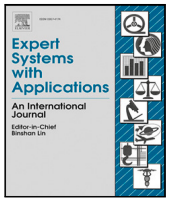
Expert Systems With Applications 237 (2024) 121608

Contents lists available at ScienceDirect

Expert Systems With Applications

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



Tackling the problem of noisy IoT sensor data in smart agriculture: Regression noise filters for enhanced evapotranspiration prediction

Juan Martín [a,*], José A. Sáez [b], Emilio Corchado [a]

[a] Department of Computer Science and Automatics, University of Salamanca, Plaza de los Caídos s/n, 37008 Salamanca, Spain
[b] Department of Statistics and Operations Research, University of Granada, Fuente Nueva s/n, 18071 Granada, Spain

ARTICLE INFO

Dataset link: https://CRAN.R-project.org/package=rgnoisefilt

Keywords:
IoT sensor data
Data quality
Smart agriculture
Evapotranspiration prediction
Noise filtering

ABSTRACT

In smart agriculture, the accurate prediction of evapotranspiration plays a crucial role in optimizing water usage and maximizing crop yield. However, the increasing adoption of IoT sensor technologies has resulted in the accumulation of large amounts of data, which are frequently contaminated by noise and pose a significant challenge to extract reliable knowledge through data modeling. This research addresses the problem of noisy IoT sensor data and its impact on evapotranspiration prediction, an essential aspect of agricultural practices. The effect of noise on sensor variables and evapotranspiration is extensively analyzed by simulating different noise levels in evapotranspiration datasets collected from various agricultural areas in Spain, enabling a comprehensive evaluation of its impact on the performance of data science models. Despite the potential consequences of this type of errors, a noise preprocessing stage is often overlooked in existing literature in this field, which is necessary to improve data quality prior to modeling. In order to address this challenge, this paper proposes the usage of regression noise filters as approach to mitigate the detrimental effects of noisy IoT sensor data on evapotranspiration prediction. Additionally, we introduce the *rgnoisefilt* R package, which offers a practical and efficient implementation of noise filtering techniques for regression datasets, providing a valuable solution for handling noisy data in smart agriculture applications. The experimental results obtained emphasize the negative impacts of noise on evapotranspiration prediction performance and highlight the importance of an appropriate data treatment to mitigate system deterioration. Furthermore, the findings of this research emphasize the efficacy of the regression noise filters implemented in the *rgnoisefilt* software, enhancing the performance of the models built and providing a valuable tool for improving data quality in smart agriculture.

1. Introduction

Agriculture is a crucial sector to satisfy the food demand of the growing world population, in such a way that its cultivation objectives must be achieved under an efficient production approach (Adegbeye et al., 2020). The integration of information and communication technologies (ICT) and the Internet of Things (IoT) has transformed traditional agricultural systems into smart agriculture, enabling the collection, analysis and processing of large volumes of field data (Catalano et al., 2022; Colizzi et al., 2020; Pérez-Padillo et al., 2022). These data, obtained through multi-sensor systems, serve as valuable inputs for developing data science models that optimize resource management, such as land and water (Togneri et al., 2022, 2023). However, the reliability of IoT sensor data poses a significant challenge due to the presence of errors, which requires the implementation of preprocessing techniques to enhance data quality. This fact is particularly relevant in the context of smart agriculture, where accurate predictions of evapotranspiration are crucial for efficient water management and crop yield maximization (Barriga et al., 2022).

Evapotranspiration, which is defined as the combined processes of water loss through evaporation from soil surfaces and transpiration from crops, plays a vital role in agricultural water balance (Kumar et al., 2014; Martín et al., 2021b). The accurate estimation of evapotranspiration depends on various atmospheric, plant and soil parameters, which are measured using IoT sensors in smart agriculture systems (Douna et al., 2021; Glaroudis et al., 2020). However, errors and imperfections are inherent in these sensor data, arising from device limitations, measurement tool constraints and transcription errors (Teh et al., 2020). These imperfections introduce noise, which corrupts the dataset and can adversely affect the performance of data science methods used for evapotranspiration prediction. Thus, the application

* Corresponding author.
E-mail addresses: juanmartin@usal.es (J. Martín), joseasaezm@ugr.es (J.A. Sáez), escorchado@usal.es (E. Corchado).

https://doi.org/10.1016/j.eswa.2023.121608
Received 29 June 2023; Received in revised form 30 August 2023; Accepted 11 September 2023
Available online 16 September 2023

of preprocessing techniques to improve data quality before modeling is highly recommended in this domain.

In order to mitigate the negative consequences of noisy data, noise filters (Jiang et al., 2021; Nematzadeh et al., 2020) are data preprocessing approaches used to detect and remove erroneous values in a dataset while preserving relevant information. There exists a multitude of works showing how these techniques improve the performance of data science methods with noisy data, especially in the context of classification datasets (Garcia et al., 2016; Nematzadeh et al., 2020). However, the continuous nature of evapotranspiration implies that this type of data are considered as regression problems, where there is relatively less literature that deepens the general understanding of noise filtering or highlights applications where its usage is especially suitable. Thus, due to the frequency of noise in datasets proceeding from measurement devices (Teh et al., 2020), such as IoT weather sensors used in evapotranspiration prediction, it is important to study error handling in this type of data prior to modeling. On the other hand, even though this type of techniques usually provides good results processing errors in the output variable in classification datasets (Brodley & Friedl, 1999), their behavior dealing with attribute noise tends to depend more on data characteristics (Sáez et al., 2013). For this reason, it is particularly interesting to study the behavior of noise filters when addressing errors in input attributes in the field of regression. Finally, it is also worth mentioning that, despite the usefulness of noise filtering techniques in mitigating the consequences of erroneous data, there is a paucity of public software that implements these methods for regression problems. Making these techniques available to both smart agriculture practitioners and researchers will provide them with a useful tool for handling this type of complex noisy data.

This paper aims to analyze the impact of noise on evapotranspiration datasets. In addition, it presents the *rgnoisefilt* R package, which provides a comprehensive and diverse set of noise filters specifically designed for regression datasets. This software, which is available from the *Comprehensive R Archive Network* (CRAN) repository at https://CRAN.R-project.org/package=rgnoisefilt, is proposed as a tool to improve data quality in evapotranspiration prediction. Thus, it will be used to study the suitability of noise filtering techniques when dealing with noisy evapotranspiration data, comparing their usage against not considering preprocessing.

The experimentation carried out will be based on 20 real-world evapotranspiration datasets from relevant agricultural areas in Spain, where the introduction of noise in the input attributes (corresponding to data obtained from meteorological sensors) and the output variable (corresponding to evapotranspiration) will be analyzed separately. Up to 9 different levels of additional noise (from 0% to 40%, by increments of 5%) will be introduced into the aforementioned variables. Thus, a comprehensive experiment will be conducted, involving a total of 340 noisy evapotranspiration datasets, to examine the effects of various types and levels of errors, as well as the operation of regression noise filters within this context. Similar to related studies (Sáez & Corchado, 2022; Sáez et al., 2013), this research will evaluate the regression performance achieved when employing a highly noise-sensitive method, such as the *Nearest Neighbor* (NN) algorithm (Taunk et al., 2019), to estimate the improvement in data quality through noise filtering. Its usage will allow a better appreciation of both the effect of noisy samples on performance and the effectiveness of noise filtering methods. Additionally, the synergy between noise filters and algorithms traditionally considered robust against noise, such as *Random Forest* (RF) (Hansch, 2018) and *eXtreme Gradient Boosting* (XGBoost) (Chen & Guestrin, 2016), will be analyzed to assess whether noise treatment also benefits these methods when dealing with noisy evapotranspiration data. Finally, a study will be conducted to examine the operation of the noise filters before model building. Thus, inspired by existing research on robustness measures for classification problems (Sáez et al., 2016), a metric for quantifying the magnitude of error removed by the noise filters will be introduced and an analysis of the different sizes of treated errors will be performed. The results obtained in the experimentation will be contrasted using the appropriate statistical tests (Derrac et al., 2011) to draw robust conclusions about the use of noise filters with evapotranspiration data. It should be noted that, even though the presence of errors may be visually apparent in other specific fields of research, such as those studying label noise in image classification problems, this is not always the case in other applications like the one addressed in this work, where it is more common to analyze the problem of noisy data by examining its consequences (Nettleton et al., 2010; Zhu & Wu, 2004).

As a summary, the main contributions of this work, considering both theoretical and practical aspects, include:

1. *Addressing an under-explored problem.* This research emphasizes an issue that is not usually addressed in the evapotranspiration prediction literature, such as the usage of noisy IoT sensor data for model building.
2. *Error analysis and comparison.* A detailed analysis supported by empirical experiments of how different types and levels of noise affect the prediction of evapotranspiration is carried out.
3. *Proposal for the usage of noise filters.* The principles of regression noise filtering approaches are translated to a new domain, evapotranspiration prediction, as a tool to enhance data quality.
4. *Software implementation.* The *rgnoisefilt* R package, which provides a comprehensive collection of noise filters for regression data, is developed and presented as a significant practical contribution to both the research community and applications in smart agriculture.
5. *Synergy analysis based on robustness.* The evaluation of how regression noise filtering techniques perform with algorithms of varying noise sensitivity in evapotranspiration prediction is conducted, providing insights for robust modeling.
6. *Operation of noise filters.* The operation of the regression noise filters is analyzed from the point of view of the data, focusing especially on the magnitude of the errors treated depending on the noise type.

The remaining of this paper is organized as follows. Section 2 introduces noise filtering techniques in data science tasks. Section 3 details the characteristics of the *rgnoisefilt* package, from its installation to the usage of all its functionalities. Then, Section 4 describes the experimental framework, whereas Section 5 focuses on analyzing the impacts of errors in evapotranspiration datasets and the operation of regression noise filters in this context. Finally, Section 6 summarizes the contributions and results of this work and proposes some future research lines.

2. Noise filtering in data science tasks

Noise does not distinguish the nature of the dataset and can affect the two main types of supervised problems in data science (Gupta & Gupta, 2019; Muthukumar et al., 2020): classification, in which the output is a class label, and regression, in which the output is a numeric value. In classification, noise filtering techniques (Garcia et al., 2016; Nematzadeh et al., 2020) have been proposed as a solution for dealing with erroneous samples that are detrimental to model building. Noise treatment as a step prior to the learning stage has the advantage that erroneous samples do not affect the model construction process. Thus, noise filtering has proven to be advantageous for improving the quality of noisy data, even in scenarios involving complex borderline samples positioned at class boundaries (Rout et al., 2023; Sáez & Romero-Béjar, 2022). As a consequence, the performance of the models created is often improved compared to not considering noise preprocessing (Wang et al., 2020). Even though the elimination of noisy samples by filtering has been traditionally studied in classification, recent research shows that this type of techniques can also be applied to regression problems with satisfactory results (Jiang et al., 2021; Martín et al., 2021a). The filtering options for both classification and regression are briefly presented below.

2.1. Noise filters in classification

In the field of classification, a large number of noise filtering techniques are found in the specialized literature (Garcia et al., 2012; Nematzadeh et al., 2020). They are usually divided into two groups according to their characteristics: *(i) similarity* filters (Devroye et al., 1996), which are those mainly based on distances among samples, and *(ii) ensemble* filters (Brodley & Friedl, 1999), which are those based on the predictions of several classifiers.

The main and most popular representation within similarity filters is *Edited Nearest Neighbors* (ENN) (Devroye et al., 1996). This is based on a simple but effective principle in practice: removing a sample if its class label is different from that of the majority of its nearest neighbors, that is, its other nearest samples. Some research works show the benefits of this type of approach in a variety of applications, such as, for example, medical (Benhar et al., 2020). Many other similarity filters can be found in the literature with bases similar to ENN. For example, *Condensed Nearest Neighbors* (CNN) (Devroye et al., 1996) performs a classification with NN and stores the misclassified samples, which are taken as a training set. This process is repeated until the training set correctly classifies all unstored samples. Another example is *Reduced Nearest Neighbors* (RNN) (Gates, 1972), which includes an additional stage to remove those samples that do not impact the performance of the k-NN classifier.

On the other hand, ensemble filters usually build several classifiers from the training data and then, based on a voting scheme, decide which samples are finally removed (Verbaeten & Van Assche, 2003). The main noise filter within this group, which serves as the inspiration for many other filters, is *Ensemble Filter* (EF) (Brodley & Friedl, 1999). It classifies the training data using a μ-fold cross-validation considering classifiers belonging to different paradigms, such as decision trees, nearest neighbors and linear discriminant analysis. The noisy samples are then removed using a voting scheme on the predictions of these classifiers: consensus voting removes those samples that are misclassified by all classification models, whereas majority voting removes those samples that are misclassified by more than half of the classification models. Other noise filtering options based on ensembles are *Iterative-Partitioning Filter* (IPF) (Khoshgoftaar & Rebours, 2007), which builds a decision tree on each fold to evaluate the whole dataset in an iterative filtering process, *Dynamic Classification Filter* (DF) (Garcia et al., 2012), which applies up to nine classifiers to perform noise filtering using those with best predictions, and *Class Noise Detection and Classification* (CNDC) (Nematzadeh et al., 2020), which combines aspects of both filtering paradigms (ensembles and similarity).

2.2. Noise filters in regression

Regression noise filters are responsible for detecting and removing erroneous samples in datasets with a numeric output variable (Kordos & Blachnik, 2012). In this context, an adaptation of different classification noise filters to regression problems was recently proposed by Martín et al. (2021a). Their approach was based on the observation that most noise filters in classification usually determine whether a sample x_i contains noise by comparing its actual label y_i with the predicted label y'_i generated by a model derived from the data (Brodley & Friedl, 1999; Verbaeten & Van Assche, 2003). When these outputs are different, the sample x_i is identified as potentially noisy. Based on this premise, to transform noise filters traditionally used in classification to regression problems, it is necessary to define a method to compare the actual output value y_i with the predicted one y'_i in regression problems, since the continuous nature of the regressand generally leads to inequality in such comparisons. For this purpose, a similarity function is used within the regression noise filter:

$$similar(y_i, y'_i) = \begin{cases} true, & \text{if } |y_i - y'_i| \leq \tau \\ false, & \text{otherwise} \end{cases} \quad (1)$$

where τ is the *noise filtering threshold*. This function is included in a regression filter when the corresponding classification filter compares the predicted and actual labels (y'_i and y_i, respectively) for a sample x_i.

Another interesting approach found in the specialized literature for adapting classification noise filters to regression problems was proposed by Arnaiz-González et al. (2016b), which offers the advantage of not requiring filter modifications. Their approach involves discretizing the numerical output variable, transforming the regression dataset into a classification problem, thereby enabling the utilization of well-known classification noise filters, such as ENN. In the discretization process, the output variable is divided into equal-width intervals, testing various numbers of bins and selecting the configuration maximizing an entropy-based criterion. Once the filtering is completed with the optimal discretization, the original numerical values of the output variable are restored for those samples that were not removed. In addition, there exist other adaptations of classification noise filters to regression problems, such as the DROP2-RE and DROP3-RE methods (Arnaiz-González et al., 2016a), which are based on the well-known DROP algorithm family. These techniques focus on minimizing prediction errors while selectively removing samples from the dataset. Thus, DROP2-RE accumulates prediction errors for associated samples, only removing a sample if these errors do not increase. Based on DROP2-RE, DROP3-RE includes an additional initial noise filtering stage and organizes the samples based on the distance to their nearest enemy.

Other noise filtering proposals are originally designed for regression problems, such as the *Covering Distance Filtering* (CDF) algorithm (Jiang et al., 2021). CDF is based on the concept of *covering interval*, which is built from the predictions of multiple regression models, to divide the samples into two subsets: D_{in} for samples within the interval and D_{out} for those outside. Samples in D_{in} are considered to have low noise and are retained in the final set of clean samples. Then, the samples in D_{out} are removed one by one based on their absolute noise, stopping the removal operation at the maximum value of an objective function. Finally, it is worth considering the application of certain filtering techniques from the field of time series to regression problems (Guillen et al., 2010; Stojanović et al., 2014). For example, the approach introduced by Guillen et al. (2010), which is based on mutual information for selecting samples within the training set, can be adapted to handle noise in regression problems.

It should be noted that, even though some of the aforementioned noise filters for classification tasks, such as ENN or CNN, can be found in public packages (see, for example, the *UBL* package (Branco et al., 2016)), there is no software offering their joint implementations for regression problems. In order to provide a comprehensive and diverse collection of noise filtering methods for regression problems to practitioners in smart agriculture and, in general, to the data science community, this paper introduces the first public software implementing a wide variety of regression noise filters, the *rgnoisefilt* R package, which is described in detail in the following section.

3. Noise handling in regression problems with the rgnoisefilt package

This section describes the *rgnoisefilt* package. First, Section 3.1 presents the installation process, whereas Section 3.2 shows the contents of the software. Then, Section 3.3 illustrates the documentation related to the functions implemented. Finally, how regression filters can be called and the value returned by them are explained in Sections 3.4 and 3.5, respectively.

3.1. Installation

The *rgnoisefilt* package can be installed in R from the CRAN servers using the command:

Table 1
Regression noise filters implemented in the *rgnoisefilt* package.

Filter type	Noise filter	Reference	Function	Tuning parameters
Similarity filters	*All-k Edited Nearest Neighbors*	Tomek (1976a)	`regAENN`	`x, y, t, k`
	Blame Based Noise Reduction	Delany and Cunningham (2004)	`regBBNR`	`x, y, t, k`
	Condensed Nearest Neighbors	Devroye et al. (1996)	`regCNN`	`x, y, t`
	Edited Nearest Neighbors	Devroye et al. (1996)	`regENN`	`x, y, t, k`
	Generalized Edition	Koplowitz and Brown (1981)	`regGE`	`x, y, t, k`
	Reduced Nearest Neighbors	Gates (1972)	`regRNN`	`x, y, t`
	Condensed Nearest Neighbors by Discretization	Devroye et al. (1996)	`discCNN`	`x, y`
	Edited Nearest Neighbors by Discretization	Devroye et al. (1996)	`discENN`	`x, y, k`
	Neighborhood Cleaning Rule by Discretization	Laurikkala (2001)	`discNCL`	`x, y, k`
	Tomek Links by Discretization	Tomek (1976b)	`discTL`	`x, y`
	Decremental Reduction Optimization Procedure - 2	Arnaiz-González et al. (2016a)	`rfDROP2`	`x, y, k`
	Decremental Reduction Optimization Procedure - 3	Arnaiz-González et al. (2016a)	`rfDROP3`	`x, y, k`
	Mutual Information-based Filter	Guillen et al. (2010)	`rfMIF`	`x, y, k, alpha`
Ensemble filters	*Cross-Validation Committees Filter*	Verbaeten and Van Assche (2003)	`regCVCF`	`x, y, t, nfolds, vote`
	Dynamic Filter	Garcia et al. (2012)	`regDF`	`x, y, t, nfolds, m, vote`
	Ensemble Filter	Brodley and Friedl (1999)	`regEF`	`x, y, t, nfolds, vote`
	Fusion of Multiple Filters	Garcia et al. (2016)	`regFMF`	`x, y, t, vote`
	Hybrid Remove-Repair Filter	Miranda et al. (2009)	`regHRRF`	`x, y, t, vote`
	Iterative-Partitioning Filter	Khoshgoftaar and Rebours (2007)	`regIPF`	`x, y, t, nfolds, vote, p, s, i`
	Iterative-Robust Filter	Verbaeten (2002)	`regIRF`	`x, y, t`
	Regressand Noise Detection	Nematzadeh et al. (2020)	`regRND`	`x, y, t, nfolds, vote`
	Covering Distance Filtering	Jiang et al. (2021)	`rfCDF`	`x, y, subsets, VCdim, prob`

```
R> install.packages("rgnoisefilt")
```

In addition, the software can also be installed from GitHub with the mandate:

```
R> devtools::install_github("juanmartinsantos/
  rgnoisefilt/pkg/rgnoisefilt")
```

The above commands install all the dependencies of the package (see the *Imports* section of the CRAN website[1] for the *rgnoisefilt* package), including the regression algorithms necessary for the operation of the regression noise filters. In order to access all the functions contained in the package, it is necessary to load it using:

```
R> library(rgnoisefilt)
```

3.2. Package content

The *rgnoisefilt* package includes the implementation of a total of 22 well-known and varied regression noise filters belonging to both similarity and ensemble approaches. Table 1 shows these regression filters, along with the reference to the original proposal on which they are based, the name of the function within the package and its parameters for the default method. Note that filters whose function names start with `reg` consider the adaptation to regression problems based on threshold proposed by Martín et al. (2021a), whereas filters starting with `disc` consider the adaptation through discretization proposed by Arnaiz-González et al. (2016b). In addition, the software offers custom `print`, `summary` and `plot` functions that allow clearly and concisely displaying the result of the noise filtering process. These functions are detailed in Section 3.5.

3.3. Documentation

The information corresponding to each regression noise filter has been properly documented using *roxygen2* (Wickham et al., 2021). It can be consulted using the `help()` and `?` commands. For example, in order to check the documentation of the `regENN` noise filter, we can use:

```
R> help(regENN)
R> ?regENN
```

These commands display a user guide on the usage of the `regENN` filter, including details about the operation of the filter, the ways to call its associated functions, a description of both its arguments and the output value, additional references to research papers where the filter has been used and examples of use of the implemented functions. On the other hand, the full documentation for all the noise filters can also be found at https://cran.r-project.org/web/packages/rgnoisefilt/rgnoisefilt.pdf.

3.4. Usage of the regression noise filters

For processing noisy regression data, each noise filter in the *rgnoisefilt* package provides two standard ways of use:

- *default method*. It is characterized in that it receives the dataset through two arguments, being one a data frame containing the input attributes (parameter `x`) and the other a double vector with the regressand value for each sample (parameter `y`).
- *formula class method*. This method allows passing the whole data frame (attributes and response variable) in one of its argument (parameter `data`). In addition, the relation between the input attributes and the output variable must be indicated using the corresponding expression in the `formula` argument.

Below is an illustrative example on how to use these two methods for filtering the *rock* regression dataset found in the *datasets* package (R Core Team, 2023). The *rock* dataset contains 3 input attributes (*area*, *peri* and *shape*) of measurements of 48 rock samples from a petroleum reservoir, along with the output permeability (*perm*) of each one. The `default` method for the regression filters has the following general form:

```
filtername(x, y, p1, . . . , pn)
```

where `filtername` is the name of the function of the regression noise filter, `x` is a data frame with the input attributes, `y` is a double vector with the output variable for each sample and `p1`, ..., `pn` are specific parameters for each noise filter (see Table 1). Thus, for example, in order to use the `default` method of the `regENN` filter, we can type:

[1] https://CRAN.R-project.org/package=rgnoisefilt.

```
R> regENN(x = rock[,-ncol(rock)],
  y = rock[,ncol(rock)], t = 0.2, k = 5)

## Noise filter:
Edited Nearest Neighbors

## Parameters:
- k = 5
- t = 0.2

## Number of noisy and clean samples:
- Noisy samples: 12/48 (25%)
- Clean samples: 36/48 (75%)
```

As can be seen from the output information returned by the noise filter, `regENN` has detected noise in 12 of the samples in the dataset. On the other hand, instead of receiving the data using the `x` and `y` arguments, the method of `formula` class receives the complete dataset (input and output variables) in its `data` argument, whereas the `formula` argument is an expression that indicates the relationship between these variables. Continuing with the previous example, this method can be used as follows:

```
R> regENN(formula = perm ~ ., data = rock,
  t = 0.2, k = 5)
```

In this case, the number of clean and noisy samples using the `formula` class method coincides with that determined by the `default` method. Note that, although some filters, such as `regENN` (Devroye et al., 1996) in the example above, involve deterministic processes for noise detection (that is, for the same input data, they always return the same set of noisy samples), other filters, such as those based on ensembles (Brodley & Friedl, 1999; Miranda et al., 2009), may include random processes in their operation. For this reason, it may be advisable to set the seed for random number generation (using the `set.seed()` function) before calling the regression noise filters to ensure the replicability of the experiments.

3.5. Value returned by noise filtering functions

All the regression noise filters return an object of `rfdata` class. This is designed to unify the output value of all the methods included in the software. The `rfdata` class is a list of 11 elements with the most relevant information of the noise filtering process, including:

- `xclean`: a data frame with the input attributes of clean samples (without errors).
- `yclean`: a double vector with the output regressand of clean samples (without errors).
- `numclean`: an integer with the amount of clean samples.
- `idclean`: an integer vector with the indices of clean samples.
- `xnoise`: a data frame with the input attributes of noisy samples (with errors).
- `ynoise`: a double vector with the output regressand of noisy samples (with errors).
- `numnoise`: an integer with the amount of noisy samples.
- `idnoise`: an integer vector with the indices of noisy samples.
- `filter`: the full name of the noise filter used.
- `param`: a list of the argument values.
- `call`: the function call.

Note that, in order to access the elements returned by the noise filter in the object of `rfdata` class, the $ operator can be used. For example, if `out` is an object of `rfdata` class, `out$numnoise` can be used to access the amount of noisy samples detected by the noise filter in the dataset:

```
R> out <- regENN(x = rock[,-ncol(rock)],
  y = rock[,ncol(rock)], t = 0.2, k = 5)

R> out$numnoise

[1] 12
```

In order to properly display the results of the `rfdata` class in the R console, three specific `print`, `summary` and `plot` functions are implemented. The `print` function presents the basic information of the noise filtering process, including the name of the noise filter, its parameters and the number of noisy and clean samples in the dataset. Thus, in order to check the output of the application of the `regENN` filter in the above example, we can type:

```
R> print(out)

## Noise filter:
Edited Nearest Neighbors

## Parameters:
- k = 5
- t = 0.2

## Number of noisy and clean samples:
- Noisy samples: 12/48 (25%)
- Clean samples: 36/48 (75%)
```

On the other hand, the `summary` function displays the above information on the noise filtering process along with other supplementary details, such as the function call and the indices of the noisy samples. In addition to the `rfdata` object, it receives the logical argument `showid`, which allows the indices of noisy samples to be displayed if desired. As an example, the results of applying the `summary` method to the output of `regENN` are as follows:

```
R> summary(out, showid = TRUE)

##############################################
  ##########
     Noise filtering process: Summary
##############################################
  ##########

## Original call:
regENN(x = rock[, -ncol(rock)],
  y = rock[, ncol(rock)], t = 0.2, k = 5)

## Noise filter:
Edited Nearest Neighbors

## Parameters:
- k = 5
- t = 0.2

## Number of noisy and clean samples:
- Noisy samples: 12/48 (25%)
- Clean samples: 36/48 (75%)

## Indices of noisy samples:
26, 29, 33, 37, 38, 39, 40, 41, 42, 43, 44, 48
```

Finally, the `rfdata` class also enables displaying a graphical representation that allows for comparing data distributions before and after the noise filtering process using the `plot` method, which can be invoked as follows:

```
R> plot(out, var = 1:4, fun = "mean")
```

This function generates a plot for each of the variables specified by the `var` parameter, allowing the comparison of their value distributions before filtering, using the descriptive statistic specified by `fun`, with the data distributions from samples identified as clean and noisy by the filtering method. Specifically, the description of the parameters used by the `plot` function is as follows:

- `var` is an integer vector with the indices of variables whose distributions are compared, considering the attributes in the order in which they appear in the original data, with the output variable in the last position.
- `fun` is a character containing the name of the descriptive statistic function to compute for each distribution of the variable, or a user-defined function that returns a value from a distribution of numeric data. Some options for `fun` include ‘‘`mean`’’, ‘‘`median`’’ or ‘‘`sd`’’ (standard deviation).

Thus, applying the `plot` method to the output obtained from filtering the *rock* dataset using the parameters `var = 1:4` and `fun = ‘‘mean’’` results in the graph in Fig. 1. This figure shows, for example, how the mean of the data in the output variable in those samples identified as noisy by the filter is notably higher than the means of the original data and the data identified as clean, which are more similar. Thus, the graph constructed by the `plot` function allows visualizing how the distribution of each variable behaves throughout the noise filtering process, enabling the identification of differences and patterns between the original, clean and noisy datasets.

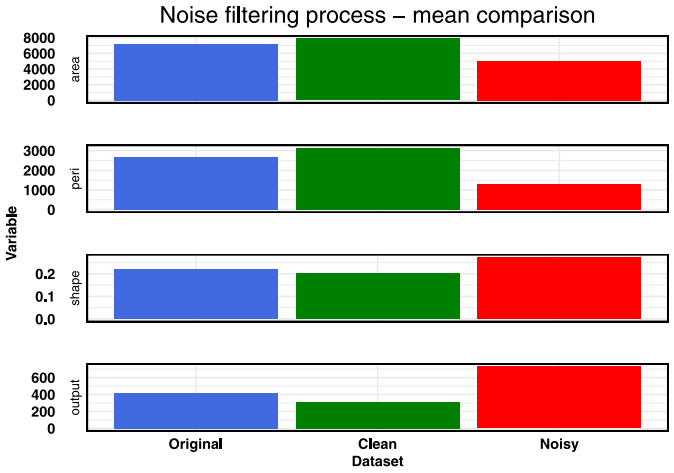


Fig. 1. Graphic representation obtained after applying the `plot` method to the filtering of the *rock* dataset using `var = 1:4` and `fun = ‘‘mean’’`.

4. Experimental framework to study noise-related phenomenology in evapotranspiration datasets

This section details the experimental framework designed to analyze the noise problem in evapotranspiration prediction. Section 4.1 describes the evapotranspiration datasets used, whereas Section 4.2 explains the methodology followed for the analysis of the results.

4.1. Real-world evapotranspiration datasets

The experimentation considers a five-year period of daily sensor data from meteorological stations of 20 relevant agricultural areas in Spain Martín et al. (2021b). The datasets, along with the number of samples and weather stations for each, are shown in Table 2. Each dataset contains 12 input attributes plus evapotranspiration (*output*).

Table 2
Description of the real-world evapotranspiration datasets used.

Dataset	Region	Stations	Samples	Dataset	Region	Stations	Samples
`als`	Almería (South)	11	19 249	`bar`	Barcelona	1	1752
`aln`	Almería (North)	11	19 496	`ler`	Lérida	1	1759
`cad`	Cádiz	4	7002	`ger`	Gerona	1	1699
`sev`	Sevilla	6	10 611	`cac`	Cáceres	5	7387
`hue`	Huelva	13	23 809	`tol`	Toledo	5	8881
`mus`	Murcia (South)	11	19 394	`avi`	Ávila	3	4081
`mun`	Murcia (North)	11	18 472	`sal`	Salamanca	10	17 245
`vas`	Valencia (South)	10	17 819	`seg`	Segovia	3	4923
`van`	Valencia (North)	10	17 572	`val`	Valladolid	5	8166
`tar`	Tarragona	1	1777	`zam`	Zamora	6	9988

The attributes included are: precipitation (PR), maximum temperature (T_{max}), minimum temperature (T_{min}), mean temperature (T), maximum relative humidity (RH_{max}), minimum relative humidity (RH_{min}), mean relative humidity (RH), mean wind speed at 2 m height (u_2), maximum wind speed at 2 m height (u_{2max}), mean wind direction at 2 m height (du_2), maximum wind direction at 2 m (du_{2max}) and solar radiation (R_s). All the variables in the datasets are normalized to the interval [0,1] before use.

In order to study the impact of noise on the above datasets, two types of noise (regressand noise and attribute noise) are considered separately, the introduction of which is based on widely used approaches to simulate noise in classification data (Sáez, 2022, 2023). For both types of noise, to inject a noise level $x\%$ into a evapotranspiration dataset, $x\%$ of the samples of each affected variable are selected and their values are replaced by other random ones within the domain. The noise levels range from $x = 0\%$ to $x = 40\%$, in steps of 5%. Note that the lowest noise level considered, that is, 0%, does not necessarily imply that the datasets are free of noise. This noise level refers to the original version of the data, which originate from real-world sources and could potentially contain inherent, uncontrolled noise to some extent –a problem relatively common when dealing with real-world datasets of meteorological variables and evapotranspiration data (Wang et al., 2018, 2014). The following procedure is employed to inject noise:

1. In a copy of the original dataset, the desired noise level $x\%$ is injected into the chosen variables.
2. The original dataset and the noisy copy are partitioned into 5 equivalent folds, that is, each fold contains the same samples in both datasets.
3. The training partitions are built from the noisy copy, whereas the test partitions are built with the samples from the original dataset.

As a consequence of the aforementioned process, 160 noisy datasets are created for each type of noise that, along with the 20 original datasets, results in a total of 340 noisy datasets.

4.2. Methodology of analysis

In order to estimate the regression performance, five runs of a 5-fold cross-validation are considered, allowing robust performance results to be obtained (Wong & Yeh, 2020). For the preprocessing of the 340 noisy datasets created, two of the most representative filters among those implemented within the *rgnoisefilt* package are chosen: regENN and regEF, which are based on the ENN (Devroye et al., 1996) and EF (Brodley & Friedl, 1999) filters. These can be considered as reference methods within similarity and ensemble filters, respectively. Since some of the noise filters in the implemented software are inspired by the principles of these methods, they tend to provide similar conclusions to those presented in this paper. In spite of their lower complexity, both regENN and regEF have shown a good balance in terms of regression performance and preprocessing speed when dealing with noisy evapotranspiration data in preliminary experiments, so they

are the final choice to analyze the operation of noise filtering in this context. The efficacy of the noise filters is analyzed based on the regression performance of the models built, which is evaluated using the RMSE metric (Eq. (2)):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y'_i - y_i)^2}, \quad (2)$$

where y'_i are the predicted values, y_i are the real values and n is number of samples. The analysis of results is divided into five different parts, each one described in a separate section:

1. *Analysis of the impact of noise on evapotranspiration datasets* (Section 5.1). This section aims to identify the type of noise (regressand noise or attribute noise) that is most detrimental to evapotranspiration prediction, as well as the noise levels that cause the most noticeable deterioration in regression performance.
2. *Suitability of regression noise filters on evapotranspiration datasets with regressand noise* (Section 5.2). This section analyzes how filtering techniques affect the prediction performance when using data with different regressand noise levels affecting the evapotranspiration variable.
3. *Suitability of regression noise filters on evapotranspiration datasets with attribute noise* (Section 5.3). This section discusses how noise filters influence the quality of evapotranspiration datasets with different attribute noise levels affecting data collected from weather sensors.
4. *Noise filtering efficacy in evapotranspiration problems with robust regression models* (Section 5.4). This section specifically focuses on examining the effectiveness of noise filters with evapotranspiration data when employing robust regression algorithms, which are commonly known for their tolerance in handling erroneous data.
5. *Analysis of effectiveness and operation of regression noise filters* (Section 5.5). This section analyzes the effectiveness of noise filters with evapotranspiration datasets from the point of view of the data processed, carrying out a study on the errors most commonly eliminated by the filters based on their magnitude.

The estimation of the regression performance in Sections 5.1–5.3 is obtained using the NN algorithm (Taunk et al., 2019) that, due to its high sensitivity to noise, will allow to easily observe the effect of different types of noise (in input and output variables), as well as the effectiveness of noise filters dealing with these errors. Section 5.4 examines the usage of RF (Hansch, 2018) and XGBoost (Chen & Guestrin, 2016), which are traditionally regarded as noise-tolerant algorithms. The parameter setup for the noise filters considers the default values in the *rgnoisefilt* package, with $k = 5$ in regENN and majority voting with $\mu = 10$ in regEF for robust modeling. For RF, the main parameters include the number of attributes chosen at each split ($mtry = 4$) and the number of trees in the forest ($ntree = 100$), whereas for XGBoost a tree booster is employed considering the learning rate ($eta = 0.3$), the maximum tree depth ($max\_depth = 6$) and the maximum number of iterations ($nrounds = 150$). Note that, in order to ensure the widest dissemination of the findings in this research, all the source code for the developed regression noise filters remains open source in the CRAN repository[2] for easy use by the research community. Additionally, the regression algorithms employed are available through well-known R packages such as *FNN* (Beygelzimer et al., 2023), *randomForest* (Liaw & Wiener, 2002) and *xgboost* (Chen et al., 2023). Moreover, to facilitate the replication and extension of this work, all datasets used are publicly accessible.[3]

[2] https://CRAN.R-project.org/package=rgnoisefilt.
[3] https://github.com/juanmartinsantos/rgnoisefilt.

Finally, the *Wilcoxon's* test (Derrac et al., 2011) is applied to compare the performance of each regression algorithm with and without filtering, obtaining its associated p-values (p_{wil}). A difference will be considered as significant if the p-value is lower than 0.1. Due to the large number of experimental results involved, averaged results for each noise type, level and filter are shown in this work, but it must be noted that the conclusions are based on the appropriate statistical tests (Derrac et al., 2011), which consider all the results.

5. Consequences and treatment of errors in evapotranspiration datasets

This section analyzes the impact of errors in evapotranspiration datasets and the suitability of treatment using noise filtering techniques. Section 5.1 presents a comparison of the consequences of noise on the input and output variables in evapotranspiration data and evaluates their effects on regression performance. Then, Section 5.2 discusses the effectiveness of noise filtering in improving prediction performance on data with regressand noise and Section 5.3 focuses on attribute noise. Finally, Section 5.4 analyzes the synergy between noise filters and robust regression algorithms, whereas Section 5.5 examines the effectiveness of the filters focusing on the magnitude of the errors removed.

5.1. Analysis of the impact of noise on evapotranspiration datasets

Fig. 2 shows the comparison of RMSE results obtained by NN for each noise level on evapotranspiration datasets when they contain regressand noise (in red) versus attribute noise (in yellow). The analysis of these results provides the following insights:

- *On the impact of regressand noise on evapotranspiration datasets.* The first fact to highlight from the results shown in Fig. 2 is the great impact that regressand noise has when creating models from evapotranspiration data. For example, with a noise level of 5%, the RMSE deteriorates by 133% with respect to the original data, showing how small amounts of errors in the output variable have significant consequences on model performance. This performance deterioration is even more pronounced with increasing noise level. Thus, the performance decline for noise levels 10%–15% exceeds 200%, whereas it surpasses 300% for noise levels 20%–25%. At the highest noise level (40%), the deterioration of performance compared to not considering additional regressand noise reaches its maximum value, approximately 500%. These results reflect the importance of treating this type of errors using noise preprocessing methods before building models from evapotranspiration data.
- *On the impact of attribute noise on evapotranspiration datasets.* The above loss of regression precision when considering higher noise levels is also observed when considering attribute noise. Thus, with the initial noise levels 5%–15%, the models experience a performance loss of up to 36%. For noise levels 20%–30%, this negative effect increases from 50% to 77% and, finally, at the highest noise levels (35%–40%), the model performance deteriorates by 91% and 109%, respectively.
- *Regressand noise versus attribute noise in evapotranspiration data.* In the comparison of how regressand noise affects regression performance versus attribute noise, it can be seen that both types of noise have a different impact on system performance. Thus, Fig. 2 shows how noise in the output variable has a more detrimental effect compared to noise in input attributes, since the RMSE obtained is higher at each noise level. For example, at the lowest noise level 5%, the RMSE for regressand noise (0.0961) is more than twice that for attribute noise (0.0465). From this noise level onwards, the difference between both types of noise is even greater. Thus, at the noise levels 10%–15%, the RMSE for regressand noise is 2.4–2.7 times that of attribute noise, whereas at noise levels 20%–40%, the difference is almost 3 times.

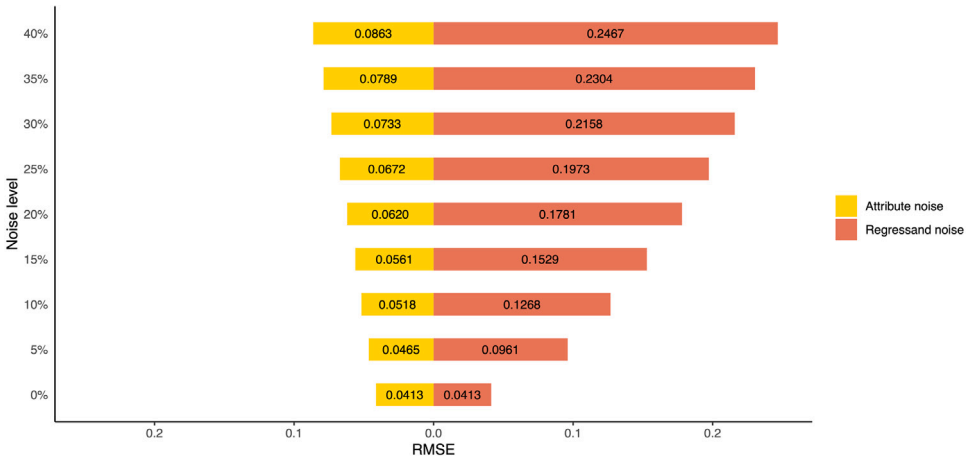


Fig. 2. RMSE results obtained by NN on evapotranspiration data for each level of regressand noise and attribute noise.

The above results show how noisy data, whether it affects input or output variables in evapotranspiration problems, is detrimental to system performance. For this reason, it is necessary to apply techniques to improve data quality before building regression models if the data are affected by errors. Thus, the usage of noise filtering methods to deal with the most detrimental type of errors, that is, regressand noise, is analyzed in Section 5.2, whereas their effectiveness in datasets with attribute noise is analyzed in Section 5.3.

5.2. Suitability of regression noise filters on evapotranspiration datasets with regressand noise

Table 3 presents the results obtained in evapotranspiration data with different regressand noise levels without preprocessing (*None*) and once they are preprocessed with the regENN and regEF filters. It includes the RMSE results of NN per noise level, the standard deviation (*sd*), the number of datasets in which each method performs best compared to the rest in the order *None*/regENN/regEF (*wins*), the *best* and the *worst* result achieved by each technique, the quartiles (Q_1, Q_2 and Q_3) and the *skewness* of the distribution of RMSE results. The *p*-values corresponding to *Wilcoxon*'s test (p_{wil}) after comparing each noise filter against *None* are also shown for each noise level (those *p*-values showing significant differences are highlighted in bold). Additionally, Fig. 3 shows the box-plots with the distribution of the RMSE results of each of the methods for each noise level. Based on the results of Table 3 and Fig. 3, the following conclusions can be drawn:

- *On the behavior of regression noise filters on evapotranspiration data without additional noise.* Table 3 shows how the regENN and regEF filters are able to preserve, and even improve, the original performance of the model (*None*) in the absence of additional noise. Thus, regENN obtains the best performance among the three methods, with the lowest RMSE value. Nevertheless, the difference in performance between regENN and the other two methods is relatively small. These results suggest that, although noise filters are designed to mitigate the noise problem, they do not negatively affect model performance when additional noise is not considered.
- *On the behavior of regression noise filters on evapotranspiration data with regressand noise.* The regression models improve their performances using both noise filters against not preprocessing at all the noise levels. Thus, at the noise level 5%, the regression error of *None* is more than twice that of the noise filters. This behavior is continued when the noise level increases and both filters maintain a better performance than *None*. Among them, regEF is shown as the most effective, with a performance deterioration of 82% between the case without noise and the highest noise level. On the other hand, Fig. 3 shows that, initially, both filters obtain similar performances but it is possible to appreciate differences between them at higher noise levels. The above results suggest that regEF is more suitable to mitigate regressand noise in evapotranspiration data. This superiority of regEF is also reflected in the study of quartiles, since it outperforms the other methods in these measures when the data is affected by noise.
- *On the stability of the results offered by the regression noise filters*. regEF and regENN usually have the lowest standard deviation values compared to *None*, indicating that they produce more homogeneous results. It should be noted that the standard deviation of the noise filters does not necessarily increase along with the noise level, indicating that their results may be even more homogeneous in the presence of noise. The interquartile range at the different noise levels, which can be calculated from the Q_1 and Q_3 quartiles, is also an indicator of the stability of the results of each method. For *None* and regENN, the interquartile range is generally greater at the highest noise levels (35%–40%). However, this situation does not occur for regEF, showing a greater homogeneity of these results. These facts are another indicator that noise filters can improve both the performance and stability of the results obtained when dealing with noisy evapotranspiration data.
- *On the number of datasets in which each method obtains the best result.* Regarding the datasets with regressand noise in which each method offers the best result, Table 3 shows that regENN is the best without additional noise in 17 datasets against *None* and 18 against regEF. For the following noise levels, regEF consistently performs better than regENN and *None* for most datasets. It outperforms *None* in all the 20 datasets and only loses to regENN in one dataset at the noise level 5%.
- *On the best and worst result obtained by each method with evapotranspiration data.* Both filtering techniques outperform *None* in terms of the best RMSE value obtained across all the noise levels, except in the case without additional noise. The performance of the filters is comparable up to a noise level 15%, after which regEF clearly exhibits better results than regENN. This trend is maintained in the worst RMSE obtained by each technique, with regEF consistently achieving the lowest RMSE for all the noise levels. In fact, its worst result is below 0.0850 (at the noise level 40%), which is reached by *None* and regENN at the noise levels 5% and 25%, respectively.
- *On the distribution shape of the RMSE values of the noise filters.* *None* has its highest positive skewness values at the noise levels 0, 10, 35 and 40%, indicating that the distribution is moderately skewed to the right. For the other noise levels (5, 15%–30%), the

Table 3
RMSE results obtained by NN after applying regression noise filters on evapotranspiration datasets with regressand noise.

Method	Metric	0%	5%	10%	15%	20%	25%	30%	35%	40%
None	*RMSE*	0.0413	0.0961	0.1268	0.1529	0.1781	0.1973	0.2158	0.2304	0.2467
	sd	0.0080	0.0086	0.0068	0.0063	0.0074	0.0071	0.0104	0.0095	0.0098
	wins	−/13/15	−/0/0	−/0/0	−/0/0	−/0/0	−/0/0	−/0/0	−/0/0	−/0/0
	best	0.0287	0.0798	0.1162	0.1405	0.1679	0.1864	0.1932	0.2163	0.2344
	worst	0.0580	0.1149	0.1458	0.1630	0.1901	0.2105	0.2318	0.2473	0.2662
	Q_1	0.0371	0.0899	0.1232	0.1485	0.1716	0.1923	0.2087	0.2243	0.2397
	Q_2	0.0382	0.0952	0.1267	0.1530	0.1774	0.1965	0.2150	0.2275	0.2444
	Q_3	0.0466	0.1008	0.1308	0.1569	0.1839	0.2021	0.2239	0.2385	0.2538
	skewness	0.7536	0.3330	0.8971	−0.0297	0.2420	0.1190	−0.2520	0.5123	0.6231
regENN	*RMSE*	0.0409	0.0445	0.0493	0.0565	0.0655	0.0737	0.0862	0.0985	0.1106
	sd	0.0075	0.0071	0.0066	0.0075	0.0073	0.0057	0.0068	0.0063	0.0063
	wins	17/−/18	20/−/1	20/−/0	20/−/0	20/−/0	20/−/0	20/−/0	20/−/0	20/−/0
	best	0.0287	0.0333	0.0391	0.0473	0.0570	0.0671	0.0783	0.0905	0.1022
	worst	0.0556	0.0580	0.0621	0.0747	0.0837	0.0889	0.1039	0.1113	0.1227
	Q_1	0.0367	0.0402	0.0450	0.0517	0.0607	0.0690	0.0821	0.0926	0.1055
	Q_2	0.0382	0.0424	0.0472	0.0536	0.0631	0.0721	0.0849	0.0984	0.1115
	Q_3	0.0455	0.0495	0.0537	0.0597	0.0691	0.0769	0.0888	0.1038	0.1149
	skewness	0.6325	0.5869	0.6479	1.0680	1.0494	1.2078	1.0020	0.3182	0.2135
	p_{wil}	**0.0528**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
regEF	*RMSE*	0.0412	0.0435	0.0463	0.0493	0.0522	0.0559	0.0595	0.0656	0.0749
	sd	0.0073	0.0070	0.0066	0.0069	0.0068	0.0068	0.0064	0.0064	0.0060
	wins	5/2/−	20/19/−	20/20/−	20/20/−	20/20/−	20/20/−	20/20/−	20/20/−	20/20/−
	best	0.0293	0.0324	0.0366	0.0399	0.0424	0.0455	0.0516	0.0564	0.0641
	worst	0.0556	0.0576	0.0595	0.0622	0.0647	0.0703	0.0717	0.0778	0.0841
	Q_1	0.0372	0.0397	0.0425	0.0444	0.0481	0.0524	0.0553	0.0616	0.0714
	Q_2	0.0387	0.0413	0.0442	0.0471	0.0497	0.0536	0.0565	0.0646	0.0749
	Q_3	0.0461	0.0478	0.0506	0.0533	0.0570	0.0596	0.0639	0.0682	0.0796
	skewness	0.5990	0.6324	0.7419	0.7202	0.6788	0.8347	0.8023	0.5898	−0.1619
	p_{wil}	0.1230	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**

skewness is between −0.5 and 0.5, showing that the distribution is approximately symmetric (Bulmer, 1979). For regENN, the skewness values are higher than 0.5 up to a noise level 10%, so the distribution is moderately skewed to the right. At the noise levels 15%–30%, the skewness values exceed +1, showing that the distribution is highly skewed (Bulmer, 1979), whereas the skewness suggests an approximately symmetric distribution for the noise levels 35%–40%. On the other hand, regEF shows skewness values greater than 0.5 for noise levels ranging from 0 to 35%, indicating that the distribution is also moderately skewed towards the right. The above results show that the distribution of RMSE values obtained by the filters at most noise levels is skewed to the right (which does not occur for *None* to the same extent), indicating that many of these results are grouped around relatively low values in the distribution. This fact, together with the best average performance and lower dispersion, supports the better behavior of noise filters dealing with evapotranspiration data with regressand noise.

- *On the statistical significance of the improvements using regression noise filters compared to not preprocessing.* The superiority of the noise filters over not preprocessed data is also supported by the results of *Wilcoxon*'s test, which reveals significant differences between the filtered and unfiltered data at all the noise levels (except for regEF without additional noise, where no statistical differences are appreciated).

The analysis of results in this section reveals that regression noise filters are an effective tool for dealing with errors in the output variable of evapotranspiration datasets. They offer better results compared to not preprocessing, as shown by the RMSE results and the statistical tests performed, being also more homogeneous in all the datasets addressed. The next section analyzes how these methods behave when noise affects input variables in evapotranspiration datasets, which represents a complex scenario where errors can alter the values of any attribute.

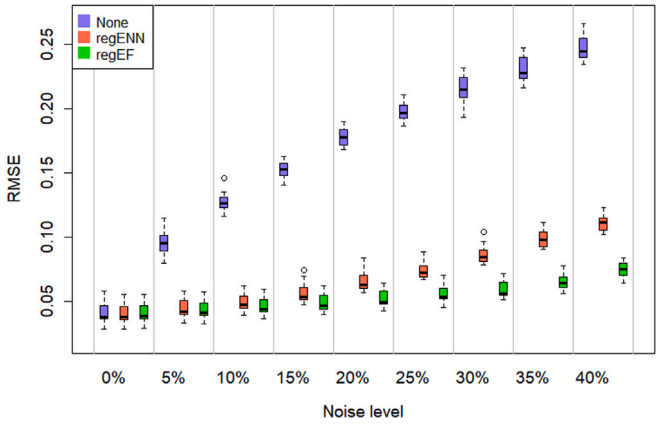


Fig. 3. Box-plots representing the distribution of RMSE results obtained by NN in evapotranspiration datasets at each regressand noise level.

5.3. Suitability of regression noise filters on evapotranspiration datasets with attribute noise

Table 4 shows the results obtained by NN for evapotranspiration data with different attribute noise levels, whereas Fig. 4 illustrates the box-plots representing the RMSE distribution for each method. They compare the results without preprocessing (*None*) and using the regENN and regEF filters.

From these results, the following comments can be made:

- *On the behavior of regression noise filters on evapotranspiration data with attribute noise.* The regression performance using the noise filters against not preprocessing improves at all the noise levels. At the lowest noise levels, the three methods obtain a similar performance. However, from the noise level 15% onwards, both filters have clearly better performances than *None*, with regEF

Table 4
RMSE results obtained by NN after applying regression noise filters on evapotranspiration datasets with attribute noise.

Method	Metric	0%	5%	10%	15%	20%	25%	30%	35%	40%
None	*RMSE*	0.0413	0.0465	0.0518	0.0561	0.0620	0.0672	0.0733	0.0789	0.0863
	sd	0.0080	0.0083	0.0092	0.0093	0.0110	0.0119	0.0139	0.0134	0.0154
	wins	–/13/15	–/0/0	–/0/0	–/0/0	–/0/0	–/0/0	–/0/0	–/0/0	–/0/0
	best	0.0287	0.0356	0.0413	0.0461	0.0511	0.0555	0.0604	0.0639	0.0681
	worst	0.0580	0.0637	0.0721	0.0804	0.0915	0.1040	0.1117	0.1096	0.1195
	Q_1	0.0371	0.0413	0.0461	0.0500	0.0546	0.0599	0.0641	0.0690	0.0769
	Q_2	0.0382	0.0431	0.0480	0.0521	0.0566	0.0626	0.0670	0.0747	0.0796
	Q_3	0.0466	0.0516	0.0566	0.0609	0.0683	0.0722	0.0805	0.0866	0.0937
	skewness	0.7536	0.9539	1.0927	1.3794	1.3370	1.8118	1.5317	1.0812	1.1040
regENN	*RMSE*	0.0409	0.0452	0.0501	0.0543	0.0594	0.0648	0.0697	0.0756	0.0815
	sd	0.0075	0.0073	0.0081	0.0081	0.0089	0.0108	0.0111	0.0120	0.0130
	wins	17/–/18	20/–/4	20/–/0	20/–/1	20/–/1	20/–/0	20/–/0	20/–/0	20/–/0
	best	0.0287	0.0350	0.0402	0.0451	0.0498	0.0542	0.0576	0.0616	0.0668
	worst	0.0556	0.0597	0.0669	0.0726	0.0798	0.0975	0.0952	0.1010	0.1111
	Q_1	0.0367	0.0406	0.0444	0.0484	0.0532	0.0583	0.0625	0.0678	0.0736
	Q_2	0.0382	0.0426	0.0473	0.0511	0.0553	0.0607	0.0651	0.0726	0.0763
	Q_3	0.0455	0.0502	0.0550	0.0586	0.0654	0.0701	0.0772	0.0838	0.0876
	skewness	0.6325	0.7418	0.8625	1.0720	1.0168	1.6996	1.2049	0.9399	1.2088
	p_{wil}	**0.0528**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
regEF	*RMSE*	0.0412	0.0447	0.0488	0.0530	0.0575	0.0624	0.0669	0.0723	0.0779
	sd	0.0073	0.0070	0.0073	0.0080	0.0090	0.0101	0.0103	0.0107	0.0119
	wins	5/2/–	20/16/–	20/20/–	20/19/–	20/19/–	20/20/–	20/20/–	20/20/–	20/20/–
	best	0.0293	0.0349	0.0393	0.0439	0.0473	0.0518	0.0547	0.0592	0.0646
	worst	0.0556	0.0588	0.0650	0.0694	0.0776	0.0908	0.0906	0.0970	0.1058
	Q_1	0.0372	0.0405	0.0440	0.0480	0.0514	0.0562	0.0609	0.0646	0.0711
	Q_2	0.0387	0.0423	0.0464	0.0501	0.0535	0.0585	0.0622	0.0700	0.0730
	Q_3	0.0461	0.0499	0.0536	0.0578	0.0627	0.0673	0.0750	0.0799	0.0845
	skewness	0.5990	0.6841	0.7624	0.9553	1.0055	1.4258	1.1195	0.8693	1.1933
	p_{wil}	0.1230	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**

achieving the lowest RMSE. Fig. 4 also shows this trend in the results, with the RMSE increasing in each technique along with the noise level. These results suggest that both filters are appropriate to mitigate the presence of noise in the input attributes of evapotranspiration data, being regEF slightly better at the highest noise levels. The quartiles also reflect this good performance of regEF compared to the other methods, showing lowest RMSE values at most of the noise levels.

- *On the stability of the results offered by the regression noise filters with attribute noise*. The standard deviation of each method shows that the lowest values are usually obtained using regEF, revealing that it produces more homogeneous results. Furthermore, all the methods exhibit a similar behavior, with their standard deviation values generally increasing along with the noise level. Note that these results differ from those previously shown for regressand noise, where the dispersion does not necessarily increase along with the noise level.
- *On the number of datasets in which each method obtains the best result*. Table 4 shows that regENN and regEF consistently outperform *None* in those cases considering the introduction of additional noise. These results highlight the importance of employing preprocessing techniques on evapotranspiration data regardless of the type of noise. Regarding the comparison between noise filters, regEF appears to be more effective in countering attribute noise in evapotranspiration data, generally providing better results.
- *On the best and worst result obtained by each method in evapotranspiration data*. Table 4 indicates that all the methods obtain close values in terms of the *best* RMSE at the lowest noise levels. For intermediate-high noise levels, regENN and regEF generally show larger differences compared to *None*. This trend is usually maintained when considering the *worst* RMSE obtained by each technique.
- *On the distribution shape of the RMSE values of the noise filters*. It is worth noting that all the methods have positive skewness values for all the noise levels, which indicates that the distributions are usually skewed towards the right. For example, at the noise levels 0%–5%, *None* presents values between 0.5 and 1, indicating that the distribution is moderately skewed (Bulmer, 1979). Similarly, regENN shows skewness values in the same range at the noise levels 0%–10%, whereas regEF exhibits this range until the noise level 15%. Regarding the subsequent noise levels, all the methods generally have skewness values above 1. These results are somewhat different than those of regressand noise, where a different skewness is observed depending on the method and noise level.
- *On the statistical significance of the improvements using regression noise filters compared to not preprocessing*. The effectiveness of the noise filters compared to the unprocessed data when dealing with evapotranspiration datasets with attribute noise is reinforced by the results of *Wilcoxon*'s test, which shows significant differences between both noise filters and *None* for all the noise levels (except with regEF without additional noise).

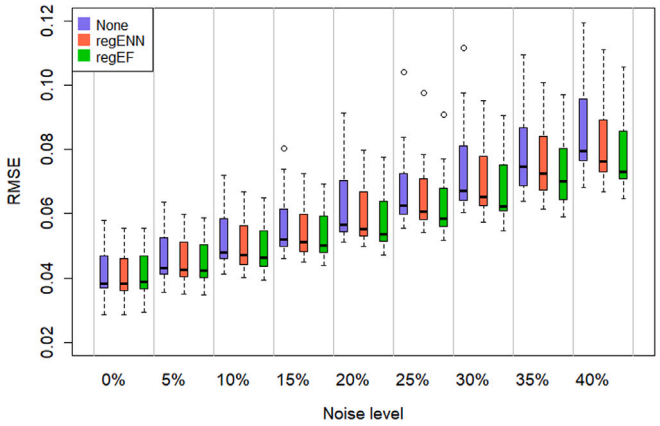


Fig. 4. Box-plots representing the distribution of RMSE results obtained by NN in evapotranspiration datasets at each attribute noise level.

The results of the filtering methods on evapotranspiration datasets with attribute noise also show the good performance of these preprocessing techniques in this scenario. Although the impact of attribute

Table 5
RMSE results of robust regression algorithms after applying noise filters on evapotranspiration datasets.

Noise	Algorithm	Filter	0%	5%	10%	15%	20%	25%	30%	35%	40%
Regressand	RF	None	0.0218	0.0341	0.0467	0.0608	0.0756	0.0891	0.1012	0.1137	0.1260
		regENN	0.0218	0.0224	0.0235	0.0250	0.0277	0.0311	0.0368	0.0438	0.0518
		p_{wil}	0.8124	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
		regEF	0.0219	0.0225	0.0232	0.0244	0.0260	0.0283	0.0313	0.0362	0.0434
		p_{wil}	0.2455	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
	XGBoost	None	0.0216	0.0480	0.0635	0.0790	0.0941	0.1089	0.1218	0.1335	0.1460
		regENN	0.0215	0.0230	0.0252	0.0285	0.0331	0.0377	0.0448	0.0531	0.0623
		p_{wil}	0.6026	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
		regEF	0.0216	0.0231	0.0247	0.0269	0.0295	0.0328	0.0365	0.0424	0.0506
		p_{wil}	0.4859	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
Attribute	RF	None	0.0218	0.0250	0.0278	0.0305	0.0331	0.0358	0.0386	0.0422	0.0453
		regENN	0.0218	0.0244	0.0269	0.0293	0.0318	0.0343	0.0370	0.0400	0.0432
		p_{wil}	0.8124	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
		regEF	0.0219	0.0241	0.0266	0.0288	0.0313	0.0335	0.0362	0.0393	0.0421
		p_{wil}	0.2455	**<0.001**	**0.0014**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
	XGBoost	None	0.0216	0.0266	0.0299	0.0334	0.0367	0.0402	0.0439	0.0481	0.0525
		regENN	0.0215	0.0251	0.0281	0.0308	0.0340	0.0368	0.0401	0.0429	0.0468
		p_{wil}	0.6026	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**
		regEF	0.0216	0.0244	0.0275	0.0303	0.0331	0.0356	0.0385	0.0417	0.0451
		p_{wil}	0.4859	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**	**<0.001**

noise is typically lower than that of regressand noise, applying noise filters improves regression performance, usually providing statistically better results than not considering preprocessing. Among the noise filtering techniques, the ensemble-based regEF filter generally performs better than the similarity-based regENN filter.

5.4. Noise filtering efficacy in evapotranspiration problems with robust regression models

Table 5 presents the RMSE obtained by each robust regression algorithm considered (RF and XGBoost) for each type and level of noise. It compares the performance results of each filtering method to those without preprocessing. Additionally, the *p*-values from *Wilcoxon*'s test (p_{wil}) for these comparisons are also presented.

In the presence of additional noise levels from 5% to 40%, the filtering methods consistently outperform no preprocessing for both robust regression algorithms, RF and XGBoost, in handling regressand and attribute noise. These results show statistically significant differences, even in those cases with relatively low levels of additional noise. This fact highlights the effectiveness of filtering methods to improve regression performance when dealing with noisy evapotranspiration datasets, despite the usage of robust algorithms that are traditionally known for their ability to handle erroneous data. Thus, noise filtering plays a significant role in improving data quality when using robust methods through the handling of those noisy samples that these algorithms have not been able to manage effectively.

In the scenario with no additional noise (0%), while the noise filters do not exhibit statistically significant performance improvements, they do not deteriorate the results either. This fact may be attributed to the capacity of robust regression models to handle small amounts of errors in the data more efficiently, making the application of noise filters have a limited margin for improvement. However, it is important to note that, even in this case, filtering offers other additional advantages, such as a reduction in the size of the training dataset, which can potentially speed up the model creation process and improve its interpretability. Thus, the performance improvements are statistically significant in scenarios that consider additional noise in the data and, even in the absence of additional noise, noise filters offer other additional advantages while maintaining system performance, making that their general usage can be recommended for regression tasks involving evapotranspiration data.

5.5. Analysis of effectiveness and operation of regression noise filters

This section analyzes the effectiveness of noise filters with evapotranspiration datasets before building regression models, aiming to deepen the understanding of their data treatment. Since we are dealing with noise affecting numerical variables, either in the regressand or the input attributes, it is especially interesting to carry out this analysis from the point of view of the magnitude of the errors on which the filtering process focuses. Because the noise introduction process is performed in a supervised manner, the exact magnitude of all the errors introduced into the data can be known. These errors are defined as the absolute difference between the value in the original dataset and the corresponding corrupted value in the noisy dataset. Based on them, and inspired by previous works on robustness metrics in classification problems (Sáez et al., 2016), we define the *Error Mitigation Index* (EMI) as follows:

$$EMI = \frac{E_x - E_f}{E_x} \cdot 100, \tag{3}$$

where E_x is the total error in a noisy dataset with a noise level of $x\%$ and E_f is the total error after applying the filtering process. The EMI metric provides information about the percentage by which the error magnitude in the data has been reduced after using the regression noise filter, which is an indicator of how well the filter handles noise. Additionally, it is also interesting to analyze the type of errors, based on their magnitude, that are predominantly addressed by the filtering process. In order to achieve this, the errors in the noisy dataset are ordered from smallest to largest and three groups of equal size are created: *low* (representing the smallest errors), *medium* (representing the intermediate errors) and *high* (representing the largest errors). Examining the percentage of errors removed in each of these groups using the filtering methods will allow a deeper understanding of their behavior in regression problems. Table 6 shows the mean EMI results and error removal percentages in each group (*low*, *medium* and *high*) for all the evapotranspiration datasets, considering each noise type, filtering method and noise level.

From the results in Table 6, several observations arise for both noise types. As the EMI results show, both filters achieve remarkable reductions in total error magnitude when dealing with regressand noise in evapotranspiration datasets, a noise type they are originally designed for. Thus, the error reduction percentages for regENN range between

Table 6
EMI results and error removal by magnitude group (*low*, *medium* and *high*) for each noise type, filter and noise level.

Noise	Filter	Metric	5%	10%	15%	20%	25%	30%	35%	40%
Regressand	regENN	*EMI*	88.01	86.70	85.82	84.51	84.22	82.84	82.18	81.30
		Low	4.84	5.94	8.75	10.44	13.93	16.07	18.57	21.81
		Medium	83.99	77.55	74.63	71.33	70.10	67.23	64.97	63.14
		High	99.90	99.77	99.49	98.89	98.62	97.51	96.96	95.91
	regEF	*EMI*	91.98	91.64	91.92	92.06	92.59	92.72	92.80	92.76
		Low	16.05	16.73	21.65	23.91	30.34	34.59	37.95	42.82
		Medium	94.12	92.27	91.89	91.89	91.86	91.18	90.53	89.27
		High	100.00	100.00	99.99	100.00	99.99	99.98	100.00	100.00
Attribute	regENN	*EMI*	3.48	4.32	5.81	7.50	9.39	11.73	14.31	16.82
		Low	0.94	1.69	2.59	3.76	5.18	7.00	8.88	11.27
		Medium	2.05	2.80	4.31	5.95	7.47	9.67	12.04	14.33
		High	4.43	5.33	6.94	8.73	10.80	13.34	16.07	18.67
	regEF	*EMI*	11.81	14.89	18.20	21.90	25.43	29.50	33.66	37.89
		Low	4.43	7.56	10.58	13.66	17.48	21.10	25.39	29.47
		Medium	8.74	11.54	15.27	18.69	22.28	26.73	30.58	34.67
		High	14.08	17.31	20.49	24.40	27.90	31.88	36.14	40.42

88.01% in data with a noise level of 5% and 81.30% considering a noise level of 40%. On the contrary, regEF achieves higher EMI values and its percentages slightly increase at highest noise levels, reaching an error reduction of 92.76% for datasets with 40% of noise. The percentages of errors removed in each magnitude group highlight the emphasis of filters on addressing errors in the *high* group, achieving rates higher than 95% for regENN and close to 100% for regEF. Errors in the *medium* group also show relevant reductions, while the *low* group receives less attention from both filters. These results reveal a difference from the common approach to handling class noise in classification problems, where the removal of most mislabeled samples is generally beneficial for performance (Zhu & Wu, 2004). On the other hand, the removal of noisy samples in regression problems seems to be influenced by the magnitude of the introduced errors and excluding samples with small errors that still contribute to model construction may not be imperative. Finally, it should be noted that, for all magnitude groups and noise levels, regEF provides higher removal rates than regENN, indicating that it is a more aggressive filter. This fact could explain its better results shown in Sections 5.2–5.3 and make this type of filter more recommendable for the treatment of noisy evapotranspiration data.

Regarding attribute noise, even though both EMI values and error removal by magnitude group are still considerable (particularly for regEF, which achieves noise reductions between 11.81% and 37.89%), the results obtained are notably lower compared to regressand noise. This interesting finding requires an analysis of its potential causes, which may help to better understand how regression noise filters deal with different noise types and their implications in model building. First, attribute noise is known to be usually more complex to handle than output noise due to the potential presence of errors in multiple attributes within a single sample, while still having a unique output variable value (Hulse et al., 2007). On the other hand, it should be noted that, although a sample may contain error in one of the attributes, its other attributes and its regressand value may still contain valuable information for model building. This fact is even more relevant if the error is small, if the attribute has little importance in the regression task or if it is highly correlated with other attributes that have a greater weight in the final model. These circumstances may imply that this attribute has little impact on the construction of models and, therefore, it is not always necessary to eliminate these samples with errors. Another aspect to consider is that, due to the process of introducing noise independently into each attribute of the dataset, it is possible that, even with low noise levels, many of the samples in the dataset may have some of their attributes containing errors. In this context, removing all samples containing errors from the dataset would not be feasible with any of the filters, as it would significantly hinder the model construction process. Finally, as observed in Section 5.1, attribute noise has a smaller impact on regression performance compared to regressand noise. This result may imply that an exhaustive noise elimination process is not essential to achieve performance enhancements. The aforementioned facts suggest a more selective attribute noise removal process, focused on eliminating the most relevant errors that might affect model construction.

6. Conclusions

This research has analyzed in detail a common problem in the field of evapotranspiration prediction within smart agriculture: the presence of noise in IoT sensor data. Based on the specialized literature focused on studying noise effects (Nematzadeh et al., 2020; Puri & Gupta, 2021), a comprehensive analysis of the impacts on performance of different types and levels of errors in evapotranspiration datasets has been conducted. The results obtained have revealed that the presence of noise has negative effects on regression performance, emphasizing the need for appropriate noise mitigation strategies within this context as demonstrated in other fields dealing with sensor data, such as manufacturing and industrial processes (Liu et al., 2020). Among the types of noise studied, regressand noise has shown a greater impact on system performance than attribute noise. A rapid deterioration in performance has been observed at low noise levels for both noise types, with differences becoming more pronounced as the noise level increases. These results obtained with regression datasets can be considered equivalent to those in the field of classification, where the noise phenomenology has been widely studied and class noise is known for its greater disruptive power (Zhu & Wu, 2004). Thus, the findings of this research have shown that the difference in impact between regressand and attribute noises becomes apparent even when considering the minimum noise level of 5%: while attribute noise results in a 13% performance degradation, regressand noise deteriorates it by up to 133%. This fact underscores the importance of analyzing the noise type that may affect the dataset treated, as it can significantly impact the prediction performance of preprocessing methods and models built (Sáez et al., 2013).

In order to mitigate the negative effects of sensor noise in evapotranspiration datasets, this paper has presented the *rgnoisefilt* package, which has been proposed as a solution to enhance the quality of noisy evapotranspiration data. Difference from the classification domain, where several R packages implement noise treatment methods (Branco et al., 2016; Zheng & Jin, 2020), this software stands out as a comprehensive collection of noise filtering techniques designed specifically for regression data, which makes it a valuable tool for the scientific community and applications in smart agriculture. Thus, two of the most representative noise filters within the *rgnoisefilt* package, regENN and regEF, have been proposed and evaluated to improve the reliability and quality of noisy evapotranspiration data. The results with

regressand noise datasets have shown that both techniques enhance the performance compared to not preprocessing, aligning with the general behavior of noise filters when dealing with label noise in classification problems (Garcia et al., 2016). Generally, regEF has obtained the best performances and has produced more stable results with the lowest standard deviation values. Thus, although both filters show improvements over not preprocessing in this scenario, these findings suggest that regEF is better suited for mitigating the effects of regressand noise in evapotranspiration data and it can be effectively used to enhance the quality of these data in practical applications. On the other hand, the analysis on evapotranspiration data with attribute noise has shown that employing noise filtering techniques can also improve the regression performance. This result with noisy evapotranspiration regression data is of particular interest since, in the field of classification, the effectiveness of filtering attribute noise is usually more dependent on data characteristics (Sáez et al., 2013).

In addition to the previous findings, a study of the synergy between noise filters and algorithms traditionally considered robust against noise, such as RF and XGBoost, has been conducted. The results obtained show that both algorithms obtain statistically significant performance improvements when employing noise filtering techniques with additional noise levels ranging from 5% to 40%, for both regressand and attribute noise. Additionally, a study has been conducted to examine the operation of the noise filters before model building and, based on existing works on robustness measures for classification problems (Sáez et al., 2016), the EMI metric has been proposed to quantify the magnitude of error removed. An analysis of the types of errors that are the focus of the filtering process has also been conducted, revealing that larger errors are removed in greater proportions. Thus, the behavior of noise filters in regression problems appears to be influenced by error magnitude. Excluding samples with small errors, which still contribute to model construction, may not be essential. This differs from the common approach in classification problems, where removing most mislabeled samples generally improves the performance (Zhu & Wu, 2004).

Finally, it is important to highlight that the noise filtering methods have shown competitive performance results when no additional noise has been introduced into the data, particularly when using the regENN filter with the NN noise-sensitive algorithm. When using the RF and XGBoost robust algorithms, the noise filters do not show statistically significant performance improvements in the absence of additional noise in the data; however, they also do not cause performance degradation. Even in this case, filtering offers benefits such as reducing the size of the training dataset, potentially accelerating the model creation process and improving interpretability. These facts support the general recommendation to use noise filters for regression tasks involving noisy evapotranspiration data.

In future works we plan to study the operation of regression noise filters when other types of errors affect smart agriculture datasets, such as Gaussian noise, as well as designing new noise preprocessing techniques to correct, instead of eliminating, errors in evapotranspiration data. Additionally, expanding this research to include diverse sensor datasets from various domains, such as healthcare monitoring or industrial process control, could provide additional knowledge on the performance of regression noise filters under different conditions.

CRediT authorship contribution statement

Juan Martín: Investigation, Methodology, Software, Formal analysis, Writing – original draft, Writing – review & editing. **José A. Sáez:** Supervision, Project administration, Conceptualization, Methodology, Writing – review & editing. **Emilio Corchado:** Supervision, Project administration, Conceptualization.

Declaration of competing interest

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

Data availability

The software associated with this paper is openly available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=rgnoisefilt.

Acknowledgments

The authors wish to thank the anonymous reviewers for their valuable time, corrections and insightful suggestions, which have helped to identify key issues and improve the quality of the manuscript.

References

Adegbeye, M., Ravi, P., Obaisi, A., Elghandour, M., Oyebamiji, K., Salem, A., Morakinyo-Fasipe, O., Cipriano-Salazar, M., & Camacho-Díaz, L. (2020). Sustainable agriculture options for production, greenhouse gasses and pollution alleviation, and nutrient recycling in emerging and transitional nations - An overview. *Journal of Cleaner Production*, *242*, Article 118319.

Arnaiz-González, A., Díez-Pastor, J. F., Rodríguez, J. J., & García-Osorio, C. I. (2016a). Instance selection for regression: Adapting DROP. *Neurocomputing*, *201*, 66–81.

Arnaiz-González, A., Díez-Pastor, J. F., Rodríguez, J. J., & García-Osorio, C. I. (2016b). Instance selection for regression by discretization. *Expert Systems with Applications*, *54*, 340–350.

Barriga, J., Blanco-Cipollone, F., Trigo-Córdoba, E., García-Tejero, I., & Clemente, P. (2022). Crop-water assessment in Citrus (Citrus sinensis L.) based on continuous measurements of leaf-turgor pressure using machine learning and IoT. *Expert Systems with Applications*, *209*, Article 118255.

Benhar, H., Idri, A., & Fernández-Alemán, J. (2020). Data preprocessing for heart disease classification: A systematic literature review. *Computer Methods and Programs in Biomedicine*, *195*, Article 105635.

Beygelzimer, A., Kakadet, S., Langford, J., Arya, S., Mount, D., & Li, S. (2023). FNN: Fast nearest neighbor search algorithms and applications. URL: https://CRAN.R-project.org/package=FNN - R package version 1.1.3.2.

Branco, P., Ribeiro, R., & Torgo, L. (2016). UBL: an R package for utility-based learning. CoRR, abs/1604.08079.

Brodley, C., & Friedl, M. (1999). Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, *11*, 131–167.

Bulmer, M. (1979). *Principles of statistics*. Courier Corporation.

Catalano, C., Paiano, L., Calabrese, F., Cataldo, M., Mancarella, L., & Tommasi, F. (2022). Anomaly detection in smart agriculture systems. *Computers in Industry*, *143*, Article 103750.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proc. 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785–794).

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., Li, Y., & Yuan, J. (2023). xgboost: Extreme Gradient Boosting. URL: https://CRAN.R-project.org/package=xgboost - R package version 1.7.5.1.

Colizzi, L., Caivano, D., Ardito, C., Desolda, G., Castrignanò, A., Matera, M., Khosla, R., Moshou, D., Hou, K.-M., Pinet, F., Chanet, J.-P., Hui, G., & Shi, H. (2020). Introduction to agricultural IoT. In *Agricultural internet of things and decision support for precision smart farming* (pp. 1–33). Academic Press.

Delany, S., & Cunningham, P. (2004). An analysis of case-base editing in a spam filtering system. In *European conference on case-based reasoning, Vol. 3155* (pp. 128–141).

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*(1), 3–18.

Devroye, L., Györfi, L., & Lugosi, G. (1996). Condensed and edited nearest neighbor rules. In *A probabilistic theory of pattern recognition* (pp. 303–313). New York: Springer.

Douna, V., Barraza, V., Grings, F., Huete, A., Restrepo-Coupe, N., & Beringer, J. (2021). Towards a remote sensing data based evapotranspiration estimation in Northern Australia using a simple random forest approach. *Journal of Arid Environments*, *191*, Article 104513.

Garcia, L. P. F., Lorena, A. C., & de Carvalho, A. C. P. L. F. (2012). A study on class noise detection and elimination. In *2012 Brazilian symposium on neural networks* (pp. 13–18).

Garcia, L. P. F., Lorena, A. C., Matwin, S., & de Carvalho, A. C. P. L. F. (2016). Ensembles of label noise filters: A ranking approach. *Data Mining and Knowledge Discovery*, *30*(5), 1192–1216.

Gates, G. (1972). The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, *18*(3), 431–433.

Glaroudis, D., Iossifides, A., & Chatzimisios, P. (2020). Survey, comparison and research challenges of IoT application protocols for smart farming. *Computer Networks*, *168*, Article 107037.

Guillen, A., Herrera, L., Rubio, G., Pomares, H., Lendasse, A., & Rojas, I. (2010). New method for instance or prototype selection using mutual information in time series prediction. *Neurocomputing*, *73*(10–12), 2030–2038.

Gupta, S., & Gupta, A. (2019). Dealing with noise problem in machine learning data-sets: A systematic review. *Procedia Computer Science*, *161*, 466–474.

Hansch, R. (2018). *Handbook of random forests: Theory and applications for remote sensing*. Singapore: World Scientific.

Hulse, J., Khoshgoftaar, T., & Huang, H. (2007). The pairwise attribute noise detection algorithm. *Knowledge and Information Systems*, *11*(2), 171–190.

Jiang, G., Wang, W., Qian, Y., & Liang, J. (2021). A unified sample selection framework for output noise filtering: An error-bound perspective. *Journal of Machine Learning Research*, *22*(18), 1–66.

Khoshgoftaar, T., & Rebours, P. (2007). Improving software quality prediction by noise filtering techniques. *Journal of Computer Science and Technology*, *22*, 387–396.

Koplowitz, J., & Brown, T. (1981). On the relation of performance to editing in nearest neighbor rules. *Pattern Recognition*, *13*(3), 251–255.

Kordos, M., & Blachnik, M. (2012). Instance selection with neural networks for regression problems. In *Artificial neural networks and machine learning – ICANN 2012* (pp. 263–270).

Kumar, M., Bharti, B., Quilty, J., Adamowski, J., & Pandey, A. (2014). Modeling of daily pan evaporation in sub tropical climates using ANN, LS-SVR, Fuzzy Logic, and ANFIS. *Expert Systems with Applications*, *41*(11), 5267–5276.

Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. In *Artificial intelligence in medicine: Proc. 8th conference on artificial intelligence in medicine in Europe* (pp. 63–66).

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R News*, *2*(3), 18–22, URL: https://CRAN.R-project.org/package=randomForest.

Liu, Y., Dillon, T., Yu, W., Rahayu, W., & Mostafa, F. (2020). Noise removal in the presence of significant anomalies for industrial IoT sensor data in manufacturing. *IEEE Internet of Things Journal*, *7*(8), 7084–7096.

Martín, J., Sáez, J., & Corchado, E. (2021a). On the regressand noise problem: Model robustness and synergy with regression-adapted noise filters. *IEEE Access*, *9*, 145800–145816.

Martín, J., Sáez, J., & Corchado, E. (2021b). On the suitability of stacking-based ensembles in smart agriculture for evapotranspiration prediction. *Applied Soft Computing*, *108*, Article 107509.

Miranda, A., Garcia, L., Carvalho, A., & Lorena, A. (2009). Use of classification algorithms in noise detection and elimination. In *Hybrid artificial intelligence systems, Vol. 5572* (pp. 417–424). Springer Berlin Heidelberg.

Muthukumar, V., Vodrahalli, K., Subramanian, V., & Sahai, A. (2020). Harmless interpolation of noisy data in regression. *IEEE Journal on Selected Areas in Information Theory*, *1*(1), 67–83.

Nematzadeh, Z., Ibrahim, R., & Selamat, A. (2020). Improving class noise detection and classification performance: A new two-filter CNDC model. *Applied Soft Computing*, *94*, Article 106428.

Nettleton, D., Orriols-Puig, A., & Fornells, A. (2010). A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial Intelligence Review*, *33*, 275–306.

Pérez-Padillo, J., Puig, F., García, J., & Montesinos, P. (2022). IoT platform for failure management in water transmission systems. *Expert Systems with Applications*, *199*, Article 116974.

Puri, P., & Gupta, M. (2021). Knowledge discovery from noisy imbalanced and incomplete binary class data. *Expert Systems with Applications*, *181*, Article 115179.

R Core Team (2023). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing, URL: https://www.R-project.org/.

Rout, N., Mishra, D., & Mallick, M. (2023). Behaviour of imbalanced data in presence of borderline and noisy examples using hybrid SPIDER2-IPF boosting ensemble method. *Journal of Survey in Fisheries Sciences*, *10*(4S), 1685–1711.

Sáez, J. (2022). Noise models in classification: Unified nomenclature, extended taxonomy and pragmatic categorization. *Mathematics*, *10*(20), 3736.

Sáez, J. (2023). Noise simulation in classification with the noisemodel R package: Applications analyzing the impact of errors with chemical data. *Journal of Chemometrics*, *37*(5), Article e3472.

Sáez, J., & Corchado, E. (2022). ANCES: A novel method to repair attribute noise in classification problems. *Pattern Recognition*, *121*, Article 108198.

Sáez, J., Luengo, J., & Herrera, F. (2013). Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. *Pattern Recognition*, *46*(1), 355–364.

Sáez, J., Luengo, J., & Herrera, F. (2016). Evaluating the classifier behavior with noisy data considering performance and robustness: The Equalized Loss of Accuracy measure. *Neurocomputing*, *176*, 26–35.

Sáez, J., & Romero-Béjar, J. (2022). On the suitability of bagging-based ensembles with borderline label noise. *Mathematics*, *10*(11), 1892.

Stojanović, M., Božić, M., Stanković, M., & Stajić, Z. (2014). A methodology for training set instance selection using mutual information in time series prediction. *Neurocomputing*, *141*, 236–245.

Taunk, K., De, S., Verma, S., & Swetapadma, A. (2019). A brief review of nearest neighbor algorithm for learning and classification. In *2019 international conference on intelligent computing and control systems* (pp. 1255–1260).

Teh, H. Y., Kempa-Liehr, A. W., & Wang, K. I.-K. (2020). Sensor data quality: a systematic review. *Journal of Big Data*, *7*, 11.

Togneri, R., Felipe dos Santos, D., Camponogara, G., Nagano, H., Custódio, G., Prati, R., Fernandes, S., & Kamienski, C. (2022). Soil moisture forecast for smart irrigation: The primetime for machine learning. *Expert Systems with Applications*, *207*, Article 117653.

Togneri, R., Prati, R., Nagano, H., & Kamienski, C. (2023). Data-driven water need estimation for IoT-based smart irrigation: A survey. *Expert Systems with Applications*, *225*, Article 120194.

Tomek, I. (1976a). An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-6*(6), 448–452.

Tomek, I. (1976b). Two modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-6*(11), 769–772.

Verbaeten, S. (2002). Identifying mislabeled training examples in ILP classification problems. In *Proceedings of twelfth Belgian-Dutch conference on machine learning* (pp. 1–8).

Verbaeten, S., & Van Assche, A. (2003). Ensemble methods for noise elimination in classification problems. In *Multiple classifier systems: Proc. 4th international workshop, Vol. 2709* (pp. 317–325). Springer Berlin Heidelberg.

Wang, D., Borthwick, A. G., He, H., Wang, Y., Zhu, J., Lu, Y., Xu, P., Zeng, X., Wu, J., Wang, L., Zou, X., Liu, J., Zou, Y., & He, R. (2018). A hybrid wavelet de-noising and Rank-Set Pair Analysis approach for forecasting hydro-meteorological time series. *Environmental Research*, *160*, 269–281.

Wang, Y., Pan, Z., & Pan, Y. (2020). A training data set cleaning method by classification ability ranking for the k-nearest neighbor classifier. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(5), 1544–1556.

Wang, D., Singh, V. P., Shang, X., Ding, H., Wu, J., Wang, L., Zou, X., Chen, Y., Chen, X., Wang, S., & Wang, Z. (2014). Sample entropy-based adaptive wavelet de-noising approach for meteorologic and hydrologic time series. *Journal of Geophysical Research: Atmospheres*, *119*(14), 8726–8740.

Wickham, H., Danenberg, P., Csárdi, G., & Eugster, M. (2021). roxygen2: In-Line documentation for R. URL: https://CRAN.R-project.org/package=roxygen2 - R package version 7.1.2.

Wong, T., & Yeh, P. (2020). Reliable accuracy estimates from k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, *32*(8), 1586–1594.

Zheng, W., & Jin, M. (2020). fmf: Fast class noise detector with multi-factor-based learning. URL: https://CRAN.R-project.org/package=fmf - R package version 1.1.1.

Zhu, X., & Wu, X. (2004). Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, *22*(3), 177–210.