Article

Regression Models for Symbolic Interval-Valued Variables

Jose Emmanuel Chacón¹ and Oldemar Rodríguez^{2,*}¹ National Bank of Costa Rica, San José 11501-2060, Costa Rica; jechaconch@bncr.fi.cr² School of Mathematics, CIMPA, University of Costa Rica, San José 11501-2060, Costa Rica

* Correspondence: oldemar.rodriguez@ucr.ac.cr; Tel.: +506-2511-6606

Abstract: This paper presents new approaches to fit regression models for symbolic interval-valued variables, which are shown to improve and extend the center method suggested by Billard and Diday and the center and range method proposed by Lima-Neto, E.A. and De Carvalho, F.A.T. Like the previously mentioned methods, the proposed regression models consider the midpoints and half of the length of the intervals as additional variables. We considered various methods to fit the regression models, including tree-based models, K-nearest neighbors, support vector machines, and neural networks. The approaches proposed in this paper were applied to a real dataset and to synthetic datasets generated with linear and nonlinear relations. For an evaluation of the methods, the root-mean-squared error and the correlation coefficient were used. The methods presented herein are available in the the RSDA package written in the R language, which can be installed from CRAN.

Keywords: regression; decision trees; random forest; boosting; K-nearest neighbors; support vector machines; neural networks; interval-valued variables; symbolic data analysis

Citation: Chacón, J.E.; Rodríguez, O. Regression Models for Symbolic Interval-Valued Variables. *Entropy* **2021**, *23*, 429. <https://doi.org/10.3390/e23040429>

Academic Editor: Nikola Poljak

Received: 15 February 2021

Accepted: 30 March 2021

Published: 6 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Statistical and data mining methods have been developed mainly in cases in which variables take a single value. In real life, there are many situations in which the use of these type of variables can lead to an important loss of information or result in time-consuming calculations. In the case of quantitative variables, a more complete description can be achieved by describing an ensemble of statistical units in terms of interval data, that is the value taken by a variable is a closed interval in the real numbers.

It is especially useful when it is convenient to summarize large datasets in such a way that the resulting summary is of a manageable size and still maintains as much information as possible from the original dataset. For example, suppose we want to substitute the information of all transactions made by the owner of a credit card with a unique “transaction” summarizing all the original transactions. This is achieved thanks to the fact that this new transaction will have in its fields not only numbers, but also intervals defined by, for example, the minimum and maximum purchase.

The statistical treatment of interval-valued data was considered in the context of symbolic data analysis (SDA) introduced by E. Diday in [1], the objective of which is to extend the classic statistical methods to the study of more complex data structures that include, among others, interval-valued variables. A complete presentation on symbolic data analysis can be found in [2–4].

Research on SDA has focused primarily on unsupervised learning, with few contributions in the field of regression, which have been made mainly based on linear models. In this work, we explore nonlinear regression algorithms in conjunction with the center method and the center and range method for interval-valued data to try to improve the results involving classical linear regression for the center and center and range methods proposed in [5,6], respectively, and the extended lasso and ridge regression for interval-value data proposed in [7]. In this way, we are able to study how more sophisticated algorithms can improve the traditional models based on linear methods.

Furthermore, we extend the tool kit of algorithms available for regression on interval-valued data, taking advantage of the properties and the large number of algorithms for real-valued data. We explore the classical linear regression models, tree-based regression models (regression trees, random forest, and boosting), K-nearest neighbors regression, support vector machines regression, and regression using neural networks.

Section 2 gives a summarized presentation of the different regression models for real-valued data. Section 3 presents a summary of the center method and the center and range methods in the context of each regression model considered. Section 4 presents an experimental evaluation with a real dataset and simulated datasets, which evidences the differences and improvements among the models. Finally, Section 5 gives the concluding remarks based on the results.

2. Regression Methods

In this section, we provide a short review of the main regression methods and their mathematical formulation.

2.1. Classical Linear Regression Methods

We present only a summary of the classical linear regression model, while a complete presentation can be found in [8,9]. Linear regression models for decades have been some of the most important predictive methods in statistics; in fact, this continues today as one of the most important tools in statistics and data mining. The idea is, given an input vector $x^t = (x_1, x_2, \dots, x_p)$, where p is the number of variables and x^t represents the transpose of x , we want to predict the response variable y through the following linear model:

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^p x_j \hat{\beta}_j, \quad (1)$$

where β_0 is called the intercept. If a constant one is included with vector x and β_0 in the coefficients vector β , the linear model can be written in vectorial form as a product as follows:

$$\hat{y} = x^t \hat{\beta}, \quad (2)$$

To fit the linear model in the training data, the most popular estimation method is least squares. In this approach, we pick the coefficients β to minimize the residual sum of squares (RSS):

$$\text{RSS}(\beta) = \sum_{i=1}^n (y_i - x_i^t \beta)^2. \quad (3)$$

where n is the number of observations in the dataset. $\text{RSS}(\beta)$ is a quadric function; therefore, its minimum always exists. Note that it can be written as:

$$\text{RSS}(\beta) = (y - X\beta)^t (y - X\beta), \quad (4)$$

where X is an $n \times p$ matrix wherein each row is a vector in the training dataset; y is an n size vector (the output vector in the training dataset). It is well known that, if $X^t X$ is a nonsingular matrix, the solution is given by:

$$\hat{\beta} = (X^t X)^{-1} X^t y. \quad (5)$$

The approximate value by this model for the component x_i can be estimated as $\hat{y}_i = x_i^t \hat{\beta}$, and the fitted values for a new case $x^t = (1, x_1, \dots, x_p)$ are given by $\hat{y} = x^t \hat{\beta}$.

In practice, there are various methods to find the coefficients $\hat{\beta}$, but the existing methods, and in particular the least squares method, are labor intensive and time consuming with large datasets, and others are not accurate enough with these kinds of datasets. A new non-iterative algorithm for identifying multiple regression coefficients based on the

SGTMneural-like structure for the case of large volumes of data processing was proposed in [10], where the high efficiency of the method for the accuracy and speed in comparison with the existing methods was established.

Extensions to this model include shrinkage linear regression models, such as the ridge and lasso models. These models impose different types of regularization on the parameters: L2 regularization, used in ridge, and L1, used in lasso. A complete presentation of these models can be found in [8,11].

2.2. Tree-Based Regression Models

We now present a summary of the three main tree-based regression models. A complete presentation of these models can be found in [8].

2.2.1. Regression Trees

There are two main steps for regression using decision trees: First, we begin by dividing the predictor space in $\mathbb{R}^n$ into J non-overlapping regions $R_1, \dots, R_J$; second, for every testing example that falls in R_j , we predict the response variable as the mean of the response variable over the training examples in R_j .

The regions R_j could be of any geometrical shape, but for the sake of simplicity and interpretation, we restrict ourselves only to rectangular regions (boxes). Therefore, we are searching for a partition of the predictor space into boxes R_j such that they minimize the RSS, which can be written as:

$$RSS = \sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2.$$

It is computationally unfeasible to consider every possible partition of the predictor space; for this reason, the method uses top-down and greedy recursive binary splitting. It is binary because every split of a predictor variable results in a division of the predictor space into two sets; it is top-down because it starts building the tree from the top to the leaves; and it is greedy because, at each step, the best split of the predictor is made without looking ahead and without picking a split that will lead to a lower RSS in some future step.

To construct the tree, first, we consider all the predictor variables and all the possible binary splits for their values. If the variable is numerical, all the possible values s in the range values of the variable are reconsidered, and if the variable is categorical, we consider all possible partitions in two sets of the values of the variable. We then select the variable and the split of that variable that leads to the greatest reduction in RSS. This produces two regions corresponding to two branches of the tree. We then continue looking for the best predictor and the best partition on each of the resulting regions, and the process ends when a stopping criterion is reached. It is common to build a large, complex tree and then prune it in order to reduce over fitting.

Once the regions $R_1, \dots, R_J$ have been created, we can use it to predict the response for new examples as the mean of the response of the training observations in the region to which the new example belongs.

2.2.2. Random Forest

The idea is to build a given number of regression trees on bootstrap sets (that is, obtain distinct datasets by repeatedly sampling with replacement observations from the original dataset) and use, in each tree, a random subset of m of the original predictor variables. Usually, m is taken to be $m = \sqrt{p}$, where p is the total number of variables.

In this case, the prediction of a new example is the mean over the predictions of the individual trees. The idea of random forest is to decorrelate the trees, thereby making the average of the resulting trees less variable.

2.2.3. Boosting

In this method, the idea is to sequentially construct trees to repeatedly modify versions of the training data and the loss function, thereby producing a sequence of regression models, $G_j(x)$, whose predictions are then combined and weighted according to the error they produce.

Initially, all the N training examples (x_i, y_i) have the same weights $w_i = \frac{1}{N}$. In each following step of the training process, the data are modified by updating these weights w_i . At step m , those observations with a higher error by the model $G_{m-1}(x)$ induced at the previous step have their weights increased, whereas the weights are decreased for those that have a lower error, and those weights in turn are taken into account by the loss function. As a result of this weight actualization, each successive model in the process is then forced to concentrate on those training observations that present higher errors by previous models in the sequence.

2.3. K-Nearest Neighbors

Given a number K and a testing example x_0 , we identify the set $\mathcal{N}_0$ of the nearest K training examples to x_0 . The prediction $\hat{y}_0$ of the response variable for x_0 is the mean of the response variable of the examples in $\mathcal{N}_0$, that is:

$$\hat{y}_0 = \frac{1}{K} \sum_{i \in \mathcal{N}_0} y_i.$$

We can use any distance between examples, but it is recommended that the distance that minimizes the testing RSS is used. To select the appropriate number of neighbors K , it is recommended that cross-validation be used to compare the RSS of the resulting models, using $K = 1, \dots, \sqrt{n}$, where n is the total number of examples.

More details on the K-nearest neighbors regression model were presented in [8].

2.4. Support Vector Machines

The support vector machines model for regression is an extension of the linear regression model that uses a metric that is less sensitive to outliers using a fixed margin to ignore the errors that are within this margin by means of a ϵ -insensitive function in the loss function and defines a hyperplane that is meant to adjust the data and define the prediction.

Given a threshold ϵ , the idea is to define a margin such that examples with residues within the margin do not contribute to the regression fitting, while residue examples outside the margin contribute proportionally to their magnitude. Therefore, the outlier observations have a limited effect, and the examples in which the model fits well do not have an effect on the model.

The loss function of this model is given by:

$$LF(\beta) = C \sum_{i=1}^n L_{\epsilon}(y_i - \hat{y}_i) + \sum_{j=1}^p \beta_j^2,$$

where C is the cost penalty, which penalizes large residuals, $\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}$,

and $L_{\epsilon}(\xi) = \begin{cases} |\xi| - \epsilon & \text{si } |\xi| > \epsilon \\ 0 & \text{si } |\xi| \leq \epsilon \end{cases}$ is the ϵ -insensitive function. We search for parameters $\hat{\beta}_j$ that minimize $LF(\beta)$.

A complete presentation of this model can be found in [8].

2.5. Neural Networks

For a neural network model, the function f that approximates the true relation of the data $y = f^*(x)$ takes the form of a composition $f(x) = (f^{(n)} \circ f^{(n-1)} \circ \dots \circ f^{(1)})(x)$. In this

chain structure, $f^{(i)}$ is called the i -th layer of the network, n is the depth of the network, and $f^{(n)}$ is the output of the network (which in the regression setup is a real-valued function), and the other layers are called hidden layers and are typically vector-valued functions. The neural network model is associated with a directed acyclic graph describing how the functions are composed together, and the idea of using many layers of vector-valued representations is that each one can learn distinct specific patterns in the data.

The training examples specify directly what the output layer must do at each point x ; that is, it must produce a value that is close to the true value y . The behavior of the hidden layers is not directly specified by the training data; instead, the learning algorithm must decide how to use these layers to best implement an approximation of the true value y .

Neural networks are usually trained using stochastic gradient descent, which involves computing the gradients of complicated functions, and the back-propagation algorithm is used to efficiently compute these gradients.

Full details of the mathematical formulation of the neural network model can be found in [8,12].

3. Regression Models for Symbolic Interval-Valued Variables

In this section, we summarize the center method and the center and range method; a complete presentation of which can be found in [5,6,13–15], respectively. We also propose an approach to the center method and the center and range method in the context of the other regression models considered.

3.1. Center Method

In the center method, the β parameters are estimated based on the interval's midpoints. In this method, there are predictors $X_1, \dots, X_p$ and a response to be predicted Y , all of which are interval valued. Therefore, X is an $n \times p$ matrix, where each row is a vector of components of the training dataset $x_i = (x_{i1}, \dots, x_{ip})$ with $x_{ij} = [a_{ij}, b_{ij}]$, and each component of the Y variable is also an interval $y_i = [y_{Li}, y_{Ui}]$.

We denote by X^c the matrix with the interval's midpoints of the matrix X , that is $x_{ij}^c = (a_{ij} + b_{ij})/2$, and we denote by $y_i^c = (y_{Li} + y_{Ui})/2$ the midpoints of Y . The idea of the center method is to fit a linear regression model over $X^c = ((x_1^c)^t, \dots, (x_n^c)^t)^t$ with $(x_i^c)^t = (1, x_{i1}^c, \dots, x_{ip}^c)$ for $i = 1, \dots, n$ and $y^c = (y_1^c, \dots, y_n^c)^t$. If $(X^c)^t X^c$ is nonsingular from (5), we know that the unique solution for β is given by:

$$\hat{\beta} = ((X^c)^t X^c)^{-1} (X^c)^t y^c. \quad (6)$$

The value of the prediction for $y = [y_L, y_U]$ for a new case $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ is estimated as follows:

$$\hat{y}_L = (x_L)^t \hat{\beta} \quad \text{and} \quad \hat{y}_U = (x_U)^t \hat{\beta}, \quad (7)$$

where $(x_L)^t = (1, a_1, \dots, a_p)$ and $(x_U)^t = (1, b_1, \dots, b_p)$.

3.2. Center and Range Method

With the center and range method, Lima Neto and De Carvalho fit the linear regression model for interval-valued variables by using the information contained in the midpoints and in the interval ranges, in order to improve the quality of the prediction of the center method. The idea is to fit two regression models, the first with the midpoint of the interval and the second with the ranges of those same intervals. Just like the center method, there are $X_1, \dots, X_p$ predictors and a response Y , and all these variables are interval valued. Therefore, X is an $n \times p$ matrix, where each row is a vector of a component of the training dataset $x_i = (x_{i1}, \dots, x_{ip})$ with $x_{ij} = [a_{ij}, b_{ij}]$, and each component of the variable Y is also an interval $y_i = [y_{Li}, y_{Ui}]$.

To fit the first regression model, we proceed in the same way as in the center method, that is if we denote by X^c the midpoint's matrix ($x_{ij}^c = (a_{ij} + b_{ij})/2$) and we denote by

$y_i^c = (y_{Li} + y_{Ui})/2$ the midpoints of Y , the center and range method fits a first linear regression model over $X^c = ((x_1^c)^t, \dots, (x_n^c)^t)^t$ with $(x_i^c)^t = (1, x_{i1}^c, \dots, x_{ip}^c)^t$ for $i = 1, \dots, n$ and $y^c = (y_1^c, \dots, y_n^c)^t$. In this case, if $(X^c)^t X^c$ is nonsingular, then we know that the unique solution for β^c is given by:

$$\hat{\beta}^c = ((X^c)^t X^c)^{-1} (X^c)^t y^c. \quad (8)$$

To fit the second regression model, half of the value of the range of each interval is used. For this, we denote by X^r the matrix that contains in each component half of the interval ranges of the matrix X , i.e., $x_{ij}^r = (b_{ij} - a_{ij})/2$, and we denote by $y_i^r = (y_{Ui} - y_{Li})/2$ half of the interval-valued variable Y . The center and range method then fits a second linear regression model over $X^r = ((x_1^r)^t, \dots, (x_n^r)^t)^t$ with $(x_i^r)^t = (1, x_{i1}^r, \dots, x_{ip}^r)^t$ for $i = 1, \dots, n$ and $y^r = (y_1^r, \dots, y_n^r)^t$. In this case, if $(X^r)^t X^r$ is nonsingular from Equation (5), we know that the solution for β^r is given by:

$$\hat{\beta}^r = ((X^r)^t X^r)^{-1} (X^r)^t y^r, \quad (9)$$

so each case in the training dataset is represented by two vectors $w_i = (x_i^c, y_i^c)$ and $r_i = (x_i^r, y_i^r)$ for $i = 1, \dots, n$. The prediction value for $y = [y_L, y_U]$ for a new case $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ is then estimated as follows:

$$\hat{y}_L = \hat{y}^c - \hat{y}^r \quad \text{and} \quad \hat{y}_U = \hat{y}^c + \hat{y}^r, \quad (10)$$

with:

$$\hat{y}^c = (x^c)^t \hat{\beta}^c \quad \text{and} \quad \hat{y}^r = (x^r)^t \hat{\beta}^r, \quad (11)$$

where $(x^c)^t = (1, x_1^c, \dots, x_p^c)$ and $(x^r)^t = (1, x_1^r, \dots, x_p^r)$.

This model cannot mathematically guarantee that $\hat{y}_{Li} \leq \hat{y}_{Ui}$ for all $i = 1, \dots, n$, a problem addressed by Lima Neto and De Carvalho in [6].

Extensions to this model include shrinkage linear regression models for symbolic interval-valued data, which involve a generalization of the ridge and lasso models for interval-valued data. A complete presentation of these models can be found in [7].

3.3. Tree-Based Regression for Symbolic Interval-Valued Variables

3.3.1. Regression Trees Center Method

We begin by dividing the predictor space of interval midpoints in $\mathbb{R}^n$ into J non-overlapping regions $R_1^c, \dots, R_J^c$.

We search for a partition of the predictor space into boxes R_j^c such that they minimize RSS^c , which can be written as:

$$RSS^c = \sum_{j=1}^J \sum_{i \in R_j^c} (y_i^c - \hat{y}_{R_j^c}^c)^2.$$

To construct the tree, first, we consider all the predictor variables and all the possible binary splits for their values. If the variable is numerical, all the possible values s in the range values of the variable are reconsidered, and if the variable is categorical, we consider all possible partitions in two sets of the values of the variable. We then select the variable and the split of that variable, which leads to the greatest reduction in RSS^c . This produces two regions corresponding to two branches of the tree. We then continue looking for the best predictor and the best partition on each of the resulting regions, and the process ends when a stopping criterion is reached.

Once the regions $R_1^c, \dots, R_J^c$ have been created, the model can be used to predict the response of a new example $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ as:

$$\hat{y}_L = \sum_{j=1}^J c_j 1_{\{x_L \in R_j^c\}} \quad \text{and} \quad \hat{y}_U = \sum_{j=1}^J c_j 1_{\{x_U \in R_j^c\}},$$

where c_j is the mean of the response centers of the training observations in R_j^c , $x_L = (a_1, \dots, a_p)$ and $x_U = (b_1, \dots, b_p)$.

3.3.2. Regression Tree Center and Range Method

We begin by dividing the predictor space of interval midpoints in $\mathbb{R}^n$ into J non-overlapping regions $R_1^c, \dots, R_J^c$, and by dividing the predictor space of interval ranges in $\mathbb{R}^n$ into L non-overlapping regions $R_1^r, \dots, R_L^r$.

In this case, we are searching for a partition of the predictor space of centers and ranges into boxes R_j^c and R_j^r such that they minimize the RSS^c and RSS^r , respectively, which can be written as:

$$RSS^c = \sum_{j=1}^J \sum_{i \in R_j^c} (y_i^c - \hat{y}_{R_j^c}^c)^2 \quad \text{and} \quad RSS^r = \sum_{j=1}^L \sum_{i \in R_j^r} (y_i^r - \hat{y}_{R_j^r}^r)^2.$$

To construct the tree, first, we consider all the predictor variables and all the possible binary splits for their values. If the variable is numerical, all the possible values s in the range values of the variable are reconsidered, and if the variable is categorical, we consider all possible partitions in two sets of the values of the variable. We then select the variable and the split of that variable, which leads to the greatest reduction in RSS^c and RSS^r . This produces two regions corresponding to two branches of the tree. We then continue looking for the best predictor and the best partition on each of the resulting regions, and the process ends when a stopping criterion is reached.

Once the regions $R_1^c, \dots, R_J^c$, and $R_1^r, \dots, R_L^r$, have been created, the model can be used to predict the response of a new example $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ as:

$$\hat{y}_L = \hat{y}^c - \hat{y}^r \quad \text{and} \quad \hat{y}_U = \hat{y}^c + \hat{y}^r,$$

with:

$$\hat{y}^c = \sum_{j=1}^J c_j^c 1_{\{x^c \in R_j^c\}} \quad \text{and} \quad \hat{y}^r = \sum_{j=1}^L c_j^r 1_{\{x^r \in R_j^r\}},$$

where c_j^c and c_j^r are the means of the response centers and ranges of the training observations in R_j^c and R_j^r , respectively.

3.3.3. Random Forest Center Method

In this method, the idea is to build a given number M of regression trees, T_j^c , on bootstrap sets of the center data, using in each tree a random subset of m of the original predictor variables in X^c .

In this case, the prediction of a new example $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ is the mean over the predictions of the individual trees:

$$\hat{y}_L = \frac{1}{M} \sum_{j=1}^M T_j^c(x_L) \quad \text{and} \quad \hat{y}_U = \frac{1}{M} \sum_{j=1}^M T_j^c(x_U),$$

where $x_L = (a_1, \dots, a_p)$ and $x_U = (b_1, \dots, b_p)$.

3.3.4. Random Forest Center and Range Method

With this method, the idea is to build a given number M of regression trees, T_j^c , on bootstrap sets of the center data and a given number L of regression trees, T_j^r , on bootstrap

sets of the range data for each tree, using in each tree a random subset of m of the original predictor variables in X^c and X^r .

In this case, the prediction of a new example $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ is given by:

$$\hat{y}_L = \hat{y}^c - \hat{y}^r \quad \text{and} \quad \hat{y}_U = \hat{y}^c + \hat{y}^r,$$

with

$$\hat{y}^c = \frac{1}{M} \sum_{j=1}^M T_j^c(x^c) \quad \text{and} \quad \hat{y}^r = \frac{1}{L} \sum_{j=1}^L T_j^r(x^r).$$

3.3.5. Boosting Center Method

We construct trees sequentially and repeatedly modify versions of the training center data, thereby producing a sequence of regression models, G_j^c , whose predictions are then combined and weighted according to the error they produce.

Initially, all the N training examples (x_i^c, y_i^c) have the same weights $w_i^c = \frac{1}{N}$. On each following step of the training process, the data are modified by updating these weights w_i^c . At step m , those observations with a higher error by the model $G_{m-1}^c(x)$ induced at the previous step have their weights increased, whereas the weights are decreased for those that have a lower error.

In this case, the prediction of a new example $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ is the mean over the predictions of the individual trees:

$$\hat{y}_L = \sum_{j=1}^M \alpha_j^c G_j^c(x_L) \quad \text{and} \quad \hat{y}_U = \sum_{j=1}^L \alpha_j^c G_j^c(x_U),$$

where α_j^c measures the error of the j -th model, $x_L = (a_1, \dots, a_p)$ and $x_U = (b_1, \dots, b_p)$.

3.3.6. Boosting Center and Range Method

We construct trees sequentially and repeatedly modify versions of the training center and range data, thereby producing two sequences of regression models, G_j^c and G_j^r , whose predictions are then combined and weighted according to the error they produce.

Initially, all the N center training examples (x_i^c, y_i^c) have the same weights $w_i^c = \frac{1}{N}$, and the same applies to the range training examples. In each following step of the training process, the data are modified by updating these weights w_i^c and w_i^r . At step m , those observations with a higher error by the model $G_{m-1}^c(x)$ and $G_{m-1}^r(x)$ induced at the previous step have their weights increased, whereas the weights are decreased for those that have a lower error.

In this case, the prediction of a new example $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ is given by:

$$\hat{y}_L = \hat{y}^c - \hat{y}^r \quad \text{and} \quad \hat{y}_U = \hat{y}^c + \hat{y}^r,$$

with:

$$\hat{y}^c = \sum_{j=1}^M \alpha_j^c G_j^c(x^c) \quad \text{and} \quad \hat{y}^r = \sum_{j=1}^M \alpha_j^r G_j^r(x^r),$$

where α_m^c measures the error of the j -th center model and α_j^r measures the error of the j -th range model.

3.4. K-Nearest Neighbors Center Method

Given a number K and a testing example $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ in the symbolic dataset, we identify the sets $\mathcal{N}_0$ of the nearest K training examples to x^c . The prediction $\hat{y}$ of the response variable for x is given by:

$$\hat{y}_L = \frac{1}{K} \sum_{x_i \in \mathcal{N}_0} (y_i)_L \quad \text{and} \quad \hat{y}_U = \frac{1}{K} \sum_{x_i \in \mathcal{N}_0} (y_i)_U.$$

3.5. K-Nearest Neighbors Center and Range Method

Given the numbers K^c and K^r and a testing example x in the symbolic dataset, we identify the sets $\mathcal{N}^c$ and $\mathcal{N}^r$ of the nearest K^c and K^r training examples to x^c and x^r , respectively. The prediction $\hat{y}$ of the response variable for x is given by:

$$\hat{y}_L = \hat{y}^c - \hat{y}^r \quad \text{and} \quad \hat{y}_U = \hat{y}^c + \hat{y}^r,$$

with:

$$\hat{y}^c = \frac{1}{K^c} \sum_{x_i^c \in \mathcal{N}^c} y_i^c \quad \text{and} \quad \hat{y}^r = \frac{1}{K^r} \sum_{x_i^r \in \mathcal{N}^r} y_i^r.$$

3.6. Support Vector Machines Center Method

Given a threshold ϵ^c , the idea is to define a margin such that examples with residues within the margin do not contribute to the regression fitting, while residue examples outside the margin contribute proportionally to their magnitude.

The loss function of this model is given by:

$$LF^c(\beta) = C \sum_{i=1}^n L_\epsilon(y_i^c - \hat{y}_i^c) + \sum_{j=1}^p \beta_j^2,$$

where:

$$L_\epsilon(\xi) = \begin{cases} |\xi| - \epsilon^c & \text{si } |\xi| > \epsilon^c \\ 0 & \text{si } |\xi| \leq \epsilon^c. \end{cases}$$

We search for parameters $\hat{\beta}^c$ that minimize $LF(\beta)$, so the value of the prediction for $y = [y_L, y_U]$ for a new case $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ is estimated as follows:

$$\hat{y}_L = (x_L)^t \hat{\beta}^c \quad \text{and} \quad \hat{y}_U = (x_U)^t \hat{\beta}^c,$$

where $(x_L)^t = (1, a_1, \dots, a_p)$ and $(x_U)^t = (1, b_1, \dots, b_p)$.

3.7. Support Vector Machines Center and Range Method

Given thresholds ϵ^c and ϵ^r , we define margins such that examples with residues within these margin do not contribute to regression fitting, while residue examples outside the margins contribute proportionally to their magnitude.

The loss function of the center and ranges models is given by:

$$LF^c(\beta) = C \sum_{i=1}^n L_{\epsilon^c}(y_i^c - \hat{y}_i^c) + \sum_{j=1}^p \beta_j^2 \quad \text{and} \quad LF^r(\beta) = C \sum_{i=1}^n L_{\epsilon^r}(y_i^r - \hat{y}_i^r) + \sum_{j=1}^p \beta_j^2,$$

where:

$$L_\epsilon(\xi) = \begin{cases} |\xi| - \epsilon & \text{si } |\xi| > \epsilon \\ 0 & \text{si } |\xi| \leq \epsilon. \end{cases}$$

We search parameters $\hat{\beta}^c$ that minimize $LF(\beta)$, so the value of the prediction for $y = [y_L, y_U]$ for a new case $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ is estimated as:

$$\hat{y}_L = \hat{y}^c - \hat{y}^r \quad \text{and} \quad \hat{y}_U = \hat{y}^c + \hat{y}^r,$$

with:

$$\hat{y}^c = (x^c)^t \hat{\beta}^c \quad \text{and} \quad \hat{y}^r = (x^r)^t \hat{\beta}^r,$$

where $(x^c)^t = (1, x_1^c, \dots, x_p^c)$ and $(x^r)^t = (1, x_1^r, \dots, x_p^r)$.

3.8. Neural Networks Center Method

We consider a neural network model on the center data, and the function f^c that approximates the true relation of these data $y = f^c(x)$ takes the form of a composition $f^c(x) = (f_n^c \circ f_{n-1}^c \circ \dots \circ f_1^c)(x)$.

In this case, the prediction of a new example $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ is given by:

$$\hat{y}_L = (f_n^c \circ f_{n-1}^c \circ \dots \circ f_1^c)(x_L) \quad \text{and} \quad \hat{y}_U = (f_n^c \circ f_{n-1}^c \circ \dots \circ f_1^c)(x_U),$$

where $x_L = (a_1, \dots, a_p)$ and $x_U = (b_1, \dots, b_p)$.

3.9. Neural Networks Center and Range Method

We consider two neural networks models, one for the center data and another for the range data. The function f^c that approximates the true relation of these data $y = f^c(x)$ takes the form of a composition $f^c(x) = (f_n^c \circ f_{n-1}^c \circ \dots \circ f_1^c)(x)$.

In this case, the prediction of a new example $x = (x_1, \dots, x_p)$ with $x_j = [a_j, b_j]$ is given by:

$$\hat{y}_L = \hat{y}^c - \hat{y}^r \quad \text{and} \quad \hat{y}_U = \hat{y}^c + \hat{y}^r,$$

with:

$$\hat{y}^c = (f_n^c \circ f_{n-1}^c \circ \dots \circ f_1^c)(x^c) \quad \text{and} \quad \hat{y}^r = (f_n^r \circ f_{n-1}^r \circ \dots \circ f_1^r)(x^r).$$

4. Experimental Evaluation

As done by Lima Neto and De Carvalho in [6], the evaluation of the results of these interval-valued regression models was carried out using the following metrics: the lower boundary root-mean-squared-error $RMSE_L$, the upper boundary root-mean-squared-error $RMSE_U$, the square of the lower boundary correlation coefficient r_L^2 , and the square of the upper boundary correlation coefficient r_U^2 .

$$RMSE_L = \sqrt{\frac{\sum_{i=1}^n (y_{Li} - \hat{y}_{Li})^2}{n}} \quad \text{and} \quad RMSE_U = \sqrt{\frac{\sum_{i=1}^n (y_{Ui} - \hat{y}_{Ui})^2}{n}}, \quad (12)$$

$$r_L^2 = \left(\frac{\text{Cov}(y_L, \hat{y}_L)}{S_{y_L} S_{\hat{y}_L}} \right)^2 \quad \text{and} \quad r_U^2 = \left(\frac{\text{Cov}(y_U, \hat{y}_U)}{S_{y_U} S_{\hat{y}_U}} \right)^2, \quad (13)$$

where $y_i = [y_{Li}, y_{Ui}]$ and its corresponding prediction is $\hat{y}_i = [\hat{y}_{Li}, \hat{y}_{Ui}]$, $y_L = (y_{L1}, \dots, y_{Ln})^t$, $\hat{y}_L = (\hat{y}_{L1}, \dots, \hat{y}_{Ln})^t$, $y_U = (y_{U1}, \dots, y_{Un})^t$, $\hat{y}_U = (\hat{y}_{U1}, \dots, \hat{y}_{Un})^t$; as is usual, $\text{Cov}(\Psi, \Phi)$ denotes the covariance among variables Ψ and Φ ; S_Ψ denotes the standard deviation of variable Ψ .

For the experimental evaluation, we used the following hyperparameters for the models.

- Lasso and ridge: folds used in K-fold cross-validation to find the best tuning parameter λ : 10
- Regression trees: minimum number of observations in a node: 20; maximum depth of any node of the final tree: 10.
- Random forest: number of trees to grow: 500; number of variables randomly sampled at each split: $\frac{p}{3}$, where p is the total number of variables.
- Boosting: number of trees to grow: 500; shrinkage parameter applied to each tree in the expansion: 0.1; highest level of variable interactions allowed: 1.
- K-nearest neighbors: maximum number of neighbors considered: 20; kernel to use: triangular.
- Support vector machines: kernel to use: radial.
- Neural networks: number of layers: 1; number of neurons: 10; threshold as stopping criteria: 0.05; maximum steps for the training process: 10^5 .

All examples presented in this section were developed using the package RSDA (R for symbolic data analysis) constructed by the authors of this paper for applications in symbolic data analysis (see [16]). The datasets used in this section are also available in the RSDA package. A reproducible document with all the R code for the examples can be downloaded from <http://www.oldemarrodriguez.com/publicaciones.html>, accessed on 29 March 2021.

4.1. Cardiological Interval Dataset

To illustrate the application of the methods, we considered the data based on [17] and taken from [3], shown in Table 1, in which the pulse rate (Pulse), systolic blood pressure (Sys), diastolic blood pressure (Diast), Art1 and Art2 were recorded as an interval for each of the patients, where Art1 and Art2 are artificial variables added to the data table. The goal was to predict (Pulse). The dataset consisted of 44 rows and 5 attributes. Table 1 presents a glimpse of the dataset. To measure the performance of the models, we split the dataset into training and testing sets, using 70% and 30%, respectively.

Table 1. Cardiological interval dataset. Syst, systolic; Diast, diastolic; Art, artificial.

	Pulse	Syst	Diast	Art1	Art2
1	[44, 68]	[90, 100]	[50, 70]	[6, 9]	[1, 6]
2	[60, 72]	[90, 130]	[70, 90]	[2, 9]	[4, 6]
3	[56, 90]	[140, 180]	[90, 100]	[9, 10]	[4, 7]
4	[70, 112]	[110, 142]	[80, 108]	[8, 9]	[5, 5]
5	[54, 72]	[90, 100]	[50, 70]	[5, 7]	[6, 9]
⋮	⋮	⋮	⋮	⋮	⋮
44	[65, 67]	[90, 150]	[78, 90]	[8, 9]	[7, 9]

After closer inspection of Table 2, it can be seen that the neural network had the best metrics for $RMSE_L$, r_L^2 , and r_U^2 , and had the second-best $RMSE_U$, the random forest center and range model being the best one. As a result, the neural network model was best for this dataset.

As mentioned before, the neural network center and range model consist of 2 neural networks with 1 layer and 10 neurons, and this selection was made to avoid overfitting the small dataset. This simple neural network model was able to capture the relation in the data better than the other models, and there was a large difference in performance, even when compared with the second best.

Table 2. Performance of the methods in the cardiological interval dataset.

Method	RMSE _L	RMSE _U	r_L^2	r_U^2
LM CM	15.8464	13.7108	0.1744	0.2238
Ridge CM	16.8055	13.5504	0.2234	0.2456
Lasso CM	17.2616	13.9484	0.0358	0.1652
RT CM	18.6783	15.9769	0.0055	0.0765
RF CM	15.0495	13.9372	0.3548	0.2406
KNN CM	16.2914	12.9731	0.3370	0.4125
SVM CM	15.5756	13.4378	0.4358	0.4307
NNet CM	9.2535	13.9276	0.6602	0.5340
LM CRM	12.3913	13.5961	0.2989	0.2191
Ridge CRM	12.5081	13.0821	0.3689	0.2368
Lasso CRM	13.5452	13.8853	0.1654	0.1381
RT CRM	12.7732	13.6137	0.2623	0.1730
RF CRM	7.6566	7.3574	0.7988	0.8172
KNN CRM	9.8981	10.7309	0.5889	0.4518
SVM CRM	11.9789	12.5868	0.3324	0.2728
NNet CRM	5.6585	7.4140	0.9270	0.9010

CM: center method. CRM: center and range method. LM: linear regression. Ridge: ridge regression. Lasso: lasso regression. RT: regression trees. RF: random forest. KNN: K-nearest neighbors. SVM: support vector machines. NNet: neural networks.

In the previous table, the boosting method was omitted because the table is too short for the model to be fitted.

4.2. Monte Carlo Experiments and Applications

The usefulness of the methods proposed in this paper was evaluated through experiments with synthetic interval-valued datasets with different linear and nonlinear configurations.

4.2.1. Synthetic Linear Symbolic Interval Datasets

We followed the approach of Lima Neto and De Carvalho in [14] and constructed a symbolic interval dataset with the center and range values of the intervals simulated according to a linear relationship.

Our goal was to construct a symbolic dataset with 375 rows and 4 variables. We used 250 rows as the learning set and 125 as the test set. We followed these steps to construct the synthetic dataset.

1. The random variable X_j^c was uniformly distributed in $[20, 40]$.
2. The random variable Y^c was related to the random variables X_j^c according to $Y^c = \mathbf{X}^c \boldsymbol{\beta} + \epsilon$, where $\mathbf{X}^c = (1, X_1^c, X_2^c, X_3^c)$, $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \beta_3)$ with $\beta_j \sim U[-10, 10]$, and $\epsilon \sim U[a, b]$.
3. The random variable Y^r was related to the variable Y^c according to $Y^r = Y^c \beta^* + \epsilon^*$, in the same manner X_j^r was related to X_j^c according to $X_j^r = X_j^c \beta^* + \epsilon^*$, where $\beta^* \sim U[0.5, 1.5]$ and $\epsilon^* \sim U[i, j]$.

The next table shows different configurations for the values of a, b and i, j .

As per Lima Neto and De Carvalho, these configurations in Table 3 took into account the error at the midpoints combined with the degree of dependence between the midpoints and ranges, with two levels of variability: high variability error $U[-20, 20]$, low variability error $U[-5, 5]$, a high degree of dependence $U[1, 5]$, and a low degree of dependence $U[10, 20]$.

We present the results for all the datasets.

For dataset D1, we were able to see that the lasso, ridge, and LM center and range models were the best.

For dataset D2, the lasso, ridge, and LM center and range models were the best, and the KNN showed the next best performance.

Once again, for dataset D3, the lasso, ridge, and LM center and range model were the best models, without any close competitors.

For dataset D4, the lasso, ridge, LM, and the boost center and range model were the best. Therefore, the boost model was able to perform well in this dataset.

Note that in Tables 4–7, some models have NA on r_L^2 or r_U^2 , and this is because for those models, either the prediction $\hat{y}_L$ or $\hat{y}_U$ had zero standard deviation, and thus, the square of the correlation coefficient cannot be calculated.

Table 3. Configurations with the center and range with a linear relationship. D1, Dataset 1.

D1	$\epsilon \sim U[-20, 20]$	$\epsilon^* \sim U[1, 5]$
D2	$\epsilon \sim U[-20, 20]$	$\epsilon^* \sim U[10, 20]$
D3	$\epsilon \sim U[-5, 5]$	$\epsilon^* \sim U[1, 5]$
D4	$\epsilon \sim U[-5, 5]$	$\epsilon^* \sim U[10, 20]$

Table 4. Performance of the methods for the D1 dataset.

Method	RMSE _L	RMSE _U	r_L^2	r_U^2
LM CM	94.2771	97.2412	0.2129	0.9710
Ridge CM	66.2051	67.6800	0.2131	0.9711
Lasso CM	92.9969	95.7236	0.2129	0.9710
RT CM	222.4937	324.0148	NA	NA
RF CM	258.4727	313.7617	NA	NA
Boost CM	235.2453	286.0998	NA	NA
KNN CM	248.2890	295.5031	0.0731	0.5838
SVM CM	332.8712	376.9175	0.0434	0.0282
NNet CM	133.2041	134.9265	0.2086	0.7714
LM CRM	11.1395	22.2623	0.2756	0.9716
Ridge CRM	10.2776	24.5461	0.2791	0.9719
Lasso CRM	11.0720	22.2897	0.2760	0.9717
RT CRM	25.5772	52.6583	0.0401	0.8410
RF CRM	11.9748	34.3630	0.1905	0.9521
Boost CRM	13.9365	25.9654	0.1388	0.9613
KNN CRM	12.0253	28.2744	0.2130	0.9563
SVM CRM	11.6466	27.3321	0.1579	0.9571
NNet CRM	12.6617	25.0032	0.1415	0.9641

NA: number not available.

4.2.2. Synthetic Nonlinear Symbolic Interval Datasets

We considered a synthetic interval-valued dataset with a nonlinear relation between the response variable and the explanatory variables for the midpoint and range of the intervals.

In order to test the predictive power of the proposed methods, we followed the approach of Lima Neto and De Carvalho in [18] and constructed a symbolic interval dataset with the values of the center and range of the intervals simulated according to a nonlinear relationship between the response variable and the explanatory variables.

Table 5. Performance of the methods for the D2 dataset.

Method	RMSE _L	RMSE _U	r_L^2	r_U^2
LM CM	208.2321	211.0179	0.0391	0.8514
Ridge CM	182.7565	185.2648	0.0389	0.8501
Lasso CM	207.0609	209.7936	0.0391	0.8509
RT CM	92.9444	119.6915	NA	NA
RF CM	109.5064	127.3871	NA	NA
Boost CM	84.7520	108.0348	NA	NA
KNN CM	99.2145	120.9664	0.0331	0.5261
SVM CM	153.5649	161.4119	NA	0.0196
NNet CM	41.5299	76.7332	0.0221	0.2926
LM CRM	12.3272	22.9804	0.7451	0.8829
Ridge CRM	10.8678	23.4012	0.7399	0.8817
Lasso CRM	12.2189	23.0000	0.7453	0.8826
RT CRM	21.3481	35.0539	0.2794	0.7257
RF CRM	11.3682	27.7170	0.5811	0.8503
Boost CRM	14.3557	23.3277	0.6593	0.8799
KNN CRM	12.4306	24.2769	0.7061	0.8723
SVM CRM	13.3351	24.3496	0.6763	0.8685
NNet CRM	15.5105	24.7335	0.6063	0.8644

NA: number not available.

Table 6. Performance of the methods for the D3 dataset.

Method	RMSE _L	RMSE _U	r_L^2	r_U^2
LM CM	731.6593	731.9502	0.8309	0.9835
Ridge CM	713.0555	711.7151	0.8309	0.9835
Lasso CM	730.6541	730.8220	0.8311	0.9835
RT CM	429.9846	405.3798	0.0000	NA
RF CM	428.2619	391.7614	0.0194	NA
Boost CM	392.7227	354.0930	0.2566	NA
KNN CM	413.8011	364.5694	0.4734	0.4805
SVM CM	491.7221	468.3171	0.1116	0.1944
NNet CM	343.0967	302.1161	0.8369	0.6365
LM CRM	30.4423	29.5111	0.9709	0.1197
Ridge CRM	34.7499	27.7174	0.9707	0.1324
Lasso CRM	30.6647	29.3510	0.9709	0.1198
RT CRM	70.3581	49.0615	0.8417	0.0329
RF CRM	54.0816	28.2500	0.9458	0.0724
Boost CRM	37.4534	31.9037	0.9576	0.0635
KNN CRM	41.8289	32.3533	0.9563	0.0203
SVM CRM	37.0102	29.4140	0.9615	0.0598
NNet CRM	36.1038	34.2631	0.9597	0.0487

NA: number not available.

Our goal was to construct a symbolic dataset with 3000 rows and 4 variables and compare the methods using a K-fold cross-validation scheme with 10 folds. We followed the following steps to construct the synthetic dataset.

1. The center random variables X_j^c were uniformly distributed in the interval $[-6, 6]$; that is, $X_j^c \sim \mathcal{U}[-6, 6]$.
2. The center random variable Y^c was related to the random variables X_j^c according to the logistic function:

$$Y_i^c = \frac{\theta_0^c}{\theta_1^c + e^{\theta_2^c X_{i1}^c + \dots + \theta_{n+1}^c X_{in}^c}} + \epsilon_i^c,$$

where X_{ji}^c and Y_i^c are the i th entries of the variables X_j^c and Y^c , respectively, $\theta_0^c \sim \mathcal{U}[1.9, 2.1]$, $\theta_1^c \sim \mathcal{U}[2.9, 3.1]$, and $\theta_m^c \sim \mathcal{U}[0.9, 1.1]$ for $m = 2, \dots, n+1$, and the error component ϵ_i^c is normally distributed, where $\epsilon_i^c \sim \mathcal{N}(0, 0.05)$.

3. The range random variables X_j^r were uniformly distributed in the interval $[1, 4]$; that is, $X_j^r \sim \mathcal{U}[1, 4]$.
4. The range random variable Y^r was related to the random variables X_j^r according to the exponential function:

$$Y_i^r = \theta_0^r + e^{-(\theta_1^r X_{1i}^r + \dots + \theta_n^r X_{ni}^r)} + \epsilon_i^r,$$

where X_{ji}^r , Y_i^r are the i th entries of the variables X_j^r , Y^r , respectively, $\theta_0^r \sim \mathcal{U}[0, 0.5]$ and $\theta_m^r \sim \mathcal{U}[0.9, 1.1]$ for $m = 1, \dots, n$, and the error component ϵ_i^r is normally distributed, where $\epsilon_i^r \sim \mathcal{N}(0, 0.01)$.

Table 7. Performance of the methods for the D4 dataset.

Method	RMSE _L	RMSE _U	r_L^2	r_U^2
LM CM	209.5566	210.1047	0.0041	0.9305
Ridge CM	165.3670	165.6865	0.0042	0.9304
Lasso CM	205.1182	205.6540	0.0041	0.9306
RT CM	273.0309	289.5941	NA	NA
RF CM	274.3768	301.9242	NA	NA
Boost CM	241.1328	259.7999	NA	NA
KNN CM	252.3822	275.0876	0.0040	0.2819
SVM CM	339.6949	354.9830	NA	0.0714
NNet CM	184.6237	212.1473	0.0022	0.3013
LM CRM	18.1846	18.8634	0.0012	0.9473
Ridge CRM	16.7686	20.6743	0.0012	0.9470
Lasso CRM	18.0800	18.9213	0.0012	0.9474
RT CRM	23.3177	29.6393	0.0116	0.8671
RF CRM	15.7857	25.8713	0.0098	0.9272
Boost CRM	17.6997	19.1021	0.0089	0.9466
KNN CRM	18.8748	22.6928	0.0002	0.9299
SVM CRM	19.0737	21.1123	0.0053	0.9362
NNet CRM	21.0460	21.8625	0.0019	0.9287

NA: number not available.

As per Lima Neto and De Carvalho, the previous configuration had a high nonlinearity degree at the midpoints and a high nonlinearity degree in the ranges, as defined by the error components.

We here present the results for the synthetic dataset.

We note from Table 8 that every CRM method outperformed all of the CM methods. On closer inspection, we noticed that the neural network CRM model had the best cross-validation performance among all methods, and we observed that the proposed nonlinear models generally had better evaluation metrics. The second best method was the KNN CRM model, followed by the SVM CRM model.

Table 9 shows the standard deviation of the methods among the 10 folds.

As we can see from Table 9, the standard deviation was low for all methods among the folds, which gave us confidence in our results.

Table 8. Ten-fold cross-validation mean metrics.

Method	RMSE _L	RMSE _U	r_L^2	r_U^2
LM CM	0.6656	0.6656	0.7322	0.7306
Ridge CM	0.6493	0.6493	0.7322	0.7306
Lasso CM	0.6634	0.6634	0.7322	0.7306
RT CM	0.2766	0.2927	0.3274	0.2710
RF CM	0.2426	0.2667	0.3991	0.3599
Boost CM	0.1717	0.1757	0.6961	0.6962
KNN CM	0.2525	0.2687	0.3356	0.3091
SVM CM	0.2798	0.2907	0.1969	0.2017
NNet CM	0.2642	0.2715	0.2695	0.2764
LM CRM	0.1411	0.1408	0.7743	0.7746
Ridge CRM	0.1418	0.1415	0.7743	0.7746
Lasso CRM	0.1411	0.1408	0.7743	0.7746
RT CRM	0.1495	0.1495	0.7462	0.7453
RF CRM	0.0681	0.0683	0.9531	0.9525
Boost CRM	0.1428	0.1425	0.7684	0.7687
KNN CRM	0.0580	0.0581	0.9621	0.9619
SVM CRM	0.0617	0.0617	0.9570	0.9568
NNet CRM	0.0514	0.0519	0.9701	0.9694

Table 9. Ten-fold cross-validation standard deviation metrics.

Method	RMSE _L	RMSE _U	r_L^2	r_U^2
LM CM	0.0111	0.0092	0.0123	0.0197
Ridge CM	0.0109	0.0093	0.0123	0.0197
Lasso CM	0.0111	0.0092	0.0123	0.0197
RT CM	0.0093	0.0040	0.0376	0.0318
RF CM	0.0075	0.0050	0.0268	0.0206
Boost CM	0.0089	0.0071	0.0305	0.0211
KNN CM	0.0075	0.0047	0.0221	0.0225
SVM CM	0.0073	0.0069	0.0190	0.0359
NNet CM	0.0079	0.0049	0.0209	0.0212
LM CM	0.0055	0.0049	0.0144	0.0129
Ridge CM	0.0054	0.0047	0.0144	0.0129
Lasso CM	0.0055	0.0048	0.0144	0.0129
RT CM	0.0117	0.0115	0.0463	0.0464
RF CM	0.0036	0.0030	0.0051	0.0047
Boost CM	0.0056	0.0051	0.0144	0.0139
KNN CM	0.0019	0.0012	0.0026	0.0026
SVM CM	0.0014	0.0017	0.0024	0.0030
NNet CM	0.0023	0.0023	0.0024	0.0028

5. Conclusions and Future Work

In this paper, we proposed 12 new methods of fit regression models to interval-valued variables, all based on the central idea of fitting regression models for the centers and ranges of the intervals and extending the ideas of the nonlinear methods for real-valued data. We presented new approaches to fit regression models for symbolic interval-valued variables, which were shown to improve and extend the center method the center and range method proposed by Lima Neto and De Carvalho in [6,13,14,18].

In the experimental evaluation, we found that the use of nonlinear methods greatly improved the prediction results in the regression problems. With the cardiological dataset, a simple neural network was able to radically improve the predictions in comparison to the other methods, especially when compared to those based on linear methods. In the Monte Carlo experiments, as expected, the linear models were the best when using synthetic linear symbolic interval datasets, and only the boosting center and range model was close in performance in one of the datasets. When using the synthetic nonlinear symbolic interval

data, we observed the real power and advantages of the nonlinear framework for regression with interval-valued data; in particular, we saw the benefit of using neural networks, as this was once again the model that best captured the underlying structure of the data, when combined with the center and range approach. When comparing with the linear models, we saw that the neural networks center and range model had a $RMSE_L$ and a $RMSE_U$ of 0.051, which was more than half lower than the root-mean-squared-errors of those center and range models based on linear methods of 0.141; in a similar way, the neural network had a r_L^2 and a r_U^2 of 0.97, which was much higher than the square of the correlation coefficients of those based on linear methods of 0.77. In addition, we want to note that almost all of the proposed models outperformed the classical models based on linear approaches and that these results did not consider hyperparameter tuning in the methods, which in turn represents an opportunity to further improve the results.

Based on the results found, we not only achieved our goal of extending the tool kit of regression models for interval-valued datasets, that had focused on linear methods, but also, we demonstrated the predictive advantages of making the extension to nonlinear methods. This is relevant due to the fact that, in real-life applications, data rarely follow a linear structure.

The methods proposed, just as the original center method, have the problem explained by Lima Neto and De Carvalho in [6], which is that it cannot be mathematically guaranteed that $\hat{y}_L \leq \hat{y}_U$. In future work, we will apply the idea proposed in [6], which consists of generating the regression models using certain restrictions that will allow us to guarantee that the methods satisfy this restriction. This was not included in this paper because it was expected to cause confusion in the results, since it would not have been clear if improvements in predictions were due to the applications of shrinkage or to the application of restrictions in the regression methods.

Author Contributions: The authors contributed equally to this work. Both the authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data sharing not applicable.

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

References

1. Diday, E. Introduction à l'approche symbolique en Analyse des Données. In *Premières Journées Symbolique-Numérique*; Université Paris IX Dauphine: Paris, France, 1987.
2. Billard, L.; Diday, E. From the statistics of data to the statistics of knowledge: Symbolic data analysis. *J. Am. Stat. Assoc.* **2003**, *98*, 470–487. [CrossRef]
3. Billard, L.; Diday, E. *Symbolic Data Analysis: Conceptual Statistics and Data Mining*; John Wiley & Sons Ltd.: Letchworth, UK, 2006.
4. Bock, H.-H.; Diday, E. (Eds.) *Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data*; Springer: Berlin, Germany, 2000.
5. Billard, L.; Diday, E. Regression analysis for interval-valued data. In *Data Analysis, Classification and Related Methods*; Springer: Berlin, Germany, 2000; pp. 369–374.
6. Lima-Neto, E.A.; De Carvalho, F.A.T. Constrained linear regression models for symbolic interval-valued variables. *Comput. Stat. Data Anal.* **2010**, *54*, 333–347. [CrossRef]
7. Rodríguez, O. Shrinkage Linear Regression for Symbolic Interval-Valued Variables. 2018. Available online: <https://editions-rnti.fr/?inprocid=1002533> (accessed on 29 March 2021).
8. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning; Data Mining, Inference and Prediction*; Springer: New York, NY, USA, 2008.
9. Hastie, T.; Zou, H. Regularization and variable selection via the elastic net. *Stat. Soc. B* **2005**, *67*, 301–320.
10. Izonin, I.; Tkachenko, R.; Kryvinska, N.; Greguš, M. Multiple linear regression based on coefficients identification using non-iterative SGTm neural-like structure. In *Proceedings of the International Work-Conference on Artificial Neural Networks*, Cadiz, Spain, 14–16 June 2019; Springer: Cham, Switzerland, 2019. [CrossRef]
11. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Soc. Ser. B* **1996**, *58*, 267–288. [CrossRef]
12. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: London, UK, 2016.

13. Lima Neto, E.d.A.; de Carvalho, F.d.A.T. An exponential-type kernel robust regression model for interval-valued variables. In *Information Sciences*; Elsevier: Amsterdam, The Netherlands, 2018; Volume 454, pp. 419–442.
14. Lima-Neto, E.A.; De Carvalho, F.A.T. Centre and range method to fitting a linear regression model on symbolic interval data. *Comput. Stat. Data Anal.* **2008**, *52*, 1500–1515. [[CrossRef](#)]
15. Rodríguez, O. Classification et Modèles Linéaires en Analyse des Données Symboliques. Ph.D. Thesis, Paris IX-Dauphine University, Paris, France, 2000.
16. Rodríguez, O.; Agüero, C.; Arce, J.; Chacón, J.E. RSDA-R to Symbolic Data Analysis. R Package Version 1.2. 2020. Available online: <http://CRAN.R-project.org/package=RSDA> (accessed on 29 March 2021).
17. Raju, S.R.K. Symbolic Data Analysis in Cardiology. In *Symbolic Data Analysis and Its Applications*; Diday, E., Gowda, K.C., Eds.; Ceremade, Université Paris-Dauphine: Paris, France, 1997; pp. 245–249.
18. Lima Neto, E.d.A.; de Carvalho, F.d.A.T. Nonlinear regression applied to interval-valued data. *Pattern Anal. Appl.* **2017**, *20*, 809–824. [[CrossRef](#)]