

UNIVERSIDADE FEDERAL DE MINAS GERAIS
Graduate Program in Electrical Engineering - PPGEE

Silvana Mara Ribeiro

**Imputation by Decomposition and by
Time Series Nature: Novel Imputation
Methods for Missing Data in Time
Series**

Belo Horizonte - Minas Gerais

21 de julho de 2022

Silvana Mara Ribeiro

**Imputation by Decomposition and by Time Series
Nature: Novel Imputation Methods for Missing Data in
Time Series**

Final thesis presented to the Graduate Program in Electrical Engineering of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

Supervisor: Cristiano Leite de Castro

Belo Horizonte - Minas Gerais

21 de julho de 2022

R484i

Ribeiro, Silvana Mara.

Imputation by decomposition and by time series nature [recurso eletrônico]: novel imputation methods for missing data in time series / Silvana Mara Ribeiro. - 2021.

1 recurso online (92 f. : il., color.) : pdf.

Orientador: Cristiano Leite de Castro.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f.89-92.

Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia Elétrica - Teses. 2. Análise de séries temporais - Teses. 3. Ausência de dados (Estatística) – Teses. 4. Ciências Sociais – Métodos Estatísticos – Teses. I. Castro, Cristiano Leite de. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

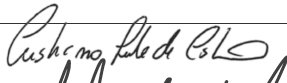
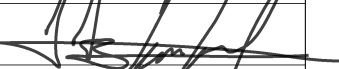

CDU: 621.3(043)



UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
Programa de Pós-Graduação em Engenharia Elétrica

**ATA DA 1239ª DEFESA DE DISSERTAÇÃO DE MESTRADO
DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO da aluna **Silvana Mara Ribeiro** - registro de matrícula de número 2019699871. Às 16:00 horas do dia 28 do mês de julho de 2021, reuniu-se na Escola de Engenharia da UFMG a Comissão Examinadora da DISSERTAÇÃO DE MESTRADO para julgar, em exame final, o trabalho intitulado "**Imputation By Decomposition And By Time Series Nature: Novel Imputation Methods For Missing Data In Time Series**" da Área de Concentração em Sistemas de Computação e Telecomunicações, Linha de Pesquisa Inteligência Computacional. O Prof. Cristiano Leite de Castro, orientador da aluna, abriu a sessão apresentando os membros da Comissão e, dando continuidade aos trabalhos, informou aos presentes que, de acordo com o Regulamento do Programa no seu Art. 8.16, será considerado APROVADO na defesa da Dissertação de Mestrado o candidato que obtiver a aprovação unânime dos membros da Comissão Examinadora. Em seguida deu início à apresentação do trabalho pela Candidata. Ao final da apresentação seguiu-se a arguição da candidata pelos examinadores. Logo após o término da arguição a Comissão Examinadora se reuniu, sem a presença da Candidata e do público, e elegeu o Prof. Cristiano Leite de Castro para presidir a fase de avaliação do trabalho, constituída de deliberação individual de APROVAÇÃO ou de REPROVAÇÃO e expedição do resultado final. As deliberações individuais de cada membro da Comissão Examinadora foram as seguintes:

| Membro da Comissão Examinadora | Instituição de Origem | Deliberação | Assinatura |
|--|-----------------------|-------------|---|
| Prof. Dr. Cristiano Leite de Castro - Orientador | (UFMG) | APROVADA |  |
| Prof. Dr. Luis Antonio Aguirre | DELT (UFMG) | APROVADA |  |
| Prof. Dr. Frederico Gadelha Guimarães | DEE (UFMG) | APROVADA |  |

Tendo como base as deliberações dos membros da Comissão Examinadora a Dissertação de Mestrado foi APROVADA. O resultado final de APROVAÇÃO foi comunicado publicamente à Candidata pelo Presidente da Comissão, ressaltando que a obtenção do Grau de Mestre em ENGENHARIA ELÉTRICA fica condicionada à entrega do TEXTO FINAL da Dissertação de Mestrado. A Candidata terá um prazo máximo de 30 (trinta) dias, a partir desta data, para fazer as CORREÇÕES DE FORMA e entregar o texto final da Dissertação de Mestrado na secretaria do PPGE/UFMG. As correções de forma exigidas pelos membros da Comissão Examinadora deverão ser registradas em um exemplar do texto da Dissertação de Mestrado, cuja verificação ficará sob a responsabilidade do Presidente da Banca Examinadora. Nada mais havendo a tratar o Presidente encerrou a reunião e lavrou a presente ATA, que será assinada pelo Presidente da Comissão Examinadora. Belo Horizonte, 28 de julho de 2021.



ASSINATURA DO PRESIDENTE DA COMISSÃO EXAMINADORA

I dedicate my dissertation work to my family and friends. A special feeling of gratitude to my loving and supportive parents, Cristina and Geraldo who always take care of me and encourage me and support me to follow through with all of my endeavours. My sister Paula, who inspires me always and is an amazing little sister. My husband, Rodrigo, who is always cheering me on and giving me strength to pursue my dreams. I also dedicate this dissertation to my friends, Leonardo, Priscila and Roberto, who have been guiding me and inspiring me now for so many years. To all my aunts, uncles and cousins who have been such a big part of my life and are always by my side, my special thank you.

Acknowledgements

I would like to thank my supervisor, Professor Cristiano Leite de Castro, whose expertise was invaluable in formulating the research and methodology. The work accomplished would not have been possible without your insightful feedback and it has brought this dissertation to a higher level.

"The truth has become an insult." - Chimamanda Ngozi Adichie, Half of a Yellow Sun

Resumo

Um passo importante, porém muitas vezes negligenciado, durante a análise de dados de séries temporais é a imputação de dados ausentes. Nessa dissertação, as características de séries temporais e mecanismos de perda são descritos para ajudar na identificação de qual método de imputação deve ser utilizado para imputar dados ausentes, juntamente com uma revisão bibliográfica de métodos de imputação e seu funcionamento. Os métodos de imputação recomendados pela literatura são utilizados para imputar dados sintéticos com diferentes características e os resultados são discutidos. Dois novos métodos de imputação de séries temporais são apresentados e comparados com métodos de imputação clássicos e métodos do estado-da-arte. O primeiro método de imputação apresentado é o de Imputação pelo Padrão. Esse método se baseia na premissa que utilizando-se o método de imputação recomendado pela literatura para cada padrão de série temporal se obterá os melhores resultados. Heurísticas de separação das séries temporais por padrão foram desenvolvidas. O segundo método apresentado é o de Imputação por Decomposição. Esse método consiste em decompor a série temporal e depois imputar cada um de seus componentes pelos métodos recomendados pela literatura. As combinações desses métodos e o filtro de Kalman também foram testados. Os métodos de imputação discutidos são utilizados para imputar dados de índices financeiros e rastreadores de instabilidade, dados sobre a COVID-19 e dados sobre a dengue. Predições são realizadas com os dados dos casos de estudo e os resultados são apresentados. Os resultados obtidos pelo método de Imputação por Padrão combinado com o filtro de Kalman são consistentemente satisfatórios, apesar de nem sempre obter os melhores resultados. O método de Imputação por Decomposição também obteve bons resultados, principalmente quando algum tempo foi gasto para investigar qual de suas variações se adequou melhor a cada conjunto de dados. No geral, ambos os métodos mostraram resultados similares e/ou melhores que os métodos de imputação clássicos.

Palavras-chave: Dados Ausentes, Séries Temporais, Métodos de Imputação, Padrão, Decomposição.

Abstract

Dealing with missingness in time series data is a very important, but oftentimes overlooked, step in data analysis. In this dissertation, the pattern of time series data and missingness mechanisms are described to help identify which imputation method should be used to impute missing data, along with a review of imputation methods and how they work. Recommended methods from literature are used to impute synthetic data of different pattern and the results are discussed. In this dissertation, two new methods to impute missing time steps are presented and compared to other classical imputation methods, as well as state-of-the-art methods. The first imputation method presented is Imputation by Pattern. This method is based on the premise that imputing the data using the literature-recommended methods will achieve the best results. Heuristics are proposed to separate the time series by pattern. The second imputation method presented is Imputation by Decomposition. This method consists in decomposing the time series in its components and then imputing them using the literature-recommended methods. The combination of these methods and the Kalman filter are also tested. The discussed imputation methods are used to impute a financial indexes and instability trackers data set, a COVID-19 data set and a deng data set and then predictions are made and the results are presented. The Imputation by Pattern method combined with the Kalman filter achieved consistently satisfactory results, although it did not always achieve the best results. The Imputation by Decomposition method achieved good results, specially when some time was spent investigating which variation worked better with each data set. Overall, both imputation method achieved similar, and in some cases, better results than the classical imputation methods.

Keywords: Missing Data, Time Series, Imputation Methods, Pattern, Decomposition.

List of Figures

| | |
|---|----|
| Figure 1 – Time series with trend and without seasonality | 20 |
| Figure 2 – Time series without trend and with seasonality | 20 |
| Figure 3 – White Noise | 21 |
| Figure 4 – Time series with trend and with seasonality | 21 |
| Figure 5 – Autocorrelation of time series with trend and without seasonality | 22 |
| Figure 6 – Autocorrelation of time series without trend and with seasonality | 22 |
| Figure 7 – Autocorrelation of time series with trend and with seasonality | 22 |
| Figure 8 – Autocorrelation of white noise | 23 |
| Figure 9 – Flowchart of methods to deal with missing data | 25 |
| Figure 10 – LSTM cell | 30 |
| Figure 11 – Flowchart of the Synthetic Data Experiment | 36 |
| Figure 12 – S&P 500 close index classified as BEAR or BULL | 49 |
| Figure 13 – COVID-19 Asia Confirmed cases | 51 |
| Figure 14 – Deng total cases | 51 |
| Figure 15 – Diagram of the methodology | 53 |
| Figure 16 – Diagram showing information about the original data sets and the final daily data set after the merger | 56 |
| Figure 17 – Diagram of the Data Preparation step of the COVID-19 data set | 58 |
| Figure 18 – Diagram of the Data Preparation step of the Deng data set | 59 |
| Figure 19 – BULL and BEAR real data and prediction made by model generated using the Kalman imputed data set | 65 |
| Figure 20 – BULL and BEAR real data and prediction made by model generated using the Pattern & Kalman imputed data set | 65 |
| Figure 21 – BULL and BEAR real data and prediction made by model generated using the Decomposition & Kalman imputed data set | 65 |
| Figure 22 – BULL and BEAR real data and prediction made by model generated using the Kalman imputed data set | 67 |
| Figure 23 – BULL and BEAR real data and prediction made by model generated using the Regression imputed data set | 68 |
| Figure 24 – BULL and BEAR real data and prediction made by model generated using the Missing Indicators imputed data set | 68 |

| | |
|--|----|
| Figure 25 – BULL and BEAR real data and prediction made by model generated using the Pattern & Kalman imputed data set | 68 |
| Figure 26 – Real COVID-19 data and prediction made by model generated using the LSTM imputed data set | 69 |
| Figure 27 – Real COVID-19 data and prediction made by model generated using the Pattern & Kalman imputed data set | 70 |
| Figure 28 – Real COVID-19 data and prediction made by model generated using the KNN imputed data set | 70 |
| Figure 29 – Real Deng data and prediction made by model generated using the Additive Decomposition imputed data set | 71 |
| Figure 30 – Real Deng data and prediction made by model generated using the Kalman imputed data set | 71 |
| Figure 31 – Real Deng data and prediction made by model generated using the Pattern & Kalman imputed data set | 72 |

List of Tables

| | |
|--|----|
| Table 1 – Average RMSE of the imputed data sets for each method | 37 |
| Table 2 – Separation Methods versus the error when classifying 500 time series of each pattern with $k_w = 2$ | 45 |
| Table 3 – Separation Methods versus the error when classifying 500 time series of each pattern with $k_w = 5$ | 45 |
| Table 4 – Separation Methods versus the error when classifying 500 time series of each pattern with $k_w = 10$ | 46 |
| Table 5 – Missingness analysis of each feature in relation to S&P 500 before merging the data sets | 54 |
| Table 6 – Missingness analysis of each feature after merging and re-sampling the data sets by each week’s mean | 55 |
| Table 7 – Missingness analysis of each feature after merging and re-sampling the data sets by each week’s Friday | 57 |
| Table 8 – Visual analysis of the autocorrelation plot of each feature | 57 |
| Table 9 – Missingness analysis of each feature of the COVID-19 data set | 58 |
| Table 10 – Missingness analysis of each feature of the Deng data set | 60 |
| Table 11 – Imputation Methods Characteristics | 62 |
| Table 12 – Comparison of the mean F1 scores and best F1 scores for each imputed data set and ranking. | 65 |
| Table 13 – Confusion Matrices of predictions made using the Financial Indexes data sets | 66 |
| Table 14 – Comparison of the mean F1 scores and best F1 scores for each imputed data set and ranking. | 67 |
| Table 15 – Comparison of the mean RMSE scores and best RMSE scores for each imputed data set and ranking. | 69 |
| Table 16 – Comparison of the mean RMSE scores and best RMSE scores for each imputed data set and ranking. | 71 |
| Table 17 – Comparison of the mean RMSE scores and best RMSE scores for each imputed data set and ranking. | 72 |

List of Algorithms

| | | |
|---|--|----|
| 1 | Pseudocode of the function <code>isWhiteNoise(ts)</code> | 41 |
| 2 | Pseudocode of the function <code>isSeasonal(ts)</code> | 42 |
| 3 | Pseudocode of the function <code>isTrended(ts)</code> | 43 |
| 4 | Pseudocode of the function <code>isTrendedAndSeasonal(ts)</code> | 43 |
| 5 | Pseudocode of the function <code>separate(df)</code> | 44 |
| 6 | Pseudocode of the generation of the synthetic data for testing | 47 |

List of abbreviations and acronyms

| | |
|------|---|
| UFMG | Universidade Federal de Minas Gerais |
| LSTM | Long Short-Term Memory |
| MCAR | Missing Completely At Random |
| MAR | Missing At Random |
| NMAR | Not Missing at Random |
| TBM | Temporal Belief Memory |
| GAIN | Generative Adversarial Imputation Network |
| RMSE | Root Mean Squared Error |

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 16 |
| 1.1 | Motivation | 16 |
| 1.2 | Objective | 17 |
| 1.3 | Contributions | 17 |
| 1.4 | Structure | 18 |
| 2 | Theoretical Background | 19 |
| 2.1 | Problem Formulation | 19 |
| 2.2 | Data Pattern | 20 |
| 2.3 | Missingness Mechanisms | 23 |
| 3 | Literature Review | 24 |
| 3.1 | General Purpose Imputation Methods | 24 |
| 3.1.1 | Mean imputation, median imputation, and mode imputation | 24 |
| 3.1.2 | Random Sample Imputation | 26 |
| 3.2 | Classical Imputation Methods for Time Series | 26 |
| 3.2.1 | Forward and Backward Filling | 26 |
| 3.2.2 | Multiple Imputation | 27 |
| 3.2.3 | Linear Interpolation | 27 |
| 3.2.4 | Spline Interpolation | 27 |
| 3.2.5 | Missing Indicators | 27 |
| 3.2.6 | Expectation Maximisation | 27 |
| 3.3 | State-of-the-art imputation Methods for Time Series | 28 |
| 3.3.1 | ST-2SMR | 28 |
| 3.3.2 | TBM - Temporal Belief Memory | 28 |
| 3.3.3 | GAIN - Generative Adversarial Imputation Network | 28 |
| 3.3.4 | Long Short-Term Memory - LSTM | 29 |
| 3.3.5 | Kalman Filter | 32 |
| 4 | Experiment with Synthetic Data | 35 |
| 4.1 | Methodology | 35 |
| 4.2 | Results | 37 |
| 5 | Novel Imputation Methods | 39 |
| 5.1 | Imputation by Decomposition | 39 |
| 5.2 | Imputation by Time Series Pattern | 41 |
| 6 | Case Studies | 48 |
| 6.1 | Financial Indexes and Instability Trackers | 48 |
| 6.2 | COVID-19 | 50 |
| 6.3 | Deng | 50 |

| | | |
|----------|---|-----------|
| 7 | Experimental Methodology | 53 |
| 7.1 | Data Collection and Data Preparation | 54 |
| 7.1.1 | Financial Indexes and Instability Trackers | 54 |
| 7.1.2 | COVID-19 | 58 |
| 7.1.3 | Deng | 59 |
| 7.2 | Training and Test Data separation | 61 |
| 7.3 | Imputation | 61 |
| 7.4 | XGBoost | 62 |
| 8 | Results | 64 |
| 8.1 | Financial Indexes and Instability Trackers Resampled by the Week's Mean | 64 |
| 8.2 | Financial Indexes and Instability Trackers Resampled by Fridays | 66 |
| 8.3 | COVID-19 | 68 |
| 8.4 | Deng | 70 |
| 8.5 | Ranking | 72 |
| 9 | Conclusion | 74 |
| | References | 76 |

Chapter 1

Introduction

This chapter gives the introduction to this work and in the next sections the motivation, objective, and contributions of the work are presented. Finally, the structure of the dissertation is described.

1.1 Motivation

With the advent of the Internet and of the use of more and more sensors on tools, machines, and vehicles, the amount of data produced in the world has reached significant numbers. According to a study conducted in [DOMO, 2017], in 2017 the Internet alone produced 2.5 millions of bytes per day. Along with the increase in the quantity of data generated, the number of applications that utilize them has also grown. Machine learning models have become a part of daily life for both people and the industry, allowing the automation of numerous tasks as well as process optimization. The more verisimilar the data, the more representative is the machine learning model and the better it performs. However, missing data is a common problem in data acquisition. Given that data can rarely be perfectly collected and many problems such as sensor and transmission failure, human error, and even differences in the rate of the collection might occur, learning to impute data is an important step in data analysis [Hang et al., 2011]. In cases in which it is necessary to join sequential data from different sources, the obtained data set frequently becomes full of missing data [Kim and Chi, 2018].

When missingness occurs in time series data, the problem can comprehend univariate or multivariate time series, different missingness mechanisms, and different time series pattern, such as trended, seasonal, white noise and combination trended and seasonal data. A good understanding of these factors can be helpful when choosing the appropriate imputation method. Choosing methods that take into account the temporal information intrinsic of time series data is also important [Luo et al., 2018].

Much research has already been done on the subject of time series imputation,

treating specific data sets and comparing the effectiveness of imputation methods. Imputation methods, using information intrinsic to the data set, impute the missing values. Research using traffic data has been reported [Wu et al., 2019] to mitigate the challenges that Intelligent Transportation Systems encounter due to missingness that affects the ability to predict traffic. Another study used meteorological data to evaluate different (conventional and modern) imputation methods in time series [Yozgatligil et al., 2013]. A study [Wijesekara and Liyanage, 2020] tackling data imputation in air quality data was conducted testing different imputation methods to mitigate bias and inefficiency in modeling. More recent works [Saad et al., 2020b] [Saad et al., 2020a] focus on comparing deep learning and/or machine learning imputation methods in time series.

1.2 Objective

The goal of this work is to study and evaluate solutions for the missing data problem in time series. As such, this work presents a literature review of the proper imputation methods to impute missing data in time series, which is the result of the study and research done regarding the imputation methods and their usage. As a result of the study, two new imputation methods (Imputation by Pattern and Imputation by Decomposition) are proposed, using the knowledge acquired during research. To evaluate the researched imputation methods, their impact is assessed on the results of three case studies.

1.3 Contributions

This dissertation presents two new imputation methods. The first one is called Imputation by Decomposition. It consists of decomposing the time series and imputing its parts using the appropriate imputation method according to the literature, then reassembles the time series. The second method presented is called Imputation by Pattern. It consists of separating each time series by its pattern (seasonal, trended, white noise or, combination of seasonal and trended) and then imputing each group using the method recommended in the literature. A variant of both methods using the Kalman filter [Bishop and Welch, 2001] after the imputation is also tested.

In a very instructive and procedural way, this dissertation presents the recommended steps to be taken when treating missingness in time series data, alongside notebooks to illustrate these steps¹ and help apply them to other problems involving time series. The usage of the library² created containing the new imputation methods is also illustrated by notebooks and the library itself is available.

¹ <https://github.com/silvanaribeiro/dissertacao>

² <https://github.com/silvanaribeiro/imputationLibrary>

The imputation methods were applied to real problems involving time series and compared to other imputation methods from literature. It was verified if the newly proposed imputation methods have good performance when applied to three real-world problems, having distinct characteristics and levels of difficulty. The results achieved by the Imputation by Pattern combined with the Kalman filter method perform well, obtaining accurate results for all data sets, and, although not always achieving the best result, it works fairly well for all of them. Both of the proposed methods proved to be effective for the tested data sets, showing similar performances to the classical imputation methods; it is also noted that the Imputation by Decomposition should be tested with its possible combinations to determine which one works better for each particular data set.

1.4 Structure

This chapter presented the motivation behind the study conducted and its objective. The remainder of this document is organized as follows: Chapter 2 presents the theoretical background: the missing data problem formulation, information regarding the pattern of time series and loss mechanisms. Chapter 3 presents the literature review: general purpose, classical and state-of-the-art imputation methods are presented. Chapter 4 presents the methodology and results of the experiment conducted with synthetic data. Chapter 5 presents in detail the two new imputation methods. In Chapter 6 the data used in the case studies and the case studies themselves are explained. In section 7, the experimental methodology used to compare the selected methods for the case studies is presented, alongside details of implementation. Section 8 brings the the results found and Section 9 presents the conclusions.

Chapter 2

Theoretical Background

In the following sections the missing data problem formulation is presented, along with concepts regarding time series pattern and missingness mechanisms.

2.1 Problem Formulation

Let X be a data set of dimensions n rows (samples) \times d columns (features) and M be the mask matrix, with the same dimensions $n \times d$, that only takes values in $\{0, 1\}^{d \times n}$. The mask matrix represents which values are missing from the X matrix by placing the value 1 at the position of the missing values. Each value of the imputed matrix \tilde{X} for all i and j can be defined as follows:

$$\tilde{X}_{i,j} = \begin{cases} X_{i,j}, & \text{if } M_{i,j} = 0 \\ *, & \text{if } M_{i,j} = 1. \end{cases} \quad (2.1)$$

The goal is to create a new data set \tilde{X} in which all positions marked as one on the mask M are imputed by some imputation method and the values from all other positions come from the corresponding position in X .

Hereafter, a row of a data set will be referred to as a sample and a column of a data set will be referred to as a feature. In addition, data that cannot be directly measured [Curado et al., 2014] will be referred to as unobserved data, whereas data that can be directly measured will be referred to as observed data. As an example, when dealing with hospital patients, observed data would be the patient's temperature, weight, and height, whereas unobserved data would be pain level, health, and well-being.

2.2 Data Pattern

When dealing with missing data in time series it is important to identify the missingness mechanism of the data to choose the most appropriate method for imputation [Kim and Chi, 2018]. Further, it might be helpful to identify the characteristics of the series. The following concepts regarding the characteristics of time series data were extracted mainly from the book “Forecasting: principles and practice” [?].

A time series might be trended, meaning it presents a long decrease or increase with time, linear or not, and it can even change from increasing to decreasing. An example of trended data is presented in Figure 1.

A time series might be seasonal, meaning it is affected by seasonal factors of a fixed and known frequency. Seasonal data should not be confused with cyclic data, which is defined by data that rises and falls in intervals of unfixed length. An example of seasonal data is presented in Figure 2.

A white noise time series has none, or close to none autocorrelation [?]. An example of white noise data is presented in Figure 3.

A time series might also be a combination of seasonal and trended data. An example of seasonal and trended data is presented in Figure 4.

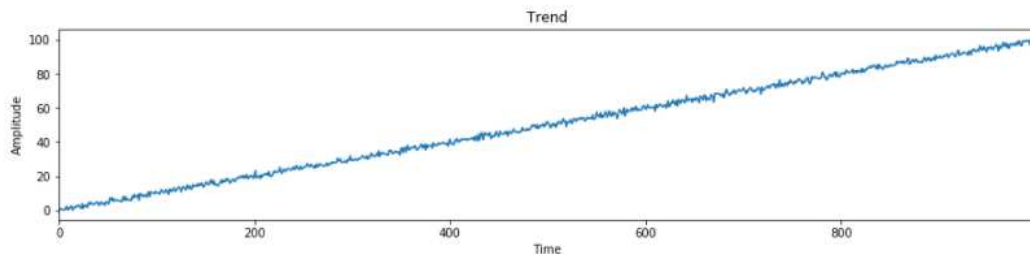


Figure 1 – Time series with trend and without seasonality

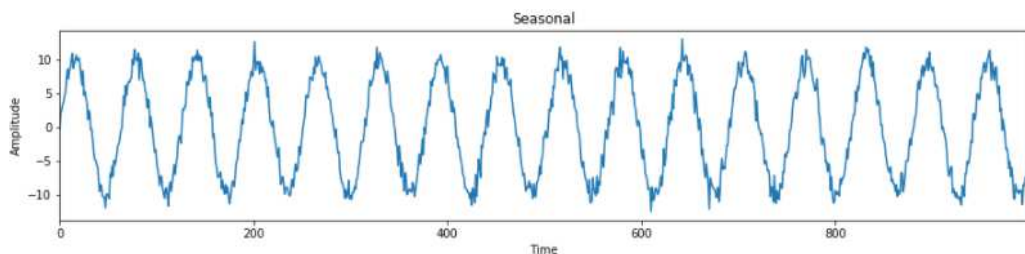


Figure 2 – Time series without trend and with seasonality

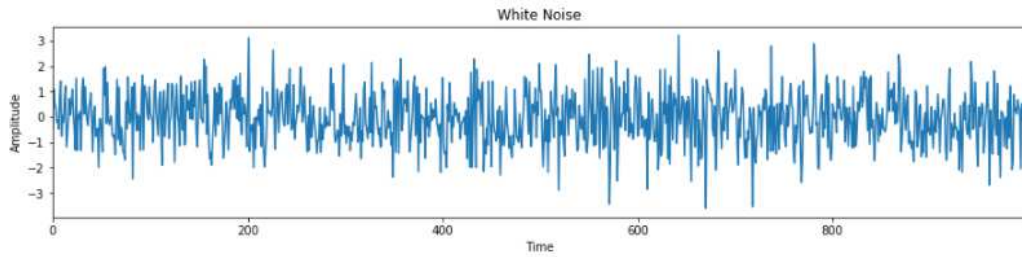


Figure 3 – White Noise

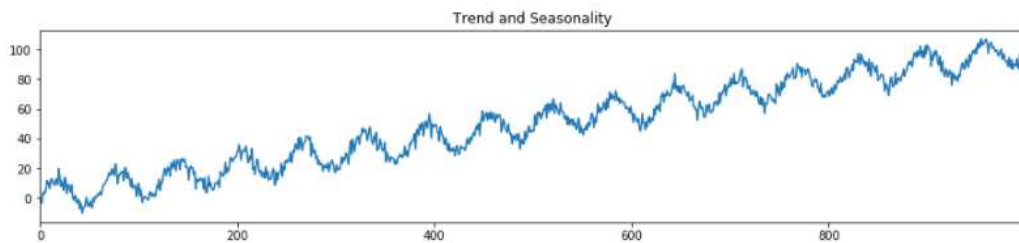


Figure 4 – Time series with trend and with seasonality

An intuitive and visual way to identify trend and seasonality in time series is to calculate and plot its autocorrelation function, which measures the linear relationship between lagged values of a time series.

The autocorrelation plot for a trended time series slowly decreases as the lag increases, as can be seen in Figure 5, which is the autocorrelation function of the time series shown in Figure 1.

The autocorrelation plot for a seasonal time series presents larger values for the multiples of the seasonal frequency, as can be seen in Figure 6, which is the autocorrelation function of the time series shown in Figure 2.

The autocorrelation plot for a seasonal and trended time series presents a combination of the aforementioned effects, as can be seen in Figure 7, which is the autocorrelation function of the time series shown in Figure 4.

The autocorrelation plot for a white noise time series is expected to have 95% of the spikes to lie within an interval of $\pm 1.96/\sqrt{T}$, where T is the length of the time series, as can be seen in Figure 8, which is the autocorrelation function of the time series shown in Figure 3.

Besides visual analysis, statistical tests usually applied to verify the characteristics of the time series are:

- **Cox-Stuart Test:** Recommended to verify the existence of increasing or decreasing trend in time series data, this test is based on the hypothesis that there will not exist

any trend if the probability of one sample being smaller than its successor is the same as the probability of this same sample being greater than its successor for the majority of the samples in the data. Otherwise, the data is trended [Amaral, 2014].

- Autocorrelation Testing of a certain order: Recommended to verify the existence of seasonality in time series data, this test verifies if there is a correlation between the signal and itself for a certain order. For example, if there exists yearly seasonality in a series, the correlation of order twelve will be meaningful [?].
- Autocorrelation Testing: Recommended to verify if a time series is white noise, this test verifies if there is a correlation between the signal and itself only at time zero. In other words, for a signal to be white noise, the autocorrelation of the signal must be different from zero only for order zero [Aguirre, 2004].

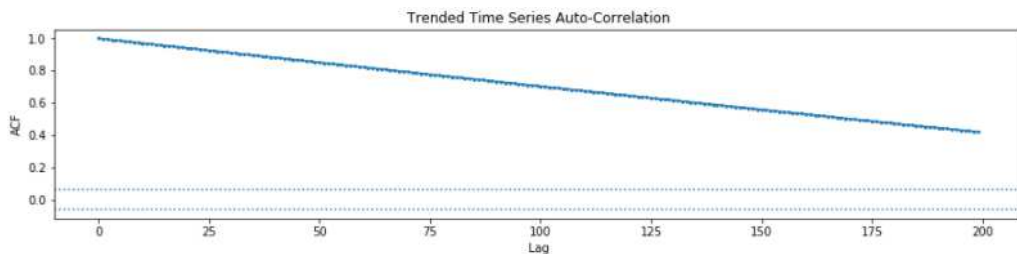


Figure 5 – Autocorrelation of time series with trend and without seasonality

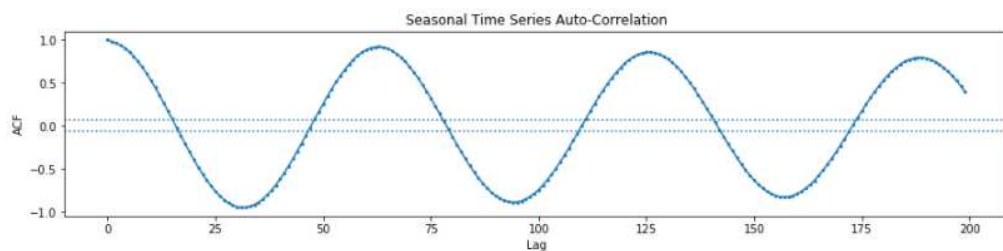


Figure 6 – Autocorrelation of time series without trend and with seasonality

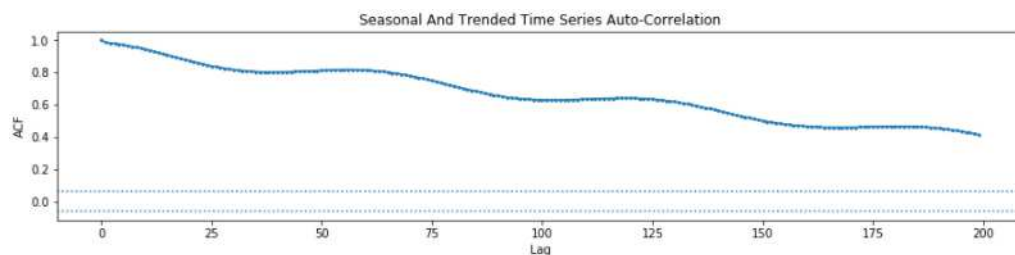


Figure 7 – Autocorrelation of time series with trend and with seasonality

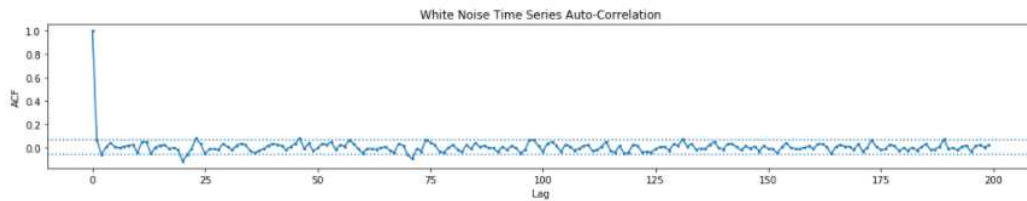


Figure 8 – Autocorrelation of white noise

2.3 Missingness Mechanisms

Concerning the missingness mechanisms, the data may suffer from a structural deficiency or the loss may occur randomly.

The structural deficiency can be defined as a component of a feature that was omitted from the data [Kuhn and Johnson, 2019]. As an example, in a data set about real state, the lack of information about the number of cars that fit in the garage might mean that, in fact, the building does not have a garage.

In case the loss is random, there are three categories of missingness mechanisms [Kuhn and Johnson, 2019]:

- MCAR - Missing Completely At Random: The probability of the data being lost is the same for all data. This probability depends neither on the observed nor the non-observed data. This means that there is no logic behind the loss. The data is lost by a random process, e.g, as a result of a sensor failure.
- MAR - Missing At Random: The probability of the data being lost is not the same for all data. This probability depends on the observed data, but not on the non-observed. That means that the absence of the data can be predicted by the observed data. For example, consider a data set about income and education level, people with low education level tend to not inform their incomes. That implies that the absence of the data about income can be predicted by the information about the education level.
- NMAR - Not Missing At Random: The missing data is related to the non-observed data. In other words, the missingness is related to factors not taken into account, for instance, in a data set about income, both the lower and higher incomes are not disclosed by the respondents. It is not possible to define which of the extremities the missing data belongs to nor predict if the data will be informed or not.

Chapter 3

Literature Review

Applications that simply delete the samples or features that have missing values or straightforwardly ignore the missingness, can produce biased results and incorrect conclusions about the studies being conducted [Wijesekara and Liyanage, 2020].

The deletion of features and samples might be a viable option if the missingness mechanism is MCAR and the loss rate is low [Kim and Chi, 2018]. If this approach cannot be used, it is recommended that the data be imputed.

Performing analyses ignoring the missingness of the data is a risky approach, especially if the loss rate is not low, and if the missingness mechanism is not MCAR [Pratama et al., 2016]. In certain cases, it is not even an option to do so, depending on the application and algorithms being used on the analysis.

Figure 9 brings a flowchart pointing out the more appropriate imputation methods depending on the characteristics of the data and missingness mechanisms associated.

3.1 General Purpose Imputation Methods

Some classical imputation methods work for both time series and other types of data. Some examples are mean imputation, median imputation, mode imputation, and random sample imputation [Pratama et al., 2016].

In the next sections, general purpose, classical, and state-of-the-art imputation methods are described.

3.1.1 Mean imputation, median imputation, and mode imputation

These methods consist of computing the mean, median, and mode, respectively, of the features where the loss occurs and imputing those computed values whenever the

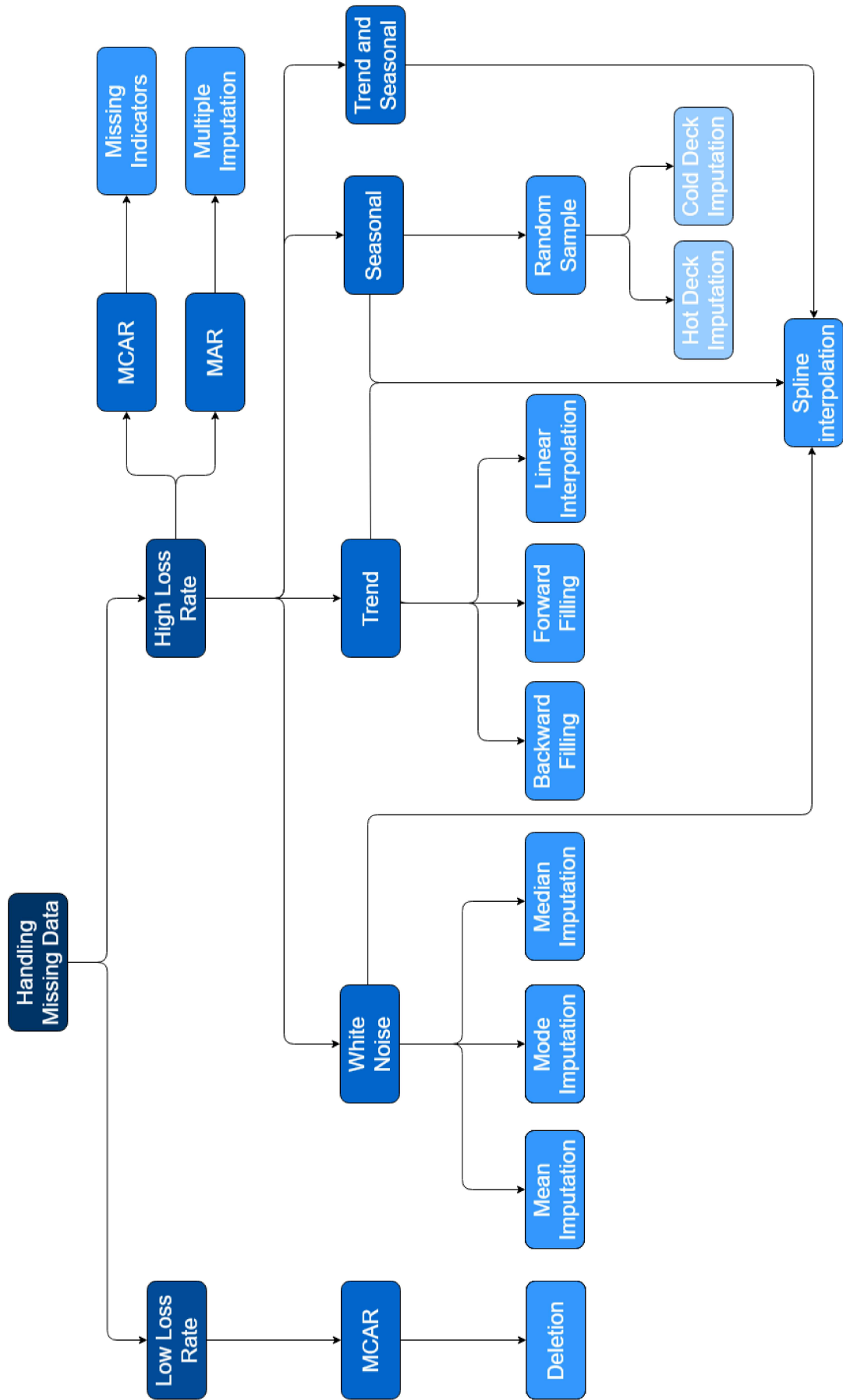


Figure 9 – Flowchart of methods to deal with missing data

missingness occurs. The mean imputation method is recommended for normally distributed data, whereas mode and mean imputation were invented to account for not normally distributed data. To reduce the influence of outliers, the median imputation method is recommended. These methods are appropriate when dealing with stationary time series, e.g, white noise. Those methods decrease the variance of the data and therefore affect the standard deviation, which can cause bias [Pratama et al., 2016].

3.1.2 Random Sample Imputation

This approach uses randomly selected values from the feature in question to do the imputation. The value can be randomly selected from the whole set of possible values or it can be selected randomly from a subset of it. This approach can be appropriate in case the subset is carefully chosen and for series with seasonality and without trend. It is important to mention two methods that are part of the random sample imputation method: Hot Deck Imputation and Cold Deck Imputation [Kalton and Kish, 1984].

- Hot Deck Imputation: Finds samples that have similar values on the other features as the sample with a missing value and, among the selected samples, chooses one randomly to impute. This approach restricts the possible values to be imputed to values that have already occurred on the data set and increases variability, resulting in more accurate standard deviations [Andridge and Little, 2010].
- Cold Deck Imputation: This method is similar to Hot Deck Imputation, however, instead of choosing randomly between similar samples within the same data set, it replaces the missing values using a different data set [Duma et al., 2013].

3.2 Classical Imputation Methods for Time Series

Concerning well-known imputation methods specific for time series data, the following methods should be mentioned:

3.2.1 Forward and Backward Filling

Forward Filling, also known as Last observation (or value) carried forward, imputes the missing value with the last seen observation. Backward Filling, also known as Next Observation (or Value) Carried Backward, imputes the missing value with the next seen observation. These methods assume that adjacent data points are similar. However, this is especially false if there is seasonality on the data [Molnar et al., 2008]. It is important to notice that for problems using streaming data the backward filling method cannot be used, since the next observation is not available immediately to impute the current missing value.

3.2.2 Multiple Imputation

This method imputes a missing value n -times ($n > 1$). They are imputed based on the observed values for a given sample and the relations observed in the data for other samples. The values represent a probability distribution to reproduce the uncertainty of the value being imputed [Azur et al., 2011]. Since this method has a random element within itself, these values should be similar, nonetheless different. The data sets with the imputed values are then evaluated and the results are combined so that more realistic conclusions and estimates can be obtained. This method works especially well for imputing survey data, which in general have a MAR missingness mechanism [Little and Rubin, 2019]. If made correctly, multiple imputation generates unbiased estimates of the parameters and accurate standard deviations.

3.2.3 Linear Interpolation

This method assumes that an estimated point will be on the vector that connects the nearest points on the right and left. In other words, it uses a linear function to approximate the function that represents the data well and to compute the missing value [Rantou, 2017]. Linear Interpolation also assumes that adjacent data points are similar and, thus, have the same problems as Forward and Backward Filling.

3.2.4 Spline Interpolation

This method uses polynomial functions to approximate a function that represents the data well and to calculate the missing value [Rantou, 2017]. It solves, to a certain extent, the problem of assuming that adjacent data points are similar since it can represent more abrupt variations of the data, although it depends on the polynomials chosen to approximate the function.

3.2.5 Missing Indicators

This method imputes a default value chosen by the user of the method in every occurrence of missing value and creates a new feature that indicates missingness with the flag 1 on the position where it occurs and 0 where it does not. The results will be biased if the missingness mechanism is not MCAR and if the original features themselves are correlated [Groenwold et al., 2012].

3.2.6 Expectation Maximisation

This method is an iterative procedure that uses the other features to impute a value and then checks if the imputed value is the most probable. In case it is not the most

probable, re-computes a new value. This behavior is repeated until the most probable value is found. Although it preserves the relation between features, it underestimates the standard deviation [Dempster et al., 1977].

3.3 State-of-the-art imputation Methods for Time Series

A few of the most recent methods that should be mentioned when dealing with imputation in time series data are:

3.3.1 ST-2SMR

This method is composed of two steps to reconstruct missingness in space-temporal data. The first step is a coarse-grained interpolation to eliminate the influence of continuous missing data on the general result. Then, based on the result obtained in the first step, a dynamic selection sliding window algorithm is used to identify the most relevant data to make a fine-grained interpolation. Finally, the results are integrated using a neural network model [Cheng and Lu, 2017].

3.3.2 TBM - Temporal Belief Memory

This method deals with missingness using recurrent neural networks [Kim and Chi, 2018]. It is an imputation method that, unlike conventional neural networks, does not ignore the real interval between consecutive samples, taking into account time continuity and identifying the lack of data. It computes the belief of the last observed value in time for each feature and imputes the data based on the individual belief both towards the future and the past [Kim and Chi, 2018].

3.3.3 GAIN - Generative Adversarial Imputation Network

This method is an adaptation of Generative Adversarial Networks. It has a generator that observes components of the real data and generates an imputed vector. Then, a discriminator takes the imputed vector and tries to determine which values are real and which ones were generated. The discriminator is given hints of the real distribution of the data to ensure it forces the generator to improve the imputation. Once the generator succeeds in deceiving the discriminator, the process is terminated [Yoon et al., 2018].

3.3.4 Long Short-Term Memory - LSTM

Most of the concepts used hereafter to explain LSTM were extracted from [Siami-Namini and Namin, 2018], [Greff et al., 2017], and [Abraham, 2005]. Long Short-Term Memory is a type of Recurrent Neural Network that has the capability to remember values from earlier stages with the purpose of improving the prediction of future values.

To better understand LSTM it is important to have some knowledge of Artificial Neural Networks and Recurrent Neural Networks. Artificial Neural Networks are generalizations of mathematical models of biological nervous systems. Neurons are basic processing elements of neural networks. They represent the effects of synapses by weighing the connections that modulate the input signals. The nonlinear characteristic of the neuron is represented by a transfer (or activation) function. The neuron impulse is therefore computed as the weighted sum of the input signals transformed by the transfer function. A learning algorithm is then used to adjust the weights, granting learning capability to the neuron.

The neuron output signal O is given by (3.1)

$$O = f \left(\sum_{j=1}^n w_j x_j \right) \quad (3.1)$$

where w_j is the weight vector and $f(\cdot)$ is the activation function.

Neural Networks are formed by a combination of artificial neurons having nonlinear transfer functions. They might have different architectures but, the basic one consists of three neuron layers: input, hidden, and output layers. Feed-forward networks have signal flow from the input layer to the output layer, whereas Recurrent Neural Networks have feedback connections. A Recurrent Neural Network uses sequential observations and learns from the earlier stages to predict future values. The hidden layers store information captured in the earlier stages, but they can only remember a few steps of the sequence. As a result, RNNs are not suitable to predict longer sequences of data.

To solve the problem aforementioned, Long Short-Term Memory (LSTM) recurrent networks were created. Using gates, LSTMs are able to choose if data in each cell will be disposed of, filtered, or added for the next cell. The gates are based on the sigmoidal neural network layer.

There are three types of gates controlling each cell, as shown in Figure 10: The Forget gate, which outputs a number between 0 and 1, indicating if the data should be completely forgotten or completely kept at the extremes, or how much of it should be kept in-between. The Memory gate, that chooses which new data need to be stored in the cell. A sigmoid layer (input door layer) chooses which new values will be modified. A hyperbolic tangent layer produces a vector of new values that are candidates to being

added to the state. Finally, the Output gate, that decides what will be the output of each cell based on the cell's state, filtered, and newly added data.

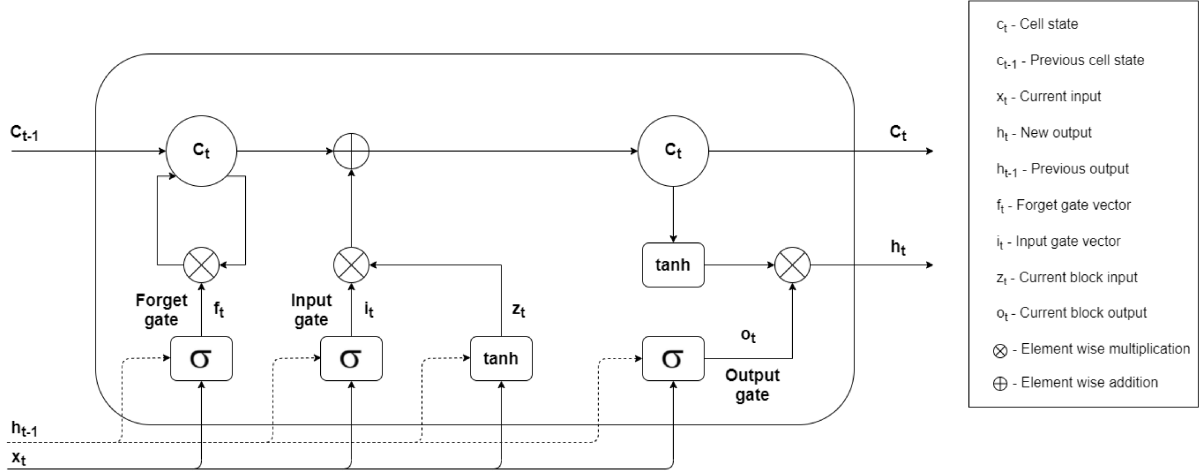


Figure 10 – LSTM cell

Considering the forward pass, let x^t be the input vector at time t , N be the number of LSTM blocks, M the number of inputs, W_z, W_s, W_f, W_o the input weights, R_z, R_s, R_f, R_o the recurrent weights, p_s, p_f, p_o the peephole weights, and b_z, b_s, b_o the bias weights. The formulas for the LSTM layer forward pass can be written as

$$\bar{z}_t = W_z x_t + R_z y_{t-1} + b_z \quad (3.2)$$

$$z_t = \tanh(\bar{z}_t) \quad (3.3)$$

$$\bar{i}_t = W_i x_t + R_i y_{t-1} + p_i \odot c_{t-1} + b_i \quad (3.4)$$

$$i_t = \sigma(\bar{i}_t) \quad (3.5)$$

$$\bar{f}_t = W_f x_t + R_f y_{t-1} + p_f \odot c_{t-1} + b_f \quad (3.6)$$

$$f_t = \sigma(\bar{f}_t) \quad (3.7)$$

$$c_t = z_t \odot i_t + c_{t-1} \odot f_t \quad (3.8)$$

$$\bar{o}_t = W_o x_t + R_o y_{t-1} + p_o \odot c_t + b_o \quad (3.9)$$

$$o_t = \sigma(\bar{o}_t) \quad (3.10)$$

$$y_t = h(c_t) \odot o_t \quad (3.11)$$

where σ is the logistic sigmoid activation function $\sigma(x) = (1/1 + e^{-x})$. The pointwise multiplication of two vectors is denoted by \odot .

Regarding the backpropagation through time, the deltas inside the LSTM block are calculated as

$$\delta y_t = \Delta_t + R_z^T \delta z_{t+1} + R_i^T \delta i_{t+1} + R_v^T \delta f_{t+1} + R_o^T \delta o_{t+1} \quad (3.12)$$

$$\delta o_t = \delta y_t \odot h(c_t) \odot \sigma'(\bar{o}_t) \quad (3.13)$$

$$\delta c_t = \delta y_t \odot o_t \odot h'(c_t) + p_o \odot \delta o_t + p_i \odot \delta i_{t+1} + p_f \odot \delta f_{t+1} + \delta c_{t+1} \odot \delta f_{t+1} \quad (3.14)$$

$$\delta f_t = \delta c_t \odot c_{t-1} \odot o'(\bar{f}_t) \quad (3.15)$$

$$\delta i_t = \delta c_t \odot z_t \odot o'(\bar{i}_t) \quad (3.16)$$

$$\delta z_t = \delta c_t \odot i_t \odot o'(\bar{z}_t) \quad (3.17)$$

$$\delta x_t = W_z^T \delta z_t + W_i^T \delta i_t + W_f^T \delta f_t + W_o^T \delta o_t \quad (3.18)$$

where Δ_t is the vector of the deltas passed down from the layer above.

The gradients for the weights are calculated as follows, where $*$ can be any of z, i, f, o and $\langle \cdot, \cdot \rangle$ denotes the outer product of two vectors:

$$\delta W_* = \sum_{t=0}^T \langle \delta_{*t}, x_t \rangle \quad (3.19)$$

$$\delta p_i = \sum_{t=0}^{T-1} c_t \odot \delta i_{t+1} \quad (3.20)$$

$$\delta R_* = \sum_{t=0}^{T-1} \langle \delta *_{t+1}, y_t \rangle \quad (3.21)$$

$$\delta p_f = \sum_{t=0}^{T-1} c_t \odot \delta f_{t+1} \quad (3.22)$$

$$\delta b_* = \sum_{t=0}^T \delta *_{t+1} \quad (3.23)$$

$$\delta p_o = \sum_{t=0}^{T-1} c_t \odot \delta o_{t+1} \quad (3.24)$$

To use LSTM as an imputation method there are three possible courses of action:

- Remove samples that contain missing time steps. In other words, the rows that contain missing values are deleted from the data set
- Mark the missing time steps and force the network to learn their meaning. The missing values are imputed with a default value chosen by the user and then the model is informed of the default value to learn from the missingness
- Mark the missing time steps and exclude them from calculations in the model. The missing values are imputed with a default value chosen by the user and then the model is informed that it should ignore those values when calculating the model.

Afterward, the model can be generated and the values predicted for the missing time steps can be used to impute them.

3.3.5 Kalman Filter

The concepts used hereafter to explain the Kalman Filter were extracted from [Bishop and Welch, 2001] and [Aguirre, 2004]. The Kalman filter provides an efficient computational solution of the least-squares method. It supports the estimation of states even when the pattern of the modeled system is unknown. It is an optimal recursive estimator. The filter assumes that the system under study can be described by a linear model. The discrete Kalman Filter tries to estimate the state $x \in \mathbb{R}^s$ of the linear stochastic difference shown by:

$$x_{k+1} = A_k x_k + B u_k + w_k, \quad (3.25)$$

with a measurement $y \in \mathbb{R}^t$ shown by (3.26)

$$y_k = H_k x_k + v_k, \quad (3.26)$$

where w_k represents the process noise and v_k represents the measurement noise. They are assumed to be mutually independent, white and with normal probability distributions as shown by :

$$p(w) \sim N(0, Q) \quad (3.27)$$

$$p(v) \sim N(0, R). \quad (3.28)$$

The matrix $A^{s \times s}$ in (3.25) is the dynamic matrix relating the state of time steps k and $k + 1$. The matrix $B^{s \times l}$ relates the control input $u \in \mathbb{R}^l$ to the state x . The matrix $H^{t \times s}$ in (3.26) relates the state to the measurement y_k .

The Kalman filter predicts and then corrects the prediction once a measurement is available. Thus, the Kalman filter has two groups of equations: time update (predictor) equations;

$$\hat{x}_{k+1}^- = A_k \hat{x}_k + B u_k, \quad (3.29)$$

$$\hat{P}_{k+1}^- = A_k P_k A_k^T + Q_k, \quad (3.30)$$

$$K_k = \hat{P}_k^- H_k^T (H_k \hat{P}_k^- H_k^T + R_k)^{-1}, \quad (3.31)$$

and measurement update (corrector) equations;

$$K_{k+1} = P_{k+1}^- H_{k+1}^T (H_{k+1} P_{k+1}^- H_{k+1}^T + R_{k+1})^{-1}, \quad (3.32)$$

$$\hat{x}_{k+1} = \hat{x}_{k+1}^- + K(z_{k+1} - H_{k+1} \hat{x}_{k+1}^-), \quad (3.33)$$

$$P_{k+1} = (I - K_{k+1} H_k) P_{k+1}^-. \quad (3.34)$$

The former group is responsible for propagating forward (in time) the *a priori* statement, obtaining a prediction for the next step, whereas the latter is responsible for the correction, incorporating the new measurement into the estimate to improve it.

Equation (3.32) computes the Kalman gain K_k , (3.33) updates the estimate with measurement z_k , and (3.34) updates the error covariance. Regarding the time update equations, (3.29) projects the state ahead and (3.30) projects the error covariance ahead. Initial estimates for \hat{x}_0^- and P_0^- need to be provided.

To use the Kalman filter as an imputation method the time update step is done normally, but the measurement update step is skipped whenever there is a missing time step. For those cases, the value predicted by the time update step is used to impute the missing time step.

Chapter 4

Experiment with Synthetic Data

To illustrate the effectiveness of some of the methods and verify the recommendations given by the literature, an experiment was conducted using univariate times series such as the ones of Figures 1, 2, 3, and 4.

In the next sections, the methodology and results of the experiment using synthetic data are presented.

4.1 Methodology

A flowchart representing the steps of the experiment is presented in Figure 11.

The first step of the experiment was to generate four synthetic univariate time series of each pattern: one trended, one seasonal, one white-noise, and one trended and seasonal time series.

The second step was the random removal of 20% of the samples of each one of the four time series to emulated missingness, making the missingness mechanism MCAR. Then, using the flowchart shown in Figure 9, one appropriate method was chosen for each type of data. For the seasonal data, the Random Sample method was chosen. For the trended data, the Forward Filling method was chosen. For the trended and seasonal data, the Spline Interpolation method was chosen. For the white-noise data, the Mean Imputation method was chosen. As the time series being imputed is univariate, the Random Sample imputation method was implemented choosing a value randomly from a subset chosen from the own time series. A window of size corresponding to 6% of the length of the time series is used to choose the subset. This window is centered on the position of the missing value, which means that the randomly selected value can come from samples preceding or proceeding the missing value.

Each one of the time series was imputed using all four methods previously chosen, accounting for the third step of the experiment. For reproducibility purposes, the library

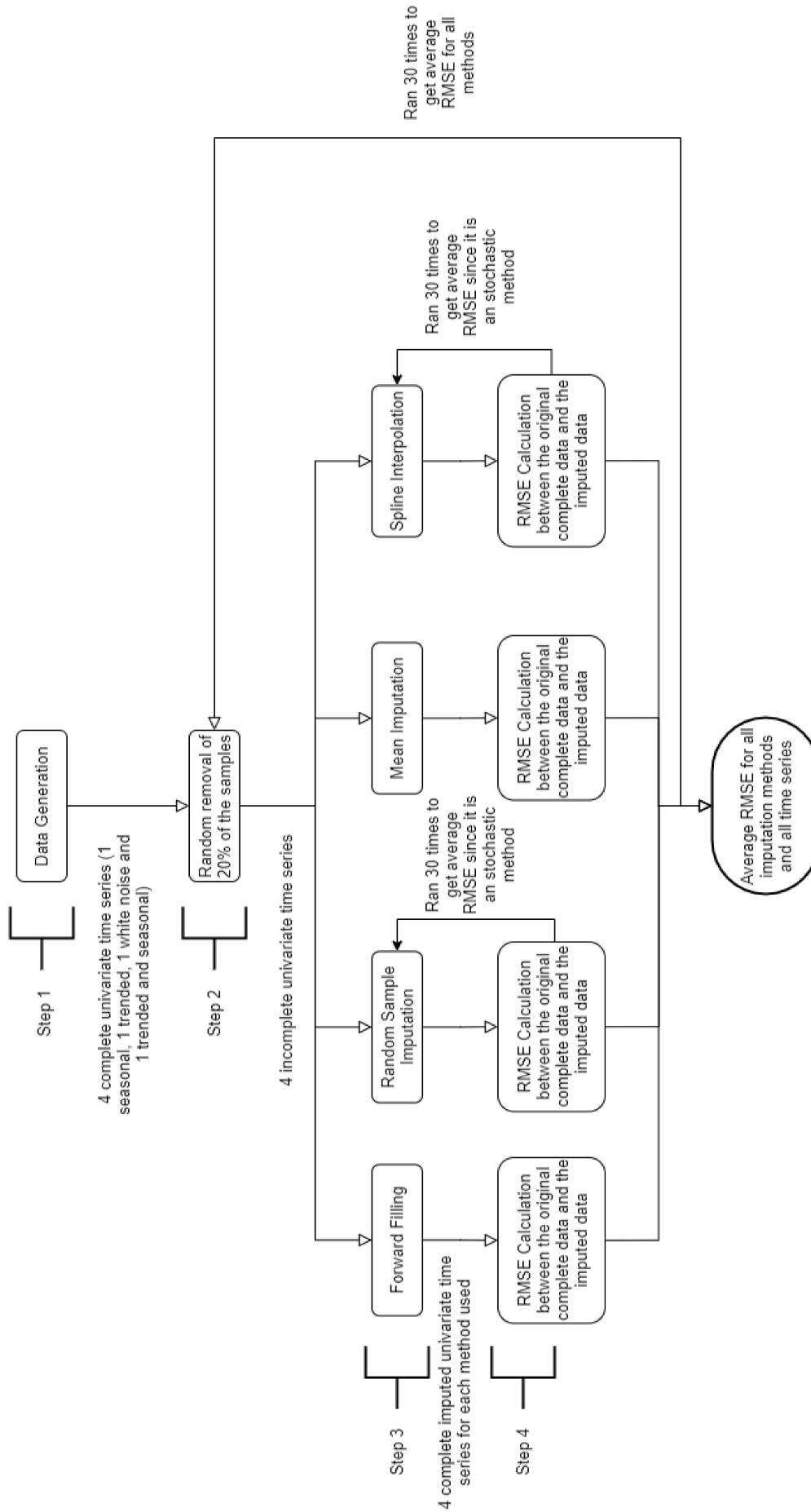


Figure 11 – Flowchart of the Synthetic Data Experiment

created with implementations of the imputation methods aforementioned is available¹ and it is called ImputationLibrary. The fourth step consisted of calculating the Root Mean Square Error between the imputed data and the real data.

Considering that Spline Interpolation and Random Sample are stochastic methods, they were run 30 times to compute the average RMSE.

The whole process of removing random samples and imputing them using the chosen imputation methods (steps 2 through 4) was repeated 30 times to calculate the average RMSE. That means that the stochastic methods were actually run 900 times.

For reproducibility purposes, the notebooks in which the experiment was conducted are available² and correspond to the notebooks starting with the prefix 00.

4.2 Results

The results obtained are shown in Table 1.

| Data \ Method | Forward Filling | Random Sample | Mean Imputation | Spline Interpolation |
|------------------|--|--------------------|-----------------------------------|-------------------------|
| Trend & Seasonal | $3.675 \pm 8.882e^{-16}$ | 23.616 ± 2.077 | $171.020 \pm 8.527e^{-14}$ | $8.421 \pm 1.776e^{-1}$ |
| Seasonal | 0.578 ± 0.0 | 20.943 ± 1.636 | $11.289 \pm 3.553e^{-15}$ | $0.624 \pm 2.220e^{-1}$ |
| Trend | $0.414 \pm 1.110e^{-16}$ | 19.771 ± 1.497 | $9.717 \pm 3.553e^{-15}$ | 0.432 ± 0.0 |
| White Noise | $0.481 \pm 5.551e^{-17}$ | 0.403 ± 0.035 | 0.213 ± 0.0 | 0.542 ± 0.0 |

Table 1 – Average RMSE of the imputed data sets for each method

Observing Table 1 it can be seen that the best RMSE obtained for the time series with trend and seasonality was achieved by the Forward filling method, followed by the Spline Interpolation method, which is the one recommended by literature. For the seasonal data, the best result was not achieved by the literature recommended random sample method, it was also achieved by the forward filling method. However, this result was followed closely by the result achieved by the Spline Interpolation method, which is recommended for all cases. For the data with trend, the best result achieved was by the forward filling method, which is the one recommended by the literature. Finally, for white noise data, the best result achieved was by the mean imputation method, which is also the one recommended by literature.

It is important to notice that, as expected, Spline interpolation did a fairly good job for all types of time series. Even for white noise data, where it performed the worst, it is still close to the other methods. This result is expected since the spline interpolation method is recommended by the literature to be used with time series data of all pattern.

¹ <https://github.com/silvanaribeiro/imputationLibrary>

² <https://github.com/silvanaribeiro/MissingDataInTimeSeries>

The results for the random sample imputation method show how difficult it is to choose an appropriate subset from which to randomly select from.

Generally speaking, using the literature recommended methods is the best choice to be made when deciding how to impute missing data. That being said, it is important to note that this experiment was conducted using synthetic data. This makes the study controlled and in the case where real data is used, the behaviour should be similar but can vary due to several factors, such as rate of missingness, amplitude of the noise, and etc.

Chapter 5

Novel Imputation Methods

This chapter presents in detail the two new imputation methods proposed by this work. The section presenting the imputation by Time Series Pattern also includes the results of the tests conducted to verify the assertiveness of the separation heuristics. While the Imputation by decomposition tries to decompose each time series into its three components (seasonal, trend and noise components), it may find trend and seasonal components where there are none. As a consequence, this method cannot be used to classify time series as being trended and/or seasonal. For that purpose, the Imputation by Pattern method was created.

5.1 Imputation by Decomposition

Imputation by Decomposition is one of the two imputation methods proposed in this work. The idea is that the time series is decomposed into the seasonal, trend, and remainder (assumed to be white-noise) components, and each component is then imputed by the literature recommended imputation methods. Seasonal additive or multiplicative decomposition using moving averages is used to decompose each time series in its components. The following concepts regarding decomposition were extracted mainly from the book “Forecasting: Principles and Practice” [Hyndman and Athanasopoulos, 2018]. The additive decomposition of a time series y_t is given by:

$$y_t = S_t + T_t + R_t, \quad (5.1)$$

where y_t is the data, S_t is the seasonal component, T_t is the trend-cycle component, R_t is the remainder component, and t is the period. The multiplicative decomposition of a time series y_t is given by:

$$y_t = S_t \times T_t \times R_t, \quad (5.2)$$

where y_t is the data, S_t is the seasonal component, T_t is the trend-cycle component, R_t is the remainder component, and t is the period.

To understand the seasonal decomposition method, it is important to first understand the moving average smoothing that is used to estimate the trend-cycle component of the time series. A moving average of order m is:

$$\hat{T}_t = \frac{1}{m} \sum_{j=-d}^d y_{t+j} \quad (5.3)$$

where $m = 2d + 1$, meaning the trend-cycle component at time t is obtained by averaging values of the time series within d periods of t . The idea is that values that are close in time are likely to be also close in value. The average eliminates some of the randomnesses in the data. This is called m-MA (Moving average of order m). A moving average of a moving average can be used to produce an even-order moving average symmetric and the most common use of a centered moving average is for estimating the trend-cycle from seasonal data. The combination of moving averages results in weighted moving averages, represented by

$$\hat{T}_t = \frac{1}{m} \sum_{j=-d}^d a_j y_{t+j}, \quad (5.4)$$

where $d = (m - 1)/2$ and the weights are $[a_{-d}, \dots, a_d]$.

Additive decomposition is done by computing the trend component by using $2 \times m$ -MA if m is even or m -MA if m is odd, then calculating the detrended series $y_t - \hat{T}_t$. Afterward, the seasonal component for each season is calculated by averaging the detrended values for that season. These seasonal component values are then adjusted to add to zero and strung together. Finally, the remainder component is calculated by subtracting the estimated seasonal and trend components from the time series. The multiplicative decomposition follows the same steps as the additive decomposition, but the subtractions are replaced by divisions. The python library statsmodels was used to implement the mentioned decompositions. Before the decomposition, the missing values are imputed using a very small valued placeholder ($0.1 \times e^{-5}$). To determine which type of decomposition is more suitable for the time-series data, firstly the multiplicative decomposition is attempted, since in theory it can decompose more complex time-series. However, if there are numeric problems, such as zero or negative values, and so on, the additive decomposition is done. The decomposition using solely additive decomposition is also done separately to test its performance.

Once the seasonal, trend, and remainder components are obtained, they are each imputed with literature recommended methods for each of their natures. The seasonal component is imputed using spline interpolation. The trend component is imputed by the

forward-filling imputation method. The remainder component is assumed to be white noise, and so it is imputed by its mean. The imputed components are then added or multiplied, depending on the type of decomposition done, to form a new imputed time series.

5.2 Imputation by Time Series Pattern

The second imputation method proposed in this work is the imputation by Time Series Pattern. It is more of a strategy than a method *per se*, and it works by firstly determining if the time series is seasonal, trended, white-noise, or a combination of seasonal and trended and then imputing it with the literature recommended method for its pattern.

To determine if the time series is close enough to white noise, firstly it is standardized by removing its mean and then scaling it to unit variance. Afterward, the autocorrelation function of the standardized time series is calculated. The autocorrelation plot for white noise time series is expected to have 95% of its spikes to lie within an interval of $\pm 1.96/\sqrt{T}$, where T is the length of the time series [Hyndman and Athanasopoulos, 2018]. The values of the autocorrelation are then tested within the above-mentioned limit, and in case 95% of them lie within the limit, the time series is considered to be close enough to white noise. The pseudocode for the white noise pattern classification is shown by Algorithm 1.

Algorithm 1 Pseudocode of the function `isWhiteNoise(ts)`

```

normalized ← standardize(ts)
corr ← correlate(normalized, normalized)
upper_limit ←  $\frac{1.96}{\sqrt{\text{length}(ts)}}$ 
lower_limit ←  $\frac{-1.96}{\sqrt{\text{length}(ts)}}$ 
COUNT ← 0
for i = 0 to i < length(corr) do
    if corr[i] ≤ upper_limit or corr[i] ≥ lower_limit then
        count = count + 1
    end if
end for
if count ≤ 0.05 × length(ts) × 2 then
    return Time series is white noise
else
    return Time series is NOT white noise
end if

```

Granted that the time series is not white noise, it is checked to be seasonal. The test starts by also standardizing the time series. Then, the autocorrelation is calculated and a moving average with a window of size two is applied to it to avoid the occurrence of small peaks. The peaks within the data are located using the function `find_peaks` from python's `scipy` library and the distance between them are checked to verify if all peaks lie

within the same distance (with a tolerance of ± 2 time steps. Then, if at least 95% of the peaks lie within the same distance, the time series is considered to be seasonal. This test is based on the fact that visually speaking, the autocorrelation plot for a seasonal time series presents larger values for the multiples of the seasonal frequency [Hyndman and Athanasopoulos, 2018]. The pseudocode for the seasonal test is shown by Algorithm 2.

Algorithm 2 Pseudocode of the function `isSeasonal(ts)`

```

if isWhiteNoise(ts) then
    return Time series is NOT Seasonal
end if
normalized  $\leftarrow$  standardize(ts)
corr  $\leftarrow$  correlate(normalized, normalized)
corr  $\leftarrow$  rollingWindowMean(corr, 2)
peaks  $\leftarrow$  findPeaks(corr)
count  $\leftarrow$  0
if length(peaks) < 2 then
    return Time series is NOT Seasonal
else if length(peaks)  $\geq$  6 then
    comparison  $\leftarrow$  peaks[5] - peaks[4]
else
    comparison  $\leftarrow$  peaks[2] - peaks[1]
end if
for i = 0 to i < length(peaks) do
    if peaks[i + 1] - peaks[i] > comparison + 2
    or peaks[i + 1] - peaks[i] < comparison - 2 then
        count = count + 1
    end if
if count > length(ts)  $\times$  0.05 then
    return Time series is NOT Seasonal
else
    return Time series is Seasonal
end if
end for

```

If the time series is neither white noise nor seasonal, it is tested to be trended. This test starts by making a seasonal additive decomposition to the time series. The mean of the trend component is then evaluated and if it is not close to zero (with some tolerance) it is determined to be trended. The pseudocode for the trended pattern classification is shown by Algorithm 3.

A time series that is both trend and seasonality will pass both the trend and seasonal tests, but not the white noise one. The pseudocode for the trended and seasonal pattern classification is shown by Algorithm 4.

The methods that classify the pattern of the time series are not exact, so the possibility exists that a time series is not assigned to any pattern-based group. In this

Algorithm 3 Pseudocode of the function `isTrended(ts)`

```

t, s, r ← seasonalMeanDecompose(ts)
if t.mean() > 0.19 then
  return Time series is trended
else
  return Time series is NOT trended
end if

```

Algorithm 4 Pseudocode of the function `isTrendedAndSeasonal(ts)`

```

if not isWhiteNoise(ts) and isTrended(ts)
and isSeasonal(ts) then
  return Time series is trended and seasonal
else
  return Time series is NOT trended and seasonal
end if

```

case, the mean of the distances between each non-assigned time series and each group time series is calculated. Then, the smallest mean distance is determined and the non-assigned time series is then assigned to the corresponding group. The pseudocode for the separation method is shown by Algorithm 5. The variable df referenced by the algorithm is a data frame in which each column is a time series and each row is a time step.

Once the time series is classified as trended, seasonal, white noise, or trended and seasonal, it is then imputed by a literature recommended imputation method. Trended time series are imputed using forward filling imputation, Seasonal time series are imputed using spline interpolation imputation, white noise time series are imputed using mean imputation, and seasonal and trended time series are imputed using spline interpolation.

As previously mentioned, the pattern classification methods are not exact, they are heuristics. To test their effectiveness, 500 synthetic data time series of each pattern were generated with random components, and random noise. Since the data generation process is controlled, it was possible to compute the accuracy of the classifications when running the Algorithms 1, 2, 3, and 4 over the data frame (DF) of 500 synthetic time series for each Pattern (totaling 2000 time series). The pseudocode of the generation of synthetic data is shown by Algorithm 6. The `isWhiteNoise()` method was tested by running it over the 500 white-noised time series and computing if any of them was not classified as white noise. After, the `isWhiteNoise()` method was run over the 500 Seasonal, 500 Trended and 500 Trended and Seasonal time series and it was registered if any of them were wrongly misclassified as white noise. This procedure was repeated for all three other methods: `isSeasonal()`, `isTrended()`, and `isTrendedAndSeasonal()`. More test were conducted with higher noise amplitudes k_w . One fact worth noting is that the seasonality in trended and seasonal time series should be recognized and those time series should be classified as seasonal by the `isSeasonal()` method. The same should occur for trended and seasonal

Algorithm 5 Pseudocode of the function `separate(df)`

```

count_white_noise ← 0
count_t_and_s ← 0
count_trended ← 0
count_seasonal ← 0
count_n_a ← 0
for  $i = 0$  to  $i < \text{length}(df.\text{columns})$  do
  if isWhiteNoise(df[:,i]) then
     $df\_white\_noise[:, \text{count\_white\_noise}] \leftarrow df[:, i]$ 
     $\text{count\_white\_noise} \leftarrow \text{count\_white\_noise} + 1$ 
  else if isTrendedAndSeasonal(df[:,i]) then
     $df\_t\_and\_s[:, \text{count\_t\_and\_s}] \leftarrow df[:, i]$ 
     $\text{count\_t\_and\_s} \leftarrow \text{count\_t\_and\_s} + 1$ 
  else if isTrended(df[:,i]) then
     $df\_trended[:, \text{count\_trended}] \leftarrow df[:, i]$ 
     $\text{count\_trended} \leftarrow \text{count\_trended} + 1$ 
  else if isSeasonal(df[:,i]) then
     $df\_seasonal[:, \text{count\_seasonal}] \leftarrow df[:, i]$ 
     $\text{count\_seasonal} \leftarrow \text{count\_seasonal} + 1$ 
  else
     $df\_n\_a[:, \text{count\_n\_a}] \leftarrow df[:, i]$ 
     $\text{count\_n\_a} \leftarrow \text{count\_n\_a} + 1$ 
  end if
end for
for  $i = 0$  to  $i < \text{length}(df\_n\_a.\text{columns})$  do
   $\text{dist\_w\_n} \leftarrow \text{meanDist}(df\_n\_a[:, i], df\_white\_noise)$ 
   $\text{dist\_s} \leftarrow \text{meanDist}(df\_n\_a[:, i], df\_seasonal)$ 
   $\text{dist\_t} \leftarrow \text{meanDist}(df\_n\_a[:, i], df\_trended)$ 
   $\text{dist\_t\_s} \leftarrow \text{meanDist}(df\_n\_a[:, i], df\_t\_and\_s)$ 
   $\text{min\_dist} \leftarrow \min(\text{dist\_w\_n}, \text{dist\_s}, \text{dist\_t}, \text{dist\_t\_s})$ 
  if  $\text{min\_dist} == \text{dist\_w\_n}$  then
     $df\_white\_noise[:, \text{count\_white\_noise}] \leftarrow df\_n\_a[:, i]$ 
     $\text{count\_white\_noise} \leftarrow \text{count\_white\_noise} + 1$ 
  else if  $\text{min\_dist} == \text{dist\_t\_and\_s}$  then
     $df\_t\_and\_s[:, \text{count\_t\_and\_s}] \leftarrow df\_n\_a[:, i]$ 
     $\text{count\_t\_and\_s} \leftarrow \text{count\_t\_and\_s} + 1$ 
  else if  $\text{min\_dist} == \text{dist\_trended}$  then
     $df\_trended[:, \text{count\_trended}] \leftarrow df\_n\_a[:, i]$ 
     $\text{count\_trended} \leftarrow \text{count\_trended} + 1$ 
  else if  $\text{min\_dist} == \text{dist\_seas}$  then
     $df\_seasonal[:, \text{count\_seasonal}] \leftarrow df\_n\_a[:, i]$ 
     $\text{count\_seasonal} \leftarrow \text{count\_seasonal} + 1$ 
  end if
end for

```

time series and the `isTrended()` method.

For $k_w = 2$, the white noise test got all 500 white noise time series correctly and did not misclassify any other time series of different pattern as white noise. The seasonality test recognized 97% of the seasonal time series as seasonal, recognized that there was seasonality in all time series that were both trended and seasonal, and did not misclassify any other time series of different pattern as seasonal. The trended test recognized 100% of the trended time series as trended, recognized that there was some trend in all time series that were both trended and seasonal, and misclassified only one seasonal time series as trended. The trended and seasonal test got everything correctly. The errors of miss-classification are shown in Table 2.

| Method v. Error | White Noise | Seasonal | Trended | Trended and Seasonal |
|-------------------------------------|-------------|----------|---------|----------------------|
| <code>isWhiteNoise()</code> | 0% | 0% | 0% | 0% |
| <code>isSeasonal()</code> | 0% | 3% | 0% | 0% |
| <code>isTrended()</code> | 0% | 0,2% | 0% | 0% |
| <code>isTrendedAndSeasonal()</code> | 0% | 0% | 0% | 0% |

Table 2 – Separation Methods versus the error when classifying 500 time series of each pattern with $k_w = 2$

Other tests with bigger k_w , meaning higher amplitude noise, were also done and the results continued to be satisfactory. For $k_w = 5$, the white noise test got all 500 white noise time series correctly and misclassified 6% of seasonal time series as white noise. The seasonality test recognized 88% of the seasonal time series as seasonal, also recognized that there was seasonality in all time series that were both trended and seasonal, and did not misclassify any other time series of different pattern as seasonal. The trended test recognized 100% of the trended time series as trended, recognized that there was some trend in all time series that were both trended and seasonal, and misclassified 6% of seasonal time series as trended. The trended and seasonal test recognized 100% of the trended and seasonal time series as trended and seasonal and misclassified 4% of seasonal time series as trended and seasonal. The errors of miss-classification are shown in Table 3.

| Method v. Error | White Noise | Seasonal | Trended | Trended and Seasonal |
|-------------------------------------|-------------|----------|---------|----------------------|
| <code>isWhiteNoise()</code> | 0% | 6% | 0% | 0% |
| <code>isSeasonal()</code> | 0% | 12% | 0% | 0% |
| <code>isTrended()</code> | 0% | 6% | 0% | 0% |
| <code>isTrendedAndSeasonal()</code> | 0% | 0% | 4% | 0% |

Table 3 – Separation Methods versus the error when classifying 500 time series of each pattern with $k_w = 5$

For $k_w = 10$, the white noise test got all 500 white noise time series correctly and misclassified 12.4% of seasonal time series as white noise. The seasonality test recognized 77% of the seasonal time series as seasonal, also recognized that there was seasonality in all time series that were both trended and seasonal, and misclassified 4.8% of trended time series as seasonal. The trended test recognized 100% of the trended time series as trended, recognized that there was some trend in all time series that were both trended and seasonal, and misclassified 13.4% of seasonal time series as trended. The trended and seasonal test recognized 100% of the trended and seasonal time series as trended and seasonal and misclassified 10.6% of seasonal time series and 5% of trended time series as trended and seasonal. The errors of miss-classification are shown in Table 4.

| Method v. Error | White Noise | Seasonal | Trended | Trended and Seasonal |
|-------------------------------------|-------------|----------|---------|----------------------|
| <code>isWhiteNoise()</code> | 0% | 12.4% | 0% | 0% |
| <code>isSeasonal()</code> | 0% | 23% | 4.8% | 0% |
| <code>isTrended()</code> | 0% | 13.4% | 0% | 0% |
| <code>isTrendedAndSeasonal()</code> | 0% | 10.6% | 5% | 0% |

Table 4 – Separation Methods versus the error when classifying 500 time series of each pattern with $k_w = 10$

The library¹ created with implementations of the two newly proposed methods is available for use and consultation. The generation of the synthetic time series and implementation of the tests conducted are also available at the repository.

¹ <https://github.com/silvanaribeiro/imputationLibrary>

Algorithm 6 Pseudocode of the generation of the synthetic data for testing

```

lower_bound ← 1
upper_bound ← 10
lower_bound2 ← 100
upper_bound2 ← 200
upper_bound_w_n ← 2
for  $i = 0$  to  $i \leq 500$  do
  white_noise[ $i$ ] ← random.gauss(0.0, 1.0)
end for
for  $i = 0$  to  $i \leq 500$  do
   $k_s$  ← random.int(lower_bound, upper_bound)
   $k_w$  ← random.int(lower_bound, upper_bound_w_n)
   $p$  ← random.int(lower_bound, upper_bound)
  xlim = 100
   $x$  ← [0, 0.1, 0.2, ..., 100]
   $y$  ←  $\sin(p \times x) \times k_s$ 
  seasonal ←  $y + \text{white\_noise} \times k_w$ 
end for
for  $i = 0$  to  $i \leq 500$  do
   $k_t$  ← random.int(lower_bound, upper_bound)
   $k_w$  ← random.int(lower_bound, upper_bound_w_n)
  xlim = 100
   $x$  ← [0, 0.1, 0.2, ..., 100]
  trend ←  $k_t \times x + \text{white\_noise} \times k_w$ 
end for
for  $i = 0$  to  $i \leq 500$  do
   $k_t$  ← random.int(lower_bound, upper_bound)
   $k_w$  ← random.int(lower_bound, upper_bound_w_n)
  xlim = 50
   $x$  ← [0, 0.1, 0.2, ..., 50]
   $x2$  ← flip( $x$ )
   $x$  ← append( $x, x2$ )
  trend ←  $k_t \times x + \text{white\_noise} \times k_w$ 
end for
for  $i = 0$  to  $i \leq 500$  do
   $k_s$  ← random.int(lower_bound, upper_bound)
   $k_t$  ← random.int(lower_bound, upper_bound)
   $k_w$  ← random.int(lower_bound, upper_bound_w_n)
   $p$  ← random.int(lower_bound, upper_bound)
  xlim = 100
   $x$  ← [0, 0.1, 0.2, ..., 100]
   $y$  ←  $\sin(p \times x) \times k_s$ 
  seasonal_and_trended ←  $y + x \times k_t + \text{white\_noise} \times k_w$ 
end for

```

Chapter 6

Case Studies

The following sections present the three case studies used in this work to verify the effectiveness of the imputation methods. The data and their origins are presented, as well as the objective of each case study.

6.1 Financial Indexes and Instability Trackers

The case study consists of generating a model to predict (classify) US Market instability (BEAR or BULL markets shown in Figure 12) through supervised learning and given a group of stock market indexes and a group of trackers that correlate market volatility to other subjects and keywords mentioned in newspaper articles [Chen and Tsang, 2018] [Chen, 2019]. However, the data is full of missingness, to begin with, and, as most data sets have different granularities, once merged the problem becomes even greater. Being so, treating the missingness of the data set becomes crucial to achieve the final objective of predicting market instability.

The data from financial indexes were taken from Yahoo Finance, and are described as follows [Haugen, 1986]:

- Standard and Poor's 500 (S&P 500): Stock Market index that tracks the stocks of the 500 biggest large-capital US Companies.
- Volatility Index (VIX): Index based on S&P 500 that measures the market's expectation of future volatility.
- Dow Jones Industrial Average (Dow 30): Stock Market index that combines the stock price of 30 large, publicly-traded companies to determine the industrial average.
- NASDAQ 100: Stock Market Index that includes the shares of the 100 largest American and international non-financial companies that are traded on the Nasdaq electronic stock exchange.

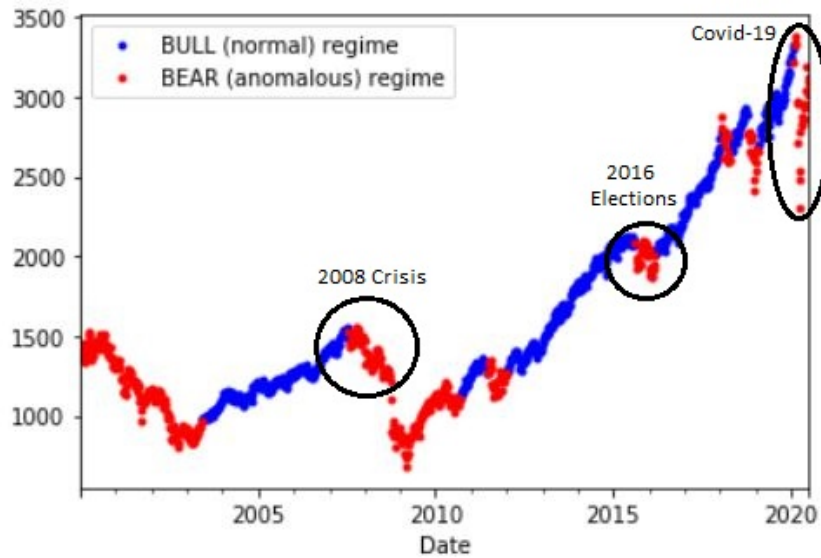


Figure 12 – S&P 500 close index classified as BEAR or BULL

- NIKKEI 225: Japanese Stock Market Index that includes the shares of Japan's top 225 companies traded on the Tokyo Stock Exchange.
- FTSE 100: Stock Market Index that consists of the 100 most highly capitalized companies in the UK.
- Hang Seng (HSI): Stock Market Index that consists of the largest companies that trade on the Hong Kong Exchange. It covers approximately 65% of the total market capitalization of the Hong Kong Exchange.
- Euronext 100: European Stock Market Index that consists of the 100 largest and most liquid blue-chip stocks traded on Euronext exchanges.

The trackers were taken from Economic Policy Uncertainty and are, as follows:

- US Equity Market Volatility Index: Newspaper-based Equity market Volatility tracker that is constructed based on the counts of the monthly occurrences of the terms "economic", "economy", "financial", "stock market", "equity", "equities", "Standard and Poors" (and variants), "volatility", "volatile", "uncertain", "risk" and "risky" on eleven major US newspapers [Baker et al., 2019]. This tracker has monthly granularity.
- Daily Infectious Disease Equity Market Volatility Tracker: Newspaper-based Infectious Disease Equity Market Volatility Tracker that is constructed based on the counts of the daily occurrences of the terms "economic", "economy", "financial", "stock market", "equity", "equities", "Standard and Poors" (and variants),

"volatility", "volatile", "uncertain", "risk", "risky", "epidemic", "pandemic", "virus", "flu", "disease", "coronavirus", "mers", "sars", "ebola", "H5N1" and "H1N1" on approximately 3,000 US Newspapers [Baker et al., 2019]. This tracker has daily granularity.

- Geopolitical Risk Index: Newspaper-based Geopolitical Risk Index that is constructed by counting the occurrence of words related to geopolitical tensions in 11 leading international newspapers [Caldara and Iacoviello, 2018]. This index has monthly granularity.
- Trade Policy Uncertainty and Market Volatility: Newspaper-based Trade Policy Uncertainty tracker that is constructed based on the counts of the frequency of occurrences of trade policy and uncertainty terms across major newspapers over the world [Caldara et al., 2020]. This tracker has monthly granularity.

6.2 COVID-19

This data set can be found at Kaggle¹ and contains information about COVID-19 confirmed cases, recovered patients, and deaths of several cities throughout the world. The time window is 30 minutes, although there are reported gaps. This data set has different rows for each city. In order to turn it into a typical time series, the information of confirmed cases, recovered patients and deaths were grouped by region and turned into column information instead of a row for each region. The objective for this problem is to predict Asia's confirmed cases using information from other regions throughout time. One particularity of this problem is that the target variable, confirmed cases in Asia, has missing time steps as shown in Figure (13).

6.3 Deng

This data set can be found at Driven Data² and it is from a competition to predict disease spread. It has information from two cities: San Juan and Iquitos. This data set's granularity is weekly. It contains information about daily climate, precipitation, and vegetation. This data set also contains different rows for different cities for the same date and this information was transformed into columns for the same date. The objective for this problem is to predict total cases of Deng for both cities as shown in Figure (14).

For each city there are the following features:

¹ COVID-19 Timeseries <https://www.kaggle.com/lihyalan/2020-corona-virus-timeseries>

² DengAI: Predicting Disease Spread <https://www.drivendata.org/competitions/44/dengai-predicting-disease-spread/page/82/>

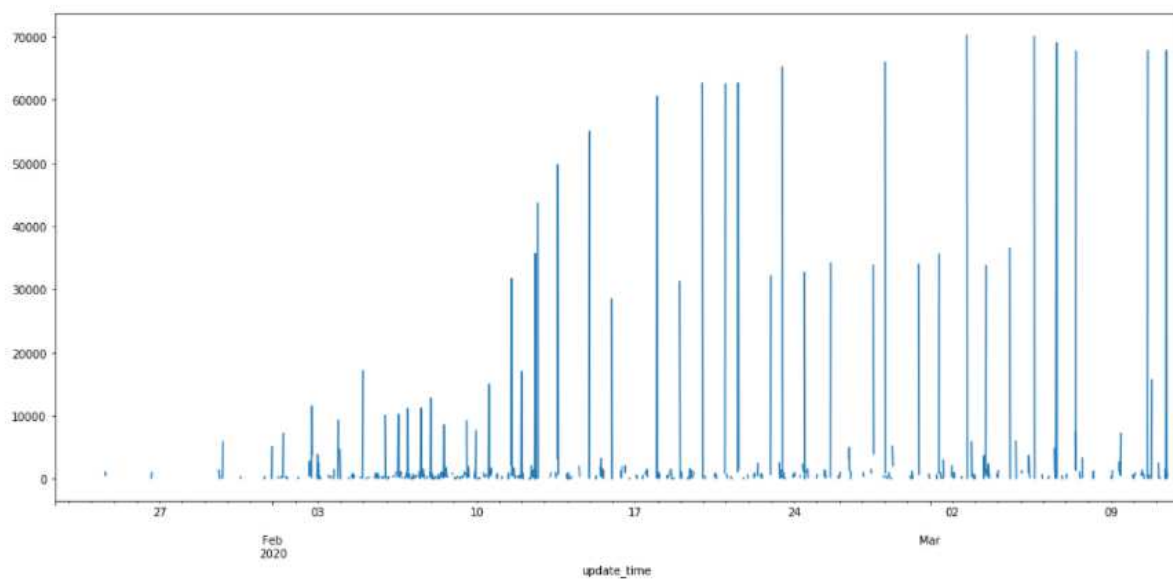


Figure 13 – COVID-19 Asia Confirmed cases

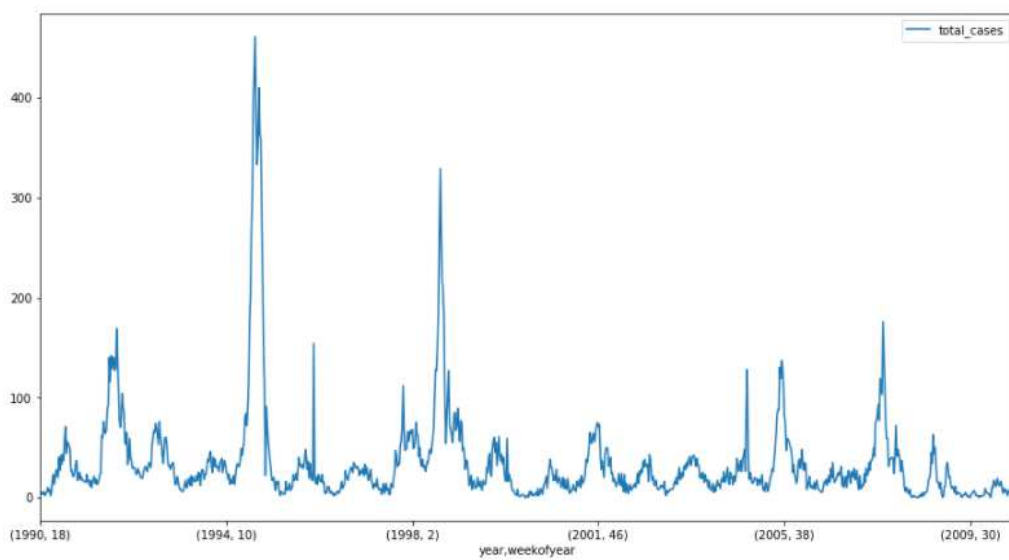


Figure 14 – Deng total cases

- `week_start_date`: Date of measurement
- `station_max_temp_c`: Maximum Temperature measurement from NOAA's GHCN weather station
- `station_min_temp_c`: Minimum temperature measurement from NOAA's GHCN weather station
- `station_avg_temp_c`: Average temperature measurement from NOAA's GHCN weather station
- `station_precip_mm`: Total precipitation measurement from NOAA's GHCN weather station
- `station_diur_temp_rng_c`: Diurnal temperature range measurement from NOAA's GHCN weather station
- `precipitation_amt_mm`: Total precipitation measurement from PERSIANN satellite
- `reanalysis_sat_precip_amt_mm`: Total precipitation measurement from NOAA's NCEP Climate Forecast System Reanalysis
- `reanalysis_dew_point_temp_k`: Mean dew point temperature measurement from NOAA's NCEP Climate Forecast System Reanalysis
- `reanalysis_air_temp_k`: Mean air temperature measurement from NOAA's NCEP Climate Forecast System Reanalysis
- `reanalysis_relative_humidity_percent`: Mean relative humidity measurement from NOAA's NCEP Climate Forecast System Reanalysis
- `reanalysis_specific_humidity_g_per_kg`: Mean specific humidity measurement from NOAA's NCEP Climate Forecast System Reanalysis
- `reanalysis_precip_amt_kg_per_m2`: Total precipitation measurement from NOAA's NCEP Climate Forecast System Reanalysis
- `reanalysis_max_air_temp_k`: Maximum air temperature measurement from NOAA's NCEP Climate Forecast System Reanalysis
- `reanalysis_min_air_temp_k`: Minimum air temperature measurement from NOAA's NCEP Climate Forecast System Reanalysis
- `reanalysis_avg_temp_k`: Average air temperature measurement from NOAA's NCEP Climate Forecast System Reanalysis
- `reanalysis_tdtr_k`: Diurnal temperature range measurement from NOAA's NCEP Climate Forecast System Reanalysis

Chapter 7

Experimental Methodology

A diagram of the experimental methodology is shown in Figure 15 and an explanation of each step is given as follows; for implementation details see the Github repository¹.

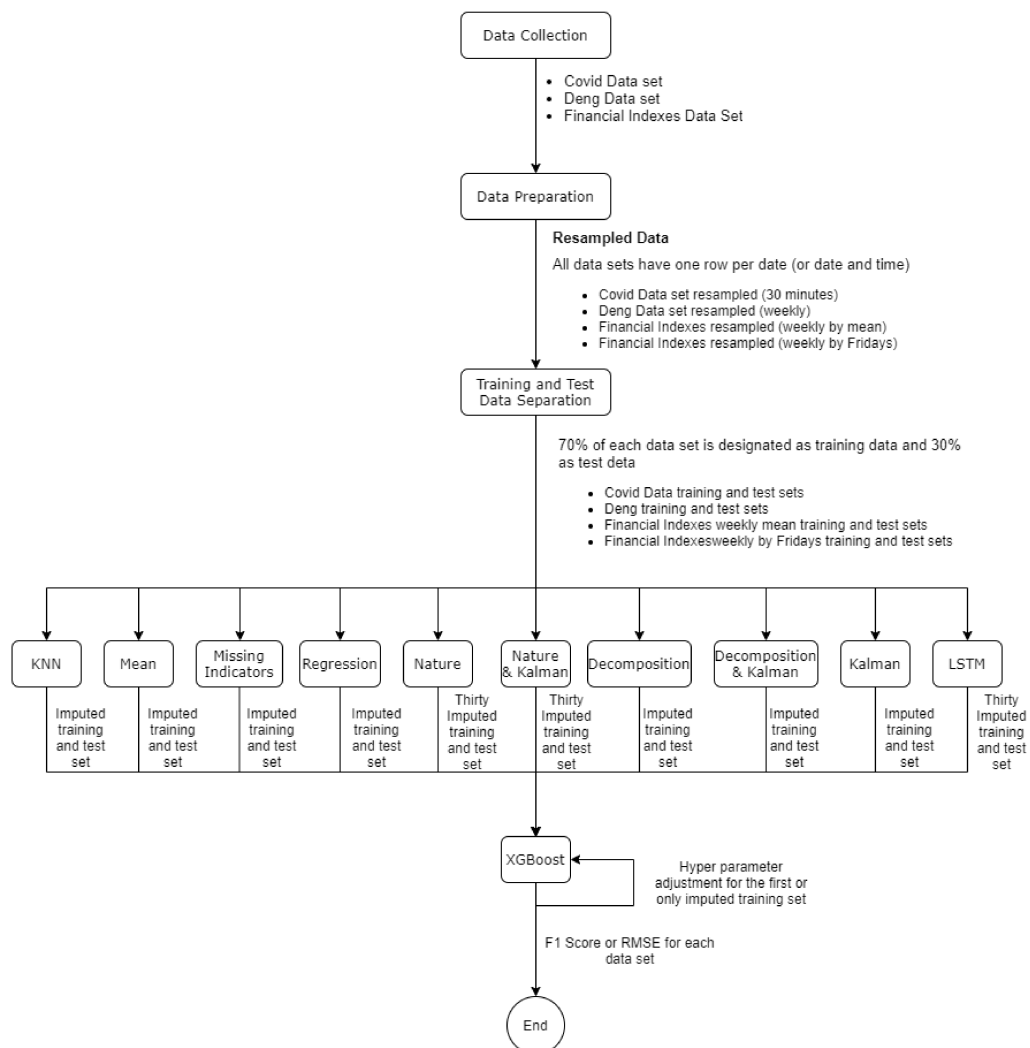


Figure 15 – Diagram of the methodology

¹ <https://github.com/silvanaribeiro/dissertacao/>

7.1 Data Collection and Data Preparation

The data sets were collected manually from all sources. As the three data sets have distinct characteristics the data preparation step was very different for each one of them and is presented separately in the following subsections.

7.1.1 Financial Indexes and Instability Trackers

Seven different data sources were used for this case study. As the granularity of the data being analyzed is different for each data source used, once they are merged the resulting data set becomes full of missing values. Some of the individual data sets acquired from the seven sources already present missingness even before the merge. As the missingness of the data can, in some cases, be predicted by the temporal feature, but there is also missingness that cannot be predicted by the observed data, the missingness mechanism is considered a combination of MAR and MCAR.

An analysis of the missingness in each data source was conducted to understand them better. Using the S&P500 index as a base, the dates for which a value was expected for all indexes were set and the analysis was done. This step is available at repository² and corresponds to the notebooks starting with prefix 02 in the directories *FinancialIndex_FridayResampling* and *FinancialIndex_MeanResampling*. Some of the missingness occurs because different markets open on different days and some occur because the data is not available. The results obtained for each data set were as shown in Table 5.

| Data set | Missing dates | Missing values |
|----------------------------|---------------|----------------|
| S&P 500 | 0 | 0 |
| VIX | 0 | 0 |
| Dow 30 | 0 | 0 |
| NASDAQ 100 | 0 | 0 |
| NIKKEI 225 | 173 | 124 |
| FTSE 100 | 283 | 61 |
| HSI | 165 | 84 |
| Euronext 100 | 37 | 32 |
| Us Market Volatility Index | NA | 0 |
| Daily Infectious Disease | NA | 0 |
| Geopolitical Risk Index | NA | 0 |
| Trade Policy uncertainty | NA | 24 |

Table 5 – Missingness analysis of each feature in relation to S&P 500 before merging the data sets

The data were merged as illustrated by Figure 16. From the financial Indexes data

² <https://github.com/silvanaribeiro/dissertacao>

sources, the close and volume features were used from each index. From the trackers, the features used were the Overall EMV tracker, Daily Infectious EMV index, GPR, US Trade Policy Uncertainty, Japanese Trade Policy Uncertainty, and Trade Policy Uncertainty EMV Fraction. The obtained data set has 22 features and 7383 samples. The period of analysis ranges from January 2, 2015, to March 13, 2020. This step is available at the repository and corresponds to the notebook starting with the prefix 03 in the directories *FinancialIndex_FridayResampling* and *FinancialIndex_MeanResampling*.

The daily data set was resampled to weekly granularity by calculating each week’s average. Another data set was created by resampling the data set using information from each Friday. The final data sets are each composed of 1054 dates (rows or samples) and 22 features (columns). The period of analysis ranges from January 2, 2015, to March 13, 2020. This step is available at the repository and also corresponds to the notebook starting with prefix 03 in the directories *FinancialIndex_FridayResampling* and *FinancialIndex_MeanResampling*.

After merging the 12 data sets acquired from the 12 data sources, of different granularity and re-sampling it, another analysis was conducted to verify missingness. This step is available at the repository and corresponds to the notebook starting with the prefix 04 in the directories *FinancialIndex_FridayResampling* and *FinancialIndex_MeanResampling*. The results are presented by Tables 6 and 7 for the data set resampled by the weeks’ mean and Fridays respectively.

| Feature | Missing values |
|-----------------------------------|----------------|
| S&P 500 Close and Volume | 0 |
| VIX Close and Volume | 0 |
| Dow 30 Close and Volume | 0 |
| NASDAQ 100 Close and Volume | 0 |
| NIKKEI 225 Close and Volume | 2 |
| FTSE 100 Close and Volume | 45 |
| HSI Close and Volume | 0 |
| Euronext 100 Close and Volume | 0 |
| Us Market Volatility Index | 813 |
| Daily Infectious Disease | 0 |
| Geopolitical Risk Index | 813 |
| Trade Policy uncertainty | 820 |
| US Trade Policy uncertainty | 820 |
| Japanese Trade Policy uncertainty | 820 |

Table 6 – Missingness analysis of each feature after merging and re-sampling the data sets by each week’s mean

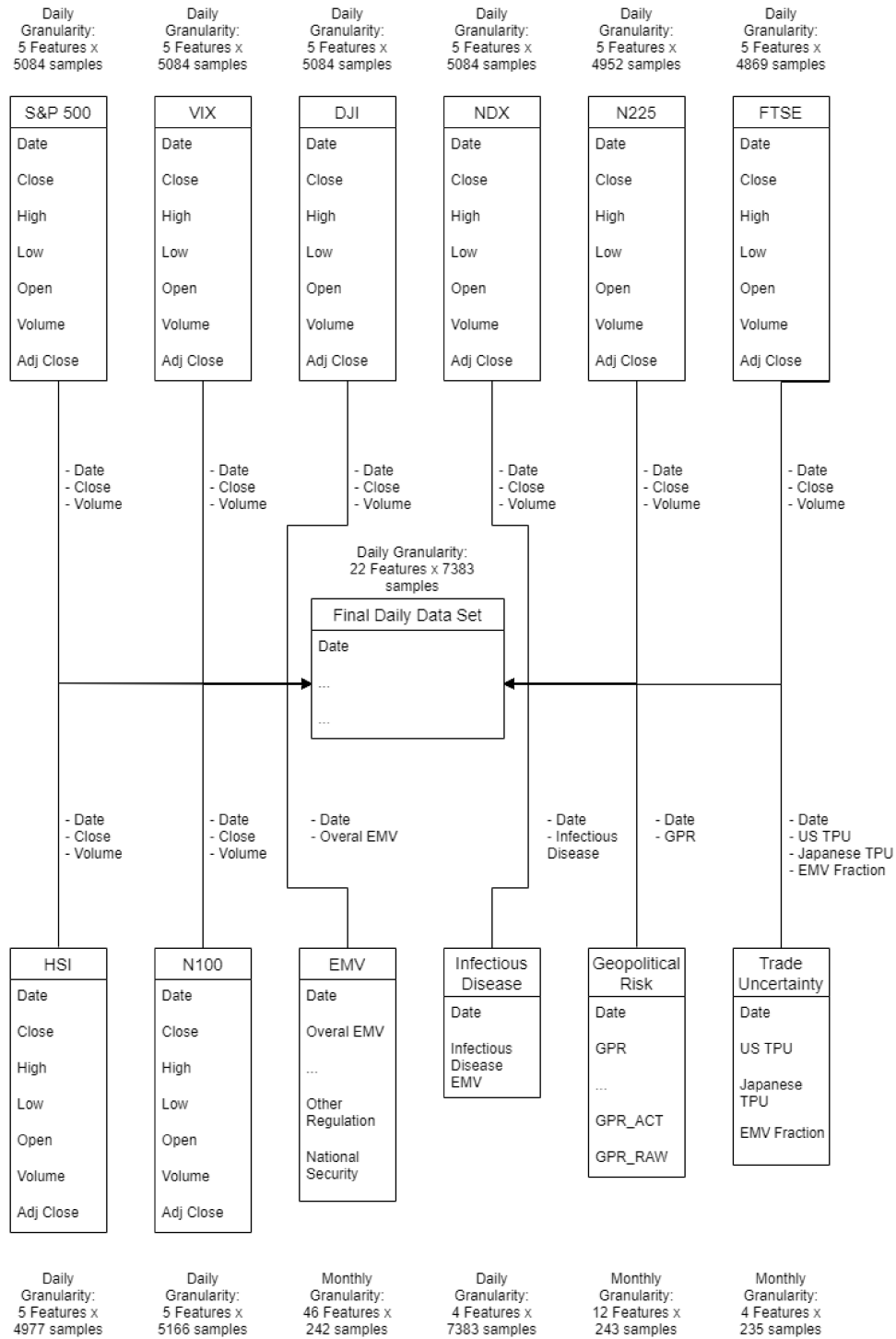


Figure 16 – Diagram showing information about the original data sets and the final daily data set after the merger

| Feature | Missing values |
|-----------------------------------|----------------|
| S&P 500 Close and Volume | 2299 |
| VIX Close and Volume | 2299 |
| Dow 30 Close and Volume | 2299 |
| NASDAQ 100 Close and Volume | 2299 |
| NIKKEI 225 Close and Volume | 2431 |
| FTSE 100 Close and Volume | 2514 |
| HSI Close and Volume | 2406 |
| Euronext 100 Close and Volume | 2217 |
| Us Market Volatility Index | 7141 |
| Daily Infectious Disease | 0 |
| Geopolitical Risk Index | 7140 |
| Trade Policy uncertainty | 7148 |
| US Trade Policy uncertainty | 7148 |
| Japanese Trade Policy uncertainty | 7148 |

Table 7 – Missingness analysis of each feature after merging and re-sampling the data sets by each week’s Friday

The autocorrelation plot of every feature was visually analyzed as shown in Table 8. It was concluded that the data set is a mixture of all types of time series pattern, which makes it difficult to choose one method that would fit all cases. The autocorrelation plots of the features can be found at the repository and correspond to the notebook 05 in the directories *FinancialIndex_FridayResampling* and *FinancialIndex_MeanResampling*.

| Feature | Trended | Seasonal | White-noise |
|-----------------------------------|---------|----------|-------------|
| S&P 500 Close and Volume | X | | |
| VIX Close and Volume | X | | |
| Dow 30 Close and Volume | X | | |
| NASDAQ 100 Close and Volume | X | | |
| NIKKEI 225 Close and Volume | X | | |
| FTSE 100 Close and Volume | X | | |
| HSI Close and Volume | X | | |
| Euronext 100 Close and Volume | X | | |
| Us Market Volatility Index | X | X | |
| Daily Infectious Disease | | | X |
| Geopolitical Risk Index | X | X | |
| Trade Policy uncertainty | X | X | |
| US Trade Policy uncertainty | X | X | |
| Japanese Trade Policy uncertainty | X | | |

Table 8 – Visual analysis of the autocorrelation plot of each feature

7.1.2 COVID-19

The data was collected, the information about confirmed cases, recovered patients and deaths has been grouped by region. Afterward, the rows containing information from each region were transformed into columns and merged by date and time, as shown by Figure 17. Finally, the data set was resampled to 30-minute windows.

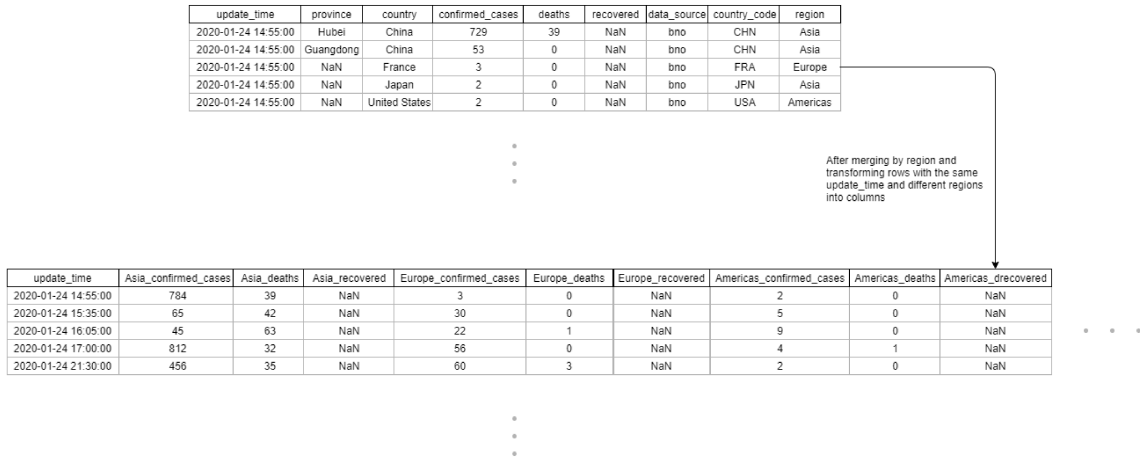


Figure 17 – Diagram of the Data Preparation step of the COVID-19 data set

Afterward, an analysis was conducted to verify missingness. This step and the previous ones are available at the repository and correspond to the notebook starting with the prefix 01 in the *Covid* directory. The results are presented by Table 9.

| Feature | Missing values |
|--------------------------|----------------|
| Asia_confirmed_cases | 747 |
| Asia_deaths | 747 |
| Asia_recovered | 747 |
| Europe_confirmed_cases | 1681 |
| Europe_deaths | 1681 |
| Europe_recovered | 1681 |
| Americas_confirmed_cases | 1725 |
| Americas_deaths | 1725 |
| Americas_recovered | 1725 |
| Oceania_confirmed_cases | 1781 |
| Oceania_deaths | 1781 |
| Oceania_recovered | 1781 |
| Africa_confirmed_cases | 1802 |
| Africa_deaths | 1802 |
| Africa_recovered | 1802 |

Table 9 – Missingness analysis of each feature of the COVID-19 data set

7.1.3 Deng

The data was collected and the rows containing information from each city were transformed into columns and merged by date. The data from the target variable, Deng total cases, was originally from each city, but the information for each city and each date was added so that the data predicted is the total cases in all cities, as shown by Figure 18.

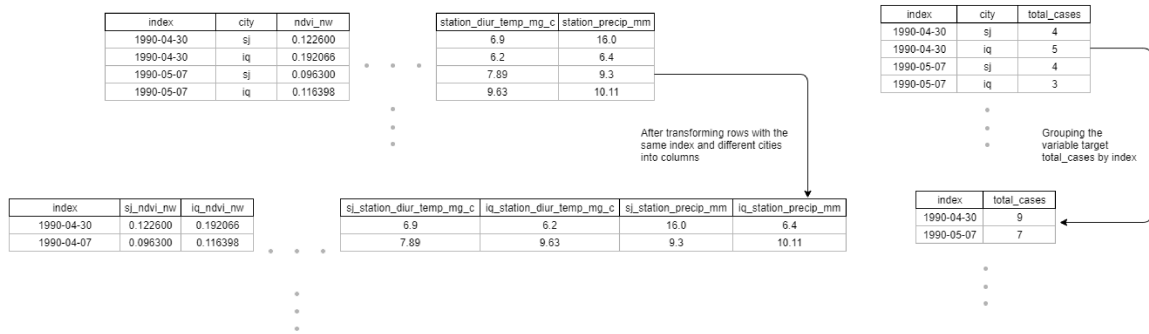


Figure 18 – Diagram of the Data Preparation step of the Deng data set

Afterward, an analysis was conducted to verify missingness. This step and the previous ones are available at the repository and correspond to the notebook starting with the prefix 01 in the *Deng* directory. The results are presented by Table 10.

| Feature | Missing values |
|--|----------------|
| sj_ndvi_ne | 210 |
| sj_ndvi_nw | 56 |
| sj_ndvi_se | 20 |
| sj_ndvi_sw | 20 |
| sj_precipitation_amt_mm | 9 |
| sj_reanalysis_air_temp_k | 8 |
| sj_reanalysis_avg_temp_k | 8 |
| sj_reanalysis_dew_point_temp_k | 8 |
| sj_reanalysis_max _{air} _temp_k | 8 |
| sj_reanalysis_min_air_temp_k | 8 |
| sj_reanalysis_precip_amt_kg_per_m2 | 8 |
| sj_reanalysis_relative_humidity_percent | 8 |
| sj_reanalysis_sat_precip_amt_mm | 11 |
| sj_reanalysis_specific_humidity_g_per_kg | 8 |
| sj_reanalysis_tdtr_k | 8 |
| sj_station_avg_temp_c | 8 |
| sj_station_diur_temp_rng_c | 8 |
| sj_station_max_temp_c | 8 |
| sj_station_min_temp_c | 8 |
| sj_station_precip_mm | 8 |
| iq_ndvi_ne | 532 |
| iq_ndvi_nw | 532 |
| iq_ndvi_se | 532 |
| iq_ndvi_sw | 532 |
| iq_precipitation_amt_mm | 533 |
| iq_reanalysis_air_temp_k | 533 |
| iq_reanalysis_avg_temp_k | 533 |
| iq_reanalysis_dew_point_temp_k | 533 |
| iq_reanalysis_max _{air} _temp_k | 533 |
| iq_reanalysis_min_air_temp_k | 533 |
| iq_reanalysis_precip_amt_kg_per_m2 | 533 |
| iq_reanalysis_relative_humidity_percent | 533 |
| iq_reanalysis_sat_precip_amt_mm | 533 |
| iq_reanalysis_specific_humidity_g_per_kg | 533 |
| iq_reanalysis_tdtr_k | 533 |
| iq_station_avg_temp_c | 566 |
| iq_station_diur_temp_rng_c | 566 |
| iq_station_max_temp_c | 543 |
| iq_station_min_temp_c | 537 |
| iq_station_precip_mm | 545 |

Table 10 – Missingness analysis of each feature of the Deng data set

7.2 Training and Test Data separation

For all three data sets, approximately 70% of the data was designated as the training set and the remaining 30% for the test set. For this separation, the temporal factor of the data was taken into account and the data was not reordered. More information regarding the separation of data in training and test data set can be found at the repository³.

- Financial Indexes: The data is available from January 1, 2000 to March 13, 2020. The training set has data from January 1, 2000, to December 26, 2014, accounting for 782 weekly time steps. The test set has data from January 2, 2015, to March 13, 2020, accounting for 272 weekly time steps.
- COVID-19: The data is available from January 22, 2020, at 9 AM to March 11, 2020, at 9:30 PM. The training set has data from January 22, 2020, at 9 AM to February 26, 2020, at 2 PM, accounting for 1667 30-minute-window time steps. The test set has data from February 26, 2020 at 2 PM to March 11, 2020 at 9:30 PM, accounting for 715 30-minute-window time steps.
- Deng: The data is available from April 30, 1990, to June 25, 2010. The training set has data from April 30, 1990, to June 3, 2004, accounting for 734 weekly time steps. The test set has data from June 10, 2004 to June 25, 2010, accounting for 315 weekly time steps.

7.3 Imputation

All three data sets were imputed using the same imputation methods. The test data sets were imputed using only information from the training data sets. For instance, when imputing the test set using the Mean Imputation method, the mean of the training data set was used.

Table 11 shows the characteristics of each imputation method: if it is deterministic, how many times it was executed and if the data were standardized. For the LSTM imputation method, the data standardization was done by a Min-Max Scaler. For all other methods that standardized the data, it was done by removing its mean and then scaling to unit variance before the imputation.

For the imputation using Missing Indicators the empty time steps were filled with the value zero. The imputation using Decomposition, Additive Decomposition, and the imputation by Pattern combined with the Kalman filter were done by firstly imputing the data using the new methods and afterward applying the Kalman filter just to smooth the data. The imputation using LSTM for the Financial Index data set, used 30 time steps to

³ <https://github.com/silvanaribeiro/dissertacao/>

| Imputation Method\ Characteristics | Deterministic | Times Run | Standardized |
|---------------------------------------|---------------|-----------|--------------|
| KNN | Yes | 1 | Yes |
| Mean | Yes | 1 | No |
| Missing Indicators | Yes | 1 | No |
| Regression | Yes | 1 | Yes |
| Pattern | No | 30 | No |
| Pattern & Kalman | No | 30 | No |
| Decomposition | Yes | 1 | No |
| Additive Decomposition | Yes | 1 | No |
| Decomp. & Kalman | Yes | 1 | No |
| Add. Decomp & Kalman | Yes | 1 | No |
| Kalman | Yes | 1 | No |
| LSTM | No | 30 | Yes |

Table 11 – Imputation Methods Characteristics

predict 1 future time step, whereas, for the COVID-19 and Deng data sets, 10 time steps were used. Thirty epochs were used to fit the LSTM model.

7.4 XGBoost

The XGBoost⁴ algorithm was used to train models for the three data sets. XGBoost is an optimized gradient boosting library. It uses the Gradient Boosting framework [Friedman, 2001], providing a parallel tree boosting.

- **Financial Indexes:** The objective in this case is a binary classification using logistic regression. The F1 score is the evaluation metric used for this data set. The tuning of the hyperparameters was done manually and separately for each imputed data set. The usage of each hyperparameter can be found in the library documentation⁵. The hyperparameters that were tuned, as well as the ranges tested were as follows:
 - `colsample_bytree`: 0.3-1, step 0.1.
 - `gamma`: 0-10, step 1.
 - `learning_rate`: 0.1, 0.01, 0.001, 0.0001, 0.00001.
 - `max_delta_step`: 0-10, step 1.
 - `max_depth`: 0-10, step 1.
 - `min_child_weight`: 1-10, step 1.

⁴ <https://xgboost.readthedocs.io/en/latest/build.html>

⁵ "XGBoost Parameters" <https://xgboost.readthedocs.io/en/latest/parameter.html>

-
- `n_estimators`: 30-300, step 5, 300-2000, step 100.
 - `reg_alpha`: 0.1, 0.01, 0.001, 0.0001, 0.00001.
 - `scale_pos_weight`: 1-10, step 1, 20-1000, step 5.
 - `subsample`: 0.3-1, step 0.1.
- COVID-19 and Deng: The objective in this case is regression with squared error loss. The RMSE is the evaluation metric used for these data sets. The hyperparameters that were tuned, as well as the ranges tested, were as follows:
 - `learning_rate`: 0.1-0.9, step 0.1, 0.01, 0.001, 0.0001, 0.00001.
 - `max_depth`: 0-10, step 1.
 - `n_estimators`: 20-300, step 5, 300-1500, step 100.

Chapter 8

Results

The following sections present the results found for each data set and the imputation methods are ranked according to their effectiveness. Finally, the mean rank for each imputation method is presented.

8.1 Financial Indexes and Instability Trackers Resampled by the Week's Mean

The results for the Financial Indexes and Instability Trackers data set resampled using the week's mean are presented in Table 12. For the stochastic methods, the results presented by the column Mean F1 are the mean evaluation metric for the 30 rounds, along with its standard deviation. If the standard deviation is not presented, it is because the method is deterministic and was executed only once. The column Best F1 presents the best result achieved on the 30 rounds. Finally, the imputation methods are ranked using the mean and the best results. As it can be noted, the ranking did not change when considering the mean F1 or the best F1.

The best results were achieved by Kalman, Pattern plus Kalman, and Decomposition plus Kalman, in that order. Figures 19, 20 and 21 show the real versus predicted regime for the three best results in order. It can be noted that the results obtained are very similar.

| Imputation Method\ Data set | Mean F1 | Best F1 | Rank - Mean F1 | Rank - Best F1 |
|--------------------------------|---------------------|---------|----------------|----------------|
| KNN | 0.7162 | 0.7162 | 8 | 8 |
| Mean | 0.7480 | 0.7480 | 5 | 5 |
| Missing Indicators | 0.7407 | 0.7407 | 6 | 6 |
| Regression | 0.7343 | 0.7343 | 7 | 7 |
| Pattern | 0.7142 \pm 0 | 0.7142 | 9 | 9 |
| Pattern & Kalman | 0.8644 \pm 0 | 0.8644 | 2 | 2 |
| Decomposition | 0.7074 | 0.7074 | 10 | 10 |
| Additive Decomposition | 0.7074 | 0.7074 | 10 | 10 |
| Decomp. & Kalman | 0.8305 | 0.8305 | 3 | 3 |
| Add. Decomp & Kalman | 0.8166 | 0.8166 | 4 | 4 |
| Kalman | 0.8717 | 0.8717 | 1 | 1 |
| LSTM | 0.6416 \pm 0.0154 | 0.6720 | 11 | 11 |

Table 12 – Comparison of the mean F1 scores and best F1 scores for each imputed data set and ranking.

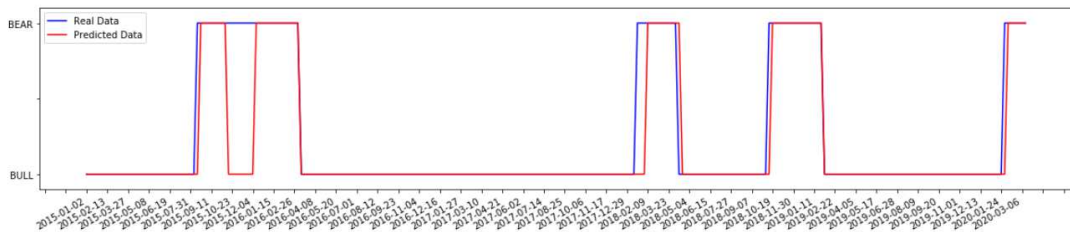


Figure 19 – BULL and BEAR real data and prediction made by model generated using the Kalman imputed data set

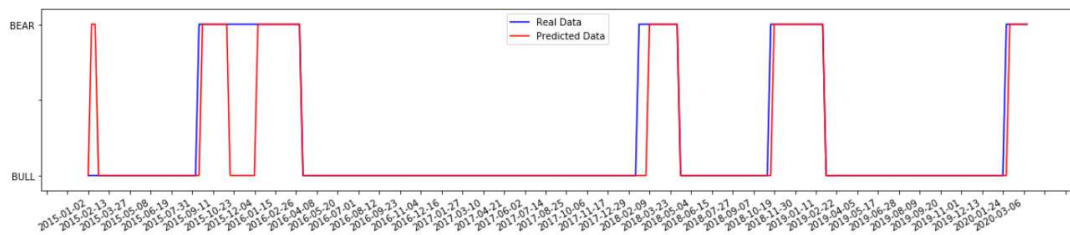


Figure 20 – BULL and BEAR real data and prediction made by model generated using the Pattern & Kalman imputed data set

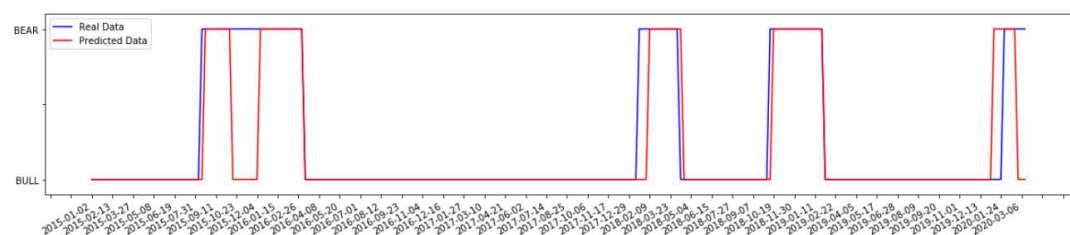


Figure 21 – BULL and BEAR real data and prediction made by model generated using the Decomposition & Kalman imputed data set

For the Financial Indexes data sets, whose problems are classifications, the Confusion Matrices using each imputation method are presented in Table 13. For the stochastic methods, the confusion matrices presented are the mean of the results of the thirty runs.

| Imputation Method | Resample by Mean | | | Resample by Friday | | |
|--|------------------|------|-------|--------------------|------|-------|
| KNN | Real v. Pred. | True | False | Real v. Pred. | True | False |
| | True | 53 | 30 | True | 48 | 21 |
| | False | 12 | 177 | False | 17 | 186 |
| Mean | Real v. Pred. | True | False | Real v. Pred. | True | False |
| | True | 49 | 17 | True | 47 | 16 |
| | False | 16 | 190 | False | 18 | 191 |
| Missing Indicators | Real v. Pred. | True | False | Real v. Pred. | True | False |
| | True | 50 | 20 | True | 47 | 16 |
| | False | 15 | 187 | False | 18 | 191 |
| Regression | Real v. Pred. | True | False | Real v. Pred. | True | False |
| | True | 47 | 16 | True | 48 | 16 |
| | False | 18 | 191 | False | 17 | 191 |
| Pattern | Real v. Pred. | True | False | Real v. Pred. | True | False |
| | True | 55 | 34 | True | 47 | 21 |
| | False | 10 | 173 | False | 17 | 186 |
| Pattern & Kalman | Real v. Pred. | True | False | Real v. Pred. | True | False |
| | True | 51 | 2 | True | 38 | 1 |
| | False | 14 | 205 | False | 27 | 206 |
| Decomposition | Real v. Pred. | True | False | Real v. Pred. | True | False |
| | True | 52 | 30 | True | 49 | 27 |
| | False | 13 | 177 | False | 16 | 180 |
| Additive Decomposition | Real v. Pred. | True | False | Real v. Pred. | True | False |
| | True | 52 | 30 | True | 48 | 26 |
| | False | 13 | 177 | False | 17 | 181 |
| Decomposition & Kalman | Real v. Pred. | True | False | Real v. Pred. | True | False |
| | True | 49 | 4 | True | 34 | 1 |
| | False | 16 | 203 | False | 31 | 206 |
| Additive Decomposition & Kalman | Real v. Pred. | True | False | Real v. Pred. | True | False |
| | True | 49 | 6 | True | 33 | 1 |
| | False | 16 | 201 | False | 32 | 206 |
| Kalman | Real v. Pred. | True | False | Real v. Pred. | True | False |
| | True | 51 | 1 | True | 46 | 0 |
| | False | 14 | 206 | False | 19 | 207 |
| LSTM | Real v. Pred. | True | False | Real v. Pred. | True | False |
| | True | 45 | 30.4 | True | 24 | 0 |
| | False | 20 | 176.6 | False | 41 | 207 |

Table 13 – Confusion Matrices of predictions made using the Financial Indexes data sets

8.2 Financial Indexes and Instability Trackers

Resampled by Fridays

The results for the Financial Indexes and Instability Trackers data set resampled using Fridays are presented in Table 14. For the stochastic methods, the results presented by the column Mean F1 are the mean evaluation metric for the 30 rounds, along with its standard deviation. If the standard deviation is not presented, it is because the method is

| Imputation Method\ Data set | Mean F1 | Best F1 | Rank - Mean F1 | Rank - Best F1 |
|--------------------------------|---------------------|---------|----------------|----------------|
| KNN | 0.7164 | 0.7164 | 5 | 5 |
| Mean | 0.7344 | 0.7344 | 3 | 3 |
| Missing Indicators | 0.7344 | 0.7344 | 3 | 3 |
| Regression | 0.7442 | 0.7442 | 2 | 2 |
| Pattern | 0.7011 \pm 0.0020 | 0.7121 | 6 | 6 |
| Pattern & Kalman | 0.7308 \pm 0 | 0.7308 | 4 | 4 |
| Decomposition | 0.6950 | 0.6950 | 7 | 7 |
| Additive Decomposition | 0.6906 | 0.6906 | 8 | 8 |
| Decomp. & Kalman | 0.68 | 0.68 | 9 | 9 |
| Add. Decomp & Kalman | 0.6666 | 0.6666 | 10 | 10 |
| Kalman | 0.8288 | 0.8288 | 1 | 1 |
| LSTM | 0.5393 \pm 0 | 0.5393 | 11 | 11 |

Table 14 – Comparison of the mean F1 scores and best F1 scores for each imputed data set and ranking.

deterministic and was executed only once. The column Best F1 presents the best result achieved on the 30 rounds. Finally, the imputation methods are ranked using the mean and the best results.

The best results were achieved by Kalman, Regression, and Missing Indicators and Mean tied for the third place. It is important to note that the results achieved by Pattern plus Kalman are fairly close to third place. Figures 22, 23 and 24 show the real versus predicted regime for the three best results in order. Figure 25 shows the results achieved by Pattern plus Kalman method. As it can be noted, although the Regression and Missing Indicators achieved better results, the Pattern plus Kalman did a better job in predicting real BULL market. This is probably the result of the Kalman filter having smoothed the data and as such, prevented the model to learn small variations as BULL. As it can be noted, the ranking did not change when considering the mean F1 or the best F1. The confusion matrices are shown in Table 13.

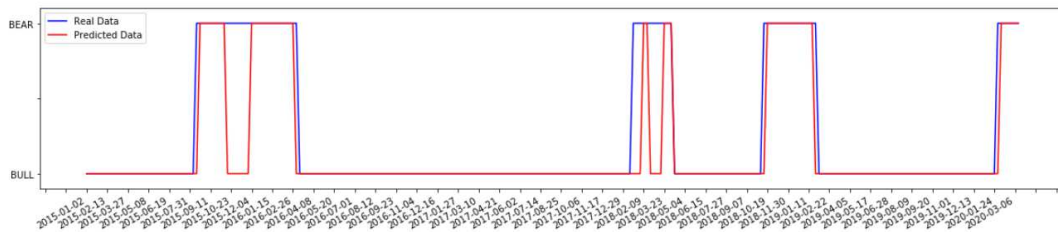


Figure 22 – BULL and BEAR real data and prediction made by model generated using the Kalman imputed data set

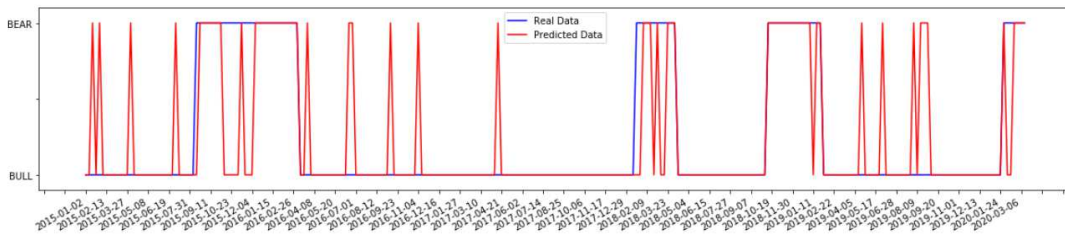


Figure 23 – BULL and BEAR real data and prediction made by model generated using the Regression imputed data set

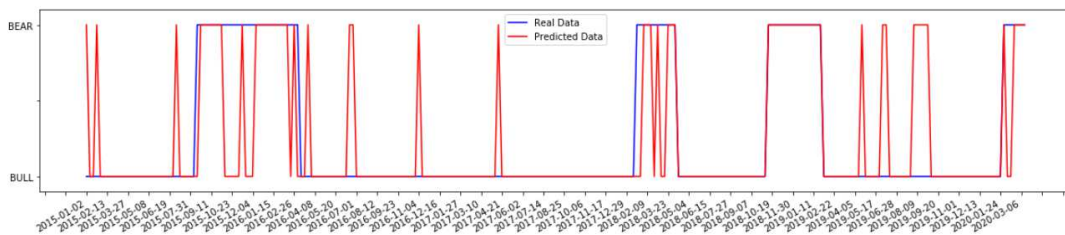


Figure 24 – BULL and BEAR real data and prediction made by model generated using the Missing Indicators imputed data set

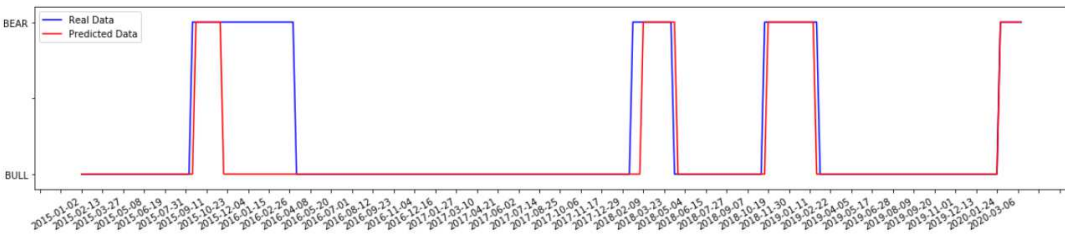


Figure 25 – BULL and BEAR real data and prediction made by model generated using the Pattern & Kalman imputed data set

8.3 COVID-19

The results for the COVID-19 data set are presented in Table 15. As the target variable was imputed, both the RMSE with the imputed values and with only previously known values (Real RMSE) are presented. For the stochastic methods, the results presented by the column Mean RMSE are the mean evaluation metric for the 30 rounds, along with its standard deviation. If the standard deviation is not presented, it is because the method is deterministic and was executed only once. The column Best RMSE presents the best result achieved on the 30 rounds. Finally, the imputation methods are ranked using the mean and the best results.

| Imputation Method\ Data set | Mean RMSE | Mean True RMSE | Best RMSE | Best True RMSE | Rank Mean RMSE | Rank Best RMSE |
|---------------------------------|-----------------|-------------------|-----------|-------------------|-------------------|-------------------|
| KNN | 4959.24 | 8048.64 | 4959.24 | 8048.64 | 3 | 3 |
| Mean | 7686.72 | 13351.19 | 7686.72 | 13351.19 | 6 | 6 |
| Missing Indicators | 7796.75 | 13542.29 | 7796.75 | 13542.29 | 7 | 7 |
| Regression | 18983.71 | 13346.56 | 18983.71 | 13346.56 | 8 | 8 |
| Pattern | 7684.22 ± 0 | 13345.79 ± 0 | 7684.22 | 13345.79 | 5 | 5 |
| Pattern & Kalman | 3470.46 ± 0 | 13793.94 ± 0 | 0.7308 | 13793.94 | 2 | 2 |
| Decomposition | Inf | Inf | 13685.59 | 13685.59 | 9 | 9 |
| Additive Decomposition | Inf | Inf | 13202.91 | 13202.91 | 9 | 9 |
| Decomp. & Kalman | Inf | Inf | 26726.02 | 26726.02 | 9 | 9 |
| Add. Decomp & Kalman | Inf | Inf | 19176.85 | 19176.85 | 9 | 9 |
| Kalman | 6930.11 | 6930.11 | 13900.60 | 13900.60 | 4 | 4 |
| LSTM | 558.33 ± 375.82 | 13803.86 ± 60.24 | 106.60 | 13836.69 | 1 | 1 |

Table 15 – Comparison of the mean RMSE scores and best RMSE scores for each imputed data set and ranking.

The best results were achieved by the data sets imputed using LSTM, Pattern and Kalman, and KNN, in this order, when taking into account the imputed target variable. One fact worth mentioning is that the LSTM imputation method takes approximately one and a half hours to impute each data set, while all the other methods take minutes. Regarding the real RMSE, the best results were achieved by the KNN imputed data set, followed by the Additive Decomposition imputed and Regression imputed data sets. The RMSE computed using the imputed target variable for all data sets imputed by a variation of the Imputation by Decomposition method present the Inf value because the imputed values are very small in magnitude, and when computing the RMSE the biggest float available in python is not big enough to store it. As it can be noted, the ranking did not change when considering the mean RMSE or the best RMSE.

Figures 26, 27 and 28 show the real versus predicted data for the three best results in order. As it can be noted, the first two models were not able to model the highest peaks well, but the model generated using KNN was able to detect the peaks occurrence but not its amplitude.

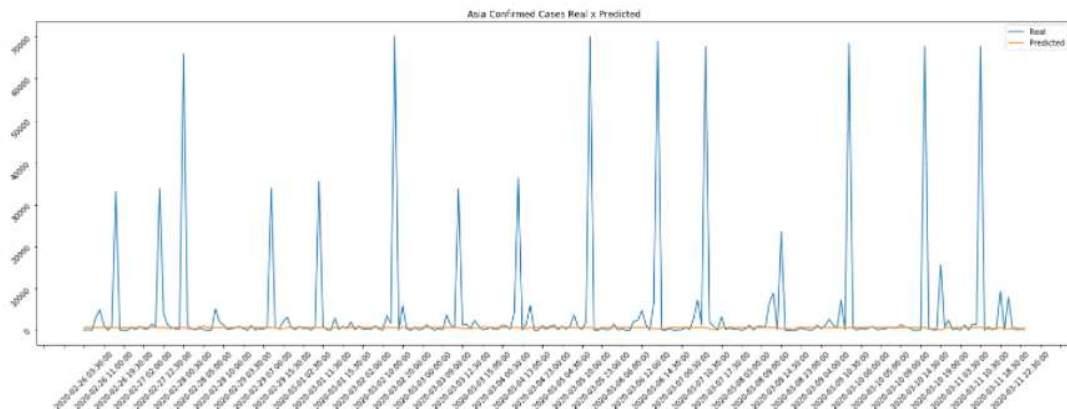


Figure 26 – Real COVID-19 data and prediction made by model generated using the LSTM imputed data set

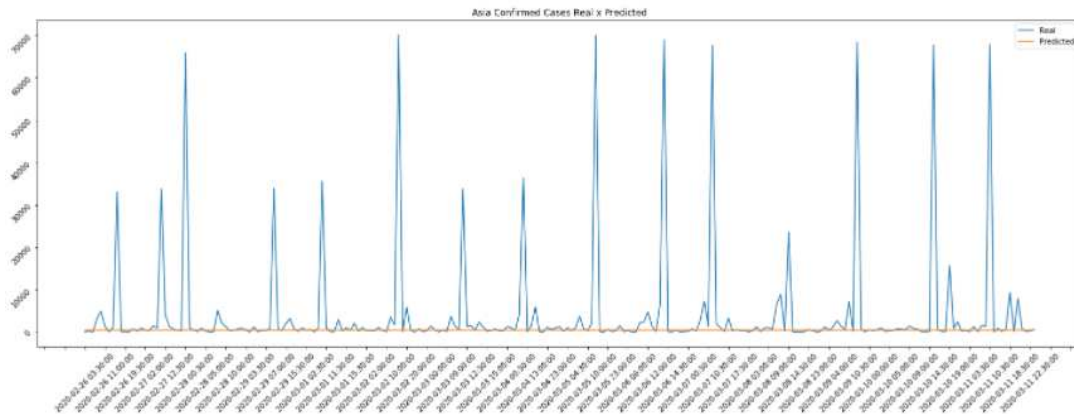


Figure 27 – Real COVID-19 data and prediction made by model generated using the Pattern & Kalman imputed data set

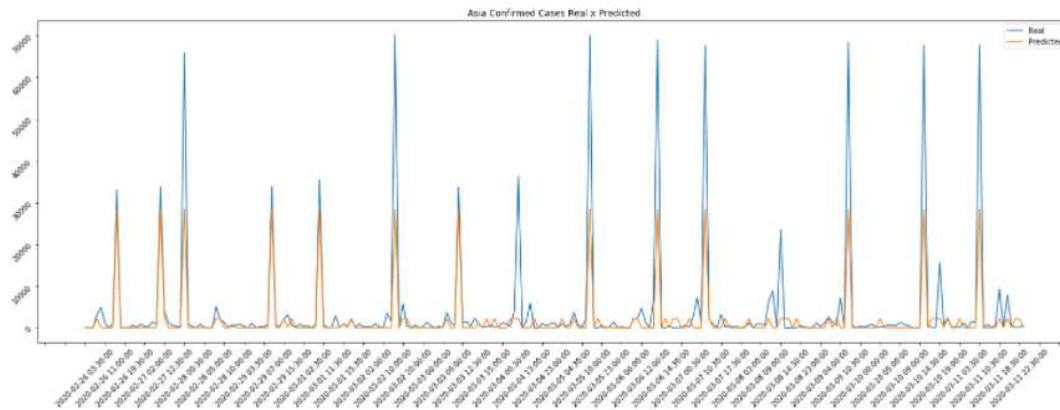


Figure 28 – Real COVID-19 data and prediction made by model generated using the KNN imputed data set

8.4 Deng

The results for the Deng data set are presented in Table 16. For the stochastic methods, the results presented by the column Mean RMSE are the mean evaluation metric for the 30 rounds, along with its standard deviation. If the standard deviation is not presented, it is because the method is deterministic and was executed only once. The column Best RMSE presents the best result achieved on the 30 rounds. Finally, the imputation methods are ranked using the mean and the best results.

The best results were achieved by Additive Decomposition plus Kalman, Kalman, and Pattern Plus Kalman, in this order. However, the best result achieved by the Pattern and Kalman sets was superior to all other results, having reach an RMSE of 26.40.

Figures 29, 30 and 31 show the real versus predicted data for the three best results

| Imputation Method\ Data set | Mean RMSE | Best RMSE | Rank Mean RMSE | Rank Best RMSE |
|--------------------------------|--------------------|-----------|-------------------|-------------------|
| KNN | 28.59 | 28.59 | 5 | 6 |
| Mean | 29.66 | 29.66 | 9 | 11 |
| Missing Indicators | 27.67 | 27.67 | 4 | 5 |
| Regression | 28.62 | 28.62 | 6 | 8 |
| Pattern | 29.27 ± 0.5760 | 28.6 | 11 | 7 |
| Pattern & Kalman | 27.39 ± 0.6902 | 26.40 | 3 | 1 |
| Decomposition | 29.22 | 29.22 | 8 | 10 |
| Additive Decomposition | 29.69 | 29.69 | 10 | 12 |
| Decomp. & Kalman | 28.91 | 28.91 | 7 | 9 |
| Add. Decomp & Kalman | 26.99 | 26.99 | 1 | 3 |
| Kalman | 27.16 | 27.16 | 2 | 4 |
| LSTM | 29.84 ± 2.22 | 26.83 | 12 | 2 |

Table 16 – Comparison of the mean RMSE scores and best RMSE scores for each imputed data set and ranking.

in order. As it can be noted, none of the models was able to model the highest peaks very well.

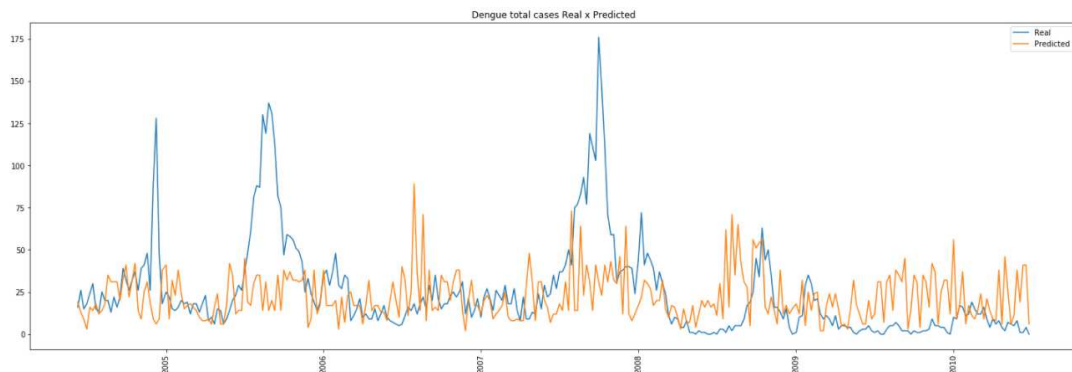


Figure 29 – Real Deng data and prediction made by model generated using the Additive Decomposition imputed data set

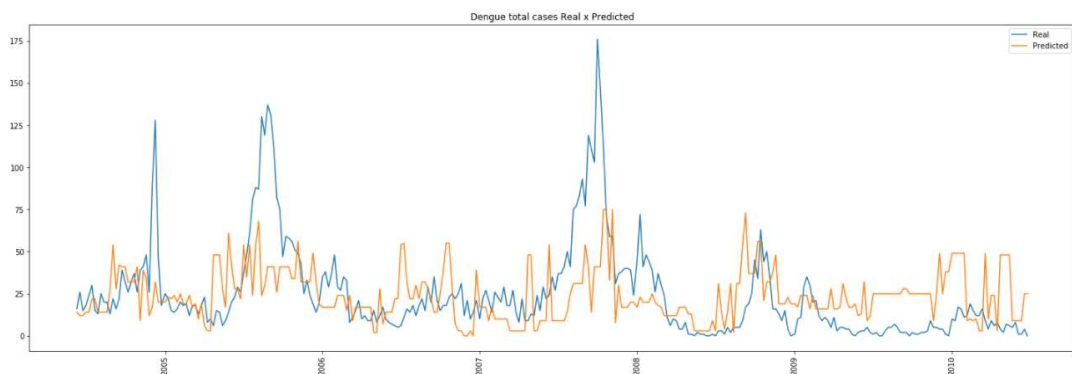


Figure 30 – Real Deng data and prediction made by model generated using the Kalman imputed data set

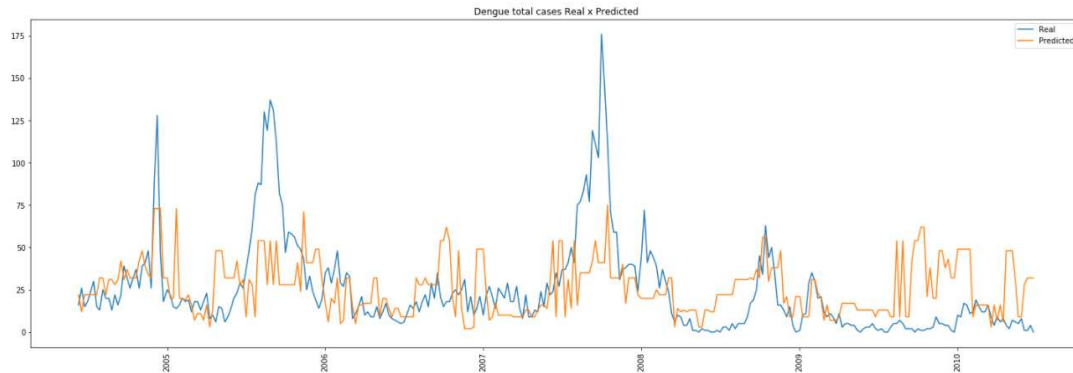


Figure 31 – Real Deng data and prediction made by model generated using the Pattern & Kalman imputed data set

8.5 Ranking

The mean ranking of the mean and best evaluation metrics were computed for each imputation method and the results are presented by Table 17. As it can be seen, on average the best results were achieved by the Kalman imputed data sets, followed by Pattern and Kalman and Missing Indicators. When considering the best results, the Pattern and Kalman data sets achieved the best ranking, followed by Kalman and Missing Indicators.

| Imputation Method\ Data set | Mean Rank | Mean Best Rank |
|--|------------------|-----------------------|
| KNN | 4 | 4 |
| Mean | 5 | 5 |
| Missing Indicators | 3 | 3 |
| Regression | 5 | 5 |
| Pattern | 8 | 6 |
| Pattern & Kalman | 2 | 1 |
| Decomposition | 9 | 8 |
| Additive Decomposition | 11 | 9 |
| Decomp. & Kalman | 7 | 7 |
| Add. Decomp & Kalman | 6 | 6 |
| Kalman | 1 | 2 |
| LSTM | 10 | 5 |

Table 17 – Comparison of the mean RMSE scores and best RMSE scores for each imputed data set and ranking.

Overall, the two new methods proposed in this work achieved good results, being very competitive when compared to the other imputation methods and adapting well for different types of time series. The Pattern and Kalman imputation method proved to be a good alternative for imputation, as well as the Kalman imputation method. The results achieved by the imputation by Decomposition method were good, but it is necessary to test if only Additive decomposition should be done, or if both Additive and Multiplicative decomposition will work best for each particular data set. The combination of Decomposition plus the Kalman filter should also be experimented with.

Chapter 9

Conclusion

This work explored the problem of missing data in time series. It was observed that literature recommended methods work very well for well-behaved data. However, when trying to impute a real data set with mixed types of time series and/or different types of loss mechanisms it is very important to try several different methods and choose the one that fits the problem better.

Additionally, when dealing with missing data in time series, choosing an appropriate imputation method can have great impact on the final results. The classical imputation methods, although easy to implement and comprehend, have limited success for some time series. However, some methods such as Kalman and Pattern and Kalman imputation methods have shown to achieve good results for the different real-world data time series used as case-studies in this work. This might indicate their ability to also work well for different real-world data. The imputation by Decomposition method works well, but its combinations (Decomposition plus Kalman, Additive Decomposition, and Additive Decomposition plus Kalman) should be tested to discover which works best for each particular data set. Although the LSTM-based imputation method is promising due to its inherent properties for handling sequences, it has shown to be complex to implement. It also depends on the hyperparameters to achieve satisfactory results and the results vary significantly between executions. It is also important to note that the LSTM imputation method takes approximately one and a half hours to impute each data set, while all the other methods take minutes.

The author believes that the main findings of the study are:

- reinforcing the importance of analyzing and categorizing the set of time series data before imputation. The classical theory of time series provides the mechanisms and tools for such analysis. Coupled with the literature-recommended imputation methods, this strategy can yield good results compared to state-of-the-art more complex imputation methods;

- Real-world time series usually come with noise. Therefore, the use of smoothing mechanisms such as the Kalman Filter to filter noise during imputation can also lead to better results.

The literature-recommended approach to choosing the imputation method for time series data, consists on finding out some of its characteristics such as its pattern and missing mechanisms. It seems only logical that separating the data by pattern and then using the literature-recommended imputation methods to impute each group will achieve good results. Filtering noisy data is also a well-known approach to improve predictions, hence using a filter to, not only smooth the data, but also impute it, seemed like a good strategy. The imputation by Decomposition idea was based on the fact that, if the imputation by Pattern produced good results, imputing each component of a time series with its literature-recommended method could result in a more assertive imputation as a whole.

This work produced a library to separate time series by its pattern, in which the user does not have to input any information regarding the time series (such as period or frequency). No visual analysis of the data has to be done to evaluate its pattern and the user does not need worry about which imputation method works best for each pattern. Four other separation methods were used to try to separate time series by pattern: Kolmogorov-Smirnov Clustering, FTCA, Agglomerative Clustering and K-means. The goal was to experiment a data-driven approach to achieve time series clustering, compared to the heuristics presented. The synthetic time series generated by Algorithm 6 were used to test the four algorithms but their performances were nowhere near as good as the performance of the separation accomplished by the algorithm 5.

The imputation by Decomposition method is also available for users of the library and, as noted, achieved accurate results, specially when some time was taken to investigate which of its variants worked best for the data set in question.

The opportunities to continue this work include: investigating the effects of the methods in a real problem, but with all data present. This way, the removal of the samples for testing can be controlled, its rate of missingness can be varied, but the data set would still represent a real-word scenario; investigate how a irregular sampling of data would have affected the synthetic data experiment; use the proposed imputation methods to try to detect outliers. This could be done by assuming that each time step was missing, one at a time, and imputing it. Then, the imputed value is compared to the real value, and if the values are too different (compared to a threshold) the time step could be considered an outlier.

References

- A. Abraham. Artificial neural networks. *Handbook of measuring system design*, 2005.
- L. A. Aguirre. *Introdução à identificação de sistemas—Técnicas lineares e não-lineares aplicadas a sistemas reais*. Editora UFMG, 2004.
- M. V. S. G. d. Amaral. *Ajuste de modelos e comparação de séries temporais para dados de vazão específica em microbacias pareadas*. PhD thesis, Universidade de São Paulo, 2014.
- R. R. Andridge and R. J. Little. A review of hot deck imputation for survey non-response. *International statistical review*, 78(1):40–64, 2010.
- M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf. Multiple imputation by chained equations: what is it and how does it work? *International Journal of Methods in Psychiatric Research*, 20(1):40–49, 2011.
- S. R. Baker, N. Bloom, S. J. Davis, and K. J. Kost. Policy news and stock market volatility. Technical report, National Bureau of Economic Research, 2019.
- G. Bishop and G. Welch. An introduction to the kalman filter. *Proc of SIGGRAPH, Course*, 8(27599-23175):41, 2001.
- D. Caldara and M. Iacoviello. Measuring geopolitical risk. *FRB International Finance Discussion Paper*, (1222), 2018.
- D. Caldara, M. Iacoviello, P. Molligo, A. Prestipino, and A. Raffo. The economic effects of trade policy uncertainty. *Journal of Monetary Economics*, 109:38–59, 2020.
- J. Chen. *Studying Regime Change using Directional Change*. PhD thesis, University of Essex, 2019.
- J. Chen and E. P. Tsang. Classification of normal and abnormal regimes in financial markets. *Algorithms*, 11(12):202, 2018.
- S. Cheng and F. Lu. A two-step method for missing spatio-temporal data reconstruction. *ISPRS International Journal of Geo-Information*, 6(7):187, 2017.

- M. A. S. Curado, J. Teles, and J. Marôco. Analysis of variables that are not directly observable: influence on decision-making during the research process. *Revista da Escola de Enfermagem da USP*, 48(1):146–152, 2014.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.
- DOMO. Data never sleeps 5.0. Eletronically published, 2017.
- M. Duma, T. Marwala, B. Twala, and F. Nelwamondo. Partial imputation of unseen records to improve classification using a hybrid multi-layered artificial immune system and genetic algorithm. *Applied Soft Computing*, 13(12):4461–4480, 2013.
- J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
- K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017. doi: 10.1109/TNNLS.2016.2582924.
- R. H. Groenwold, I. R. White, A. R. T. Donders, J. R. Carpenter, D. G. Altman, and K. G. Moons. Missing covariate data in clinical research: when and when not to use the missing-indicator method for analysis. *Cmaj - Canadian Medical Association Journal*, 184(11):1265–1269, 2012.
- Y. Hang, S. Fong, and W. Chen. Aerial root classifiers for predicting missing values in data stream decision tree classification. In *Proceedings of the SIAM International Conference on Data Mining (SDM'11)*, pages 1–10, 2011.
- R. A. Haugen. *Modern investment theory*. Prentice Hall, 1986.
- R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.
- G. Kalton and L. Kish. Some efficient random imputation methods. *Communications in Statistics-Theory and Methods*, 13(16):1919–1939, 1984.
- Y.-J. Kim and M. Chi. Temporal belief memory: Imputing missing data during rnn training. In *In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-2018)*, 2018.
- M. Kuhn and K. Johnson. *Feature engineering and selection: A practical approach for predictive models*. CRC Press, 2019.

- R. J. Little and D. B. Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.
- Y. Luo, X. Cai, Y. Zhang, J. Xu, and X. Yuan. Multivariate time series imputation with generative adversarial networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 1603–1614, 2018.
- S. Mara Ribeiro and C. L. de Castro. Time series imputation by nature and by decomposition. *Waiting for approval*.
- S. Mara Ribeiro and C. L. de Castro. Missing data in time series: A review of imputation methods and case study. *Learning and Nonlinear Models - Revista Da Sociedade Brasileira De Redes Neurais - Special Issue: Time Series Analysis and Forecasting Using Computational Intelligence*, 19(2), 2021.
- F. J. Molnar, B. Hutton, and D. Fergusson. Does analysis using “last observation carried forward” introduce bias in dementia research? *Cmaj - Canadian Medical Association Journal*, 179(8):751–753, 2008.
- I. Pratama, A. E. Permanasari, I. Ardiyanto, and R. Indrayani. A review of missing values handling methods on time-series data. In *2016 International Conference on Information Technology Systems and Innovation (ICITSI)*, pages 1–6. IEEE, 2016.
- K. Rantou. Missing data in time series and imputation methods. *University of the Aegean, Samos*, 2017.
- M. Saad, M. Chaudhary, F. Karray, and V. Gaudet. Machine learning based approaches for imputation in time series data and their impact on forecasting. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2621–2627. IEEE, 2020a.
- M. Saad, L. Nassar, F. Karray, and V. Gaudet. Tackling imputation across time series models using deep learning and ensemble learning. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3084–3090. IEEE, 2020b.
- S. Siami-Namini and A. S. Namin. Forecasting economics and financial time series: Arima vs. lstm. *arXiv preprint arXiv:1803.06386*, 2018.
- W. Wijesekara and L. Liyanage. Comparison of imputation methods for missing values in air pollution data: Case study on sydney air quality index. In *Future of Information and Communication Conference*, pages 257–269. Springer, 2020.
- P. Wu, L. Xu, and Z. Huang. Imputation methods used in missing traffic data: a literature review. In *International Symposium on Intelligence Computation and Applications*, pages 662–677. Springer, 2019.

-
- J. Yoon, J. Jordon, and M. Schaar. Gain: Missing data imputation using generative adversarial nets. In *International Conference on Machine Learning*, pages 5689–5698. PMLR, 2018.
- C. Yozgatligil, S. Aslan, C. Iyigun, and I. Batmaz. Comparison of missing value imputation methods in time series: the case of turkish meteorological data. *Theoretical and Applied Climatology*, 112(1):143–167, 2013.