Handling Missing Values in Time Series Forecasting

Giorgos Moysiadis

Department of Informatics

Aristotle University of Thessaloniki

gmousiad@csd.auth.gr

Nikos Mikrou

Department of Informatics

Aristotle University of Thessaloniki

nikomikr@csd.auth.gr

# Abstract

Texty text.

# Introduction

Every data scientist wishes to have access to clean, easily understandable data. Unfortunately, reality isn’t quite so sweet. We must devote a significant portion of our time to data exploration and cleaning. However, a good exploratory analysis is essential for extracting the most useful insights and producing better results.

A detailed overview of the dynamics present in the data is a good starting point for making the best decisions in the context of predictive applications. There are numerous options available, ranging from predictive architecture to preprocessing techniques. One of the most important, yet undervalued, is the method for dealing with missing values.

The absence of observations does not all have the same meaning. Missing values could be due to a lack of information or to issues with the ingestion process. In most cases, there is no universal rule that applies to all situations for filling in missing values. What we can do is comprehend the realm of analysis. When working with time series, we must consider the system’s correlation dynamics as well as the data’s temporal dependencies.

# Missing Values

Missing data is a widespread issue. Data may be missing because it was never collected, records were destroyed, or a variety of other factors. In the medical field, a patient’s respiratory rate may have been measured by not being recorded (perhaps because it was deemed unnecessary/unimportant) [1]. An imputation algorithm can be used to estimate missing values based on observed/measured data, such as the patient’s systolic blood pressure and heart rate [2, 3]. A significant amount of search has gone into developing imputation algorithms for medical data. Many other applications, such as image concealment, data compression, and counterfactual estimation, make use of imputation algorithms [4, 5].

Missing data can be categorized into three types: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). MCAR is when the missingness occurs randomly, and the missing data is unrelated to any other variables in the dataset. MAR is when the missingness is dependent on some observed variables in the dataset. Finally, MNAR is when the missingness is dependent on unobserved variables in the dataset, which can lead to bias in the analysis.

In machine learning, there are several approaches for dealing with missing data. One method is to remove any samples that have missing data, also known as list wise deletion. This method is simple, but it can result in information loss and a smaller sample size. Another approach is to use estimates to fill in the missing values, such as mean or median imputation, mode imputation, or regression imputation. However, if the imputed values are not representative of the true values, these methods can introduce bias.

Multiple imputation is a more advanced technique for dealing with missing data that generates multiple imputed datasets, each with a different imputed value, and then combines the results from each dataset. Another approach is to utilize probabilistic models, such as Bayesian networks or Gaussian processes, to model the uncertainty in missing data and provide more accurate estimates.

# Approach

We attempt to solve a time series forecasting task with missing values. We investigate various strategies for dealing with missing observations in time series. We attempt to compare various available methodologies, ranging from standard linear interpolation to more sophisticated techniques. Our experiments are carried out using only scikit-learn.

## Experiment setup

Our scope is to test how different imputation strategies affect the performance of time series forecasting. For this purpose, we first generate some hourly synthetic time series with daily and weekly seasonalities. Secondly, we artificially generate some missing intervals and insert them into our time series.

We are also going to fetch data from Google Trends. Google Trends works by analyzing the volume of searches for specific keywords or phrases entered into the Google search engine. It allows users to customize their searches in several ways, including:

1. Timeframe: Users can specify the time period for which they want to see search data, ranging from the past hour to the past five years.
2. Geography: Users can choose to view search data for specific countries, states, or cities.

Google Trends also provides users with several tools for analyzing and visualizing the data. These include:

1. Interest over time: This graph shows how the popularity of a search term has changed over time. It allows users to compare the popularity of multiple search terms and to see how events or news stories may have affected search trends.
2. Related queries: This feature shows the other search terms that are commonly searched for in conjunction with a particular term. This can help users identify related topics and interests.

## Methods

### IterativeImputer

We have found that scikit-learn provides the IterativeImputer class that works by iteratively filling in missing values in a dataset with a regression model trained on observed values. The missing values are imputed based on the values predicted by the regression model in each iteration, and the regression model is updated using the imputed values. This procedure is repeated until convergence or a predetermined number of iterations is reached. IterativeImputer has an advantage over other imputation techniques in that it can handle datasets with mixed data types, such as continuous and categorical variables. It accomplishes this by treating categorical variables as dummy variables and including them as binary predictors in the regression model. It also has the ability to estimate the uncertainty in the imputed values. This is accomplished by generating multiple imputations for the missing values and calculating the variance between the imputations. When creating an instance of the IterativeImputer class, the user can specify the number of imputations to be generated.

### KNN Imputer

The basic idea behind KNN imputation is to replace missing values in a dataset with values that are similar to the values around them. This is accomplished by locating the K nearest neighbors to the missing value, where K is a user-defined parameter indicating the number of neighbors to consider. To identify the K nearest neighbors, the algorithm computes the distance between the missing value and all of the other observed values in the dataset. Once the K nearest neighbors have been identified, the missing value is imputed by taking the mean (for continuous data) or mode (for categorical data) of the values of the K nearest neighbors.

How to choose the value of K is an important consideration when using KNN imputation. If K is too small, the imputed values may be sensitive to data noise, whereas if K is too large, the imputed values may be biased toward the mean or mode of the entire dataset. There is no universally optimal value for K, which is usually determined by trial and error or cross-validation.

Another critical consideration is how to handle missing values that occur in close proximity to other missing values. One method is to impute the missing values iteratively, imputing each value based on the values imputed in the previous iteration. This procedure is repeated until convergence or a predetermined number of iterations is reached.

KNN imputation is easy to implement, can handle datasets with mixed data types, and does not require assumptions about the distribution of the data. On the other hand, it may not perform well when there are a large number of missing values. Lastly, the choice of the distance metric and the value of K can significantly affect the quality of the imputed values.

# T-Spiral

T-Spiral is a library that takes advantage of using scikit-learn and provides estimators for time series forecasting. More specific T-Spiral provides four forecasting techniques.

1. *Recursive Forecasting*

The fitted estimator is called iteratively to predict multiple steps ahead.

1. *Direct Forecasting*

A regressor is fitted on the lagged data for each time step to forecast.

1. *Stacking Forecasting*

Multiple recursive time series forecasters are fitted and combined on the final portion of the training data.

1. *Rectified Forecasting*

Multiple recursive time series forecasters are fitted on different sliding window training bunches.

## Dataset generation

To evaluate the performance of the algorithms mentioned above, a collection of artificially generated datasets were created. In order to generate the data, use utility functions which will be explained analytically.

The first function generates a sinusoidal waveform with added noise. It takes four parameters 1) ‘timesteps’: the number of time steps or samples in the waveform 2) ‘amp’: the amplitude of the sinusoidal waveform 3) ‘freq’ : the frequency of the sinusoidal waveform 4) ‘noise’ : the standard deviation of the noise to be added to the waveform. Overall, the function create a time array, generates random noise and calculates the sinusoidal waveform using the following formula

*amp \* sin(X \* (2\*pi / freq)) , X time steps*

The second function generates a random walk time series and it takes timesteps and noise as parameters. It generates random numbers with a normal distribution and then cumulatively summing those numbers. The resulting array represents the random walk time series.

The third function generates a step function for time series by creating an array that repeats a pattern every 24 time steps and adding a shift to adjust the timing. It then converts this array into a binary step function by taking modulo 7 and casting the result to integers. Finally, it scales the step function according to the desired amplitude and returns the resulting time series.

Finally, utilize a function that generates a time series by combining the above functions with different frequencies and noise levels.

Following, demonstrates time series patterns and time series simulation as derived from the functions that mentioned above.

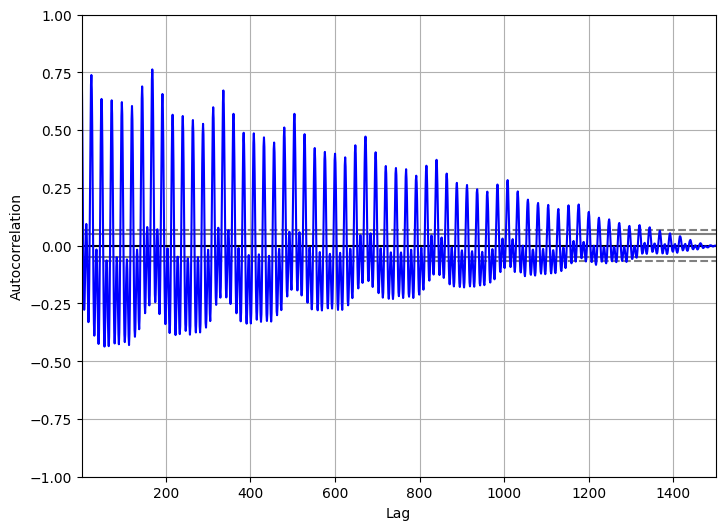


Figure 1. Time series pattern

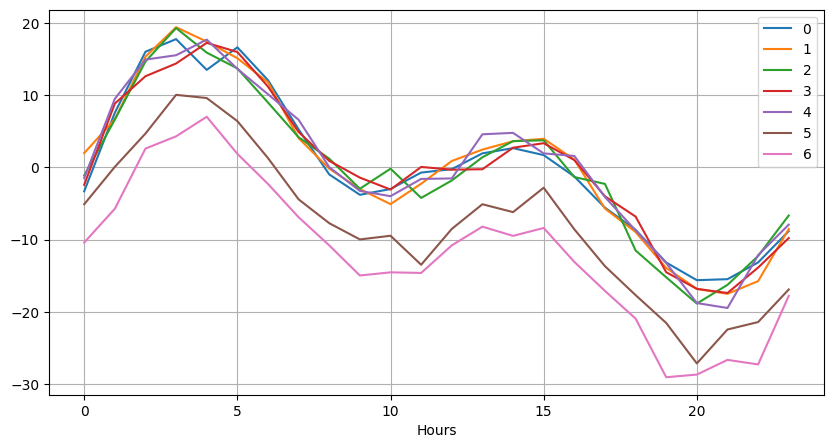


Figure 2. Time series pattern

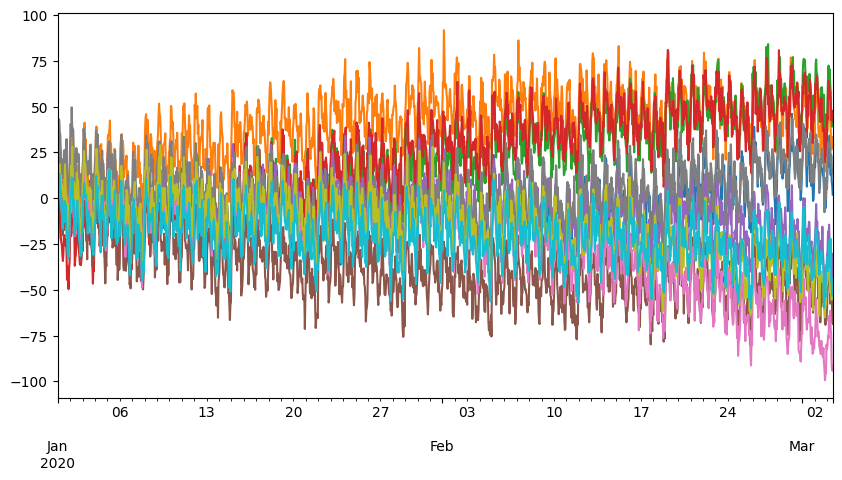


Figure 3. Time series simulation

Last but not least, simulate missing value intervals and insert them into time series. The following graph demonstrates the time series with missing values.

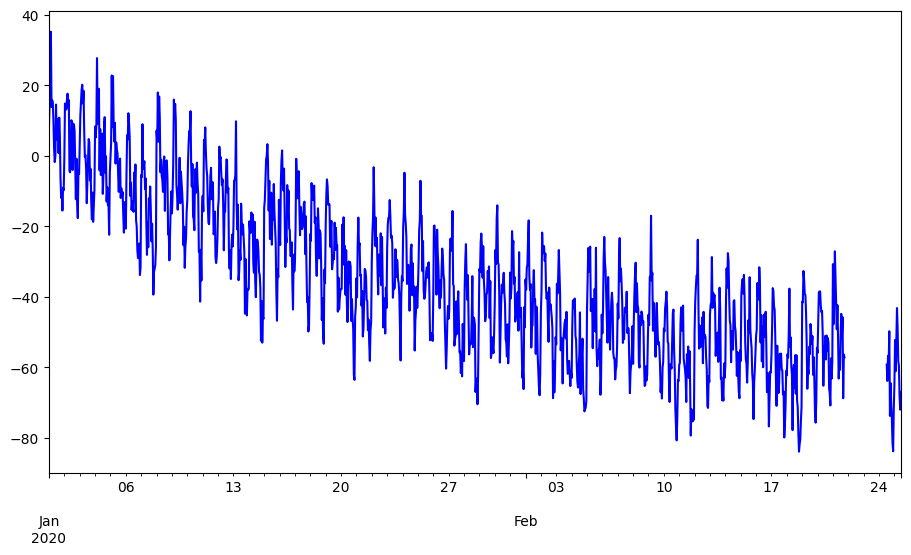
**

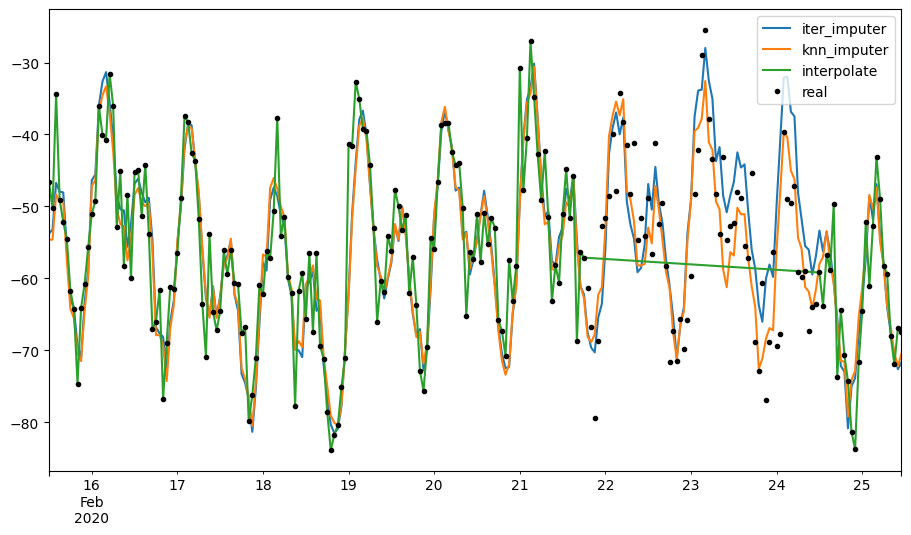
Figure 4. Time Series with missing values

## Results T-Spiral

At this point, starting to test how forecasting accuracy changes according to the methodology used to fill in the missing values.

1. *Reconstruction Ability*

Focusing on the reconstruction ability, iterative and KNN imputations look very promising. With a simple interpolation, we are limited to connecting the closer observations without taking the nature of the system into account. Using a KNN or an iterative imputation, we can replicate the seasonality patterns and the underlying dynamics present in the data.

Figure 5. Reconstruction ability

1. *Evaluation*

For the evaluation, utilize metrics such as Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) in order to assess the accuracy of the forecasts.

RMSE, or Root Mean Square Error, measures the average difference between the predicted values and the actual values in the forecast. It calculates the square root of the average of the squared differences between the predicted and actual values. RMSE is commonly used in forecasting to assess the overall goodness of fit and accuracy of the model's predictions. It gives higher weight to larger errors, making it sensitive to outliers. In the context of comparing different forecasting models or techniques, lower RMSE values indicate better accuracy.

MAE, or Mean Absolute Error, measures the average absolute difference between the predicted values and the actual values in the forecast. It calculates the average of the absolute differences between the predicted and actual values. MAE gives equal weight to all errors, regardless of their magnitude, making it less sensitive to outliers compared to RMSE. MAE provides a measure of the average forecasting error and represents the average magnitude of the errors. It is commonly used to evaluate the forecast accuracy and the level of precision in predicting the future values.

* Interpolation forecast : RMSE = 9.29

By guessing the missing values based on the nearby observed values, the interpolation technique filled in the gaps left by the missing data. The interpolation forecast produced an RMSE of 9.29, meaning that there was an average discrepancy of 9.29 between the interpolated values and the actual values.

* Iterative Imputer Forecast : RMSE = 6.95

The iterative imputation method estimated the missing data by an iterative procedure and a ridge regression model. With an RMSE of 6.95, the iterative imputer forecast beat interpolation. With an average difference between the actual values and the imputed values of around 6.95, this finding shows that the iterative imputer offered forecasts that were more accurate than those produced using interpolation.

* K-nearest Neighbors Imputer Forecast : RMSE = 6.70

The values of the nearest neighbors in the data were taken into account as part of the k-nearest neighbors imputation procedure, which estimated the missing values. It combined a ridge regression model with a k-nearest neighbors imputation technique. With an RMSE of 6.70, the k-nearest neighbors imputer forecast fared better than interpolation. With an average difference between the actual values and the imputed values of about 6.70, this result indicates that the k-nearest neighbors imputer delivered more accurate forecasts than interpolation.

Overall, compared to interpolation, the iterative imputer and k-nearest neighbors imputer approaches both showed better forecasting ability.

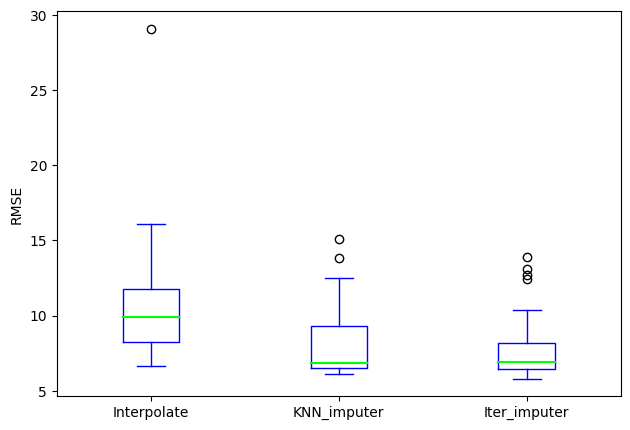


Figure 6. RMSE comparison

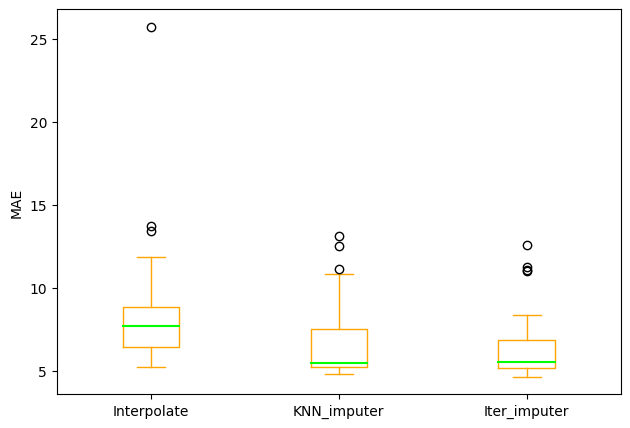


Figure 7. MAE comparison

# Our Implementation

Having investigated classical techniques and reviewing their results we went ahead and experimented by trying to implement our own technique to fill the missing values in a time series.

Instead of using synthetic data we will experiment with real data, using Google Trends via the “pytrends” library for Python. Our main focus will be the “Interest over time” value that is represented as a time series in various time frames. In ”Interest over time” the numbers represent search interest relative to the highest point on the chart for the given region and time. A value of 100 is peak popularity for the term. A value of 50 means that the term is half as popular. A score of 0 means that there was not enough data for this term.

Με ευρετικους/μπακαλικους τροπους θα φτιαχνουμε τα missing data μας απο τα trends.

Google Trends a dataframe containing the interest\_over\_time of a query during a specific timeframe based on a geolocation. If, during that timeframe, it returns an empty dataframe, because there was no data to be found, we create our own dataframe containing zeros for that timeframe.

# query

The top-k dominating query retrieves the *k* objects from the

# Conclusions

Three approaches are used in this assignment to solve implementations are scalable and are dependent on the

##### References

1. Yoon, J., Davtyan, C., and van der Schaar, M. Discovery and clinical decision support for personalized healthcare. IEEE journal of biomedical and health informatics, 21(4):1133–1145, 2017.
2. Yoon, J., Zame, W. R., and van der Schaar, M. Deep sensing: Active sensing using multi-directional recurrent neural networks. In International Conference on Learning Representations, 2018c.
3. Barnard, J. and Meng, X.-L. Applications of multiple imputation in medical studies: from aids to nhanes. Statistical methods in medical research, 8(1):17–36, 1999.
4. Yoon, J., Jordon, J., and van der Schaar, M. GANITE: Estimation of individualized treatment effects using generative adversarial nets. In International Conference on Learning Representations, 2018a
5. Kreindler, D. M. and Lumsden, C. J. The effects of the irregular sample and missing data in time series analysis. Nonlinear Dynamical Systems Analysis for the Behavioral Sciences Using Real Data, pp. 135, 2012.