Top Journal manuscript No.
(will be inserted by the editor)

Supplementary Material

Antoni Torres–Signes · María P. Frías · María D. Ruiz-Medina

Received: date / Accepted: date

Abstract This Supplementary Material provides additional information about the empirical comparative study carried out in our paper.

Hard– and soft–data implementation of ML models

We briefly describe the implementation of ML models in the hard– and soft–data categories.

Multilayer Perceptron (MLP)

MLP shares the philosophy of nonlinear regression, in terms of a link function g, the hidden node output, defining the following approximation of the response:

$$\widehat{y} = \eta_0 + \sum_{k=1}^{NH} \eta_k g(\beta_k^T \mathbf{x}), \tag{1}$$

from the *input* vector $\mathbf{x}^T = (\mathbf{1}, \mathbf{x})$ augmented with 1, and the weight vector β_k associated with the kth hidden node, $k = 1, \ldots, NH$, defining $\boldsymbol{\beta} =$

A. Torres-Signes
Department of Statistics and Operation Research, Faculty of Sciences, University of Málaga, Spain
E-mail: atisignes@uma.es

M. P. Frías
Department of Statistics and Operation Research, Faculty of Sciences, University of Jaén, Spain
E-mail: mpfrias@ujaen.es

M.D. Ruiz-Medina
Department of Statistics and Operation Research, Faculty of Sciences, University of Granada, Spain
E-mail: mruiz@ugr.es

$(\beta_1^T, \ldots, \beta_{NH}^T)^T$. Usually, the logistic function $g(u) = \frac{1}{1+\exp(-u)}$ is considered. The hidden node outputs are also weighted by the components of the vector $(\eta_0, \ldots, \eta_{NH})$. In this context, $\widehat{\mathbf{y}}$ is usually referred as the network output. MLP allows the approximation of any given continuous function on a compact set, from a given network with a finite number of hidden nodes. Optimization algorithms are applied to obtain the weights from the least–squares loss function. From (1), our implementation of MLP from soft–data is given by

$$\widehat{y}_m = \widehat{y}(\mathbf{x}_m) = \ln\left(\widehat{\lambda_{t+1}}\right)(\phi_m) = \eta_0 + \sum_{k=1}^{NH} \eta_k g(\beta_k^T \mathbf{x}_m), \tag{2}$$

where, for $m = 1, \ldots, M(T)$,

$$\mathbf{x}_m = \left(\ln\left(\lambda_{t+1-1}\right)(\phi_m), \ldots, \ln\left(\lambda_{t+1-j_0}\right)(\phi_m)\right)^T. \tag{3}$$

Parameter j_0 refers to the number of temporal lags incorporated in the prediction. The truncation parameter $M(T)$ plays a similar role to parameter $k(T)$ in the paper. Here, T denotes the number of temporal training nodes.

Radial Basis Function Neural Network (RBF)

RBF works with node functions, depending on a center and scale parameters, fitting the local smoothness of the response. Specifically, from an initial blank network, the nodes are sequentially added, around the training pattern, until an acceptable error is reached. All the output layer weights are then recomputed using the least squares formula. Gaussian functions have been widely selected as node functions. Particularly, in this case, our implementation from soft–data has been achieved from the formula: For $m = 1, \ldots, M(T)$,

$$\widehat{y}_{t+1}(\mathbf{x}_m) = \ln\left(\widehat{\lambda_{t+1}}\right)(\phi_m) = \sum_{j=1}^{NH} \eta_j \exp\left(\frac{\|\mathbf{x}_m - c_j\|^2}{\beta^2}\right) \tag{4}$$

where, as usual, the weight parameters η_j, $j = 1, \ldots, NH$, define the linear combination of radial basis functions. Here, the scalar spread parameter β, and the vector parameters c_j, $j = 1, \ldots, NH$, respectively provide the width, and the centers of the node functions. The input vectors are defined as in (3).

Support Vector Regression (SVR)

SVR implementation involves a loss function leading to a balance between model complexity and precision (accurate prediction). A bias parameter b is also considered in its formulation as reflected in the following equation:

$$y = f(\mathbf{x}) = \boldsymbol{\beta}^T \mathbf{x} + b \tag{5}$$

where the loss function

$$\mathcal{L} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{m=1}^{M} |y_m - f(x_m)|_\epsilon \,, \tag{6}$$

is considered. Here, x_m and y_m respectively denote the mth training input vector and the target output, for $m = 1, \ldots, M$, and

$$|y_m - f(x_m)|_\epsilon = \max\{0, |y_m - f(x_m)| - \epsilon\} \,.$$

Thus, the errors below ϵ are not penalized. The solution to the optimization problem associated with the loss function (6) is obtained from the corresponding gradient of the Lagrangian function, involving Lagrange multipliers, that determine the optimal weights from the training data points.

From equation (5), in the soft–data category, we solve the constrained optimization problem:

$$\begin{aligned} & y_{t+1}(\mathbf{x}_m) = f(\mathbf{x}_m) = \boldsymbol{\beta}^T \mathbf{x}_m + b \\ & \mathcal{L} = \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{\beta} + C\sum_{m=1}^{M(T)} \left| \ln\left(\lambda_{t+1}\right)(\phi_m) - \sum_{j=1}^{j_0} \ln(\lambda_{t+1-j})(\phi_m)\beta_j - b \right|_\epsilon \\ & = \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{\beta} + C\sum_{m=1}^{M(T)} |y_m - f(\mathbf{x}_m)|_\epsilon \,, \end{aligned} \tag{7}$$

where, for $m = 1, \ldots, M(T)$, the imput vector $\mathbf{x}_m$ is defined as in (3).

Bayesian Neural Network (BNN)

The design of BNN involves Bayesian estimation, and the concept of regularization. The network parameters or weights are considered random variables, following a prior probability distribution. Smooth fits are usually favored in the selection of the prior probability of the weights to reduce model complexity. The posterior probability distribution of the weights is obtained after data are observed. The network prediction is then computed. Specifically, the optimal prediction is obtained by minimizing the following expression:

$$J = \nu E_{\mathbf{O}} + (1-\nu)E_{\mathbf{W}}, \tag{8}$$

where $E_{\mathbf{O}}$ is the sum of the square errors in the network output, based on the posterior distribution of the parameters, $E_{\mathbf{W}}$ represents the sum of the squares of the weights or the network parameters, and $\nu \in (0, 1)$ denotes the regularization parameter. Let L be the number of parameters. An L–dimensional Gaussian prior probability distribution is usually assumed for the network parameters with zero–mean and variance–covariance matrix $\frac{1}{2(1-\nu)} I_{L\times L}$, with $I_{L\times L}$ denoting the $L \times L$ identity matrix. Thus,

$$p(\mathbf{w}) = \left[\frac{1-\nu}{\pi}\right]^{L/2} \exp\left(-(1-\nu)E_W(\mathbf{w})\right). \tag{9}$$

This prior puts more weight onto small network parameter values close to zero. The posterior probability density, given the observed data $\mathbf{O} = \mathbf{o}$, and the value ν of the regularization parameter, is defined as

$$p(\mathbf{w}/\mathbf{o}, \nu) = \frac{p(\mathbf{o}/\mathbf{w}, \nu)p(\mathbf{w}/\nu)}{p(\mathbf{o}/\nu)}. \tag{10}$$

Considering that the errors are also Gaussian distributed, the conditional probability of the data $\mathbf{O}$ given the parameters ν and $\mathbf{w}$, is obtained as

$$p(\mathbf{o}/\mathbf{w}, \nu) = \left(\frac{\nu}{\pi}\right)^{M/2} \exp\left(-\nu E_{\mathbf{O}}(\mathbf{o})\right), \tag{11}$$

where M denotes the number of training data points. From equations (9)–(11),

$$p(\mathbf{w}/\mathbf{o}, \nu) = c \exp\left(-J(\mathbf{w}, \mathbf{o}, \nu)\right), \tag{12}$$

where c denotes the normalizing constant. The conditional probability of the parameter ν given de observed data $\mathbf{O} = \mathbf{o}$ is also computed under a Bayesian framework as

$$p(\nu/\mathbf{o}) = \frac{P(\mathbf{o}/\nu)p(\nu)}{p(\mathbf{o})}. \tag{13}$$

Equations (12) and (13) are maximized to obtain the optimal weights and the regularization parameter ν, respectively.

In our soft–data implementation from (3), the corresponding optimization problem is formulated by conditioning to the empirical spatial projections. Note that, in the selected Gaussian prior probability framework, the error projections are also Gaussian, and our choice of the function basis, diagonalizing the empirical autocovariance operator of the errors, leads to a projected error vector with independent Gaussian components, suitable normalized by the empirical eigenvalues. Hence, optimization from equations (12) and (13) can be implemented in a similar way to hard–data BNN.

Generalized Regression Neural Network (GRNN)

GRNN is based on kernel regression. The kernel estimator is computed from the weighted sum of the observed responses, or target outputs associated with the training data points in a neighborhood of the objective data point x, where prediction must be computed. Thus, the training data points are selected in the vicinity of the given objective point x. Specifically, the following formula is applied in the approximation of the response value at the point x :

$$\widehat{y}(x) = \sum_{j=1}^{T} \frac{\mathcal{K}\left(\frac{\|x-x_j\|}{h}\right)}{\sum_{l=1}^{T} \mathcal{K}\left(\frac{\|x-x_l\|}{h}\right)} y_j, \tag{14}$$

where y_j is the target output for training data point x_j, for $j = 1, \ldots, T$, and $\mathcal{K}$ is the kernel function. Usually an isotropic rapidly decreasing kernel function

is considered, e.g., the Gaussian kernel $\mathcal{K}(u) = \exp\left(-u^2/2\right)/\sqrt{2\pi}$ constitutes a common choice. The bandwidth parameter h defines the smoothness of the fit. Thus, h controls the size of the smoothing region. Hence, large values of h correspond to a stronger smoothing than the smallest values allowing a larger degree of local variation. In our soft–data implementation of (14), denote by $\Phi_{M(T)}(H)$, the subspace of H obtained by projection of functions in H onto $\{\phi_1, \ldots, \phi_{M(T)}\}$, and $\mathbf{y}_t = \left(\ln(\lambda_t)(\phi_1), \ldots, \ln(\lambda_t)(\phi_{M(T)})\right)^T$, $t \geq 1$, then,

$$\widehat{\mathbf{y}_{t+1}} = \sum_{j=1}^{j_0-1} \frac{\mathcal{K}\left(\frac{\|\mathbf{y}_t - \mathbf{y}_{t-j}\|_{\Phi_{M(T)}(H)}}{h}\right)}{\sum_{l=1}^{j_0-1} \mathcal{K}\left(\frac{\|\mathbf{y}_t - \mathbf{y}_{t-l}\|_{\Phi_{M(T)}(H)}}{h}\right)} \mathbf{y}_{t-j+1}, \tag{15}$$

where, as before, parameter j_0 refers to the number of temporal lags incorporated in the prediction.

Gaussian Processes (GP)

A good performance is usually observed in the implementation of GP regression, based on the multivariate normal probability distribution assumption, characterizing the observed responses at the different training data points. Specifically, we consider the observation model

$$\mathbf{Y} = \mathbf{Z} + \boldsymbol{\epsilon}, \tag{16}$$

where the additive noise vector $\boldsymbol{\epsilon}$ is independent of $\mathbf{Z}$, and has independent and identically distributed zero–mean Gaussian components with variance $\sigma^2_{\boldsymbol{\epsilon}}$. A multivariate normal distribution of the random vector $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, with covariance matrix $\boldsymbol{\Sigma}_{\mathbf{Z}}(X, X)$ is assumed. This matrix provides the variances and covariances between the function values $Z(x_i)$, and $Z(x_j)$, $i, j = 1, \ldots, N$, at the training data points $X = (x_1, \ldots, x_N)$. The conditional Gaussian distribution of $\mathbf{Z}$, given $\mathbf{Y}$, leads to the solution to the inverse estimation problem (16). Thus, the estimation of $\mathbf{Z}$, for a given input vector $\mathbf{x}_\star$, is obtained as

$$\widehat{\mathbf{Z}}_{\mathbf{x}_\star} = E\left[\mathbf{Z}/\mathbf{y}, \mathbf{x}_\star, X\right] = \boldsymbol{\Sigma}_{\mathbf{Z}}(\mathbf{x}_\star, X)\left[\boldsymbol{\Sigma}_{\mathbf{Z}}(X, X) + \sigma^2_{\boldsymbol{\epsilon}}\mathbf{I}\right]^{-1}\mathbf{y}. \tag{17}$$

In the soft–data category, an alternative multivariate implementation is achieved in terms of projection $\Phi_{M(T)}$ onto $\{\phi_1, \ldots, \phi_{M(T)}\}$. Hence, $\boldsymbol{\Sigma}_{\mathbf{Z}}$ is replaced by the matrix covariance operator

$$R_{\mathbf{Z}}^{X,X} = \begin{pmatrix} \Phi^\star_{M(T)}\mathcal{R}_{1,1}\Phi_{M(T)} & \ldots & \Phi^\star_{M(T)}\mathcal{R}_{1,T}\Phi_{M(T)} \\ \Phi^\star_{M(T)}\mathcal{R}_{2,1}\Phi_{M(T)} & \ldots & \Phi^\star_{M(T)}\mathcal{R}_{2,T}\Phi_{M(T)} \\ \vdots & \ldots & \vdots \\ \Phi^\star_{M(T)}\mathcal{R}_{T,1}\Phi_{M(T)} & \ldots & \Phi^\star_{M(T)}\mathcal{R}_{T,T}\Phi_{M(T)} \end{pmatrix},$$

associated with the T temporal training nodes considered, and the projection operator $\Phi_{M(T)}$ onto $\{\phi_1, \ldots, \phi_{M(T)}\}$, involved in the definition of our training

soft–data points X. Here, $\mathcal{R}_{i,j} = E\left[[\ln(\Lambda_i) - E[\ln(\Lambda_i)]] \otimes [\ln(\Lambda_j) - E[\ln(\Lambda_j)]]\right]$, $i, j = 1, \ldots, T$, define the autocovariance and cross-covariance operators of the functional values at the training data points. In this case, $\sigma^2_{\boldsymbol{\epsilon}}\mathbf{I}$ becomes the diagonal matrix autocovariance operator $diag\,(R_{0,\boldsymbol{\epsilon}})$ of the H^T–valued innovation process $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_T)$. That is,

$$\begin{aligned}\widehat{\mathbf{Z}}_{\mathbf{x}_\star} &= E\left[\mathbf{Z}/\mathbf{y}, \mathbf{x}_\star, X, \Phi_{M(T)}\right] \\ &= R_{\mathbf{Z}}^{\mathbf{x}_\star, X}\left[R_{\mathbf{Z}}^{X,X} + \Phi^\star_{M(T)} diag\,(R_{0,\boldsymbol{\epsilon}})\,\Phi_{M(T)}\right]^{-1}\Phi_{M(T)}(\mathbf{y}).\end{aligned}$$

Observed hard– and soft–data ML SMAPE sample values

The Symmetric Mean Absolute Percentage Errors (SMAPEs) associated with ML estimation results, based on the overall functional sample, from hard– and soft–data, are respectively displayed in Tables 1 and 2. See also Figures 1–4 below, where the observed and estimated COVID–19 mortality log–risk and cumulative cases curves are respectively displayed.

Table 1: ***Hard–data***. As indicated, displayed values must be multiplied by 10^{-2}

SC ($\mathbf{x10^{-2}}$)	**GRNN**	**MLP**	**SVR**	**BNN**	**RBF**	**GP**
C1	0.1964	0.0611	0.0665	0.0535	0.0483	0.0490
C2	0.6117	0.1172	0.0588	0.0678	0.0609	0.0567
C3	0.1565	0.0375	0.0328	0.0284	0.0300	0.0273
C4	0.0969	0.0314	0.0129	0.0191	0.0167	0.0185
C5	0.2044	0.0427	0.0290	0.0356	0.0334	0.0328
C6	0.1571	0.0319	0.0161	0.0230	0.0217	0.0218
C7	0.4889	0.0545	0.0619	0.0583	0.0569	0.0503
C8	0.0808	0.0273	0.0161	0.0180	0.0189	0.0153
C9	0.7231	0.1976	0.0850	0.1102	0.0318	0.0352
C10	0.2185	0.0526	0.0532	0.0446	0.0428	0.0415
C11	0.1255	0.0487	0.0298	0.0350	0.0338	0.0316
C12	0.5216	0.1666	0.1210	0.1022	0.0870	0.0887
C13	0.3584	0.0592	0.0548	0.0613	0.0490	0.0412
C14	0.1342	0.0286	0.0201	0.0202	0.0182	0.0180
C15	0.6064	0.1470	0.1307	0.0931	0.0864	0.0927
C16	0.2456	0.0681	0.0674	0.0573	0.0514	0.0532
C17	0.0655	0.0356	0.0147	0.0207	0.0181	0.0200
M.	0.2936	0.0710	0.0512	0.0499	0.0415	0.0408
T.	4.9916	1.2078	0.8707	0.8480	0.7053	0.6936

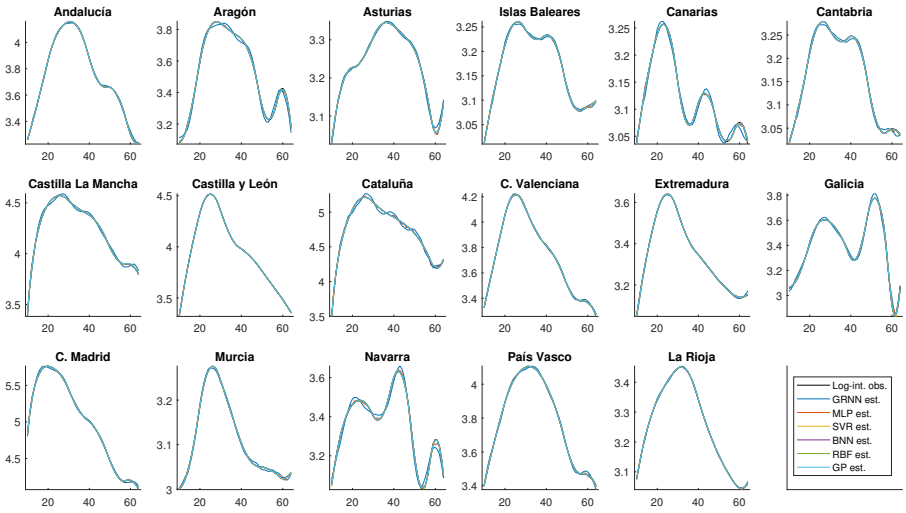

Fig. 1: ***Hard–data category***. Observed and estimated COVID–19 mortality log–risk curves, from the implementation of Generalized Regression Neural Network (GRNN), Multilayer Perceptron (MLP), Support Vector Regression (SVR), Bayesian Neural Network (BNN), Radial Basis Function Neural Network (RBF), and Gaussian Processes (GP)

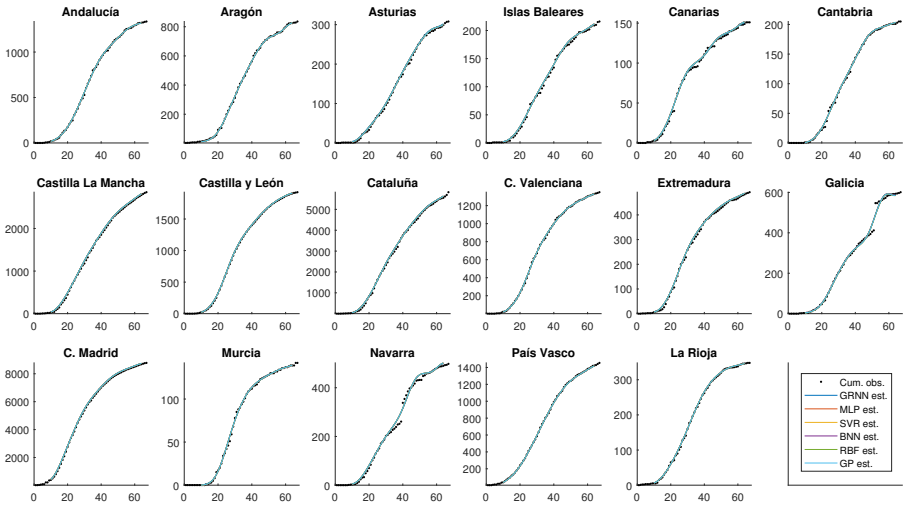

Fig. 2: ***Hard–data category***. Observed and estimated COVID–19 mortality cumulative cases curves from the implementation of Generalized Regression Neural Network (GRNN), Multilayer Perceptron (MLP), Support Vector Regression (SVR), Bayesian Neural Network (BNN), Radial Basis Function Neural Network (RBF), and Gaussian Processes (GP)

Table 2: ***Soft–data***. As indicated, displayed values must be multiplied by 10^{-2}

SC (x10^{-2})	**GRNN**	**MLP**	**SVR**	**BNN**	**RBF**	**GP**
C1	0.1526	0.0857	0.0592	0.0501	0.0223	0.0305
C2	0.1831	0.1393	0.0619	0.0606	0.0248	0.0285
C3	0.1024	0.0804	0.0457	0.0374	0.0268	0.0278
C4	0.0431	0.0212	0.0157	0.0154	0.0117	0.0120
C5	0.0610	0.0362	0.0242	0.0240	0.0133	0.0138
C6	0.0260	0.0167	0.0125	0.0135	0.0118	0.0121
C7	0.3734	0.1301	0.0914	0.0737	0.0298	0.0398
C8	0.0762	0.0435	0.0290	0.0238	0.0244	0.0180
C9	0.4850	0.2467	0.1470	0.0818	0.0199	0.0360
C10	0.1663	0.0607	0.0460	0.0421	0.0236	0.0276
C11	0.1533	0.0540	0.0394	0.0383	0.0180	0.0209
C12	0.3641	0.2325	0.1320	0.0887	0.0369	0.0480
C13	0.2827	0.1350	0.0707	0.0699	0.0215	0.0304
C14	0.0361	0.0197	0.0117	0.0121	0.0102	0.0104
C15	0.3590	0.1843	0.1226	0.1049	0.0290	0.0506
C16	0.1759	0.0754	0.0566	0.0489	0.0243	0.0302
C17	0.0878	0.0352	0.0190	0.0219	0.0122	0.0134
N.	0.1840	0.0939	0.0579	0.0475	0.0212	0.0265
T.	3.1282	1.5966	0.9845	0.8070	0.3605	0.4497

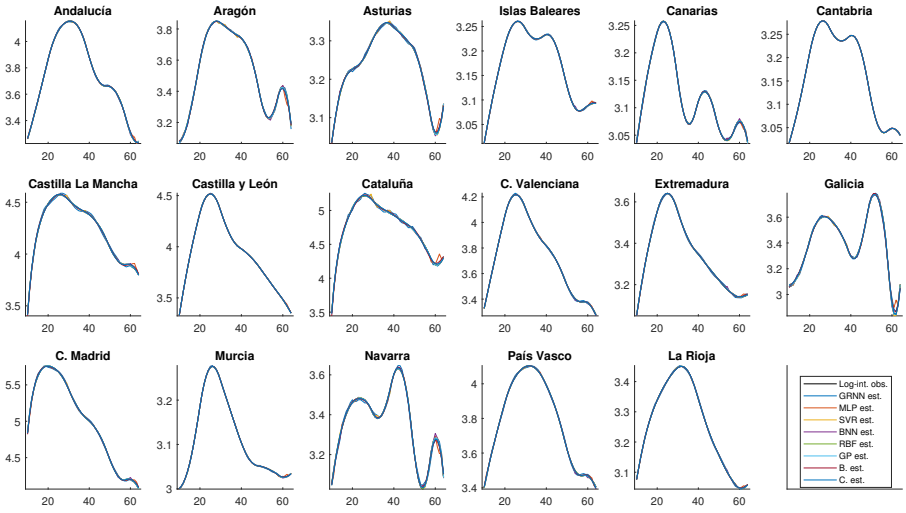

Fig. 3: ***Soft–data category***. Observed and estimated COVID–19 mortality log–risk curves, from the implementation of Generalized Regression Neural Network (GRNN), Multilayer Perceptron (MLP), Support Vector Regression (SVR), Bayesian Neural Network (BNN), Radial Basis Function Neural Network (RBF), Gaussian Processes (GP), and trigonometric regression combined with classical and Bayesian residual prediction

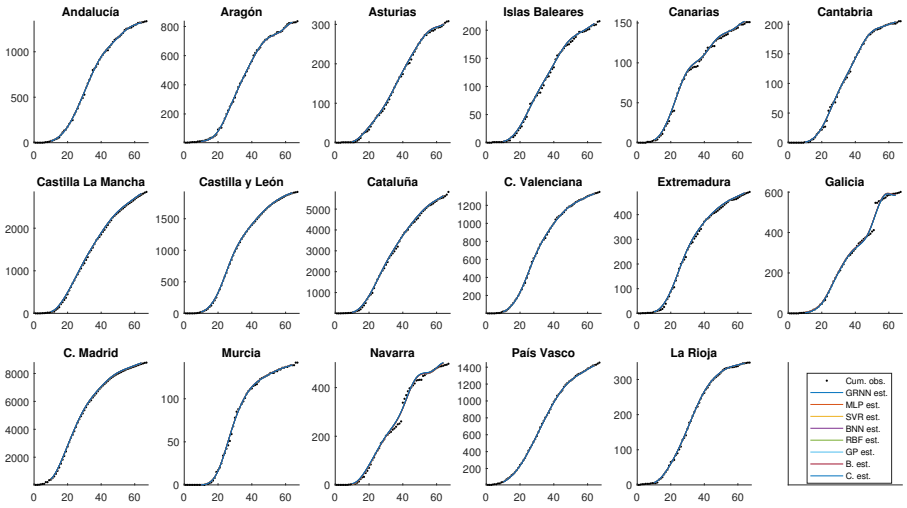

Fig. 4: ***Soft–data category***. Observed and estimated COVID–19 mortality cumulative cases curves, from the implementation of Generalized Regression Neural Network (GRNN), Multilayer Perceptron (MLP), Support Vector Regression (SVR), Bayesian Neural Network (BNN), Radial Basis Function Neural Network (RBF), Gaussian Processes (GP), and trigonometric regression combined with classical and Bayesian residual prediction

Randon 5–fold cross–validation

The random 5–fold cross–validation SMAPEs obtained from implementation of the six ML regression models tested are displayed in Table 3, for hard–data category, and in Table 4, for soft–data category.

Table 3: ***Hard–data category***. Averaged SMAPEs for 10 running of random 5–fold cross–validation. (As indicated, displayed values must be multiplied by 10^{-2})

SC(x10^{-2})	**GRNN**	**MLP**	**SVR**	**BNN**	**RBF**	**GP**
C1	0.1962	0.0845	0.0890	0.0709	0.0635	0.0592
C2	0.6150	0.1531	0.0711	0.0805	0.0782	0.0710
C3	0.1541	0.0479	0.0438	0.0338	0.0371	0.0325
C4	0.0984	0.0388	0.0190	0.0226	0.0214	0.0226
C5	0.2065	0.0555	0.0378	0.0414	0.0429	0.0397
C6	0.1585	0.0423	0.0227	0.0257	0.0266	0.0263
C7	0.4957	0.0786	0.0771	0.0707	0.0712	0.0587
C8	0.0804	0.0352	0.0226	0.0215	0.0247	0.0189
C9	0.7280	0.2170	0.1061	0.0866	0.0421	0.0475
C10	0.2208	0.0724	0.0751	0.0577	0.0541	0.0494
C11	0.1273	0.0618	0.0373	0.0460	0.0451	0.0385
C12	0.5237	0.1706	0.1441	0.1449	0.1126	0.1065
C13	0.3637	0.0717	0.0701	0.0671	0.0605	0.0480
C14	0.1359	0.0388	0.0307	0.0235	0.0220	0.0213
C15	0.6105	0.1705	0.1592	0.1228	0.1101	0.1121
C16	0.2479	0.0931	0.0926	0.0753	0.0665	0.0632
C17	0.0667	0.0414	0.0195	0.0255	0.0241	0.0246
M.	0.2958	0.0867	0.0657	0.0598	0.0531	0.0494
T.	5.0293	1.4731	1.1177	1.0166	0.9026	0.8400

Table 4: ***Soft–data category***. Averaged SMAPEs, based on 10 running of random 5–fold cross–validation. (As indicated, displayed values must be multiplied by 10^{-2})

SC(x10^{-2})	**GRNN**	**MLP**	**SVR**	**BNN**	**RBF**	**GP**
C1	0.1569	0.0958	0.0695	0.0604	0.0240	0.0337
C2	0.1836	0.1835	0.0697	0.0880	0.0286	0.0329
C3	0.1029	0.1309	0.0474	0.0491	0.0273	0.0284
C4	0.0433	0.0299	0.0171	0.0181	0.0133	0.0130
C5	0.0609	0.0524	0.0282	0.0264	0.0153	0.0159
C6	0.0259	0.0239	0.0133	0.0148	0.0130	0.0129
C7	0.3783	0.2136	0.1018	0.0963	0.0309	0.0439
C8	0.0774	0.0467	0.0315	0.0320	0.0292	0.0207
C9	0.4968	0.2926	0.1458	0.1329	0.0254	0.0417
C10	0.1710	0.0935	0.0541	0.0489	0.0277	0.0316
C11	0.1556	0.0914	0.0456	0.0441	0.0221	0.0247
C12	0.3759	0.2235	0.1509	0.1396	0.0402	0.0560
C13	0.2894	0.1599	0.0903	0.0775	0.0259	0.0368
C14	0.0375	0.0250	0.0124	0.0153	0.0114	0.0109
C15	0.3646	0.2410	0.1378	0.1297	0.0334	0.0573
C16	0.1792	0.0828	0.0677	0.0578	0.0292	0.0344
C17	0.0900	0.0724	0.0216	0.0256	0.0129	0.0144
M.	0.1876	0.1211	0.0650	0.0622	0.0241	0.0299
T.	3.1893	2.0589	1.1047	1.0567	0.4100	0.5090