then remove it
   b. If a variable only contains outliers or missing values then remove it
   c. If a variable is constant (without taking into account outliers and missing values) then remove it
   d. If the percentage of outliers and missing values of a variable is greater than 50% then remove the variable
   e. Otherwise, replace the outliers and missing values with the average value of the variable
3. A dataframe containing clean data (no outliers, no missing values, and no constant values) for data mining activities is obtained and returned.

Algorithm 4. self-X outliers and missing values treatment

Require: dataset containing at least one numerical variable
Require: threshold to detect outlier: 3 (by default) to detect extreme outliers or 1.5 to detect slight outliers
for each NUMERICAL variable *v* **do**
  detect outliers such that $outlier < Q_1 - threshold * (Q_3 - Q_1)$ or $outlier > Q_3 + threshold * (Q_3 - Q_1)$

end for
for each sample *s* **do**
  if all values are outliers or missing values **then**
    remove row *s*
  end if
end for
for each NUMERICAL variable *v* **do**
  if all values are outliers or missing values **then**
    remove variable *v*
  else if only one value (without taking into account outliers and missing values) **then**
    remove variable *v*
  else if *Percentage Of Outliers* > 50% **then**
    remove variable *v*
  else
    outliers = *mean*(*v*)
  end if
end for
dataset containing clean data (no outliers, no constant values) for data mining activities

3.2. Feature selection

The methodology used in the "Feature selection" activities is presented below. In all the cases the selection of the features is based on weights obtained as explain in section self-X autonomic unsupervised and supervised feature selection.

3.2.1. Unsupervised feature selection

The unsupervised feature selection process applied to the data is as follows (Algorithm 5):

1. The data is normalized.
2. Different unsupervised feature selection techniques are applied to the data to establish different importance values of the variables considered. Specifically 4 methods:
   a. Evaluation with PCA. 3 variants of the method are considered:
      i. Selection of the first principal component.
      ii. Selection of all those principal components that are above the mean eigenvalues.
      iii. Selection of all principal components that contain 90% of the explained variance.
   b. In all cases, the importance value of the variable is assigned based on the sum of the loadings of the selected principal components, normalized by the total sum of all loadings.
   c. Evaluation with FA. 1 variant of the method is considered:
      i. FA on all factors. To determine the number of factors, Horn's Parallel Analysis is applied. A Maximum Neighborhood factorization and Varimax rotation are used.

   In all cases, the importance value of the variable is assigned based on the *communality* value of the variable, normalized by the sum of all *communalities*.
3. Based on the evaluation obtained, a decision-making process is carried out based on:
   a. OWA operator: applying the RIM quantifier $Q_\alpha(r) = r^\alpha$, $r \geq 0$ with *orness* = 0.4 on the evaluations, an importance is established for each feature, selecting in each case only the most important ones.
4. A subset of features is selected.

Algorithm 5. self-X autonomic unsupervised feature selection

Require: dataset containing clean data (no outliers, no missing values, no constant values) for data mining activities. ONLY numerical input variables allowed (that it can be normalized or not)
Z-score standardization
for FeatureSelectionAlgorithm =PCA:FA **do**
  if PCA **then**
    l_{ij} =Perform PCA over covariances matrix
    for each variable *v* **do**
      $weight_1(v)$: Only first PC is considered
      $weight_2(v)$: PCs whose *eigenvalue* is over *mean*(*eigenvalue*) are considered
      $weight_3(v)$: PCs whose $\sum(ExplainedVariance(P_j)) \geq 90\%$ are considered
    end for
  else if FA **then**
    c_{ij} Apply FA(number of factors according to Horn's Parallel Analysis, "maximum
    likelihood" estimation and "varimax" rotation)
    for each variable *v* **do**
      $weight_4(v)$: All factors are considered
    end for
  end if
  Set *rank*(*features*) based on standarized *weight*
end for
apply OWA on *rank*(*features*)
select variables with highest OWA(*rank*(*features*))

3.2.2. Supervised feature selection

The supervised feature selection process applied to the data is as follows (Algorithm 6):

1. The available data sets are randomly divided into 5 training and test datasets to have a 5-fold cross-validation process.

2. Different supervised feature selection techniques are applied to the data to establish different importance values of the variables considered. First an evaluator filter method based on RELIEF is used and then 6 different direct search algorithms (cutt-off method) are applied using a 5-fold cross validation approach:

   a. Select K best

   b. Select percentile

   c. Select threshold

   d. Select threshold range

   e. Select difference

   f. Select slope

   In all cases, the importance value of the variable is assigned based on the number of times it has been selected (Section Evaluation of the importance of variables).

3. Based on the evaluation obtained, a multi-criteria decision-making process is carried out based on:

   a. OWA operator: applying the RIM quantifier $Q_\alpha(r) = r^\alpha$, $r \geq 0$, with *orness* = 0.4 on the evaluations, an importance is established for each feature, selecting in each case only the most important ones.

4. A subset of features is selected.

Algorithm 6. self-X autonomic unsupervised feature selection

Require: dataset containing clean data (no outliers, no missing values, no constant values) for data mining activities.
ONLY numerical input variables allowed (that it can be normalized or not)
Require: vector with the names of the input variables
Require: vector with the name of the output variable. Only one output variable is allowed
generate randomly 5-fold cross-validation
for FeatureSelectionAlgorithm =RELIEF **do**
  if RELIEF **then**
    for *RELIEFoption_i* = K.best:percentile:threshold:threshold.range:difference:slope **do**
      for CrossValidation CV_1:CV_5 **do**
        perform RELIEF($RELIEFoption_i$,CV_j)
        select $BestFeatures_{ij}$
      end for
      calculate $RELIEFselection_i = \sum_j BestFeatures_{ij}$
    end for
    set *rank*(*features*) based on *RELIEFselection*
  end if
end for
apply OWA on *rank*(*features*)
select input variables with highest OWA(*rank*(*features*))

3.3. Unsupervised anomaly detection

Based on the approaches presented in Section self-X autonomic unsupervised anomaly detection, following procedure to detect anomalies in an unsupervised way is defined.

Specifically, the unsupervised anomaly detection process applied to the data is as follows (Algorithm 7):

1. Different unsupervised anomaly detection methods are applied to the data to establish when an anomaly occurs. Specifically, 2 methods (Section Detection of anomalies):

   a. Evaluation with Outliers Analysis. In this case, the values that all the numerical variables take at the same time are analysed to determine if any of them is an outlier. If yes, the sample is labelled as *TRUE*, otherwise *FALSE*.

   b. Evaluation with CUSUM RLS filter. As previously, the values of all the numerical variables are analysed at the same time to determine whether any of them present or not an abrupt change. Logical values *TRUE* or *FALSE* are then assigned to the sample depending on whether or not an abrupt change is detected.

2. Based on the evaluation obtained, a multi-criteria decision-making process is carried out based on:

   a. OWA operator: applying the RIM quantifier $Q_\alpha(r) = r^\alpha$, $r \geq 0$ with *orness* = 0.9 on the evaluations, the sample is validated or not as an anomaly.

3. Samples are validated or not as anomalies

Algorithm 7. self-X autonomic unsupervised anomalies detection

Require: dataset containing time-ordered data numerical variables
Require: threshold to detect anomalies: 3 to detect extreme anomalies or 1.5 (by default) to detect slight anomalies
for AnomalyDetectionMethod = OutlierAnalysis:CUSUMRLSfilter **do**
  if OutlierAnalysis **then**
    for each NUMERICAL variable *v* **do**
      calculate Q_3 and Q_1
      for each value sample *vs* **do**
        if vs_i is an outlier **then**
          $OutlierAnomaly\ (vs_i) = TRUE$
        end if
      end for
      $OutlierAnomaly\ (vs) = any(OutlierAnomaly\ (vs_i))$
    end for
  end if
  if CUSUMRLSfilter **then**
    for each NUMERICAL variable *v* **do**
      for each value sample *vs* **do**

if vs_i represent an abrupt change **then**
AbruptChangeAnomaly (vs_i) = TRUE
end if
end for
AbruptChangeAnomaly (*vs*) = *any*(*AbruptChangeAnomaly* (vs_i))
end for
end if
apply OWA on *rank*(*OutlierAnomaly* (*vs*),*AbruptChangeAnomaly* (*vs*)) to validate the anomaly
end for
dataset containing analysed data and a final column per sample indicating whether (*TRUE*) or not (*FALSE*) an anomaly has been detected

3.4. Predictive algorithm selection

Finally, following principles presented in Section self-X autonomic predictive algorithm selection, a procedure has been defined to select the predictive algorithm that best fits the data.

In that case the predictive algorithm selection process applied to the data is as follows (Algorithm 8):

1. Each set of input variables is considered separately, so modelling is performed first with the first set, then with the second, and so on. In this way it will be used, for example:
   a. A set with the names of variables selected in the supervised or unsupervised feature selection process,
   b. Other set with the names of the variables selected by a domain expert,
   c. Another set with the names of all possible input variables,
   d. Any other set of variables.
2. Different normalization options are applied:
   a. min-max normalization: transforms a variable defined in the interval [*min*, *max*] to the normalized interval [*newmin*, *newmax*]. In this aproach: *newmin* = 0 and *newmax* = 1:
   $$v_{new} = \frac{v - min}{max - min}(newmax - newmin) + newmin = \frac{v - min}{max - min}$$
   b. Z-score normalization: transforms one set of data into another, with mean zero and standard deviation one.
   $$v_{new} = \frac{v - mean_V}{standev_V}$$
   c. In addition, non-normalized data is also considered.
3. The available data sets are randomly divided into 5 training and test datasets to have a 5-fold cross-validation process.
4. Different predictive algorithms are applied to the data using different options (Table 3). In all cases, the accuracy of the models is evaluated based on different error metrics: MSE, RSE, RMSE, MAPE, MAE, R^2 (Section Evaluation of the accuracy of the models).
5. Based on the evaluation obtained, a multi-criteria decision-making process is carried out based on:
   a. OWA operator: applying the RIM quantifier $Q_\alpha(r) = r^\alpha$, $r \geq 0$ with *orness* = 0.4 on the accuracy of each model to select the most relevant (most accuracy, less test error) model, normalization, input dataset and fold-cross validation. The steps defined in this selection process are:
      i. Select the most relevant model (most accurate) after ranking them according to mean errors. Remove those models whose accuracy is worse than the average and random values
      ii. Select the most relevant normalization (most accurate) of the previously selected model based on the scores given according to the mean errors.
      iii. Select the most relevant input dataset (most accurate) of the previously selected model and normalization, based again on the scores given according to the mean errors.
      iv. Select one of the models generated in the cross-validation strategy is selected based on error.
6. The most relevant predictive algorithm is selected.

Algorithm 8. self-X autonomic predictive algorithm selection

Require: dataset containing clean data (no outliers, no missing values, no constant values) for data mining activities.
ONLY numerical input variables allowed (that it can be normalized or not)
Require: list with different sets of inputs for modelling
Require: vector with the name of the output variable. Only one output variable is allowed
Require: list with execution options
generate randomly 5-fold cross-validation
for each sample *s* **do**
if all values are missing values **then**
remove row *s*
end if
end for
for *FeatureSelection* **do**
for PredictiveAlgorithm=LM:NN:SVM:RF:FRB **do**
for *NormalizationOption* = Z-score:min-max:non-normalized **do**
normalize the data
for CrossValidation CV_1:CV_5 **do**
for *ModelingOption* **do**
train PredictiveAlgorithm(*FeatureSelection*, CV_i, *NormalizationOption*, *ModelingOption*)
calculate $AccuracyMetrics_{tra}$ and $AccuracyMetrics_{tst}$
end for
end for
end for
end for

calculate *AccuracyMetrics* = *mean*(*AccuracyMetrics*$_{tst}$)

end for

set *rank*(*ModelingOption*) based on *AccuracyMetrics*

apply OWA on *rank*(*ModelingOption*)

select *PredictiveAlgorithm* with best OWA(*rank*(*ModelingOption*)))

set *rank*(*NormalizationOption*) based on *AccuracyMetrics* of the selected model

apply OWA on *rank*(*NormalizationOption*)

select *NormalizationOption* based on OWA(*rank*(*Normalization Option*))

set *rank*(*FeatureSelection*) based on *AccuracyMetrics* of the selected model and normalization option

apply OWA on *rank*(*FeatureSelection*)

select *FeatureSelection* based on OWA(*rank*(*FeatureSelection*))

set *rank*(*CrossValidation*) based on *AccuracyMetrics* of the selected model and normalization option and feature selection

apply OWA on *rank*(*CrossValidation*)

select *CrossValidation* based on OWA(*rank*(*CrossValidation*))

select most relevant predictive model

Table 3. Predictive model generation parameters.

PredictiveAlgorithm	ModelingOption	Usage
Linear Models (LM)	generalized	stats::glm(*y* ~ *x*, data)
	non-generalized	stats::lm(*y* ~ *x*, data)
Neural Networks (NN)[2]	backpropagation	neuralnet::neuralnet(*y* ~ *x*, data)
	single-hidden-layer, default options	nnet::nnet(x, y, size=10)
	single-hidden-layer, tuning	nnet::nnet(x, y, size=30, linout = TRUE, maxit = 5000)
	multilayer perceptron for regression	RSNNS::mlp(x, y, size=5, learnFuncParams=c(0.1), maxit=350, inputsTest, targetsTest)
Support Vector Machines (SVM)[3]	default options	e1071::svm(*y* ~ *x*, data)
	tuning	e1071::svm(*y* ~ *x*, data, cost=10, gamma=0.01)
Random Forest (RF)[4]	default options	mRF <- randomForest::randomForest (x, y)
		n.ar < – which.min(mRF$mse)
		randomForest::randomForest (x, y, ntree=n.ar)
	tuning	mRF < - randomForest::randomForest (x, y, importance=TRUE, proximity= TRUE)
		n.ar < - which.min(mRF$mse)
		randomForest::randomForest (x, y, ntree=n.ar, importance=TRUE, proximity= TRUE)
	for regression	randomForest::randomForest (*y* ~ *x*, data,importance= TRUE, proximity=TRUE)
Fuzzy Rule-Based Systems (FRBS)[5]	Wang and Mendel	frbs::frbs.learn(data.train, range.data, method.type = "WM", num.labels = 5, type.mf = "TRIANGLE", type.tnorm = "MIN", type.snorm = "MAX", type.defuz = "WAM", type.implication.func = "ZADEH")
	ANFIS	frbs::frbs.learn(data.train, range.data, method.type = "ANFIS", num.labels = 5, max.iter = 10, step.size = 0.01, type.tnorm = "MIN", type.snorm = "MAX", type. implication.func = "ZADEH")
	DENFIS	frbs::frbs.learn(data.train, range.data, method.type = "DENFIS", Dthr = 0.1, max.iter = 10, step.size = 0.01, d = 2,name = "Sim-0")

[2] https://cran.r-project.org/package=neuralnet, https://cran.r-project.org/package=nnet, https://cran.r-project.org/package=RSNNS

[3] https://cran.r-project.org/package=e1071[24]

[4] https://cran.r-project.org/package=randomForest

[5] https://cran.r-project.org/package=frbs[25]

3.5. Implementation: How to use

This section shows how to use the functionalities included in the "selfX" package and explained before. In all the cases the codes to use the procedure is showed as well as the outputs obtained by console. Moreover, `mtcars`[6] is used as dataset to demonstrate the functionality of the R package. This dataset is a built-in dataset in R. Its data is related to automobiles design and performance, and contains 11 features for 32 samples.

Data Preparation: Data Cleaning

```
R > library(selfX)

# load dataset
R > data(mtcars)

# run initial.cleandata method
R > cleanDF <- initial.cleandata(mtcars)

R > View(cleanDF)      # see Figure 5 for details
```

Data Preparation: Statistical Test

```
R > library(selfX)

# load dataset
R > data(mtcars)

# run test.normality method
R > NormalTest <- test.normality(mtcars)

R > View(NormalTest)   # see Figure 6 for details

# run test.correlation method
R > CorrelationTest <- test.correlation(mtcars)

R > View(CorrelationTest)   # see Figure 7 for details
```

Data Preparation: Outlier and Missing Values Treatment

```
R > library(selfX)

# load dataset
R > data(mtcars)

# run outliers.treatment method
R > cleanDF.NoOutliers.NoMissing <- outliers.missing.treatment(mtcars)

R > View(cleanDF.NoOutliers.NoMissing)   # see Figure 8 for details
```

Unsupervised Feature Selection

```
R > library(selfX)

# load dataset
R > data(mtcars)

# run pca.fa.owa method
R > FS_unsupervised <- pca.fa.owa(mtcars)

R > View(FS_unsupervised)   # see Figure 9 for details
```

Supervised Feature Selection

```
R > library(selfX)

R > library(foreach)


# load dataset
R > data(mtcars)

# establish inputs and output
R > varsINPUT <- c("cyl","disp","hp","drat","wt","qsec","vs","am","gear","carb")
R > varsOUTPUT <- c("mpg")

# run relief.owa method
R > FS_supervised <- relief.owa(mtcars, varsINPUT,varsOUTPUT)

R > View(FS_supervised)   # see Figure 10 for details
```

Unsupervised Anomaly Detection

```
R > library(selfX)

# load dataset
R > data(mtcars)

# establish inputs and output
R > varsINPUT <- c("mpg","disp","hp","drat","wt","gear")

# run unsupervised.anomalies.detection.owa method
R > anomaliesDF <- unsupervised.anomalies.detection.owa(mtcars, varsINPUT)

R > View(anomaliesDF)   # see Figure 11 for details
```

3.6. Operation: How to install

This section include the instructions to install "selfX" package in RStudio, an integrated development environment (IDE) for R programming language. The only requirement to be able to use this package is to have the R installed in version 4.0 or higher. Then, to start working with the packages in R, it must be installed as indicated below. Remember that, after installation, the package must be loaded in order to be able to use the functions.

[6] https://www.rdocumentation.org/packages/datasets/versions/3.6.2/topics/mtcars

cleanDF

Filter

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4

Showing 1 to 24 of 32 entries, 11 total columns

Figure 5. **Data Preparation: Data Cleaning Output.**

NormalTest

Show Attributes

Name	Type	Value
NormalTest	list [2]	List of length 2
NormalVariables	character [4]	'mpg' 'drat' 'wt' 'qsec'
NoNormalVariables	character [7]	'cyl' 'disp' 'hp' 'vs' 'am' 'gear' ...

Figure 6. **Data Preparation: Normality Statistical Test Output.**

CorrelationTest

Filter

	correlations	pvalue	rvalue
1	cyl VS disp	9.956527e-09	0.8144263
2	disp VS cyl	9.956527e-09	0.8144263

Figure 7. **Data Preparation: Correlation Statistical Test Output.**

cleanDF.NoOutliers.NoMissing

Filter

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4

Showing 1 to 24 of 32 entries, 11 total columns

Figure 8. **Data preparation: Outlier and missing values treatment output.**

FS_unsupervised

Show Attributes

Name	Type	Value
FS_unsupervised	list [1]	List of length 1
varsINPUTselected	character [3]	'cyl' 'carb' 'qsec'

Figure 9. **Unsupervised feature selection output.**

FS_supervised

Show Attributes

Name	Type	Value
FS_supervised	list [1]	List of length 1
varsINPUTselected	character [3]	'disp' 'wt' 'hp'

Figure 10. **Supervised feature selection output.**

Predictive Algorithm Selection

```
R > library(selfX)
R > library(foreach)

# load dataset and define inputs/output
R > data(mtcars)
R > varsINPUTopc1 <- c("cyl","disp","wt")
R > varsINPUTopc2 <- c("cyl","disp","hp","drat","wt","qsec","vs","am","gear","carb")
R > varsINPUTlist <- list(varsINPUTopc1,varsINPUTopc2)
R > varsOUTPUT <- c("mpg")
```

```
# run modeling.owa method in selfX way (default options)
R > SelectedModel_1 <- modeling.owa(mtcars, varsINPUTlist, varsOUTPUT)

... self-X modelling ... cyl disp wt
... the execution may take a few minutes to complete depending on the number of data ...
... self-X modelling ... cyl disp hp drat wt qsec vs am gear carb
... the execution may take a few minutes to complete depending on the number of data ...
FINAL SELECTION
Self-X feature selection: cyl disp hp drat wt qsec vs am gear carb
Self-X selected algorithm: Random Forest
Self-X selected normalization: Z-Score
Self-X selected FCV: 3

... completed successfully ...
```

```
R > View(SelectedModel_1)  # see Figure 12 for details
```

```
# run modeling.owa method in selfX way, only SVM and Randon Forest
R > runOPC <- list(runLM = FALSE, runNN = FALSE, runSVM = TRUE, runRF = TRUE, runFRBS = FALSE)
R > SelectedModel_2 <- modeling.owa(mtcars, varsINPUTlist, varsOUTPUT, runOPC)

... self-X modelling ... cyl disp wt
... the execution may take a few minutes to complete depending on the number of data ...
... self-X modelling ... cyl disp hp drat wt qsec vs am gear carb
... the execution may take a few minutes to complete depending on the number of data ...
FINAL SELECTION
Self-X feature selection: cyl disp wt
Self-X selected algorithm: Random Forest
Self-X selected normalization: min-max (between 0-1)
Self-X selected FCV: 1
... completed successfully ...
```

```
R > View(SelectedModel_2)   # see Figure 13 for details
```

```
# run modeling.owa method in debugger mode, only NN and FRBS
R > runOPC <- list(runLM = FALSE, runNN = TRUE, runSVM = FALSE, runRF = FALSE, runFRBS = TRUE)
R > SelectedModel_3 <- modeling.owa(mtcars,varsINPUTlist,varsOUTPUT,runOPC,selfX = FALSE)

... 5 FCV creating ...
... 5 FCV created ...

... self-X modelling ... cyl disp wt
... the execution may take a few minutes to complete depending on the number of data ...
... running OWA ...

       DEBUGGING MODE
... Self-X selected algorithm (options): NNet4_N_TST (NNet4_N_TST)
... accurate metrics (MSE_nor1, RMSE_nor1, RSE_nor1, MAPE_nor1, MAE_nor1, R2_nor1, MSE_nor2, RMSE_nor2, RSE_nor2,
MAPE_nor2, MAE_nor2, R2_nor2): 7.29, 2.65, 0.25, 11.5, 2.21, -Inf, 7.5, 2.69, 0.26, 11.7, 2.18, -Inf
... running OWA ...
... Self-X selected normalization (options): Z-Score (Z-Score)

... self-X modelling ... cyl disp hp drat wt qsec vs am gear carb
... the execution may take a few minutes to complete depending on the number of data ...
... running OWA ...

       DEBUGGING MODE
... Self-X selected algorithm (options): NNet4_N_TST (NNet4_N_TST)
... accurate metrics (MSE_nor1, RMSE_nor1, RSE_nor1, MAPE_nor1, MAE_nor1, R2_nor1, MSE_nor2, RMSE_nor2, RSE_nor2,
MAPE_nor2, MAE_nor2, R2_nor2): 16.43, 3.78, 0.6, 16.48, 3.13, -Inf, 16.77, 3.66, 0.61, 15.84, 3.04, -Inf
... running OWA ...
... Self-X selected normalization (options): min-max (between 0-1) (min-max (between 0-1))
... running OWA ...
... Self-X selected features (options): SC1_nnet2_default_N_TST (SC1_nnet2_default_N_TST)
... running OWA ...
... Self-X selected fold cross-validation (options): FCV3 (FCV3)
```

```
FINAL SELECTION
Self-X feature selection: cyl disp wt
Self-X selected algorithm: Neural Networks
Self-X selected normalization: Z-Score
Self-X selected FCV: 3

... completed successfully ...
```

```
R > View(SelectedModel_3)   # see Figure 14 for details
```

- Open R studio.

```
R version 4.3.2 (2023-10-31 ucrt) -- "Eye Holes"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

[Workspace loaded from ~/.RData]
R >
```

- Use install.packages function to install "devtools" package from CRAN

```
R> install.packages(devtools)
Loading required package: foreach

Installing package into 'C:/../AppData/Local/R/win-library/4.3' (as 'lib' is unspecified)
also installing the dependencies 'credentials', 'gitcreds', ....., 'rversions', 'urlchecker'

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/credentials_2.0.1.zip'
Content type 'application/zip' length 221665 bytes (216 KB)
downloaded 216 KB
```

```
probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/gitcreds_0.1.2.zip'
Content type 'application/zip' length 99460 bytes (97 KB)
downloaded 97 KB

.....

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/urlchecker_1.0.1.zip'
Content type 'application/zip' length 36322 bytes (35 KB)
downloaded 35 KB

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/devtools_2.4.5.zip'
Content type 'application/zip' length 435953 bytes (425 KB)
downloaded 425 KB

package 'credentials' successfully unpacked and MD5 sums checked
package 'gitcreds' successfully unpacked and MD5 sums checked
.....
package 'urlchecker' successfully unpacked and MD5 sums checked
package 'devtools' successfully unpacked and MD5 sums checked

The downloaded binary packages are in

    C:/../AppData/Local/Temp/.../downloaded_packages
R >
```

- Download the file *selfX_1.0.0.tar.gz* with the source code of the package.

- Use install_local function to install "selfX" R package from the file (*.tar.gz*) downloaded in the previous step.Remember to correctly specify the local path of the file.

anomaliesDF

Filter

	mpg	disp	hp	drat	wt	gear	outliers
Mazda RX4	21.0	160.0	110	3.90	2.620	4	FALSE
Mazda RX4 Wag	21.0	160.0	110	3.90	2.875	4	FALSE
Datsun 710	22.8	108.0	93	3.85	2.320	4	FALSE
Hornet 4 Drive	21.4	258.0	110	3.08	3.215	3	FALSE
Hornet Sportabout	18.7	360.0	175	3.15	3.440	3	FALSE
Valiant	18.1	225.0	105	2.76	3.460	3	FALSE
Duster 360	14.3	360.0	245	3.21	3.570	3	FALSE
Merc 240D	24.4	146.7	62	3.69	3.190	4	FALSE
Merc 230	22.8	140.8	95	3.92	3.150	4	FALSE
Merc 280	19.2	167.6	123	3.92	3.440	4	FALSE
Merc 280C	17.8	167.6	123	3.92	3.440	4	FALSE
Merc 450SE	16.4	275.8	180	3.07	4.070	3	FALSE
Merc 450SL	17.3	275.8	180	3.07	3.730	3	FALSE
Merc 450SLC	15.2	275.8	180	3.07	3.780	3	FALSE
Cadillac Fleetwood	10.4	472.0	205	2.93	5.250	3	TRUE
Lincoln Continental	10.4	460.0	215	3.00	5.424	3	TRUE
Chrysler Imperial	14.7	440.0	230	3.23	5.345	3	FALSE
Fiat 128	32.4	78.7	66	4.08	2.200	4	FALSE
Honda Civic	30.4	75.7	52	4.93	1.615	4	FALSE
Toyota Corolla	33.9	71.1	65	4.22	1.835	4	FALSE
Toyota Corona	21.5	120.1	97	3.70	2.465	3	FALSE
Dodge Challenger	15.5	318.0	150	2.76	3.520	3	FALSE
AMC Javelin	15.2	304.0	150	3.15	3.435	3	FALSE
Camaro Z28	13.3	350.0	245	3.73	3.840	3	FALSE

Figure 11. Unsupervised anomaly detection output.

SelectedModel1	list [2]	List of length 2
bestMODEL	list [18] (S3: randomForest.formu	List of length 18
call	language	randomForest(formula = ecuacion, data = trn_dataPRUEBA_NN, importance = TRUE, ...
type	character [1]	'regression'
predicted	double [26]	23.8 19.7 25.5 12.3 13.5 20.4 ...
mse	double [500]	26.4 15.5 15.1 16.0 11.0 11.4 ...
rsq	double [500]	0.232 0.550 0.562 0.533 0.681 0.669 ...
oob.times	integer [26]	182 173 169 192 188 183 ...
importance	double [10 x 2]	6.411 9.268 6.749 1.872 11.094 0.154 120.692 181.422 165.158 58.727 ...
importanceSD	double [10]	0.629 0.722 0.579 0.361 0.837 0.190 ...
localImportance	NULL	Pairlist of length 0
proximity	double [26 x 26]	1.0000 0.0635 0.2679 0.0000 0.0000 0.1935 0.0635 1.0000 0.0909 0.0563 0.0156 0.3 ...
ntree	double [1]	500
mtry	double [1]	3
forest	list [11]	List of length 11
coefs	NULL	Pairlist of length 0
y	double [26]	21.5 17.8 30.4 14.7 10.4 19.7 ...
test	NULL	Pairlist of length 0
inbag	NULL	Pairlist of length 0
terms	formula	mpg ~ cyl + disp + hp + drat + wt + qsec + vs + am + gear + carb
accuracy	list [1 x 6] (S3: data.frame)	A data.frame with 1 row and 6 columns
MSE	double [1]	3.940097
RMSE	double [1]	1.984968
RSE	double [1]	0.1017062
MAPE	double [1]	7.234681
MAE	double [1]	1.568714
R2	double [1]	6.654232e+15

Figure 12. Predictive algorithm selection output (1/3).

SelectedModel2	list [2]	List of length 2
bestMODEL	list [18] (S3: randomForest.formu	List of length 18
call	language	randomForest(formula = ecuacion, data = trn_dataPRUEBA_NN, importance = TRUE, ...
type	character [1]	'regression'
predicted	double [25]	28.1 16.1 19.2 29.2 18.7 15.8 ...
mse	double [500]	8.43 11.84 10.27 9.70 9.25 9.22 ...
rsq	double [500]	0.743 0.639 0.687 0.704 0.718 0.719 ...
oob.times	integer [25]	193 183 198 191 184 190 ...
importance	double [3 x 2]	10.2 14.3 14.1 192.4 265.5 256.3
importanceSD	double [3]	0.680 0.999 0.965
localImportance	NULL	Pairlist of length 0
proximity	double [25 x 25]	1.0000 0.0000 0.0000 0.8806 0.0000 0.0000 0.0000 1.0000 0.0278 0.0000 0.0339 0.4 ...
ntree	double [1]	500
mtry	double [1]	1
forest	list [11]	List of length 11
coefs	NULL	Pairlist of length 0
y	double [25]	32.4 10.4 17.8 33.9 19.2 15.0 ...
test	NULL	Pairlist of length 0
inbag	NULL	Pairlist of length 0
terms	formula	mpg ~ cyl + disp + wt
accuracy	list [1 x 6] (S3: data.frame)	A data.frame with 1 row and 6 columns
MSE	double [1]	3.377409
RMSE	double [1]	1.837773
RSE	double [1]	0.07804728
MAPE	double [1]	10.54508
MAE	double [1]	1.54741
R2	double [1]	1.330918e+16

Figure 13. **Predictive algorithm selection output (2/3).**

SelectedModel3	list [4]	List of length 4
SC1	list [2]	List of length 2
allMODELS	list [5]	List of length 5
DF_MEANaccuracy	list [72 x 12] (S3: data.frame)	A data.frame with 72 rows and 12 columns
SC2	list [2]	List of length 2
allMODELS	list [5]	List of length 5
DF_MEANaccuracy	list [72 x 12] (S3: data.frame)	A data.frame with 72 rows and 12 columns
bestMODEL	list [14] (S3: nn)	List of length 14
call	language	neuralnet::neuralnet(formula = ecuacion, data = trn_dataPRUEBA)
response	double [26 x 1]	0.450 -0.463 -0.712 -0.811 0.450 1.711 ...
covariate	double [26 x 3]	-1.2249 1.0149 1.0149 1.0149 -1.2249 -1.2249 -0.9902 0.3637 0.9705 0.3637 ...
model.list	list [2]	List of length 2
err.fct	function	function(x, y) { ... }
act.fct	function	function(x) { ... }
linear.output	logical [1]	TRUE
data	list [26 x 4] (S3: data.frame)	A data.frame with 26 rows and 4 columns
exclude	NULL	Pairlist of length 0
net.result	list [1]	List of length 1
weights	list [1]	List of length 1
generalized.weights	list [1]	List of length 1
startweights	list [1]	List of length 1
result.matrix	double [9 x 1]	1.91e+00 9.97e-03 5.36e+03 1.35e+00 1.29e-01 1.22e-01 ...
accuracy	list [1 x 6] (S3: data.frame)	A data.frame with 1 row and 6 columns
MSE	double [1]	12.67226
RMSE	double [1]	3.559812
RSE	double [1]	0.4978788
MAPE	double [1]	15.30158
MAE	double [1]	3.110447
R2	double [1]	2.140154e+16

Figure 14. **Predictive algorithm selection output (3/3).**

```
R >; devtools::install_local("selfX_1.0.0.tar.gz")
These packages have more recent versions available.
It is recommended to update all of them.
Which would you like to update?
1: All
2: CRAN packages only
3: None
4: parallelly (1.36.0 -> 1.37.0 ) [CRAN]
5: RcppArmad... (0.12.6.6.1 -> 0.12.8.0.0) [CRAN]
6: recipes (1.0.9 -> 1.0.10 ) [CRAN]
Enter one or more numbers, or an empty line to skip updates:

— R CMD build —
✓ checking for file 'C:/../AppData/Local/Temp/.../selfX/DESCRIPTION' (543ms)
– preparing 'selfX':
✓ checking DESCRIPTION meta-information
– checking for LF line-endings in source and make files and shell scripts
– checking for empty or unneeded directories
– building 'selfX_1.0.0.tar.gz'

Installing package into 'C:/../AppData/Local/R/win-library/4.3' (as 'lib' is unspecified)

* installing *source* package 'selfX' ...
** using staged installation
** R
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (selfX)

R >
```

- Load the package

```
R > library(selfX)
Loading required package: foreach

R >
```

4. Conclusions

A detailed description of all the functionalities included in the "selfX" R package has been included in this contribution.

At the moment, the version 1.0.0 of this package includes the implementation of several self-X methods for dealing with (1) supervised and unsupervised feature selection, (2) unsupervised anomaly detection, (3) predictive algorithm selection, and (4) cleaning data, statistical tests or outliers treatment activities in a more autonomic way.

Subsequent versions of the package will include, among others, the following improvements:

- allow the end user to provide their own data sets to apply the 5-fold cross-validation technique.
- increase the types of statistical tests, including normality tests for more than 5000 samples and/or other correlation tests.
- define additional "self-X autonomic clustering algorithm selection" and "self-X autonomic normalization methods" procedures and include them in the package

This package has been developed within the s-x-AIPI project, which has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement № 101058715 (https://s-x-aipi-project.eu/). Moreover, this R package is implemented as open-source software and offered to the R community for use and improvement through the AI on-demand (AIoD) platform[7] (Table 4).

Table 4. "self-X" R package: credits.

Link for download	https://www.ai4europe.eu/research/ai-catalog/s-x-aipi-self-x-open-source-toolset-r
Developed by	CARTIF Technology Centre
Contact Details	Industrial and Digital Systems Division
	s-x-aipi@cartif.es
License	GNU General Public License (GPL) v3

Data and software availability

Source data

No source data has been used in the development of this software tool.

[7] https://aiod.eu/

Underlying data
No underlying data has been used in the development of this software tool.

Extended data
No extended data has been used in the development of this software tool.

Software availability
The R package has been included in Zenodo at https://doi.org/10.5281/zenodo.11612977[23]. The resource is open accessible under license GNU General Public License (GPL) v3.

Besides, "selfX" R package can be also downloaded from AI on-Demand (AIoD) platform, through the AI assets catalogue available at https://www.ai4europe.eu/ai-community/projects/s-x-aipi?category=ai_assets. Moreover, the direct access web link to version 1.0.0 of the R package is https://www.ai4europe.eu/research/ai-catalog/s-x-aipi-self-x-open-source-toolset-r.

References

1. Mao KZ: **Identifying critical variables of principal components for unsupervised feature selection.** *IEEE Trans Syst Man Cybern B Cybern.* 2005; **35**(2): 339–344.
PubMed Abstract | Publisher Full Text
2. Malhi A: **PCA-Based feature selection scheme for machine defect classification.** *IEEE Trans Instrum Meas.* 2004; **35**(6): 1517–1525.
Publisher Full Text
3. Krzanowski WJ, Marriott FHC: **Multivariate analysis (2 vols)**. Edward Arnold, 1994.
Reference Source
4. Mahwah NJ: **Factor analysis**. *Journal of consumer psychology`s special issue on methodological statistical concerns of the experimental behavioral researcher*. Iacobucci, Dawn. 2001.
Reference Source
5. Ledesma RD, Valero-Mola P: **Determining the number of factors to retain in EFA: an easy-to-use computer program for carrying out parallel analysis.** *Practical Assessment, Research and Evaluation.* 2007; **12**.
Publisher Full Text
6. Kim JO, Mueller CW: **Introduction to factor analysis - what it is and how to do it**. Beverly Hills. CA: Sage, 1978.
Reference Source
7. Darton RA: **Rotation in factor analysis.** *The Statistician.* 1980; **29**: 167–194.
Publisher Full Text
8. Aragón-Royó F, Jiménez-Vílchez A, Arauzo-Azofra A, *et al.*: **FSinR: an exhaustive package for feature selection.** *arXiv.* 2020.
Publisher Full Text
9. Kira K, Rendell LA: **A practical approach to feature selection**. *de Machine Learning Proceedings 1992*. Elsevier, 1992; 249–256.
Publisher Full Text
10. Tukey JW: **Exploratory data analysis**. Addison-Wesley Pub. Co, 1977; 688.
Reference Source
11. Gustafsson F: **Adaptive filtering and change detection**. Wiley, 2001.
Publisher Full Text
12. Ljung L: **System identification**. Prentice Hall PTR, 1998; 609.
13. Ripley BD: **Pattern recognition and neural networks**. Cambridge University Press, 1996.
Publisher Full Text
14. Vapnik VN: **Statistical learning theory**. Wiley, 1998; 736.
Reference Source
15. Breiman L: **Random forests.** *MACH LEARN.* 2001; **45**: 5–32.
Publisher Full Text
16. Zadeh LA: **Fuzzy sets.** *INFORM CONTROL.* 1965; **8**(3): 338–353.
Publisher Full Text
17. Herrera F: **Genetic fuzzy systems: taxonomy, current research trends and prospects.** *Evolutionary Intelligence.* 2008; **1**: 27–46.
Publisher Full Text
18. Witten IH, Frank E, Hall MA: **Data mining: practical machine learning tools and techniques**. Third ed., Morgan Kaufmann, 2011.
Publisher Full Text
19. Yager RR: **On ordered weighted averaging aggregation operators in multicriteria decisionmaking.** *IEEE Trans Syst Man Cybern.* 1988; **18**(1): 183–190.
Publisher Full Text
20. Yager RR: **Quantifier guided aggregation using OWA operators.** *Int J Intell Syst.* 1996; **11**: 49–73.
Publisher Full Text
21. Liu X, Han S: **Orness and parameterized RIM quantifier aggregation with OWA operators: a summary.** *Int J Approx Reason.* 2008; **48**(1): 77–97.
Publisher Full Text
22. Yager RR: **Families of OWA operators.** *Fuzzy Sets Syst.* 1993; **59**(2): 125–148.
Publisher Full Text
23. Galende M, Corral A, Reñones A: **s-X-AIPI self-X open-source toolset in R.** *Zenodo.* 2024.
http://www.doi.org/10.5281/zenodo.11612977
24. Chang CC, Lin CJ: **LIBSVM: a library for support vector machines.** *ACM Trans Intell Syst Technol.* 2011; **2**(3): 1–27.
Publisher Full Text
25. Riza LS, Bergmeir C, Herrera F: **frbs: fuzzy rule-based systems for classification and regression in R.** *J Stat Softw.* 2015; **65**(6): 1–30.
Publisher Full Text

Open Peer Review

Current Peer Review Status: ✗ ✗

Version 1

Reviewer Report 14 December 2024

https://doi.org/10.21956/openreseurope.19529.r46718

© **2024 Van Lissa C.** This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

✗

Caspar J. Van Lissa
Tilburg University, Tilburg, The Netherlands

This publication describes a package with several functions for data preprocessing and exploratory analysis. The package duplicates functionality that already exists in better developed packages which are properly substantiated by methodological research (e.g., https://www.tidymodels.org/); it falls short of all established standards for R-packages in (https://cran.r-project.org/doc/manuals/R-exts.html) and general good practices for open source software (e.g., https://www.bestpractices.dev/en, Lamprecht A, et al (2020 [Ref-1]) https://content.iospress.com/download/data-science/ds190026?id=data-science/ds190026), and violates basic principles of statistics and machine learning (e.g., Hastie, Tibsirani & Friedman, 2009 [Ref-2]). I will describe each of my concerns in some detail below.

1) Is the rationale for developing the new software tool clearly explained? No - all of the provided functionality already exists in standard R-packages on CRAN (notably the tidymodels suite, and its predecessor, `caret`).

2) Is the description of the software tool technically sound? No - the description is an ad-hoc list of statistical and machine learning techniques, which are presented with a lack of awareness of their underlying assumptions and proper ways to combine them. Here are some examples:

2.1) The authors perform missing data imputation on the whole dataset, preceding ML modeling. This introduces training/test split cross-contamination, which will bias predictive performance metrics in ML analyses (Hastie, Tibsirani & Friedman, 2009 [Ref-2]).

2.2) The authors perform statistical "model assumption" checks, but most of the models they use don't make the assumptions they check. E.g., the authors write "it is necessary to perform statistical tests on the variables to know whether or not they follow a normal distribution", but as far as I can tell, none of the models discussed assume normal predictors. Perhaps the authors are making the common but basic misunderstanding that linear regression's assumption of "normality" (of residuals) applies to the predictor variables - but this is wrong (evident from the formula, but see for example: https://www.columbia.edu/cu/bnet/StatisticalFallacy_July2014.pdf).

Moreover, the whole practice of adjusting analytic decisions based on "assumption checks" is problematic, because it risks overfitting the data. Assumptions are called "assumption" for a reason; we assume some properties of the data-generating mechanism and model them accordingly. The data can give us signals that some assumptions might not be warranted, and we should certainly report and discuss these signals - blindly following statistical assumption checks is not good practice (e.g. Shatz I 2024 [Ref-3]).

2.3) "it is necessary to perform statistical tests on the variables to know [...] if there are correlated variables that cannot be taken into account in the following activities": Again, rote testing for correlations makes no sense. Firstly, correlations assume linear relations, but many of the analyses mentioned do not assume linearity. Secondly, it's not clear what the concern is - multicolinearity is primarily an issue when interpreting the model, not for building predictive models per se. Also, if one of the downstream steps is PCA, then high correlations are not a problem.

2.4) "locating and treating outliers [...] is an essential task": The perspective on handling outliers based on univariate parametric distance measures is naive (e.g., see Zimek A et al 2018 [Ref-4] , Smiti A 2020 [Ref-5]). A definition of outliers is "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism". Thus, to identify outliers, we must first define a model that could have given rise to the data. In ML applications, we rarely have such a model. Secondly, an outlier is only an outlier if that model is correct; it could also be a signal that the model is incorrect. Since ML applications are typically model-agnostic, it's probably better not to exclude outliers but instead include them in error analyses when evaluating model performance.

2.5) This section "3.2.1. Unsupervised feature selection" deals with feature engineering rather than feature selection. Section "3.2.2. Supervised feature selection" deals with standard cross-validated ML analyses, but it's not necessary to perform separate cross-validation for feature selection prior to ML analysis ("3.4. Predictive algorithm selection") because many ML models intrinsically perform feature selection.

3) Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others? No - while the R-code is of course accessible once the package is installed, to be properly FAIR, the code should also be deposited in a standard indexed repository (CRAN and GitHub or similar public Git repository). The methods are mostly ad-hoc (procedures invented by the authors but not supported by methodological research and oftentimes at odds with it; see comment 2).

4) Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool? No - because the methods are ad-hoc, research would be required to validate the proposed algorithms. Since the proposed algorithms violate well-known principles of statistics and machine learning, they would then be found to be suboptimal.

I am sorry that my review is not more encouraging - but I am legitimately concerned about the potential harmful impact of making unsubtantiated methods easily accessible in open source software.

With sincere regards,

Caspar van Lissa, PhD
Tilburg University, dept. Methodology & Statistics
c.j.vanlissa@tilburguniversity.edu

A note on signed reviews

In the spirit of open science, transparency, and accountability, I sign my reviews. In the context of the replication crisis, I consider these principles to be more important than the anonymity of blind review. I have reviewed this paper in good faith, to the best of my ability, and with the sole intention of providing helpful suggestions that might improve the quality of the work. If my comments cause offense, please know that this was unintentional. Reviewing this paper took a substantial amount of work. Even if you choose not to resubmit your paper to this same journal, I ask that you consider addressing my comments. The Creative Commons Attribution-NoDerivatives 4.0 International License gives you the right to share (parts of) my review with attribution, if you so choose. Best of luck!

LICENSE:
This review is licensed under a Creative Commons Attribution-NoDerivatives 4.0 International License. The terms of this license are available at http://creativecommons.org/licenses/by-nd/4.0/ Practically speaking: You can use my review as usual, but the publisher cannot assert copyright over it.

References

1. Lamprecht A, Garcia L, Kuzak M, Martinez C, et al.: Towards FAIR principles for research software. *Data Science*. 2020; **3** (1): 37-59 Publisher Full Text
2. Hastie T, Tibshirani R, Friedman J: The Elements of Statistical Learning. Publisher Full Text
3. Shatz I: Assumption-checking rather than (just) testing: The importance of visualization and effect size in statistical diagnostics.*Behav Res Methods*. 2024; **56** (2): 826-845 PubMed Abstract | Publisher Full Text
4. Zimek A, Filzmoser P: There and back again: Outlier detection between statistical reasoning and data mining algorithms. *WIREs Data Mining and Knowledge Discovery*. 2018; **8** (6). Publisher Full Text
5. Smiti A: A critical overview of outlier detection methods. *Computer Science Review*. 2020; **38**. Publisher Full Text

Is the rationale for developing the new software tool clearly explained?

No

Is the description of the software tool technically sound?

No

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

No

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

No

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Open reproducible science, machine learning, research software engineering

I confirm that I have read this submission and believe that I have an appropriate level of expertise to state that I do not consider it to be of an acceptable scientific standard, for reasons outlined above.

Reviewer Report 26 September 2024

https://doi.org/10.21956/openreseurope.19529.r43335

© 2024 Dudek A. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



Andrzej Dudek
Uniwersytet Ekonomiczny we Wrocławiu, Wrocław, Poland

In current form package and paper is not acceptable. There are many good sources describing the minimal requirements for R packages like https://www.carlboettiger.info/2013/06/13/what-I-look-for-in-software-papers.html
There three minimal criteria to be completed for publishing the paper describing the package:
Package have to published on CRAN
Package should contain automatic tests
Package code should use in passing optimal arguments to called functions.
None of them is fulfilled so I recommend to not accept the paper at this point and ask author(s) to re-submit it after fixing the reported issues

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
No

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
No

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
No

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: clustering, software packages, machine learning

I confirm that I have read this submission and believe that I have an appropriate level of expertise to state that I do not consider it to be of an acceptable scientific standard, for reasons outlined above.