computers

MDPI

Article

Pm2.5 Time Series Imputation with Deep Learning and Interpolation

Anibal Flores [1,2,*], Hugo Tito-Chura [1,2], Deymor Centty-Villafuerte [3] and Alejandro Ecos-Espino [4,5]

[1] Grupo de Investigación en Inteligencia Artificial, Universidad Nacional de Moquegua, Urb. Ciudad Jardin-Pacocha-Ilo, Moquegua 18611, Peru
[2] Departamento Académico de Ingeniería de Sistemas e Informática, Universidad Nacional de Moquegua, Urb. Ciudad Jardin-Pacocha-Ilo, Moquegua 18611, Peru
[3] Departamento de Ciencias Sociales y Humanidades, Universidad Nacional de Moquegua, Prolongación Calle Ancash S/N, Moquegua 18001, Peru
[4] Departamento Académico de Ciencias Básicas, Universidad Nacional de Moquegua, Prolongación Calle Ancash S/N, Moquegua 18001, Peru
[5] Instituto de Investigación para el Desarrollo del Perú (IINDEP), Universidad Nacional de Moquegua, Prolongación Calle Ancash S/N, Moquegua 18001, Peru
* Correspondence: afloresg@unam.edu.pe

Abstract: Commonly, regression for time series imputation has been implemented directly through regression models, statistical, machine learning, and deep learning techniques. In this work, a novel approach is proposed based on a classification model that determines the NA value class, and from this, two types of interpolations are implemented: polynomial or flipped polynomial. An hourly pm2.5 time series from Ilo City in southern Peru was chosen as a study case. The results obtained show that for gaps of one NA value, the proposal in most cases presents superior results to techniques such as ARIMA, LSTM, BiLSTM, GRU, and BiGRU; thus, on average, in terms of R^2, the proposal exceeds implemented benchmark models by between 2.4341% and 19.96%. Finally, supported by the results, it can be stated that the proposal constitutes a good alternative for short-gaps imputation in pm2.5 time series.

Keywords: time series imputation; classification; deep learning; polynomial interpolation; flipped polynomial interpolation

Citation: Flores, A.; Tito-Chura, H.; Centty-Villafuerte, D.; Ecos-Espino, A. Pm2.5 Time Series Imputation with Deep Learning and Interpolation. *Computers* **2023**, *12*, 165. https://doi.org/10.3390/computers12080165

Academic Editors: Phivos Mylonas, Katia Lida Kermanidis and Manolis Maragoudakis

Received: 18 July 2023
Revised: 2 August 2023
Accepted: 10 August 2023
Published: 16 August 2023

1. Introduction

Time series forecasting is one of the most active research topics [1] in machine learning and deep learning; it is used in different domains such as biology, finance, meteorology, and social sciences [2], among others.

Missing values in time series is a common problem [2], and many machine and deep learning techniques do not work with missing data [3], so time series imputation becomes a very important task. It is also very important to mention that the quality of data for time series forecasting is very important [4], hence the importance of a good time series imputation process.

In this work, a novel approach for time series imputation is proposed, which combines a math technique with a deep learning one. The imputation problem is approached as a classification problem, and the proposal uses a classification model to determine the class of NA value, and from the class, an interpolation technique is implemented. The main motivation stems from the analysis of different short gaps of NA values in time series, where it was possible to identify two classes. The first, according to Figure 1a, comprises the NA values whose real values fit better to a polynomial interpolation. The second, according to Figure 1b, comprises the NA values whose real values fit better to a flipped polynomial interpolation. As can be seen in Figure 1, the NA value can be located above or below the

line that joins the prior value (p0) with the next value (n0). Depending on the preceding values p1, p0 and following n0, n1, and derivatives, the NA value could be estimated more accurately with polynomial interpolation or flipped polynomial interpolation. The classification model was trained with 12 features, including the aforementioned p1, p0, n0, and n1, adding p2 and n2 and other derivatives of them that are described in detail in the Section 3 of this work.

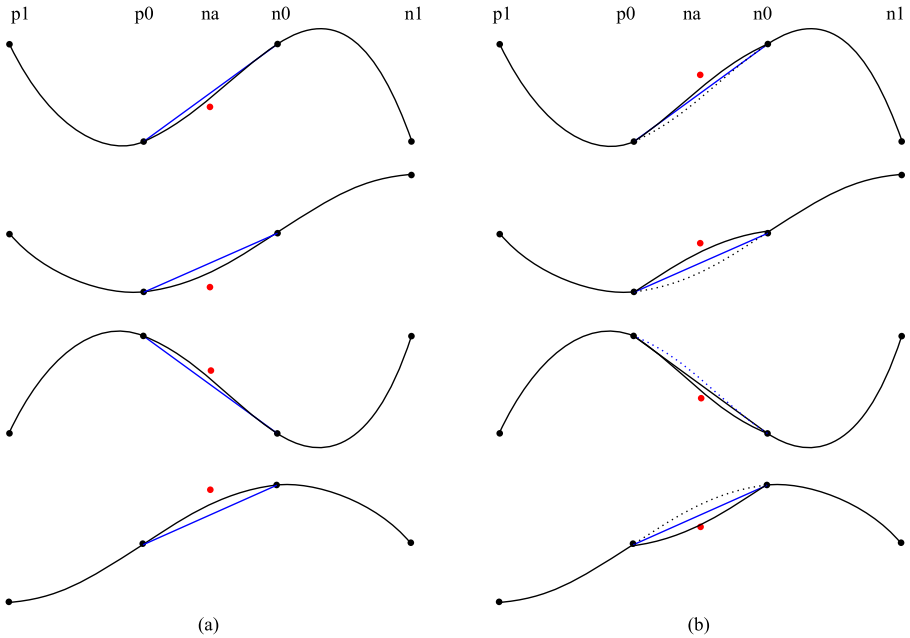


Figure 1. Types of NA values. (**a**) Class 0 and (**b**) class 1.

The study case selected to validate the novel approach corresponds to hourly pm2.5 time series obtained from an environmental monitoring station in Ilo City in southern Peru. The study case was chosen due to the importance of the analysis of the pm2.5 time series since, as is known, long-term exposure to pm2.5 can cause diverse health issues, including heart disease [5], lung cancer [6], chronic obstructive pulmonary disease [7], lower respiratory infections (LRIs) [8], ischemic stroke [9], diabetes mellitus [10], and others. However, the pm2.5 time series presents NA values due to multiple factors such as equipment failure [11], power outages, etc., as occurs with the data from the monitoring station chosen. Therefore, the importance of implementing imputation techniques in order to recover lost data and made it available for the implementation of forecasting models.

The novel approach was compared with other state-of-the-art models for time series imputation, such as long short-term memory (LSTM) [12], bidirectional LSTM [13], gated recurrent unit (GRU) [14], bidirectional GRU, ARIMA, local average of nearest neighbors (LANN) [15], and polynomial interpolation.

The main contributions of this work are cited next:

- A novel approach for pm2.5 time series imputation based on deep learning classification;
- An ensemble deep learning model for NA values classification;
- A comparative analysis between the proposal approach versus benchmark models.

The main limitations of this work are as follows: It is only focused on the analysis of short gaps, gives NA sequences of a single value in pm2.5 time series, and the interpolations are performed considering only four values: p1, p0, n0, and n1.

The rest of the paper is organized in Section 2 that briefly describes the work performed for pm2.5 time series imputation; Section 3, where the complete process for implementation of the proposal is described; and Section 4, where the achieved results are described and discussed according to the pros and cons of proposal results. The paper ends with Section 5.

2. Related Work

Some works for pm2.5 time series imputation where the use of machine learning and deep learning techniques were proposed are listed below in chronological order.

Xiao et al., 2018 [16] proposed an ensemble model for pm2.5 time series imputation. The proposal presents superior results to models such as random forest (RF), extreme gradient boosting (XGBoosting), and the generalized additive model (GAM). Yuan et al., 2018 [17] proposed the RNN long short-term memory (LSTM) for the imputation process and compared this technique with two classic techniques: moving average and mean imputation. The results show the enormous superiority of LSTM over the other two techniques.

Belachsen et al., 2022 [18] proposed a multivariate KNN technique for half-hourly pm2.5 time series reaching an NMAE between 0.21 and 0.26. Saif-ul-Allah et al., 2022 [19] proposed the recurrent neural network known as GRU, reaching an RMSE of 10.60 ug/m^3 and surpassing other models such as SVM, LSTM, and BiLSTM. Alkabbani et al., 2022 [20] proposed a multivariate random forest model for pm2.5 time series imputation, and the results were compared only with linear interpolation, showing that random forest achieves the best RMSE (3.756 ug/m^3).

Yldz et al., 2022 [21] proposed a transformer model for multivariate time series imputation; they experimented with air quality (pm2.5) and healthcare time series. For pm2.5, they worked with hourly data of 12 months, obtaining a MAE = 8.31 ug/m^3 that exceeded those obtained by benchmark models such as BRITS, RDIS, V-RIN, etc.

Lee et al., 2023 [22] proposed four machine learning models, namely GAIN, FCDNN, random forest, and KNN, and the best results were obtained with the GAIN model with R^2 = 0.895.

Reviewing other related works for time series imputation, in general, it can be noted that these address techniques ranging from moving averages: simple moving average (SMA), linear weighted moving average (LWMA), exponential weighted moving average (EWMA), and interpolation techniques (linear, spline, and Stinneman), all generally implemented in the imputeTS library [2,18] of R. On the other hand, there were also deep learning-based ones, including LSTM [23,24], GRU [25], and GAN [26] and some variants such as BRITS [27], GRU-D [28], and B-CIPIT [29].

The main difference between the related works and the proposal is that these works to estimate NA values generally apply a regression model in some cases associated with a time series transformation. In our model, at the beginning, NA features were extracted and labeled; then, features were normalized; next, a time series classification task with deep learning models was performed, followed by the implementation of an ensemble model; finally the class of an NA value was estimated, and from this, the proper polynomial interpolation was performed to obtain the estimated NA value. Figure 2 summarizes this process.

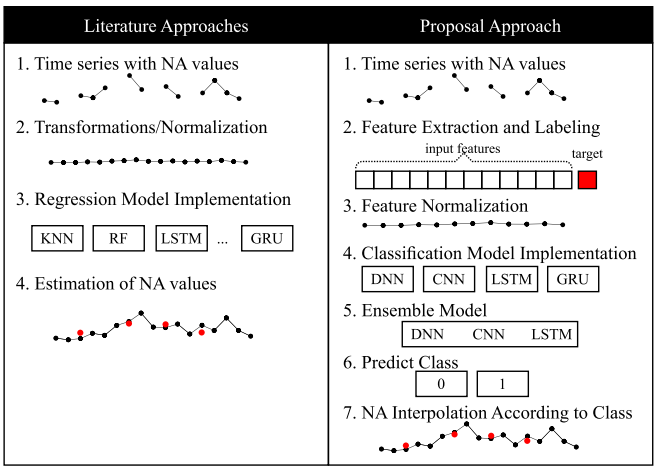


Figure 2. Comparison between literature approaches and proposal approach.

3. Materials and Methods

The methodology used for the implementation of proposal approach is summarized in Figure 3.

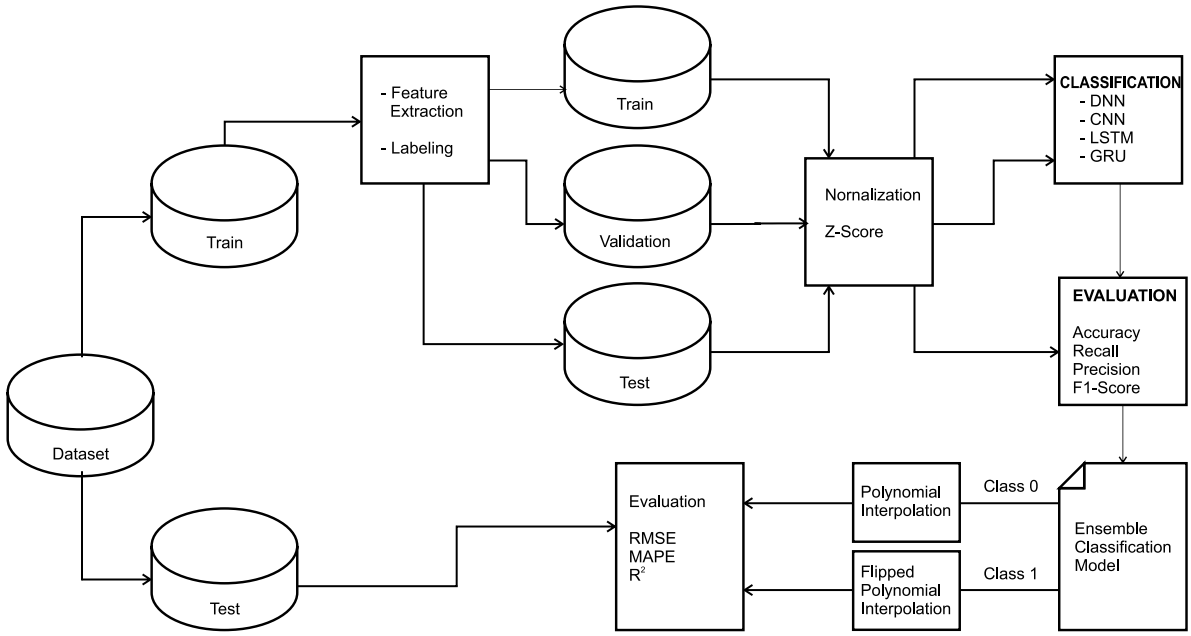


Figure 3. Methodology for the implementation of proposal approach.

3.1. Data Preparation

The hourly data used for this work were downloaded from OEFA's server located at http://fiscamb.oefa.gob.pe/vig_amb/ accessed on 2 May 2023, and they oscillate between 1 August 2020 and 30 April 2023, corresponding to the Pacocha environmental monitoring station located at Ilo Province and Pacocha District in southern Peru. The downloaded data present several missing values, so the days that presented some missing values were discarded. After this process, the total available records numbered 56,424. As it is a regression problem, the dataset was just split into two sets (training and test) [30]. For training, 48,792 records were used (80%), and 7632 records were used for testing (20%). Figure 4 shows a sample of 1000 h of the selected time series. Data normalization was performed for extracted features during the implementation of classification models; it is described in Section 3.2.3.

3.2. Implementation of Classification Models

For classification models, the dataset was split into three sets: training, validation, and test. For the proposal, from 48,792 records corresponding to the training data, 70% was used for training, 10% for validation, and the remaining 20% was used for testing the classification models.

Therefore, at this stage, the first task we performed corresponds to feature extraction and labeling of NA values, which is described below.

3.2.1. Feature Extraction and Labelling

The features that were extracted from the time series are described in Table 1.

The motivation to use the described features for NA values classification was the method by which moving averages techniques work: they use a parameter *k* that corresponds to the window size. The window size tells the technique how many values before and after the NA value should be used for the corresponding estimation; thus, p2, p1, p0, n0, n1, and n2 were obtained considering a window size k = 3. For future work, higher

values could be used. The rest of the features were derived or synthetic; that is, they were generated to provide more information for model training.

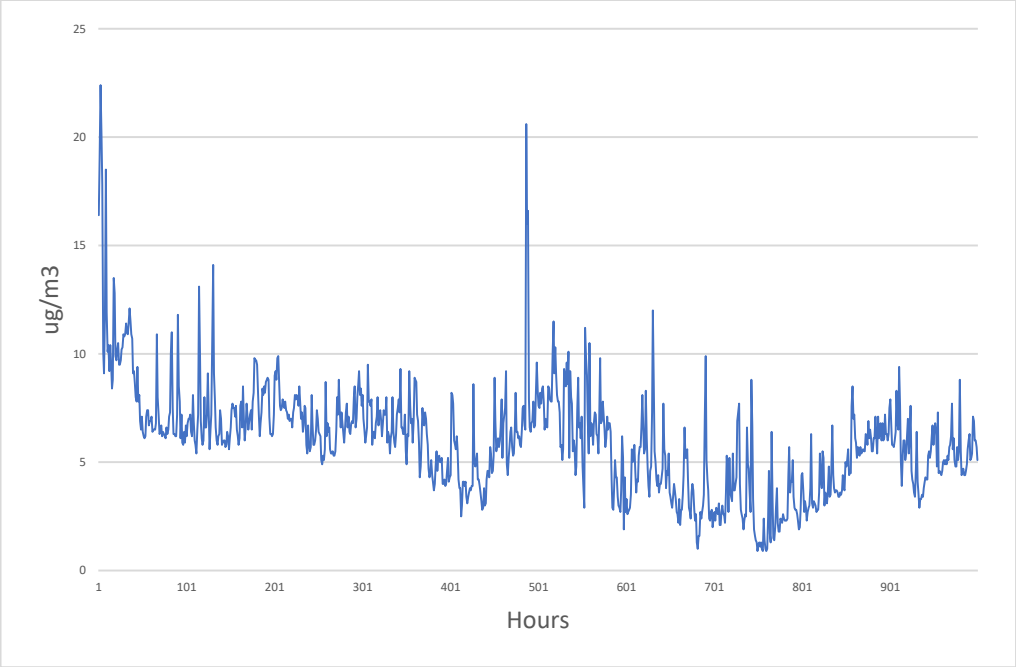


Figure 4. Sample of 1000 h of pm2.5 time series.

Table 1. Features for NA values.

Feature	Description
p0	Prior value to NA
p1	Before p0
p2	Before p1
n0	Next value of NA
n1	After n0
n2	After n1
mid	Mean between prior (p0) and next value (n0)
mid1	Mean between prior values (p1) and next value (p0)
mid2	Mean between next values (n0) and next value (n1)
diff	Difference between prior and next values
slope1	Slope between p1 and p0
slope2	Slope between n0 and n1
label	Class of NA value. 0, corresponding to polynomial interpolation 1, corresponding to flipped polynomial interpolation.

Given a *t* time series, the features shown in Table 1 were extracted using the algorithm shown in Figure 5.

The algorithm shown in Figure 5 received as an argument a time series t, which was traversed from beginning to end. As can be seen, the feature extraction was quite simple: p2, p1, p0, n0, n2, and n2 were estimated from the position of the NA value; mid, mid1, and mid2 are averages of p1, p0, n0, and n1; diff, slope1, and slope2 are the differences between p1, p0, n0, and n1. However, the estimation of the class or label is somewhat complex, and it requires that certain conditions be met; these conditions are detailed in Table 2.

3.2.2. Feature Selection

For this task, a correlation matrix was implemented, which can be seen in Figure 6.

```
procedure getFeatures(t)
Begin
    nt=t.length-3
    c0=0
    c1=0
    label=""
    for(i=0 → nt)
    Begin
        k=i
        p2=t[k-1]
        p1=t[k]
        p0=t[k+1]
        na=t[k+2]
        n0=t[k+3]
        n1=t[k+4]
        n2=t[k+5]
        mid=(p0+n0)/2
        mid1=(p0+p1)/2
        mid2=(n0+n1)/2
        diff=(p0-n0)
        slope1=(p1-p0)
        slope2=(n1-n0)
        if(na<=mid)
        Begin
            if(p1>=p0)
            Begin
                c0=c0+1
                label=0
            End
            else
            Begin
                c1=c1+1
                label=1
            End
        End
        else
        Begin
            if(p1>=p0)
            Begin
                c1=c1+1
                label=1
            End
            else
            Begin
                c0=c0+1
                label=0
            End
        End
        reg=[p2,p1,p0,n0,n1,n2,mid,mid1,mid2,diff,slope1,slope2,label]
        features.push(reg)
    End
    return(features)
End
```

Figure 5. Feature extraction algorithm of NA values.

According to the last row or last column of the correlation matrix, the relationship between the input features and the target feature was not strong; thus, it would not be easy to obtain good results in terms of accuracy, precision, recall, and f1-score. In this work, it was decided to use all the input features for classification models.

Table 2. Conditions for NA classes.

Class	Conditions
0 (polynomial interpolation)	According to Figure 1a, the conditions to apply polynomial interpolation are two cases: Case 1: When na is below the line (p0 to n0), it can be stated through the next conditions: na $\leq$ mid p1 $\geq$ p0 Case 2: When na is above the line (p0 to n0). The two conditions to be met are: na > mid p0 > p1
1 (flipped polynomial interpolation)	According to Figure 1b, the conditions to apply flipped polynomial interpolation are two: Case 1: When na is below the line (p0 to n0), it can be stated through the next conditions: na < mid p1 < p0 Case 2: When na is above the line (p0 to n0). The two conditions to be met are: na > mid p0 $\leq$ p1

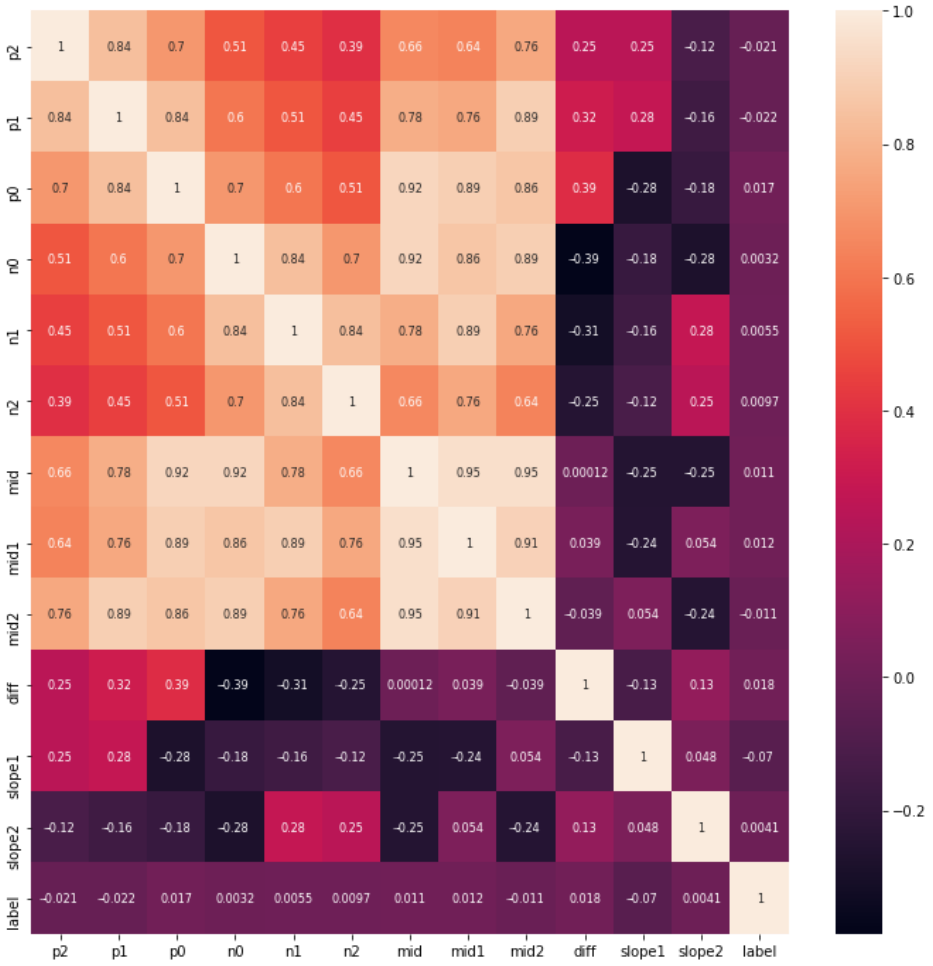


Figure 6. Correlation matrix.

3.2.3. Normalization

Before the implementation of deep learning models, normalization is very important in order to ensure their faster convergence. In this study, z-score normalization was used; it can be implemented through Equation (1).

$$X' = \frac{x - \bar{x}}{\sigma} \quad (1)$$

where

x': normalized vector;
x: original feature vector;
$\bar{x}$: mean of the feature vector;
σ: standard deviation of the feature vector.

3.2.4. Deep Learning Classification Models

In this stage, four deep learning models were implemented, including deep neural networks (DNN), convolutional neural networks (DNN), long short-term memory (LSTM), and gated recurrent unit (GRU), whose architectures are described in Table 3.

Table 3. Architectures of deep learning classification models.

Model	Hyperparameters
DNN	(0, 20, 40, 10, 1), learning_rate: 0.0001, dropout_rate (0.1)
CNN	(20, 50, 10, 1), learning_rate: 0.0001, dropout_rate (0.1)
LSTM	(30, 30, 30, 1), learning_rate: 0.0001, dropout_rate (0.1)
GRU	(30, 30, 30, 1), learning_rate: 0.0001, dropout_rate (0.1)

According to Table 3, all classification models use a learning rate of 0.0001 and dropout rates of 0.1 after all layers except the output layer. The DNN presents an architecture with a greater number of layers than the other models, all with relu as the activation function with 10 neurons in the input layer; 20, 40, and 10 in the hidden layers; and 1 neuron in the output layer with sigmoid the as activation function. Regarding the CNN model, it presents its first two Conv1-type layers with relu as activation function, followed by a MaxPooling1D layer with pool_size = 2, then a dense layer of 10 neurons, and, finally, a dense layer with 1 neuron. The LSTM and GRU RNNs present identical architectures: one input layer with 30 neurons, two hidden layers with 30 neurons each, and an output layer of 1 neuron.

All models were compiled with Adam as optimizer, loss function: binary_crossentropy, 200 epochs, and batch_size = 1000. Jupyter and Tensorflow 2.9.0 were used.

3.2.5. Evaluation

The results of the classification models based on deep learning are shown in Table 4. These are described in terms of accuracy (2), precision (3), recall (4), and f1-score (5).

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (2)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4)$$

$$f1 - \text{score} = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})} \quad (5)$$

Table 4. Results of deep learning classification models.

Model	Accuracy	Recall	Precision	F1-Score
DNN	**0.5972**	0.3083	**0.5891**	0.4048
CNN	0.5912	0.4223	0.5522	**0.4786**
LSTM	0.5859	0.3644	0.5513	0.4388
GRU	0.5859	**0.4046**	0.5458	0.4647

Once the classification models were compiled, they were evaluated in test data. The respective confusion matrices are shown in Figure 7.

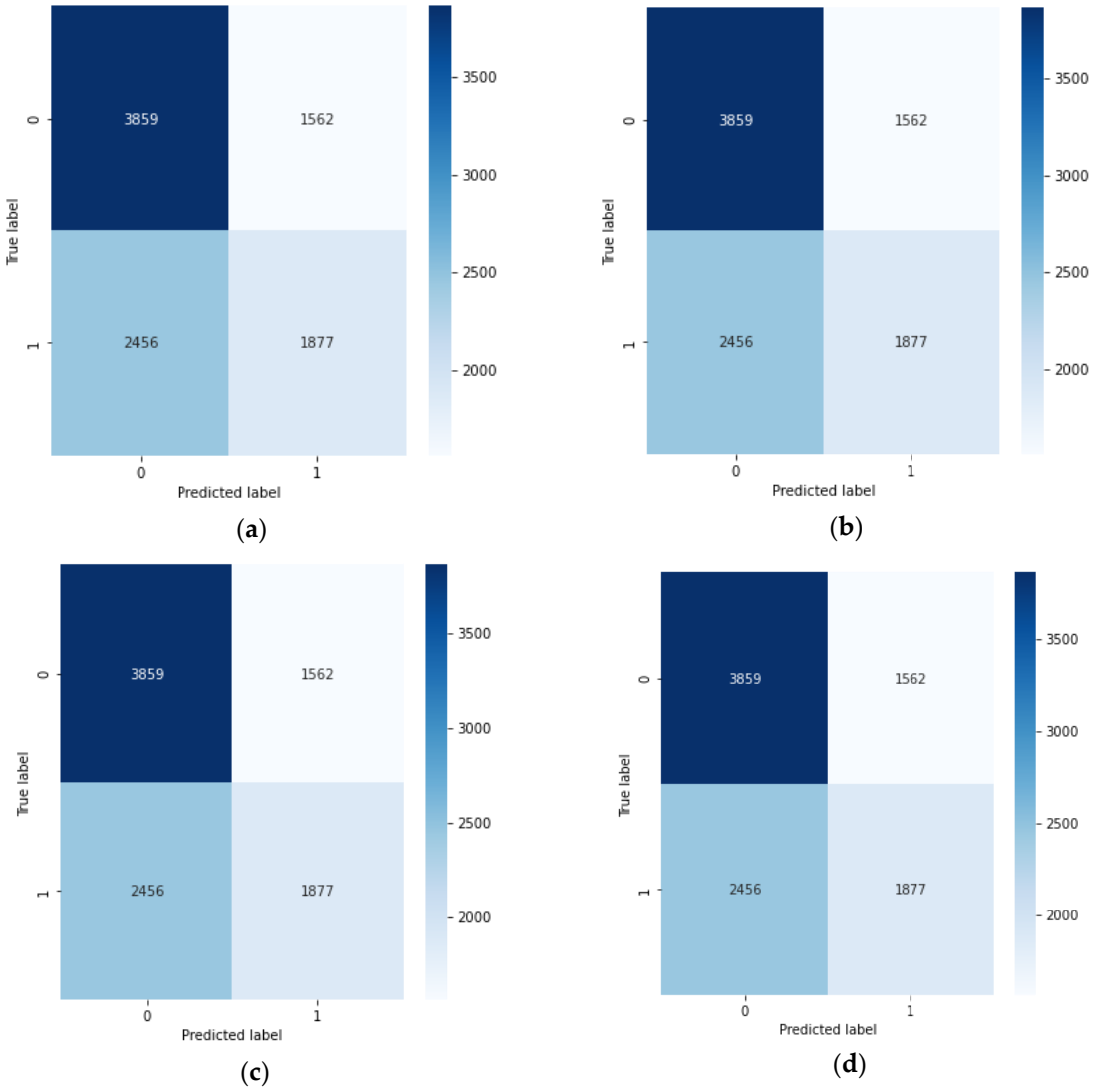


Figure 7. Confusion matrices of implemented classification models. (**a**) DNN, (**b**) CNN, (**c**) LSTM, and (**d**) GRU.

From the confusion matrices and Equations (2) and (5)–(7), the metrics shown in Table 4 were estimated.

According to Table 4 and Figure 7, it can be seen that the results of the classification models are not good, and it can be seen that the main difficulty presented by the implemented models is the low capacity to detect true positives (NAs of class 1); this is reflected in the recall below 43% and f1-score below 48%.

In order to improve the results, the strategy of implementing an ensemble model based on average was used. For this, it was experimented by assembling all the models and

combinations of three models. Figure 8 shows the respective ROC curves, and as it can be seen, the ensemble model based on DNN, CNN, and LSTM presented the best area under the curve (AUC of 0.611), which is why this ensemble model was chosen for the imputation process. In terms of accuracy, recall, precision, and f1-score, the ensemble model reached 0.5859, 0.4045, 0.5457, and 0.4647, respectively.

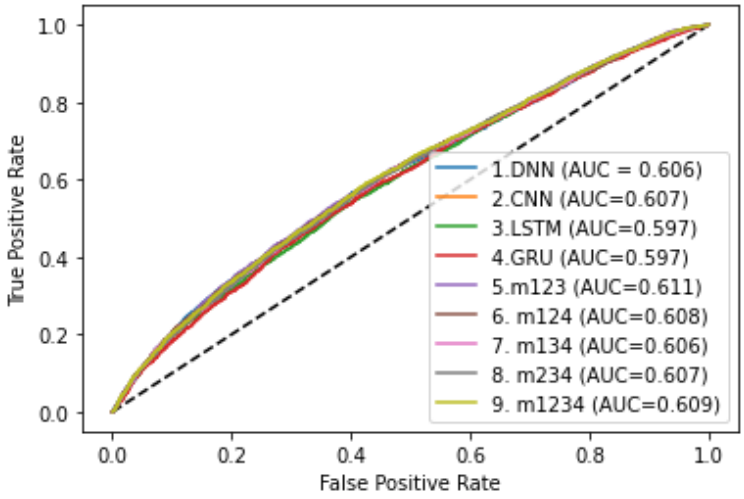


Figure 8. ROC curves of implemented and ensemble models.

3.3. Imputation of NA Values

Once the classification model for NA class estimation was obtained, the pm2.5 time series imputation process using the proposal was as described below.

3.3.1. Generation of NA Values in Test Data

NA values were generated considering gaps of a single NA value. Three different sets of NA values were implemented: 19.98% (1525 items), 25.00% (1907 items), and 33.32% (2543 items).

The NA values insertion mechanism was quite simple: to achieve 19.98% of NA values, one NA value was inserted into the test data every four items. To achieve 25.00%, NA values were inserted every three items, and to achieve 33.32%, NA values were inserted every two items. In this way, for each set, it was possible to obtain different amounts and configurations for the NA values. The partial result of this process can be visualized in Figure 9.

19.98% NAs	16.5	16.6	15.1	15.1	**NA**	15	27.7	16.6	16.6	**NA**	18.7	21	23.6	26	**NA**	30	28.3	28.8	36.6	**NA**	27.8	28
25.00% NAs	16.5	16.6	15.1	**NA**	15.6	15	27.7	**NA**	16.6	16.9	18.7	**NA**	23.6	26	37	**NA**	28.3	28.8	36.6	**NA**	27.8	28
33.32% NAs	16.5	16.6	**NA**	15.1	15.6	**NA**	27.7	16.6	**NA**	16.9	18.7	**NA**	23.6	26	**NA**	30	28.3	**NA**	36.6	32.7	NA	28

Figure 9. Sets of NA values in test data.

3.3.2. Class Estimation for NA Values

To estimate the class of each NA value in every NA set, a Python function was created.

The .h5 files correspond to DNN, CNN, and LSTM models that are part of the selected ensemble model to estimate the classes of NA values. The Python function getClass receives as the input parameter the characteristics of all NA values to be estimated, and then, it returns the classes to which they correspond. Figure 10 shows such a function.

```
procedure getClass(f)
Begin
   ff = f.split("\n")
   nf = len(ff)-1
   features = []
   for (i = 0 → nf)
   Begin
       fields = ff[i].split(";")
       features.push(fields)
   End
   #Normalizing data according training stats
   mean = [-4.62x10^-6,-4.57x10^-6,-6.12x10^-6,-5.55x10^-6,-6.51x10^-6,-5.17x10^-6,8.66x10^-6,1.98x10^-6,2.01x10^-6,1.80x10^-9,-5.89x10^-9,2.39x10^-9]
   std = [1.0000024,1.0000026,0.9999996,1.0000011,1.0000051,1.,1.0000088,1.0000032,1.0000062,0.9999938,0.9999984,0.99999243]
   features -= mean
   features /=std
   m1 = load_model('DNN_5972.h5')
   m2 = load_model('CNN_5912.h5')
   m3 = load_model('LSTM_5859.h5')
   #the predictions
   p1 = m1.predict(features)
   p2 = m2.predict(features)
   p3 = m3.predict(features)
   p123 = keras.layers.average([p1,p2,p3])
   res=""
   for (i = 0 → features.length)
   Begin
       if p123[i]>0.5
         res=res+"1*"
       else
         res=res+"0*"
   End
   return(res)
End
```

Figure 10. Procedure for Classes Estimation of NA values.

3.3.3. Interpolation according to Class Estimation

For polynomial interpolation, the first step is to determine the coefficients of the polynomial function; for this, there are various techniques, including the Lagrange method, which is described below.

Given the n points $(x_0, y_0), \ldots, (x_n-1, y_n-1)$, the Lagrange polynomial is estimated through Equation (6).

$$p(x) = \sum_{i=0}^{n-1} y_i \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)} \quad (6)$$

From (6), the coefficients are obtained, and the polynomial function can be implemented, and from it, any point can be estimated, in this case, the point corresponding to the *NA* value. The polynomial function is similar to what is shown in Equation (7).

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} \quad (7)$$

The algorithm that implements the interpolations according to the estimated class is shown in Figure 11.

According to Figure 10, the getNA procedure for NA estimation receives as parameters the NA class and the vector y that contains the four values to be used by polynomial interpolation: p1, p0, n0, and n1. The p0 and n0 are interpolated, and mid is obtained; then, the coefficients of the polynomial function are obtained for the four values in y, and for this, a polynomial procedure is used (See Figure 12). With the polynomial obtained, the value in position 1.5 is estimated, which corresponds to the NA value according to Figure 1; for this, the procedure interpolate is used (see Figure 13), and for na, this is the final result for NA values of class 0. For the case of NA values of class 1, the na value is flipped; for this, the absolute distance d between the na value and mid is determined, and depending on the location of the na value according to mid, d is subtracted or added.

```
procedure getNA(class,y)
Begin
    x=[0,1,2,3]
    p1=y[0]
    p0=y[1]
    n0=y[2]
    n1=y[3]
    mid=(p0+n0)/2
    poly=polynomial(4,x,y)
    na=interpolate(poly,1.5)
    d=|na-mid|
    if(class==1)
    Begin
        if(mid<na)
            na=mid-d
        else
            na=mid+d
    End
    return(na)
End
```

Figure 11. Algorithm for NA estimation.

```
1  procedure polynomial(points, Xs, Ys)
2  Begin
3      term=[]
4      poly=[]
5      for (i = 0 → points-1)
6          poly[i] = 0.0
7      for (i = 0 → points-1)
8      Begin
9          prod = 1.0
10         for (j=0 → points-1)
11             term[j] = 0.0
12         for (j=0 → points-1)
13         Begin
14             if (i == j)
15                 continue
16             prod = prod * (Xs[i] - Xs[j]);
17         End
18         prod = Ys[i]/prod
19         term[0] = prod
20         for (j=0 → points-1)
21         Begin
22             if (i == j)
23                 continue
24             for (k = points - 1 → 1)
25             Begin
26                 term[k] = term + term[k-1]
27                 term[k-1] = term[k-1]*(-Xs[j])
28             End
29         End
30         for (j = 0 → points-1)
31         Begin
32             poly[j] = poly[j] + term[j]
33         End
34     End
35     return (poly)
36 End
```

Figure 12. Algorithm of polynomial procedure.

```
procedure interpolate(poly,v)
Begin
    np=mpoly.length
    s=0
    for(i=0→np-1)
    Begin
        s=s+mpoly[i]*Math.pow(v,i)
    End
    return(s)
End
```

Figure 13. Algorithm of interpolate procedure.

The polynomial procedure receives as parameters the number of points (points) and the vectors of values in x (Xs) and y (Ys). With these data, according to Equation (6), for each point, Equation (8) is estimated, and from this, Equation (9) must be estimated.

$$\text{temp} = \frac{y_i}{\prod_{j \neq i}(x_i - x_j)} \tag{8}$$

$$\text{term} = \text{temp} * \left(\prod_{j \neq i}(x - x_j)\right) \tag{9}$$

Between lines 9 and 19 of the algorithm, Equation (8) is estimated, and Equation (9) is estimated between lines 20 and 29. The last part of the algorithm generates the coefficients of the polynomial function.

The interpolate procedure algorithm receives as parameters the coefficients of the polynomial (poly) and the position of the value to be estimated (v = x = 1.5), which in this case corresponds to the position of the NA value, and from this, Equation (7) is implemented.

The source code of the proposal can be downloaded from the next link: https://drive.google.com/drive/folders/1qL-k80rXqjFVmi-4fubg6nFvQbj3-Xq6?usp=drive_link accessed on 2 August 2023.

3.4. Implementation of Benchmark Models

Seven benchmark models were implemented in order to compare the proposal performance; these models include polynomial interpolation, LANN, ARIMA, long short-term memory (LSTM), bidirectional LSTM, gated recurrent unit (GRU), and bidirectional GRU.

Polynomial interpolation and ARIMA were implemented in R language using the imputeTS library. For ARIMA, the auto.arima model was used.

Deep regression models were implemented in Python using tensorflow 2.9.0, and the hyperparameters can be seen in Table 5.

Table 5. Hyperparameters for deep regression models.

Model	Hyperparameters
LSTM	architecture: [40, 30, 30, 40, 1], dropout_rate: 0.2
BiLSTM	architecture: [30, 30, 30, 1], dropout_rate: 0.1
GRU	architecture: [40, 30, 30, 40, 1], dropout_rate: 0.1
BiGRU	architecture: [30, 30, 30, 1], dropout_rate: 0.1

All deep regression models use Adam as the optimizer, mean standard error (mse) as the loss function, and 0.001 as learning rate. Also, the number of epochs used was 100, and the batch_size was 100, too. Finally, all layers used relu as the activation function except the output one-neuron layer, which used sigmoid as the activation function.

3.5. Evaluation

The proposal is evaluated in terms of root mean squared error (RMSE), mean absolute percentage error (MAPE), and R^2, which are implemented through Equations (10)–(12)

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(P_i - O_i)^2}{n}} \tag{10}$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(O_i - P_i)}{O_i} * 100\right| \tag{11}$$

$$R^2 = 1 - \left(\sum_{i=1}^{n}\left(O_i - P_i\right)^2 \Big/ \sum_{i=1}^{n}\left(O_i - \bar{O}\right)^2\right) \tag{12}$$

where

n: Number of observed/predicted values;
P_i: Vector of predicted values;
O_i: Vector of observed values;
$\bar{O}$: Mean of observed values.

4. Results and Discussion

This section describes the results achieved by the proposal, comparing them with other models in the literature.

4.1. Results

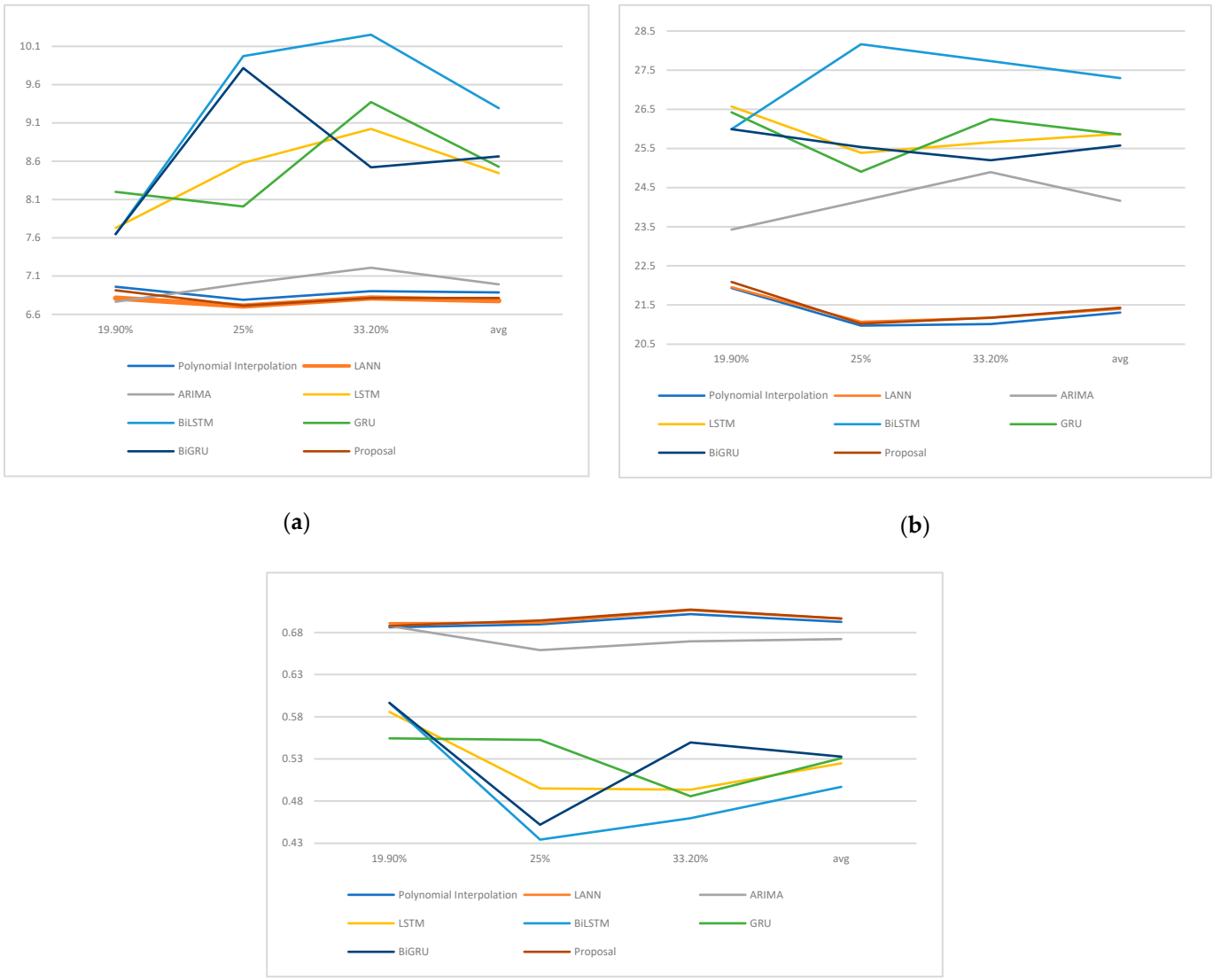
The results achieved by the proposal are described below; likewise, these are compared with other techniques and models of the literature and the state of the art.

According to Table 6 and Figure 14a, it can be seen that, in terms of RMSE, the lowest error was reached by the proposal in one of the three NA sets; it was the best in the third NA set (33.32%) with 6.8134 ug/m^3. For the first NA set (19.98%), the best technique was ARIMA with RMSE = 6.7654 ug/m^3, followed by LANN with RMSE = 6.8123 ug/m^3, and with the proposal with RMSE = 6.9148 ug/m^3 in third place. For the second NA set (25.00%), the best technique was LANN with RMSE = 6.7088 ug/m^3, followed by the proposal with RMSE = 6.7137 ug/m^3, and with polynomial interpolation in third place with RMSE = 6.7912 ug/m^3.

Table 6. RMSEs of implemented models.

Model	19.98% NAs	25.00% NAs	33.32% NAs	Avg
Polynomial Interpolation	6.9916	6.7912	6.9028	6.8852 ± 0.0866
ARIMA	**6.7654**	7.0014	7.2092	6.9920 ± 0.2220
LANN	6.8123	**6.7088**	6.8150	6.7787 ± 0.0606
LSTM	7.7294	8.5795	9.0225	8.4438 ± 0.6571
BiLSTM	7.6487	9.9728	10.2524	9.2913 ± 1.4294
GRU	8.1990	8.0098	9.3725	8.5271 ± 0.7382
BiGRU	7.6487	9.8169	8.5198	8.6618 ± 1.0910
Proposal	6.9148	6.7136	**6.8134**	**6.8139** ± 0.1005

In terms of MAPE, according to Table 7 and Figure 14b, in all sets of NAs, the polynomial interpolation model presented the best results, surpassing all implemented models, including the proposal. For the first NA set (19.98%), LANN was in second place with MAPE = 21.9548, and the proposal was in third place with MAPE = 22.0901. For the second NA set, (25.00%), the proposal was in second place with MAPE = 21.0242%, and in third place was LANN with MAPE = 21.0718. For the third NA set (33.32%), LANN was in second place with MAPE = 21.1710, followed by the proposal with MAPE = 21.1758.



(**a**) (**b**)

(**c**)

Figure 14. Metrics of implemented models. (**a**) RMSE, (**b**) MAPE, and (**c**) R^2.

Table 7. MAPEs of implemented models.

Model	19.98% NAs	25.00% NAs	33.32% NAs	Avg
Polynomial interpolation	**21.9339**	**20.9743**	**21.0103**	21.3061 ± 0.5439
ARIMA	23.4291	24.1574	24.8961	24.1609 ± 0.7335
LANN	21.9548	21.0718	21.1710	21.3992 ± 0.4837
LSTM	26.5726	25.3866	25.6597	25.8730 ± 0.6211
BiLSTM	25.9940	28.1648	27.7320	27.2970 ± 1.1489
GRU	26.4241	24.9054	26.2472	25.8589 ± 0.8305
BiGRU	25.9940	25.5355	25.1984	25.5760 ± 0.3993
Proposal	22.0902	21.0242	21.1759	21.4301 ± 0.5767

In terms of R^2, according to Table 8 and Figure 14c, the proposal model outperformed all benchmark models in two of three NA sets. In the first NA set (19.98%), LANN was the best with R^2 = 0.6911, followed by the proposal and polynomial interpolation with 0.6879 and 0.6861, respectively. In the second NA set (25.00%), the proposal was the best with R^2 = 0.6941, followed by LANN and polynomial interpolation with 0.6916 and 0.6859,

respectively. Finally, in the third NA set, the proposal was also the best with R^2 = 0.7072, followed by LANN and polynomial interpolation with 0.7062 and 0.7018, respectively.

Table 8. R^2s of implemented models.

Model	19.98% NAs	25.00% NAs	33.32% NAs	Avg
Polynomial interpolation	0.6862	0.6896	0.7019	0.6925 ± 0.0083
ARIMA	0.6877	0.6590	0.6721	0.6721 ± 0.0097
LANN	**0.6911**	0.6916	0.7063	0.6963 ± 0.0087
LSTM	0.5857	0.4950	0.4935	0.5248 ±0.0527
BiLSTM	0.5964	0.4344	0.4596	0.4968 ± 0.0872
GRU	0.5545	0.5524	0.4857	0.5309 ± 0.0391
BiGRU	0.5964	0.4519	0.5495	0.5326 ± 0.0737
Proposal	0.6880	**0.6941**	**0.7072**	**0.6964** ± 0.0098

On average, according to the above, there is a notable difference between the proposal and the benchmarks models based on deep learning, such as LSTM, BiLSTM, GRU, and BiGRU; in terms of RMSE, the difference is between 1.6298 ug/m^3 and 2.4773 ug/m^3; in terms of MAPE, it is between 4.1459% and 5.8669%; and in terms of R^2, it is between 16.3818% and 19.9618%.

However, comparing the proposal with benchmark models such as LANN, polynomial interpolation, and ARIMA, the difference is smaller. On average, in terms of RMSE, LANN alone is better than the proposal by a small 0.03527 ug/m^3. In terms of MAPE, only polynomial interpolation and LANN are better than the proposal by 0.1239% and 0.0309%, respectively. In terms of R^2, the proposal is better than polynomial interpolation, LANN, and ARIMA by 0.3898%, 0.0134%, and 2.4341%, respectively.

Graphically, Figure 15 shows the results of the best benchmark models and the proposal for the first 100 items of ground truth for the different sets of NAs.

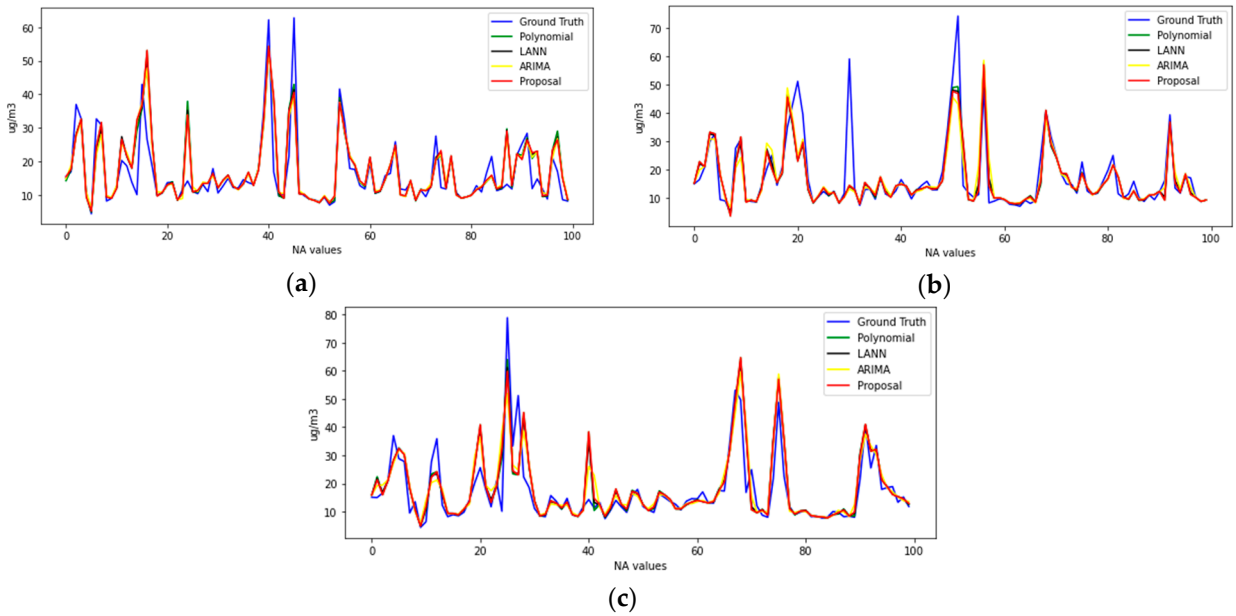


Figure 15. First 100 predicted values: proposal vs. best benchmark models (polynomial interpolation, LANN, and ARIMA). (**a**) NA set 19.98%, (**b**) NA set 25.00%, and (**c**) NA set 33.32%.

4.2. Discussions

Through analysis of the results, it can be seen that the closer models to the proposal model are LANN and polynomial interpolation. Polynomial interpolation served as the basis for the proposal, as it was used to estimate the NA values of class 0. In the proposal, thanks to the implementation of the flipped polynomial interpolation for NA class

1, the proposal outperformed the polynomial interpolation benchmark on average by 0.0712 ug/m^3 and 0.3898% in terms of RMSE and R^2, respectively.

The good results obtained in the estimation of NA values even though the classification models did not achieve good performances, according to Figure 1, prove that both interpolations have similar estimates, so the class estimation errors do not affect them greatly.

Despite what is mentioned in the preceding paragraph, the main weakness of the proposal is the complexity of the classification task to estimate the NA class since it was not possible to identify good input features that have a higher correlation with the target feature. The classification models in most of the evaluation metrics present values below 60%; this can be improved by using a larger amount of data for the training phase and generating or obtaining better input features. A better classification rate would make it possible to identify the NA classes with greater accuracy and, from this, obtain better NA value estimations, reducing the RMSE and MAPE and increasing the R^2 coefficient.

Another aspect that should be analyzed for future work lies in the distance between the real NA value and the value that can be estimated by polynomial interpolation. If the real NA value is further from the polynomial curve, the estimation error will be larger. This constitutes a limitation of this type of technique to estimate more complex values. This type of technique is good for NA values that are close to the line between p0 and n0. For this reason, LANN produces good results, too.

Also, even though in most cases the proposal presented higher R^2s than the benchmark ones, the R^2 coefficient has to be improved, as in two of three NA sets, the proposal presented a value below 0.7, and just in one set, it presented an R^2 above 0.7. As was mentioned in the previous case, improving the classification rate will improve the R^2 of the proposal, too.

On the other hand, considering that related works used other datasets, and each dataset presents different characteristics, the comparison is carried out only for reference. According to Table 9, it can be seen that in terms of RMSE, the proposal is only below the work [20], which obtained an RMSE of 3.756 ug/m^3; in terms of R^2, the proposal with R^2 of 0.6946 is below the work [22], which reported an R^2 of 0.895; and in terms of MAE, the proposal with MAE = 3.4944 ug/m^3 exceeded the work [21] with MAE = 8.31 ug/m^3.

Table 9. Results of related work models.

Work	Technique	Data	Frequency	Metric	Value
Yuan et al., 2018 [17]	LSTM	30,700	Hourly	RMSE	17.78
Belachsen et al., 2022 [18]	KNN	140,256	Half-hourly	NMAE	[0.21–0.26]
Saif-ul-Allah et al., 2022 [19]	GRU	2514	Daily	RMSE	10.60
Alkabbani et al., 2022 [20]	Random forest			RMSE	3.756
Yldz et al., 2022 [21]	Transformers	8760	Hourly	MAE	8.31
Lee et al., 2023 [22]	GAIN	26,281	Hourly	R^2	0.895
Proposal		56,424	Hourly	RMSE	6.8140
				R^2	0.6964
				MAE	3.4944

5. Conclusions and Future Work

5.1. Conclusions

Although the classification rate of the types or classes of NA values did not exceed 60% in most of the analyzed metrics, the estimation results of NA values obtained in terms of RMSE, MAPE, and R^2 are very promising because when compared with the results of benchmark models, the proposal managed to widely outperform the state-of-the-art models, such as LSTM, BiLSTM, GRU, BiGRU and ARIMA, demonstrating that for short gaps, the proposal is a good alternative.

5.2. Future Work

As previously mentioned, this work experimented only with short gaps—gaps of one NA value. For future work, it would be important to adapt the proposal for gaps of more than one value. Likewise, as mentioned above, the classification models only reached accuracies, recalls, precisions, and f1-scores below 60%, which indicates that there is still a wide margin for improvement; in this sense, other architectures of deep learning could be implemented, offering a greater number of layers and different configurations of neurons, among others. Also, the dataset could be enriched through data augmentation techniques for time series classification in order to generate a greater diversity of rows that could help to improve the performance of the models. Also, LANN could be exploited through the creation of a new class (2), which could contain the NA values that can be estimated with greater accuracy with linear interpolation instead of polynomial interpolation.

On the other hand, fractal theory could help to find other time series features that could improve the time series classification process. Likewise, the proposal approach of this work can also be adapted for time series of similar contexts, such as pm10, SO^2, etc., and other different ones like meteorology, biology, finance, etc.

Author Contributions: Conceptualization, A.F., H.T.-C. and D.C.-V.; methodology, A.F.; software, A.F.; validation, A.F., H.T.-C. and A.E.-E.; formal analysis, A.E.-E.; investigation, A.F.; resources, D.C.-V.; data curation, A.F. and H.T.-C.; writing—original draft preparation, A.F.; writing—review and editing, H.T.-C. and D.C.-V.; visualization, A.F.; supervision, A.E.-E.; project administration, H.T.-C.; funding acquisition, A.E.-E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding and the APC was funded by Universidad Nacional de Moquegua.

Data Availability Statement: Data are available at "http://fiscamb.oefa.gob.pe/vig_amb/ (accessed on 2 May 2023)" or by contact to the corresponding author.

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

References

1. Spadon, G.; Hong, S.; Brandoli, B.; Matwin, S.; Rodrigues, J.F., Jr.; Sun, J. Pay Attention to Evolution: Time Series Forecasting with Deep Graph-Evolution Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 5368–5384. [CrossRef] [PubMed]
2. Moritz, S.; Bartz-Beielstein, T. imputeTS: Time series missing value imputation in R. *R J.* **2017**, *9*, 207–218. [CrossRef]
3. Peker, N.; Kubat, C. A Hybrid modified deep learning data imputation method for numeric datasets. *Int. J. Intell. Syst. Appl. Eng.* **2021**, *9*, 6–11. [CrossRef]
4. Chen, C.; Xue, X. A novel coupling preprocessing approach for handling missing data in water quality prediction. *J. Hydrol.* **2023**, *617*, 128901. [CrossRef]
5. Oh, J.; Choi, S.; Han, C.; Lee, D.-W.; Ha, E.; Kim, S.; Bae, H.-J.; Pyun, W.B.; Hong, Y.-C.; Lim, Y.-H. Association of long-term exposure to $PM_{2.5}$ and survival following ischemic heart disease. *Environ. Res.* **2023**, *216*, 114440. [CrossRef] [PubMed]
6. Huang, F.; Pan, B.; Wu, J.; Chen, E.; Chen, L. Relationship between exposure to $PM_{2.5}$ and lung cancer incidence and mortality: A meta-analysis. *Oncotarget* **2017**, *8*, 43322–43331. [CrossRef] [PubMed]
7. Su, J.; Ye, Q.; Zhang, D.; Zhou, J.; Tao, R.; Ding, Z.; Lu, G.; Liu, J.; Xu, F. Joint association of cigarette smoking and $PM_{2.5}$ with COPD among urban and rural adults in regional China. *BMC Pulm. Med.* **2021**, *21*, 87. [CrossRef]
8. Bu, X.; Xie, Z.; Liu, J.; Wei, L.; Wang, X.; Chen, M.; Ren, H. Global PM2.5-attributable health burden from 1990 to 2017: Estimates from the Global Burden of disease study 2017. *Environ. Res.* **2021**, *197*, 111123. [CrossRef]
9. Chen, Z.; Liu, P.; Xia, X.; Wang, L.; Li, X. The underlying mechanism of PM2.5-induced ischemic stroke. *Environ. Pollut.* **2022**, *310*, 119827. [CrossRef]
10. Lee, M.; Ohde, S. $PM_{2.5}$ and diabetes in the Japanese population. *Int. J. Environ. Res. Public Health* **2021**, *18*, 6653. [CrossRef]
11. Liu, Q.; Liu, W.; Mei, J.; Si, G.; Xia, T.; Quan, J. A New Support Vector Regression Model for Equipment Health Diagnosis with Small Sample Data Missing and Its Application. *Shock. Vib.* **2021**, *2021*, 6675078. [CrossRef]
12. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
13. Graves, A.; Fernández, S.; Schmidhuber, J. Bidirectional LSTM networks for improved phoneme classification and recognition. In Proceedings of the International Conference on Artificial Neural Networks, Warsaw, Poland, 11–15 September 2005; Lecture Notes in Computer Science; Springer Science and Business Media LLC: Berlin/Heidelberg, Germany, 2005; pp. 799–804. [CrossRef]
14. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555.

15. Flores, A.; Tito, H.; Silva, C. Local average of nearest neighbors: Univariate time series imputation. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 45–50. [CrossRef]
16. Xiao, Q.; Chang, H.H.; Geng, G.; Liu, Y. An Ensemble Machine-Learning Model To Predict Historical $PM_{2.5}$ Concentrations in China from Satellite Data. *Environ. Sci. Technol.* **2018**, *52*, 13260–13269. [CrossRef]
17. Yuan, H.; Xu, G.; Yao, Z.; Jia, J.; Zhang, Y. Imputation of missing data in time series for air pollutants using long short-term memory recurrent neural networks. In Proceedings of the 2018 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Singapore, 8–12 October 2018; pp. 1293–1300. [CrossRef]
18. Belachsen, I.; Broday, D.M. Imputation of Missing $PM_{2.5}$ Observations in a Network of Air Quality Monitoring Stations by a New *k*NN Method. *Atmosphere* **2022**, *13*, 1934. [CrossRef]
19. Saif-Ul-Allah, M.W.; Qyyum, M.A.; Ul-Haq, N.; Salman, C.A.; Ahmed, F. Gated Recurrent Unit Coupled with Projection to Model Plane Imputation for the $PM_{2.5}$ Prediction for Guangzhou City, China. *Front. Environ. Sci.* **2022**, *9*, 816616. [CrossRef]
20. Alkabbani, H.; Ramadan, A.; Zhu, Q.; Elkamel, A. An Improved Air Quality Index Machine Learning-Based Forecasting with Multivariate Data Imputation Approach. *Atmosphere* **2022**, *13*, 1144. [CrossRef]
21. Yldz, A.Y.; Koc, E.; Koc, A. Multivariate Time Series Imputation with Transformers. *IEEE Signal Process. Lett.* **2022**, *29*, 2517–2521. [CrossRef]
22. Lee, Y.S.; Choi, E.; Park, M.; Jo, H.; Park, M.; Nam, E.; Kim, D.G.; Yi, S.-M.; Kim, J.Y. Feature extraction and prediction of fine particulate matter ($PM_{2.5}$) chemical constituents using four machine learning models. *Expert Syst. Appl.* **2023**, *221*, 119696. [CrossRef]
23. Yang, J.; Lai, X.; Zhang, L. Auto-Associative LSTM for Multivariate Time Series Imputation. In Proceedings of the 2022 41st Chinese Control Conference (CCC), Hefei, China, 25–27 July 2022. [CrossRef]
24. Li, D.; Li, L.; Li, X.; Ke, Z.; Hu, Q. Smoothed LSTM-AE: A spatio-temporal deep model for multiple time-series missing imputation. *Neurocomputing* **2020**, *411*, 351–363. [CrossRef]
25. Zaman, M.A.U.; Du, D. A Stochastic Multivariate Irregularly Sampled Time Series Imputation Method for Electronic Health Records. *Biomedinformatics* **2021**, *1*, 166–181. [CrossRef]
26. Zhang, W.; Luo, Y.; Zhang, Y.; Srinivasan, D. SolarGAN: Multivariate solar data imputation using generative adversarial network. *IEEE Trans. Sustain. Energy* **2020**, *12*, 743–746. [CrossRef]
27. Cao, W.; Zhou, H.; Wang, D.; Li, Y.; Li, J.; Li, L. BRITS: Bidirectional recurrent imputation for time series. In Proceedings of the Advances in Neural Information Processing Systems 31 (NeurIPS 2018), Montréal, QC, Canada, 3–8 December 2018.
28. Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; Liu, Y. Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Sci. Rep.* **2018**, *8*, 6085. [CrossRef]
29. Guo, Y.; Poh, J.W.J.; Wong, C.S.Y.; Ramasamy, S. Bayesian Continual Imputation and Prediction For Irregularly Sampled Time Series Data. In Proceedings of the ICASSP 2011—IEEE International Conference on Acoustics, Speech and Signal Processing, Singapore, 23–27 May 2022; pp. 4493–4497. [CrossRef]
30. Brownlee, J. Ensemble Learning Algorithms with Python. Machine Learning Mastery. 2021.