mathematics

MDPI

Article

Reconstruction and Prediction of Chaotic Time Series with Missing Data: Leveraging Dynamical Correlations Between Variables

Jingchan Lv [1,2], Hongcun Mao [1], Yu Wang [1] and Zhihai Yao [1,*]

[1] Department of Physics, Changchun University of Science and Technology, Changchun 130022, China; ljingchan@mails.cust.edu.cn (J.L.); 2020200021@mails.cust.edu.cn (H.M.); 2021200025@mails.cust.edu.cn (Y.W.)
[2] College of Physics and Electronic Information, Baicheng Normal University, Baicheng 137000, China
* Correspondence: yaozh@cust.edu.cn

Abstract: Although data-driven machine learning methods have been successfully applied to predict complex nonlinear dynamics, forecasting future evolution based on incomplete past information remains a significant challenge. This paper proposes a novel data-driven approach that leverages the dynamical relationships among variables. By integrating Non-Stationary Transformers with LightGBM, we construct a robust model where LightGBM builds a fitting function to capture and simulate the complex coupling relationships among variables in dynamically evolving chaotic systems. This approach enables the reconstruction of missing data, restoring sequence completeness and overcoming the limitations of existing chaotic time series prediction methods in handling missing data. We validate the proposed method by predicting the future evolution of variables with missing data in both dissipative and conservative chaotic systems. Experimental results demonstrate that the model maintains stability and effectiveness even with increasing missing rates, particularly in the range of 30% to 50%, where prediction errors remain relatively low. Furthermore, the feature importance extracted by the model aligns closely with the underlying dynamic characteristics of the chaotic system, enhancing the method's interpretability and reliability. This research offers a practical and theoretically sound solution to the challenges of predicting chaotic systems with incomplete datasets.

Keywords: chaotic time series prediction; missing data reconstruction; multivariate time series; dynamical correlation; machine learning

MSC: 37M10

check for updates

Academic Editor: Li-pang Chen

Received: 26 November 2024
Revised: 30 December 2024
Accepted: 1 January 2025
Published: 3 January 2025

Citation: Lv, J.; Mao, H.; Wang, Y.; Yao, Z. Reconstruction and Prediction of Chaotic Time Series with Missing Data: Leveraging Dynamical Correlations Between Variables. *Mathematics* **2025**, *13*, 152. https://doi.org/10.3390/math13010152

Copyright: © 2025 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

Chaotic time series research is fundamental to scientific computing, as it captures the nonlinear dynamical behaviors intrinsic to complex systems. Since the 1990s, machine learning techniques have been increasingly applied to chaotic time series prediction, resulting in diverse methods, such as genetic algorithms [1], neuro-fuzzy systems [2], wavelet neural networks [3], nonlinear autoregressive models [4], Hybrid Connected Complex Neural Networks (HCNNs) [5], ant colony optimization [6], Monte Carlo cross-validation [7], and scaled UKF-NARX hybrid models [8]. Advances in computational power and data processing have enabled models like LSTM neural networks [9], Echo State Networks (ESNs) [10,11], Physics-Informed Neural Operators [12], Transformers [13], DeepVar [14], and Non-Stationary Transformers (NSTs) [15] to effectively capture the intricate dynamics

of chaotic time series. Among these, NSTs have shown exceptional capability in addressing non-stationarity through series stationarization and de-stationary attention. In recent years, the prediction of multi-dimensional or high-dimensional chaotic time series has also garnered widespread attention. For example, Liu [16] proposed a hybrid Proper Orthogonal Decomposition (POD) and Next-Generation Reservoir Computing (NGRC) method, Xiong [17] developed the Dynamic Adaptive Graph Convolutional Transformer (DGCT) model, and Fu et al. [18] introduced the CTF-former model. These approaches have provided new insights into multi-task prediction of chaotic time series.

In these applications, a common default assumption is that the input data are complete. However, in practice, missing data are frequently encountered due to sensor failures, storage device malfunctions, network interruptions, or natural disasters, leading to incomplete datasets [19]. The continuity of time series and the contextual information required by models are severely disrupted. The loss of information makes it difficult to capture long-term dependencies, which is detrimental to any model attempting to extract patterns from the data. When the proportion of missing data exceeds 5%, the model's predictions may become biased, and when the missing rate surpasses 15%, the model's performance deteriorates significantly [20–22].

Researchers have explored various methods for managing missing data in chaotic time series prediction. Jerez [23] demonstrated the effectiveness of neural network models in breast cancer diagnosis using three statistical and three machine learning approaches for missing data. Similarly, Laña [24] showed that machine learning-based imputation methods enhance traditional traffic congestion prediction models. Although random sampling approximation techniques [25] have achieved good results, they require large sample sizes to ensure accuracy. The FI-GEM network [26] performs well with incomplete time series but suffers from extended training times and high computational demands in large-scale or high-dimensional datasets. Zhao Pengcheng [27] proposed an auxiliary variable method to address truncated chaotic missing data. Rangel et al. [28] utilized artificial neural networks and k-Nearest Neighbor (k-NN) techniques to reconstruct and predict incomplete chaotic time series, achieving significant improvements even at missing rates exceeding 7%.

However, current research primarily relies on statistical and mathematical modeling to handle missing data. Traditional imputation methods, such as mean or median filling and simple regression models, leverage statistical properties. Although easy to apply, these methods may result in information loss or introduce significant bias [29]. In contrast, algorithms such as k-Nearest Neighbor, Expectation-Maximization, Matrix Factorization, and Multiple Imputation using Chained Equations [20,30–32] consider multiple influencing parameters of the real system and their interrelationships as comprehensively as possible, thereby reducing imputation bias. Recently, interpolation models based on Generative Adversarial Networks (GANs) [33] have achieved higher accuracy; however, their training process may encounter issues such as mode collapse and difficulty in convergence. The SAITS model [34] improved interpolation performance through a self-attention mechanism, but it exhibits weak capabilities in modeling nonlinear relationships and is highly sensitive to hyperparameter selection.

In summary, the current methods for handling chaotic time series with missing data still have significant shortcomings, including: (1) insufficient capability for dynamic modeling of missing data, making it difficult to fully capture the temporal dependencies and coupling relationships between variables; (2) low computational efficiency or excessive sensitivity to model parameters, making them unsuitable for large-scale or high-dimensional scenarios. These issues significantly limit the broad applicability of chaotic time series prediction in practical applications.

A chaotic system is a dynamic system that evolves over time. Compared to general time series, the variation of each state variable depends not only on its own values but also on the influence of other state variables. We argue that when addressing the problem of missing data in chaotic time series, fully considering the dynamical characteristics of the chaotic sequence—starting from time-delay embedding vectors and the topological features of the reconstructed system [35–38], and leveraging the dynamical relationships among variables—should yield better results than the aforementioned algorithms, such as k-Nearest Neighbor.

Based on the above considerations, this study proposes a new framework called LNS_PM (the integration of LightGBM and Non-Stationary Transformers for the prediction model) for handling prediction tasks of chaotic time series with random missing data at high missing rates (30–50%). The framework preprocesses missing data using LightGBM [39] by constructing a fitting function $f(x)$ to capture and simulate the complex coupling relationships among variables within the chaotic system that dynamically evolve over time. The fitting function $f(x)$ is then used to reconstruct variables with missing data, restoring their completeness and providing a complete dataset for subsequent time series prediction using NST. This method not only considers the temporal dependencies of chaotic time series data but also fully leverages the nonlinear interactions between variables, thereby significantly improving the accuracy and reliability of system behavior predictions. It effectively addresses the limitations of existing chaotic time series prediction methods in handling missing data and achieves direct prediction of incomplete chaotic time series, ensuring consistency and efficiency from input to output.

2. Model

As illustrated in Figure 1, the LNS_PM framework proposed in this paper comprises two primary modules: LightGBM and NST. These modules are responsible for missing data imputation and time series prediction. The model exploits the dynamical correlations between variables, effectively reconstructing the missing data in chaotic time series and facilitating high-precision predictions.

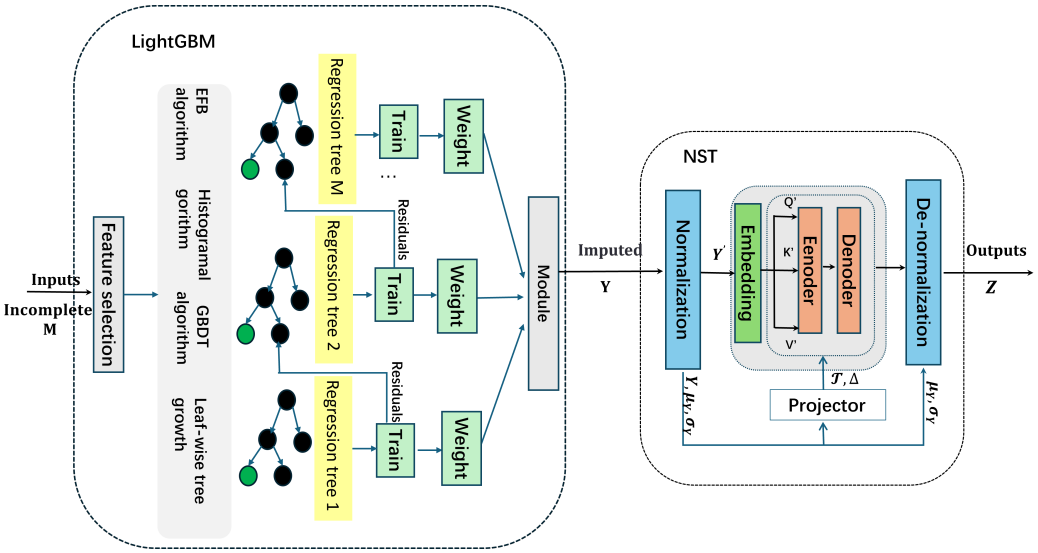


Figure 1. Framework of the LNS_PM.

By setting a series of discrete time points (time vector t_{sim}, of length p), we numerically solve the chaotic dynamical equations at each time step Δt_{sim}, obtaining approximate values of the system states. This process forms a matrix with with p time steps and $q-1$ state variables, denoted as $p \times (q-1)$. The time vector t_{sim} is then added as an

additional column to this matrix, resulting in a data matrix $M_{p\times q}$, which contains both time information and state variables, Here, the first column represents the time information, while the remaining $q-1$ columns correspond to the system's state variable values at the respective time points.To simulate missing data, one column of the system's state variables is randomly set to be completely missing, with a missing rate ranging from 10% to 50%.

$$\mathbf{M} = \begin{bmatrix} t_1 & x_{1,1} & x_{1,2} & \cdots & x_{1,q-1} \\ t_2 & x_{2,1} & x_{2,2} & \cdots & x_{2,q-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_p & x_{p,1} & x_{p,2} & \cdots & x_{p,q-1} \end{bmatrix}$$

In the phase dedicated to reconstructing the missing data, the primary objective is to formulate a fitting function $f(\mathbf{x})$ that can precisely predict the target variable based on the provided input variables. This reconstruction employs LightGBM, which builds decision trees to model and capture the nonlinear interdependencies intrinsic to the data. This process initiates with the data matrix decomposition, laying the groundwork for subsequent modeling and analysis.

We define an indicator vector $\mathbf{I} = (I_1, I_2, \ldots, I_p)^T$ to identify rows containing missing values. Specifically, if the i-th row in $\mathbf{M}$ has a missing value in the designated column, then $I[i] = 1$; otherwise, $I[i] = 0$. Based on this indicator vector, we define two index sets: $\mathcal{I}_{\text{miss}}$ and $\mathcal{I}_{\text{obs}}$, $\mathcal{I}_{\text{miss}} = \{i \in \{1, 2, \ldots, p\} : I_i = 1\}$, representing the indices of rows with missing values, and $\mathcal{I}_{\text{obs}} = \{i \in \{1, 2, \ldots, p\} : I_i = 0\}$, representing the indices of rows without missing values. Using these index sets, we separate the data matrix $\mathbf{M}$ into two submatrices: $\mathbf{M}_{\text{miss}} = (M[i, :])_{i\in\mathcal{I}_{\text{miss}}}$, which contains rows with missing values, and $\mathbf{M}_{\text{obs}} = (M[i, :])_{i\in\mathcal{I}_{\text{obs}}}$, which contains rows without missing values. Here, $\mathbf{M}_{\text{miss}}$ consists of the rows in M indexed by $\mathcal{I}_{\text{miss}}$, while $\mathbf{M}_{\text{obs}}$ consists of the rows in M indexed by $\mathcal{I}_{\text{obs}}$.

To maintain the continuity of indices in the new matrices, we reindex $\mathbf{M}_{\text{obs}}$ and assume it consists of m rows and $n+1$ columns. Next, $\mathbf{M}_{\text{obs}}$ is divided into two matrices: $\mathbf{X}$ and $\mathbf{y}$. The matrix $\mathbf{X}$ represents the input feature vector matrix, containing m samples and n features, while $\mathbf{y}$ corresponds to the vector of m labels.

$$\mathbf{M}_{\text{obs}} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} & y_1 \\ x_{21} & x_{22} & \cdots & x_{2n} & y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} & y_m \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

We consider a feature vector x_{il}, where $i = 1, 2, \ldots, m$ and $l \in (1, n)$, as an example to illustrate the process of reconstructing missing values. The reconstruction process begins by initializing $f_0(x_l)$ as:

$$f_0(x_l) = \arg\min_c \sum_{i=1}^{m} (y_i - c)^2, \tag{1}$$

where c is a constant representing the mean value of the target variable y. Subsequently, the Gradient Boosted Decision Tree (GBDT) algorithm with a leaf-wise learning strategy is applied to construct the fitting function. Splits are performed on different values s of the feature vector x_{il}. For a given split point s, the loss function is defined as:

$$L(s) = \sum_{i\in R_L(s)} (r_{ti})^2 + \sum_{i\in R_R(s)} (r_{ti})^2, \tag{2}$$

where $R_L(s)$ and $R_R(s)$ represent the subsets of samples split by s. The residual r_{ti}, representing the negative direction of the gradient, is defined as:

$$r_{ti} = -\frac{\partial(y_i - f(x_{il}))^2}{\partial f(x_{il})}\Big|_{f(x_{il}) = f_{t-1}(x_{il})} = 2(y_i - f_{t-1}(x_{il})). \quad (3)$$

Here, y_i is the target value and $f_{t-1}(x_{il})$ is the prediction value of the model for the sample in the $t-1$ iteration. The residual is calculated as the difference between y_i and $f_{t-1}(x_{il})$, reflecting the model's error at the current iteration. Using (x_{il}, r_{ti}), the t-th regression tree is fitted, and the optimal split point s^* is chosen to minimize the loss: $s^* = \arg\min_s L(s)$. Based on s^*, each leaf node's region is denoted as $R_{tj}, j = 1, 2, \ldots, J$. For each leaf region j, the optimal fitting value c_{tj} minimizes the loss within that region:

$$c_{tj} = \arg\min_c \sum_{x_{il} \in R_{tj}} (y_i - f_{t-1}(x_{il}) - c)^2. \quad (4)$$

where R_{tj} represents the set of samples in leaf node j. By minimizing the squared loss for each leaf node, the model can dynamically adjust the leaf node predictions to accommodate the local distribution characteristics of missing data. By setting the derivative of Equation (4) to zero, we find:

$$c_{tj} = \frac{1}{|R_{tj}|} \sum_{x_{il} \in R_{tj}} (y_i - f_{t-1}(x_{il})). \quad (5)$$

Here, $|R_{tj}|$ denotes the number of samples in the leaf node region. This gradient-based optimization mechanism enables LightGBM to progressively approach the true distribution of missing values with each iteration. The fitting function for the t-th regression tree is:

$$f_t(x_l) = f_{t-1}(x_l) + \sum_{j=1}^{J} c_{tj} I_{leaf}(x_l \in R_{tj}), \quad (6)$$

where $I_{leaf}(x_l \in R_{tj})$ is an indicator function that determines whether a sample belongs to region R_{tj}. After T iterations, the final model combining all regression trees is:

$$f(x_l) = f_0(x_l) + \sum_{t=1}^{T} \sum_{j=1}^{J} c_{tj} I_{leaf}(x_l \in R_{tj}). \quad (7)$$

When dealing with high-dimensional variables that contain multiple sparse features, using the Exclusive Feature Bundling (EFB) algorithm alongside the histogram algorithm can significantly reduce memory consumption and improve the efficiency of feature splitting. The EFB algorithm works by bundling mutually exclusive features—those where only one feature can have a non-zero value for any given sample—into a single new feature, thus reducing the total number of features. For each feature's split, the histogram algorithm divides continuous features into a fixed number of buckets, where each bucket represents a range of continuous values. This mapping of feature values to bucket indices simplifies the search for split points, reducing computational complexity. Following this approach, we construct a function to model the relationship between inputs and outputs. By inputting the feature variables of $\mathbf{M}_{\text{miss}}$, we obtain the corresponding target variable, which represents the imputed missing values. By merging the imputed data with the original dataset and sorting them by time, we obtain the complete dataset $\mathbf{Y}$. Furthermore, during the decision tree generation process, the model can assess feature importance based on the performance of input features during training.

The completed data $\mathbf{Y}$ are passed to the NST module for time series prediction. First, $\mathbf{Y}$ undergoes a normalization process, transforming the raw data into a standardized form with mean $\mu_{\mathbf{Y}}$ and standard deviation $\sigma_{\mathbf{Y}}$, resulting in the normalized sequence $\mathbf{Y}'$:

$$\mathbf{Y}' = \frac{\mathbf{Y} - 1\mu_{\mathbf{Y}}^{\top}}{\sigma_{\mathbf{Y}}} \tag{8}$$

We begin with the original attention formula [15]:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\sigma_{\mathbf{Y}}^2 \mathbf{Q}'\mathbf{K}'^{\top} + 1\mu_{\mathbf{Q}}^{\top}\mathbf{K}^{\top}}{\sqrt{d_k}}\right) \tag{9}$$

Here, $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ represent the query, key, and value matrices computed from the input sequence $\mathbf{Y}$, respectively. The normalized matrices are defined as $\mathbf{Q}' = \frac{\mathbf{Q} - 1\mu_{\mathbf{Q}}^{\top}}{\sigma_{\mathbf{Y}}}$, where $\mu_{\mathbf{Q}}$ is the mean of $\mathbf{Q}$ along the temporal dimension. Similarly, $\mathbf{K}'$ and $\mathbf{V}'$ are computed in the same manner. The term d_k denotes the dimension of the key vectors.

In addition to the normalized components $\mathbf{Q}'$ and $\mathbf{K}'$ derived from the stationary sequence $\mathbf{Y}'$, the attention formula also incorporates non-stationary information, such as $\sigma_{\mathbf{Y}}$, $\mu_{\mathbf{Q}}$, and the unnormalized $\mathbf{K}$, which are partially removed during the normalization process. To capture the non-stationary properties of the input chaotic time series, the formula introduces two de-trending factors: $\tau = \sigma_{\mathbf{Y}}^2 \in \mathbb{R}^{+}$ and $\Delta = \mathbf{K}\mu_{\mathbf{Q}} \in \mathbb{R}^{m \times 1}$. Given that strict linear assumptions rarely hold for deep models, the de-trending factors are learned directly from the statistical features of the unnormalized sequence $\mathbf{Y}$ using a simple yet effective multi-layer perceptron (MLP): $\log \tau = \text{MLP}(\sigma_{\mathbf{Y}}, \mathbf{Y})$, $\Delta = \text{MLP}(\mu_{\mathbf{Y}}, \mathbf{Y})$. The final formula for non-stationary attention is as follows:

$$\text{Attn}(\mathbf{Q}', \mathbf{K}', \mathbf{V}', \tau, \Delta) = \text{Softmax}\left(\frac{\tau \mathbf{Q}'\mathbf{K}'^{\top} + \Delta^{\top}}{\sqrt{d_k}}\right)\mathbf{V}' \tag{10}$$

The prediction result obtained after processing through the encoder-decoder structure is represented as the standardized sequence $\mathbf{Z}'$. It is then transformed back to the original time series scale through a de-normalization step, resulting in the final output $\mathbf{Z}$:

$$\mathbf{Z} = \sigma_{\mathbf{Y}}(\mathbf{Z}' + \mu_{\mathbf{Y}}). \tag{11}$$

The advantage of this modular design lies in that the LightGBM module effectively imputes missing values by using the weighted residuals of decision trees. This approach allows the model to leverage correlations between different feature variables, selecting the most predictive features to construct the tree branches. The multi-dimensional feature correlations enable the model to use information from related variables to impute missing values, thereby reconstructing complete data that adhere to the system's dynamical rules, ensuring accuracy and robustness in data imputation. On the other hand, the NST module addresses complex time series features and non-stationarity through the self-attention mechanism, avoiding over-smoothing. This mechanism allows the NST to capture dynamic changes in time series data, thereby enhancing the model's prediction accuracy and robustness.

3. Experiments

3.1. Chaotic Systems

In this section, we use four chaotic dynamical systems—the Lorenz system [40], the Hyperchaotic Lorenz system [41], the Coupled Lorenz system [42], and the Conservative Chaotic system [43]—to illustrate the LNS_PM method described in Section 2. We also compare its performance with other classical prediction models to verify its advantages.

The dimensions of the Lorenz system, Hyperchaotic Lorenz system, and Coupled Lorenz system progressively increase, reflecting the growing complexity of their chaotic behavior. This gradual progression allows for a comprehensive evaluation of the model's predictive capabilities under varying levels of complexity and dimensionality, as well as the influence of the number of feature variables on the model's performance. In contrast to relatively dissipative systems, chaos control in conservative and spatiotemporal systems is more challenging. Conservative systems, characterized by energy conservation, exhibit more complex and unpredictable dynamical behavior. In the aerospace field, the principles and methods of chaos control in conservative systems have direct and practical applications.

1. Lorenz System

The Lorenz system, proposed by the American meteorologist Edward Lorenz in 1963, is a model describing thermal convection instability. It exhibits key chaotic characteristics, including sensitivity to initial conditions and non-periodicity, making it an ideal candidate for evaluating time series prediction models. The system is governed by the following equations:

$$\begin{cases} \dot{x} = -\sigma(x - y), \\ \dot{y} = -xz + rx - y, \\ \dot{z} = xy - bz. \end{cases} \tag{12}$$

where $\sigma = 10$, $b = 8/3$, and $r = 28$. Under these conditions, the system exhibits chaotic behavior. The Lorenz system is deterministic, and governed by known parameters, which ensures repeatable dynamics under identical initial conditions. In this study, we use the classical Runge–Kutta method to solve the equations, with initial values set to $(1.01, 1.01, 0.00)$, resulting in a dataset comprising a 3×10^5 matrix.

2. Hyperchaotic Lorenz System

The Hyperchaotic Lorenz system, due to its multiple positive Lyapunov exponents, exhibits stronger resistance to interference compared to general chaotic systems, making its dynamic behavior more difficult to predict. The system can be expressed as follows:

$$\begin{cases} \dot{x} = a(y - x) + w, \\ \dot{y} = cx - y - xz, \\ \dot{z} = xy - bz, \\ \dot{w} = -yz + rw. \end{cases} \tag{13}$$

When the parameters are $a = 10$, $b = 8/3$, $c = 28$, and $-1.52 < r < -0.06$, the system exhibits hyperchaotic behavior. Specifically, when $r = -1$, the system has four Lyapunov exponents: $\lambda_1 = 0.3381$, $\lambda_2 = 0.1586$, $\lambda_3 = 0$, $\lambda_4 = -15.1752$. In this study, the initial values are set to $(1.1, 2.2, 3.3, 4.4)$, resulting in a dataset matrix of $4 \times 12,485$.

3. Coupled Lorenz System

To verify the model's predictive capabilities on higher-dimensional nonlinear systems with missing data, this study uses the Coupled Lorenz system model to simulate interactions among subsystems. The dynamic state of each subsystem k is represented by the following differential equations:

$$\begin{cases} dx_k/dt &= -10\left[x_k - y_k + c\sum_{l=1}^{N} a_{kl}^{(x,y)}(y_l - y_k)\right], \\ dy_k/dt &= 28(1 + h_k)x_k - y_k - x_k z_k, \\ dz_k/dt &= -(8/3)z_k + x_k y_k. \end{cases} \tag{14}$$

where c = 1, N = 10, and the coupling matrix $a_{kl}^{(x,y)}$ is randomly generated as 0 or 1, representing whether there is a coupling between subsystems k and l. The values of h_k are randomly generated between -0.25 and 0.25 to simulate the dynamic differences between each subsystem. The initial state of each subsystem is generated randomly within the range $[0, 0.1]$, resulting in a dataset of a $30 \times 40,000$ matrix.

4. Conservative chaotic system

To validate the generalization capability of the proposed model on Conservative Chaotic systems, this study uses a four-dimensional Conservative Chaotic System (CCS) that exhibits infinite-scroll characteristics.

$$\begin{cases} \dot{x}_1 &= ax_2, \\ \dot{x}_2 &= b\sin(x_1)x_4 - a\sin(x_1), \\ \dot{x}_3 &= cx_4 \\ \dot{x}_4 &= -b\sin(x_1)x_2 - cx_3. \end{cases} \tag{15}$$

When appropriate initial values and control parameters are selected, the system exhibits a multi-scroll phenomenon. For example, by setting $(a, b, c) = (6, 5.57, 6)$ and initial values $(1.9, 1, 1, 1)$, we conducted simulations at various times: $t = 100\,\text{s}$, $t = 200\,\text{s}$, $t = 400\,\text{s}$, $t = 600\,\text{s}$, and $t = 6000\,\text{s}$. The corresponding phase trajectories display 7, 9, 11, 12, and 56 scrolls, respectively. These simulations demonstrate that the number of scrolls increases with longer simulation times. Theoretically, as time approaches infinity, the system will exhibit an infinite number of scrolls along the x_1-axis. From an equilibrium perspective, the central equilibrium points and unstable saddle points alternate along the x_1-axis. Despite the increasing number of scrolls, the system's Hamiltonian energy remains constant at $H_3 = 1.823289$. This characteristic not only tests the model's ability to learn complex dynamic behaviors but also evaluates its effectiveness in handling chaotic phenomena observed in real-world physical systems. In this study, at $t = 200\,\text{s}$, the equations were solved using the fourth-order Runge–Kutta method, resulting in a $2 \times 100,000$ matrix.

3.2. Dataset

For each chaotic system, datasets are generated based on the system parameters specified in Section 3.1. We randomly set the target variable with missing data at rates of 10%, 20%, 30%, 40%, and 50%. These datasets are classified into training (65% of the samples), validation (15% of the samples), and test sets (20% of the samples). The training set allows the optimization of model parameters, such as weights and biases. The validation set, in contrast, is not directly involved in the optimization process but is used to fine-tune the model's hyperparameters. Table 1 summarizes the specific hyperparameters for the LNS_PM model. Here, Lr is the learning rate, λ_1 and λ_2 are the regularization coefficients controlling overfitting, ml is the minimum number of samples required in a leaf node, nl is the number of leaves used in LightGBM, ep is the number of training epochs, bs is the batch size for each training iteration, hl is the number of hidden layers in the projector, hd is the hidden layer dimension of the projector, and es is the early stopping criterion, which ensures the training process terminates when performance stops improving.

The hyperparameter hd represents the hidden layer dimension of the MLP-based projector responsible for capturing these factors. To balance the efficiency of hyperparameter tuning and model performance, we fixed the number of hidden layers at 2 and varied the hidden layer dimension within 64, 128, 256. Each experiment was repeated using different random seeds to ensure robustness, with lower RMSE/MAPE values indicating better performance. This design allows the projector to effectively model non-stationary dynamics while maintaining computational efficiency.

Table 1. Hyperparameters of LNS_PM model components.

Module	**Lr**	**es**	λ_1	λ_2	**ml**	**nl**	**ep**	**bs**
LightGBM	0.01	50	0.5	0.5	20	2^4	-	-
NST	0.0008	100	-	-	-	-	100	256
Additional NST Hyperparameters:								
hl	2 (number of hidden layers in projector)							
hd	128 (hidden layer dimension of the projector)							

In model identification, the test set is not used for training the model but is exclusively employed to evaluate the model's performance in inference mode. We included the initial state data generated by the chaotic equations as part of the training data to account for the influence of the transient on model training. This approach ensures that the training data capture the system's long-term and transient behavior, enabling the model to learn richer dynamic characteristics. We set random seeds for tree iteration, split-point selection, and global settings to enhance the stability and reproducibility of the results. This procedure ensures that data partitioning, feature selection, and other random processes remain consistent each time the code runs. Additionally, the model incorporates an early stopping mechanism. The training process ends when the model's performance on the validation set does not significantly improve over consecutive iterations. This method avoids unnecessary iterations and prevents overfitting.

Using these datasets, the performance differences between LNS_PM and other models, such as NST, DeepVar, LSTM, and ESN, can be investigated when handling complex, chaotic systems with varying degrees of missing data. Table 2 presents the parameter settings for these models. In this table, rs refers to the reservoir size, sr denotes the spectral radius, sp represents sparsity, and ts corresponds to Teacher Forcing, which we used during the training phase. Appropriate scaling of these parameters can help stabilize the training process.

Table 2. Hyperparameters of the comparison models.

	Lr	***ep***	***bs***	***hd/hs***	***rs***	***sr/sp/ts***
NST	0.0001	500	256	128	–	–
DeepVar	0.01	100	128	30	–	–
LSTM	0.001	100	32	30	–	–
ESN	–	–	–	–	500	1/0.45/1.12

3.3. Performance Metrics

We use the Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) as metrics to evaluate the effectiveness of the LNS_PM model. RMSE measures the absolute error between predicted and actual values, while MAPE quantifies the relative error, making it suitable for comparing data of different scales. Lower values of RMSE and MAPE indicate better model performance. To further assess whether the performance differences among various prediction models across multiple datasets are statistically significant, we employ analysis of variance (ANOVA). This method evaluates the statistical significance of performance differences between the models.

$$\text{RMSE} = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(z_i - \hat{z}_i)^2}, \quad \text{MAPE} = \frac{100\%}{n}\Sigma_{i=1}^{n}\left|\frac{z_i - \hat{z}_i}{z_i}\right|. \tag{16}$$

In addition, to quantify the uncertainty of the predictive performance of different models under varying data missing rates, we also calculated the confidence interval (CI) of

the prediction mean at a 95% confidence level. The calculation of the confidence interval helps to understand the stability and reliability of the model's prediction results, where a narrower confidence interval indicates more stable predictive performance. The formula for calculating the confidence interval is as follows:

$$CI = \bar{x} \pm t_{\alpha/2,df} \cdot \left(\frac{s}{\sqrt{n}} \right) \tag{17}$$

where $\bar{x}$ is the sample mean; $t_{\alpha/2,df}$ is the t-value corresponding to the chosen confidence level (e.g., 95%), with $df = n - 1$ as the degrees of freedom; s is the sample standard deviation; n is the sample size. By using this formula, the confidence interval at the 95% confidence level can be calculated, which facilitates statistical validation of the predictive model's performance.

3.4. Results Analysis

We analyzed the predictive performance of five models (LNS_PM, NST, DeepVar, LSTM, and ESN) on the chaotic systems described in Section 3.1, evaluating their performance under missing data rates ranging from 10% to 50%. For greater clarity, we plotted only the last 50 points of the predicted data (see Figure 2). Each system's results are displayed in five subplots corresponding to the different missing rates (10%, 20%, 30%, 40%, and 50%). Each subplot includes six curves representing the true values and the predictions of the LNS_PM, NST, LSTM, ESN, and DeepVar models.

From these curves, it is evident that as the missing rate increases, the discrepancy between the predicted results of all models and the actual observed values becomes more pronounced. Specifically, DeepVar demonstrates significant bias at some time points, while LSTM and ESN show particularly large deviations from the observed values. In contrast, LNS_PM and NST generally track the trends of the actual data more closely. However, at missing rates between 30% and 50%, NST exhibits a noticeable bias between its predictions and the true values. Conversely, LNS_PM consistently maintains lower fluctuations and bias, producing smoother curves that align closely with the observed values, even in regions of frequent small fluctuations.

Figure 2 also reveals that under identical missing rate conditions, reducing the amount of training data decreases prediction accuracy while increasing the number of feature variables improves predictive performance. Overall, LNS_PM demonstrates superior predictive accuracy across chaotic systems of varying dimensions and missing rates. It effectively captures the data trends and exhibits a distinct advantage in chaotic prediction with missing data.

Another perspective for evaluating the results is through scatter plots, which depict the relationship between each pair of predicted and actual values. The closer the points are to the 45-degree line (where predicted values equal actual values), the more accurate the model's predictions.

Figures 3–6 present scatter plots for the predictions from five different models for the Lorenz, Hyperchaotic Lorenz, Coupled Lorenz, and Conservative Chaotic systems, respectively. Figure 3 shows that testing different prediction models on the same dataset with identical missing rates reveals that LNS_PM delivers highly stable prediction results. Most points align closely with a straight line, indicating a strong linear relationship between the predicted and actual values and suggesting that the model effectively captures the underlying data trends. NST also demonstrates relatively stable predictions but with slightly larger discrepancies between the predicted and actual values compared to LNS_PM, which may indicate a degree of bias. DeepVar exhibits significant volatility in its predictions, with notable deviations between the predicted and actual values. Similarly, predictions

from LSTM and ESN show fluctuations, with substantial discrepancies in certain cases, reflecting instability in their performance.

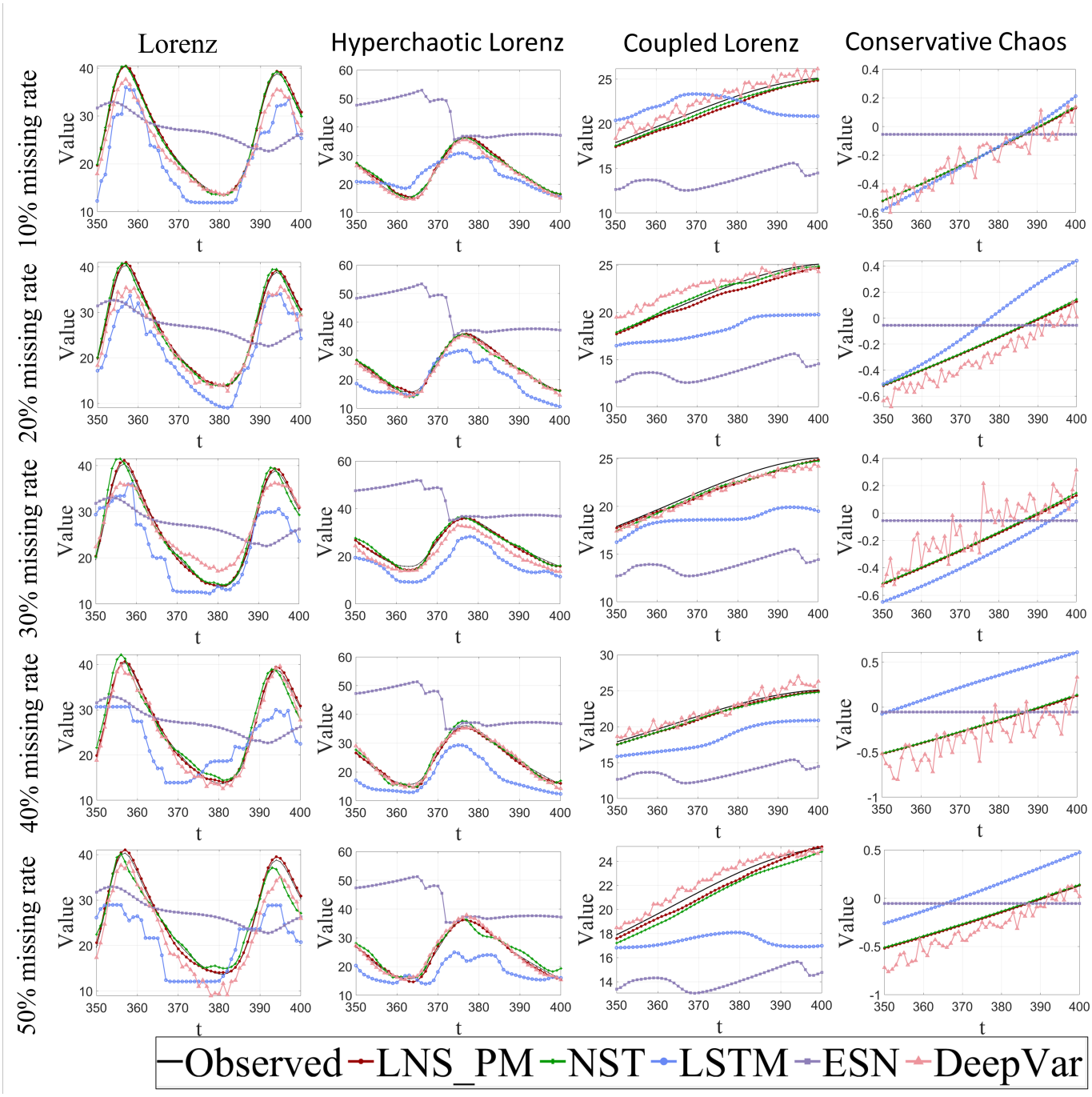


Figure 2. Prediction results of five forecasting models (colored lines) for the Lorenz system, Hyperchaotic Lorenz system, Coupled Lorenz system, and Conservative Chaotic system (columns 1 to 4), with missing rates ranging from 10% to 50% (raws 1 to 5).

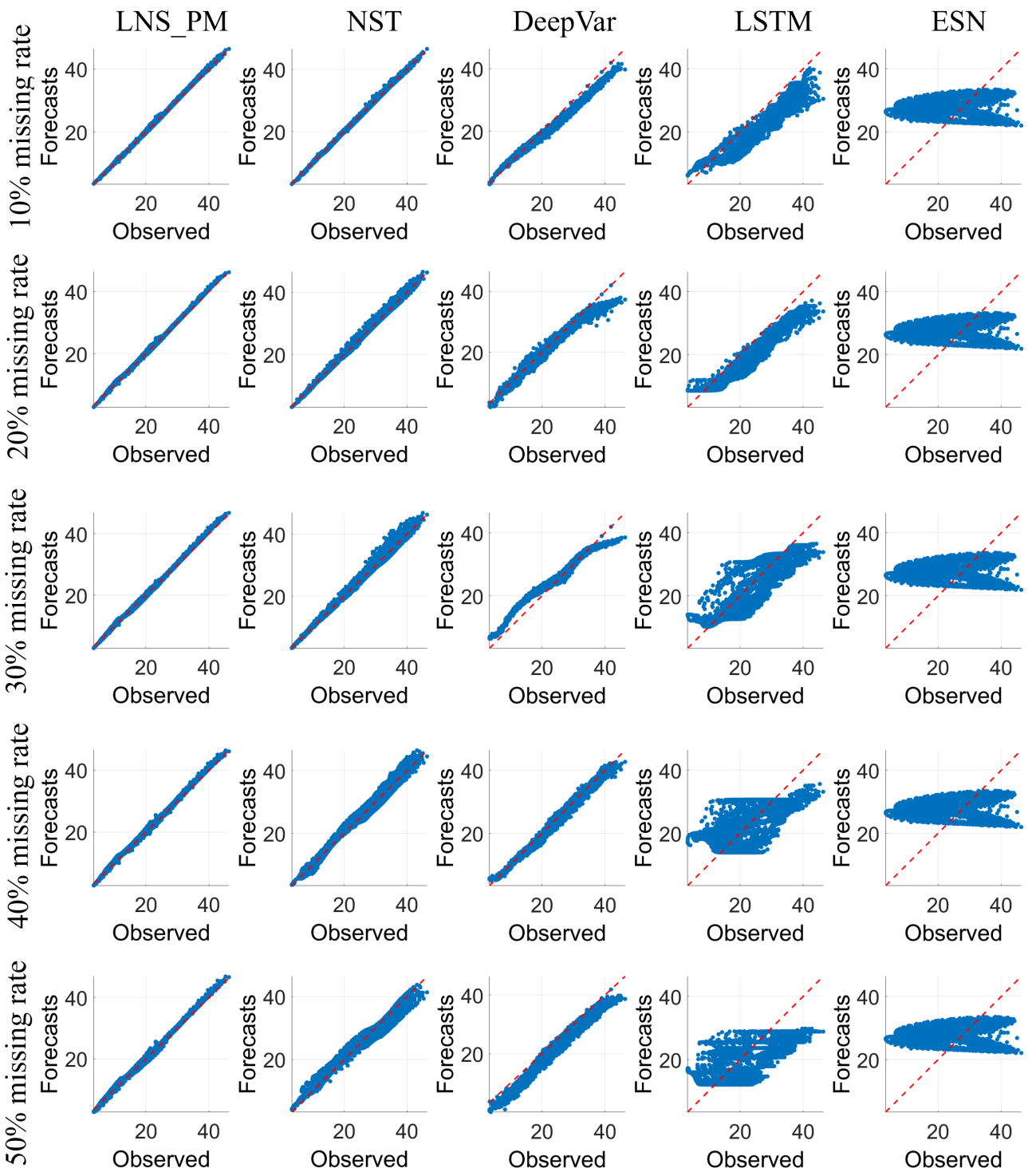


Figure 3. Scatter plots of observed versus forecasted values for the Lorenz system, with different predictors (columns 1 to 5) and increasing missing rates (rows 1 to 5, ranging from 10% to 50%). Perfect predictions align along the diagonal.

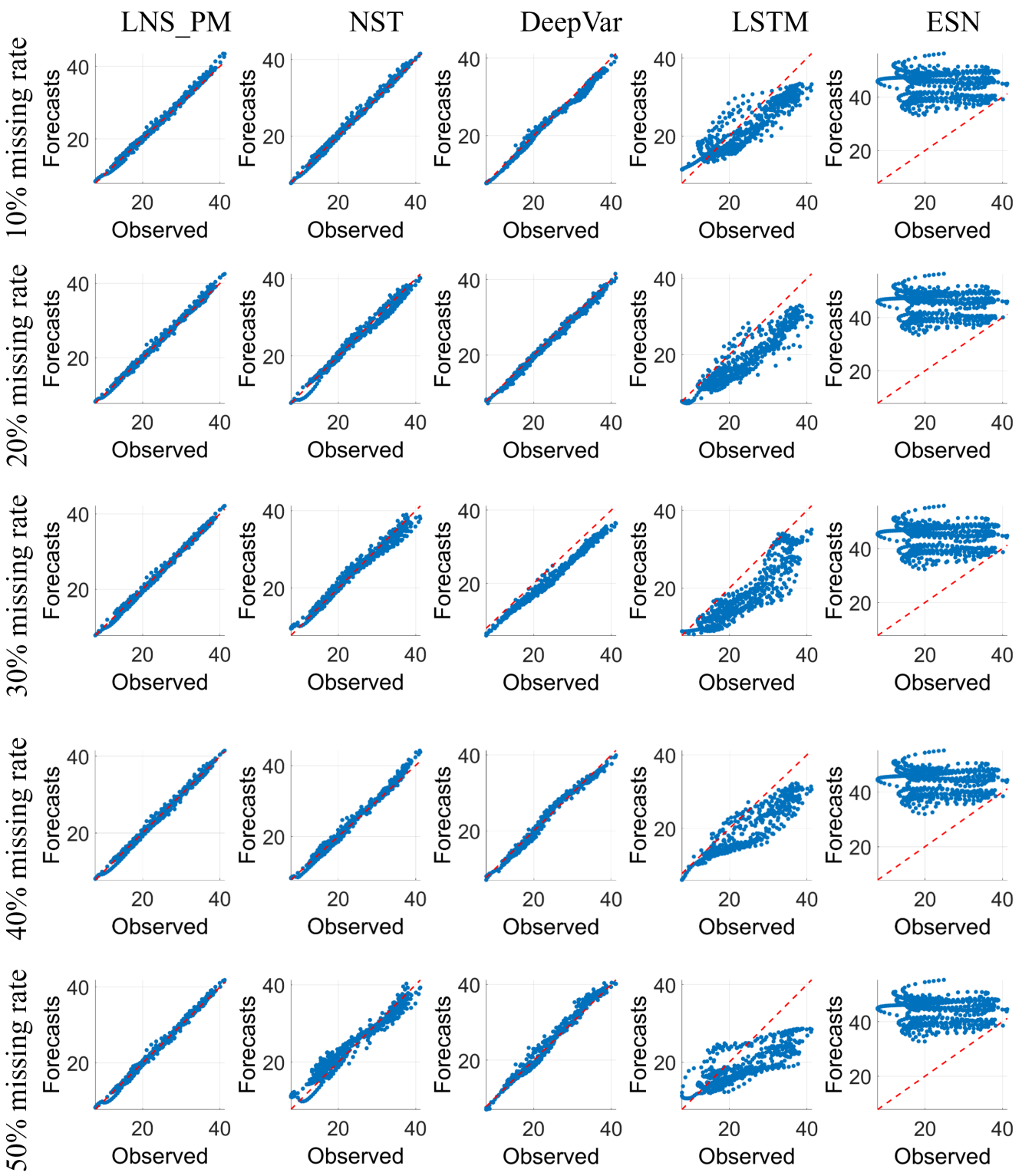


Figure 4. Scatter plots of observed versus forecasted values for the Hyperchaotic Lorenz system, with different predictors (columns 1 to 5) and increasing missing rates (rows 1 to 5, ranging from 10% to 50%). Perfect predictions align along the diagonal.

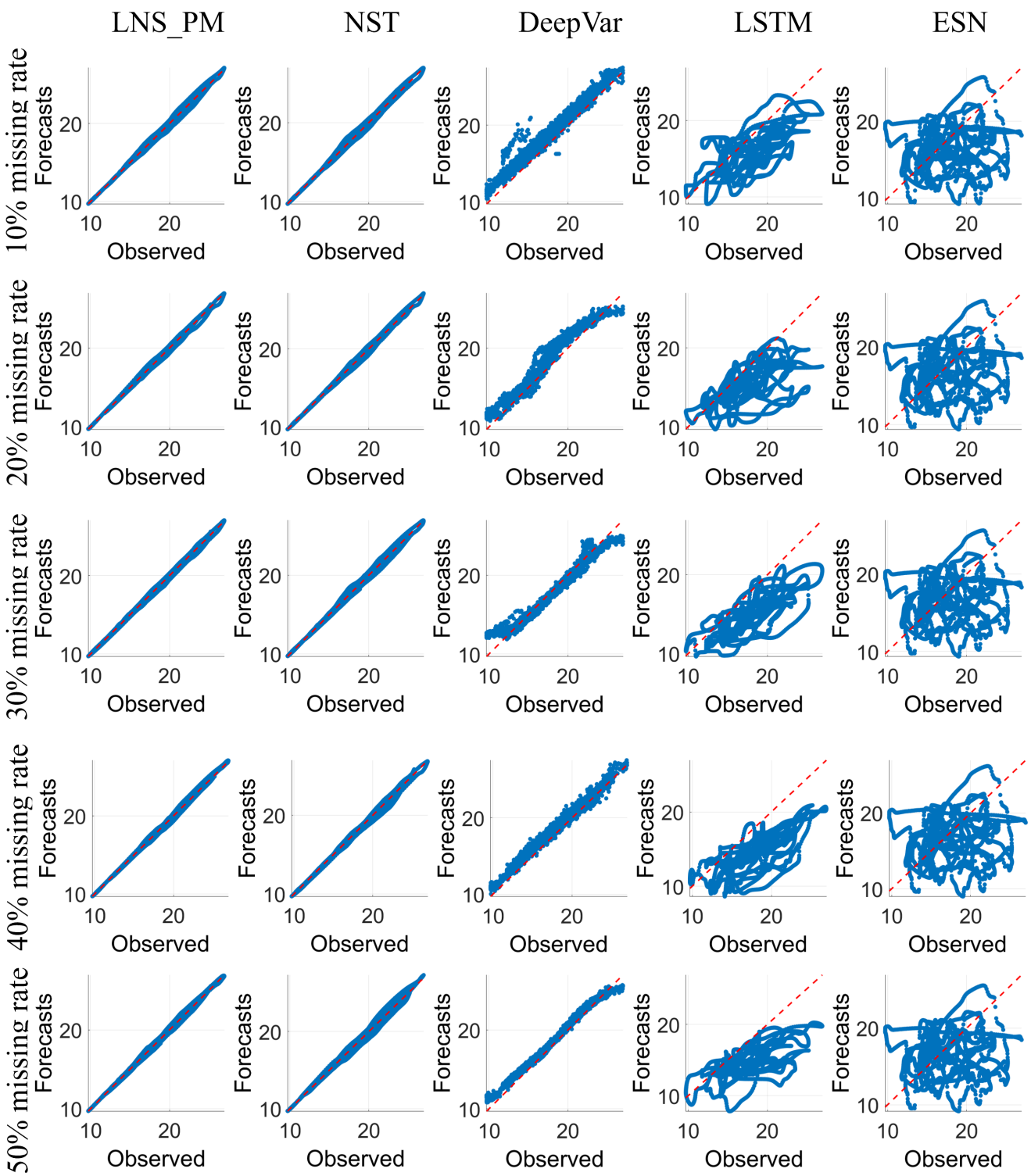


Figure 5. Scatter plots of observed versus forecasted values for the Coupled Lorenz system, with different predictors (columns 1 to 5) and increasing missing rates (rows 1 to 5, ranging from 10% to 50%). Perfect predictions align along the diagonal.

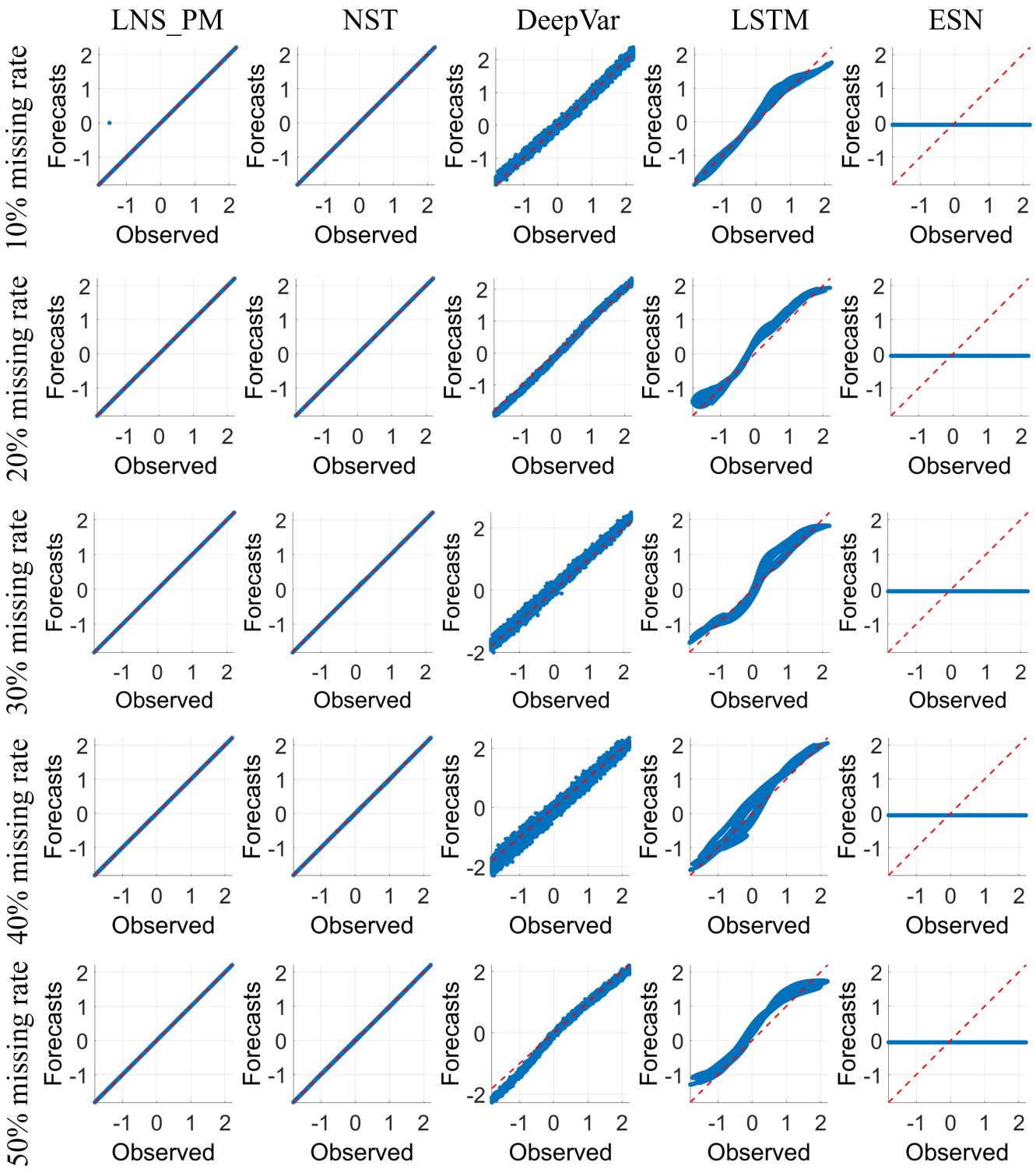


Figure 6. Scatter plots of observed versus forecasted values for the Conservative Chaotic system, with different predictors (columns 1 to 5) and increasing missing rates (rows 1 to 5, ranging from 10% to 50%). Perfect predictions align along the diagonal.

Testing the same prediction model on datasets with different missing rates reveals that LNS_PM produces consistently stable predictions across various missing rates, demonstrating its robustness across different samples. NST also shows a certain level of stability, but its prediction errors become larger at higher missing rates (30–50%). DeepVar's performance is inconsistent across the different missing rates. LSTM and ESN exhibit significant volatility

in their predictions, with large discrepancies between predicted and actual values. The scatter plot results for the Hyperchaotic Lorenz (Figure 4), Coupled Lorenz (Figure 5), and Conservative Chaotic systems (Figure 6) align with this analysis. Specifically, in Figure 6, the prediction results for the Conservative Chaotic system show that LNS_PM and NST have their predicted values well distributed near the diagonal across all missing rates, indicating good performance. On the other hand, as the missing rate increases, the predicted values of DeepVar and LSTM gradually deviate from the diagonal, suggesting that these models perform poorly at higher missing rates. ESN, on the other hand, has predicted values concentrated near the horizontal axis at all missing rates, with little variation with the observed values, indicating that this model is unsuitable for predicting chaotic time series with missing data. Overall, LNS_PM consistently delivers a high level of accuracy, with predicted values closely matching the actual ones, while the other models show larger prediction errors.

Figures 7 and 8 present the variation in the root mean square error (RMSE) and mean absolute percentage error (MAPE) for different prediction models under four chaotic systems (Lorenz, Hyperchaotic Lorenz, and Coupled Lorenz). From the figures, it can be observed that the LNS_PM model performs the best across all three Lorenz systems, particularly at higher missing rates (30% to 50%), where both RMSE and MAPE remain at relatively low values, demonstrating the model's strong robustness to missing data. In comparison, the NST model performs similarly to LNS_PM but exhibits slightly higher errors in low-dimensional systems (e.g., the Lorenz system). Moreover, as the missing rate increases, the errors for NST also increase. Traditional LSTM and ESN models are more sensitive to missing data; with increasing missing rates, their RMSE and MAPE values rise significantly, particularly in the Hyperchaotic Lorenz system, where their performance is notably poor.

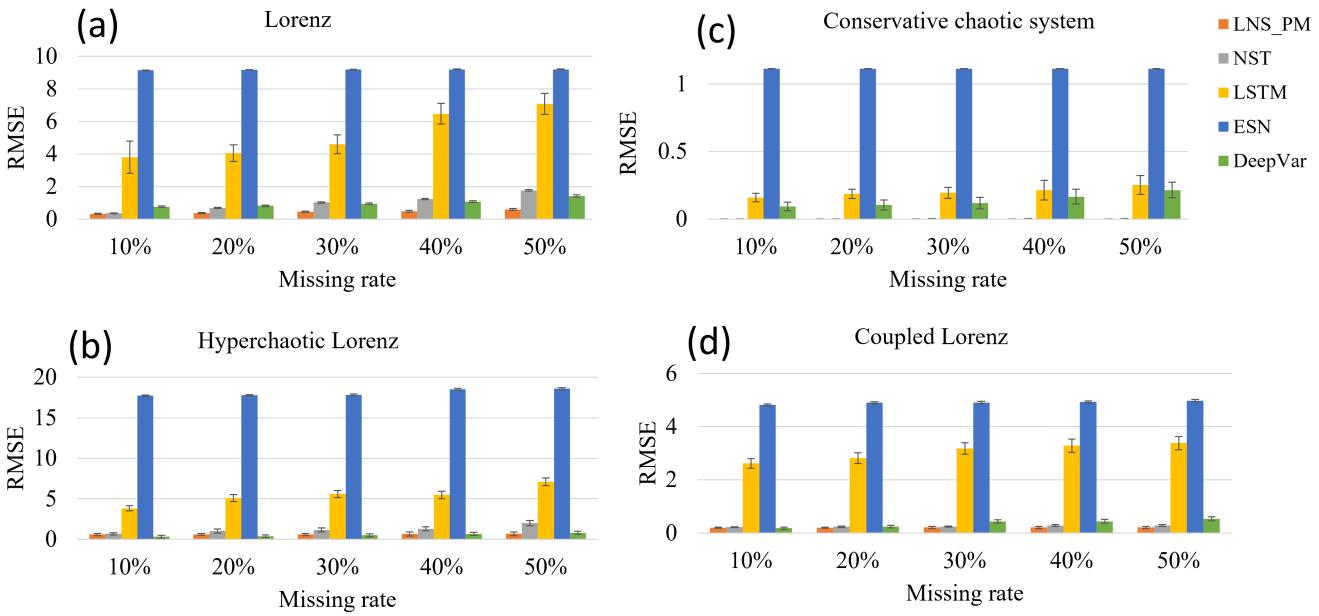


Figure 7. The bar charts display the RMSE values with error bars for four chaotic systems: Lorenz (**a**), Hyperchaotic Lorenz (**b**), Conservative Chaotic system (**c**), and Coupled Lorenz (**d**), under varying missing rates (10–50%). The results are compared across five predictors: LNS_PM (orange), NST (gray), LSTM (yellow), ESN (blue), and DeepVar (green). The error bars indicate the consistency of the RMSE results.

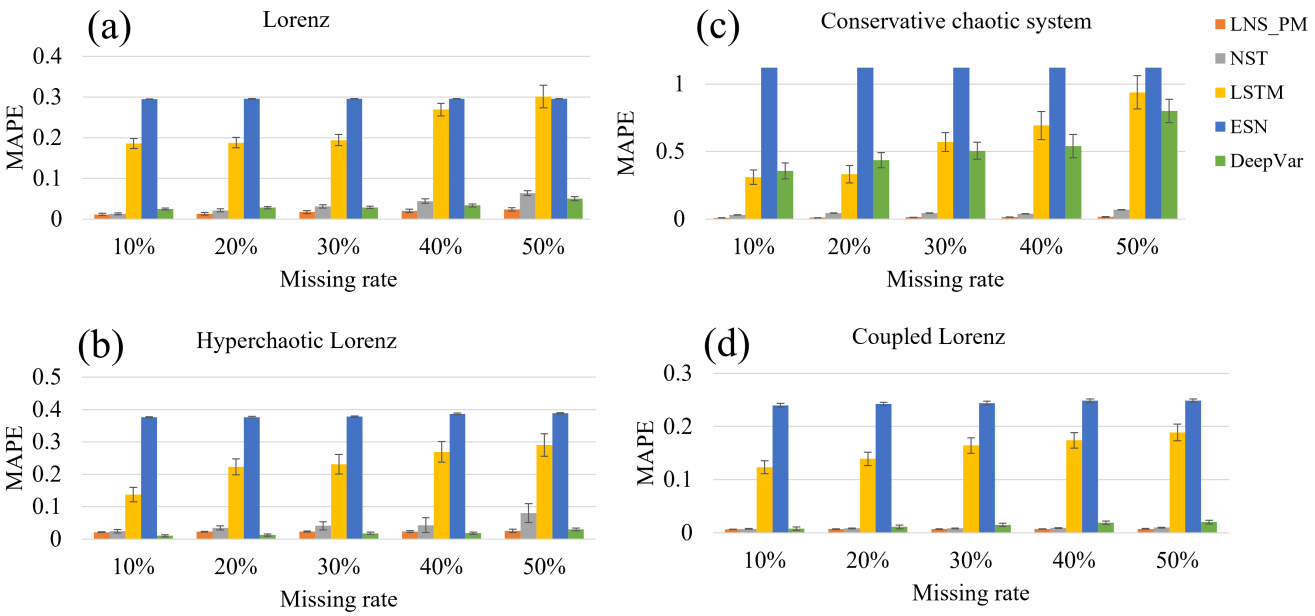


Figure 8. The bar charts display the MAPE values with error bars for four chaotic systems: Lorenz (**a**), Hyperchaotic Lorenz (**b**), Conservative Chaotic system (**c**), and Coupled Lorenz (**d**), under varying missing rates (10–50%). The results are compared across five predictors: LNS_PM (orange), NST (gray), LSTM (yellow), ESN (blue), and DeepVar (green). The error bars indicate the consistency of the MAPE results.

Furthermore, the performance of the LNS_PM model varies across systems of different dimensions. As shown in Figure 8, for the Lorenz system, where sufficient training data are available, the LNS_PM model achieves smaller prediction errors. For the Hyperchaotic Lorenz system, the increased system complexity is offset by the availability of additional feature inputs, enabling the LNS_PM model to better capture useful features and achieve superior prediction performance. For the Coupled Lorenz system, although the training data are relatively limited, the increased system dimension introduces more features, allowing the model to construct more complex mapping relationships and thereby improve prediction accuracy. The consistently low errors across all missing rates in the Coupled Lorenz system further highlight the superiority of the LNS_PM model in multi-variable systems.

In summary, the LNS_PM model demonstrates high prediction accuracy and robustness across all cases, particularly under high missing rates. This is primarily attributed to the interpolation method proposed in this study, which effectively reconstructs the missing data and enables the model to capture the temporal evolution patterns of system states. In contrast, traditional LSTM and ESN models tend to overfit incomplete data, preventing them from learning truly useful patterns. Consequently, their performance deteriorates significantly when applied to new datasets with similarly high missing rates.

Moreover, the visualization results of feature importance (Figure 9) further emphasize the critical role of the underlying dynamic features in the model's predictions. Through the feature importance analysis of the Lorenz, Hyperchaotic Lorenz, Coupled Lorenz, and Conservative Chaotic systems, we observe that "x" plays a critical role in predictions due to its coupling relationships with other variables (such as "y" and "z"), which significantly influence the prediction results. This finding aligns with previous studies, which concluded that variables with higher frequencies of appearance in one or more equations tend to perform better when used to reconstruct other variables [36]. In the Hyperchaotic Lorenz system, the introduction of the variable "w" not only expanded the system's dimensionality but also significantly altered its dynamic properties, driving the system to exhibit hyperchaotic

behavior. According to the feature importance comparison in Figure 9b, the importance of "w" follows closely after "x". This result not only highlights the pivotal role of "w" in the system but also confirms its significant contribution to the overall dynamic behavior.

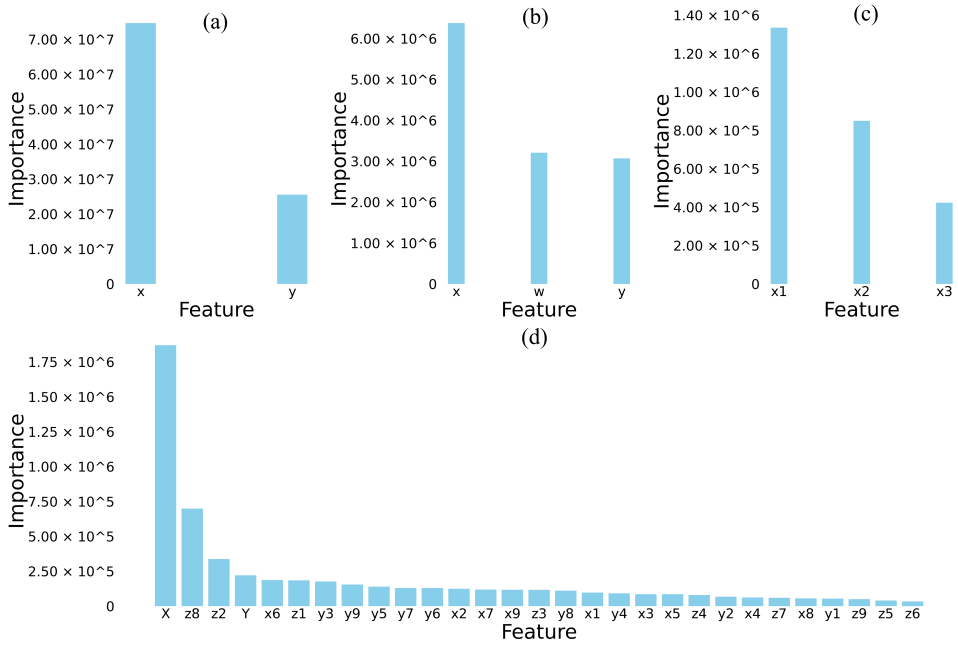


Figure 9. (**a**) Lorenz System, (**b**) Hyperchaotic Lorenz system, (**c**) Conservative Chaotic system, (**d**) Coupled Lorenz system—Visualization of the feature's impact on the model's output.

Based on the results of the statistical tests (Table 3), we observe significant differences among the models across varying dimensions. For Lorenz data, the F-statistic is 139.19, with a p-value of 2.68×10^{-14}. For Hyperchaotic Lorenz data, the F-statistic increases to 763.04, with a p-value of 1.50×10^{-21}. For Conservative Chaotic system data, the F-statistic rises further to 1447.28, accompanied by a p-value of 2.57×10^{-24}. Finally, for Coupled Lorenz data, the F-statistic is 857.41, with a p-value of 4.70×10^{-22}.

At a significance level of $\alpha = 0.05$, all p-values are well below the threshold, confirming that the performance differences among models are statistically significant across all dimensions. The extremely low p-values strongly suggest that as data dimensionality increases, the performance disparities among models become more pronounced. This result underscores the impact of data dimensionality on model behavior and highlights the importance of considering dimensional effects when evaluating model performance. Similarly, based on the CI in Table 4, we can observe that the confidence intervals of LNS_PM do not overlap with those of ESN, DeepVar, and LSTM, indicating that the differences in predictive performance among these models are statistically significant. Combined with the RMSE and MAPE results, LNS_PM not only maintains relatively low errors under different missing rates but also demonstrates a relatively stable confidence interval width.

We experimentally validated the proposed framework on datasets ranging from 3 to 30 dimensions. In higher-dimensional datasets, the increased diversity and redundancy of features provide additional information for feature inference, allowing the model to achieve greater accuracy during the completion phase. The experimental results show that the prediction accuracy of the proposed framework is comparable to classical models (such as NST and DeepVar) when the missing data ratio is low. However, as the missing data ratio increases to 50%, the proposed framework significantly outperforms these classical models. This advantage stems from the LightGBM module's strong ability to represent the complex distribution of high-dimensional data and its robustness in sparse data scenarios.

Table 3. Statistical results for different chaotic systems.

Data	F-Statistic	p-Value	Significance Level ($\alpha = 0.05$)
Lorenz	139.19	2.68×10^{-14}	Significant
Hyperchaotic Lorenz	763.04	1.50×10^{-21}	Significant
Coupled Lorenz	857.41	4.70×10^{-22}	Significant
Conservative Chaos	1447.28	2.57×10^{-24}	Significant

Table 4. Comparison of predictors with 95% confidence intervals for different missing rates (10%, 20%, 30%, 40%, 50%).

Data	LNS_PM	NST	LSTM	DeepVar	ESN
Lorenz	(23.583, 23.782)	(23.424, 23.621)	(20.242, 20.420)	(21.738, 21.906)	(26.553, 26.610)
	(23.668, 23.867)	(23.490, 23.689)	(20.256, 20.435)	(22.346, 22.523)	(26.469, 26.525)
	(23.640, 23.841)	(23.648, 23.843)	(21.375, 21.544)	(24.902, 25.056)	(26.601, 26.659)
	(23.746, 23.943)	(23.723, 23.914)	(21.962, 22.100)	(22.753, 22.948)	(26.586, 26.643)
	(23.669, 23.870)	(23.525, 23.696)	(20.679, 20.804)	(20.589, 20.791)	(26.624, 26.681)
Hyperchaotic Lorenz	(24.507, 25.053)	(24.479, 25.017)	(22.436, 22.861)	(23.522, 24.043)	(39.756, 40.127)
	(24.355, 24.899)	(23.973, 24.494)	(20.174, 20.625)	(23.515, 24.049)	(39.877, 40.253)
	(24.070, 24.617)	(24.026, 24.532)	(19.714, 20.209)	(21.302, 21.796)	(38.838, 39.212)
	(24.039, 24.579)	(24.595, 25.167)	(20.230, 20.673)	(23.899, 24.442)	(38.881, 39.253)
	(24.178, 24.725)	(24.614, 25.086)	(20.003, 20.309)	(24.591, 25.161)	(38.994, 39.356)
Coupled Lorenz	(17.261, 17.360)	(17.249, 17.348)	(16.576, 16.654)	(18.440, 18.535)	(16.915, 17.001)
	(17.298, 17.396)	(17.311, 17.410)	(16.391, 16.466)	(18.182, 18.279)	(17.094, 17.179)
	(17.271, 17.370)	(17.271, 17.369)	(15.582, 15.653)	(17.081, 17.171)	(17.132, 17.216)
	(17.277, 17.376)	(17.229, 17.327)	(15.336, 15.413)	(17.987, 18.080)	(17.044, 17.133)
	(17.269, 17.368)	(17.279, 17.377)	(15.876, 15.941)	(17.884, 17.974)	(17.042, 17.125)
Conservative Chaos	(−0.007, 0.011)	(−0.006, 0.012)	(0.048, 0.066)	(0.008, 0.025)	(−0.646, −0.645)
	(−0.006, 0.012)	(−0.005, 0.012)	(0.119, 0.138)	(−0.066, −0.047)	(−0.644, −0.643)
	(−0.006, 0.011)	(−0.007, 0.011)	(0.070, 0.088)	(0.069, 0.087)	(−0.646, −0.646)
	(−0.006, 0.012)	(−0.007, 0.011)	(0.048, 0.067)	(−0.101, −0.083)	(−0.645, −0.644)
	(−0.006, 0.011)	(−0.007, 0.011)	(0.156, 0.174)	(−0.162, −0.143)	(−0.641, −0.639)

Based on these results and the theoretical analysis, the proposed framework demonstrates excellent scalability, making it well suited for real-world complex systems with high-dimensional datasets and large missing data ratios. Furthermore, its modular design ensures high flexibility and adaptability when handling datasets of varying scales and structures.

Regarding computational cost, using a dataset with 100 feature dimensions and 100K samples as an example, the training time of the LightGBM module within the proposed framework is only a few minutes, with inference time in the millisecond range. In contrast, the NST model requires several hours of training and seconds for inference. Therefore, the LightGBM module has a low impact on the overall computational cost, offering high computational efficiency. Additionally, with the high-performance computing capabilities of modern GPUs, the proposed framework is less affected by computational resource limitations in practical applications, demonstrating strong applicability and scalability.

To further validate the superiority of the proposed data imputation method within the framework designed in this study, we conducted additional comparative experiments with two classical methods (KNN and GAN [44]). The prediction performance of LNS_PM, KNN-NST, and GAN-NST was analyzed, and the results were presented using prediction curves, scatter plots, and error plots. As shown in Figure 10, both the LNS_PM and

KNN-NST methods effectively capture the trend of the curve, with their prediction results distributed along the diagonal in the scatter plots (Figures 11–14), indicating that both methods achieve satisfactory data imputation results. However, analysis of the error plots (Figures 15 and 16) reveals that LNS_PM achieves lower prediction errors across all datasets, demonstrating higher prediction accuracy that better meets the requirements of chaotic system forecasting. This further highlights the advantages of the proposed method in handling missing data.

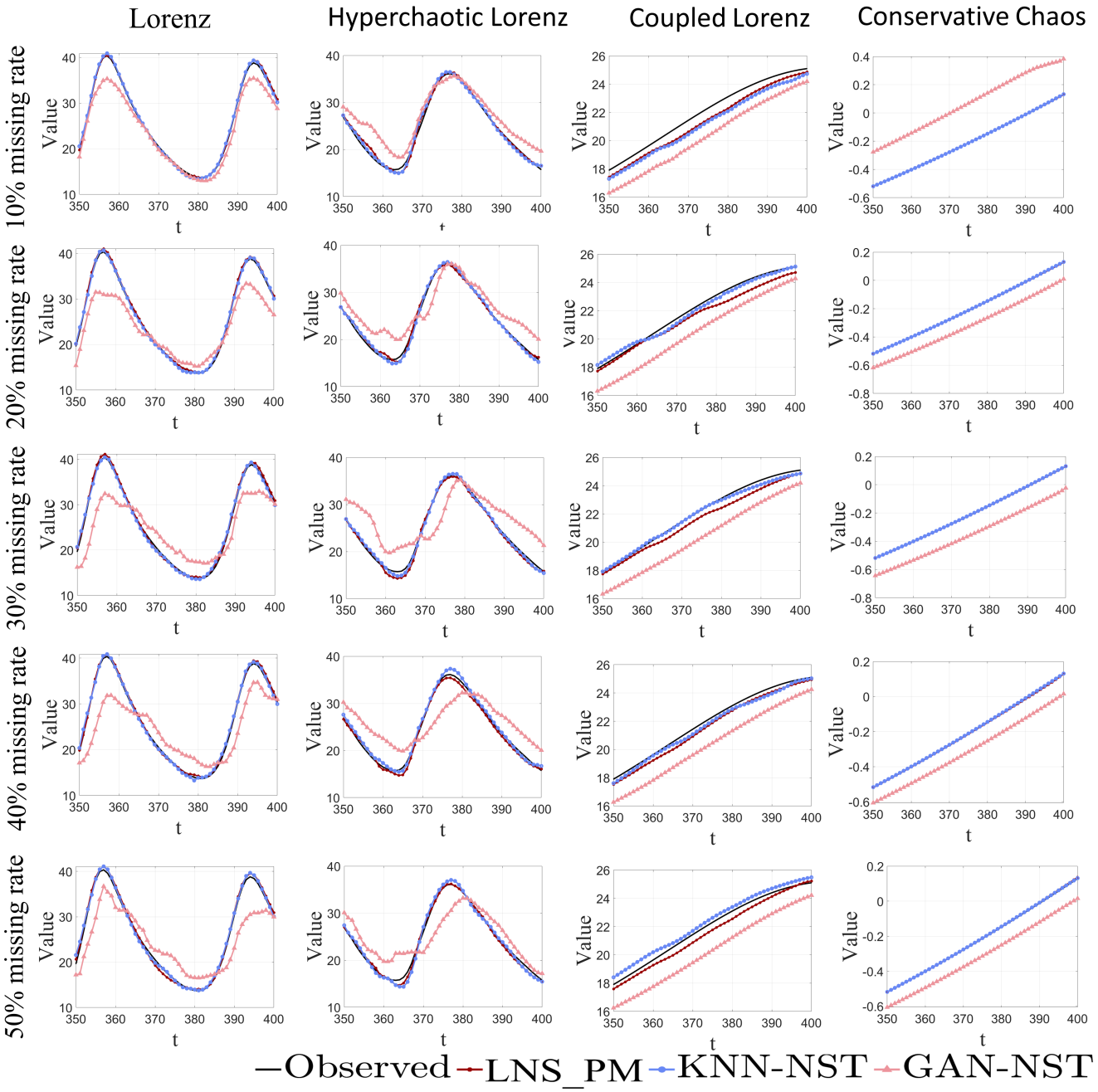


Figure 10. Prediction results of three forecasting models (colored lines) for the Lorenz system, Hyperchaotic Lorenz system, Coupled Lorenz system, and Conservative Chaotic system (columns 1 to 4), with missing rates ranging from 10% to 50% (raws 1 to 5).

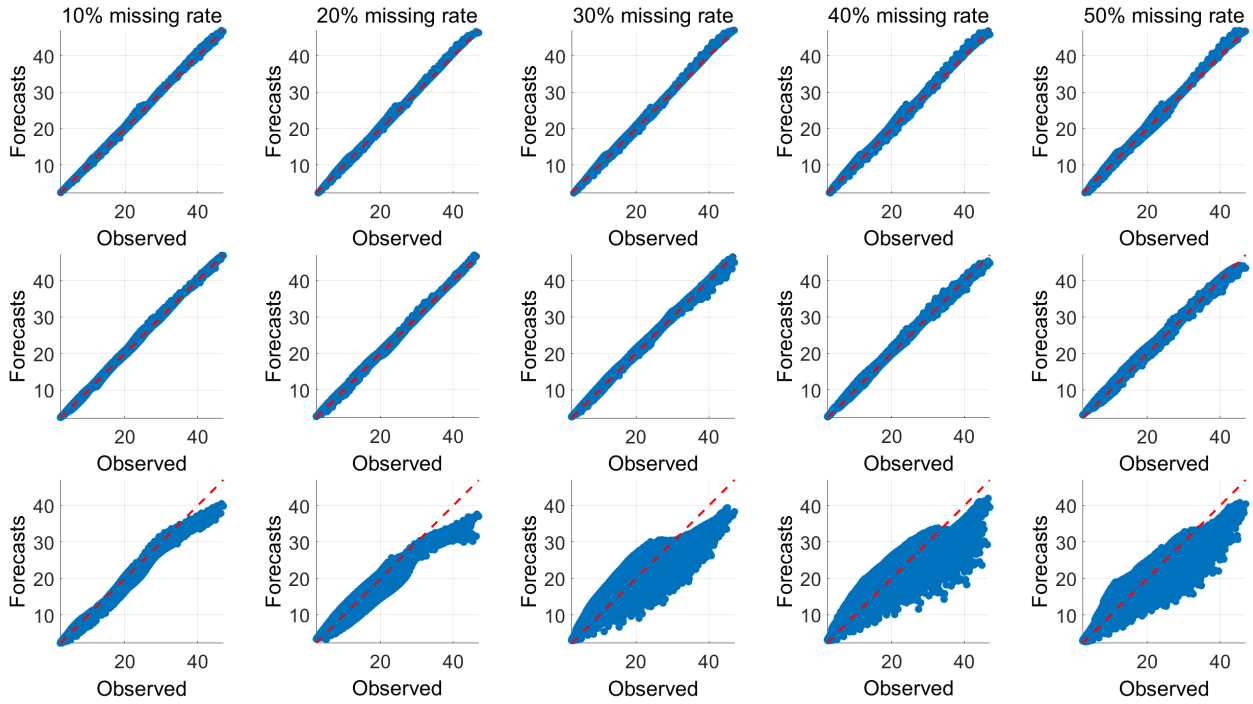


Figure 11. Scatter plots of observed versus forecasted values for the Lorenz system, with different predictors (LNS_PM, KNN-NST, GAN-NST, rows 1 to 3) and increasing missing rates (columns 1 to 5, ranging from 10% to 50%). Perfect predictions align along the diagonal.

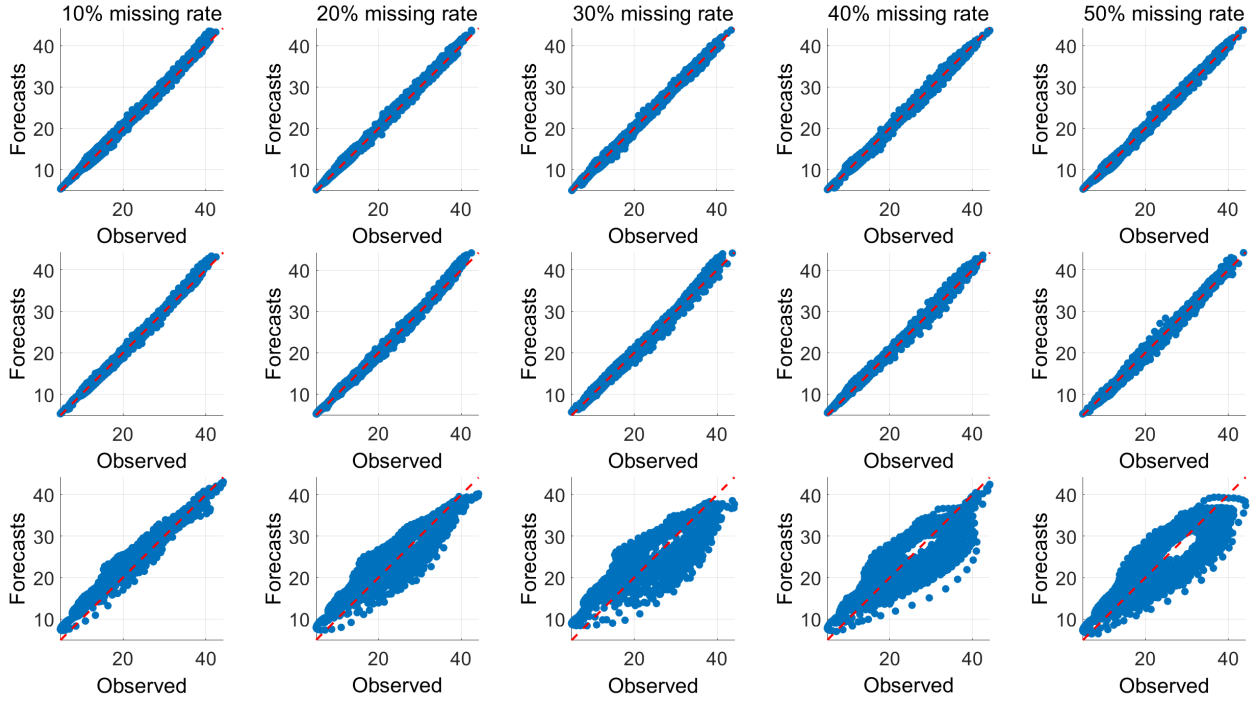


Figure 12. Scatter plots of observed versus forecasted values for the Hyperchaotic Lorenz system, with different predictors (LNS_PM, KNN-NST, GAN-NST, rows 1 to 3) and increasing missing rates (columns 1 to 5, ranging from 10% to 50%). Perfect predictions align along the diagonal.

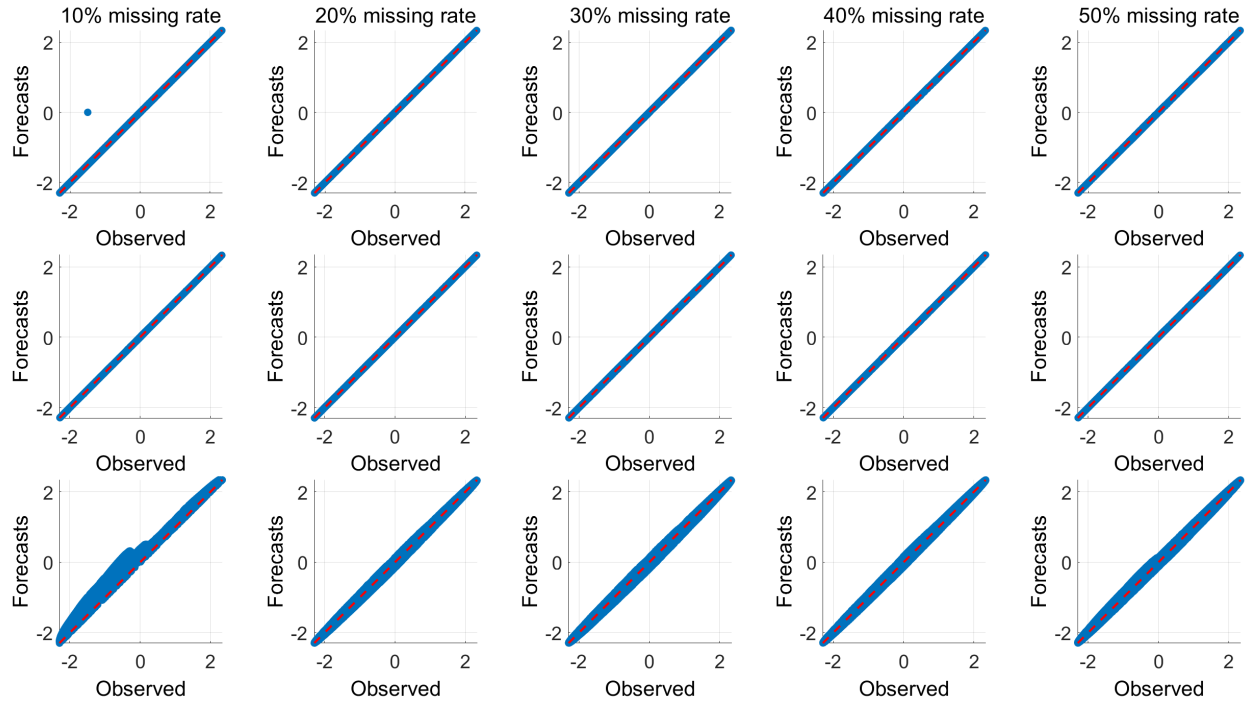


Figure 13. Scatter plots of observed versus forecasted values for the Conservative Chaotic system, with different predictors (LNS_PM, KNN-NST, GAN-NST, rows 1 to 3) and increasing missing rates (columns 1 to 5, ranging from 10% to 50%).

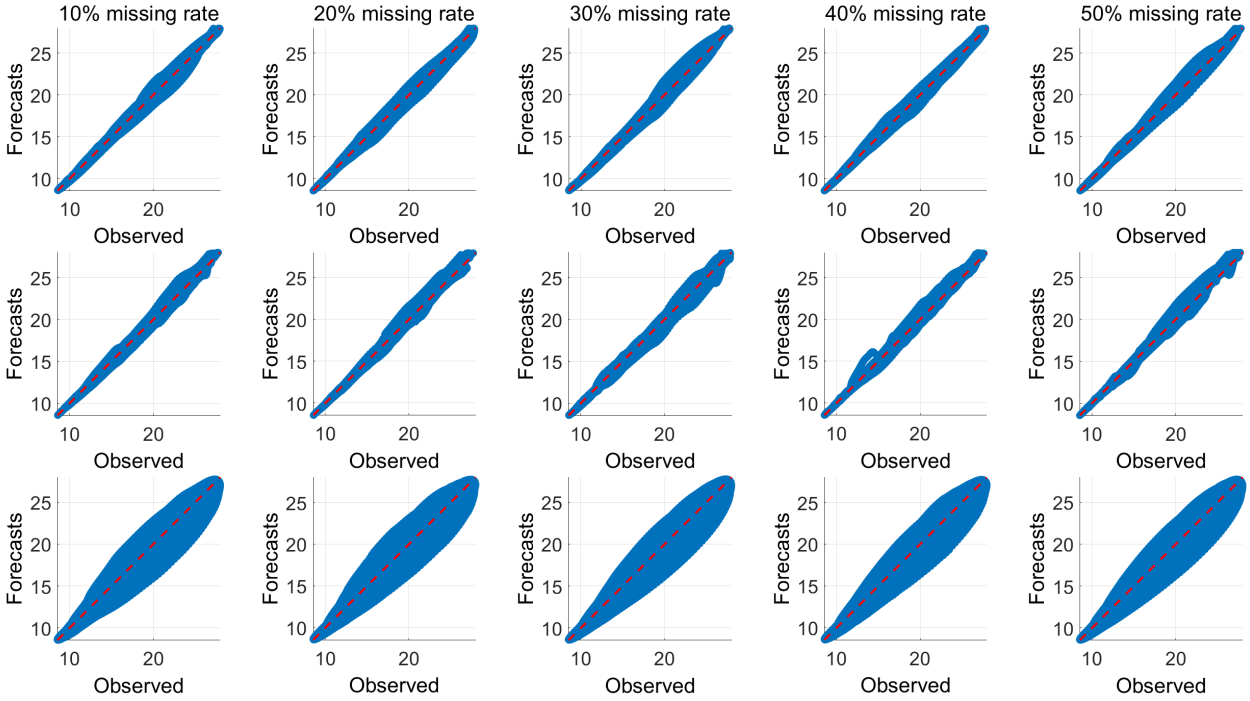


Figure 14. Scatter plots of observed versus forecasted values for the Coupled Lorenz system, with different predictors (LNS_PM, KNN-NST, GAN-NST, rows 1 to 3) and increasing missing rates (columns 1 to 5, ranging from 10% to 50%).

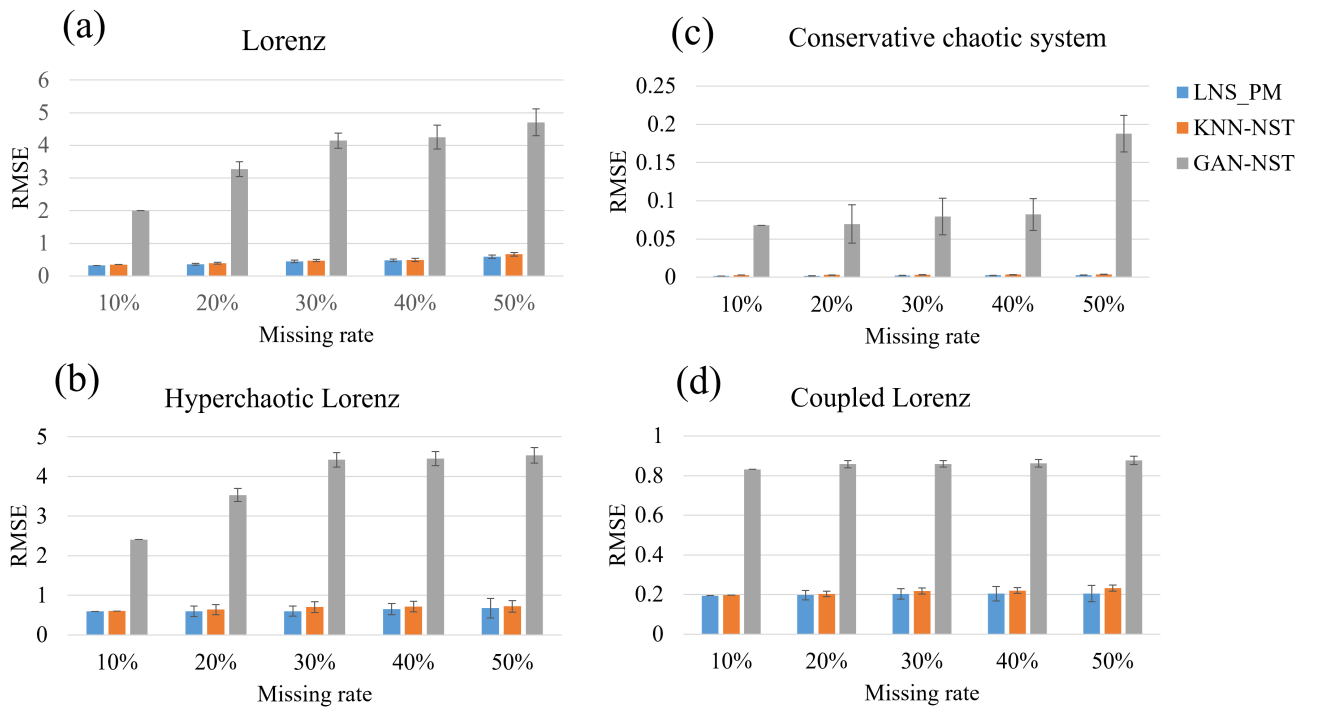


Figure 15. The bar charts display the RMSE values with error bars for four chaotic systems: Lorenz (**a**), Hyperchaotic Lorenz (**b**), Conservative Chaotic system (**c**), and Coupled Lorenz (**d**), under varying missing rates (10–50%). The results are compared across three predictors: LNS_PM (Blue), KNN-NST (Orange), GAN-NST (Gray). The error bars indicate the consistency of the RMSE results.

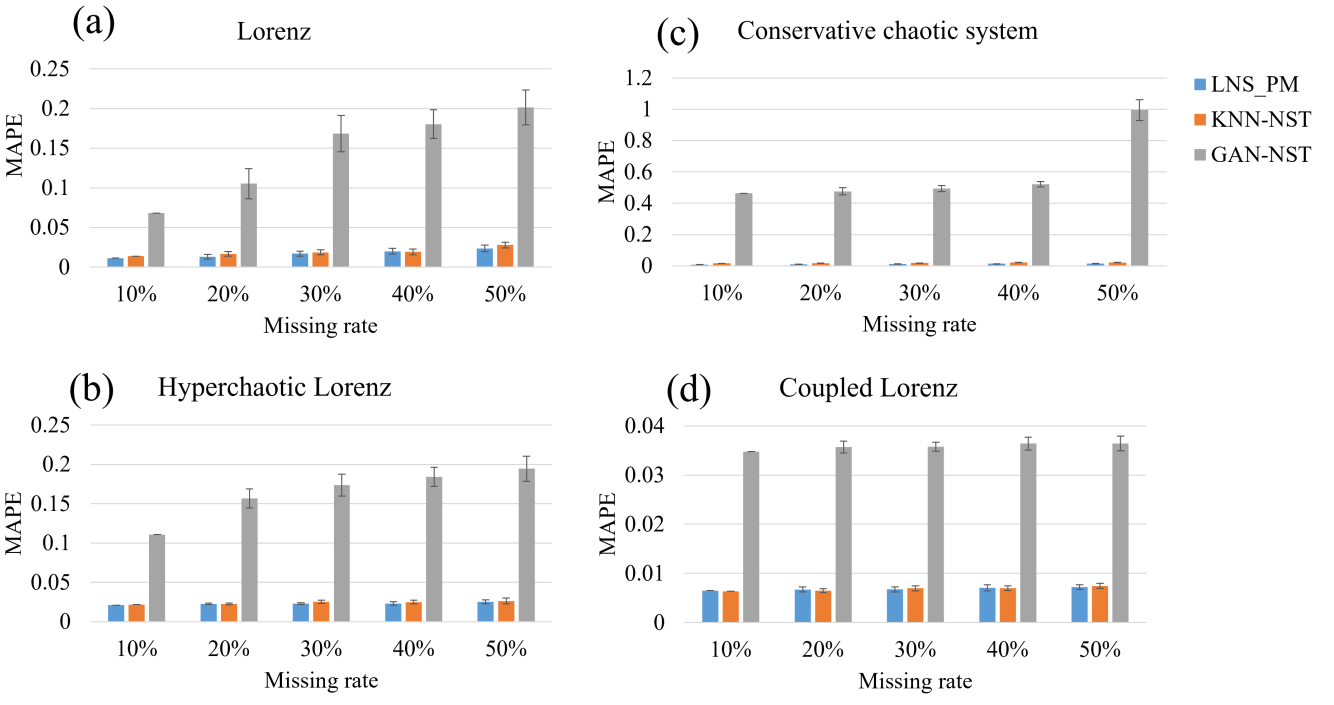


Figure 16. The bar charts display the MAPE values with error bars for four chaotic systems: Lorenz (**a**), Hyperchaotic Lorenz (**b**), Conservative Chaotic system (**c**), and Coupled Lorenz (**d**), under varying missing rates (10–50%). The results are compared across three predictors: LNS_PM (Blue), KNN-NST (Orange), GAN-NST (Gray). The error bars indicate the consistency of the MAPE results.

4. Conclusions

We presented a general model, LNS_PM, designed to address the challenge of reconstructing and predicting chaotic time series with missing data. This model integrates LightGBM, a gradient-boosting machine learning algorithm, into NST. We compared the prediction accuracy and robustness of five different models (LNS_PM, NST, LSTM, ESN, and DeepVar) on complex, chaotic system time series with missing data, specifically assessing their performance under missing rates of 10% to 50%. Additionally, we examined the impact of system dimensionality on prediction accuracy. The experimental results show that the proposed model demonstrates a significant advantage with larger missing rates, offering high robustness in predicting chaotic time series with missing data. Compared to NST without LightGBM, LNS_PM substantially reduces the prediction error across chaotic systems of varying complexity. Furthermore, the feature importance analysis aligns with the underlying dynamic relationships within the chaotic system equations, further validating the effectiveness and reliability of our approach.

Future research can explore several directions. First, integrating other advanced machine learning algorithms into LNS_PM could enhance its predictive performance. Second, applying the model to real-world chaotic datasets, such as price fluctuations in financial markets, stock market prediction, weather forecasting, and wind power generation, which often involve more 422 complex noise and nonlinear characteristics, to further validate its applicability. Future studies should also focus on developing specialized optimization techniques to address these complexities while evaluating the model's scalability and robustness.

Author Contributions: Conceptualization, H.M.; methodology, Y.W.; software, J.L.; validation, J.L.; formal analysis, J.L.; investigation, J.L.; resources, J.L.; data curation, J.L.; writing—original draft preparation, J.L.; writing—review and editing, Z.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data will be made available by the authors upon request.

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

References

1. Szpiro, G.G. Forecasting chaotic time series with genetic algorithms. *Phys. Rev. E* **1997**, *55*, 2557. [CrossRef]
2. Palit, A.K.; Popovic, D. Forecasting chaotic time series using neuro-fuzzy approach. In Proceedings of the IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339), Washington, DC, USA, 10–16 July 1999; Volume 3, pp. 1538–1543.
3. Wang, Y.; Jiang, W.; Yuan, S.; Wang, J. Forecasting chaotic time series based on improved genetic Wnn. In Proceedings of the 2008 Fourth International Conference on Natural Computation, Jinan, China, 18–20 October 2008; Volume 5, pp. 519–523.
4. Jiang, C.; Song, F. Forecasting chaotic time series of exchange rate based on nonlinear autoregressive model. In Proceedings of the 2010 2nd International Conference on Advanced Computer Control, Shenyang, China, 27–29 March 2010; Volume 5, pp. 238–241.
5. Gómez-Gil, P.; Ramírez-Cortes, J.M.; Pomares Hernández, S.E.; Alarcón-Aquino, V. A neural network scheme for long-term forecasting of chaotic time series. *Neural Process. Lett.* **2011**, *33*, 215–233. [CrossRef]
6. Shen, M.; Chen, W.N.; Zhang, J.; Chung, H.S.H.; Kaynak, O. Optimal selection of parameters for nonuniform embedding of chaotic time series using ant colony optimization. *IEEE Trans. Cybern.* **2013**, *43*, 790–802. [CrossRef]
7. Fonseca, R.; Gómez-Gil, P. Temporal validated meta-learning for long-term forecasting of chaotic time series using monte carlo cross-validation. In *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 353–367.
8. Abdulkadir, S.J.; Yong, S.P. Scaled UKF–NARX hybrid model for multi-step-ahead forecasting of chaotic time series data. *Soft Comput.* **2015**, *19*, 3479–3496. [CrossRef]
9. Sangiorgio, M.; Dercole, F. Robustness of LSTM neural networks for multi-step forecasting of chaotic time series. *Chaos Solitons Fractals* **2020**, *139*, 110045. [CrossRef]

10. Kapil, C.; Barde, V.; Seemala, G.K.; Dimri, A. Investigating forced transient chaos in monsoon using Echo State Networks. *Clim. Dyn.* **2024**, *62*, 5759–5768. [CrossRef]
11. Sun, J.; Li, L.; Peng, H. Sequence Prediction and Classification of Echo State Networks. *Mathematics* **2023**, *11*, 4640. [CrossRef]
12. Wang, Q.; Jiang, L.; Yan, L.; He, X.; Feng, J.; Pan, W.; Luo, B. Chaotic time series prediction based on physics-informed neural operator. *Chaos Solitons Fractals* **2024**, *186*, 115326. [CrossRef]
13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
14. Cheng, C.; Tan, F.; Wei, Z. DeepVar: An end-to-end deep learning approach for genomic variant recognition in biomedical literature. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 598–605.
15. Liu, Y.; Wu, H.; Wang, J.; Long, M. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 9881–9893.
16. Liu, T.; Zhao, X.; Sun, P.; Zhou, J. A hybrid proper orthogonal decomposition and next generation reservoir computing approach for high-dimensional chaotic prediction: Application to flow-induced vibration of tube bundles. *Chaos Interdiscip. J. Nonlinear Sci.* **2024**, *34*, 3. [CrossRef]
17. Xiong, L.; Su, L.; Wang, X.; Pan, C. Dynamic adaptive graph convolutional transformer with broad learning system for multi-dimensional chaotic time series prediction. *Appl. Soft Comput.* **2024**, *157*, 111516. [CrossRef]
18. Fu, K.; Li, H.; Shi, X. CTF-former: A novel simplified multi-task learning strategy for simultaneous multivariate chaotic time series prediction. *Neural Netw.* **2024**, *174*, 106234. [CrossRef]
19. Wahdany, D.; Schmitt, C.; Cremer, J.L. More than accuracy: End-to-end wind power forecasting that optimises the energy system. *Electr. Power Syst. Res.* **2023**, *221*, 109384. [CrossRef]
20. Batista, G.E.; Monard, M.C. An analysis of four missing data treatment methods for supervised learning. *Appl. Artif. Intell.* **2003**, *17*, 519–533. [CrossRef]
21. Banks, D.; House, L.; McMorris, F.R.; Arabie, P.; Gaul, W.A. Classification, Clustering, and Data Mining Applications. In Proceedings of the Meeting of the International Federation of Classification Societies (IFCS), Illinois Institute of Technology, Chicago, IL, USA, 15–18 July 2004; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2011.
22. Luengo, J.; García, S.; Herrera, F. A study on the use of imputation methods for experimentation with radial basis function network classifiers handling missing attribute values: The good synergy between rbfns and eventcovering method. *Neural Netw.* **2010**, *23*, 406–418. [CrossRef]
23. Jerez, J.M.; Molina, I.; García-Laencina, P.J.; Alba, E.; Ribelles, N.; Martín, M.; Franco, L. Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artif. Intell. Med.* **2010**, *50*, 105–115. [CrossRef]
24. Laña, I.; Olabarrieta, I.I.; Vélez, M.; Del Ser, J. On the imputation of missing data for road traffic forecasting: New insights and novel techniques. *Transp. Res. Part C Emerg. Technol.* **2018**, *90*, 18–33. [CrossRef]
25. Tresp, V.; Hofmann, R. Nonlinear time-series prediction with missing and noisy data. *Neural Comput.* **1998**, *10*, 731–747. [CrossRef] [PubMed]
26. Chiewchanwattana, S.; Lursinsap, C. FI-GEM networks for incomplete time-series prediction. In Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290), Honolulu, HI, USA, 12–17 May 2002; Volume 2, pp. 1757–1762.
27. Zhao, P.; Xing, L.; Yu, J. Chaotic time series prediction: From one to another. *Phys. Lett. A* **2009**, *373*, 2174–2177. [CrossRef]
28. Rodriguez, H.; Flores, J.J.; Morales, L.A.; Lara, C.; Guerra, A.; Manjarrez, G. Forecasting from incomplete and chaotic wind speed data. *Soft Comput.* **2019**, *23*, 10119–10127. [CrossRef]
29. Salgado, C.M.; Azevedo, C.; Proença, H.; Vieira, S.M. Missing Data. In *Secondary Analysis of Electronic Health Records*; MIT Critical Data, Ed.; Springer: Berlin/Heidelberg, Germany, 2016; pp. 143–162.
30. Hastie, T.; Mazumder, R.; Lee, J.D.; Zadeh, R. Matrix completion and low-rank SVD via fast alternating least squares. *J. Mach. Learn. Res.* **2015**, *16*, 3367–3402.
31. Mazumder, R.; Hastie, T.; Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices. *J. Mach. Learn. Res.* **2010**, *11*, 2287–2322. [PubMed]
32. White, I.R.; Royston, P.; Wood, A.M. Multiple imputation using chained equations: Issues and guidance for practice. *Stat. Med.* **2011**, *30*, 377–399. [CrossRef]
33. Brophy, E.; Wang, Z.; She, Q.; Ward, T. Generative adversarial networks in time series: A systematic literature review. *ACM Comput. Surv.* **2023**, *55*, 1–31. [CrossRef]
34. Du, W.; Côté, D.; Liu, Y. Saits: Self-attention-based imputation for time series. *Expert Syst. Appl.* **2023**, *219*, 119619. [CrossRef]
35. Shi, L.; Yan, Y.; Wang, H.; Wang, S.; Qu, S.X. Predicting chaotic dynamics from incomplete input via reservoir computing with (D+ 1)-dimension input and output. *Phys. Rev. E* **2023**, *107*, 054209. [CrossRef]
36. Chen, Y.; Qian, Y.; Cui, X. Time series reconstructing using calibrated reservoir computing. *Sci. Rep.* **2022**, *12*, 16318. [CrossRef]

37. Cunillera, A.; Soriano, M.C.; Fischer, I. Cross-predicting the dynamics of an optically injected single-mode semiconductor laser using reservoir computing. *Chaos Interdiscip. J. Nonlinear Sci.* **2019**, *29*. [CrossRef] [PubMed]
38. Lu, Z.; Pathak, J.; Hunt, B.; Girvan, M.; Brockett, R.; Ott, E. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos Interdiscip. J. Nonlinear Sci.* **2017**, *27*. [CrossRef] [PubMed]
39. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. Lightgbm: A highly efficient gradient boosting decision tree. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
40. Lorenz, E.N. Deterministic nonperiodic flow. *J. Atmos. Sci.* **1963**, *20*, 130–141. [CrossRef]
41. Wang, X.; Wang, M. A hyperchaos generated from Lorenz system. *Phys. A Stat. Mech. Its Appl.* **2008**, *387*, 3751–3758. [CrossRef]
42. Banerjee, A.; Pathak, J.; Roy, R.; Restrepo, J.G.; Ott, E. Using machine learning to assess short term causal dependence and infer network links. *Chaos Interdiscip. J. Nonlinear Sci.* **2019**, *29*. [CrossRef] [PubMed]
43. Dong, E.; Yuan, M.; Du, S.; Chen, Z. A new class of Hamiltonian conservative chaotic systems with multistability and design of pseudo-random number generator. *Appl. Math. Model.* **2019**, *73*, 40–71. [CrossRef]
44. Yoon, J.; Jordon, J.; Schaar, M. Gain: Missing data imputation using generative adversarial nets. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 5689–5698.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.