Synthetic Data Generation for Time Series Imputation: Comparing the Foundation Model Chronos with Established Methods

Sebastião Santos Lessa
Centre for Power and Energy Systems
Inesc Tec
Porto, Portugal
sebastiao.lessa@inesctec.pt

Alexandre Lucas
Centre for Power and Energy Systems
Inesc Tec
Porto, Portugal
alexandre.lucas@inesctec.pt

***Abstract*—Accurately imputing missing data is critical in time series analysis. The present work compares Foundation Model Chronos against Linear Interpolation, K-Nearest Neighbor Imputer, and Gaussian Mixture Model Imputer with three types of missing data patterns: random, short sequential chunks, and a long sequential chunk. These results confirm that for random missing values, KNN and interpolation yield the highest performance, while Chronos outperforms these on sequences. Indeed, however, for longer sequences of missing values, Chronos starts suffering from cascading errors which eventually allow the simpler imputation methods to outrank it. Another test with limited quantities of training data showed different trade-offs for the different methods. Unlike KNN and interpolation, which smooth out the gaps, Chronos generates variable synthetic data. This can be beneficial in tasks which require control or simulation. The results highlight the strengths and weaknesses of the imputers and, therefore, offer practical insights into trade-offs between computational complexities, accuracy, and suitability for time series imputation scenarios.**

***Index Terms*—Time Series, Missing Data, Imputation, Foundation Model Chronos, Forecasting.**

I. State of the Art

This section reviews the methods and technologies related to the study, with an emphasis on their application in time series analysis and data imputation.

Time Series: A time series is a series of data points in time, usually recorded at regular intervals [1] . Unlike the typical dataset, there is a sequential nature with dependencies between past and future values that makes time series data special. Examples of such data include stock prices, temperature, and retail sales. This kind of data requires analysis techniques that take into account these patterns, especially when there are gaps or losses in it.

Imputation: Imputation is the process of replacing missing data with estimated values to preserve the integrity and usability of datasets. Missing data can result from many causes, such as errors in data collection or technical failures, which, if not taken care of, will lead to biased results and lowered statistical power. Technically, imputation methods range from simple to quite complex: from replacing missing values with means or medians to using machine learning algorithms. More complex techniques give more precise estimates by using patterns and relationships in data.

In a time series analysis, imputation comes with some unique challenges and considerations due to the sequential and temporal nature of the data. Missing values can disrupt the continuity of patterns and trends in a time series. Therefore, imputation techniques have to be chosen carefully, keeping in mind the temporal dependencies that are inherent in the dataset. The objective is to try to fill the gaps in such a way that temporal consistency is preserved and the periodicity/seasonality of the data is maintained.

A. Baseline Imputers

Based on already studied methods mentioned below, these imputers will be used as a baseline:

1) Linear Interpolation: Imputation by Linear Interpolation has been used in recent studies [2] and is a simple technique in imputing missing values, where a line is simply drawn between values on either side. Though it is one of the simplest techniques, it often gives quite a good initial approximation and tends to be used as a very strong baseline.

2) K-Nearest Neighbors (KNN) Imputer: The KNN Imputer is another popular approach also found in the literature [3]. It predicts missing data using the k-nearest neighbors and assigns their values to replace gaps. By definition, KNN preserves local data patterns and tends to perform well on datasets where the data remains relatively stable.

3) Gaussian Mixture Model (GMM) Imputer: This GMM imputer, as detailed in [4], imputes the missing data by considering the observed values as a mixture of Gaussian distributions. It offers considerable flexibility to model complex probabilistic patterns and, therefore, handles intricate structures in the underlying data quite well.

B. Foundation Models for Time Series: Introducing Chronos

In the past several years, a host of deep learning models, such as RNNs, LSTMs, and even GANs as mentioned in 'Deep Learning for Multivariate Time Series Imputation: A

Survey' [5], have found broad applications in handling time series data. Most of them, however, require a lot of tuning and a huge amount of task-specific data. Foundation models, similar to those developed for natural language processing, are pre-trained on diverse datasets to enable generalization across tasks.

The foundation model for time series forecasting developed by Amazon is called Chronos [6]. It is expected to capture long-term dependencies for sequential data in the transformer-based architecture it is based on, which makes pre-training so useful for Chronos to adapt to newer datasets with no sort of retraining. It's fast and easy to scale.

The Chronos architecture, shown in Fig. 1, is optimized for two unique challenges with time series data: the treatment of temporal variability and accurate probabilistic forecasting. This study leverages Chronos to explore how well this foundation model performs against established imputation methods.

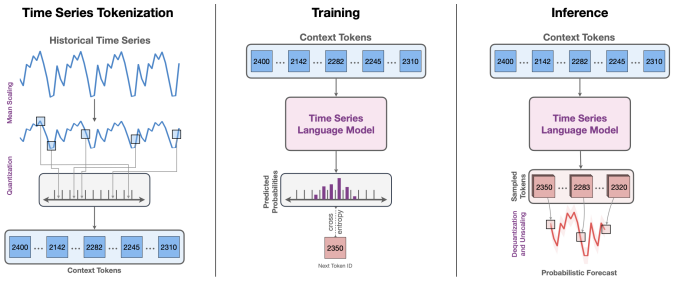

Fig. 1. Chronos architecture optimized for time series data, showing its transformer-based components. Retrieved from [6].

II. Methodology

Using the studies mentioned as references for the choice of imputation methods, a comparison with the Foundation Model Chronos is conducted to determine its effectiveness in handling missing values in time series data.

To obtain the results of this comparison, the following methodology was used:

1) Dataset preparation: The original dataset is prepared by introducing missing values into it to simulate a variety of impactful scenarios;
2) Using the different imputation methods identified, the missing values were imputed;
3) Imputation quality check using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) [7], and Auto Correlation (AC) [7];
4) Results comparison to identify the pros and cons specific to each method of imputation.

This approach ensures a fair evaluation, highlighting how Chronos fares as a time series imputer.

A. Data Preparation

To facilitate a fair comparison, missing values were artificially introduced in three ways:

1) 20% of data removed in random rows: Missing values were placed at random into the datasets, in a specified percentage in all columns except the target column. This is similar to numerous real-world scenarios where data can be absent at random intervals.

2) 20% of data removed in 5 sequences: To look at the methods in more structured missing data patterns, missing values were intentionally added in batches. These NaNs were placed over certain ranges of rows to simulate conditions where consecutive sequences may have partial data.

3) 20% of data removed in 1 sequence: For this method, missing values were placed as a single contiguous block, simulating conditions where a large section of data is missing.

B. Imputation Methods

This work compares the performance of the baseline imputation methods mentioned and the Foundation Model Chronos to deal with missing data in time series. The imputation methods were implemented as follows:

1) Linear Interpolation: Linear interpolation was used to fill missing values by computing a straight-line interpolation between neighboring data points [2]. To handle any remaining gaps, forward and backward filling methods were applied as a fallback.

2) K-Nearest Neighbors (KNN) Imputer: The KNN imputer is the `KNNImputer` function available in *Scikit-learn*. Missing data points were substituted with the average of the k-nearest neighbors [3].

3) Gaussian Mixture Model (GMM) Imputer: For the GMM imputer, the best-fitting number of Gaussian components was determined using the Bayesian Information Criterion (BIC). The model was then trained on observed values, and missing data points were imputed by sampling from the best-fitting Gaussian mixture for each column [4].

4) Foundation Model Chronos: The Chronos Imputer was implemented using the `autogluon.timeseries` library, developed by Amazon to make the Foundation Model accessible for time series tasks [5]. The imputation process was designed to adapt the forecasting abilities of Chronos and, as illustrated in Figure 2, follows an iterative workflow to handle missing values:

1) Receive the input dataset containing the time series.
2) Fits the Chronos model with the available data to initialize its predictive capabilities.
3) Identify and locate missing values in the dataset.
4) Gather past context of the missing values to ensure temporal dependencies are captured.
5) Predict the missing values using the Chronos model based on historical patterns and known covariates.
6) Fill the missing values in the dataset with the predictions and iteratively repeat the process until all missing values are filled.

C. Evaluation Metrics

The performance of the imputation methods is evaluated using three kinds of metrics:

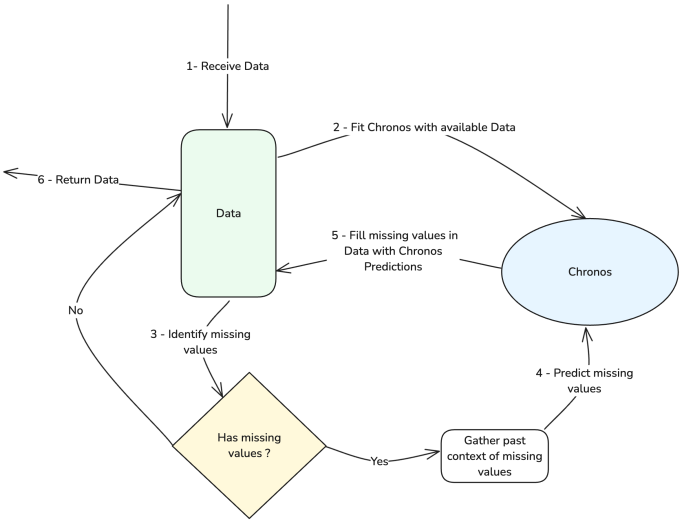

Fig. 2. Chronos imputation process.

- **Regression Errors**: A prediction was performed on the target variable using *Random Forest* and *Gradient Boosting* regressor models. The predicted values were compared to the actual values, and performance was evaluated using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) [7]. The final errors were averaged across the regression models to summarize the results. These errors evaluate how well the imputations maintain the predictive relationships in the data.
- **Imputation Errors**: Using the saved positions of the missing values, the actual and imputed values are compared for each column. Since the errors are normalized, the mean error across all imputed columns is calculated. The errors considered are as follows:
  - Normalized Mean Absolute Error (N-MAE): The mean absolute error normalized by the range of the actual values.
  - Normalized Root Mean Square Error (N-RMSE): The root mean square error normalized by the range of the actual values.
- **Autocorrelation Analysis**: The autocorrelation function (ACF) [8] measures the correlation of a dataset with its lagged versions, where a lagged version refers to the same data shifted by a certain number of time steps. In this study, the ACF values were calculated for the missing data points in each column. The overall mean ACF value of the missing data was then calculated by averaging the ACF values across the columns. These ACF values vary depending on the approach used to introduce missing values (Random, 5 short sequences, 1 long sequence). For each approach, the ACF values of the original missing data were compared with those of the imputed missing data, calculated in the same way for each method.

D. Row generation Analysis

In order to understand the differences in predictive performance among the different imputation methods, an additional experiment was conducted. In this experiment, an attempt was made to generate n values after the end of the dataset. An initial dataset was sampled from the original dataset with the minimum number of rows required for Chronos to function. Then, n NaN values were added across all features in the time series, extending the end of the dataset. This dataset, now containing missing values, was passed through the imputation methods being compared: KNN and Chronos, to generate the n values.

The aim was to fill in the missing values using each method to enable a comparison of their predictive performance. The results were analyzed by visualizing the imputed dataset alongside the actual dataset, allowing for an evaluation of each method's effectiveness in preserving the temporal structure, continuity, and underlying dynamics of the time series.

E. Experimental Setup

The experimental setup was designed to provide a fair and reproducible comparison of all imputation methods.

Dataset: The dataset, retrieved from [9], has a shape of (17420, 8), and its description is provided in Table **I**.

TABLE I
DESCRIPTION OF FEATURES

Column Name	Description
date	The recorded date
HUFL	High Useful Load
HULL	High Useless Load
MUFL	Middle Useful Load
MULL	Middle Useless Load
LUFL	Low Useful Load
LULL	Low Useless Load
OT	Oil Temperature (target)

Missing Data Simulation: The missing data was introduced using the described approaches, always masking 20% of the entries across all columns except the target variable. The original datasets, which contained no missing values, were retained to serve as a baseline for the comparisons.

Imputers Usage:

- **Linear Interpolation** and **KNN Imputer**: These methods were implemented with the default parameters and no additional tuning. The default k in the KNN Imputer is 5.
- **GMM Imputer**: The number of components was selected dynamically for each column based on the best Bayesian Information Criterion (BIC) score, ranging from 1 to 10 components.
- **Chronos**: The Chronos Bolt Base model was used without fine-tuning, as the additional runtime and computational resources required for fine-tuning were not justified for this study's objectives. The missing values

were imputed iteratively by using the minimum between the length of the sequence of missing values and the maximum prediction length of the model. This process repeated until all missing values were filled.

Chronos Imputer details: The choice to perform this process iteratively was due to the constraints imposed by the *AutoGluon framework*, which set the maximum prediction length of the Foundation Model Chronos to 60. Additionally, it is important to note that the model requires a minimum context length of 121 to initialize the model and at least 1 predicting context to mark the start of the prediction timestamp.

In this study, the initial context was provided as follows: NaN values were dropped and the rest of the entire time series served as context to initialize the model. Then each predicting context size was calculated : '$\min(\text{maximum prediction length}, \text{NaN sequence length}) \times 4$'. This ratio was determined to be optimal because increasing the prediction context results in higher computation time and costs, while decreasing it leads to poorer performance.

Chronos required significantly more time to execute compared to other imputation methods. To address this, multiprocessing was employed to parallelize the imputation process across columns. However, even with these optimizations, Chronos still exhibits a longer runtime than the other methods due to its computational complexity. The average of the final runtimes can be visualized in Figure 3.

Fig. 3. Average runtime of imputation methods, demonstrating the computational demand of Chronos compared to the other methods.

Hardware and Resource Usage: All experiments were performed on a system featuring an AMD EPYC-Milan processor (14 cores, 2.449 GHz), an NVIDIA RTX 6000 Ada Generation GPU, and 112 GB of RAM. The imputation methods Linear Interpolation, KNN, and GMM performed well on the CPU and introduced almost no computational overhead.

In comparison, Chronos required using the GPU due to the computational complexity of its transformer-based model. The architectural design incorporates attention mechanisms that address long-term dependencies alongside iterative imputation procedures, both of which require considerable computational resources. On average, the Chronos system consumed around 1.4 GB of GPU memory for each process (for each column imputation in parallel), with GPU usage averaging 32% and reaching a maximum of 76%.

III. Results

This section explains the results from the comparison of the four imputation methods: Linear Interpolation, K-Nearest Neighbors (KNN), Gaussian Mixture Models (GMM), and the Chronos Imputer. The results are presented in terms of regression errors, imputation errors, and a study of the autocorrelation.

A. Regression Errors

Figure 4 shows the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) [7] of the regression exercise performed on the original and the imputed datasets.

The objective here is not just to minimize errors, but to compare the regression scores and identify which imputation method achieves errors closest to those of the original dataset. For example, in the "20% Removed in Random Rows" approach, the KNN imputation method produces a lower error than the original dataset. This means that the KNN method may have introduced a bias or overfitted to the missing data pattern, rather than accurately representing the original data distribution.

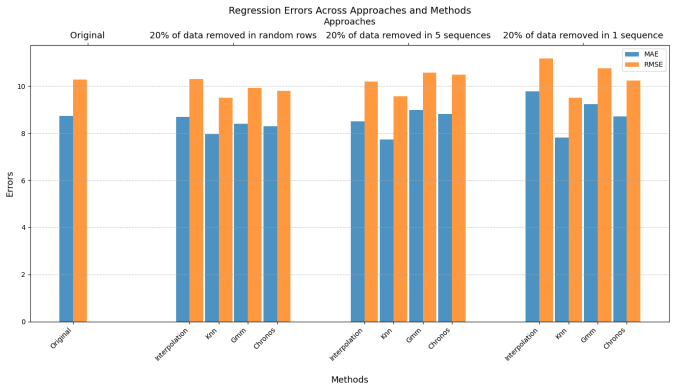

Fig. 4. Regression errors (MAE and RMSE) across different missing data approaches and imputation methods [7].

With this in mind, Table II presents the absolute differences in regression error from the original dataset for each method across all approaches. The results indicate that the performance of imputation methods depends on the missing data pattern. For "20% Removed in Random Rows" and "20% Removed in 5 Sequences," Interpolation achieves the lowest RMSE, while Chronos has the lowest MAE, making them the most effective techniques for these particular scenarios. However, for "20% Removed in 1 Sequence," Chronos is the best performer with the lowest difference to the original error. These results show that as the sequence becomes longer, temporal consistency is lost with Linear Interpolation but maintained by Chronos.

TABLE II
ABSOLUTE DIFFERENCE FROM THE ORIGINAL DATASET REGRESSION ERROR

Approach	*Method*	*MAE*	*RMSE*
20% Removed in Random Rows	Chronos	0.436	0.491
	GMM	0.331	0.374
	Interpolation	**0.029**	**0.005**
	KNN	0.773	0.790
20% Removed in 5 Sequences	Chronos	**0.095**	0.205
	GMM	0.267	0.290
	Interpolation	0.211	**0.103**
	KNN	0.987	0.712
20% Removed in 1 Sequence	Chronos	**0.0242**	**0.053**
	GMM	0.509	0.477
	Interpolation	1.053	0.893
	KNN	0.911	0.773

B. Imputation Errors

Figure 5 shows the normalized imputation errors (N-MAE and N-RMSE) for each method in three different missing data approaches.

In the random missing data approach, Linear Interpolation has the lowest errors, which is expected due to its simplicity and effectiveness in such cases. On the other hand, Chronos performs best in the 5-sequence-gap scenario, outperforming both Linear Interpolation and KNN by making predictions closer to the actual values. But in the 1-sequence-gap case, the gap length is too large even for Chronos to handle appropriately. Therefore, although there exists some temporal inconsistency loss, the Interpolation and KNN methods perform better than Chronos regarding the accuracy of imputation results.

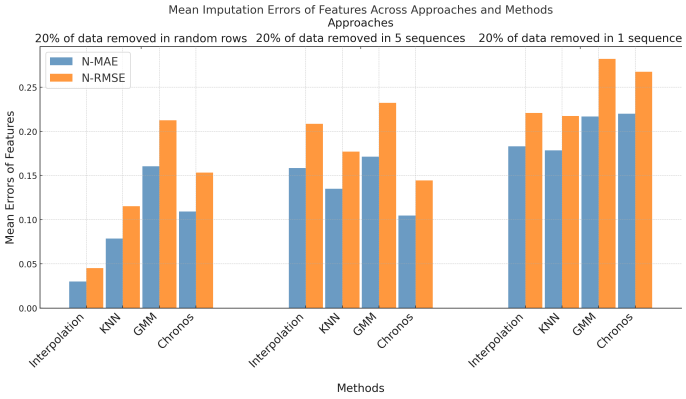

Fig. 5. Imputation errors (N-MAE and N-RMSE) across different missing data scenarios and methods.

C. Autocorrelation Analysis

Fig. 6 shows the ACF values of the various imputation methods – Linear Interpolation, KNN, GMM, and Chronos – for the three missing data approaches: random rows, 5-sequence gaps, and 1-sequence gaps. The ACF indicates how data points at different time lags are correlated.

Under the scenario of random missing data (left frame), the original data does not have much temporal dependency and all methods replicate the ACF quite well (all lines are near the original one). As the input to the imputation function only contains rows that have missing values, the temporal consistency is naturally reduced for this approach because the missing values aren't in a sequence.

In contrast, under the 5-sequence and 1-sequence gaps—middle and right frames, respectively—where temporal continuity is very important, neither Linear Interpolation nor KNN manage to retain long-range dependencies. Chronos better approximates the original ACF, particularly in the case of 5-sequences, but struggles with the larger gap observed in the 1-sequence scenario. This indicates that Chronos preserves temporal consistency in structured gaps but a bigger gap makes the little deviations from each imputed value cascade into bigger errors due to the iterative nature of the Chronos imputer.

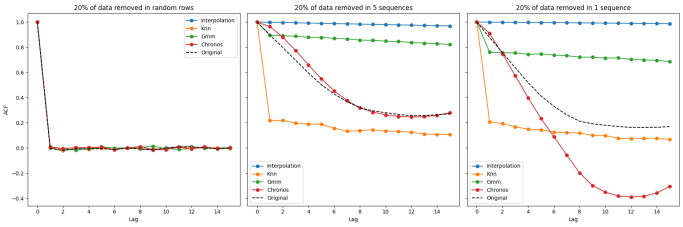

Fig. 6. Autocorrelation analysis across imputation methods for random, 5-sequence, and single-sequence missing data scenarios [8].

D. Row Generation Visualization

In Fig.7, the original feature plots of the sample time series are shown. In Figs.8 and 9, the predicted values for KNN and Chronos are displayed, starting from the vertical red line.

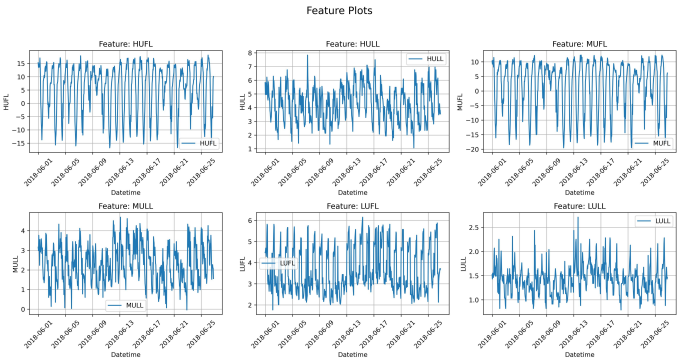

Fig. 7. Original Sample Dataset

The figures clearly show that Chronos outperforms KNN in imputing values without context. Chronos effectively retains temporal patterns, minimizes discontinuities, and handles complex dynamics inherent in the dataset. This consequently leads to better reconstruction of missing values with smoother transitions and fewer errors. These findings highlight the robustness of Chronos in time-series imputation.

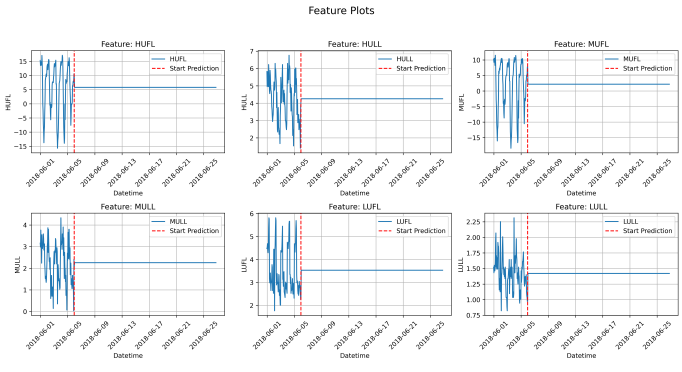

Fig. 8. Sample Dataset Imputed with KNN

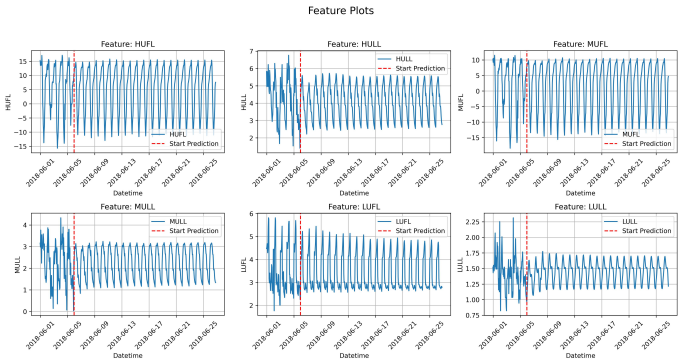

Fig. 9. Sample Dataset Imputed with Chronos

IV. Discussion

A. Objective and Findings

The existing imputation techniques are found to have performance trade-offs with the Chronos model, which is a foundation model specifically designed for time series forecasting [6]. Even with the data structured but missing, in this work, Chronos tends to be a lot more computationally demanding and time-consuming compared to some methods like Linear Interpolation.

B. Comparison of Methods

The Linear Interpolation has been a very strong baseline in handling random missing data, giving considerably low imputation errors in most use cases. However, because of its inability to model temporal dependencies, it performs poorly on sequential missing patterns.

KNN is effective for keeping the local pattern and has variable performances across different scenarios, reflecting higher regression and imputation errors regarding structured gaps. But on sequential missing patterns with extended length, it has the same problem as Linear Interpolation, performing poorly when trying to model temporal dependencies.

Despite its probabilistic nature, GMM demonstrated higher error rates across all scenarios, hence indicating its challenges in modeling complex temporal dependencies, therefore being less effective in time series imputation tasks.

Chronos, however, proved very strong at controlling sequential gaps by ensuring excellent performance with its predictions while preserving time consistency. Such iterative, transformer-based architecture has thus shown good skills in effectively capturing long-range dependencies, which obviously also introduces an advantage that this added computational expense may render not that easy to manage in general applications.

C. Practical Implications

It seems that these results make Chronos particularly useful in scenarios where maintaining temporal consistency is critical and/or where the absence of data follows a clearly regular pattern. While the choice of imputation algorithm may depend on the use case, Chronos is the best option for filling sequences that are short enough to avoid cascading errors but long enough for KNN and Linear Interpolation to struggle with maintaining temporal consistency. However, on longer sequences, where Chronos encounters cascading errors, KNN may be a better choice despite its lack of temporal consistency, which may have implications in cases requiring accurate or meaningful data. In this context, Chronos still adds value by addressing these shortcomings, particularly when such use cases are in play.

Acknowledgment

This work was partially financed by National Funds through the Portuguese funding agency, *FCT - Fundação para a Ciência e a Tecnologia*, within project LA/P/0063/2020. DOI 10.54499/LA/P/0063/2020 and by the European Union through the *EnerTEF Project* under the Grant Agreement Number 101172887.

References

[1] P. J. Brockwell and R. A. Davis, 'Introduction to Time Series and Forecasting', 2nd ed. Springer, 2002. [Online]. Available here.

[2] S. Ahmed, 'Filling Missing Data Using Interpolation Methods: Study on the Effect of Fitting Distribution', *ResearchGate*, 2013. [Online]. Available here.

[3] T. M. Beretta and C. Santaniello, 'A Study of K-Nearest Neighbour as an Imputation Method.' *ResearchGate*, 2011. [Online]. Available here.

[4] J. Abrevaya and S. G. Donald, 'A GMM Approach for Dealing with Missing Data on Regressors', *The Review of Economics and Statistics*, vol. 99, no. 4, pp. 657–662, Oct. 2017. [Online]. Available: here.

[5] A. Kumar and B. Smith, 'Deep Learning for Multivariate Time Series Imputation: A Survey', *arXiv preprint*, arXiv:2402.04059, 2024. [Online]. Available here.

[6] J. Zhang, S. Agarwal, and B. Settles, 'Chronos: Learning the language of time series', *Amazon Science*, 2023. [Online]. Available here.

[7] W. Wang and Y. Lu, 'Analysis of the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) in Assessing Rounding Model', *IOP Conference Series: Materials Science and Engineering*, vol. 324, no. 1, p. 012049, 2018. [Online]. Available here.

[8] D. K. Ludlow and H. T. Shapiro, 'Addressing Autocorrelation in Time Series Data: A Comparison of Four Analytic Methods Using Data from College Course Evaluations' *The Great Lakes Mathematics Journal*, vol. 44, no. 1, pp. 23–34, 2022. [Online]. Available here.

[9] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, 'Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting', in *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 11106–11115, 2021. [Online]. Retrieved from here.