RAFAEL MOTTA RAPP

Grid search approach to select and calibrate exponential smoothing, SARIMA and LSTM models for demand forecasting.

São Paulo
2023

RAFAEL MOTTA RAPP

Grid search approach to select and calibrate exponential smoothing, SARIMA and LSTM models for demand forecasting.

Corrected version

Dissertation presented to the Polytechnic School of the University of São Paulo to obtain the title of Master of Science.

Concentration Area:
Logistics Systems Engineering

Advisor:
Prof. Dr. Daniel de Oliveira Mota

São Paulo
2023

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 5 de Fevereiro de 2024

Assinatura do autor: Rafael Motta Rapp

Assinatura do orientador: D. P. de O. Mota

Catalogação-na-publicação

Rapp, Rafael Motta
Grid search approach to select and calibrate exponential smoothing, SARIMA and LSTM models for demand forecasting. / R. M. Rapp -- versão corr. -- São Paulo, 2024.
151 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Análise de séries temporais 2.Modelos em séries temporais 3.Previsão (análise de séries temporais) 4.Redes neurais 5.Aprendizagem profunda I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Name: Rapp, Rafael Motta

Title: Grid search approach to select and calibrate exponential smoothing, SARIMA and LSTM models for demand forecasting.

Dissertation presented to the Polytechnic School of the University of São Paulo to obtain the title of Master of Science.

Approved in: 15/12/2023

Examination Board

Prof. Dr.	PROF. DR. Daniel de Oliveira Mota
Institution:	Universidade de São Paulo - USP
Judgment:	Approved
Prof. Dr.	PROF. DR. Roberto Ivo da Rocha Lima Filho
Institution:	Universidade Federal do Rio de Janeiro - UFRJ
Judgment:	Approved
Prof. Dr.	PROF. DR. Fernando Tobal Berssaneti
Institution:	Universidade de São Paulo - USP
Judgment	Approved

ACKNOWLEDGEMENT

To my family and especially to my lovely wife, Cristiane, who was always patient and supported me on my journey in search of knowledge.

To professor Dr. Daniel de Oliveira Mota, for the guidance and improvement of my work.

To the Department of Logistics Systems Engineering of the Polytechnic School of USP (Mlog-USP) for making the development of this study possible.

To HP Inc., by providing support in the challenges encountered in my professional journey and allowed me to carry out this work.

To my friends, teachers, and assistants at the office of the Logistics Systems Engineering Course at the USP Polytechnic School, who contributed directly or indirectly to this work.

Abstract

RAPP, R. M. **Grid search approach to select and calibrate exponential smoothing, SARIMA and LSTM models for demand forecasting**. 2023. Dissertação (Mestre em Ciências) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2023.

Demand forecasting accuracy allows the possibility to better allocate and plan resources improving sales and operations planning (S&OP). This work proposes a process to calibrate and select models for time series forecasting. Exponential smoothing, SARIMA and deep learning LSTM forecasting models were selected based on its wide use in the scientific community. First these models are defined based on their hyperparameters. The domain of a hyperparameter can be real-valued, integer valued, binary or categorical. Which values to choose is critical since it will define the final functional form of the forecasting equations, including the final number of parameters that needs to be determined. Then, for each model, the hyperparameters bounded domain are defined and a list of vectors is built. Each vector is applied to the models to find the one with best performance, what is defined as a grid search process. Before the process is deployed, a discussion of pre-processing the data is presented which includes fill in missing values, defining time series periodicity and decomposition of the time series in trend/seasonality to be use as additional regressor for the models. To defined how each hyperparameter vector performed and how each model compares, a multi-criteria performance index is proposed. Finally, the results are discussed comparing all three models and what were the best results found. The diagnostic of the fitted model shows opportunities to address data pre-processing not here considered such as, transformation and/or a robust outlier's treatment for the time series. Nevertheless, the out of sample results shows the gain in prediction performance comparing to previous methods used and the process here proposed could help practitioners reduce complexity when implementing such models in a business environment.

Keywords: Time series analysis; demand forecast; neural network; ARIMA; LSTM, time series forecasting performance, error metrics.

Resumo

RAPP, R. M. **Seleção e calibração dos modelos de suavização exponencial, SARIMA e LSTM para previsão de demanda através de busca em rede**. 2023. Dissertação (Mestre em Ciências) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2023.

A precisão das previsões de demanda permite alocar e planejar melhor os recursos, melhorando o planejamento de vendas e operações. Este trabalho propõe um processo de calibração e seleção de modelos para previsão de séries temporais aplicados em uma empresa de eletroeletrônicos. Os modelos de previsão de suavização exponencial, SARIMA e aprendizagem profunda LSTM foram selecionados com base em sua ampla utilização na comunidade científica. Primeiro, esses modelos são definidos com base em seus hiperparâmetros. O domínio de um hiperparâmetro pode ser de valor real, de valor inteiro, binário ou categórico. Quais valores escolher é crítico, pois definirão a forma funcional final das equações de previsão, incluindo o número final de parâmetros que precisam ser determinados. Em seguida, os domínios limitados dos hiperparâmetros são definidos e uma lista de vetores são construídos. Cada vetor é aplicado aos modelos para encontrar aquele com melhor desempenho, um processo definido por busca em rede. Antes do processo ser implantado, é apresentada uma discussão sobre o pré-processamento dos dados. As etapas utilizadas neste trabalho incluem preenchimento de valores faltantes, definição de periodicidade da série temporal, decomposição da série temporal em tendência/sazonalidade para ser usada como um regressor adidiconal aos modelos. Para definir o desempenho de cada vetor de hiperparâmetros e como cada modelo se compara, métricas de erro são definidas, e um índice de desempenho multicritério é proposto. Por fim, os resultados são discutidos comparando os três modelos com os melhores resultados encontrado. O diagnóstico do modelo ajustado mostra oportunidades no pré-processamento dos dados aqui não considerados, como transformação e/ou tratamento robusto de outliers para a série temporal. No entanto, os resultados das previsões mostram o ganho no desempenho em comparação com os métodos utilizados anteriormente e o processo aqui proposto pode ajudar os profissionais a reduzir a complexidade ao implementar tais modelos num ambiente de negócios.

Palavras-chave: Análise de séries temporais; previsão de demanda; rede neurais; ARIMA; LSTM, desempenho de previsão em serie temporal, métrica de erro**.**

Figure List

Table List

Contents

1. Introduction

Traditionally, the demand forecasting has served as one important foundation for the production planning and supply chain. It interferes directly in activities such as sourcing, manufacturing and distribution. Besides being a technique widely used its accuracy remains a challenge for many companies due to uncertainty present in the customer behavior either in B2C and B2B environment (Syntetos *et al*. 2016; Narayanan; Sahin; Robinson, 2016; Abolghasemi *et al*., 2020). Demand forecast predicts two vital information for a business: (1) the expected demand at a given period, and (2) how much uncertainty is associated with a prediction.

According to Boone *et al*. (2019) and Kumar, Shankar and Aljohani (2020), along last years, the advances in technology and data collection systems have resulted in a huge amount of data on a wide variety of topics, incredibly fast, and at great speed, those characteristics are reported in the literature as the 4V of big data (Anagnostopoulos; Exposito, 2017) : velocity, variety, volume, and veracity. Of course, the forecast practitioners have found many benefits on this “data era” we live recently. They use data gathered from Internet of things, smart devices, many sensors managed not only by computers, but also by processing units built-in many machines, equipment, and devices. If one side of the coin such reality indicates an ocean of opportunity in terms of having data to feed forecasting mathematical models, on the other side, to reach success in a forecasting effort is at essence, the proper understanding of data used to feed the models.

Once the data handling challenge is accopmplished, the next challenge is the model selection. There are several quantitative models available in the literature to address the forecasting problems. They range from the simple (and very useful) moving average (Mentzer; Cox, 1984), passing by the exponential smoothing (considering or not trends and/or seasonality) (Holt, 1957; Winters, 196; Gardner, 1985; Taylor, 2003), the autoregressive moving average models (Box; Jenkins, 1976), until modern machine learning such as the neural networks. (Tealab, 2018; Chu; Zhang, 2003; Zhang *et al*., 2021). One question remains: what is the best technique to use to have the predictions that are more accurate and feasible to maintain?

Forecasting is often modelled in the context of time series data, which consists of values observed at discrete points in time. The available methods today for modelling time–series forecasting provide an increasing complexity if one wants to

extract all the potential from the most recent research (Parmezan; Souza; Batista, 2019). The consequence is difficulty in a business environment in deciding which forecasting model to apply and how to adopt a process to maintain it (Hlupić; Oreščanin; Petricet, 2020).

In the era of big data (Boone *et al*. (2019)) powered by the computing processing, there has been a shift in applying not only the traditional forecasting techniques but also the machine learning algorithms such as the Neural Networks (NN). Neural Networks, motivated by their data driven approximation of any linear or nonlinear function, contributed to an increased number of publications of any area in operational research according to Fildes *et al*. (2008). They formed one of the top four areas of growth in the forecasting journals in 2006 according to Crone, Hibon and Nikolopoulos (2011).

To quantify how NN models perform better in the forecasting is a discussion introduced by Crone, Hibon and Nikolopoulos (2011) and later in Makridakis, Spiliotis and Assimakopouloset (2018), through a series of competitions known as the Makridakis Competitions. The Makridakis (M) Competitions are a series of open competitions to evaluate and compare the accuracy of different forecasting models including NN. Professor Spyros Makridakis from University of Nicosia (UNIC) started these competitions to evaluate the performance of existing and new forecasting models (Makridakis; Hyndman; Petropoulos, 2020). There have been six M Competitions since 1982 (M6 started in 2022), each differing in the number of time series used, the forecasting methods experimented with and other features regarding the structure of data.

Each competition introduces new datasets and explore different forecasting techniques, so the practitioners can compare results with the pre-defined benchmarks. As computer processors became faster and inexpensive, the forecasting field expanded to include machine learning forecasting models. (Makridakis; Hyndman; Petropoulos, 2020).

In those competitions as well in the context of this work, the traditional forecasting models are referred to the exponential smoothing (ES) methods and the autoregressive moving average (ARIMA) models. These models have a pre-defined functional form with parameters that govern the predictions. For this reason, they are also called the parametric models. Machine learning methods does not have a pre-

defined functional form as this will change according to different learning features specific for each type of algorithm. Thus, they are referred to as non-parametric. How well these machine learning models perform comparing against traditional models is what these competitions explores.

An initial criticism of the results provided by the neural networks was addressed by M3 competition (Crone; Hibon; Nikolopoulos, 2011; Makridakis; Spiliotis; Assimakopouloset, 2018). The argument was lacking convincing empirical evaluations against the simpler univariate statistical methods such as the exponential smoothing methods and the auto regressive integrated moving average models. The M4 competition extended and replicated the results of the previous three competitions to identify the most accurate forecasting methods, including those based on NN as well as traditional statistical ones. According to Makridakis, Spiliotis and Assimakopouloset (2018), one of the main drivers for organizing the M4 competition is that Neural Networks forecasting are not typically evaluated rigorously against the statistical benchmarks.

The M3 and M4 Competitions identified two major faults from the assumption that the more complex the model, the better the outcome's accuracy. First, a model that best fits the data does not necessarily produce the most accurate results, a behavior known as overfitting. Sometimes overfitting the model can prevent the detection of dominant features or critical patterns in the data needed for accurate forecasts. Second, the traditional forecasting methods have a greater tolerance to overfitting the data.

On the other hand, a possible explanation for not observe better performances from neural networks is the use of univariate datasets that may not contain enough relevant data to fit, if the characteristics of time series have changed over time. Having an adequate length and additional features (multivariate), neural network models allow to capture non linearities.

A particular case of a Recurrent Neural Network (RNN) called Long Short-Term Memory (LSTM) allows to model complex sequences of data providing them with adequate length to capture long-term information relationships. Nevertheless, the neural networks in general can be criticized for their black-box nature (Makridakis; Spiliotis; Assimakopouloset, 2018).

Table 1 is an extract from Makridakis; Spiliotis; Assimakopoulos (2020) which provides a summary of the ten benchmarks as well as two additional models used as standards for comparison used in the M4 competition: the ETS or the exponential smoothing methods and the autoregressive moving average (ARIMA) models. The goal in Table 1 is to select forecasting models for this study that is being used in the scientific community.

Table 1 - The benchmarks and standards for comparison of the M4 Competition.

Methods	**Description**
Naïve 1	A random walk model, assuming that future values will be the same as that of the last known observation.
Naïve S	Forecasts are equal to the last known observation of the same period.
Naïve 2	Like Naïve 1 but the data are seasonally adjusted, if needed, by applying a classical multiplicative decomposition.
SES	Exponentially smoothing the data and extrapolating assuming no trend. Seasonal adjustments are considered as per Naïve 2.
Holt	Exponentially smoothing the data and extrapolating assuming a linear trend. Seasonal adjustments are considered as per Naïve 2.
Damped	Exponentially smoothing the data and extrapolating assuming a damped trend. Seasonal adjustments are considered as per Naïve 2.
Theta	The Theta method works by fitting two lines to the time series. A trend line, which captures the long-term direction of the series and a seasonal line, which captures the short-term fluctuations of the series. In M3 Competition first line being extrapolated using linear regression and the second one using SES. The forecasts are then combined using equal weights. Seasonal adjustments are considered as per Naïve 2.
Comb	The simple arithmetic average of SES, Holt and Damped exponential smoothing (used as the single benchmark for evaluating all other methods).
MLP	Multilayer Perceptron. A perceptron of a very basic architecture and parameterization. Some preprocessing like detrending and deseasonalization is applied beforehand to facilitate extrapolation.
RNN	A recurrent network of a very basic architecture and parameterization. Some preprocessing like detrending and deseasonalization is applied beforehand to facilitate extrapolation.

Cont'd Table 1 - The benchmarks and standards for comparison of the M4 Competition.

Methods	Description
ETS	Exponential smoothing. Automatically provides the best exponential smoothing model, indicated through information criteria.
ARIMA	An automatic selection of possible ARIMA models is performed and the best one is chosen using appropriate selection criteria.

Source: Makridakis; Spiliotis; Assimakopoulos, 2020

From Table 1, Naïve 1, Naïve S, Naïve 2, SES, Holt, Damped, Theta and Comb represents models for Statistical benchmark. MLP and RNN are the neural network models representing the machine learning benchmarks and finally the standard for comparison are the Exponential smoothing (ETS) and ARIMA.

Although the results from the M4 forecasting competition have clearly shown the potential of NNs and RNNs (Makridakis; Spiliotis; Assimakopoulos, 2020), the exponential smoothing and ARIMA have traditionally supported forecasting in a univariate context not only from their accuracy but also being relatively simple to implement comparing to complex RNN models. Regardless of the recent successes of RNNs in forecasting, the practitioner may still be reluctant to try RNNs as an alternative since they may not have the expert knowledge to achieve satisfactory accuracy. Additionally, no established guidelines exist as to when traditional statistical methods will outperform RNNs, which architecture should be used or how their parameters should be tuned for improving the forecasting accuracy.

Even for traditional forecasting methods, the implementation does become a challenge when involves multiple time series. Export knowledge is key, starting with the identification of potential issues with the series (data preprocessing), moving to which model to choose, how to calibrate its parameters and finally how to the measure performance, making this whole process lengthy and complex.

Therefore, adopting an optimized processual approach that can be implemented and measure performance between forecasting methods can help companies and practitioners explore the different type of models available in the literature and easy scale across different product lineups.

This work is based on a database of consumer and corporate electronics products for Latin America, representing over 700 B2C and B2B clients in the past 5 years. This database forms a time series of daily sales observations and other time

dependent variables. This multivariate database with such length presents a natural choice to apply a Recurrent Neural Network (RNN) model to evaluate how well can this model capture long-term information relationships. Also, according to Table 1, models such as Exponential smoothing methods - ETS from Table 1 which is referred as ES) and the autoregressive integrated moving average ARIMA are used in those competition as benchmarks to evaluate other methods. These 3 models are therefore here implemented following an optimized process that pre-process the data, calibrate the models and measure performance. The model with best performance is chosen to forecast the future sales.

After choosing the models, the next step is understanding how to calibrate it. According to Bisch *et al*. (2021) and Yang and Shami (2020) the model parametrization is challenging and search algorithms for model calibration has attracted the attention of the scientific community. To review these algorithms, the difference between parameters and hyperparameters of a forecasting model is introduced. Searching algorithms works to select the hyperparameters first, so the resulting functional form of the prediction equation can be defined. After, the parameters of this function are found by minimizing an objective function. This definition is explained in Chapter 2.

The idea of optimizing models by an automatically searching the best hyperparameters was shared by Bergstra and Bengio (2012) who introduced the Random Search algorithm to compare with the Grid Search, then a more complex iterative procedure called Bayesian search was discussed by Gelbart, Snoek and Adams (2014) and Bischl *et al*. (2017). In addition, Luo (2016) gave an overview of the search strategies and Probst, Boulesteix and Bischl (2019) who compared the impact of choosing the hyperparameters (which he refers as tuning) either jointly, tuning individual parameters or combinations.

This study is different from Thuyen *et al*. (2021), ArunKuma *et al*. (2020), Abbasimehr, Shabani and Yousefi (2020), Hewamalage, Bergmeir and Bandara (2021) studies which used the grid search technique without defining optimal boundaries for the hyperparameters search and Parmesan, Souza and Batista (2019), in how to define the hyperparameters for the LSTM case.

To conclude, the objective of this work is to propose a process to calibrate and select models for time series forecasting using grid search of the hyperparameters.

This method aims to be applied in two different forecasting model types: (1) parametric statistical forecasting (Exponential smoothing and SARIMA); and (2) non-parametric (LSTM model), using data resampling and error performance techniques. Based on the calibration delivered by the method, the forecast practitioners should be able to select the most suitable model to be used in the demand context where the observed values are discrete absolute values in time. It is also observed the gains obtained with this methodology base on previous methods used.

2. Literature Review

This chapter introduces a brief review of the fundamentals involved in modelling observations collected at fixed time intervals. It includes:

a) The dataset definition;
b) The modelling of the function form of the prediction;
c) How to partition the dataset into fitting and evaluation;
d) Time Series characteristics;
e) The summarized fundamentals of the three forecasting models chosen;
f) The search algorithm to find and evaluate the final functional form.

Forecasting in general is concerned in estimating the future values based on historical data. Inside a company it provides basis to coordinate numerous activities such as short-term daily operations planning and long-term strategic decision making, seeking to minimize risks and maximize utilities. Examples includes sales forecasting that drive the production planning, the raw material sourcing, the production capacity planning, the equipment failure prediction by looking past data behavior, the product use data that can forecast new trends, develop new products and/or enhanced software interfaces, etc.

To address real-life challenges, forecasting has been evolving in last 20 years by leveraging advances in computing and data availability, enabling the analysis of larger and more complex datasets in analytics and data science.

Forecasting methods can be divided in two groups. Qualitative Methods are based on emotions, intuitions, judgments, personal experiences, and opinions. There is no mathematical modelling involved and some examples of this includes Delphi Method (process framework wherein the main objective is to arrive at a group consensus), Market Survey, Executive Opinion, Sales Force expectation to be able to sell, etc. Quantitative Methods depend on defining a functional form with parameters that, when calibrated, can be used to forecast the future values. This functional form is defined by variables that are measured sequentially in time, at a fixed interval, known as the sampling interval. The resulting data form a time series.

The main characteristics of many time series are the trends and seasonal variations that can be modelled deterministically with mathematical functions of time.

Another important feature is that observations which are close together in time tend to be correlated (serially dependent). Much of the methodology in a time series analysis is aimed at explaining this correlation through statistical models, descriptive methods and more recent non-linear data driven models. Once a model is fitted to data and the model shows goodness of fit, it can be used to forecast the future values to guide the planning decisions. If a desirable goodness of fit is not obtained, another model could be evaluated until a satisfactory result is reached.

The time steps are the main regressor used to fit the forecasting models. When additional variables are presented in the dataset that follows the same time steps, those variables can also be used as additional regressors to the model and observe if the goodness of fit is improved. The latter has been more the case recently due to the growth of data availability. For instance, the forecasting variable could be correlated with other macro-economic variables, the weather temperature, costumer behavior, company financials, etc.

When considering additional regressors to the model, it is crucial to answer one question: Will this series be available for the forecasting period? to answer this let´s define:

a. Target variable: the time series to forecast. In data science models this is also known as response.
b. Features: time series from other sources that can be used to improve the forecasting model.

Table 2 - Time series dataset with multiple features

Time	**Feature1**	**Feature2**	**Target Variable**
T1	X_1^1	X_1^2	Y_1
T2	X_2^1	X_2^2	Y_2
T3	X_3^1	X_3^2	Y_3
T4	X_4^1	X_4^2	?

Source: Elaborated by the author

In Table 2 there is a representation of the multivariate case. The goal is to forecast target variable at T4. When modelling with multivariate series, it will be required to know what those values are when predicting the target variable. Defining X_t^i as t the regressor time step and i-th regressor, in the example above, X_4^1 and X_4^2 must be known. Different values at T4 for feature 1 and 2 could be used if the objective is to simulate different forecasting scenarios. The time and features columns can also be called as auxiliary variables.

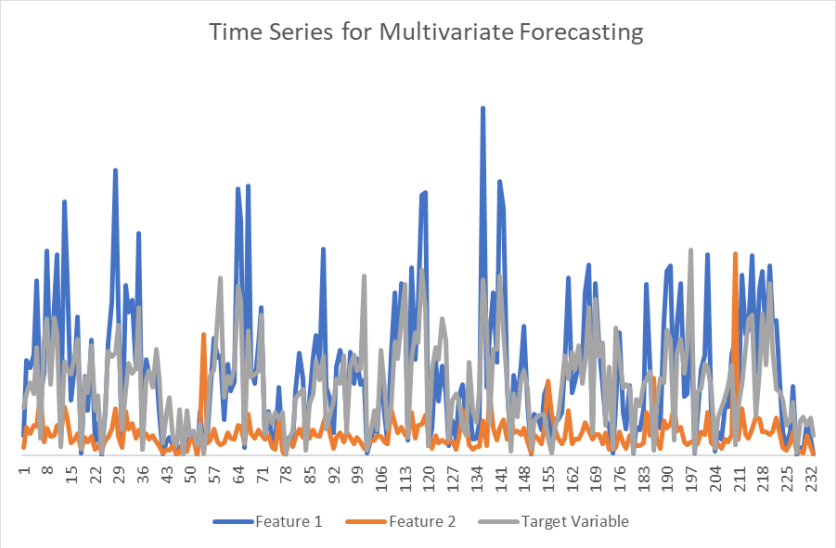


Figure 1 - Example extracted from the time series dataset.

Source: Elaborated by the Author

Figure 1 shows graphically the time series plot of the target variable, features 1 and 2. Selecting a model that can capture the behavior of feature 1 and feature 2 through time could improve the prediction results of the target variable. This is an extract from the dataset from this work, and it is not limited to two features.

Prior to applying a forecasting method, the data may require pre-processing such as checking for outlier and missing values. Other manipulations might precede the application of the forecasting method or be incorporated into the models themselves. Some forecasting methods require de-seasonalized the time series, while others address seasonality within the method. Assumptions of normality and non-constant variance could trigger transformations of the target variable which may improve forecasting results in statistical models (James *et al*., 2013).

With the dataset ready, it is time to choose the models. Relationships among features and target variable could be linear or involve nonlinear structures. When an

explicit functional form is not available, methodologies such as simulations, ensemble learning techniques or artificial neural networks might be employed.

Finally, it is important to evaluate the effectiveness of a forecasting method. A set of performance measures was used in this work that measure the absolute error, relative error to a benchmark and if the increase and/or decrease in the forecast levels has been captured. Later a combination of those metrics is done to choose the best model.

2.1. Modelling the functional form of the predictions

The goal in Table 2 is to find the quantitative response of Y_t for the predictors time and features. Assuming X_t^i represent the time series of feature "i" out of n total features and total time steps T, a function f that represent the relationship between Y and X is given by:

$$X_t^i = \left(X_1^i,\ X_2^i, \dots, X_t^i\right) \qquad 1$$

$$Y_t = f\left(X_t^i, Y_{t-j}\right) + \varepsilon \qquad 2$$

where i = 1,…n, t = 1,…T, ε is a random error term associated with f and $j < t \leq T$.

To predict new values of $\hat{Y}_t$, $\hat{f}$ is estimated such that:

$$\hat{Y}_t = \hat{f}\left(X_t^i, Y_{t-j}\right) \qquad 3$$

where $\hat{f}$ represents the estimate for f that minimize the error ε and $\hat{Y}$ represents the resulting prediction for Y.

The accuracy of $\hat{Y}$ as a prediction for Y depends on two quantities, which is defined as reducible and irreducible error (James *et al*., 2013). Accuracy of $\hat{f}$ can be improved by using the most appropriate statistical learning technique to estimate f and that is the reducible error. However, the variability associated with it also affects the accuracy of the predictions from the fact that $\boldsymbol{X_t^i}$ and $\boldsymbol{Y_{t-j}}$ doesn't completely determine $\boldsymbol{Y_t}$ and that is the irreducible error. There are variables outside of $\boldsymbol{X_t^i}$ and $\boldsymbol{Y_{t-j}}$ and independent of them that still have effect on $\boldsymbol{Y_t}$. The only way to improve irreducible error is to identify these outside influences and incorporate them as predictors.

So, by selecting the best modelling technique to estimate f among several candidates available, the reducible error can be improved and by finding additional features $\boldsymbol{X_t^i}$ that has effects on $\boldsymbol{Y_{t-j}}$, the irreducible error can also be improved. Nevertheless, only the reducible error can be measured.

The functional form f can be estimated through parametric and non-parametric models. Parametric Models reduces the problem of estimating f down to estimating a set of parameters. Taking for example the parameters such as $\beta_0, \beta_1, \ldots, \beta_n$ of a linear model:

$$\hat{Y}_t = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \cdots + \beta_k x_{k,t} \qquad 4$$

Exponential smoothing methods and ARIMA are classified as parametric models because assumes f as a model composed by a combination of parameters such as in (4).

After the final functional form is defined, the parameters that follows the regressors (in the example of a multiple leaner regression this would be the $\beta_0, \beta_1, \ldots, \beta_n$) needs to be determined. The most common approach to fit the model is referred to as least squares. This approach minimizes the least squares errors (LSE) that is obtained by (Cowpertwait; Metcalfe, 2009):

$$e_i = y_i - \hat{y}_i \qquad 5$$

$$LSE = \sum_i e_i^2 \qquad 6$$

The potential disadvantage of a parametric approach is that choosing a model that is too far from the true f, will lead to a poor estimation. Choosing a flexible model that can fit many different possible functional forms can address this but that will require estimating a greater number of parameters. In this process of estimation, these more complex models, could lead to a phenomenon known as overfitting the data, which lead to a poor performance when predicting $\hat{Y}_t$, if a validation process is not adopted.

According to James *et al*. (2013), non-parametric methods do not make explicit assumptions about the functional form of f. Instead, they seek to estimate a f that gets as close to the data points as possible. Such approach can have a major advantage over parametric approaches: by avoiding the assumption of a particular functional form for f, they have the potential to fit a wider range of possible shapes for

f. For that, defining the characteristics that will govern these possible shapes is crucial for a good generalization. This is called the hyperparameters selection and usually they require many observations (far more than is typically needed for a parametric approach) to obtain an accurate estimate for f.

The hyperparameters concept can also be extended to the parametric models. In the example of a multivariate linear regression in (4), a hyperparameter $H_k = \beta_k x_{k,t}$ can be defined and select only the components that have more statistical significance. The same applies for time series parametric models such as exponential smoothing and ARIMA. For the context of this work, the parameters and the hyperparameters are:

a) model Hyperparameter: properties that govern the entire model training process. It determines how should a function form of f be and how the learning algorithm will execute. These hyperparameters need to be initialized before training a model. In ARIMA this is often called order of autoregressive and moving average terms (p and q) or in exponential smoothing, this will be the seasonality/trend behavior (additive or Multiplicative). For neural network models this refers to architecture network like the number of neurons, hidden layers and how the algorithm learns from the training data such as batch size, epoch size and learning rate;
b) model parameters: variable who follow the hyperparameters which a value will be assigned after the model fit, which is achieved after the optimum value of the objective function is found.

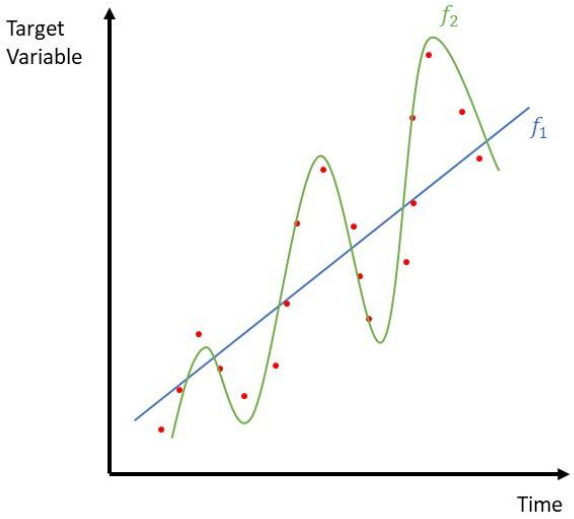


Figure 2 - Possible shapes of f.

Source: Elaborated by the author

In Figure 2, f_1 represent a less flexible model, usually a result obtaining from models with fewer hyperparameters. Although parametric models can be flexible enough to capture the behavior of the time series like f_2, usually nonparametric models could learn non linearities and generalize better the behavior of the data. This will then be observed when evaluating the effectiveness of a forecasting method.

Non-parametric, nonetheless can also suffer from overfitting if the hyperparameters are not carefully defined. This idea is discussed next by splitting the data into a training and validation partition.

2.2. Resampling data

When deploying a forecasting model, the data needs to be separated in a training set and an evaluation set. According to Berk (2016) and James *et al*. (2013), the evaluation set is used to obtain additional information about the performance of the fitted model in the training set. Such an approach may allow us to obtain information that would not be available from fitting the model using the original training sample. This splitting can be done more than once (also known as cross-validation process).

Resampling data involves repeatedly drawing samples from a training set and refitting the model of interest, evaluating its errors on each sample. Resampling approaches can be computationally expensive, because they involve fitting the same

statistical method multiple times using different subsets of the training data. However, due to recent advances in computing power, the computational requirements of resampling methods generally are not prohibitive.

For time series, resampling the data needs to follow the chronological order that the dataset is constructed, avoiding losing its characteristics as serial correlation, trend and seasonality which forecasting models are based on.

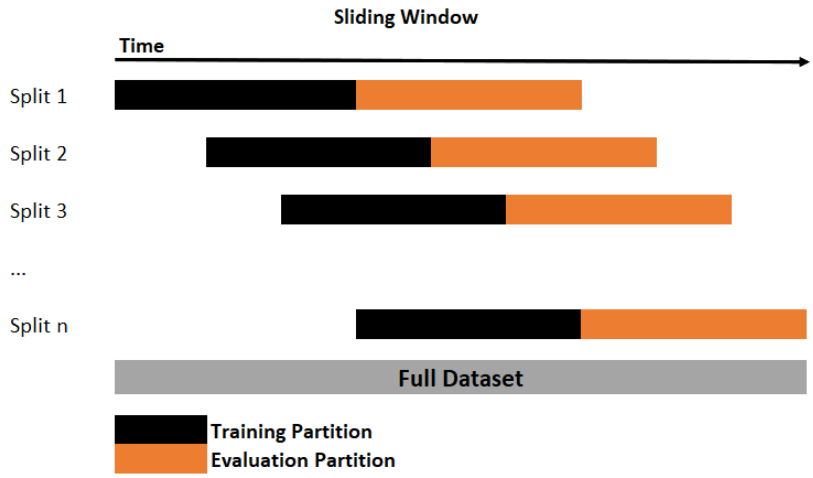


Figure 3 - Representation of cross validation for time series dataset

Source: Elaborated by the author

Figure 3 represents the cross validation in the context of time series. The training is used to fit the model, i.e., find the parameters of the final functional form. Then the Evaluation partition, also known as out of sample or test partition, is used to measure the performance of the predictions from the fitted model. Some variations of this approach exist (Faraway, 2014), as the focus now shifts to improve the evaluation partition performance.

Overfitting is when a model performs very well for training data but has a poor performance with evaluation data. This means that the training partition used contain a level of noise that fails the model in generalize the behavior of the data. Using a cross validation approach helps mitigate this issue. Other cause, as pointed out in previous section, the model is too complex (a high number of parameters) that only generalize well in the data that is being trained on.

According to Faraway (2014), when enough data is available, the problems of overfitting and model selection sometimes can be effectively addressed by the

evaluation partition. The results from the training data can be used to predict the response in the evaluation data.

For that, the evaluation partition can be further split in two, a validation and a test partition. Using the results from the training data to predict the response in the validation data. How well the predicted values correspond to the actual validation data provides feedback on performance. Finally, the test dataset is used to obtain an “honest” assessment of the whole procedure’s performance. Once a statistical learning procedure has been satisfactorily tuned using the validation dataset, there can be a proper measure of the performance in the test dataset, when comparing with more than one forecasting model.

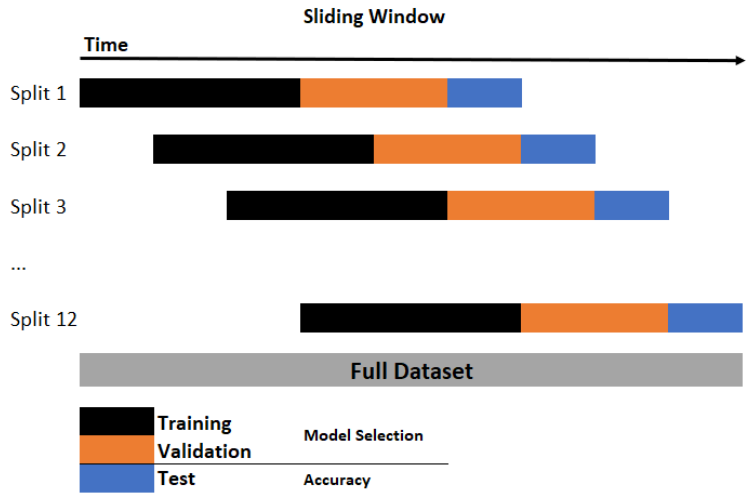


Figure 4 - Representation of cross validation for time series including an additional split to the validation partition to improve model selection.

Source: Elaborated by the author

In the methodology Chapter 3.5, the details of data resampling process seen in Figure 4 will be described. The validation partition (out of sample) will be used to select the best hyperparameters in each model. Then each models performance will be compared using the test partition.

Faraway (2014) details how data splitting provides a good option to validate a chosen model. In addition, the data split approach is only justified asymptotically, and it is not clear how large a sample has to be. In this work the approach chosen is that the test partition sizes define the size splitting definition. The less the forward-looking

prediction size, higher the opportunity to increase the cross-validation process. Section 3.5 defines how the splitting was done.

The modelling of the functional form f and data resampling procedures direct correlates with how accurate and precise the predictions will be. This is known to be the result of two competing properties in statistical learning methods called the bias and variance. James *et al*l (2013) and Berk (2016) brilliant discuss how the expect error term in a validation dataset can be described as the sum of three fundamental quantities (given that the prediction using $\hat{f}$ at point x_0 and the true value y_0):

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = Var\left(\hat{f}(x_0)\right) + \left[Bias\left(\hat{f}(x_0)\right)\right]^2 + Var(\in) \qquad 7$$

The term $E(y_0 - \hat{f}(x_0)^2)$ defines the expected validation error that is obtained if many trainings sets are drawn and tested each at x_0. $Var(\in)$ is the irreducible error previously defined and the expected validation error will never lie below this.

$\mathrm{Var}\left(\hat{\mathrm{f}}(\mathrm{x}_0)\right)$ refers to the amount by which $\hat{f}$ changes if it is estimated using different training data set. According to James *et al*. (2013), ideally the estimate for $\hat{f}$ should not vary too much between the training sets, but if the method is too sensitive it will capture noise that doesn't generalize beyond the training set (overfitting). In general, more flexible statistical methods have higher variance.

On the other hand, the bias refers to the error that is introduced by approximating a real-life problem. The true function form of $\hat{f}$ is substantially non-linear, so no matter how many training observations are given, it will not be possible to produce an equal estimate of the problem.

The Bias-Variance trade-off is the relative rate of change of these two quantities. It involves making accurate assumptions about the phenomenon under consideration and take them in consideration into the model. This will determine whether if the validation partition error increases or decreases. As the flexibility of the model is increased, the bias tends to initially decrease faster than the variance increases. Consequently, the expected validation error declines. However, at some point increasing flexibility has little impact on the bias but starts to significantly increase the variance. When this happens the validation error increases.

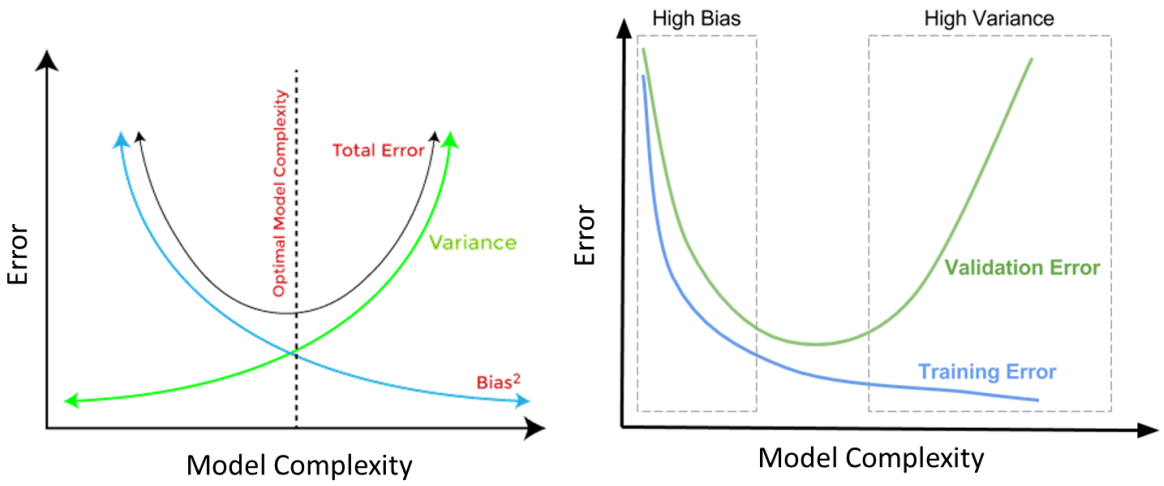


Figure 5 - The Bias-Variance trade-off representation

Source: adapted from James *et al*. (2013) and Berk (2016)

This trade-of is represented graphically in Figure 5. On the left shows the optimal point by modelling the functional form of $\hat{f}$. The training error and validation error can be seen on the right as the complexity of the model increase. To find the optimal point, this work proposes the use a resampling process using cross validation with training, validation, and test partition. The selection of the best hyperparameters will be done on the out of sample performance from the validation partition in oppose of selecting the best hyperparameters from the training partition.

2.3. Time series decomposition

A general approach to time series modelling is to plot the series and examine the main features such as:

a. Trend behaviors
b. Seasonal behaviors
c. Sharp changes in trend or seasonal behaviors
d. Any outlying observations

These characteristics are used to define the time series decomposition and the autocorrelation functions. For instance, the exponential smoothing models are a set of equations that combine linearly level, trend and seasonal components while the

autoregressive moving average model uses the autocorrelation in different time steps of the series to define the final functional form.

Exponential smoothing methods were originated in the 1950s and 1960s with the work of Brown (1959, 1963), Holt (1957) and Winters (1960). Pegels (1969) provided a simple but useful classification of the trend and the seasonal patterns depending on whether they are additive (linear) or multiplicative (nonlinear). Further steps towards putting exponential smoothing within a statistical framework were provided by Box and Jenkins (1976). However, these results did not extend to any nonlinear exponential smoothing methods.

A simple additive decomposition model is given by:

$$Y_t = M_t + S_t + Z_t \qquad 8$$

where, at time t, Y_t is the observed series, M_t is the trend, S_t is the seasonal effect, and Z_t is an error term that is, in general, a sequence of correlated random variables with mean zero.

If the seasonal effect tends to increase as the trend increases, a multiplicative model may be more appropriate:

$$Y_t = M_t S_t + Z_t \qquad 9$$

Trend gives you a general direction of the overall time series, whereas the seasonality is a regular and predictable pattern that recur at a fixed interval of time. A time series exhibits seasonality whenever there is a regular, periodic change in the mean of the series. Seasonal changes generally follow the clock and calendar -- repetitions over a day, a week, or a year are common, following behaviors surrounding dates and times.

By decomposing, the goal is to estimate the deterministic components M_t and S_t that makes the residual component Z_t stationary (Cowpertwait; Metcalfe, 2009). The statistical properties of a stationary time series like the mean and variance, do not change over time (for example the stochastic white noise is defined as mean zero and no correlation between its values at different times).

This decomposition can be used to find a satisfactory model for the process Z_t, analyses its properties, and to use it in conjunction with M_t and S_t for purposes of prediction and simulation of Y_t. The components M_t and S_t can also be used as features to improve the accuracy in nonlinear models.

Decomposition requires the frequency that the seasonal pattern repeats, which in this work is named as periodicity. It can be assigned by choosing directly from the dataset (for instance a time series composed of monthly observation can be assigned a period of 12) or calculating using the autocorrelation measure.

The following steps summarize the process to obtain a time series decomposition (Brockwell; Davis, 2016) for a stochastic seasonality i.e., the magnitude of the pattern can vary across the series:

a. trend is first estimated by applying a moving average filter specially chosen to eliminate the seasonal component and to dampen the noise. If the period d is even (d = 2q), then:

$$\widehat{m}_t = \frac{0.5y_{t-q} + y_{t-q+1} + \cdots + y_{t+q-1} + 0.5y_{t+q}}{d}, q < t < n - q \quad (10)$$

If the period is odd, d = 2q + 1, then:

$$\widehat{m}_t = \frac{0.5y_{t-q} + y_{t-q+1} + \cdots + y_{t+q-1} + 0.5y_{t+q}}{d}, q + 1 \leq t \leq n - q \quad (11)$$

b. differencing: “de-trended” series D_t is calculated as:

$$D_k = \left\{\left(y_{k+jd} - \widehat{m}_{k+jd}\right), q < k + jd \leq n - q\right\} \quad (12)$$

$$additive\ effect: D_t = Y_t - \widehat{M}_t \quad (13)$$

$$multiplicative\ effect: \mathrm{D_t} = \frac{\mathrm{Y_t}}{\widehat{\mathrm{M}}_\mathrm{t}} \quad (14)$$

c. use the de-trended series to average it over a given periodicity, compute the seasonal component S_t and extend S_t through all the dataset:

$$\hat{S}_k = D_k - d^{-1}\sum_{i=1}^{d} D_i\,, k = 1, \ldots, d\ and\ \hat{S}_k = \hat{S}_{k-d}, k > d \quad (15)$$

d. the “de-seasonalised” time series is found by subtracting the time series data with seasonal data $\widehat{D}_t = Y_t - \hat{S}_t$;

e. find the final trend line $\widehat{M}_t$ that fits the “de-seasonalised” time series data using steps a and b;

f. to complete the decomposition, the remainder, Z_t, is calculated by additive and multiplicate respectively as:

$$additive\ effect: Z_t = Y_t - \widehat{M}_t - \hat{S}_t \quad (16)$$

$$multiplicative\ effect: Z_t = \frac{Y_t}{\widehat{M}_t \hat{S}_t} \quad (17)$$

The centered moving averages in (10) and (11) are an example of a smoothing procedure that is applied retrospectively to a time series with the objective of identifying an underlying trend. Smoothing procedures can use points before and after the time at which the smoothed estimate is to be calculated. A consequence is that the smoothed series will have some points missing at the beginning and the end unless the smoothing algorithm is adapted for the end points.

A smoothing algorithm commonly used is the nonparametric method Seasonal and Trend decomposition using Loess - STL (Cleveland *et al*., 1990). This uses a locally weighted regression technique known as Lowess.

Consider $y_t = \hat{f}(x_t)$, the idea is to define a neighborhood for which a conditional mean $\bar{y}_0$ is to be computed. For each unique value of x, a nearest neighbor conditional mean for y is computed. This linear regression within each neighborhood leads to a form of smoothing based on locally weighted regressions (Lowess). Cleveland, who invented the procedure, although common refer to “local regression” and therefore the “Loess” abbreviation of locally estimated scatterplot smoothing is also used.

A formulation proposed by Berk (2016) is below. Each local regression at each x_0 is constructed by minimizing the weighted sum of squares (RSS) with respect to the intercept and slope for the t ≤ T observations included in the window. The observations in the window are represented with * below:

$$RSS^*(\beta) = (y^* - \boldsymbol{X}^*\beta)^T \boldsymbol{W}^* (y^* - \boldsymbol{X}^*\beta) \quad (18)$$

The regressor matrix $\boldsymbol{X}^*$ contain polynomial terms for the predictor, $\boldsymbol{W}^*$ is a diagonal matrix conforming to $\boldsymbol{X}^*$, with diagonal elements w_i^*, which are a function of distance from x_0, where the weighting-by-distance is done.

The overall algorithm then operates as follows:

a) Choose the smoothing parameter such as bandwidth f which is a proportion between 0 and 1,
b) Choose a point x0 and from that the (f × t = T) nearest points on x,
c) For these M nearest neighbor points, compute a weighted least squares regression line for y on x,
d) Construct the fitted value $\hat{y}_0$ for that single x_0,
e) Repeat Steps 2 through 4 for each value of x. Near the boundary values of x, constraints are imposed to increase stability,
f) Adjacent $\hat{y}_0$ are connected.

After the entire fitting process is completed, residuals are computed, and weights are constructed from these residuals. Larger residuals are given smaller weights and smaller residuals larger weights. Using these weights, the fitting process is repeated. This can be iterated until the fitted values do not change much (Cleveland 1979) or until some predetermined number of iterations is reached. The basic idea is to make observations with very large residuals less important in the fitting.

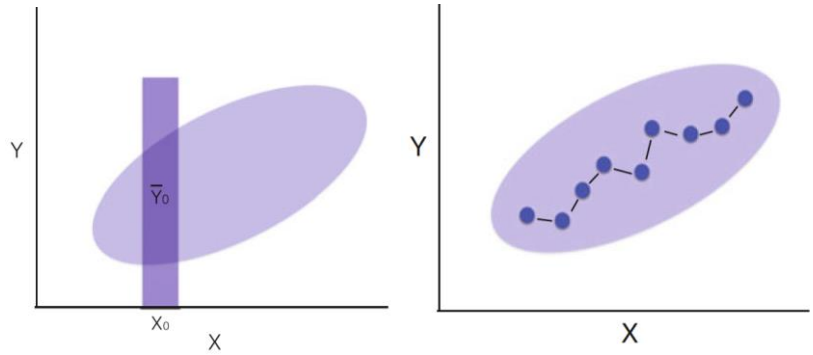


Figure 6 - Nearest Neighbor Conditional Mean on the left and the Interpolation of the conditional means on the right.

Source: Berk (2016)

Figure 6 shows how a neighborhood is defined to measure the average all the data points found in that area. This algorithm is applied using R package called “STL plus” which uses Cleveland *et al*.(1990) original publication.

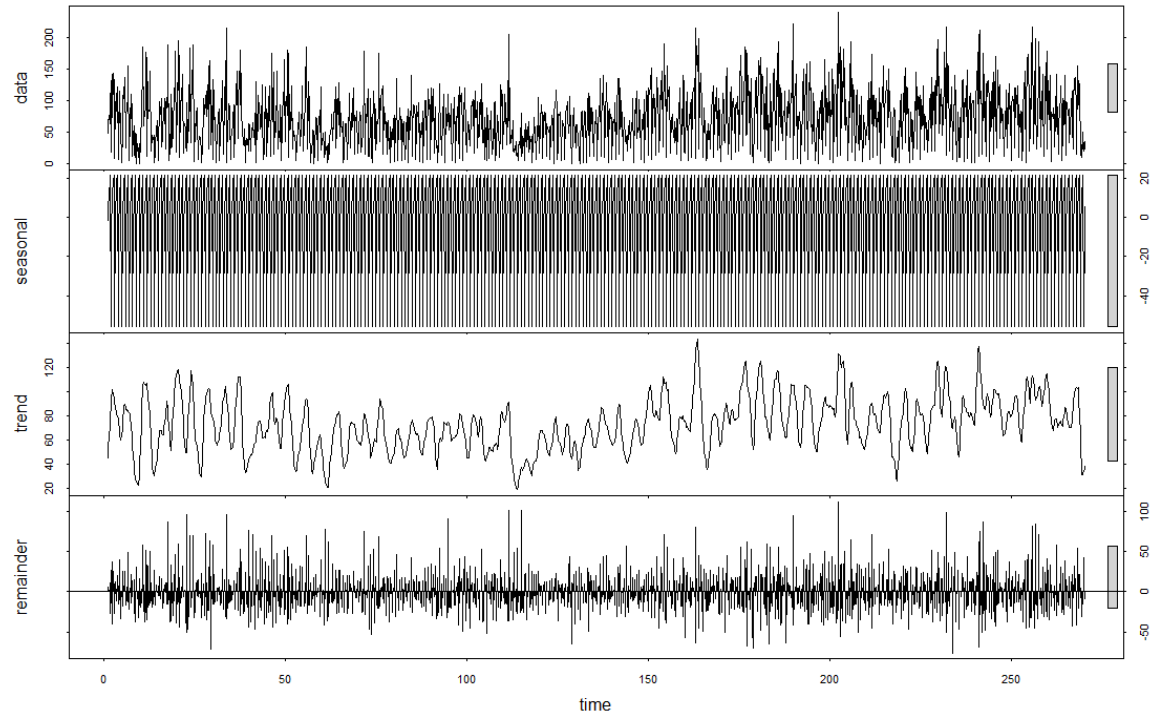


Figure 7 - Time series decomposition showing trend, seasonals and remainder.

Source: Elaborated by the Author

Figure 7 shows the resulting the decomposition resulting from R package where trend, seasonal and remainder are displayed graphically for the dataset of this research. The trend component does not show clearly signs of a increasing or decreasing pattern and the seasonality shows the recurring temporal pattern present in the data based on a specific value. This values will be defined as periodicity and for the dataset of this study is measured in section 3.2. Finally, the remainder terms or error indicate the amount of noise present and can help identify outliers in the data. This can be observed in Figure 7 as the errors are skewed to the positive side.

Secondly, modeling the trend and seasonality modelled can be used as additional regressors as shown in Table 2, leading to more precise predictions. To use Loess decomposition for prediction future values of the target value, seasonality must be considered as deterministic otherwise the future values would be unknown.

To overcome this, another approach to time series decomposition is a discrete decomposition. A linear regression can be found through the combination of trend and harmonics components (Morettin, 2006). These sinusoidal components are the discrete Fourier transform of the series and can be expanded to cover large seasonal

periods such as databases with daily observations. This define a pattern of deterministic seasonality and can be modelled using mathematical functions such as sine and cosine waves. Consider the model:

$$\hat{S}_t = \sum_{k=1}^{K} [\alpha_k \sin(\varpi t) + \beta_k \cos(\varpi)] + Z_t \qquad 19$$

where α and β are constants, ω is the frequency and Z_t is the random component.

The frequency is defined as $\varpi = \frac{2\pi k}{m}$, where m is the season period of the dataset and k is number of Fourier pairs chosen. For multiple years of daily observations yearly observations, possible values for m are $m_A = 365$ for annual seasonality or $m_M = \{28,29,30,31\}$ for monthly seasonality, while k = 12.

To further extend the seasonality replication, the day of the week is considered also as a predictor using the following dummy variables (Hyndman; Athanasopoulos, 2018):

Table 3 - Weekly Dummy Variables

	$d_{1,t}$	$d_{2,t}$	$d_{3,t}$	$d_{4,t}$	$d_{5,t}$	$d_{6,t}$
Monday	1	0	0	0	0	0
Tuesday	0	1	0	0	0	0
Wednesday	0	0	1	0	0	0
Thursday	0	0	0	1	0	0
Friday	0	0	0	0	1	0
Saturday	0	0	0	0	0	1
Sunday	0	0	0	0	0	0

Source: Hyndman; Athanasopoulos, 2018

Incorporating them into (19):

$$\hat{S}_t = \sum_{k=1}^{K} [\alpha_k \sin(\varpi t) + \beta_k \sin(\varpi t)] + \sum_{j=1}^{6} \gamma_k d_{j,t} + Z_t \qquad 20$$

Finally, the trend terms are incorporated using a polynomial of order p. The trend component of a time series represents a persistent, long-term change in the mean of the series. In the time series in this study, an increasing trend might be the

effect of a market expansion for a selected product while a decreasing trend could mean market contraction:

$$\widehat{M}_t = \delta_0 + \delta_1 t + \delta_2 t^2 + \delta_3 t^3 + \cdots \delta_p t^p = \sum_{p=0}^{P} \delta_p t^p \quad (21)$$

Expanding the regression model to:

$$\hat{Y}_t = \sum_{p=0}^{P} \delta_p t^p + \sum_{k=1}^{12} [\alpha_k \sin\left(\frac{2\pi k t}{m}\right) + \beta_k \sin\left(\frac{2\pi k t}{m}\right)] + \sum_{j=1}^{6} \gamma_k d_{j,t} + Z_t \quad (22)$$

where α_k, β_k, γ_k, δ_p are obtained through least squares regression (6):

$$LSE(\alpha_k, \beta_k, \gamma_k, \delta_p) = \sum_{t=1}^{N} (Z_t - \widehat{M}_t - \hat{S}_t) \quad (23)$$

To verify if equation (23) can replicate the entire trend and seasonality, the use of the correlograms is necessary. This will be explained in section 2.4.

Additionally, the regression standard errors can also be used to perform hypothesis tests on the coefficients. Standard errors measure the uncertainty in the estimate of a regression coefficient and is calculated as the square root of the variance of the coefficient estimate. This test determines how far from zero the estimated coefficient is. For instance, the first trend parameter δ_1:

$$H_0: \delta_1 = 0 \quad (24)$$

This probability p-value is given by the t-statistic (James *et al*., 2013):

$$t = \frac{\delta_1 - 0}{SE(\delta_1)} \quad (25)$$

which measures the number of standard deviations that δ_1 is away from 0.

2.4. Autoregressive models and correlograms

In many time series, the consecutive observations will be correlated. By measuring the autocorrelation of the series with its time steps (known as lags), it can provide an opportunity to model the trend and seasonality by adding parameters related to the most appropriate lag. The correlation structure of a time series model is defined by the correlation function, and it is estimated from the observed time series that is stationary in mean and variance. A correlation of a variable with itself at different

times is known as autocorrelation or serial correlation. Defining at k time unit of lags from the time series of size n:

a. Sample autocovariance function ck:

$$c_k(y_t, y_{t+k}) = \frac{1}{n}\sum_{t=1}^{n-k}(y_t - \bar{y})(y_{t+k} - \bar{y}) \quad 26$$

b. Sample autocorrelation function r_k:

$$\rho_k = \frac{\rho_k}{\rho_0} \quad 27$$

The autocorrelation function (ACF) is a plot of r_k against k which is called correlogram. The x-axis gives the lag k and the y-axis gives the autocorrelation r_k at each lag. Correlation is dimensionless, so there is no unit for the y-axis.

Observation of the ACF together with moving average has led to the creation of autoregressive AR(p), moving average MA(q) and a combination of both ARMA(p,q) models, where p and q are the orders of the autoregressive and moving average terms.

Autoregressive model (AR) is a linear combination of y_t past values represented by:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \omega_t \quad 28$$

The introduction of the backward shift operator B is convenient to express autoregressive models:

$$By_t = y_{t-1} \quad 29$$

$$B^n y_t = y_{t-n} \quad 30$$

$$\phi_p(B) y_t = \omega_t \quad 31$$

where p = order of the autoregressive part, ϕ_p = autoregressive parameters and ω_t = white noise ~(0,σ^2) of mean zero e variance σ^2.

Rather than using the past values of the series in a regression, a Moving Average model uses past forecast errors in a regression-like model represented by:

$$y_t = \mu + \omega_t - \theta_1 \omega_{t-1} - \theta_2 \omega_{t-2} - \cdots - \phi_q y_{t-q} \quad 32$$

$$y_t = \mu + \theta_q(B)\omega_t \quad 33$$

Where q = order of the moving average part and θ_q = moving average parameters.

Notice that each value of y_t can be thought of as a weighted moving average of the past forecast errors. However, moving average models should not be confused with the moving average smoothing from section 2.3. A moving average model is used for forecasting future values, while moving average smoothing is used for estimating the trend-cycle of past values.

The Box–Jenkins Autoregressive Moving Average (ARMA) model (Box and Jenkins 1976) can be represented by the autoregressive moving average (ARMA) (p, q) model:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \omega_t + \theta_1 \omega_{t-1} + \theta_2 \omega_{t-2} + \cdots + \beta\theta_q \omega_{t-q} \quad 34$$

Applying the backward shift operator B:

$$\phi_p(B) y_t = \theta_q(B) \omega_t \quad 35$$

An AR(p) process has an ACF that decays as damped exponentials and/or sinusoids which are infinite in length, while a MA(q) process has a finite autocorrelation function that stops at lag “q”. For a ARMA(p,q) model, the ACF which is finite in length decays as damped exponentials and/or sinusoids after lag q-p. The observations are useful to help identify the order of “p” and “q” (Morettin, 2006).

As second correlogram that can be drawn for the time series is the partial autocorrelation function (PACF). Box, Jenkins and Reinsel (1994) proposed this function as another instrument to determine the order for autoregressive component. Taking equation (28) for the autoregressive process, denoting ϕ_{kj} the j-th coefficient and ϕ_{kk} the last coefficient, from equation (27) the autocorrelations ρ_k are:

$$\rho_j = \phi_{k1}\rho_{j-1} + \phi_{k2}\rho_{j-2} + \cdots + \phi_{kk}\rho_{j-k}, \qquad j = 1, \ldots, k \quad 36$$

The Yule-Walker equations are used to represent the PACF as:

$$\begin{bmatrix} 1 & \rho_1 & \rho_2 & \ldots & \rho_{k-1} \\ \rho_1 & 1 & \rho_1 & \ldots & \rho_{k-2} \\ \vdots & \ldots & \ldots & \ldots & \vdots \\ \rho_{k-1} & \rho_{k-2} & \rho_{k-3} & \ldots & 1 \end{bmatrix} \begin{bmatrix} \phi_{k1} \\ \phi_{k2} \\ \vdots \\ \phi_{kk} \end{bmatrix} = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_k \end{bmatrix} \quad 37$$

$$\mathbf{P}_k \boldsymbol{\phi}_k = \boldsymbol{\rho}_k \quad 38$$

These equations are solved recursively for k=1,2=,3… to find the partial autocorrelation function (PACF):

$$\phi_{kk} = \frac{|\mathbf{P}_k^*|}{|\mathrm{P}_k|} \quad 39$$

where $\mathbf{P}_k$ is the autocorrelation matrix and $\mathbf{P}_k^*$ is the matrix $\mathbf{P}_k$ with the last column replaced by autocorrelation vectors. An efficient algorithm to solve this problem is the Durbin–Levinson Algorithm (Durbin, 1960). Considering the autoregressive order p:

$$\hat{\phi}_{p+1,j} = \hat{\phi}_{p,j} - \hat{\phi}_{p+1,p+1} + \hat{\phi}_{p,p-j+1} \quad 40$$

$$\hat{\phi}_{p+1,p+1} = \rho_{p+1} - \frac{\sum_{j=1}^{p} \hat{\phi}_{p,j}\, \rho_{p+1-j}}{1 - \sum_{j=1}^{p} \hat{\phi}_{p,j}\, \rho_j} \quad 41$$

$$\hat{\phi}_{1,1} = \rho_1 \quad 42$$

The algorithm starts with $\hat{\phi}_{1,1} = \rho_1$, then on p stage $\hat{\phi}_{p+1,p+1}$ are calculated based on stage p-1. Then the other coefficients are updated in a recursive pattern.

Box, Jenkins and Reinsel (1994) also shows that an AR(p) process has PACF $\phi_{kk} \neq 0$, for $k \leq p$ and $\phi_{kk} = 0$, for $k > p$. For a MA(q) process the PACF has similar behavior of the ACF of an AR(p) process which is damped exponentials and/or sinusoids. Lastly for the ARMA(p,q) process the PACF behaves as MA(q) process

The partial autocorrelation function also produces a plot of ϕ_{kk} against k. Both ACF and PACF plots then provides a good way to find the moving average order MA(q) and autoregressive order AR(p) to model the parametric form of $\hat{f}$ in an ARMA model (Morettin, 2006).

To decide what orders are negligible, it is observed that, in AR(p) process, the PACF values at lags greater than “p” are approximately independent N(0,1/p) random variables (Brockwell; Davis, 2016). This means that with 95% confidence the PACF values beyond lag “p” will fall within the bonds $\frac{\pm 1.96}{\sqrt{n}}$. The same applies to ACF for lags greater than “q”.

2.5. Stationarity, seasonality and serial correlation statistical tests.

A stationary time series means that the statistical properties like the mean and variance do not change over time (for example the Stochastic white noise with a mean zero and no correlation between its values at different times). Thus, time series with

trends or with seasonality are not stationary and its properties will affect the value of the time series at different times.

The Augmented Dickey Fuller Test for Stationarity (ADF) is a statistical significance test which the null hypothesis says that a unit root exists. Unit root is a characteristic of a time series that makes it non-stationary. Considering the AR process from equation (28) and rewriting as:

$$y_t = \theta_0 + \sum_{i=1}^{p} \phi_i y_{t-1} + \omega_t \quad 43$$

$$\Delta y_t = \theta_0 + \phi_1^* y_{t-1} + \sum_{i=1}^{p} \phi_1^* \Delta y_{t-1} + \omega_t \quad 44$$

The null hypothesis is that $\Delta y_t = \omega_t$, which means y_t difference is stationary or $\phi_1^* = 0$ (Morettin, 2006). Rejecting the null hypothesis (p-value < confidence level) indicates that the series y_t is stationary and $\phi_1^* < 1$. Indeed, the successive differences is one of the options to transform the series to stationary. In general, the n-th difference of y_t is (Morettin, 2006):

$$\Delta^n y(t) = \Delta[\Delta^{n-1} y(t)] \quad 45$$

Additionally, statistical tests can be used to evaluate the presence of deterministic seasonality and serial correlations in the series.

Considering the case of equation (22) with “p” predictors and the entire dataset of size T, a deterministic seasonality is presented if the hypothesis that all coefficients are null is rejected:

$$H_0: \delta_p = \alpha_k = \beta_k = \gamma_k = 0 \quad 46$$

This test if performed by computing the F-statistic (James *et al*, 2013):

$$F = \frac{(\sum_{t=1}^{T}(y_t - \bar{y})^2 - \sum_{t=1}^{T}(y_t - \hat{y}_t)^2)}{\sum_{t=1}^{T}(y_t - \hat{y}_t)^2} \frac{(T-p-1)}{p} \quad 47$$

of F distribution (p,T-p-1) degrees of freedom. When there is no relationship between the response and predictors the F-statistic takes on a value close to 1 (James *et al*., 2013). On the other hand, it is expected F to be greater than 1.

In regarding to serial correlation, the Ljung-Box test is a statistical test that is used to test for autocorrelation in a time series. Autocorrelation is the correlation of a

time series with itself at different lags. In other words, it is the extent to which the values of a time series are related to their own past values. It was proposed by Ljung and Box (1978) for testing goodness of fit for ARMA models using the residuals. It is based on the statistic:

$$Q^* = T(T+2)\sum_{k=1}^{h}(T-k)^{-1}r_k^2 \qquad 48$$

where T is the length of the time series, r_k is the k-th autocorrelation coefficient of the residuals and h the number of lags to test. The null hypothesis say assumes large values of $Q^* > \chi^2_{1-\alpha,h-K}$ indicate that there are significant autocorrelations in the residual series and therefore lack of fit. $\chi^2_{1-\alpha,h-K}$ is the Chi-squared test with h-K degrees of freedom where K is the number of parameters estimated in the model. Assuming K=0, this test can be used for checking autocorrelation of the original series.

2.6. Forecasting models

In the age of data science, exponential smoothing and ARIMA models are still today valuable forecasting models capable of similar performance of machine learning models.

Exponential smoothing started in the 1950s and 1960s with the work (Holt) 1957, Winters (1960) and McCormick (1969). They provided a simple but useful classification of the trend and the seasonal patterns depending on whether they are additive or multiplicative. Exponential smoothing methods were improved later in 1985, with the work of Gardner (1985), providing a thorough review and synthesis of work in exponential smoothing to that date and extended McCormick classification to include additive damped trend. Taylor (2003) included methods with a multiplicative damped trend forming the set of equations used to this date.

In 1976, Box and Jenkins integrated the existing knowledge at that time of, autoregressive and moving average concepts in one model and applied in a three-stage cycle of time series identification, model estimation (parameters) and verification (error metrics). The Autoregressive Integrated Moving Average (ARIMA) model as it is known had an enormous impact on the theory and practice time series analysis and implementation since then. With the popularization of the computer in the 90´s it expanded the use of these models for both ARIMA and exponential smoothing

methods, evolving to improved forecasting performance by better selection and fitting it´s parameters (Gooijer; Hyndman 2006).

In the year´s 2000 advances in computer science and machine learning have increased the forecaster's toolbox of methods by using data driven models such as neural networks, clustering-based forecasting and ensemble methods of bootstrap aggregating (for instance Random Forest model for regression) along with other ARIMA derived models (Gooijer; Hyndman 2006).

Despite the excitement surrounding these new developments, ARIMA and exponential smoothing are not as prone to overfitting as more complex methods. In addition to this, the idea of combining forecasts models for the same dataset has received increased attention in the forecasting community recently. The basic idea is that by combining different methods their forecast errors won´t be highly correlated, but also, the residuals from one model can be served as a regressor to the other.

As discussed in the introduction, traditional parametric models such as exponential smoothing and ARIMA continue to provide reliable results and it is not the scope of this work to review every forecasting model available but rather provide a mechanism to improve model calibration and selection through hyperparameters search and performance measure. This process can be extended to any model after the hyperparameters are mapped. Considering this, the approach of this work is summarized as follows:

a. The dataset is composed by time series daily observations;
b. It contains multiple features that can be used as additional regressors beyond the time steps;
c. Exponential smoothing method with damped parameters, Seasonal ARIMA and Recurrent Neural Network LSTM are the options selected as possible forecasting models;
d. To overcome issues with overfitting and search for the optimal point between the bias-variance trade off, a resampling process is adopted while searching for the best hyperparameters;
e. The hyperparameters search space is defined for each model;
f. A set of validation measures are chosen to select the best hyperparameters in each model and the best model;

2.7. Exponential Smoothing (ES)

The Holt Winters method, belongs to a class of exponential smoothing methods (which it is referred as ES) that aims to capture the behavior of the time series, separating it in level, trend and season.

Exponential smoothing arises from the fact that this method is a generalization of an exponentially weighted moving average a_t given by:

$$a_t = \alpha y_t + (1-\alpha)a_{t-1} \quad (49)$$

The value of α determines the amount of smoothing, and it is referred to as the smoothing parameter (Cowpertwait; Metcalfe, 2009). The Holt-Winters method generalizes Equation (52). There are two variations to this method that differ in the nature of the seasonal component. The additive method is preferred when the seasonal variations are roughly constant through the series, while the multiplicative method is preferred when the seasonal variations are changing proportional to the level of the series.

The equations that follow next are: one for the level l_t , one for the trend b_t, and one for the seasonal component s_t, with corresponding smoothing parameters α, β, γ and the damped parameter ∅. Periodicity is denoted as s.

A damped trend method is appropriate when there is a trend in the time series, but one believes that the growth rate at the end of the historical data is unlikely to continue more than a short time into the future. The damped parameter can then be applied.

Considering forecasting equation $\hat{y}_{t+k}$ made after the observation at time t+k, the Holt-Winters additive method for trend and season with damped parameter is composed by the set of equations:

$$Level: l_t = \alpha(y_t - s_{t-s}) + (1-\alpha)(l_{t-1} + \phi b_{t-1}) \quad (50)$$

$$Trend: b_t = \beta(l_t - l_{t-1}) + (1-\beta)\phi b_{t-1} \quad (51)$$

$$Season: s_t = \gamma(y_t - l_{t-1} - \phi b_{t-1}) + (1-\gamma)s_{t-s} \quad (52)$$

$$\hat{y}_{t+k} = l_t + (\phi + \phi^2 + \cdots + \phi^k)b_t + s_{t+k-s} \quad (53)$$

Now the Holt-Winters Multiplicative method for trend and season with damped parameter changes to:

$$Level: l_t = \alpha\left(\frac{y_t}{s_{t-s}}\right) + (1-\alpha)\left(l_{t-1} + {b_{t-1}}^{\emptyset}\right) \quad 54$$

$$Trend: b_t = \beta(l_t - l_{t-1}) + (1-\beta){b_{t-1}}^{\emptyset} \quad 55$$

$$Season: s_t = \gamma\left(\frac{y_t}{l_{t-1} + {b_{t-1}}^{\emptyset}}\right) + (1-\gamma)s_{t-s} \quad 56$$

$$\hat{y}_{t+k} = l_t {b_t}^{(\phi+\phi^2+\cdots+\phi^k)} s_{t+k-p} \quad 57$$

Variations of additive/multiplicative trend and season are possible, and it creates the set of hyperparameters that can be explored to find the best fit. Makridakis, Wheelwright and Hyndman (1998) introduced a taxonomy that summarize 15 exponential smoothing methods through the recursive calculation of the level, trend, damped trend and seasonal component if they are additive or multiplicative. Appendix A includes a table with all the recursive equations derived from equations (53) to (60).

Usually, you want to smooth the data enough to reduce the noise (irregular fluctuations) so that the pattern is more apparent. Normally, the smoothing parameters are searched using the optimization of an error function such as the mean squared error.

In this study the use of the trend and seasonal component as “additive”, “multiplicative” or “not applicable” will be the hyperparameters and in the methodology chapter it will be defined how to search for the best configuration.

2.8. ARIMA and Seasonal ARIMA

While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe linearly the autocorrelations in the data.

A time series y_t follows an ARMA(p,q) process if the dth differences of y_t series are an ARIMA(p,d,q) process. Differencing has the objective to make the series stationary to improve predictions. Using the notation of the backward shift operation from equation (30):

$$\emptyset_p(B)(1-B)^d y_t = \theta_q(B)\omega_t \quad (58)$$

$$(1-\emptyset_1 B-\cdots-\emptyset_1 B^p)(1-B)^d y_t = \left(1+\theta_1 B+\cdots+\theta_q B^q\right)\omega_t \quad (59)$$

where AR(p) = $(1-\emptyset_1 B-\cdots-\emptyset_1 B^p)$, MA(q) = $\left(1+\theta_1 B+\cdots+\theta_q B^q\right)$ and dth differences = $(1-B)^d$.

A seasonal ARIMA model (which will name as SARIMA), uses differencing at lag equal to the number of periodicities s to remove additional seasonal effects. Seasonal ARIMA (p,d,q) (P,D,Q) can be expressed using the backward shift operator:

$$\Phi_p(B^s)\emptyset_p(B)(1-B^s)^D(1-B)^d y_t = \Theta_Q(B^s)\theta_q(B)\omega_t \quad (60)$$

$$AR\colon \emptyset_p(B) = (1-\emptyset_1 B-\cdots-\emptyset_1 B^p) \quad (61)$$

$$MA\colon \theta_q(B) = \left(1+\theta_1 B+\cdots+\theta_q B^q\right) \quad (62)$$

$$\text{Seasonal AR: } \Phi_P(B^s) = (1-\Phi_1 B^s-\cdots-\Phi_P B^{PS}) \quad (63)$$

$$\text{Seasonal MA: } \Theta_Q(B^s) = \left(1+\Theta_1 B^s+\cdots+\Theta_Q B^{QS}\right) \quad (64)$$

where $\emptyset_p, \theta_q$, Φ_P, Θ_Q are the polynomials for orders p,q,P,Q respectively, ‘B’ is the backshift operator, ‘s’ is the seasonal lag; ‘d’ and ‘D’ are non-seasonal and seasonal differences; ‘ϕ’ and ‘Φ’ are the non-seasonal and seasonal autoregressive parameters; ‘θ’ and ‘Θ’ are the non-seasonal and seasonal moving average parameters respectively. The orders (p,d,q) (P,D,Q) are the hyperparameters for SARIMA.

The seasonal ARIMA can be extended to incorporate additional regressors in its final functional form:

$$\Phi_p(B^s)\emptyset_p(B)(1-B^s)^D(1-B)^d y_t = \Theta_Q(B^s)\theta_q(B)\omega_t + \beta_k x_{k,t} - \emptyset_p \quad (65)$$

where β_k is the coefficient value of the additional k regressors $x_{k,t}$.

ARIMA models assume a linear relationship between past observations and future values. This means that changes in past data will proportionally affect future values. An iterative procedure can be used to choose possible candidates for the hyperparameters. It starts with the stationarity of the time series. To obtain a stationary

time series, differencing process is employed till the seasonality in data fades away. The AR and MA terms of stationary time series data are then obtained by examining the patterns of partial autocorrelation function (PACF) and autocorrelation functions (ACF) plots respectively. Once tentative models are identified, the next step is the estimation of model parameters by fitting the model against a training dataset. An evaluation dataset can then be used to evaluate the model goodness of fit.

Both exponential smoothing and ARIMA belongs to a class of parametric forecasting models as a functional form f is predefined. Both models were identifying as containing hyperparameters that govern the function form of f. Finding the best hyperparameters is key to obtain better results in the evaluation dataset.

2.9. Parameter estimation for exponential smoothing and ARIMA

Exponential smoothing and SARIMA models are classified as parametric models as they can be represented by equations with independent variables. These independent variables are called parameters.

These parameters are found by apply one of the two follow methods: the least squares estimation such as in equation 23 or the maximum likelihood estimation. The method of least squares has an appealing intuitive interpretation. Its application depends on knowledge only of the means and covariances of the observations. Maximum likelihood estimation depends on the assumption of a particular distributional form of the observations. For both models here it is assumed that the observations given in equation 1 are independent random variables following a normal distribution.

According to the Gauss–Markov theorem the least squares estimation produces unbiased linear estimators with minimum variance (Cowpertwait; Metcalfe, 2009). The Maximum likelihood estimators are not generally unbiased, but in case of large sample, where the observations have the same distribution, they can be shown to have small mean squared error relative to other competing estimators.

Considering $\boldsymbol{x} = (x_1, \dots, x_t)$ is a vector of observations of independent $N(\mu, \sigma^2)$ random variables, the likelihood function is:

$$\mathcal{L}(\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} exp\left\{-\frac{1}{2\sigma_a^2}\sum_{t=1}^{T}(x_t - \mu)^2\right\} \qquad 66$$

Maximization of $\mathcal{L}$ with respect to μ and σ is equivalent to minimization of:

$$-2ln\mathcal{L}(\mu,\sigma^2) = nln(2\pi) + 2nln(\sigma) + \frac{1}{\sigma^2}\sum_{t=1}^{T}(x_t-\mu)^2 \quad 67$$

Assuming that the model has been specified i.e., the hyperparameters were selected, next is to find the parameters of the final function form. This will be obtained by minimizing the maximum likelihood (ML) function. The objective of this function is to maximize the probability of observing these realizations (given a known parametric distribution with unknown parameters). This is provided by the joint density of the observations known as the likelihood function.

According to Hyndman *et al* (2018), for ES, it is easier to represent and compute the likelihood of the state space model given by:

$$y_t = \omega(x_t - 1) + r(x_{t-1})\varepsilon_t \quad 68$$

$$x_t = \boldsymbol{f}(x_t - 1) + \boldsymbol{g}(x_{t-1})\varepsilon_t \quad 69$$

where $r(\cdot)$ and $\omega(\cdot)$ are scalar functions, $\boldsymbol{f}(\cdot)$ and $\mathbf{g}(\cdot)$ are vector functions $x_t = (l_t, b_t, s_t, s_{t-1}, \dots, s_{t-m+1})$, ε_t is the error term assuming Gaussian white noise with mean and variance σ^2. The model with additive errors has $r(x_{t-1}) = 1$ so that $y_t = \mu_t + \varepsilon_t$. The model with multiplicative errors has $r(x_{t-1}) = \mu_t$ so that $y_t = \mu_t(1 + \varepsilon_t)$. All methods from appendix A can be written in the form given in equation (69) and (70).

From the joint density function for $p(y/(\alpha,\beta,\gamma,\emptyset), x_0, \sigma^2)$, the Gaussian log likelihood is written as:

$$\mathcal{L}^*(\alpha,\beta,\gamma,\emptyset,x_0) = n\ ln\left(\sum_{t=1}^{T}\varepsilon_t^2\right) + 2\sum_{t=1}^{T} ln|r(x_{t-1})| \quad 70$$

The maximum likelihood estimates $\alpha, \beta, \gamma, \emptyset$ set of parameters can be obtained by minimizing $\mathcal{L}^*$ and T is the number of observations.

According to Morettin (2006), for ARIMA, equation (61) can be used by taking the differencing "d" $W_t = \Delta^d y_t$ to reach to the ARMA resulting model:

$$\omega_t = \widetilde{W}_t - \phi_1 \widetilde{W}_{t-1} - \cdots - \phi_p \widetilde{W}_{t-p} + \theta_1 \omega_{t-1} + \cdots + \theta_q \omega_{t-q} \qquad 71$$

where $\widetilde{W}_t = W_t - \mu$.

Assuming the normality of ω_t the joint density function:

$$f(\omega_t) = 2\pi^{-t/2} \sigma_\omega^{-t} exp\left\{ - \sum_{t=1}^{T} \frac{\omega_t^2}{2\sigma_\omega^2} \right\} \qquad 72$$

The maximum value for $\mathcal{L}^*$ in this case is found by choosing the estimates ϕ_p and θ_q in:

$$\mathcal{L}^*(\phi, \theta, \sigma_a^2) = 2\pi^{-t} exp\left\{ - \frac{1}{2\sigma_a^2} \sum_{t=1}^{T} (\widetilde{W}_t - \phi_1 \widetilde{W}_{t-1} - \cdots - \phi_p \widetilde{W}_{t-p} + \theta_1 \omega_{t-1} + \cdots + \theta_q \omega_{t-q} \right\} \qquad 73$$

For the estimators found through the maximum likelihood function, it is possible to construct confidence intervals. When T is large, the estimators have a normal asymptotic distribution with specific variances. This allows for calculating confidence intervals and conducting hypothesis tests on the parameter values.

A general discussion on the convergence properties of various numerical algorithms for computing maximum likelihood estimates was provided by Hamilton (1994).

So, by defining the hyperparameters for each model next is find the parameters in the final function form equations. This is given by maximum likelihood estimation (MLE). When one wants to focus on improving the parameters estimation, advanced statistical learning and/or neural network models can be found in publications as a tool to try to improve the results from MLE (finding a better local minimum for the objective

function) (Zhang *et al*., 2021) and improve the variance of the estimators. This was not the scope of this work.

2.10. Recurrent Neural Network – LSTM overview

Recurrent Neural Networks has shown superior feature extraction ability and has contributed to improve the accuracy of time series forecasting (Shen *et al*., 2020). Additionally, having historical time series with an adequate length and additional features (multivariate) presents an opportunity to explore if more complex models than those obtained with traditional time-series provides better performance.

Recurrent Neural Networks can automatically handle temporal structures like trends and seasonality, learn arbitrary complex mappings, support multiple inputs and outputs and principally address long-term information preservation. One of the earliest approaches to address this was the Recurrent Neural Network called Long-Short term Memory (LSTM) defined by Hochreiter and Schmidhuber (1997).

In a recurrent neural network, there is a one-to-one correspondence between the layers in the network and the specific positions in the sequence (time-stamp input). Instead of a variable number of inputs in a single input layer, the network contains a variable number of states, and each states have a single input corresponding to that time-stamp input dimension. The inputs can directly interact with down-stream hidden states depending on their positions in the sequence. In other words, the same state-wise architecture is repeated in time, and therefore the network is referred to as recurrent. Recurrent neural networks are also feed-forward networks with a specific structure based on the notion of time layering, so that they can take a sequence of inputs and produce a sequence of outputs. Such models are particularly useful for sequence-to-sequence learning applications like machine translation or for predicting the next element in a sequence.

Although any of the variants for the recurrent neural network can be used, the most common networks are the Long-Short term Memory (LSTM) and Gated recurrent units (GRU).

From equation (1) let $x_t = \{x_1^{(i)}, x_2^{(i)}, \ldots\ldots, x_t^{(i)}\}$ time series with additional regressors at the time “t” where “i” is the number of regressors. The LSTM network is trained based on a defined sequence of observations $x_N = \{x_1^{(1)}, x_2^{(1)}, \ldots\ldots, x_N^{(1)}\}$ where N is the number of samples. The individual observations are scaled by:

$$x_{t,scaled}^{(i)} = \frac{x_t^{(i)} - x_{min}^{(i)}}{x_{max}^{(i)} - x_{min}^{(i)}} \quad 74$$

LSTMs are sensitive to the scale of their inputs and normalizing all the dataset are required to improve the convergence of the optimization error function and prevent vanishing gradient problem. Standard RNN cannot bridge more than 5 to 10 time steps where the back-propagated error signals tend to either grow or shrink with every time (Bengio *et al*.; 1994 and Staudemeyer and Morris; 2019). Over many times steps the error therefore typically blows-up or vanishes. Other option is the standardization of the dataset, although this assumes that the data is normally distributed.

The training process consists in set up a sliding window of size m, m<N as the input size for the network $x_t = \{x_1^{(i)}, x_2^{(i)}, \ldots\ldots, x_m^{(i)}\}$. Then the m*i inputs are used to predict the next step $\hat{x}_{m+1}^{(i)}$. The second window $x_t = \{x_2^{(i)}, x_3^{(i)}, \ldots\ldots, x_{m+1}^{(i)}\}$ will predict $\hat{x}_{m+2}^{(i)}$. This process continues until the window slide to the end of the training data set.

Regarding the structure, in LSTM, the hidden layers from a multi-layer perceptron are replaced for cell units and each unit contains inside four different interacting layers. The first step in LSTM is to decide what information is going to throw away from the cell state. This decision is made by a sigmoid layer called the “forget gate layer” (1). It looks at previous point in time (hidden state) and the new input of data to output a number between 0 and 1. The next step is to decide what new information is going to store in the cell state. First, a sigmoid layer called the “input gate layer” (2) decides which values to update. Next, a tanh layer (3) creates a vector of new candidate which will decide whether update the previous cell state. These two results are combined (Hadamard product) and scaled to create an updated state.

Finally, the output gate (4) will be based on the current filtered cell state version. A sigmoid layer decides what parts of the cell state is going to output. Then, a tanh function (to push the values to be between -1 and 1) multiplies the output of the sigmoid gate to result only the relevant parts.

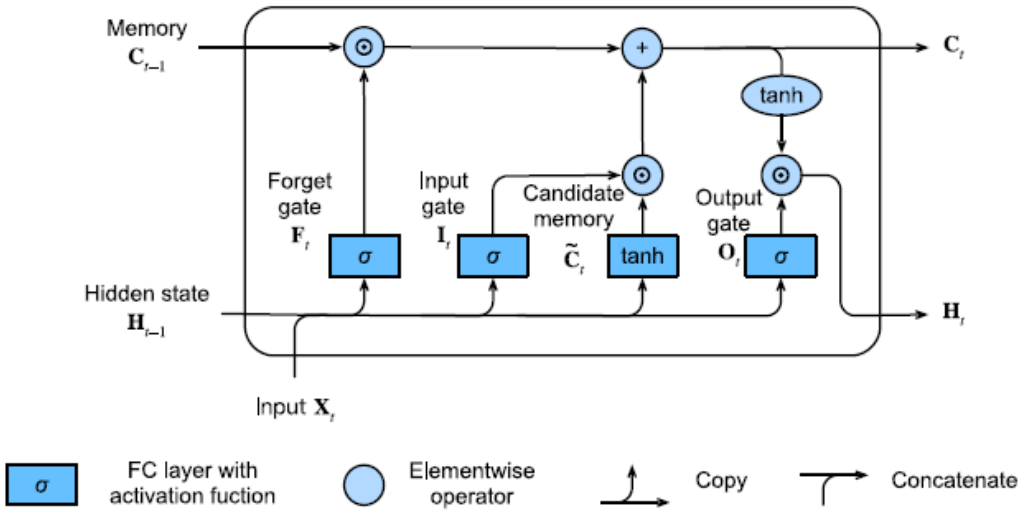


Figure 8 - Representation of units inside cells in LSTM network

Source: Adapted from Zhang *et al*. (2020)

In Figure 8: Apply the tanh function to the current cell state pointwise to obtain the transformed cell state, which now lies in [-1,1]. Pass the previous hidden state and current input data through the sigmoid activated neural network to obtain the filter vector. Apply this filter vector to the transformed cell state by the Hadamard product. Output the new hidden state.

These gates are responsible to decide what to consider in past information when updating the current cell state. The sigmoid function (indicated by σ) is used as the activation of the hidden state, and the hyperbolic tangent function (indicated by tanh) as the activation of the output. These are the default activation functions based on the work of Hochreiter and Schmidhuber (1997). A classical LSTM cell already contains quite a few non-linearities: three sigmoid functions and one hyperbolic tangent (tanh) function and there is no need to add another activation layer after each the LSTM cell or before the output layer.

Defining matrix $\mathbf{X}_t \epsilon R^{n\times d}$ of real numbers with n rows and d columns as a batch of data. R is the space of values with the number of time steps n of the batch and number of inputs d. $\mathbf{X}_t$ batch is fed to LSTM gates at current timestep t and the hidden state of the previous time step as matrix $\mathbf{H}_{t-1}\epsilon R^{n\times h}$ (number of hidden states h). There are h hidden units, the batch is of size n and the number of inputs is d. The

input gate (2) matrix $\boldsymbol{I}_t \in R^{n\times h}$, the forget gate matrix (1) $\boldsymbol{F}_t \in R^{n\times h}$ and the output gate (4) matrix $\boldsymbol{O}_t \in R^{n\times h}$ are defined as:

$$\boldsymbol{I}_t = \sigma(\mathbf{X}_t\boldsymbol{W}_{xi} + \mathbf{H}_{t-1}\boldsymbol{W}_{hi} + \boldsymbol{b}_i) \quad 75$$

$$\boldsymbol{F}_t = \sigma(\mathbf{X}_t\boldsymbol{W}_{xf} + \mathbf{H}_{t-1}\boldsymbol{W}_{hf} + \boldsymbol{b}_f) \quad 76$$

$$\boldsymbol{O}_t = \sigma(\mathbf{X}_t\boldsymbol{W}_{xo} + \mathbf{H}_{t-1}\boldsymbol{W}_{ho} + \boldsymbol{b}_o) \quad 77$$

where σ is the sigmoid function, $\boldsymbol{W}_{xi}$, $\boldsymbol{W}_{xf}$, $\boldsymbol{W}_{xo} \in R^{d\times h}$ and $\boldsymbol{W}_{hi}, \boldsymbol{W}_{hf}, \boldsymbol{W}_{ho} \in R^{h\times h}$ are the weight parameters matrices. $\boldsymbol{b}_i, \boldsymbol{b}_f, \boldsymbol{b}_o \in R^{n\times h}$ are bias parameters vectors.

The candidate memory gate (3) $\tilde{\mathbf{C}}_t \in R^{n\times h}$ is defined as:

$$\tilde{\mathbf{C}}_t = tanh(\mathbf{X}_t\boldsymbol{W}_{xc} + \mathbf{H}_{t-1}\boldsymbol{W}_{hc} + \boldsymbol{b}_c), \quad 78$$

where tanh is the tangent hyperbolic function, $\boldsymbol{W}_{xc} \in R^{d\times h}$ and $\boldsymbol{W}_{hc} \in R^{h\times h}$ are weight parameters and $\boldsymbol{b}_c \in R^{n\times h}$ is a bias parameter.

Input I_t governs how much new data to consider via matrix $\tilde{\mathbf{C}}_t$ and forgetting state $\boldsymbol{F}_t$ addresses how much of the old memory cell content matrix $\boldsymbol{C}_{t-1} \in R^{n\times h}$ should retain. Using the Hadamard product:

$$\boldsymbol{C}_t = \boldsymbol{F}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{I}_t \odot \tilde{\mathbf{C}}_t \quad 79$$

If the forget gate is always approximately 1 and the input gate is always approximately 0, the past memory cells $\boldsymbol{C}_{t-1}$ will be saved over time and passed to the current timestep. This design was introduced to alleviate the vanishing gradient problem and to better capture dependencies for time series with long range dependencies.

Finally, the hidden state computation matrix $\mathbf{H}_t \in R^{n\times h}$. Whenever the output gate is 1, it effectively passes all memory information through to the predictor, whereas

for output 0, it retain all the information only within the memory cell and perform no further processing:

$$\boldsymbol{H}_t = \boldsymbol{O}_t \odot \tanh\left(\boldsymbol{C}_t\right) \quad (80)$$

This representation is a recurrent neural network with a single unidirectional hidden layer. In search of a more flexible form for the functional function $\hat{f}$, multiple layers of LSTMs can be stacked on top of each other. This results in a mechanism that is more flexible, due to the combination of several simple layers.

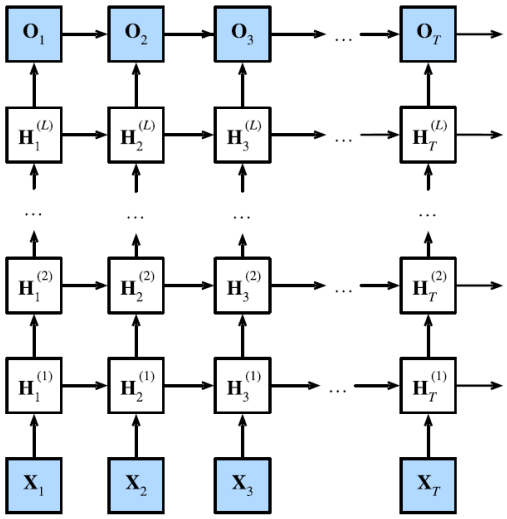


Figure 9 - Architecture of a deep recurrent neural network

Source: Zhang *et al*. (2020)

The hidden state of hidden layer $l(l = 1, \ldots, L)$ is $\boldsymbol{H}_t^{(l)} \epsilon\, R^{n \times h}$ (number of hidden units h), computing the first hidden layer and subsequent layers (as shown in Figure 9):

$$\mathbf{H}_t^{(1)} = \sigma\left(\mathbf{X}_t, \mathbf{H}_{t-1}^{(1)}\right) \quad (81)$$

$$\mathbf{H}_t^{(l)} = \sigma\left(\mathbf{H}_t^{(l-1)}, \mathbf{H}_{t-1}^{(l)}\right) \quad (82)$$

Finally, the output layer is based only in the state of hidden layer L:

$$\mathbf{O}_t = \sigma\left(\mathbf{H}_t^{(L)}\right) \quad (83)$$

The goal now is to choose the weights W and the bias b such that the predictions made at the output gate O_t best fit the true observed data. This is done by defining a loss function and an algorithm capable of searching for the best possible parameters for minimizing that loss function. According to Zhang *et al*. (2020), the most popular optimization algorithms for neural networks is the gradient descent. At each step the gradient descent, for each parameter, evaluates which way the training set loss would move if that parameters were perturbed just a small amount. Then the parameter is updated in the direction that reduces the loss.

The summary below shows the application of the loss function mean squared error for a linear case where the vectors $\mathbf{x}_t \in R^d$ and $\boldsymbol{w} \in R^d$ are:

$$\hat{x} = w_1 x_1 + \cdots + w_d x_d + b \tag{84}$$

A point forecast t is $\hat{x}^{(t)}$ and the corresponding true label is $x^{(i)}$, the squared error is given by:

$$e^t(\boldsymbol{w}, b) = (\hat{x}^{(t)} - x^{(t)})^2 \tag{85}$$

where vector weights $\boldsymbol{w}$ and bias b are the correspondents defined for (75) to (78) and the prediction $\hat{y}^{(t)} = O_t$. To average the loss of the entire dataset of n:

$$E(\boldsymbol{w}, b) = \frac{1}{n}\sum_{t=1}^{n} l^{(t)}(\boldsymbol{w}, b) = \frac{1}{n}\sum_{t=1}^{n}\left(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b - y^{(t)}\right)^2 \tag{86}$$

When training the model, the parameters $(\boldsymbol{w}^*, b^*)$ that minimize the total loss E across all training batches are given by:

$$\boldsymbol{w}^*, b^* = argmin_{\boldsymbol{w},b}\, E(\boldsymbol{w}, b) \tag{87}$$

The gradient descent iteratively reduces the error when updating the parameters in the direction that incrementally decrease the loss function. In this

algorithm, the parameters are initialized at random and in each iteration, a batch size B consisting of a fixed number of training examples, are sampled from the data updating the parameters in the direction of the negative gradient. Finally, a parameter η called learning rate, multiples the gradient value and subtract the resulting term from the current parameter values as follow:

$$(\boldsymbol{w}, b) \leftarrow (\boldsymbol{w}, b) - \frac{\eta}{|\mathrm{B}|}\sum_{i\epsilon \mathrm{B}} \partial_{(\boldsymbol{w},b)} l^{(t)}(\boldsymbol{w}, b) \quad 88$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \frac{\eta}{|\mathrm{B}|}\sum_{i\epsilon \mathrm{B}} \partial_{\boldsymbol{w}} l^{(t)}(\boldsymbol{w}, b) = W - \frac{\eta}{|\mathrm{B}|}\sum_{i\in \mathrm{B}} X^{(t)}\left(\boldsymbol{w}^T \boldsymbol{x}^{(t)} + b - y^{(t)}\right) \quad 89$$

$$b \leftarrow b - \frac{\eta}{|\mathrm{B}|}\sum_{i\epsilon \mathrm{B}} \partial_b l^{(i)}(\boldsymbol{w}, b) = b - \frac{\eta}{|\mathrm{B}|}\sum_{i\in \mathrm{B}} \left(\boldsymbol{w}^T \boldsymbol{x}^{(t)} + b - y^{(i)}\right) \quad 90$$

In neural networks, backpropagation refers to the method of calculating the gradient of neural network parameters. The method traverses the network in reverse order, from the output to the input layer, according to the chain rule from calculus.

In the case of training RNNs such as LSTM, which maintain a hidden state that carries information across time steps, the method is called backpropagation through time. It considers the dependencies between current and past inputs due to the RNN's internal memory and unrolls the RNN through time, treating each time step as a separate layer like in a feedforward network, then performs standard backpropagation across these unrolled layers. The gradients are accumulated across all time steps before updating the weights, ensuring that adjustments consider the impact of past inputs on current outputs. The backpropagation through time full definition can seen at Staudemeyer and Morris (2019) and in Zhang *et al*. (2020).

Table 4 – LSTM Hyperparameters

LSTM formulation hyperparameters	**Loss function hyperparameters**
Input sequence	Batch size
Number of layers	Learning rate
Number of hidden units	Training epochs

Source: Elaborated by the author

In summary, Table 4 shows the hyperparameters needed for both LSTM formulation and its associated gradient descent algorithm to find the weight and bias of the network, These are the network input sequence, the number of hidden layers (L), the number of cells (hidden units) h, the set cardinality $|\mathrm{B}|$ batch size and learning rate η**.** After training for some predetermined number of iterations (the training epochs or until some other stopping criteria are met), the estimated model parameters, denoted $\widehat{W}$, $\hat{b}$, are recorded. These parameters will form the function form $\hat{f}$. Section 3.8 discuss about these hyperparameters and how to set boundaries for the search space.

2.11. Brief history on model calibration and selection

This research is focus in comparing the goodness of fit for the selected models. For that the hyperparameters are selected, the model is fitted using an objective function and then compare their performance using a test dataset and error metrics.

In the 90's, with the expansion of the exponential smoothing methods and ARIMA, researchers proposed the use of meta-learning algorithms that are based on the series statistics (classifying and understanding the nature of historical forecasting data) to decide the best model to use. Time series features identification has been described in the literature already by Armstrong and Collopy (1992). In there they call Rule-based forecasting (RBF) that identifies up to 28 features of time series such as Trend, seasonality and outlies model choices such as random walk, linear regression, Holt's exponential smoothing, and Brown's exponential smoothing. Vokurka, Flores and Pearce (1996), with similar intention, applied simple exponential smoothing, Gardner's damped trend exponential smoothing (Gardner, 1985) and classical decomposition. Adya, Collopy and Armstrong (2001) automated six features that had previously been judgmentally identified in RBF, to reduce the inconsistencies in feature coding that result from differences in the experiences, abilities, and biases of the expert's judgments.

Another focus of defining the best method for a time series forecasting is through experiments by grid search models and its parameters, aiming for the best result of a chosen performance metric.

Certainly, there are obvious limitations for processes based on system-based rules. They are not dynamic and therefore require significant rework and validation

prior to implement and revalidate. This is not a trivial problem especially when the forecasting domain or situation changes (Wang; Smith-Miles; Hyndman, 2009).

After a decade of RBF research that shared hundreds of statistics descriptions to identify the best model and calibrate the parameters, Wang, Smith-Miles and Hyndman (2009) published an approach to select a model based on key time series characteristics including its trend, seasonality, serial correlation, nonlinearity, skewness, kurtosis, self-similarity, chaos, and periodicity using a meta-learning approach. The focus was to classify the series in clusters based on those 9 data characteristics and generate recommendation rules for selection of forecasting methods. Self-organizing maps (SOMs) and decision trees (DTs) are used to induce rules that explains the relationships between these characteristics and forecasting performance.

Their study has been intentionally limited in scope of forecasting methods and in the selection of global features to characterize the structural properties of univariate time series, and therefore does not propose how to narrow down local features to calibrate the chosen model.

Table 5 - Model Selection Publications

Authors	**Method**	**Hyperparameters Selection**
Collopy and Armstrong (1992)	Random- Walk Exponential smoothing	Rule Based Forecasting statistics
Vokurka, Flores and Pearce (1996))	Random- Walk Exponential smoothing	Rule Based Forecasting with graphical support
Adya, Collopy and Armstrong (2001)	Random- Walk Exponential smoothing	Rule Based Forecasting through automated heuristics
Billah, Hyndman, Koehler (2003)	Exponential smoothing	Experiments with new penalty function and reduce forecasting error
Billah *et al*. (2006)	Exponential smoothing	Experiments with three different smoothing approaches

Cont'd Table 5 - Model Selection Publications

Authors	Method	Hyperparameters Selection
Wang, Smith-Miles and Hyndman (2009)	Random-Walk Exponential smoothing ARIMA Neural Network	Meta-learning based on time series characteristics and rule induction.
Lemke and Gabrys (2010)	Exponential smoothing ARIMA Neural Network	Meta-learning based on time series characteristics and rule induction.
Parmezan, Souza and Batista (2019)	Exponential smoothing ARIMA Neural Network	Experiments for parametric and non-parametric models chosen by performance measures
Abbasimehr, Shabani and Yousefi (2020)	Exponential smoothing ARIMA Neural Network	Experiments for selecting best LSTM network topology
Bergastra and Bengio (2012) Rashid, Fattah and Awla (2018) Rere, Fanany and Arymurthy (2016) Gelbart, Snoek and Adams (2014) Liu, Simonyan and Yang (2019)	Neural Networks (but not limited to)	Hyperparameter search techniques based on optimization: Grid Search, Random Search, Metaheuristics, Bayesian Approach, Reinforcement learning

Source: Elaborated by the author

Table 5 provides a summary of the evolution found in publications concerning mostly the 3 models here selected for this study (ES, SARIMA and LSTM) and the methods employed to find the best hyperparameters. Most of publications that approaches model selection criteria for time series laid out the best statistics and optimization methods to find the parameters such as Billah *et al*. (2006) and Billah, Hyndman, Koehler (2003). As for the hyperparameters this is considered as known

(Acar; Gardner, 2012) or was already selected. Error metrics are then used to compare the best model.

Questions remains on what are the different hyperparameters and how to apply it in the dataset? What are the performance metrics and how to measure?

The performance of these models is critically determined by hyperparameters selection. In the last decade, there are growing studies from a rule-based statistics model to hyperparameters selection and calibration that can use optimization algorithms to extensive search for the best configuration according to an objective function.

Let denote a functional form $\hat{f}$ from equation (3) that takes N hyperparameters as $\hat{f}_n$. A vector of hyperparameters is denoted by $\boldsymbol{\nu} \in \hat{f}_n$, and $\hat{f}_n$ with its hyperparameters initiated to $\boldsymbol{\nu}$ is denoted by $\hat{F}_{\boldsymbol{\nu}}$.

The domain of a hyperparameter can be real-valued (e.g., learning rate), integer valued (e.g., number of layers), binary (e.g., whether to use early stopping or not), or categorical (e.g., choice of optimizer). For integer and real-valued hyperparameters, the domains are bounded to feasible values for reduced optimization time (Feurer; Hutter, 2019).

Given a dataset D composed by two partitions, the train D_{train} and validation, $D_{validation}$, the goal is to find the best hyperparameters vector $\boldsymbol{\nu}^*$ that minimize the loss function L:

$$\boldsymbol{\nu}^* = argmin_{\boldsymbol{\nu} \epsilon \hat{f}} L\left(\hat{F}_{\boldsymbol{\nu}}, D_{train}, D_{validation}\right) \quad (91)$$

where the loss function L measures the loss of a model generated by the functional form $\hat{f}$ with hyperparameters $\boldsymbol{\nu}$ on train data D_{train} and evaluated on validation data $D_{validation}$. Considered that any partition of the dataset follows the same probability distribution represented in the entire dataset D $(\mathrm{D}_{train}, D_{validation}) \sim D$. This process can be repeated more than once to validate if the hyperparameters selected were able to generalize the behavior of the entire dataset. The cross-validation, for time series, needs to obey the temporal order:

$$D_{train} = \{t_i, t_{i+1}, \dots, t_{i+n}\} \quad 92$$

$$D_{validation} = \{t_j, t_{j+1}, \dots, t_{j+m}\} \quad 93$$

where j>i, and n,m defines the size of the partitions.

Studies devoted to facilitate this optimization search process includes strategies such as grid search (Feurer; Hutter, 2019), random search (Feurer; Hutter, 2019; Bergastra; Bengio, 2012), evolutionary (Metaheuristic) algorithms (Rashid; Fattah; Awla, 2018; Rere; Fanany; Arymurthy, 2016), Bayesian optimization (Feurer; Hutter, 2019 ; Gelbart; Snoek; Adams 2014), differentiable architecture search (Liu; Simonyan; Yang, 2019), architecture search with reinforcement learning (Rere; Fanany; Arymurthy, 2016), etc.

Two experimental methods for hyperparameter optimization algorithms are the Grid Search and Random Search. Using $\boldsymbol{\nu}$ as a vector of hyperparameters that is composed N hyperparameters for the functional $\hat{f}_N$. Let's define:

$$\boldsymbol{\nu}_N^1 = \{\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \dots, \boldsymbol{\nu}_N\} \quad 94$$

where $\boldsymbol{\nu}_N^1$ is the first set of hyperparameters options for $\hat{f}_N$. Each hyperparameter N have i number of values that it can assume, then the total search space is:

$$\Omega = \prod_{n=1}^{N} i_n \quad 95$$

where i = 1,2,..I the size of the possibilities for hyperparameter n, n = 1,2,…N and Ω is total Grid Search space containing all possible configurations for model training.

In Grid Search, the order in which the hyperparameters vectors will be initiated in the function is pre-defined and all of them will be used. This suffers from the curse of dimensionality since the required number of function evaluations grows with the dimensionality of the configuration space.

A simple alternative to Grid Search is Random Search. It uses sampling of $\boldsymbol{\nu}_N^i$ at random until a certain allowed number for the search is exhausted. This works better than grid search when some hyperparameters are much more important than others (Bergastra; Bengio, 2012).

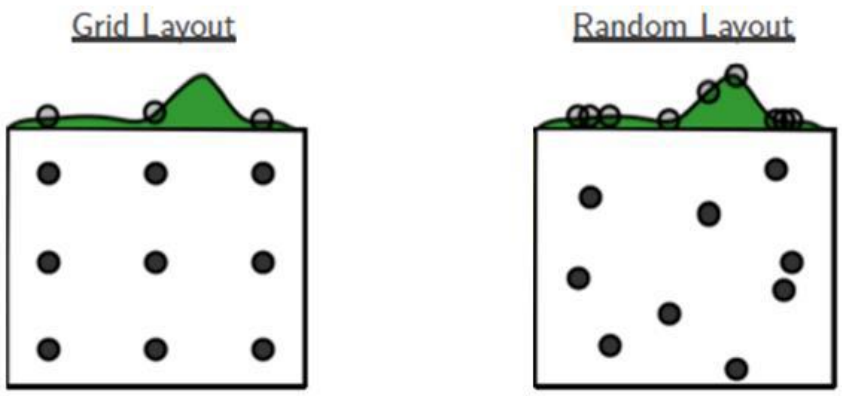


Figure 10 - Comparison of grid search and random search

Source: Adapted from Bergstra and Bengio (2012)

Figure 10 illustrates how hyperparameter space is populated by different search schemes. With Grid search nine trials are tested, and these hyperparameters are defined discretely while in Random Search the same nine trials are explored with hyperparameters defined continuously.

The choice of grid search approach was properly laid out by Bergastra and Bengio (2012) where they describe that this technique gives researchers some degree of insight into hyperparameter response function; There is no technical overhead or barrier to optimization and is simple to implement. On the other hand, depending on the hyperparameter selection, processing power and time is needed for the Grid Search.

The choice for the Grid Search is twofold:

a. Test proposed process in understanding how much computationally expensive will be for the hyperparameters space $\boldsymbol{\nu}$ defined.
b. Evaluate the bounded domain of the hyperparameters $\boldsymbol{\nu}_N$ and if it was properly defined.

In summary, this work proposes process that grid search hyperparameters for the following three models: Holt-Winters Exponential smoothing (ES), SARIMA and LSTM deep learning neural network. The resulting goodness of fit in each model and the selection of the best models are based on empiric work including:

a. Performance in the out of sample (data not used for fitting the model) is measured by a combination error metrics;
b. Observation and analysis of each model fitting diagnostic.

For that, this work relies on the hyperparameters definition based on the literature available for each of the models. The grid search algorithm was elaborated by the author. Once the hyperparameters are applied in each model (ES, SARIMA and LSTM), the calibration of the parameters was implemented using the software package Statsmodels (Seabold; Perktoldet 2010) and Keras (Chollet,2015) in Python programming language. Statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models such as ARIMA, ES, seasonal/trend decomposition using Loess (STL decomposition) and deterministic process using Fourier terms. Keras is an effective high-level neural network Application Programming Interface (API) written in Python. This open-source neural network library is designed to provide fast experimentation with deep neural networks, and has a simple, concise, and readable architecture. Lastly, the prediction performance is measure by the metrics defined by the author and visual diagnostics are provided to analyze the goodness of fit.

3. Methodology

Each section of this chapter demonstrates the steps that were defined to calibrate all three models, select the best hyperparameters and then select the best model.

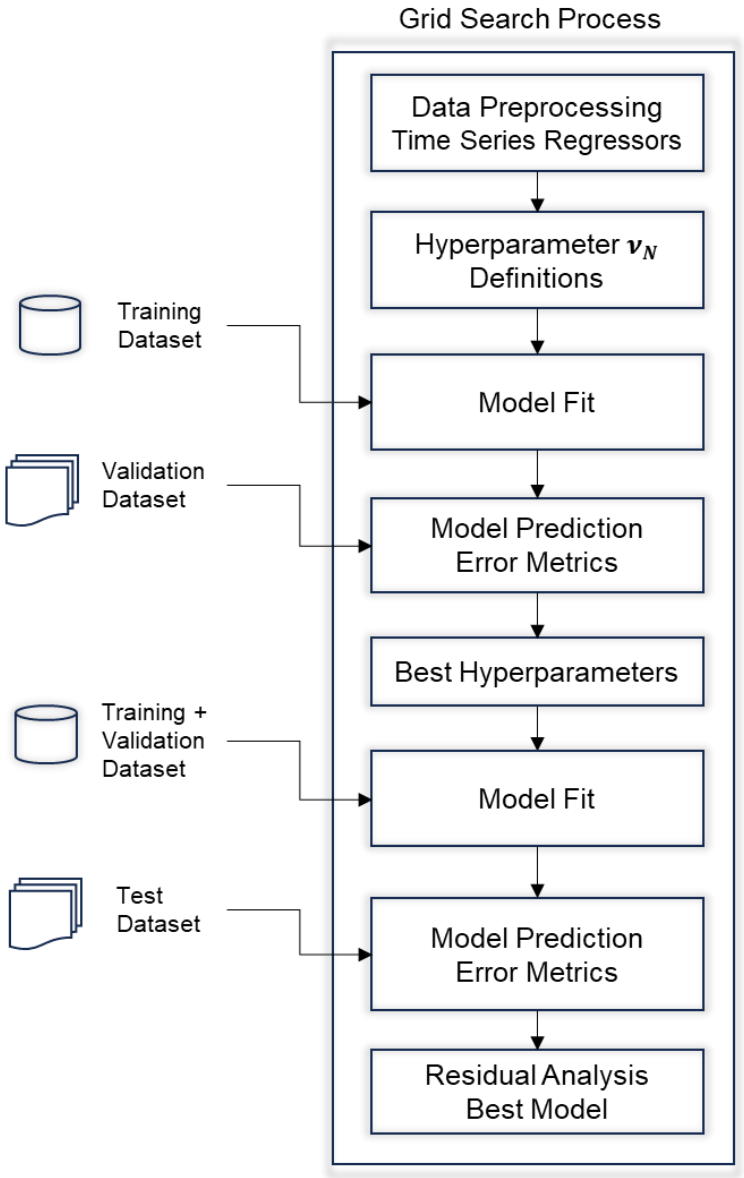


Figure 11 - Complete Grid Search Process for Hyperparameters and Model Selection.

Source: Elaborated by the author

In Figure 11, the complete process is shown. First data is preprocessed for outliers, missing values and periodicity is found. Then the hyperparameters search space are defined for all 3 models. The models are fitted in the training partition and the MCP is measure in the validation partition. Best hyperparameters vectors $\boldsymbol{\nu}^*$ are

selected and implemented in the test partition where the best model with lower MCP is selected. This result is also compared with previous baseline to show the gains with current proposal.

This process relies in an empirical methodology, and it is not limited to the time series in this study, neither is specific for the company where this dataset was extracted. It proposes to be a method to address three main challenges: (1) forecast hundreds of time series using automation, (2) make informed decision when select the best calibration in each model and (3) improve accuracy when comparing to existing baselines. Additionally, this process is also not limited to the three models here selected. Any model can be added once their hyperparameters are identified.

3.1. Data profile

The database of consumer and corporate electronics products for Latin America consists of different product families, each of them with its own time series of daily observations which builds up in hundreds of datasets. For this study a specific product family is selected. The target variable y_t had its original value changed due to confidentiality.

The time series of daily observation is presented in Figure 11. It is intended to forecast the next month daily values. The number of orders needed each day depends on customer demands based on push system which the businesses send products to retailers and distributors in anticipation of consumer purchases. This decision is used to plan supply and human resources to fulfil those orders.

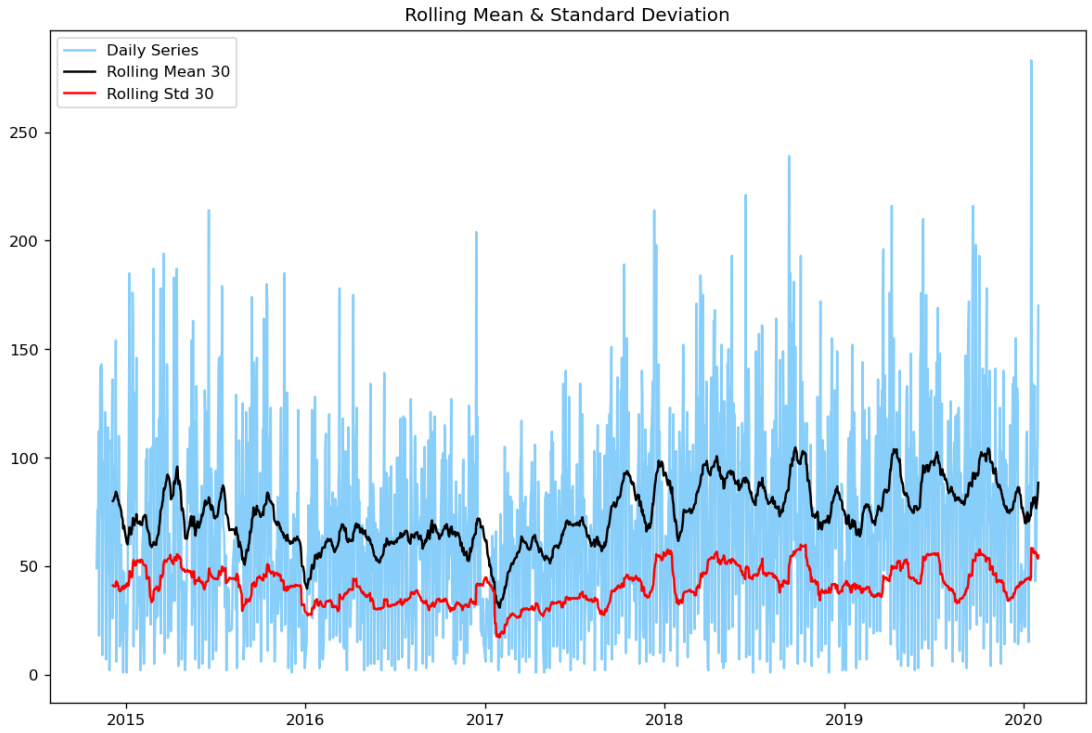


Figure 12 - Daily forecast orders

Source: Elaborated by the author

Figure 12 shows daily observations of orders received for a specific product. In this case it is a 6 days week historical were Sunday no orders were processed. This will be important for the periodicity definition in section 3.2. The period is from November 2014 to January 2020, with a total of 1616 data points. The push production model for this product does not undermines the importance of forecasting accuracy as inventory is built up in anticipation of future demand so, if forecasts are inaccurate, this will generate excess inventory impacting company's financials.

Table 6 – Dataset descriptive Statistics

Measure	**Value**	**Measure**	**Value**
Count	1616	Quartile 1 (25th percentile)	42.00
Mean	73.12	Quartile 2 (50th percentile)	72.00
Median	72.00	Quartile 3 (75th percentile)	100.00
Mode	64.00	Maximum	239.00
Standard Deviation	43.28	Skewness	0.43
Minimum	1.00	Kurtosis	0.06

Table 6 provides the descriptive statistics of the data. Measures of central tendency include the mean, median and mode. The mode is the value that occurs most often in the dataset. Skewness is a measure of a distribution's symmetry and Kurtosis is the degree of peak of a distribution. Both measures are used to insights into the shape and distribution of a dataset and their values are compared with a normal distribution which has skewness of 0 and kurtosis of 3. This dataset has a positive low skew which means that the dataset is fairly symmetrical (positive means longer tail to the right indicating more high values) and very low kurtosis indicating flatter peak and thinner tails of a more spread-out distribution. A flat-topped distribution might simply reflect the inherent nature of the phenomenon being studied here.

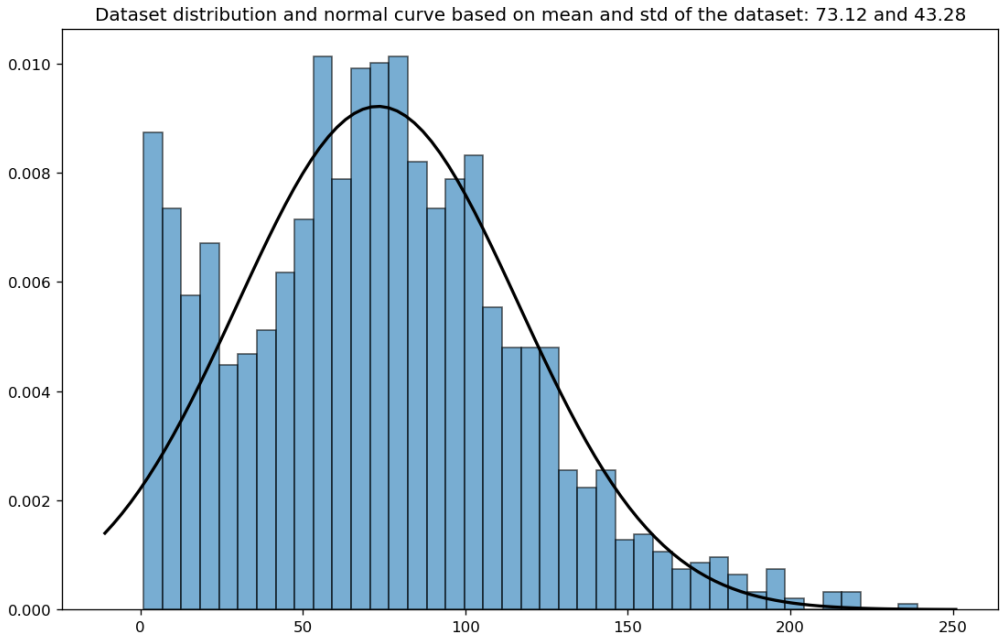


Figure 13 – Histogram chart of the dataset

Source: Elaborated by the author

Figure 13 shows the distribution of the dataset where the number of bins are defined as the squared root of the dataset size. It is possible to see the positive skewness for large values. There is a high number of lower values observed in the left tail of the distribution. The reason for this is the orders received during Saturday, which is an exceptional day for operation, as only urgent orders are processed. The quantity of those lower values is expected to be modelled by the periodicity of the series. This

seasonal pattern, when defined, is applied in all three models as discussed in sections 2.8, 2.9 and 2.10.

There are discussions in scientific community whether transforming the target variable to a normal distribution leads to an improvement in the forecasting accuracy. Nelson and Granger (1979) after fitting a linear ARIMA model to the power transformed series concluded that the Box–Cox transformation does not lead to an improvement in the forecasting performance. In their case the transformation of the non-normality economic data did not show an improvement in the predictions. Makridakis and Hibon (1979) also verified that the transformed data has little effect in improving forecasting. Granger and Newbold (1976) provided a general analytical approach to forecasting transformed series, based on the Hermite polynomials series expansion, and showed that the forecasting of the inverse logarithm transform produces a biased regressor which is carried out to the forecasting.

More recently Proietti and Lütkepohl (2013) discussed this topic and concluded that depending on the partition of the dataset the Box–Cox transformation produces forecasts which are significantly better than the untransformed data at the one-step-ahead horizon. Their case showed a transformation was indicated for 20% of the entire series, however, the advantage of the transformation diminishes for longer-term forecasts. Their method proposes a nonparametric approach for estimating the optimal transformation parameter λ from the Box-Cox transformation based on the frequency domain estimation of the prediction error variance.

Most common transformations are (consider $\mathrm{y_t}$ the time series) the logarithmic $\mathrm{z_t} = \log(\mathrm{y_t})$, power transformations $\mathrm{z_t} = y_t^p$ and the well-developed Box–Cox transform (Box; Cox, 1964). In this work, no transformation was applied when grid search for the best hyperparameters.

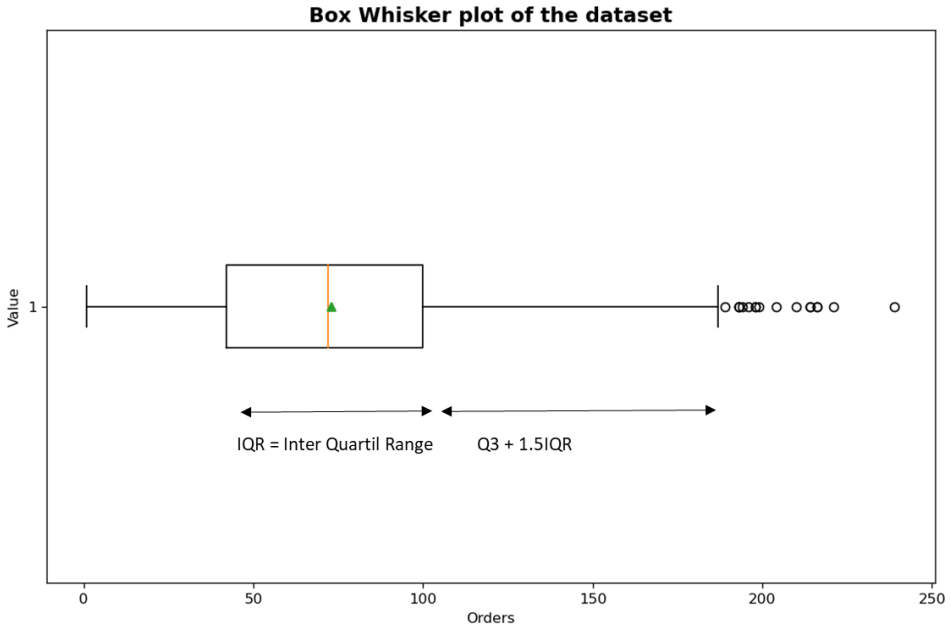


Figure 14 - Box Whisker plot of the entire dataset

To evaluate outliers the Box Whisker plot from Figure 14 uses the inter quartile range from Table 6 to detect outliers by measuring all data points that fall outside the measure Q3+1.5IQR. The large values could be classified as outliers. This is in line with the positive skewness found. The lower values outliers measured by Q1-1.5IQR are then capped at the minimum observed value which is 1 and then no value is classified as outlier in the lower part of the quartile.

The summarized Table 7 below shows additional characterization of the time series of this study:

Table 7 - Time Series Characterization

Characterization Test	**Measure**	**Result**
ADF Test - Stationarity	ADF p-value = 0 (eq 45)	Stationary
Periodicity	6-day week	Yes
Serial Correlation - Ljung-Box test	Ljung-Box test p-value = 0 (eq 48)	Yes

Source: Elaborated by the author

From Table 7 it is concluded that the series does have serial correlation and seasonality that can be used to model with exponential smoothing and SARIMA. The

stationary indicates that for SARIMA differencing is not needed and periodicity of 6 from a 6-day week is further confirmed in section 3.2.

3.2. Outliers and missing data

Outliers are defined as unusually low or high values, occurring due to data errors or special circumstances such as one-off events. These unusual values can be problematic for time series forecasting as they are both based on the extrapolation of historical patterns. Outliers can have a large impact on the forecast if the event is not expected to reoccur in the forecasted period.

Outliers in time series has been reviewed by Chandola, Banerjee and Kumar (2009) as concept of anomaly detection. Here the authors defines and review outliers in the context of time series and describe traditional statistical approaches to identify and remove outliers such as the ones found in the work of Abraham and Chuang (1989) and recent machine learning models like decision trees, neural networks and support vector machines.

A classification of those methods was also reviewed and discussed by Smiti (2020) where the possible approaches to outlier's detection such as statistical, distance-based, density-based and cluster-based methods are shown.

Although, outliers should not be removed if:

a) are caused by natural variation in the data such as seasonal patterns or sales events. Removing these outliers could lead to a biased understanding of the underlying trend and underestimate uncertainty, when the event could potentially reoccur in future;
b) are informative and is of interest to the researcher. Removing these outliers could limit the insights that can be gained from the data.

According to Figure 14, it was decided to remove the points which falls outside the range given by Q3+1.5IQR which are in total 17 points. No additional treatment was applied due to reasons pointed in (a) and (b), especially lower values which are formed by the natural occurrence of the orders in the last day of the week.

Removing these outliers generates missing data points in the time series and adding to other days which data was not available or no orders were processed, yielded a total of 42 missing datapoints.

Many methods exist for imputing missing values ranging from simple univariate methods such as last observation carried forward variation, interpolation, and model-based methods (Moritz; Bartz-Beielstein, 2017) to more complex methods such clusterization (Camacho; Lopez-Buenache, 2022).

The method used for filling the missing datapoint was the average of the weekday from previous years. For instance, if the missing datapoint is a Monday, the average of orders processed in previous Mondays is inputted. It is a simple and intuitive approach that aims to keep the periodicity of the season intact, a characteristic that is part of the ES and SARIMA parametric formulas and an important hyperparameter for LSTM as it will be explained in sections 3.7, 3.8 and 3.9.

For instance, the use of cubic spline interpolation (Moritz; Bartz-Beielstein, 2017), a simple and robust method that creates a smooth and continuous curve passing through all the data points, does not take in consideration the periodicity of the series which creates a seasonality pattern of peaks and drops in opposite of a smooth behavior.

Considering “WD” to be the weekday occurrence in the series and N the total number of occurrences of “WD”, the missing data will be given by:

$$y_t^{int} = \frac{\sum_{i=1}^{N} y_{WD}^{i}}{N} \quad (96)$$

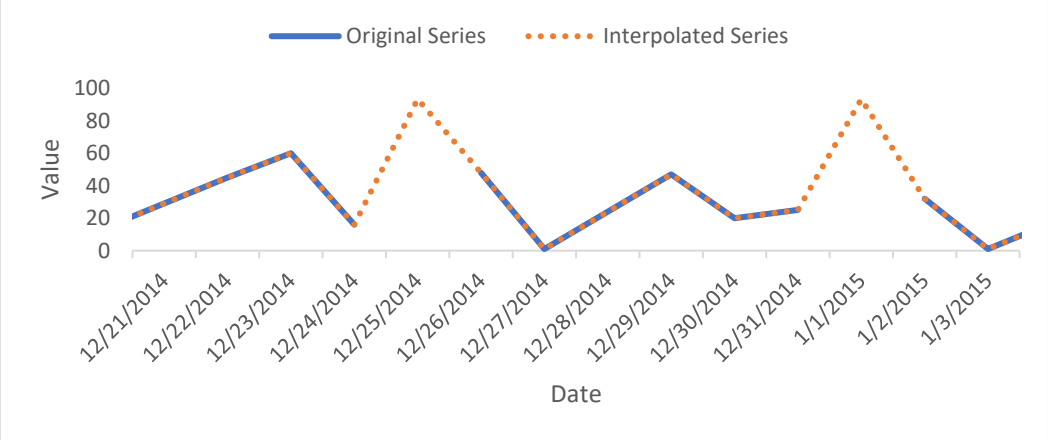


Figure 15 - Original vs Interpolated Series.

Source: Elaborated by the author

Figure 15 shows the interpolated time series for a specific time frame. Replacing missing data keeps the series integrity and therefore helps modelling seasonality and periodicity which is defined next.

Table 8 – Dataset descriptive statistics after missing data treatment.

Measure	**Value**
Count	1616
Mean	71.97
Median	72.00
Mode	59.00 / 64.00
Standard Deviation	41.10
Minimum	1.00
Quartile 1 (25th percentile)	42.00
Quartile 2 (50th percentile)	72.00
Quartile 3 (75th percentile)	99.00
Maximum	187.00
Skewness	0.23
Kurtosis	-0.40

The new data profile seen in Table 8 has it's mean and standard deviation reduced as expected. Also, skewness improved, reducing from 0.43 to 0.23, and a negative small kurtosis shows that the data is less likely to produce extreme values and has a slightly flatter central region compared to a normal distribution.

3.3. Stationarity and periodicity (or frequency)

A stationary time series means that the statistical properties like the mean and variance do not change over time (for example the Stochastic white noise with a mean zero and no correlation between its values at different times). Thus, time series with trends or with seasonality are not stationary and its properties will affect the value of the time series at different times.

Stationary time series is often done for traditional time series methods such as ARIMA, using differences or seasonal differences which is embedded in the model hyperparameters that will be calibrated. For regression-based machine learning

methods the time series will be transformed to a supervised learning dataset which is defined in section 3.9.

Periodicity (or frequency) for time series is the number of observations before the seasonal pattern repeats. Time series can often exhibit complicated seasonal pattern which leads to unknown frequencies or regular periodicity.

Periodicity can be determined by looking in the auto-correlation function after removing the trend of the series. Here are the steps implemented:

a) Differencing: remove the trend by compute the n-th differences between consecutive observations according to equation (45);

b) Find the autocorrelation function r_k for all the series length Y_k:

$$r_k = Corr(Y_k, Y_{t-k}) \quad 97$$

c) Plot r_k and chose the periodicity to be the peak with highest of all positive values which has trough before it; in the case of the time series of this work the periodicity was clearly 6 (6-day week seasonal pattern).

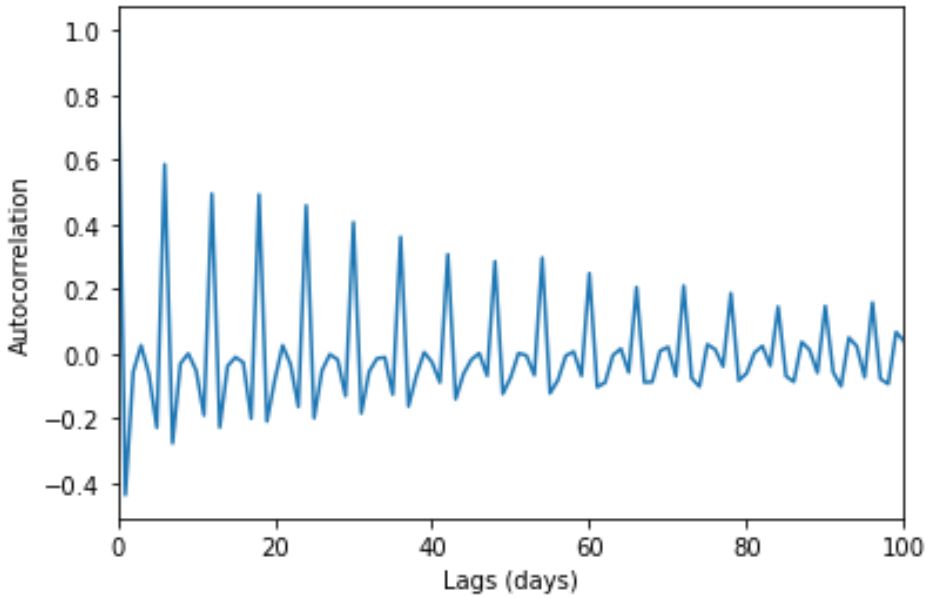


Figure 16 - Result of the periodicity calculation

Source: Elaborated by the author

In Figure 16, the peaks at every 6 lags can be seen. If no such peak is found, periodicity can be set to 1 (equivalent to non-seasonal). Huber and Stuckenschmidt (2020) presented a large-scale demand forecasting scenario that requires daily forecasts. A challenge in this case is predicting the demand on special days (e.g.,

public holidays). Their empirical evaluation provides evidence that machine learning methods were a viable alternative over traditional methods. The periodicity of 6 is not a surprise result as this time series consists of calendar weeks of 6 working days of sales.

3.4. Trend and seasonality modelling using time series decomposition.

The objective in this section is to model the trend and seasonality deterministically using equation (22). To validate how well seasonality was modelled, two ACF plots will be used: the stationary original series ACF versus the residuals ACF from fitted series. If a good model is obtained, the objective is to use it as additional regressor in the SARIMA and LSTM modelling that could lead to improvement in model fitting and forecasting.

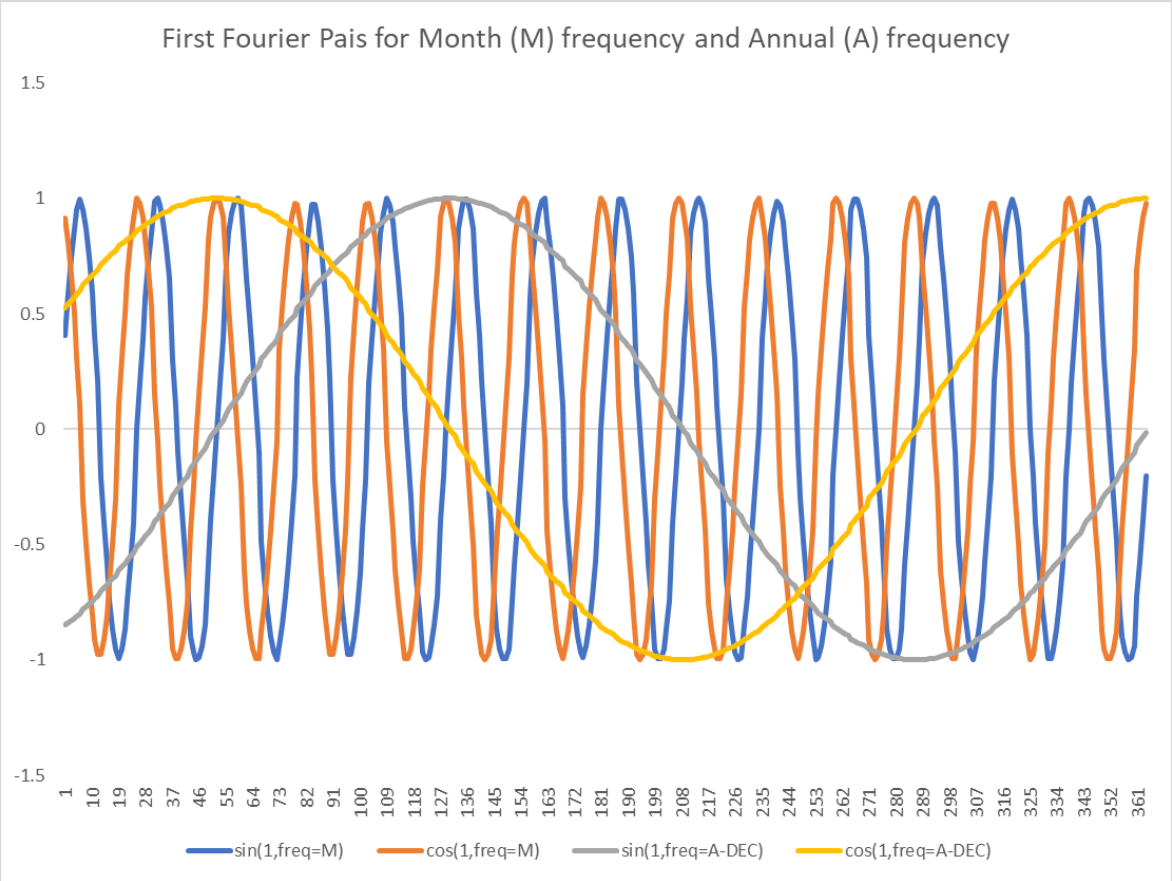


Figure 17 - Sine and Cosine for two Seasonality frequencies

Source: Elaborated by the author

Figure 17 shows the plot of k=1 for equation (22) of the sine and cosine pair in a 365-time steps expansion. This figure also compares how the seasonality differs when using m_A and m_M. The figure is limited by 365-time steps but this deterministic process expands out to the entire dataset horizon. Additionally, the dataset has a 6-day week, so the extra terms created for the full month and year seasonality are dropped.

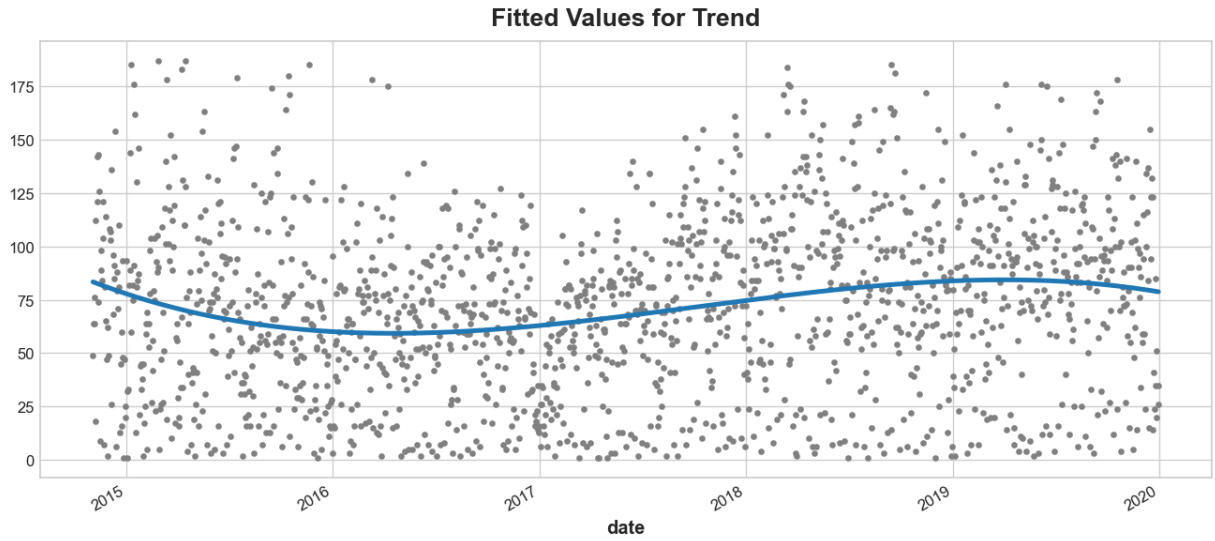


Figure 18 - Trend modelled with polynomial.

Source: Elaborated by the author

For the polynomial equation it was selected the order p=3 to allow capturing both upward and downward trend as seen in Figure 18, and the weekly variables are given by j=1,…5, following a 6-day-week series. The two candidate equations are then:

$$\hat{Y}_t = \sum_{p=0}^{3} \delta_p t^p + \sum_{k=1}^{12} [\alpha_k \sin\left(\frac{2\pi k t}{m_A}\right) + \beta_k \sin\left(\frac{2\pi k t}{m_A}\right)] + \sum_{j=1}^{5} \gamma_k d_{j,t} + Z_t \quad 98$$

$$\hat{Y}_t = \sum_{p=0}^{3} \delta_p t^p + \sum_{k=1}^{12} [\alpha_k \sin\left(\frac{2\pi k t}{m_M}\right) + \beta_k \sin\left(\frac{2\pi k t}{m_M}\right)] + \sum_{j=1}^{5} \gamma_k d_{j,t} + Z_t \quad 99$$

Both equations (98) and (99) are fitted to the entire series and the coefficients are found through equation (23). The resulting residuals are then:

$$Y_{t,deseason} = Y_t - \hat{Y}_t \quad (100)$$

$$Y_{t,std} = \frac{Y_{t,deseason}}{\sqrt{\sum_t (Y_{t,deseason})^2 / T}} \quad (101)$$

Equation (101) are the standardized residuals. This approach is useful to analyze on common scale, so they can be easily compared across different models.

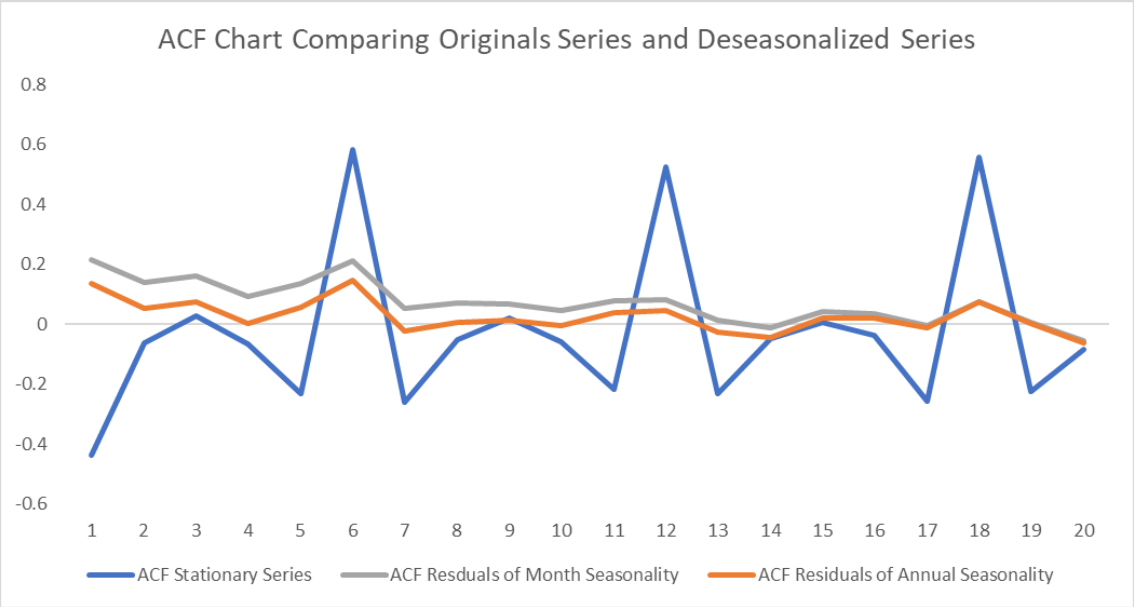


Figure 19 - ACF plot comparing the original series with the de-seasonalized series.

Source: Elaborated by the author

After the regression coefficients of equation (98) and (99) are found, the residuals are calculated in equation (100). The goal now is to evaluate how good the seasonality of the series was capture by measuring the autocorrelation function from equation (27). The plot of the autocorrelation in each lag from the de-seasonalized series using m_A and m_M can be compared with the autocorrelation from the original series. Figure 19 shows how m_A had better result in capturing the correlation with previous lags (orange line), where the same peaks found in the original series (blue line) are not seen or have a much lower value. Selecting the annual harmonic seasonality, equation (98) regression resulted in F-Statistic of 63.7 which rejects the null hypothesis in equation (47) that all coefficients are null.

Table 9 - Coefficients Found from Equation (98)

Trend	Coefficients	P>|t|
const	84.2724	0.00%
trend	-0.11	0.00%
trend_squared	0.0001	0.00%
trend_cubed	-4.183E-08	0.00%
s(2,7)	7.6897	0.10%
s(3,7)	13.3394	0.00%
s(4,7)	21.1934	0.00%
s(5,7)	21.4299	0.00%
s(6,7)	-55.3338	0.00%
cos(1,freq=A-DEC)	-4.0144	0.00%
cos(2,freq=A-DEC)	-5.6703	0.00%
cos(3,freq=A-DEC)	-2.8667	0.20%
sin(4,freq=A-DEC)	-5.2691	0.00%
cos(4,freq=A-DEC)	2.9148	0.20%
cos(5,freq=A-DEC)	-2.4967	0.80%
cos(6,freq=A-DEC)	-2.251	1.60%
cos(7,freq=A-DEC)	-3.8418	0.00%
sin(8,freq=A-DEC)	4.4676	0.00%
cos(9,freq=A-DEC)	-2.3642	1.10%
cos(10,freq=A-DEC)	-1.8182	5.00%
sin(12,freq=A-DEC)	7.8554	0.00%
cos(12,freq=A-DEC)	-14.8429	0.00%

Source: Elaborated by the author

Table 9 shows the coefficients values for $\alpha_k, \beta_k, \gamma_k, \delta_p$ created during the regression from Equation (98) and respectively p-values from Equation (25). Also, there is a total of 32 coefficients resulting from this regression, in which some coefficients had a p-value greater than or equal to 5% (from equation (24)). Backward Stepwise Selection is a technique that iteratively removes the least useful predictor, one-at-a-time and fit the regression once more. This has reduced the number of

coefficients to 21 and increased the F-Statistics to 96. The freq=A-DEC represents the frequency of annual for the Fourier pairs of the deterministic process.

Figure 20 show a specific timeframe of the series comparing the resulting regression line of the trend and seasonality modelled (fitted values in blue) against the original series (in black).

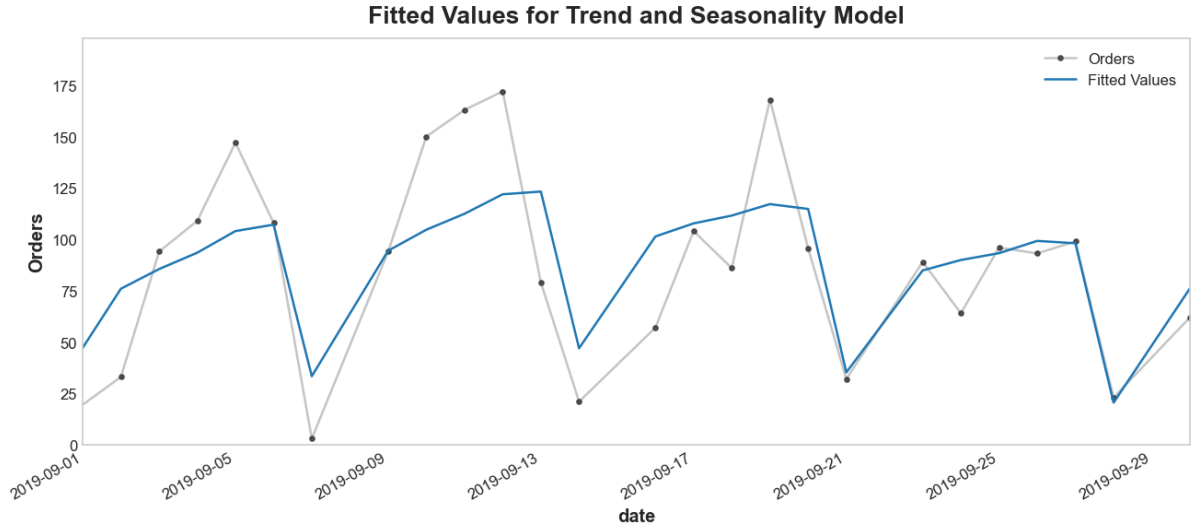


Figure 20 - Fitted values for Equation (99) vs Original Series

Source: Elaborated by the author

The resulting $\hat{Y}_t$ of the decomposition model are used as one regressor for the time series. Additionally, it was added other series (exogenous variables) that will play the role of additional regressor. These regressors has the time series with same interval as the target variable like shown in Table 2. Both SARIMA and LSTM can use multiple regressors to try improving the fitting of the target variable (orders) according to their formulation seen in 2.8 and 2.10.

These additional features represent business behavior that correlates with orders forecasting and their future data is known such as production capacity, company financial targets and material availability. A comparison of the error metrics when using these additional regressors can confirm their influence.

3.5. Training, validation and test dataset partition

When evaluating a model, the performance of the model is done on data that was not used for training it. As explained in section 2.2, this is the validation partition.

The size of the dataset in this study provides an opportunity to perform the validation in three steps:

a) Training set: This is used to fit the models with the chosen hyperparameters and the objective function. The grid search space is used for each model in order to compare the hyperparameters vector's $\boldsymbol{v}_N^i$ performance during the validation phase;
b) Validation set (out of sample): Evaluate how well the model was trained by comparing the performance of the fitted models. The hyperparameters vector for each model that had the best validation result are selected (best candidates for the test set);
c) Test set: How the best candidates for each model perform on unseen real-world data. They are applied in the test set and each model performance is compared.

This methodology involves moving along the time series adding the validation partition to the training partition at each split. For that two options are possible: sliding window or expanding window. The difference between these two can be seen in Figure 21. A sliding window hold the length of the dataset constant as new data is available while the expanding window increase the dataset with the new data.

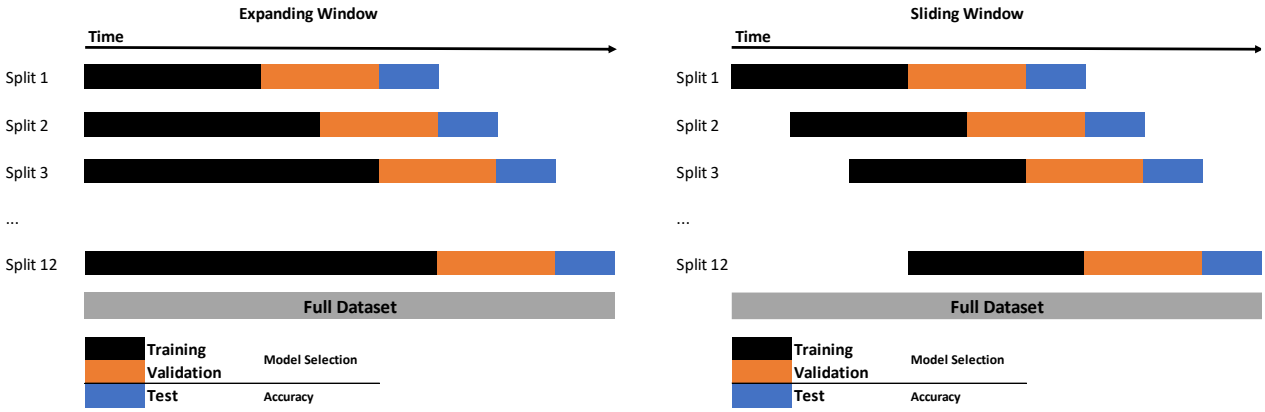


Figure 21 - Expanding window and Sliding Window approach.

Source: Elaborated by the author

The expanding window approach was selected for two reasons: according to Staudemeyer and Morris (2019) LSTM is designed to capture long term dependencies

and second the regressor created with Equation (99) was used to capture the entire seasonality for the database.

The size of each partition on a, b and c are defined as follow. The test set is used to validate a real case scenario, which for this study is a month prediction made of 6-day week. Next it is selected 12 months of test set to replicate a full year of monthly forecasts. The Validation set will contain about 6 months of data, which it will be fixed as 10% of the total records subtracted of the test set in each split.

Table 10 - Size of each partition: training, validation and test for 12 splits

Expanding Window							
Split	Total Records	Training	Validation	Test	Training %	Validation %	Test %
1	1330	1173	130	27	88%	10%	2%
2	1354	1197	133	24	88%	10%	2%
3	1380	1219	135	26	88%	10%	2%
4	1406	1242	138	26	88%	10%	2%
5	1433	1265	141	27	88%	10%	2%
6	1458	1290	143	25	88%	10%	2%
7	1485	1312	146	27	88%	10%	2%
8	1512	1336	149	27	88%	10%	2%
9	1537	1361	151	25	89%	10%	2%
10	1564	1383	154	27	88%	10%	2%
11	1590	1408	156	26	89%	10%	2%
12	1616	1431	159	26	89%	10%	2%

Source: Elaborated by the author

From Table 10, each model will process is as follow:

a) Hyperparameter is selected from search space. Model is fitted;
b) The model makes a prediction for the validation partition, the performance is measured and stored.
c) Next Hyperparameter is selected, fitted and validation performance is stored.
d) When all the 12 splits of the validation dataset are done, the best hyperparameters in each split, for each model, are selected.

e) Those best hyperparameters are then used from split 1 to 12 with the training window expanded to include the validation partition and their performance are measured in the test partition.
f) The model is chosen based on the best performance in the test partition from all 12 splits.

3.6. Training and out of sample metrics

To evaluate the performance of the models during training, validation (out of sample) and testing (out of sample), several statistical metrics can be used (Shcherbakov *et al*. 2013). The survey showed in Table 11 were selected and these measures were classified according to its objective; for model performance or for model selection. Based on that, a combination of these metrics is then proposed for selecting the best model. In Table 11, y_t is the original series, $\hat{y}_t$ the forecasted values and y_t^* the benchmark values.

Table 11 - Evaluation Metrics for validation and testing partitions.

Measure	Formula	Model Performance	Model Selection
AIC (Training)	$AIC = -2\ln(Max\ Likelihood) + 2k$	X	X
BIC (Training)	$BIC = -2\ln(Max\ Likelihood) + kln(n)$	X	X
Mean Absolute Error	$MAE = \frac{1}{T}\sum_{1}^{T}\lvert y_t - \hat{y}_t\rvert$	X	
Mean Squared Error	$MSE = x_1 = \frac{1}{T}\sum_{1}^{T}(y_t - \hat{y}_t)^2$	X	
Symmetric Mean Absolute Percentage Error	$SMAPE = \frac{1}{T}\sum_{1}^{T}\frac{\lvert y_t - \hat{y}_t\rvert}{(\lvert y_t\rvert + \lvert\hat{y}_t\rvert)/2}$	X	
Mean Absolute Scaled Error	$MASE = \frac{1}{T}\sum_{1}^{T}\frac{\lvert y_t - \hat{y}_t\rvert}{(\frac{1}{T-1})\sum_{t=2}^{n}\lvert y_t - y_{t-1}\rvert}$	X	X

Cont'd Table 11 - Evaluation Metrics for validation and testing partitions

Measure	Formula	Model Performance	Model Selection
Median Relative Absolute Error	$MdRAE = median\|r_t\|$ $r_t = \frac{(y_t - \bar{y_t})}{(y_t - y_t^*)}$		X
Geometric Mean Relative Absolute Error	$GMRAE = x_2 = \sqrt{\prod_{t=1}^{T} \frac{\|y_t - \hat{y}_t\|}{\|y_t - y_t^*\|}}$		X
Mean Direction Accuracy	$MDA = x_3$ $= 1 - \frac{1}{T}\sum_t 1_{sgn(y_t - y_{t-1}) = sgn(\hat{y}_t - \hat{y}_{t-1})}$	X	X
Multi-Criteria Performance (MCP)	$MCP = (x_1x_2 + x_1x_3 + x_2x_3)\frac{\sin\left(\frac{2\pi}{3}\right)}{2}$ where x_i are MSE, GMRAE and 1-MDA.		X

Source: Elaborated by the author

Table 11 shows the error metrics evaluated for this work and are not limited. AIC is the Akaike's Information Criterion and was one of the first approaches to model specification and it proposes a balance between goodness-of-fit and model complexity. The Schwarz Bayesian Information Criterion (BIC) also penalizes model complexity, but the penalties grow faster with increasing sample size and model complexity. A full definition of both measures is found in Brockwell and Davis (2016). Both metrics are limited to the training partition as they leverage the result of the maximum likelihood function that is used to find the parameters of the model as discussed in section 2.9.

In this work, the selection of the best hyperparameter vector in each model is done by choosing the best result found in the validation (out of sample) partition even though it might not be the best result found during the training partition. The reason for this is two:

a) Different hyperparameters can achieve similar training results;
b) Better result in out of sample cloud mean that the hyperparameters achieved a better generalization of the dataset. The test partition will then confirm this assumption.

For this task, the other metrics in Table 11 beyond AIC and BIC can be selected. They range from absolute error measurement such as MAE and MSE to relative error to a benchmark such as MdRAE and GMRAE.

In particular, the use of percentage errors is not advisable in the dataset of this study because of the considerable proportion of low actual values, which lead to high percentage errors with no direct interpretation or practical use. Another disadvantage of percentage error is that it puts a heavier penalty on forecasts that exceed the actual than on those that are less than the actual value (Armstrong; Collopy, 1992; Shcherbakov *et al*. 2013)

On the other hand, the Symmetric Mean Absolute Percentage Error is not affected by the scale of the data and does not penalize over forecasting or under forecasting due to average taken of the both the observed and predicted value. SMAPE is also expressed as a percentage.

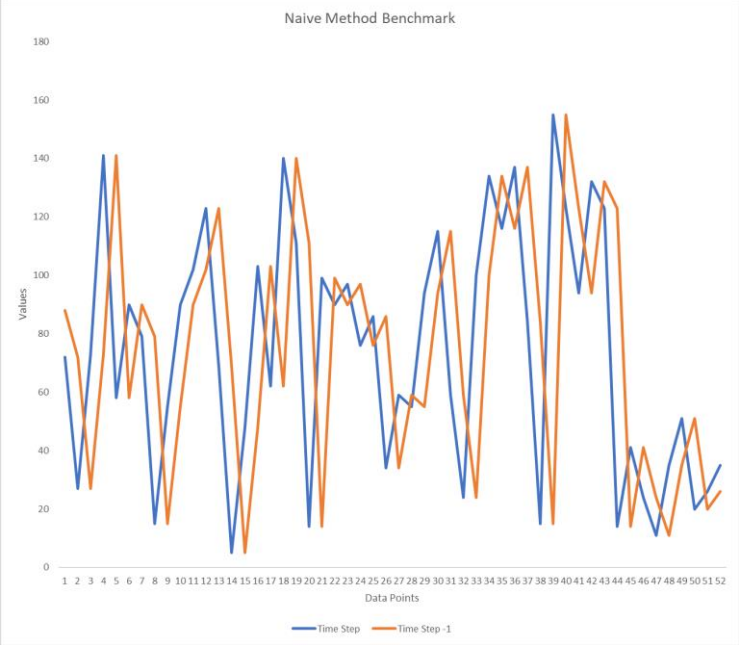


Figure 22 - Representation of Naïve benchmark

Source: Elaborated by the author

For the relative error metrics, the benchmark value chosen was the previous time series step observation, also known as Naïve Method, as seen in Figure 22. This method uses the most recent value as the forecasted value for the next time step. To

do this, the benchmark value is simply one time step shifted forward and assigned as the forecast. It is an elementary baseline that can be hard to beat.

The result expected when choosing a model measure both performance and selection type of errors (Table 8). For this reason, a combination of those metrics was used in what is defined as multi-Criteria Performance index. A proposal of this metric was introduced by Parmezan, Lee and Wu (2017).

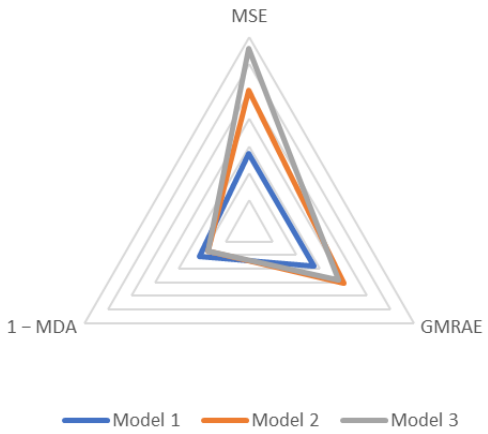


Figure 23 - Representation of MCP criteria

Source: Elaborated by the author

The multi-criteria performance index from Figure 23 is defined to be the area of the triangle which the vertices are the values for three chosen metrics. Although all metrics are measured, only three selected metrics will be used for the best hyperparameter $\boldsymbol{\nu}_N$ selection in the validation phase and for the best model in the test phase. The selection was:

a) MSE: Selected as model performance, the second moment of the error incorporates both the variance of the estimator (how widely spread the estimates are from one data sample to another) and its bias (how far off the average estimated value is from the true value). The resulting MSE value is then scaled to match the scale of the next two criteria.
b) GMRAE: For a relative error measure, both MdRAE and GMRAE are good candidates as model selection error metric. This measure will show the effect on accuracy when a change is made in a hyperparameter for a given model. Geometric mean error measure is more sensitive than median. Also values higher

than 1 for this error indicates that the naïve benchmark starts to be a better predictor.

c) MDA: Compares the forecast direction (upward or downward) to the actual direction. Not only the precision of the forecast but the level and direction of the predictions are important in planning for future orders. Values range from 0 to 1.

Choosing the best model can be a difficult task when different performance measures are available. The Multi Criteria Performance is a proposal to work around such problem. In addition, accuracy only, such as MSE is not the only goal. The goal is also to select a model that can perform better than a simple naïve forecast. Correcting predicting increase or decreasing in the sales levels are important to plan production accordingly and in the case of this study that impact is daily.

For parametric models ES and SARIMA, additionally to the MCP, the estimators resulting from the maximum likelihood function will be evaluate by their statistical importance by looking at the standard errors. Given the probability p-value found in each coefficient, only values less than 5% for each coefficient will be accepted for the model.

Finally, the concept of confusion matrix (Zhang *et al*.; 2020) is leveraged from machine learning classification task where the goal is to predict discrete categories or classes. Confusion matrices capture how well the model distinguishes between these classes. While they aren't directly applicable to regression problems, it is possible to adapt them to visualize and analyze model performance.

The confusion matrix, $\boldsymbol{C_{kxk}}$, is simply k x k matrix, where each column corresponds to the label category (ground truth) and each row corresponds to the model's predicted category. Each cells value c_{ij} is the fraction of total predictions on the validation set, where the true label was "j" and our model predicted "i".

Regression tasks involve predicting continuous numerical values, so a discretization of the continue values are needed by separating them in bins. This approach can reveal patterns in the model's accuracy across different value ranges. A heatmap is then plot comparing actual and predicted values.

Table 12 – Prediction bins classification for confusion matrix

Measure	Value	Bins
Minimum	1.00	
Quartile 1 (25th percentile)	42.00	(0,42]
Quartile 2 (50th percentile)	72.00	(42,72]
Quartile 3 (75th percentile)	99.00	(72,99]
Maximum	187.00	(99,187]

Source: Elaborated by the author

For this, the bins are defined using according to the quartile distribution of series from Table 8, where a bin is defined by the limits given in each category. For instance first bin (0,42] (where "(" means exclusive and "]" inclusive) is the range between the minimum observed value and quartile 1. All true values from the validation partition that falls into this bin will be compared with the bin that the predicted values were classified. Using the quartile of the data for the bin classification ensures that the data is balanced across all the bins.

Table 13 – Confusion Matrix based on true/predicted bins

True Value	(0, 42]	(42, 72]	(72, 99]	(99, 185]
(0, 42]	cij	cij	cij	cij
(42, 72]	cij	cij	cij	cij
(72, 99]	cij	cij	cij	cij
(99, 185]	cij	cij	cij	cij

Predicted Value

From Table 13, the true bin "j" and predicted bin "i" is counted and the matrix populated. The diagonal sum provides the total correct number of values in the correct bin which is a measure of the accuracy of the model.

Some key confusion matrix metrics includes accuracy, which calculates the overall percentage of correct predictions, precision that tells the percentage of predictions for a specific class that were correct and recall which represents the percentage of actual instances within a class that the model correctly identified. The goal is the overall performance as there is no specific bin from table 13 that is more important than the other.

For the predicted values in the validation and test partition, there is a fundamental difference between parametric models ES/SARIMA and Non-Parametric LSTM that is defined next. As mentioned in section 2.10, LSTM requires a definition of the input sequence dimension to start the training process, which will be defined as hyperparameter itself in section 3.9. To exemplify this, consider a time series $\boldsymbol{y_t}$ of size t=10 which t=1..7 is used for training and t=8,…10 for validation. In this example, for LSTM, the input sequence dimension will have a dimension of 3 time steps.

Table 14 - Fit and Prediction for Parametric Models

						Prediction	**True Value**
	$\boldsymbol{y_{t-2}}$	$\boldsymbol{y_{t-1}}$	$\boldsymbol{y_t}$			$\boldsymbol{\hat{y}_{t+1}}$	$\boldsymbol{y_{t+1}}$
Model Fit	$\boldsymbol{y_1}$	…	$\boldsymbol{y_7}$			$\boldsymbol{\hat{y}_8}$	$\boldsymbol{y_8}$
Model Prediction	$\boldsymbol{y_1}$	…	$\boldsymbol{y_7}$	$\boldsymbol{\hat{y}_8}$		$\boldsymbol{\hat{y}_9}$	$\boldsymbol{y_9}$
Model Prediction	$\boldsymbol{y_1}$	…	$\boldsymbol{y_7}$	$\boldsymbol{\hat{y}_8}$	$\boldsymbol{\hat{y}_9}$	$\boldsymbol{\hat{y}_{10}}$	$\boldsymbol{y_{10}}$

Source: Elaborated by the author

In Table 14 Parametric models will fit to the entire training size to find the parameters given by Appendix A and Equation (65) for ES and SARIMA respectively. For the prediction values, parametric models will use the predicted values on y_{t+1} to forecast y_{t+2} and then use y_{t+1} and $\mathrm{y_{t+2}}$ to forecast y_{t+3} and so on.

Table 15 - Fit and Prediction for LSTM

				Prediction	**Validation**
	$\boldsymbol{y_{t-2}}$	$\boldsymbol{y_{t-1}}$	$\boldsymbol{y_t}$	$\boldsymbol{\hat{y}_{t+1}}$	$\boldsymbol{y_{t+1}}$
Model Fit	$\boldsymbol{y_1}$	$\boldsymbol{y_2}$	$\boldsymbol{y_3}$	$\boldsymbol{\hat{y}_4}$	$\boldsymbol{y_4}$
Model Fit	$\boldsymbol{y_2}$	$\boldsymbol{y_3}$	$\boldsymbol{y_4}$	$\boldsymbol{\hat{y}_5}$	$\boldsymbol{y_5}$
Model Fit	$\boldsymbol{y_3}$	$\boldsymbol{y_4}$	$\boldsymbol{y_5}$	$\boldsymbol{\hat{y}_6}$	$\boldsymbol{y_6}$
Model Fit	$\boldsymbol{y_4}$	y_5	y_6	$\boldsymbol{\hat{y}_7}$	y_7
Model Prediction	$\boldsymbol{y_5}$	$\boldsymbol{y_6}$	$\boldsymbol{y_7}$	$\boldsymbol{\hat{y}_8}$	$\boldsymbol{y_8}$
Model Prediction	$\boldsymbol{y_6}$	$\boldsymbol{y_7}$	$\boldsymbol{\hat{y}_8}$	$\boldsymbol{\hat{y}_9}$	$\boldsymbol{y_9}$
Model Prediction	$\boldsymbol{y_7}$	$\boldsymbol{\hat{y}_8}$	$\boldsymbol{\hat{y}_9}$	$\boldsymbol{\hat{y}_{10}}$	$\boldsymbol{y_{10}}$

Source: Elaborated by the author

In Table 15 LSTM will find the weights given by Equation (75) to (78), moving along the training size with 3 times steps lag sequence. For LSTM, the size of the Input sequence dimension is used fixed as a sliding window until all the forecasted values for the validation partition are done.

Both iteration procedure from Tables 14 and 15 was applied to validation and test datasets shown in Table 10. The metrics from Table 11 are measured based on the forecasted values of the validation period and test period against the respective true value.

3.7. Exponential smoothing – Holt-Winters method hyperparameters

The Holt Winters method (Holt, 1957 and Winters, 1960), belongs to a class of exponential smoothing methods that aims to capture the behavior of the time series, separating it in trend, season and error terms.

There are two variations to this method that differ in the nature of the seasonal component. The additive method is preferred when the seasonal variations are roughly constant through the series, while the multiplicative method is preferred when the seasonal variations are changing proportional to the level of the series (Hyndman; Athanasopoulos, 2018).

By considering variations in the combinations of the trend and seasonal components, Hyndman *et al*. (2018) showed that fifteen exponential smoothing methods are possible to simulate, which are listed in Table 11 below. This taxonomy was used in the simulation of this work to calibrate and choose the best hyperparameter vector $\boldsymbol{\nu}_N$ and was introduced by Makridakis, Wheelwright and Hyndman (1998).

Table 16 - Taxonomy for Holt-Winters Exponential smoothing

Trend Component	**Seasonal Component**		
	N (None)	A (Additive)	M (Multiplicative)
N (None)	(N,N)	(N,A)	(N,M)
A (Additive)	(A,N)	(A,A)	(A,M)
Ad (Additive damped)	(Ad,N)	(Ad,A)	(Ad,M)
M (Multiplicative)	(M,N)	(M,A)	(M,M)
Md (Multiplicative Damped)	(Md,N)	(Md,A)	(Md,M)

Source: Makridakis, Wheelwright and Hyndman (1998) and Hyndman and Athanasopoulos (2018).

As the damping parameter (value from 0 to 1) can only be applied to the trend component, the N(None) option is removed. The trend component is damped so it flattens over time instead of being linear. Therefore, the hyperparameters to be considered will be:

Table 17 - Possible hyperparameters for ES (Holt-Winters) Model

Hyperparameters - $\boldsymbol{\nu_N}$	**Value - i_n**
Trend	Additive, Multiplicative
Damping	True or False
Seasonal	Additive, Multiplicative or None
Seasonal Frequency	6
Damping phi parameter	0.2, 0.4, 0.6, 0.8, 0.95

Source: Elaborated by the author

From Table 17, all hyperparameters for ES model are defined. From equation (94), N = 6 consisting of all hyperparameters of the vector $\boldsymbol{\nu_N}$ for fitting an ES model and from equation (95) the total grid search space $\Omega = 120$ configurations. For the case of “Damping = False”, the configurations which a damping value was applied to equation (95) are removed, which results in $\Omega = 60 - 24 = 36$ configurations to be fitted and evaluated in the validation partition.

The number of hyperparameters subsets to use is not cumbersome from a time processing standpoint. Here all combinations can be fitted. For SARIMA the use of the correlograms helps define the boundaries while LSTM some hyperparameters will need the use of experiments to define optimal boundaries. This is shared next.

3.8. SARIMA method hyperparameters

In section 2.8 it was showed that the Box–Jenkins Autoregressive Integrated Moving Average (ARIMA) model (Box; Jenkins, 1976) has two variations, the general non-seasonal model is written as ARIMA(p; d, q); where p is the order of auto-regression (AR), d is the degree of first differencing involved, and q is the order of moving average (MA); while the seasonal model is an extension written as

SARIMA(p; d; q)(P;D;Q)s where s denote the periodicity and P, D and Q are seasonal equivalents of p, d and q.

Table 18 - Possible hyperparameters for SARIMA Model

Hyperparameters - ν_N	**Value - i_n**
Autoregressive Order	1,2….,p
Differencing Order	0,1,2
Moving Average order	1, 2,,…,q
Seasonal Autoregressive Order	0, 1,…,P
Seasonal Differencing Order	0,1,2
Seasonal Moving Average order	0, 1,…,Q
Seasonal Frequency	6

Source: Elaborated by the author

The possibilities of the grid search space are a lot higher for this model than for exponential smoothing. To narrow down possible candidates for each of the hyperparameters (autoregressive, stationarity, moving average and seasonal equivalents) the ADF test from Equation (43) is used, and the correlograms from Equation (26) and (38) are used to define the optimal boundaries:

a) Augmented Dickey Fuller Test: The ADF test resulted that the series is stationary. The regression result showed a positive trend in Figure 17. For that the differencing order options will be 0 and 1;
b) Autocorrelation function (ACF): Consider the series stationary in the mean and the variance, by measuring the autocorrelation in Equation (26) for different number of time steps, the lags which are significant are defined for the hyperparameter. This is done by plotting the ACF correlogram. ACF plot suggests order of moving average MA process. Significant values above threshold in 1, 5, 6, 7 and 11 lags are seen;
c) ACF Spikes around seasonal lags suggest seasonal moving average process. Values above threshold for the autocorrelation seasonal in lag 6 and 12 are seen so 1 and 2 will be selected;

d) Partial Autocorrelation Function (PACF): The PACF plot from Equation (38) can provide answers to what order, the observed time series, can be modeled with an AR model. Spikes for the autocorrelation above threshold function in 1, 2, 3, 4, 5, 6 and 11 lags are observed;

e) PACF Spikes around seasonal lags suggest seasonal in autoregressive order. Values above threshold for the autocorrelation seasonal in lag 6 and 12 are observed so 1 and 2 is selected.

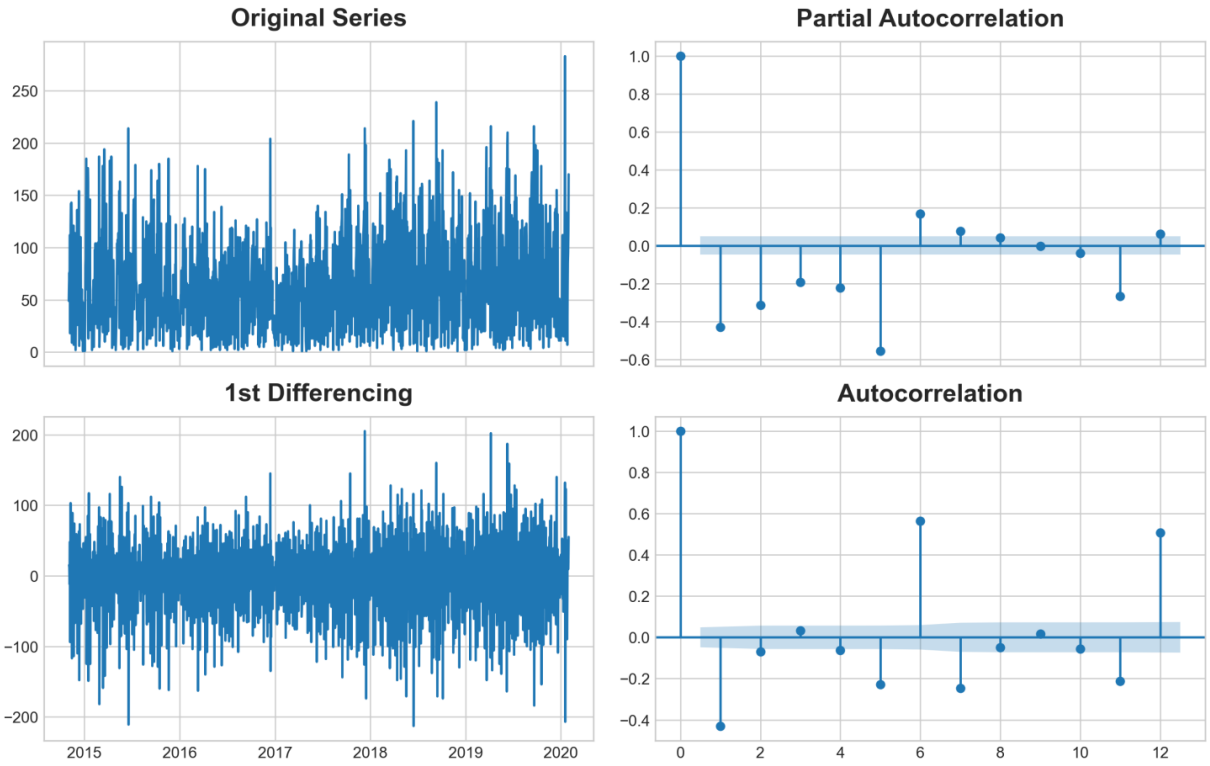


Figure 24 - Differencing on left and ACF/PACF on the right.

Source: Elaborated by the author

Based on the ACF and PACF plots, the following SARIMA orders were selected, and they can be seen in Tables 19 and 20.

Table 19 - Hyperparameters Grid Search for ARIMA model.

Hyperparameters - ν_N	**Value - i_n**
Autoregressive Order (p)	1, 2, 3, 4, 5, 11
Differencing Order (d)	0, 1
Moving Average order (q)	1, 5, 11
Seasonal Autoregressive Order (P)	0
Seasonal Differencing Order (D)	0
Seasonal Moving Average order (Q)	0
Seasonal Frequency (s)	6

Source: Elaborated by the author

Table 20 - Hyperparameters Grid Search for SARIMA model.

Hyperparameters - ν_N	**Value - i_n**
Autoregressive Order (p)	1, 2, 3, 4, 5
Differencing Order (d)	0, 1
Moving Average order (q)	1, 5
Seasonal Autoregressive Order (P)	1 ,2
Seasonal Differencing Order (D)	0, 1
Seasonal Moving Average order (Q)	1, 2
Seasonal Frequency (s)	6

Source: Elaborated by the author

Tables 19 and 20 show the breakdown configurations for ARIMA and SARIMA to be applied in equation (65). The reason for a separate hyperparameter count is for orders (p) and (q) greater than or equal to s, the seasonal components, would cause a duplication on lags (sP) and (sQ). To avoid this, equation (95) is then computed by first an ARIMA orders search space plus SARIMA orders search space.

From equation (94) N = 8 hyperparameters are selected. From Table 14 equation (95) results in $\Omega_{Arima} = 36$ and from Table 15 we have $\Omega_{SARIMA} = 160$, with a total search space of $\Omega = 392$ possible configurations.

3.9. LSTM method hyperparameters

To define the grid search space for LSTM, two main characteristics are required: the sequence of observations that define the network inputs dimension and the network topology. These two characteristics are unfolded in the following hyperparameters below:

Table 21 - Possible Hyperparameters for LSTM

Hyperparameter	Values
Time Steps	Input sequence Dimension.
LSTM Layers	How deep the model will be. Could help generalize the model.
Units (Neurons) in each LSTM cell (Hidden States)	Learning capacity of the network. Accuracy vs overfitting.
Batch size	Number of training samples to work through before the model's internal parameters are updated.
Learning rate	Controls how quickly the model is adapted to the problem depending on the number of epochs.
Noise Layer	Add noise to an existing model to prevent overfitting.
Epoch Size	Number of complete passes through the training dataset.

Source: Elaborated by the author

Next, a qualitative approach to understand each of these hyperparameters are provided and this follows according to the definitions in section 2.10:

a) Input Sequence Dimension: A supervised learning problem is comprised of input patterns (X) and output patterns (y), such that an algorithm can learn how to predict the output patterns from the input patterns. Given a sequence of numbers for a time series dataset and additional features, data can be restructured to a supervised learning problem. This is done by using the target variable previous time steps as input variables and use the next time step as the output variable, according to Table 10. The input sequence is formed then by a defined number of previous time steps that is applied to the target variable,

the regressors and the size of the training partition. Define the best timestep for the network input sequence is a hyperparameter which needs to be defined.

b) Hidden Layers: Neural networks with just one hidden layer can theoretically model even the most complex functions, provided it has enough neurons (Hornik, 1990). A choice between one or two layers is therefore selected. For the number of hidden layers in the LSTM, many recent studies suggested that a low value usually performs better (Bandara; Bergmeir; Smylet, 2020; Salinas *et al*., 2019; Smyl; Kuber, 2016; Wang *et al*., 2019);

c) Hidden States (Neurons) per Hidden Layer: If the number of Hidden States is very small, the LSTM will not memorize all necessary information to perform the prediction. If the number of Hidden States is very high LSTM could overfit on the training instances. Throughout the literature in forecasting applications there is not a universal approach about what range of hidden states should be used as this is not clear mentioned (Abbasimehr; Shabani; Yousefiet, 2020; Ozdemir; Buluş; Zoret, 2022; ArunKumar *et al*., 2020; Hewamalage; Bergmeir; Bandaraet, 2021; Nguyen *et al*., 2021). Running initial experiments, keeping all other hyperparameters fixed, an initial guidance is provided. The options of 50, 100 and 150 are selected which is between the size of the input layer and the size of the output layer. For more than one layer, to reduce the computational cost, the researchers often adopt a halving strategy, the number of hidden states consecutively reduce by half from the lower to the higher layer. For instance, if the number of hidden layers is 3 and the number of hidden states for the first LSTM layer is 100, then the number of hidden states for the second LSTM layer is 50 and for the third layer is 25 (Shen *et al*., 2020);

d) Batch size: large batch sizes often lead to training instabilities and the resulting model may not generalize as well as a model trained with a small batch size. The batch size is expected to be within range of the periodicity found for the time series and this is confirmed by running a simulation ranging from 2 to 60 with all the other hyperparameters fixed;

e) Learning rate: The algorithm to train the neural network will be the stochastic gradient descent algorithm. Stochastic gradient descent is an optimization algorithm that measures the error in the training dataset by updating the weights of the model as shown in Equations (88), (89) and (90). The amount that the

weights are updated during the training is defined as the learning rate and will then control how quickly the model is adapted to the problem. The gradient descent uses as default the learning rate of 0.01 and it will be allowed to vary to 0.1 also.

f) Noise Layer: Adding an extra layer that adds a Gaussian noise to the output of the hidden state works as regularizing effect and reduce the overfitting to a under constrained neural network model. This hyperparameter is the standard deviation of the noise that will be between 0.01 and 0.001;

g) Epoch Size: One training epoch is referred to a single iteration over all training instances (Reimers; Gurevych, 2017). Smaller learning rates require more training epochs given the smaller changes made to the weights at each update, whereas larger learning rates result in rapid changes and require fewer training epochs. If the number of training epochs is too small, the model will not capture the patterns of training instances. Also, if the epoch number is too large, the model will suffer from overfitting. Therefore, finding a suitable epoch number is important in achieving a model with high performance. The simulation did in this work for batch size helped us define the number of epochs suitable for the model.

The Hyperparameter selection for Neural networks is an active area of research and the success still suffer from empirical choices of those hyperparameters. Optimization and evolutionary algorithms are some of the state-of-the-art options to maximize performance (Jaderberg *et al*., 2017).

Like SARIMA, the grid space possibilities can grow exponentially and there are some hyperparameters such as the number of layers, the learning rate and the epoch size that significantly increase the model fitting time.

Table 22 - Hyperparameters Grid Search for LSTM model.

Hyperparameter	Values
Input Sequence Time Steps	(a)
Hidden Layers	1, 2, 3
Hidden States per Hidden Layer	50, 100, 150
Batch size	(d)
Learning rate	0.1, 0.01, 0.001
Epoch Size	(g)
Gaussian Noise	0.01, 0.001

Source: Elaborated by the author

Table 22 shows the values of the hyperparameters that were defined in b, c, e and f items.

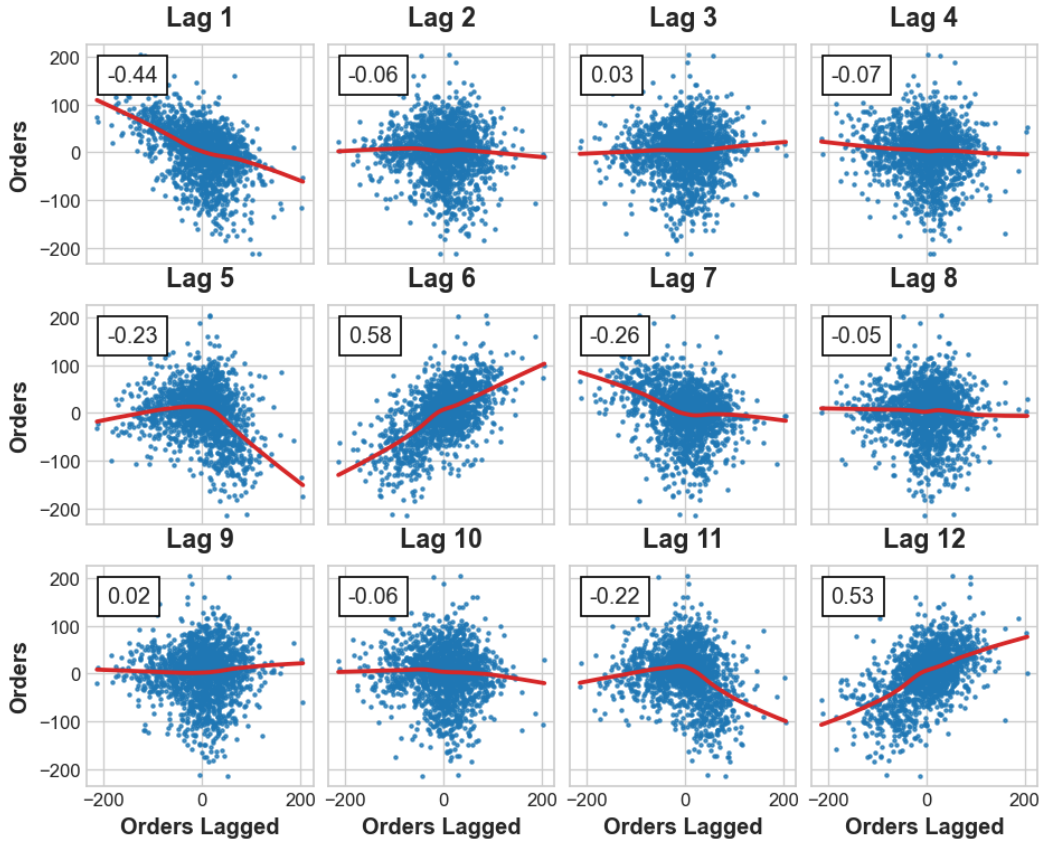


Figure 25 - Plots of lags vs original series

Source: Elaborated by the author

To define the values for the inputs sequence time steps, from item (a), a scatter plot of the target variable y_t and with previous lags y_{t-1} is seen in Figure 25 to

observe which lag present the higher correlation. Lags 1, 6 and 12 show higher dependencies and these will be the candidates for this hyperparameter. This result is also expected as seen in section 3.8 for the ACF plot in Figure 24.

For the batch size and epoch size, items (d) and (g), it was chosen to run experiments by fixing all other hyperparameters.

At this point, it is leveraged the previous hyperparameters to experiment which range will define the batch size. This is done looking the validation loss function of the stochastic gradient descent which is the MSE of the normalized values in all the 12 validation partitions defined in Table 10.

Table 23 - Batch size and Epoch Size hyperparameter simulation LSTM

Hyperparameter	Values
Input Sequence Time Steps	1, 6, 12
Hidden Layers	1
Hidden States per Hidden Layer	100
Batch size	2, 4, 6, 8, 10, 12, 24, 30, 40, 50, 60
Learning rate	0.01
Epoch Size	200
Gaussian Noise	0.01

Source: Elaborated by the author

Table 23 shows the hyperparameters selection to define optimal boundaries for batch size and Epoch size. Since the input shape (time steps hyperparameter) can interact directly with the batch size, all three lags found in Figure 30 will be used in this experiment.

Table 24 - Results of the lowest MSE found in each split.

Split	Batch	Time Steps	Lowest MSE
1	2	6	612
2	2	12	646
3	2	12	529
4	4	12	746
5	6	12	1698

Cont'd Table 24 - Results of the lowest MSE found in each split.

Split	Batch	Time Steps	Lowest MSE
6	4	12	1683
7	4	12	1677
8	2	6	624
9	2	12	362
10	2	12	878
11	2	12	920
12	2	12	537

In Table 24 the lowest value found for MSE, in the validation partition of each split, and the respectively batch size and time steps value are shown. Batch sizes of 2,4,6,12 provide the lowest MSE when analyzing all the 12 splits. As the batch size is increased there was no gain observed in the MSE and therefore these batches in Table 19 are going to be the optimal boundaries. This result also shows that the batch size optimal boundaries is close to the range of the series periodicity.

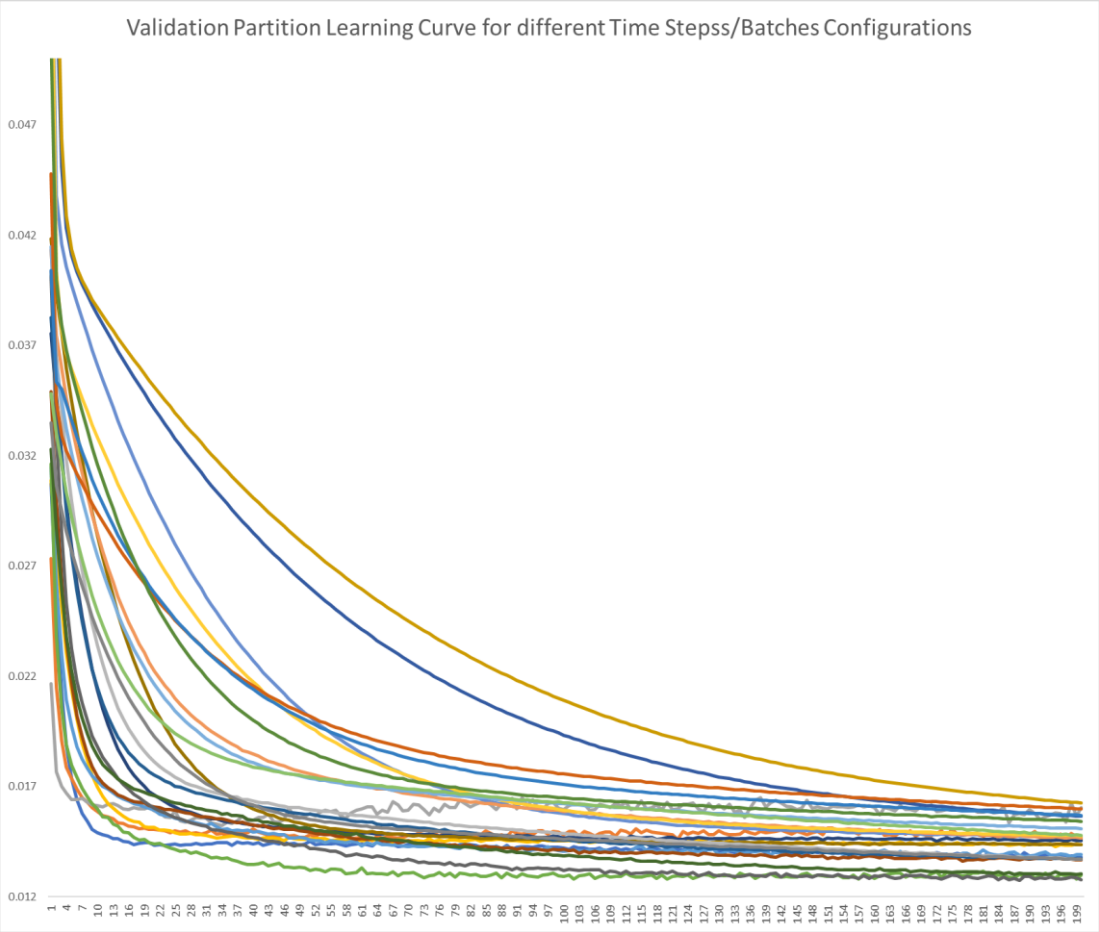


Figure 26 - Learning curves showing the Training Epochs

Source: Elaborated by the author

Figure 26 shows the learning curves of the training partition during the model fitting. The value of the gradient descent is reduced to a point where there is no gain anymore. Some of the configurations this was achieved fast (with 20-50 epochs) but it can be observed instances with almost the 200 epochs were observing small gains. To not further extend the grid search space possibilities, it was decided to fix the number of 200 epochs and apply an early stopping procedure. It stops the training process when 5 epochs have not showed any improvement. The gaussian noise added to the model will try to avoid this early stopping to be triggered.

Table 25 - Hyperparameters Grid Search for LSTM model resulting in 648 possible combinations for each dataset split.

Hyperparameter	Values
Input Sequence Time Steps	1, 6, 12
Hidden Layers	1, 2, 3
Hidden States per Hidden Layer	50, 100, 150
Batch size	2, 4, 6, 12
Learning rate	0.1, 0.01
Epoch Size	200
Gaussian Noise	0.01, 0.001

Source: Elaborated by the author

Finally, Table 25 shows all the hyperparameters value that will be used in the grid search. According to Equation (94) N = 7, and from Equation (95) resulting in $\Omega = 432$ possible hyperparameters.

4. Results

This chapter is structured as follows: section 4.1 the training and validation partition results are analyzed with respect to the best hyperparameters found for all models. In section 4.2, a visual analysis of the residuals of the fitted model for ES and SARIMA are done to provide a diagnostic of the training results beyond the error metrics numbers. It is also shown the predicted against observed values from the validation (out of sample) partition.

Section 4.3 provides the same diagnostic give in section 4.2, but now focus for non-parametric models. The training and out-of-sample MSE (stochastic gradient descent) error curves for LSTM are shown to provide a diagnostic over overfitting/underfitting, the predicted against observed values from the validation (out of sample) partition and the confusion matrix are also discussed.

Next, section 4.4 provides the test error metrics result for the best hyperparameter in each model, the best model result and the improvements gain from the proposed process.

A discussion of how the stochastic nature of the LSTM algorithm could impact the results comparing to the ES and SARIMA is done in section 4.5.

4.1. Training and validation partition results

In each of the 12 splits, the grid search provides the results for all hyperparameters vectors defined in sections 3.7, 3.8 and 3.9. The results include the training metrics AIC, BIC, MAE and MSE for ES and SARIMA and the validation metrics MAE, MSE, GMRAE, MDA and MCP for all three models. The training metrics for LSTM is not include due to the normalization done in the dataset and the resulting metrics are in different scale.

Table 26 – **Training** (fitting) results for exponential smoothing (**ES**)

Split	Trend	Damping	Seasonal	AIC	BIC	MAE_{fit}	MSE_{fit}
1	Multi	None	Multi	11392	11413	23.25	961
2	Add	None	Multi	11608	11628	22.82	924
3	Add	None	Add	11655	11676	22.04	795
4	Multi	None	Add	11920	11941	22.25	806
5	Multi	None	Multi	12418	12439	23.45	980
6	Multi	None	Multi	12690	12711	23.57	987
7	Multi	None	Multi	12933	12954	23.60	987
8	Multi	None	Multi	13191	13212	23.57	983
9	Add	None	Add	13151	13172	21.97	790
10	Multi	None	Multi	13697	13718	23.60	983
11	Multi	None	Multi	13708	13729	23.62	985
12	Add	None	Add	13657	13678	22.05	797
					Avg	22.98	915

Source: Elaborated by the author

Starting with ES, in Table 26, “Add” means “Additive” and ‘Multi” means “Multiplicative” for trend and seasonality respectively and it shows the training results in each split. As mentioned in section 3.6 the best hyperparameters were selected based on the lowest MCP found in the validation partition (Table 27) and the model only with coefficients that are statistically significant is chosen.

Both trend and seasonal contains multiplicative and additive types for the sampling procedure of 12 splits. Changes in the trend and seasonal behavior were

observed, which makes the revalidation of the hypermeters in each forecasting cycle important. Also, no damping options were selected, mainly due to the series being stationary according to the ADF test. The increasing AIC and BIC values observed for the splits is due to the increasing number of data points used for fitting, as point out in Table 10.

Table 27 – **Validation** (out of sample) results for exponential smoothing (**ES**)

Split	Trend	Damping	Seasonal	MAE	MSE	GMRAE	MDA	MCP
1	Multi	None	Multi	25.79	1250	0.46	0.31	**0.48**
2	Add	None	Multi	23.99	1041	0.48	0.29	**0.40**
3	Add	None	Add	23.37	930	0.50	0.28	**0.37**
4	Multi	None	Add	23.16	924	0.51	0.28	**0.38**
5	Multi	None	Multi	21.66	854	0.45	0.28	**0.32**
6	Multi	None	Multi	26.66	1149	0.62	0.29	**0.53**
7	Multi	None	Multi	21.22	841	0.42	0.24	**0.28**
8	Multi	None	Multi	24.33	1079	0.49	0.26	**0.41**
9	Add	None	Add	21.76	806	0.48	0.23	**0.30**
10	Multi	None	Multi	26.36	1268	0.54	0.25	**0.49**
11	Multi	None	Multi	32.42	1784	0.64	0.25	**0.76**
12	Add	None	Add	25.51	1145	0.48	0.25	**0.41**
			Avg	24.69	1089	0.51	0.27	**0.43**

Source: Elaborated by the author

Table 27 shows the out of sample metrics for the same best hyperparameters from Table 26. Comparing the out of sample metrics for MAE and MSE with the training partition, it is seen a close average result (MAE training 22.98 vs MAE out of sample 24.69). The close values here are due to the hyperparameter selection criteria which considers the best MCP and not the best AIC. Higher errors for out of sample are expected due to the residual's variance and the standard errors of the coefficients obtained from the maximum likelihood estimators, that will carry over to the forecasted values.

Table 28 – Comparison between the selection of the best **AIC** or **MCP** for **ES**

Split	Lower AIC	Validation MAE with Lower AIC	Validation MAE with Lower MCP
1	11137	35.77	25.79
2	11396	28.58	23.99
3	11649	23.37	23.37
4	11894	25.64	23.16
5	12143	23.18	21.66
6	12400	39.77	26.66
7	12639	25.83	21.22
8	12889	45.18	24.33
9	13144	21.76	21.76
10	13396	40.86	26.36
11	13406	56.40	32.42
12	13650	25.51	25.51
	Avg	32.65	24.69

Source: Elaborated by the author

Although MSE was chosen for the MCP calculation in the decision criteria, MAE provides a practical and intuitive interpretation of the errors. Using that, a comparison between how MAE would have performed in the validation partition when the model with the lowest training AIC is chosen can be seen in Table 28. On average, ES had 24.69 units of error in prediction with the lowest MCP while the selection of the lowest AIC would have yielded 32.65. This is only possible when the model calibration is done through 3 partitions of the dataset.

Table 29 – **Training** (fitting) results for **SARIMA**

Split	p	d	q	P	D	Q	AIC	BIC	MAE_{fit}	MSE_{fit}
1	1	0	1	1	0	1	10673	10713	16.87	493
2	1	0	1	0	0	0	10924	10955	17.37	515
3	5	1	1	0	0	0	11305	11356	18.49	572
4	1	0	1	1	0	1	11418	11459	17.31	513
5	1	0	1	0	0	0	11660	11691	17.71	533
6	1	0	1	0	0	0	11903	11934	17.71	532
7	1	0	1	1	0	1	12138	12179	17.47	518
8	1	0	1	1	0	1	12377	12419	17.43	516
9	4	1	1	1	1	1	13137	13195	19.74	652
10	1	0	1	0	0	0	12874	12906	17.88	541
11	1	0	1	0	0	0	12883	12915	17.88	541
12	4	1	1	1	1	1	13658	13716	19.94	663
								Avg	17.98	549

Source: Elaborated by the author

Like ES, Table 29 shows the best hyperparameters vectors for SARIMA found according to the lowest MCP in the validation partition and their respective training metrics. The letters “p” and “P” are the order of the autoregressive and seasonal autoregressive respectively. The letters “q” and “Q” are the order of the moving average and seasonal moving average respectively. “d” and “D” are the differencing used for non-seasonal and seasonal components.

There is not a predominance of higher orders such as “4”, or “5” for “p” or “q” am “2” for “P” or “Q”, which means that those cases, even if the MCP were lower, the criteria to select models only with coefficients that were statistically significant was triggered. More complex SARIMA models contained several coefficients which their p values may be higher than 5%. This can also cause multicollinearity where two or more coefficients are highly correlated, providing redundant information. Removing

them can help reduce multicollinearity and improve the stability of the model's predicitons.

Table 30 – **Validation** (out of sample) results for **SARIMA** for best MCP

Split	p	d	q	P	D	Q	MAE	MSE	GMRAE	MDA	MCP
1	1	0	1	1	0	1	21.30	742	0.43	0.23	**0.26**
2	1	0	1	0	0	0	20.71	704	0.45	0.26	**0.26**
3	5	1	1	0	0	0	18.31	547	0.40	0.25	**0.19**
4	1	0	1	1	0	1	18.17	541	0.41	0.27	**0.21**
5	1	0	1	0	0	0	18.95	605	0.44	0.27	**0.24**
6	1	0	1	0	0	0	19.65	632	0.45	0.30	**0.26**
7	1	0	1	1	0	1	21.65	763	0.49	0.26	**0.30**
8	1	0	1	1	0	1	22.74	813	0.53	0.26	**0.34**
9	4	1	1	1	1	1	22.25	729	0.58	0.24	**0.32**
10	1	0	1	0	0	0	21.83	735	0.52	0.28	**0.32**
11	1	0	1	0	0	0	21.92	739	0.50	0.27	**0.31**
12	4	1	1	1	1	1	20.32	649	0.42	0.26	**0.24**
					Avg		20.65	683	0.47	0.26	0.27

Source: Elaborated by the author

For both tables 29 and 30, the training and out of sample results were better than ES model. SARIMA has more flexibility in its formulation that allows for a increases fitting performance. In this model, the validation MAE is higher than the training MAE as it was observed for ES.

Interpreting the other metrics that compose MCP shows that SARIMA, on average, had GMRAE of 0.47 and ES GMRAE of 0.51, which is significantly lower than 1, which means that both models are better than the naïve model. The MDA had 0.26 for SARIMA and 0.27 for ES, showing that only 26% and 27% of the time the direction of the predictions was wrong. MSE on average was the metric that made the difference having a better result for SARIMA (683 vs 1089 for ES).

Table 31 – Comparison between the selection of the best **AIC** or **MCP** for **SARIMA**

Split	Lower AIC	Validation MAE with Lower AIC	Validation MAE with Lower MCP
1	10667	21.40	21.30
2	10924	20.71	20.71
3	11170	18.27	18.31
4	11397	18.33	18.17
5	11660	18.95	18.95
6	11903	19.65	19.65
7	12132	21.74	21.65
8	12371	22.76	22.74
9	12605	23.29	22.25
10	12874	21.83	21.83
11	12883	21.92	21.92
12	13117	22.36	20.32
	Avg	20.93	20.65

Source: Elaborated by the author

Here the choice of the lower MCP for the hyperparameters vector selection did not change the results for the validation MAE. The use of additional regressors defined in section 3.4 provides good generalization for less flexible models such as the ones with lower AIC (lower “p” and “q” orders and/or no seasonal “P” and “D” orders), improving the results in the validation partition. For splits 5, 6, 10 and 11, the model with lowest MCP in the validation partition is also the model with lowest AIC in the training partition.

In SARIMA, the observed results in the out of sample partition also shows that the model generalizes well to unseen data as no significant increase in error was observed. As the model increase in complexity, overfitting can occur as pointed out in section 2.2. This will be important to verify in the LSTM results which is a model made of thousands of parameters.

The training results for LSTM are formed by scaled data from the gradient descent algorithm which is not direct comparable with the training results of the parametric models.

The model hyperparameters are presented below in the following order: Time steps, layers, hidden states, batch size, learning rate, epochs and gaussian noise.

Table 32 - Validation (out of sample) results for **LSTM** for best MCP

Split	Model	MAE	MSE	GMRAE	MDA	MCP
1	1, 1, 50, 2, 0.1, 200, 0.001	20.21	643	0.42	0.22	0.22
2	1, 1, 50, 12, 0.01, 200, 0.001	21.97	728	0.48	0.25	0.28
3	1, 2, 100, 2, 0.1, 200, 0.001	18.18	529	0.43	0.26	0.20
4	1, 2, 100, 2, 0.1, 200, 0.01	18.29	531	0.44	0.26	0.21
5	1, 3, 50, 6, 0.1, 200, 0.01	18.82	576	0.46	0.29	0.25
6	1, 2, 50, 2, 0.1, 200, 0.001	18.47	572	0.43	0.29	0.23
7	12, 2, 50, 6, 0.1, 200, 0.001	20.17	680	0.46	0.26	0.26
8	12, 3, 50, 4, 0.01, 200, 0.01	21.68	772	0.48	0.28	0.31
9	12, 1, 50, 12, 0.1, 200, 0.001	20.39	682	0.50	0.29	0.29
10	12, 1, 50, 4, 0.01, 200, 0.001	20.18	630	0.49	0.24	0.25
11	12, 1, 50, 2, 0.01, 200, 0.001	20.24	653	0.45	0.25	0.25
12	12, 1, 50, 12, 0.1, 200, 0.001	19.77	606	0.42	0.26	0.23
	Avg	19.87	633	0.45	0.26	0.25

Source: Elaborated by the author

The interpretation of the $\boldsymbol{\nu}_N^*$ vectors for LSTM does not provide an intuitive conclusion due to the nature of the LSTM algorithm. All these configuration changes from um split to the other is the algorithm finding the best network topology. The resulting model is formed by thousands of parameters (weights and bias) and the learning curves during training and validation provides answer if overfitting was avoided as it will be shown in next section.

SARIMA validation metrics were marginally lower on average than LSTM, which is an interesting result. Comparing Table 30 with Table 32, on average SARIMA had MCP of 0.27 and LSTM with MCP of 0.25, a result that shows how SARIMA proves to be a flexible model to compare with Neural Networks as point out by Makridakis,

Spiliotis, Assimakopouloset (2018) and as discussed in sections 1 and 2.6. The test partition will provide a measure of how good SARIMA and LSTM has generalized with the data based on the hyperparameters here selected.

The difference set of hyperparameters observed in each split for all models shows the importance to consider new available data to recalibrate the model for a new prediction. The process from Figure 11 that takes the recent data and recalibrates the model with a new run of hyperparameters, every time a new forecasting range is provided, makes sure that the recent changes in data are capture. It also guarantees that the full features of each model are being explored for the best performance.

Form this section it is possible to conclude that the selection of the best hyperparameter using a validation partition provides good results. Additionally, MCP is a criterion that challenges the model to generalize the dataset in different behaviors, with accuracy on MSE, benchmark with GMRAE and trend with MDA. The last partition will provide which model is the winner in those metrics and how much improvement was gained comparing to with the existing guideline.

4.2. Parametric model diagnostics – ES and SARIMA

In this section, a series of plots are presented with the objective to perform a diagnostic of the training results beyond the error metrics shown in previous section. The purpose of this is to reveal graphically whether the model adequately captures the true patterns and relationships in the data. This analysis provides areas for model improvement, such as incorporating additional predictors, adjusting model structure, or transforming the target variable. This will focus on split 1, where it has 1173 datapoints for training and 130 for validation (out of sample). Appendix B includes all other splits for SARIMA and appendix C includes the splits for LSTM, the best two models per previous section.

The visualization is composed of:

a) Standardized Residuals vs Fitted values: It can reveal issues like non-linearity, heteroscedasticity (unequal variance of errors), and outliers.
b) ACF plot of the residuals: detect whether the errors in a model are independent or exhibit patterns of correlation over time.

c) Q-Q plots: Assess if the standardized residuals follow a normal distribution. Non-normal residuals could indicate issues such as (one but not necessarily all of them): Non-linear relationships, outliers and heteroscedasticity.
d) Validation (Out of sample) vs Predicted plot: visualize the forecasted values against the true values.

Table 33 – Coefficients (parameters) for ES split 1 best hyperparameters.

	coef	std err	z	P>|z|	[0.025	0.975]
smoothing_level	0.1307	0.016	8.059	0%	0.099	0.163
smoothing_trend	0.0012	0	4.763	0%	0.001	0.002
smoothing_seasonal	0.0636	0.011	6.05	0%	0.043	0.084

Source: Elaborated by the author

Table 33 shows the resulting parameters (coefficients) found for split 1 best hyperparameters. The ‘std err’ column is the standard error of the coefficients and estimate the error of the predicted value. It shows how strong is the effect of the residual error on the estimated parameters and therefore how accurately the model estimates the coefficient's unknown value.

The ‘z’ is equal to the values of coefficients ‘coef’ divided by ‘std err’, also known as the standardized coefficient. The P>|z| column is the p-value of the coefficient given by t statistics as discussed in equation (25), were it shows that all coefficients are statistically significant (less than 5%). The last two columns are the confidence intervals of the coefficients which is a range of values that the coefficient will be in with the respective level of confidence.

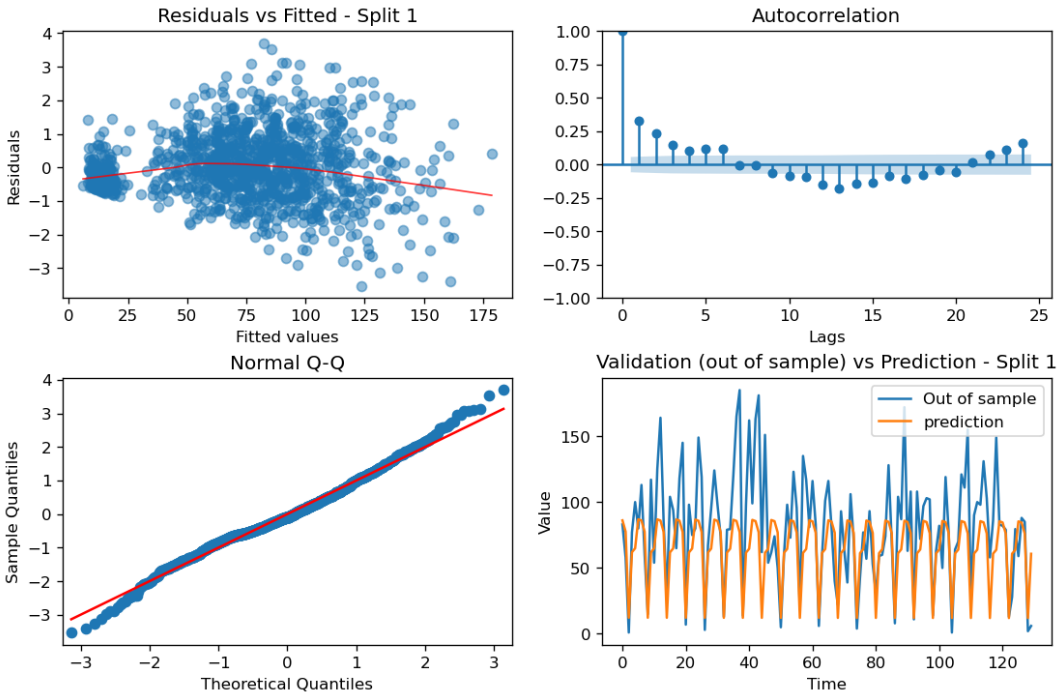


Figure 27 - Split **1** diagnostics plot for **ES** model.

Source: Elaborated by the author

In Figure 27, top left chat is the residuals plot against the fitted values. The red line is the Loess adjusted regression. Residuals that are normally distributed and follow a white noise behavior with mean zero have a Loess regression line with no trend and close to zero. The top right is the correlogram ACF of the residuals. If the residuals are strict random white noise, this plot shows no significant correlation in any lag, meaning that the ACF plot will show no spikes at any lag. The standardized residuals from equation 102 are useful when comparing different models by having the same scale.

The bottom left chart is the or quantile-quantile plot (Q-Q), a graphical tool used in statistics to compare two probability distribution. In the Q-Q plot, each data point from the dataset is put in its own quantile, then a theoretical normal distributed data point is generated from the corresponding theoretical quantile. These two points are plotted against each other. If the points in the plot follow a straight line, it means that the residuals are normally distributed. The bottom right chart are the predicted values against the real observed values in the test partition.

Observing figure 27, the split 1 for ES model, bottom right chart shows that the validation (Out of sample) vs prediction plot does not capture the different peaks observed through time for the target variable. The ES formulation does not provide enough flexibility, but it does capture the minimum observed values given by the periodicity of 6 from the series. Also, the residuals vs fitted values does not show a constant variance of the errors. For a constant variance across the series, known as homoscedasticity, it is expected that:

a) Random variation above and below 0;
b) No apparent “patterns;
c) The width of the points is relatively constant.

This might be caused by a underfitting of the model during the training partition, the distribution of the data which does not follow a normal behavior and/or the presence of outliers in the series, which is further observed in the normal Q-Q plot. In the Q-Q plot the points deviate from the red line in the tails (ends) of the plot, suggesting that the data has heavier tails than a normal distribution with more extreme values (outliers). Since SARIMA has shown better results according to table 29, it’s diagnostics plots could reveal if the model were able overcome these difficulties with the dataset.

Also, a persistent autocorrelation exits, not strong as the ACF of the original series, but the model failure to fully capture the exiting correlations within the time series. This result is expected, as ES is a model which does not address autocorrelation of the series, but rather models the trend and seasonality using exponentially weighted parameters which are given by the hyperparameters seen in section 3.7.

Table 34 – Coefficients (parameters) for SARIMA split 1 best hyperparameters.

	coef	std err	z	P>|z|	[0.025	0.975]
x1	0.7848	0.044	17.865	0.0%	0.699	0.871
x2	0.0001	6.70E-06	17.867	0.0%	0	0
ar.L1	0.5016	0.127	3.953	0.0%	0.253	0.75
ma.L1	-0.3362	0.134	-2.502	1.2%	-0.6	-0.073
ar.S.L6	0.5639	0.118	4.764	0.0%	0.332	0.796
ma.S.L6	-0.4145	0.13	-3.18	0.1%	-0.67	-0.159

Source: Elaborated by the author

Moving to SARIMA, Table 34 shows the coefficients resulted from the fit of the best hyperparameters selected (lowest MCP) and in which all coefficients are statistically significant 9p value less than 5%). In the table, x1 is the additional regressor resulting from the trend and seasonality modeling from section 3.4 and x2 is an regressor containing values of material availability projection. The remaining coefficients are the autoregressive components “ar.L1“ for order p =1, the “ar.S.L1“ is the seasonal P=1, moving average component “ma.L1“ is the q=1 and “ma.S.L6“ is the seasonal component Q=1.

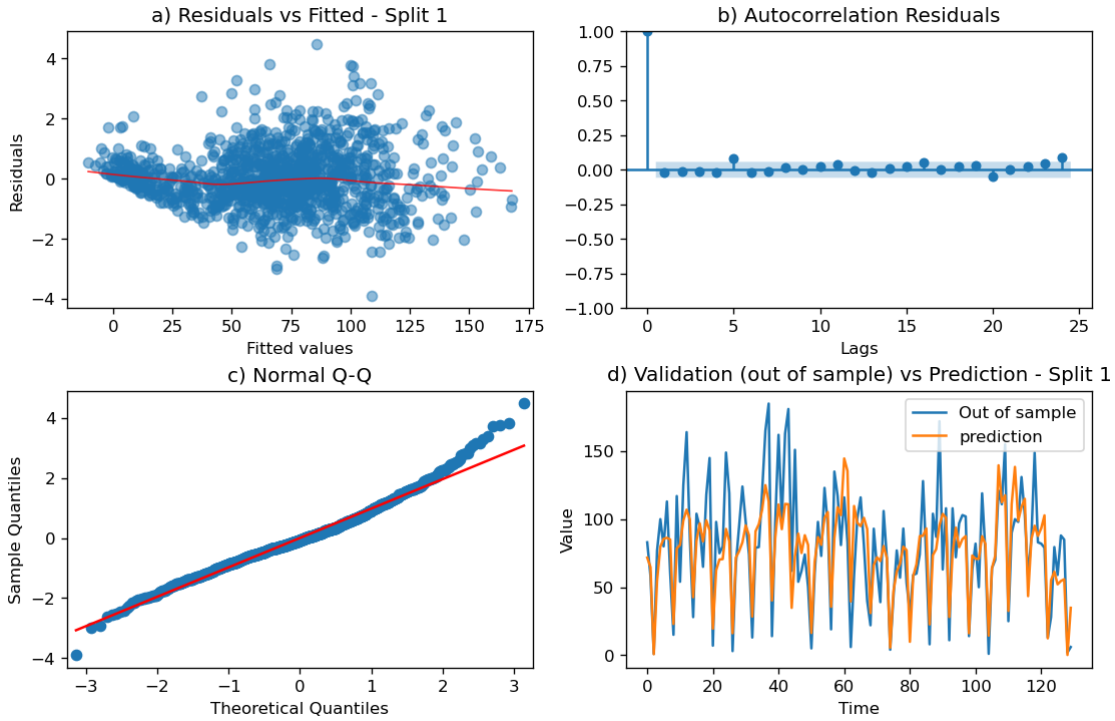


Figure 28 - Split **1** diagnostics plot for **SARIMA**

Source: Elaborated by the author

In Figure 28, the validation (Out of sample) vs prediction plot for SARIMA shows how this model were able to better capture the different peaks observed through time due to the modeling of the correlation of the series. The ACF residuals plot shows no significant spikes at any lag. The variance lags are all similar in magnitude to each other and the lack of behaviors such as peaks, exponential decays or sinusoids suggest a good model fit.

The residuals scatter plot also shows improvement towards a non-constant variance, although still some predicted values does provide higher errors where some points shows deviations away from a mean of zero (white noise). The resulting Loess regression line is closer to a white noise behavior.

The normal Q-Q plot shows that the residuals does deviate from normality in the tails of the plot suggesting again the presence of outliers. This concludes that performing an outliers treatment in the dataset or data transformation could lead an improvement in this diagnostic.

For this analysis SARIMA provides better goodness of fit than ES. The remaining splits for SARIMA are all similar in analysis here provided and they are available in appendix B.

4.3. Non-parametric model diagnostics – LSTM

LSTM is a model that after trained, results in thousands of parameters, given by the weights and bias show in its formulation from section 2.10. LSTM models are trained by calling the “fit” function from the python package “Keras” used in this work. This function returns a variable called history that contains a trace of the loss value of the gradient descend of the training partition and the out of sample partition. These scores which here given by the MSE of the gradient descent are recorded at the end of each epoch.

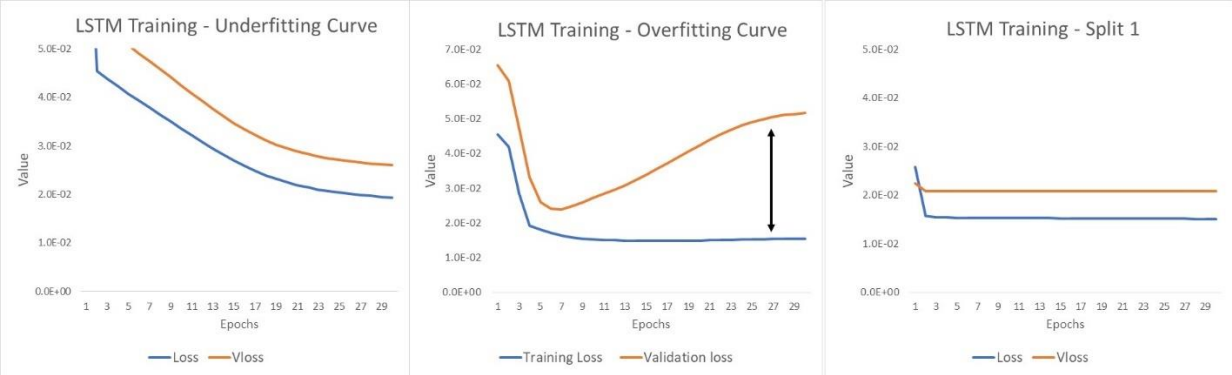


Figure 29 – Split **1** diagnostics plot for **LSTM**

Source: Elaborated by the author

Figure 29 shows three plots where is possible to see the training loss and the validation loss in each epoch (the low value is due to the normalization of the data). The plot on the left shows a case of underfitting that can occurs if the hyperparameters provided does not allows the model to have the flexibility needed such as insufficient hidden states or number of epochs.

The plot in the middle is the overfitting case where the training loss have converged to very low value, but the loss on the validation set improves to a point and then begins to degrade. This can be diagnosed from the plot where the train and validation loss slopes down but, the validation loss hits an inflection point, and starts to slope up again. The plot on the right is the resulting curves observed from the best hyperparameters chosen for split 1 according to the lowest MCP.

Figure 30 – LSTM training curves and out of sample prediction

Source: Elaborated by the author

The complete training and validation losses observed for split 1 is seen in Figure 30 at the top. The model converged in training after 125 epochs of training, meaning that changes in the weights and bias did not provide further improvements in the gradient descent. The corresponding loss for the out of sample is also shown to converge and avoid overfitting. Although the out-sample loss curve is used visually for identify underfitting or overfitting, the criteria to select the best hyperparameter is the MCP of the out-sample partition as show in Table 32. The bottom part of Figure 30 is the visual of the out of sample against prediction. The data input for LSTM also includes both additional regressors that were used for SARIMA.

The other splits are included in appendix c, in which includes curves with optimal behaviors such as split 3, 4, 5 and 6.

Before moving to the test partition results, a comparison with SARIMA predictions for split 1 is shown next in a visual plot and using the confusion matrix concept.

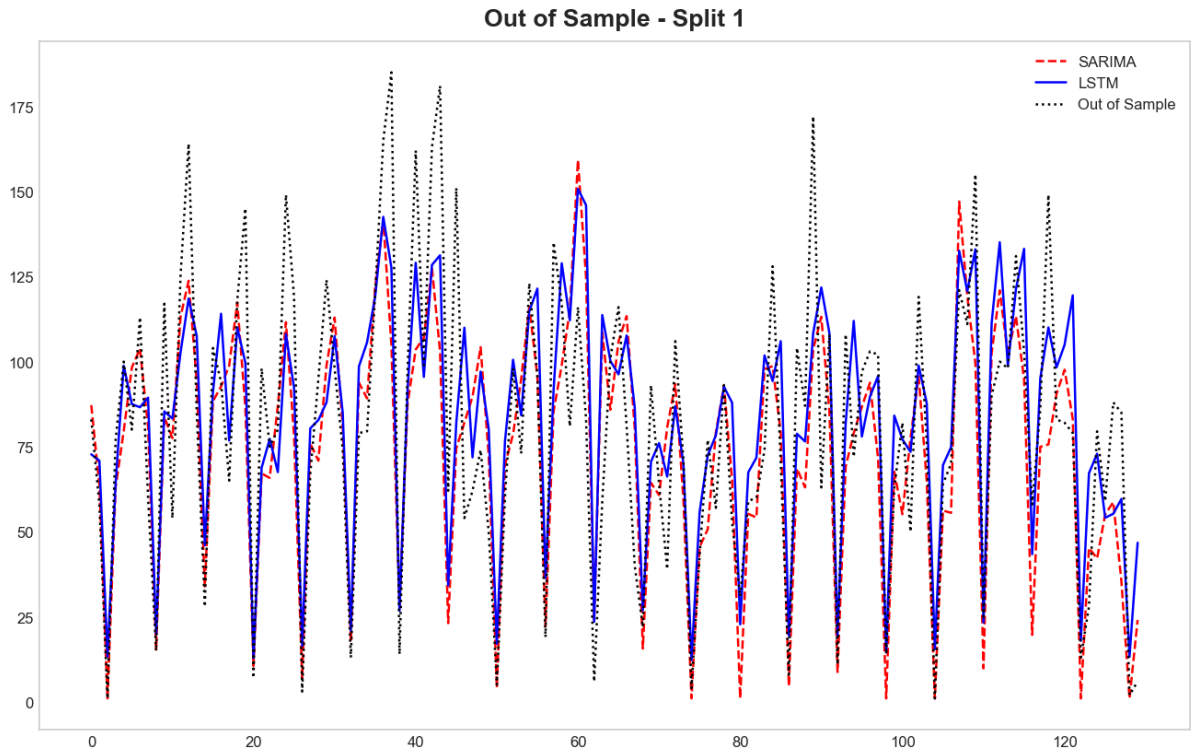


Figure 31 – Out of Sample LSTM and SARIMA from split 1

For both SARIMA and LSTM models in Figure 31, it is possible to vizualize how SARIMA and LSTM provided flexibility in capture different peaks than ES and both models had a very similar result for the predicted values which was observed in Tables 30 and 32.

The extension of the confusion matrix used for machine learning classification problems for regression provide another basis of comparison for the model selection, but it was not used to select the best hyperparameters. The objective was to evaluate overall accuracy and look for biases in the results.

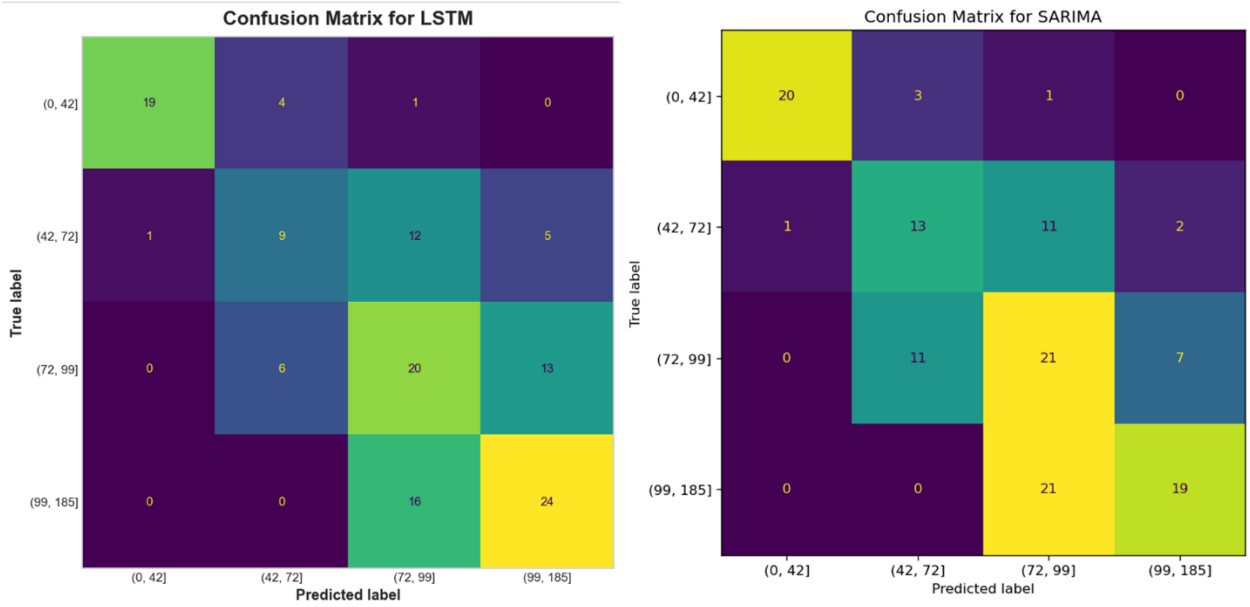


Figure 32 – Confusion matrix for LSTM split 1

Figure 32 shows the resulting confusion matrix for LSTM as well as for SARIMA. Both models had similar performance based on the validation metrics from Table 30 for SARIMA and Table 32 for LSTM. The accuracy of the confusion metrics reflects same conclusion, as LSTM had 55.4% and SARIMA with 56%, with 73 correct classifications, one more than LSTM, against a total of 130 possible values.

The lower values bin (0, 42] shows a good classification result but the higher bins of (72,99] and (99,185] have less precision as it can be seen on the adjacent cells of the matrix. This is in line with the residuals plot from SARIMA in Figure 28 which there is a wider band near those two ranges.

All 12 splits for LSTM, the confusion matrix accuracy had almost the same result and, on average resulted in 55%, which shows a stable performance across all validation partitions. Also, based on the learning curves for LSTM (appendix C), it is concluded that the hyperparameters selection provided good generalization of the data and both SARIMA and LSTM are good candidates for implementation.

4.4. Test partition results

The best hyperparameters from tables 26, 29 and 32 are applied in each test partition split to see how each model performed in a real word scenario.

Table 35 - Absolute and Relative Found in Test Partition.

Absolute Errors												
Split	*MAE*			*MSE*			*SMAPE*			*1-MDA*		
	ES	SARIMA	LSTM	ES	SARIMA	LSTM	ES	SARIMA	LSTM	ES	SARIMA	LSTM
1	24	19	19	989	568	500	45	29	37	0.38	0.31	0.31
2	22	15	15	714	369	373	34	25	23	0.35	0.22	0.22
3	35	22	20	1794	756	713	69	29	28	0.20	0.32	0.16
4	26	23	21	1127	925	897	40	35	33	0.36	0.36	0.40
5	22	20	19	850	603	495	40	40	39	0.23	0.27	0.31
6	26	30	25	1378	1256	915	34	44	33	0.33	0.13	0.13
7	22	20	20	749	604	645	28	28	22	0.27	0.31	0.35
8	16	18	16	396	518	433	27	34	28	0.15	0.35	0.27
9	31	25	21	1601	1092	659	55	43	29	0.33	0.29	0.21
10	20	22	19	652	696	522	26	28	24	0.31	0.23	0.31
11	19	21	19	541	648	623	29	36	32	0.20	0.20	0.24
12	29	23	21	1397	814	750	40	37	33	0.28	0.32	0.28
Avg	24	21	20	1016	737	627	39	34	30	0.28	0.27	0.26

Relative Errors (Naive Benchmark)												
Split	*MdRAE*			*GMRAE*			*MASE*			*MCP*		
	ES	SARIMA	LSTM	ES	SARIMA	LSTM	ES	SARIMA	LSTM	ES	SARIMA	LSTM
1	0.54	0.43	0.50	0.53	0.43	0.54	0.61	0.48	0.48	0.48	0.24	0.25
2	0.66	0.40	0.34	0.60	0.36	0.30	0.49	0.34	0.33	0.38	0.13	0.11
3	0.71	0.41	0.36	0.66	0.38	0.29	0.74	0.46	0.42	0.72	0.28	0.16
4	0.73	0.41	0.35	0.63	0.56	0.44	0.58	0.51	0.47	0.58	0.45	0.40
5	0.90	0.60	0.53	0.70	0.56	0.63	0.55	0.49	0.47	0.41	0.28	0.29
6	0.41	0.47	0.41	0.43	0.71	0.50	0.48	0.55	0.46	0.52	0.49	0.28
7	0.41	0.43	0.46	0.44	0.46	0.40	0.48	0.43	0.43	0.28	0.26	0.27
8	0.60	0.46	0.43	0.44	0.43	0.38	0.40	0.43	0.39	0.13	0.24	0.16
9	0.62	0.47	0.41	0.55	0.43	0.44	0.65	0.53	0.45	0.69	0.39	0.22
10	0.29	0.57	0.51	0.36	0.40	0.36	0.38	0.41	0.35	0.24	0.23	0.20
11	0.42	0.48	0.53	0.39	0.43	0.37	0.43	0.47	0.44	0.17	0.21	0.20
12	0.75	0.59	0.61	0.57	0.54	0.49	0.73	0.58	0.53	0.58	0.38	0.31
Avg	0.59	0.48	0.45	0.52	0.47	0.43	0.54	0.47	0.43	0.43	0.30	0.24

Source: Elaborated by the author

According to MCP criteria, on average, LSTM provide better results than SARIMA on test partition. This additional partition is design to test how well the the hyperparameters selected in the validation partition achieve a good generalization of the time series behavior.

Considering LSTM complexity and the time it takes to fit during the training, SARIMA results have exceeded expectations. The removal of coefficients with p value less than 5%, although statistically meaningful, reduces the flexibility for SARIMA. Overall the average of all metrics are very close between these two models, except for MSE which causes the MCP metric to weight in more for LSTM.

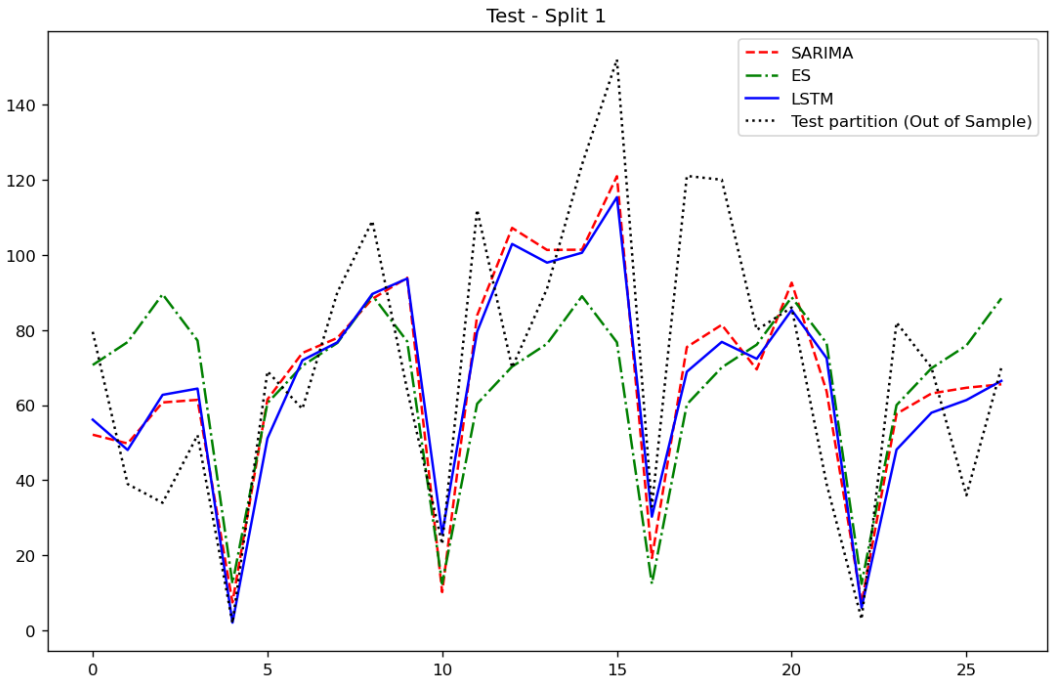


Figure 33 – Test split 1 comparison for all models

Part of this performance on SARIMA are due to the use of regressors such as the ones seen in Table 34. This is observed comparing to previous baselines used for this dataset before the implementation of this methodology.

Table 36 – Improvements in MAE observed from the base line against the methodology.

Split	*MAE using lowest AIC*		*MAE using lowest AIC*	*MAE using lowest MCP*		
	ES (base line)	SARIMA	SARIMA (w/exog)	ES	SARIMA (w/exog)	LSTM
1	31.34	24.49	18.68	24.32	19.24	19.25
2	26.31	17.38	15.11	21.65	15.11	14.61
3	37.69	28.05	20.63	35.17	21.67	19.91
4	25.32	25.67	23.38	26.02	22.98	21.06
5	22.28	22.03	20.15	22.50	20.15	19.15
6	38.55	38.36	29.84	26.17	29.84	24.80
7	21.49	21.01	20.77	22.02	19.87	19.75
8	20.50	16.80	18.53	16.37	17.74	15.95
9	40.00	31.40	21.26	30.90	25.05	21.15
10	20.73	21.82	21.59	19.97	21.59	18.51
11	20.11	19.29	20.55	18.92	20.55	19.27
12	32.38	32.16	20.50	29.04	23.33	21.32
Avg	28.06	24.87	20.91	24.42	21.43	19.56
			Grid search time (min)	15	150	3600

In Table 36 the result of the proposed methodology compares to previous forecasting method applied. The test partition MAE results are here used for its simple and intuitive interpretation:

a) Existing baseline for exponential smoothing based on best AIC found: 30% improving comparing to LSTM MAE (LSTM 19.56 x ES 28.06);
b) Existing baseline for SARIMA based on best AIC found: 21% improvement comparing to LSTM MAE (LSTM 19.56 x SARIMA 24.87).
c) When additional regressors are used for SARIMA, the resulting MAE found in the test partition on average when the hyperparameters with the lower AIC is chosen is equal in performance when the lower MCP is used.

When comparing the time taken to go through the process, LSTM takes much more time than parametric models and depending on the number of splits in the dataset this can become computing cumbersome as seen in Table 36. If processing power is available, this process can be automated without this impact.

Nevertheless, exponential smoothing when well calibrated, still provides good aproximation to the true observed values as seens in test splits 8 and 11. It is a less complex model to implement with a fast grid search time, even when testing all possibilities. For this reason business should be enconrage to have ES as a baseline where no model exists or naïve bases methods are used.

4.5. LSTM variability experiment

The grid search is the most common hyperparameter tuning approach given its simple and straightforward procedure. It is an uninformed search method, which means that it does not learn from its previous iterations. Using this method implies in testing every unique combination of hyperparameters in the search space to determine the vectors that yields the best performance. It’s easy to see the benefits of such a brute-force method, but an increase in the size of the hyperparameter search space will result in an exponential rise in run time and computation.

The random search treats iterations independently, but instead of searching for all hyperparameter sets in the search space, it evaluates a specific number of hyperparameter at random. Since it performs fewer trials in hyperparameter tuning,

the method requires less computation and run time than the grid search. The hyperparameter sets are selected at random so it runs the risk of missing the ideal set of hyperparameters if it is known that the best result will lie in the defined grid search space as presented in this work. However, with this approach, it is possible to choose an even higher set of hyperparameters to apply a random search, beyond the boundaries defined.

For neural networks another benefit of the Random Search is, by reducing the hyperparameter search time, the training process can be fit more than once for the same hyperparameters. The reason for this is that the random initial conditions of the weights (parameters) for the network can produce different results each time a given configuration is trained. So, running the training process more than once, it is possible to save not only the best hyperparameters but the best weights for each of the hidden states that was found. This experiment was done here with the test partition for all 12 splits with any fixed set of hyperparameters.

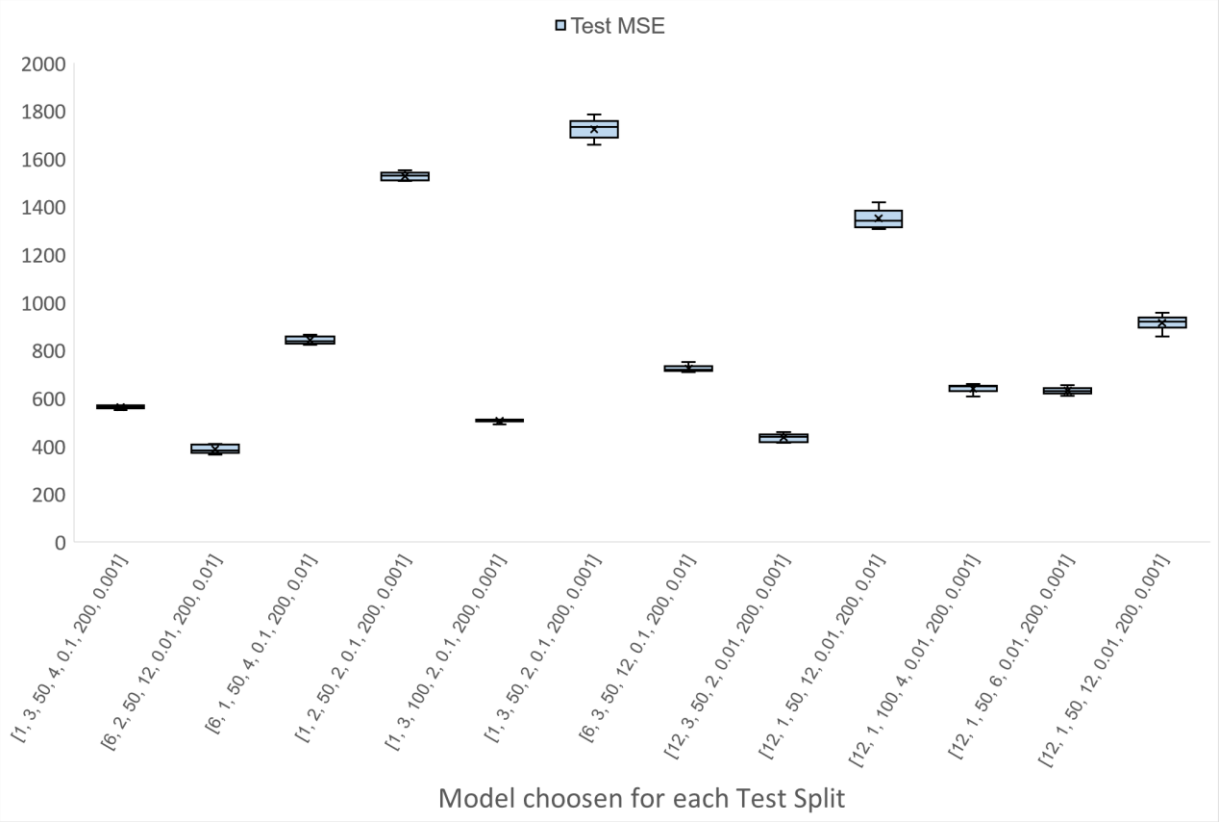


Figure 34 - Variability encounter in LSTM test partition results for MSE.

Source: Elaborated by the author

For each of the twelve selected best hyperparameters in each split, the LSTM network was trained 10 times. A box whisker plot is then presented for the three metrics that defines MCP (MSE, GMRAE and MDA). The x-axis shows the best hyperparameter vector and follows the order of the split from Table 32. Figure 34 shows the variability encounter in LSTM test partition results for MSE to not be significant due to the random initial conditions. An exception could be given for split 6 and 8 which shows a higher amplitude.

Figure 35 shows the variability encounter in LSTM test partition results for GMRAE. Splits 2, 11 and 12 shows more significant outliers than MSE.

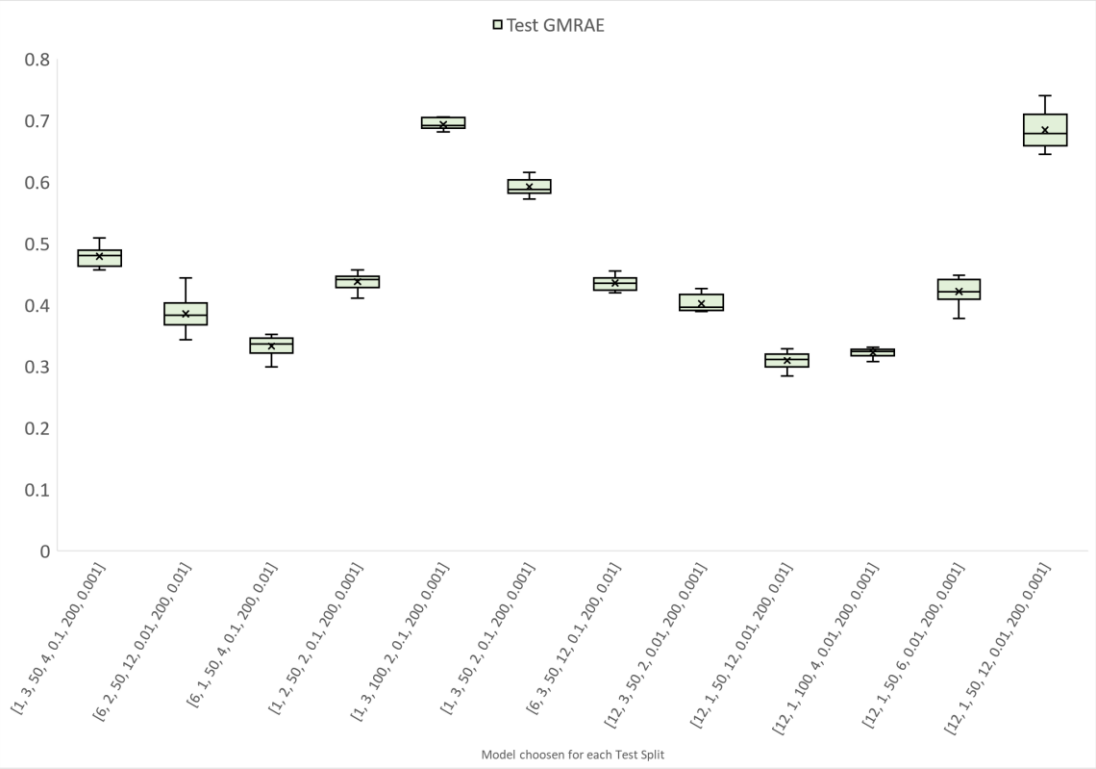


Figure 35 - Variability encounter in LSTM test partition results for GMRAE.

Source: Elaborated by the author

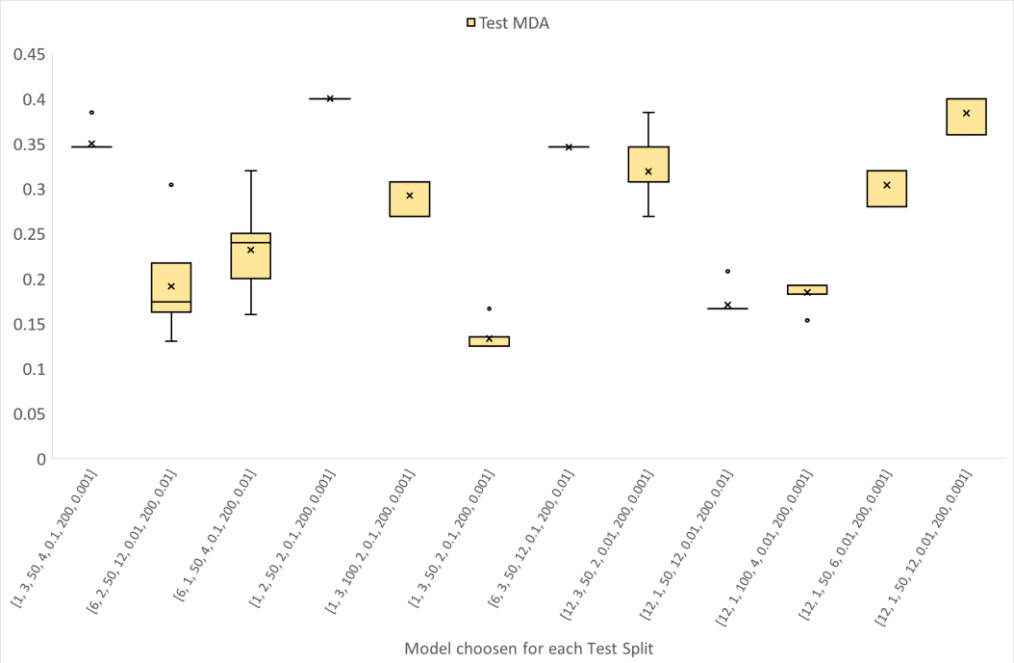


Figure 36 - Variability encounter in LSTM test partition results for MDA.

Source: Elaborated by the author

Figure 36 shows the variability encounter in LSTM test partition results for MDA. Outliers found in splits 2, 3 and 8 are significant and this may cause the MCP metric to worsen depending on the initial conditions. In consequence, final decision in which model to select could be impacted.

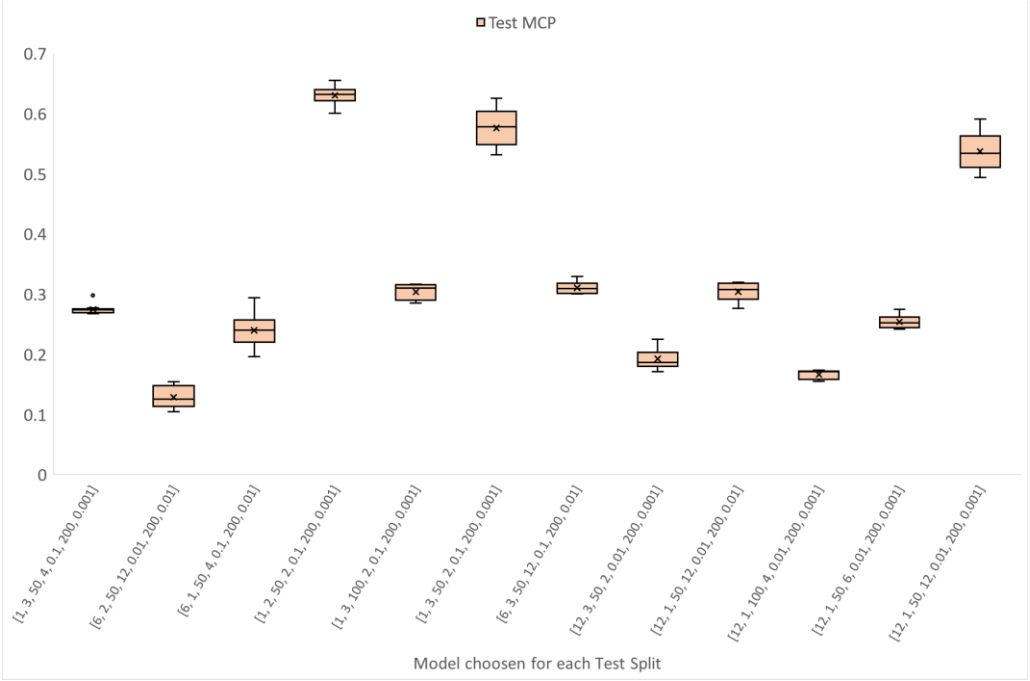


Figure 37 - Variability encounter in LSTM test partition results for MCP.

Source: Elaborated by the author

Figure 37 shows the variability encounter in LSTM test partition results for MCP as a consequence of the three previous figures. More significant outliers on splits 2 and 3 were caused by MDA from Figure 36. On split 12 it was due to GMRAE from Figure 35 and on split 6 due to MSE from Figure 34. The stochastic nature for the algorithm can provide different results, but a special attention to the MDA metric where significant changes in this value could cause an impact in the model selection decision. Nevertheless, one may argue still that the variability not to be significant. This can be true if the results found for LSTM where a lot better than on the other models, which was not observed here.

5. Conclusion and future work

In the proposed process in Figure 11, an optimized approach based on grid search framework for parametric/non-parametric models are employed, aiming to calibrate and select the best hyperparameter for each model. Such process is flexible, adaptative and efficient when dealing with multiple time series datasets that requires forecasting.

The different models and calibration methods for time series can be challenging to implement in a business environment without a pre-defined process. The implementation time for better accuracy can conflict with the speed that analysts or management needs to have when making a decision. This can lead to very simple assumptions when forecasting sales and not much accuracy. The process described here aims to overcome these challenges that once implemented can be easily scaled. Also, different models can be used for comparison, once their hyperparameters are defined, the model is added to algorithm and the results are then compared.

The test results shows that LSTM was able to achieve results marginally better than SARIMA. The flexibility found for SARIMA when using additional regressors makes it a competitive model even for non-parametric neural network that has a higher number of parameters and can capture non-linear relationships in the data.

This process also allowed to improve results from the previous baseline used (Table 36). On the other hand, opportunities and challenges remain. First, according to the SARIMA model diagnostics from Figure 28 (and also available in appendix B), (1) apply a robust outliers treatment and/or transformations of the target variable for possible improvements in residuals distribution and heteroscedasticity. Additionally, (2) the rejection of models with lags not statistically significantly reduces the flexibility

for SARIMA. Sensitivity analyses to see how the results change if a coefficient that is not statistically significant is kept in the model could help answer this question.

The use of additional regressor such as the proposed trend and seasonality modelling from section 3.4 showed to beneficial for SARIMA. Well-engineered regressors can boost the forecasting performance and reduce model complexity.

Considering the results that SARIMA and LSTM had, a hybrid model could be evaluated. It is worth noticing that in case the assumption of no autocorrelation in the residuals is violated in a forecasting model, implies that there exists information left which should be considered. Hybrid models combines the prediction of classical linear time series models (such as time series regression/ARIMA models) with nonlinear models based on machine learning methods such as Neural Networks. This has also been discussed in the M4 competition (Makridakis; Spiliotis; Assimakopoulos, 2020).

Zhang (2003) is one of the researchers who firstly proposed hybrid model by combining ARIMA as a linear model and Neural Networks as a nonlinear model for time series forecasting. Zhang's work has influenced many forecasting researchers to develop hybrid method for solving forecasting problems. The proposed hybrid system methodology consists of two steps. In the first step, an ARIMA model is used to analyze the linear part of the problem. In the second step, a neural network model is developed to model the residuals of the ARIMA model, which would be the non-linear part of it. This providing an interesting option to be used in the selection process described here and can also be seen how Smyl (2020) provided as a winning submission in M4 competitions.

A suggestion for using a hybrid model to the proposed process in this works is: First, SARIMA with the best hyperparameter found in the validation partition is fitted. SARIMA has proven to be a model that can captures autocorrelation of the series better than ES and LSTM, so it is chosen to be the model where the residuals are obtained. Then the residuals are modelled using LSTM to capture for non-linearities and long memory dependencies. The forecasting of SARIMA and LSTM are them summed to obtain the final forecasting.

Finally, another approach to forecasting is customer behavior Modelling. Today business is evolving from the product-centered to a customer-centered environment. This defines another possible approach to forecast sales that shifts from a time series modelling to modelled future customer behavior.

Internal data such as customer demographic data, transactional data, product-based data, customer review and complaint data are combined with external data such as government census, industry benchmark, market size analysis and economic data. This provides a mix of static data, multivariate time series data and textual data. In customer behavior prediction, the time series data are usually transformed into static data through aggregation and then combined with the other sources. So, the first challenge in this process is integrating multiple distributed data sources and types of data to reach the combined prediction results.

An example of data preprocessing in this case would be create a customer segments, add the historical product orders aggregated by a time interval and add other internal, external data related to each customer segment. First a classification problem is defined which predicts whether a customer segment will order the product and when. Then a regression problem predicts the quantity that each product will be ordered. Such methodology has substantial application not only in traditional S&OP forecasting but in marketing and sales, to drive customer fulfillment and retention.

Bibliography

Abbasimehr, H.; Shabani, M.; Yousefi, M. An optimized model using LSTM network for demand forecasting, Computers & Industrial Engineering, Volume 143, 106435, ISSN 0360-8352. 2020. DOI: https://doi.org/10.1016/j.cie.2020.106435.

Abolghasemi, M.; Beh, E.; Tarr, G.; Gerlach, R. Demand forecasting in supply chain: The impact of demand volatility in the presence of promotion, Computers & Industrial Engineering, Volume 142, 106380, ISSN 0360-8352. 2020. DOI: https://doi.org/10.1016/j.cie.2020.106380.

Abraham, B.; Chuang, A. Outlier Detection and Time Series Modeling. Technometrics, 31(2), 241–248, 1989. DOI: https://doi.org/10.2307/1268821.

Acar, Y.; Gardner Jr, E. S.; Forecasting method selection in a global supply chain, International Journal of Forecasting, Volume 28, Issue 4, p. 842-848, ISSN 0169-2070. 2012. DOI: https://doi.org/10.1016/j.ijforecast.2011.11.003

Adya, M.; Collopy, F.; Armstrong, J.S. Miles Kennedy, Automatic identification of time series features for rule-based forecasting, International Journal of Forecasting, Volume 17, Issue 2, p. 143-157, ISSN 0169-2070. 2001. DOI: https://doi.org/10.1016/S0169-2070(01)00079-6.

Alfaro, M.; Fuertes, G.; Vargas, M.; Sepúlveda, J.; Veloso-Poblete, M.; Martínez, E., “Forecast of chaotic series in a horizon superior to the inverse of the maximum lyapunov exponent,” Complexity, vol. 2018, Article ID 1452683, 2018. DOI: https://doi.org/10.1155/2018/1452683.

Anagnostopoulos, I.; Zeadally, S. and Exposito, E., Handling big data: research challenges and future directions. The Journal of Supercomputing, 72, p.1494-1516, 2016. DOI: https://doi.org/10.1007/s11227-016-1677-z

Syntetos, A. A.; Babai, Z.; Boylan, J. E.; Kolassa, S.. Supply chain forecasting: Theory, practice, their gap and the future, European Journal of Operational Research, Volume 252, Issue 1, 2016, p. 1-26, ISSN 0377-2217, 2016. DOI: https://doi.org/10.1016/j.ejor.2015.11.010.

Armstrong, J. S.; & Collopy, F. Error measures for generalizing about forecasting methods: Empirical comparisons. International Journal of Forecasting, Volume 8, Issue 1, p. 69-80, ISSN 0169-2070. 1992. DOI: https://doi.org/10.1016/0169-2070(92)90008-W.

ArunKumar, K.E.; Kalaga, D. V.; Kumar, M. S.; Kawaji, M.; Brenza, T. M. Comparative analysis of Gated Recurrent Units (GRU), long Short-Term memory (LSTM) cells, autoregressive Integrated moving average (ARIMA), seasonal autoregressive Integrated moving average (SARIMA) for forecasting COVID-19 trends, Alexandria Engineering Journal, Volume 61, Issue 10, p. 7585-7603, ISSN 1110-0168, 2022. DOI: https://doi.org/10.1016/j.aej.2022.01.011.

Bandara, K.; Bergmeir, C.; Smyl, S. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach, Expert Systems with Applications, Volume 140, 112896, ISSN 0957-4174, 2020. DOI: https://doi.org/10.1016/j.eswa.2019.112896.

Bengio Y.; Simard P.; Frasconi P.; Yoshua, P. F. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks / A publication of the IEEE Neural Networks Council, 5(2), p.157-166, 1994. DOI: 10.1109/72.279181

Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. The Journal of Machine Learning Research, 13. 281-305, 2012. DOI: 10.1109/72.279181.

Berk, R. A. Statistical Learning from a Regression Perspective. Springer Texts in Statistics, 2016. DOI: https://doi.org/ 10.1007/978-3-319-44048-4.

Billah, B,; Hyndman, R.; Koehler, A. B. Empirical Information Criteria for Time Series Forecasting Model Selection. Journal of Statistical Computation and Simulation. 75. 2003. DOI: https://doi.org/10.1080/00949650410001687208.

Billah, B.; King, M. L.; Snyder, R. D.; Koehler, A. B. Exponential smoothing model selection for forecasting, International Journal of Forecasting, Volume 22, Issue 2, Pages 239-247, ISSN 0169-2070, 2006. DOI: https://doi.org/10.1016/j.ijforecast.2005.08.002.

Bischl, B.; Binder, M.; Lang, M.; Pielok, T.; Richter, J.; Coors, S.; Thomas, J.; Ullmann, T.; Becker, M.; Boulesteix, A.L.; Deng, D. Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 13.2, E1484. 2021. DOI: https://doi.org/10.1002/widm.1484.

Bischl, B.; Richter, J;. Bossek, J; Horn, D.; Thomas, J.; Lang, M. mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions, arXiv, 2017, https://doi.org/10.48550/ARXIV.1703.03373

Boone, T.; Ganeshan, R.; Jain, A.; Sanders, N., R. Forecasting sales in the supply chain: Consumer analytics in the big data era, International Journal of Forecasting, Volume 35, Issue 1, Pages 170-180, ISSN 0169-2070, 2019. DOI: https://doi.org/10.1016/j.ijforecast.2018.09.003.

Box, G.E.P. and Jenkins, G.M. Time Series Analysis: Forecasting and Control, Holden-Day, San Francisco 1970. (Revised ed. 1976). DOI: https://doi.org/10.2307/3008255

Brockwell, P. J.; Davis, R. A. Introduction to Time Series and Forecasting. Springer Texts in Statistics, 2016. DOI: https://10.1007/978-3-319-29854-2.

Camacho, M.; Lopez-Buenache, G. Factor models for large and incomplete data sets with unknown group structure, International Journal of Forecasting, ISSN 0169-2070, 2022. DOI: https://doi.org/10.1016/j.ijforecast.2022.05.012.

Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly detection: A survey. ACM computing surveys (CSUR) 41.3, p.1-58, Article 15, July 2009. DOI: https://doi.org/10.1145/1541880.1541882.

Chollet, F. 2015. Keras. Available at: https://github.com/fchollet/keras.

Chu, C.; Zhang, G. P. A comparative study of linear and nonlinear models for aggregate retail sales forecasting, International Journal of Production Economics, Volume 86, Issue 3, Pages 217-231, ISSN 0925-5273, 2003. https://doi.org/10.1016/S0925-5273(03)00068-9.

Cleveland, R. B.; Cleveland, W. S.; McRae, J. E.; Terpenning, I. STL: A seasonal-trend decomposition procedure based on loess. Journal of Official Statistics, 6, p.3–33. 1990.

Collopy, F.; Armstrong, J.S. Rule-based forecasting: development and validation of an expert systems approach to combining time series extrapolations. Management science, 38.10, p.1394-1414, 1991. DOI: https://doi.org/10.1287/mnsc.38.10.1394

Cowpertwait, P. S.; Metcalfe, A. V. Introductory Time Series with R. Springer, New York , 2009. DOI: https://doi.org/10.1007/978-0-387-88698-5_10.

Crone, S. F.; Hibon, M.; Nikolopoulos, K. Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction, International Journal of Forecasting, Volume 27, Issue 3, p. 635-660, ISSN 0169-2070, 2011. DOI: https://doi.org/10.1016/j.ijforecast.2011.04.001.

De Gooijer, J. G.; Hyndman, R. J. 25 years of time series forecasting, International Journal of Forecasting, Volume 22, Issue 3, p. 443-473, ISSN 0169-2070, 2006. DOI: https://doi.org/10.1016/j.ijforecast.2006.01.001.

Durbin, J. “The Fitting of Time-Series Models.” Revue de l’Institut International de Statistique / Review of the International Statistical Institute, vol. 28, no. 3, p. 233–44. JSTOR, 1960. DOI: https://doi.org/10.2307/1401322.

Faraway, J. J. (2016). Does data splitting improve prediction?. Statistics and computing, p.49-60, 2016. DOI: https://doi.org/10.1007/s11222-014-9522-9

Feurer, M.; Hutter, F. Hyperparameter optimization, In: Hutter, F.; Kotthoff, L.; Vanschoren, J. (eds) Automated Machine Learning. The Springer Series on Challenges in Machine Learning. Springer, Cham., 2019. DOI: https://doi.org/10.1007/978-3-030-05318-5_1.

Fildes, R.; Nikolopoulos, K.; Crone, S.F.; Syntetos, A.A. Forecasting and operational research: a review. Journal of the Operational Research Society, 59.9, p.1150-1172, 2008

G.E.P. Box, D.R. Cox, An analysis of transformations, J Roy Stat Soc B, 26 (2) (1964), p. 211-252, 1964. DOI: https://doi.org/10.1111/j.2517-6161.1964.tb00553.x

Gardner, E. S. Exponential smoothing: The state of the art. Journal of Forecasting, 4.1, p.1–28, 1985. DOI: https://doi.org/10.1002/for.3980040103.

Gelbart, M. A.; Snoek, J.; Adams, R. P. Bayesian optimization with unknown constraints. In Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence (UAI'14). AUAI Press, Arlington, Virginia, USA, p. 250–259, 2014. DOI: https://doi.org/10.48550/arXiv.1403.5607

Granger, C. W. J.; Newbold, P. Forecasting transformed time series. Journal of the Royal Statistical Society, Series B, 38, p.189–203, 1976.

Hamilton, J. D. Time series analysis, Princeton University Press, Princeton, NJ.

Hewamalage, H.; Bergmeir, C.; Bandara, K. Recurrent Neural Networks for Time Series Forecasting: Current status and future directions, International Journal of Forecasting, Volume 37, Issue 1, 2021, p. 388-427, ISSN 0169-2070, 1994 DOI: https://doi.org/10.1016/j.ijforecast.2020.06.008.

Hlupić, T.; Oreščanin, D.; Petric, A. Time series model for sales predictions in the wholesale industry, 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 2020, p. 1263-1267, 2020 DOI: https://doi.org/10.23919/MIPRO48935.2020.9245255.

Hochreiter, S.; & Schmidhuber, J Long Short-Term Memory. Neural Computation; 9.8: p.1735–1780, 1997 https://doi.org/10.1162/neco.1997.9.8.1735.

Holt, C. E. Forecasting seasonals and trends by exponentially weighted averages (O.N.R. Memorandum No. 52), Carnegie Institute of Technology, Pittsburgh USA, 1957.

Hornik, K. Approximation capabilities of multilayer feedforward networks, Neural Networks, Volume 4, Issue 2, p. 251-257, ISSN 0893-6080, 1991. DOI: https://doi.org/10.1016/0893-6080(91)90009-T.

Huber, J.; Stuckenschmidt, H. Daily retail demand forecasting using machine learning with emphasis on calendric special days, International Journal of Forecasting, Volume 36, Issue 4, p. 1420-1438, ISSN 0169-2070, 2020 DOI: https://doi.org/10.1016/j.ijforecast.2020.02.005.

Hyndman, R. J., Koehler, A. B., Ord, J. K., & Snyder, R. D. Forecasting with exponential smoothing: The state space approach, Berlin: Springer-Verlag, 2018.

Hyndman, R.J.; Athanasopoulos, G. Forecasting: principles and practice, 2nd edition. 2018.

Jaderberg, M.; Dalibard, V.; Osindero, S.; Czarnecki, W. M.; Donahue,J.; Razavi, A.; Vinyals, O.; Green, T.; Dunning, I.; Simonyan, K.; Fernando, C.; Kavukcuoglu, K.. Population Based Training of Neural Networks, arXiv, 2017. DOI: https://doi.org/10.48550/arxiv.1711.09846.

Kumar, A.; Shankar, R.; Aljohani, N., R. A big data driven framework for demand-driven forecasting with effects of marketing-mix variables, Industrial Marketing Management, Volume 90, p. 493-507, ISSN 0019-8501, 2020. DOI: https://doi.org/10.1016/j.indmarman.2019.05.003.

Lemke, C.; Gabrys, B. Meta-learning for time series forecasting and forecast combination, Neurocomputing, Volume 73, Issues 10–12, p. 2006-2016, ISSN 0925-2312, 2010. DOI: https://doi.org/10.1016/j.neucom.2009.09.020.

Liu, H.; Simonyan, K.; Yang, Y. DARTS: Differentiable Architecture Search, arXiv:1806, 2018. DOI: https://doi.org/10.48550/arXiv.1806.09055.

Ljung, G.; Box, G. On a Measure of Lack of Fit in Time Series Models, Biometrika, 65, p297-303, 1978.

Luo, G. 2016. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. Network Modeling Analysis in Health Informatics and Bioinformatics, 5(1), pp.1-16.

Makridakis, S. and Hibon, M. Accuracy of Forecasting. An empirical investigation. Journal of the Royal Statistical Society, Series A, 142, p.97-145, 1979.

Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. The M4 Competition: Results, findings, conclusion and way forward, International Journal of Forecasting, Volume 34, Issue 4, p. 802-808, ISSN 0169-2070, 2018. DOI: https://doi.org/10.1016/j.ijforecast.2018.06.001.

Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. The M4 Competition: 100,000 time series and 61 forecasting methods, International Journal of Forecasting, Volume 36, Issue 1, p. 54-74, ISSN 0169-2070, 2020. DOI: https://doi.org/10.1016/j.ijforecast.2019.04.014.

Makridakis, S.; Wheelwright, S. C.; Hyndman, R. J. Forecasting: methods and applications, John Wiley & Sons, New York, 1998.

McCormick, G. P. Communications to the Editor—Exponential Forecasting: Some New Variations. Management Science, 15.5, p.311–320, 1969. DOI: https://doi.org/10.1287/mnsc.15.5.311.

Mentzer, J. T., & Cox, J. E. Familiarity, application, and performance of sales forecasting techniques. Journal of Forecasting, 3.1, p.27–36, 1984.

Morettin, Pedro A. Analise de Series Temporais. 2a Ed. São Paulo: Egard Blucher. ISBN 978-85-212-0389-6, 2006.

Moritz, S.; Bartz-Beielstein, T. ImputeTS: Time Series Missing Value Imputation in R. The R Journal, 9.1, p. 207, 2017. DOI: https://doi.org/10.32614/RJ-2017-009

Narayanan, A.; Sahin, F.; Robinson, E. P. Demand and order-fulfillment planning: The impact of point-of-sale data, retailer orders and distribution center orders on forecast accuracy. Journal of Operations Management, 65.5, p.468–486, 2019. DOI: https://doi.org/10.1002/joom.1026.

Nelson, H. L.; Granger, C. W. J. Experience with using the Box–Cox transformation when forecasting economic time series. Journal of Econometrics, 10, p.57–69, 1979.

Ngoc, T.T.; Le Van Dai; C.M.T.; Thuyen, C.M. Support vector regression based on grid search method of hyperparameters for load forecasting. Acta Polytechnica Hungarica, 18.2, p.143-158. 2021.

Nguyen, H.D.; Tran, K.P.; Thomassey, S.; Hamad, M. Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management, International Journal of Information Management, Volume 57, 102282, ISSN 0268-4012, 2021. DOI: https://doi.org/10.1016/j.ijinfomgt.2020.102282.

Ozdemir, A. C.; Buluş, K.; Zor, K. Medium- to long-term nickel price forecasting using LSTM and GRU networks, Resources Policy, Volume 78, 102906, ISSN 0301-4207, 2022. DOI: https://doi.org/10.1016/j.resourpol. 2022.102906.

Parmezan, A.R.S.; Lee, H.D.; Wu, F.C. Metalearning for choosing feature selection algorithms in data mining: proposal of a new framework, Expert Syst. Appl. 75, p.1–24, ISSN 0957-4174, 2017. DOI:https://doi.org/10.1016/j.eswa. 2017.01.013.

Parmezan, A.R.S.; Souza, V. M. A.; Batista, G. E.A.P.A. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model, Information Sciences, Volume 484, p. 302-337, ISSN 0020-0255. 2019. DOI: https://doi.org/10.1016/j.ins.2019.01.076.

Pegels, C. C. Exponential forecasting: Some new variations. Management Science, 15(5), 311–315. 1969.

Probst, P.; Boulesteix, A.L.; Bischl, B. Tunability: Importance of hyperparameters of machine learning algorithms. The Journal of Machine Learning Research, 20.1, p.1934-1965. 2019.

Proietti T; Lütkepohl H. Does the Box–Cox transformation help in forecasting macroeconomic time series?. International Journal of Forecasting, 29.1 p.88-99, 2013.

Rashid, T. A.; Fattah, P.; Awla, D. K. Using Accuracy Measure for Improving the Training of LSTM with Metaheuristic Algorithms, Procedia Computer Science, Volume 140, p. 324-333, ISSN 1877-0509, 2018. DOI: https://doi.org/10.1016/j.procs.2018.10.307.

Reimers, N.; Gurevych, I. Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. ArXiv, abs/1707.06799, 2017. DOI: https://doi.org/10.48550/arXiv.1707.06799.

Rere, L. M. R.; Fanany, M. I.; Arymurthy, A. M. Metaheuristic Algorithms for Convolution Neural Network, Computational Intelligence and Neuroscience, vol. 2016, Article ID 1537325, p.13, 2016. DOI: https://doi.org/10.1155/2016/1537325.

Salinas, D.; Flunkert, V.; Gasthaus, J.; Januschowski, T. DeepAR: Probabilistic forecasting with autoregressive recurrent networks, International Journal of orecasting, Volume 36, Issue 3, p.1181-1191, ISSN 0169-2070, 2020. DOI: https://doi.org/10.1016/j.ijforecast.2019.07.001.

Seabold, S.; Perktold, J. Statsmodels: Econometric and statistical modeling with python. Proceedings of the 9th Python in Science Conference. 2010.

Shcherbakov, M.; Brebels, A.; Shcherbakova, N.L.; Tyukov, A.; Janovsky, T.A.; Kamaev, V.A. A survey of forecast error measures. World Applied Sciences Journal, 24, p.171-176. 2013. DOI: https://doi.org/10.5829/idosi.wasj.2013.24.itmies.80032.

Shen, Z.; Zhang, Y.; Lu, J.; Xu, J.; Xiao, G. A novel time series forecasting model with deep learning, Neurocomputing, Volume 396, p. 302-313, ISSN 0925-2312, 2020. DOI: https://doi.org/10.1016/j.neucom.2018.12.084.

Smiti, A. A critical overview of outlier detection methods, Computer Science Review, Volume 38, 100306, ISSN 1574-0137, 2020. DOI: https://doi.org/10.1016/j.cosrev.2020.100306.

Smyl, S. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting, International Journal of Forecasting, Volume 36, Issue 1, p.75-85, ISSN 0169-2070, 2020. DOI: https://doi.org/10.1016/j.ijforecast.2019.03.017

Smyl, S.; Kuber, K. Data Preprocessing and Augmentation for Multiple Short Time Series Forecasting with Recurrent Neural Networks. 2016.

Snoek, J.; Larochelle, H.; Adams, R.P. Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 25. 2012

Staudemeyer, R.C. and Morris, E.R., Understanding LSTM--a tutorial into long short-term memory recurrent neural networks. arXiv preprint arXiv:1909.09586, 2019. DOI: https://doi.org/10.48550/arXiv.1909.09586

Syntetos, A. A.; Babai, Z., Boylan, J. E.; Kolassa, S.; Nikolopoulos, K. Supply chain forecasting: Theory, practice, their gap and the future, European Journal of Operational Research, Volume 252, Issue 1, 2016, P. 1-26, ISSN 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2015.11.010.

Taylor, J. W. Exponential smoothing with a damped multiplicative trend. International Journal of Forecasting, 19, p.715–725. ISSN 0169-2070, 2003. DOI: https://doi.org/10.1016/S0169-2070(03)00003-7.

Tealab A. Time series forecasting using artificial neural networks methodologies: A systematic review, Future Computing and Informatics Journal, Volume 3, Issue 2, 2018, p. 334-340, ISSN 2314-7288. DOI: https://doi.org/10.1016/j.fcij.2018.10.003.

Vokurka, R.J.; Flores, B.E.; Pearce, S.L. Automatic feature identification and graphical support in rule-based forecasting: a comparison, International Journal of Forecasting, Volume 12, Issue 4, 1996, p. 495-512, ISSN 0169-207. DOI: https://doi.org/10.1016/S0169-2070(96)00682-6.

Wang, X.; Smith-Miles, K.; Hyndman, R. Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series, Neurocomputing, Volume 72, Issues 10–12, 2009, p. 2581-2594, ISSN 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2008.10.017.

Wang, Y.; Smola, A.; Maddix, D. C.; Gasthaus, J.; Foster, D.; Januschowski, T. Deep Factors for Forecasting, arXiv, 2019. DOI: https://doi.org/10.48550/arxiv.1905.12417

Winters, P. R. Forecasting sales by exponentially weighted moving averages. Management Science, 6, p.324–342, 1960.

Yang, L.; Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 415, p.295-316, 2020. DOI: https://doi.org/10.1016/j.neucom.2020.07.061

Zhang, G.P. Time Series Forecasting using a Hybrid ARIMA and Neural Network Model. Neurocomputing, 50, p.159-175, 2003. DOI: https://doi.org/10.1016/S0925-2312(01)00702-0

Zhang, S.; Chen, Y.; Zhang, W.; Feng, R. A novel ensemble deep learning model with dynamic error correction and multi-objective ensemble pruning for time series forecasting. Information Sciences, 544, p.427–445. ISSN 0020-0255, 2021. DOI: https://doi.org/10.1016/j.ins.2020.08.053.

Zhang, A.; Lipton, Z.,C.; Li, M.; Smola, A., J. Dive into deep learning. Cambridge University Press, Release 0.14.3, 2020

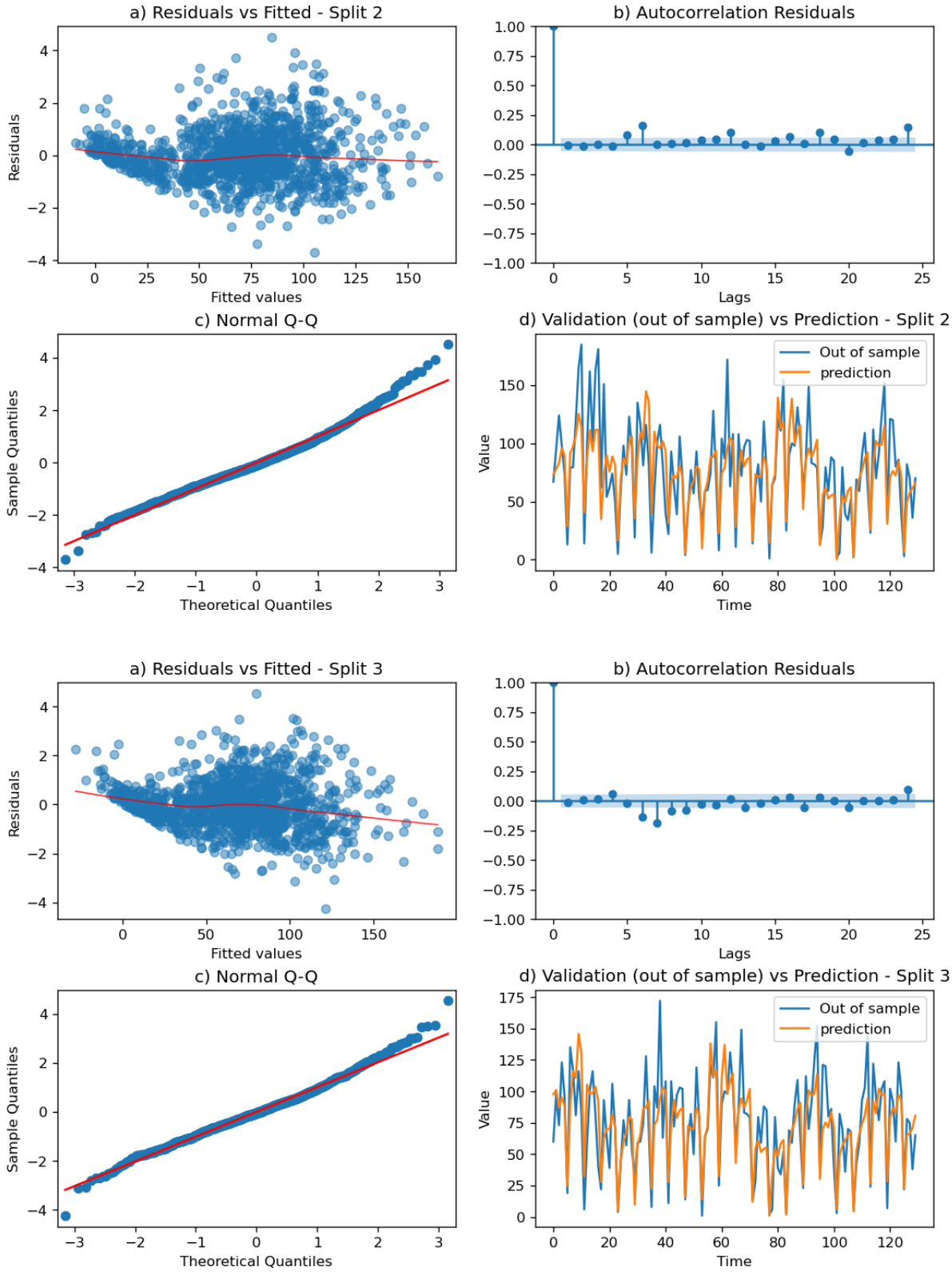
SMITH, E. B. Basic chemical thermodynamics. 6th ed. London: Imperial College Press, 2014. 226 p. ISBN: 1783263369.

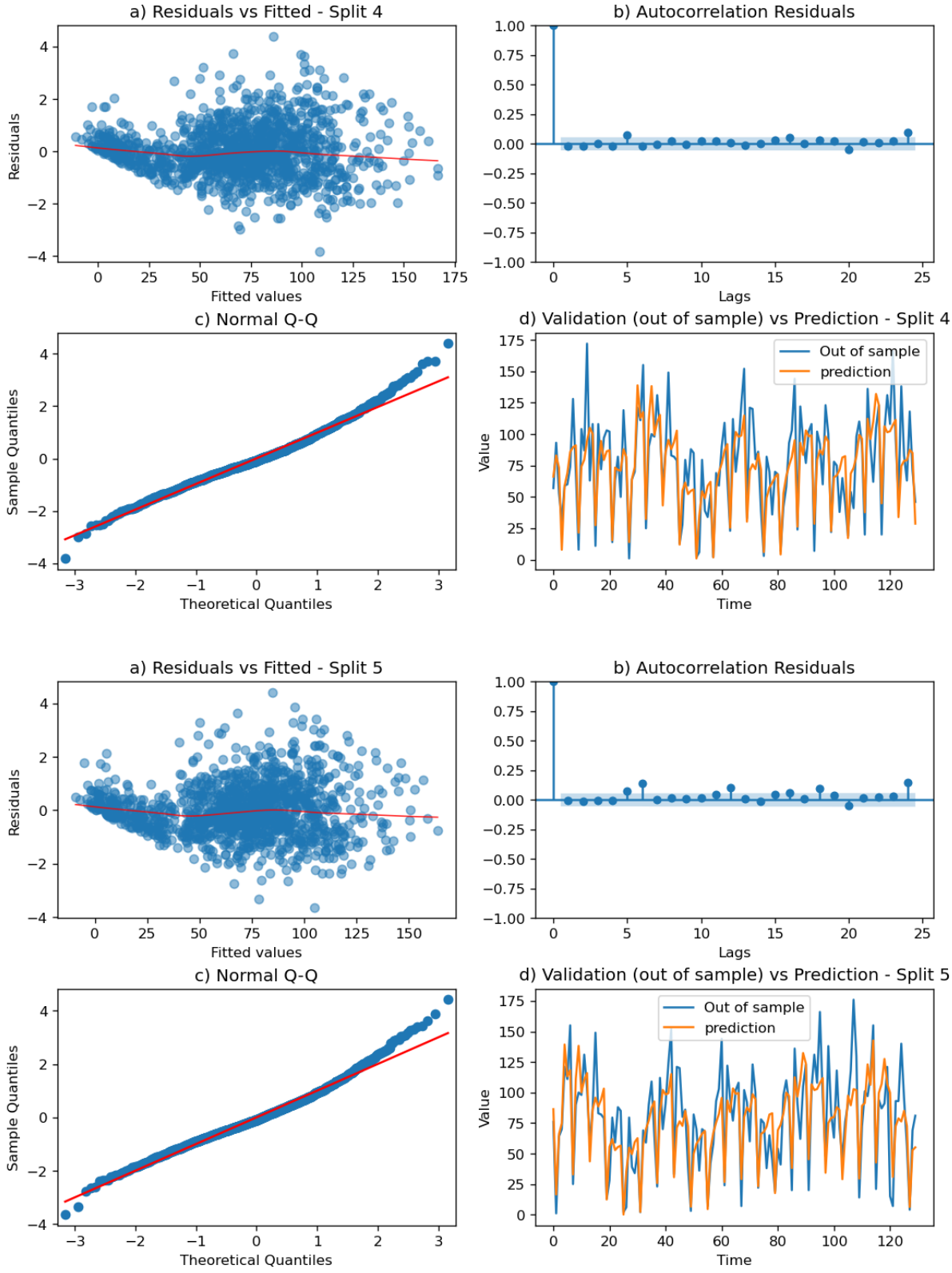
Appendix A - Recursive calculations and forecasts formulas for all Exponential smoothing methods

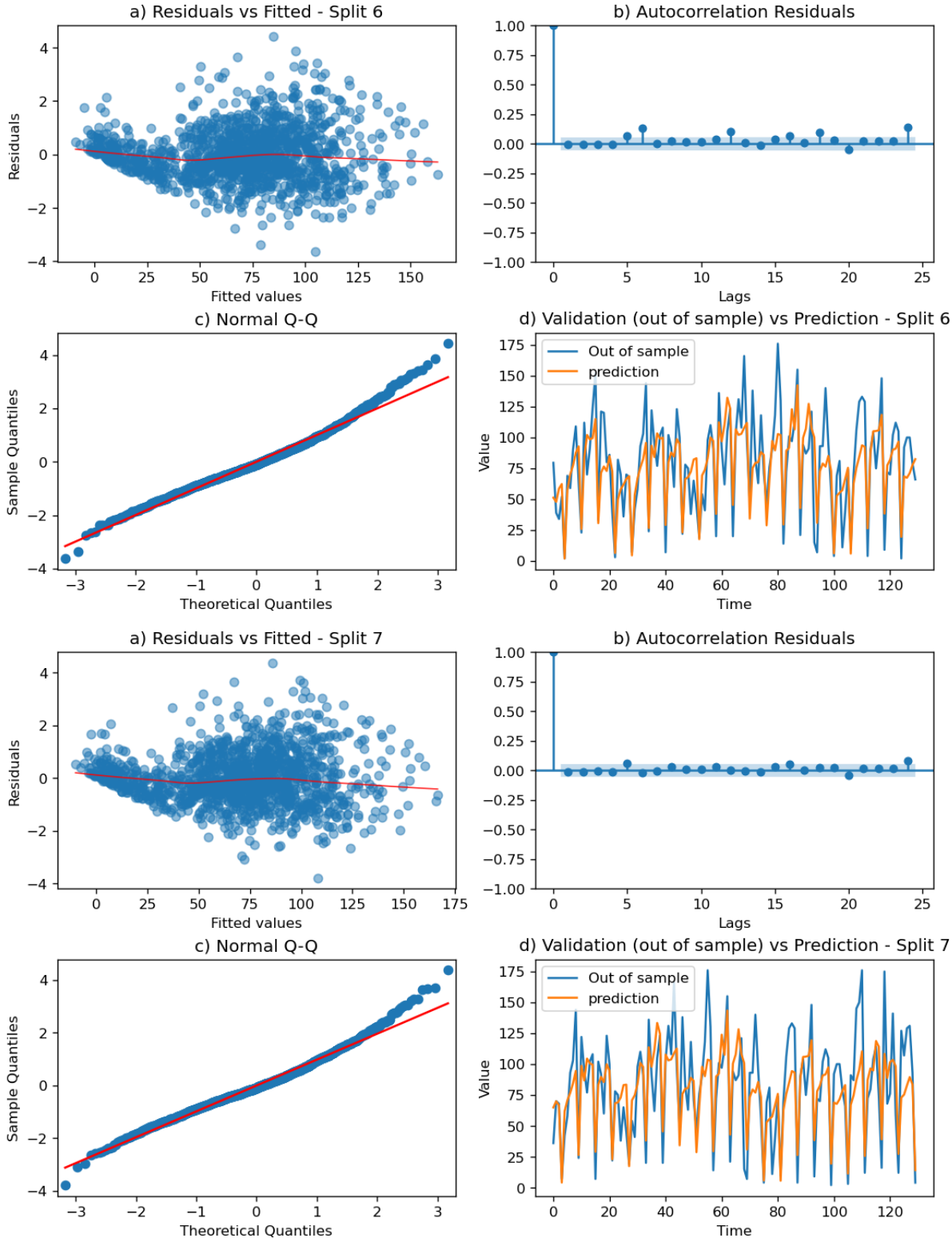
In each case l_t denotes the series level at time t, b_t denotes the slope at time t, s_t denotes the seasonal component of the series at time t and m denotes the periodicity. α, β, γ and the damped parameter ∅ are the parameters. $\emptyset_h = \emptyset + \emptyset^2 + \cdots + \emptyset^h$ and h are the forecasting point.

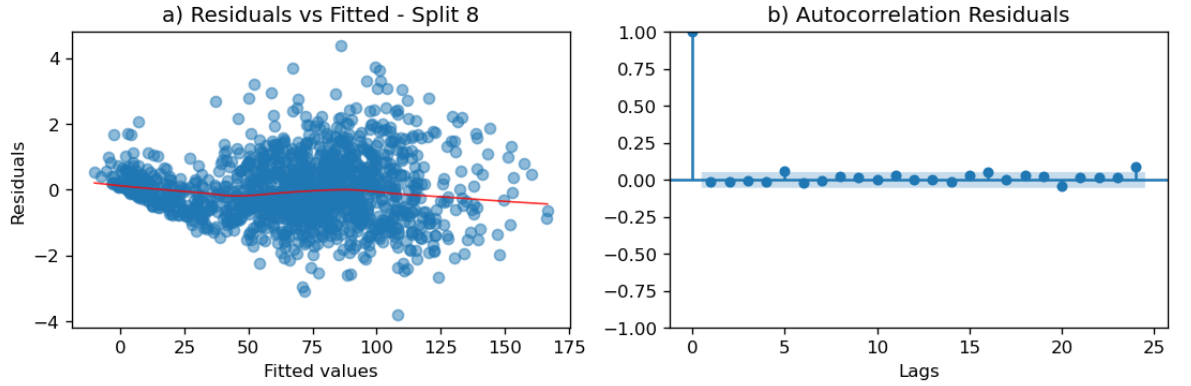
Trend	Seasonal		
	N	A	M
N	$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$ $\hat{y}_{t+h\|t} = \ell_t$	$\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)\ell_{t-1}$ $s_t = \gamma(y_t - \ell_{t-1}) + (1-\gamma)s_{t-m}$ $\hat{y}_{t+h\|t} = \ell_t + s_{t-m+h_m^+}$	$\ell_t = \alpha(y_t / s_{t-m}) + (1-\alpha)\ell_{t-1}$ $s_t = \gamma(y_t / \ell_{t-1}) + (1-\gamma)s_{t-m}$ $\hat{y}_{t+h\|t} = \ell_t s_{t-m+h_m^+}$
A	$\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$ $\hat{y}_{t+h\|t} = \ell_t + hb_t$	$\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$ $s_t = \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}$ $\hat{y}_{t+h\|t} = \ell_t + hb_t + s_{t-m+h_m^+}$	$\ell_t = \alpha(y_t / s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$ $s_t = \gamma(y_t / (\ell_{t-1} + b_{t-1})) + (1-\gamma)s_{t-m}$ $\hat{y}_{t+h\|t} = (\ell_t + hb_t)s_{t-m+h_m^+}$
A_d	$\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + \phi b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}$ $\hat{y}_{t+h\|t} = \ell_t + \phi_h b_t$	$\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + \phi b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}$ $s_t = \gamma(y_t - \ell_{t-1} - \phi b_{t-1}) + (1-\gamma)s_{t-m}$ $\hat{y}_{t+h\|t} = \ell_t + \phi_h b_t + s_{t-m+h_m^+}$	$\ell_t = \alpha(y_t / s_{t-m}) + (1-\alpha)(\ell_{t-1} + \phi b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}$ $s_t = \gamma(y_t / (\ell_{t-1} + \phi b_{t-1})) + (1-\gamma)s_{t-m}$ $\hat{y}_{t+h\|t} = (\ell_t + \phi_h b_t)s_{t-m+h_m^+}$
M	$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}b_{t-1}$ $b_t = \beta^*(\ell_t / \ell_{t-1}) + (1-\beta^*)b_{t-1}$ $\hat{y}_{t+h\|t} = \ell_t b_t^h$	$\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)\ell_{t-1}b_{t-1}$ $b_t = \beta^*(\ell_t / \ell_{t-1}) + (1-\beta^*)b_{t-1}$ $s_t = \gamma(y_t - \ell_{t-1}b_{t-1}) + (1-\gamma)s_{t-m}$ $\hat{y}_{t+h\|t} = \ell_t b_t^h + s_{t-m+h_m^+}$	$\ell_t = \alpha(y_t / s_{t-m}) + (1-\alpha)\ell_{t-1}b_{t-1}$ $b_t = \beta^*(\ell_t / \ell_{t-1}) + (1-\beta^*)b_{t-1}$ $s_t = \gamma(y_t / (\ell_{t-1}b_{t-1})) + (1-\gamma)s_{t-m}$ $\hat{y}_{t+h\|t} = \ell_t b_t^h s_{t-m+h_m^+}$
M_d	$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}b_{t-1}^{\phi}$ $b_t = \beta^*(\ell_t / \ell_{t-1}) + (1-\beta^*)b_{t-1}^{\phi}$ $\hat{y}_{t+h\|t} = \ell_t b_t^{\phi_h}$	$\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)\ell_{t-1}b_{t-1}^{\phi}$ $b_t = \beta^*(\ell_t / \ell_{t-1}) + (1-\beta^*)b_{t-1}^{\phi}$ $s_t = \gamma(y_t - \ell_{t-1}b_{t-1}^{\phi}) + (1-\gamma)s_{t-m}$ $\hat{y}_{t+h\|t} = \ell_t b_t^{\phi_h} + s_{t-m+h_m^+}$	$\ell_t = \alpha(y_t / s_{t-m}) + (1-\alpha)\ell_{t-1}b_{t-1}^{\phi}$ $b_t = \beta^*(\ell_t / \ell_{t-1}) + (1-\beta^*)b_{t-1}^{\phi}$ $s_t = \gamma(y_t / (\ell_{t-1}b_{t-1}^{\phi})) + (1-\gamma)s_{t-m}$ $\hat{y}_{t+h\|t} = \ell_t b_t^{\phi_h} s_{t-m+h_m^+}$

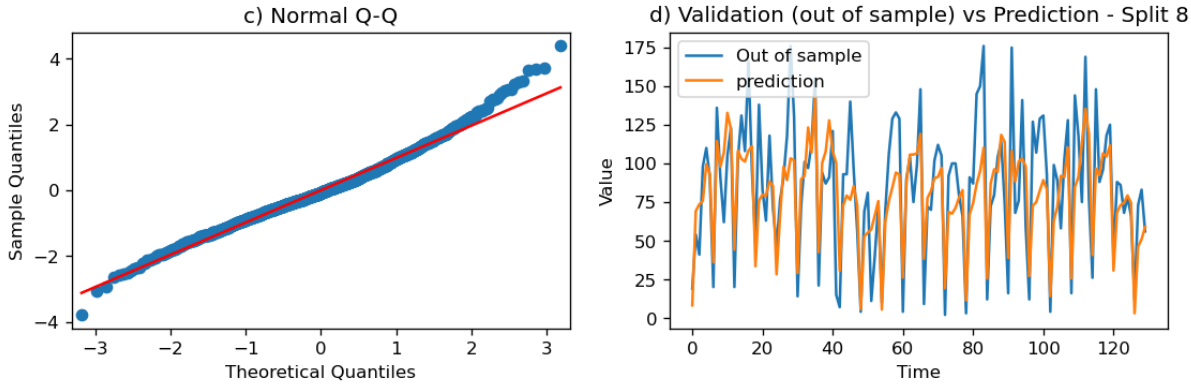
Appendix B – SARIMA residuals plots and validation predictions

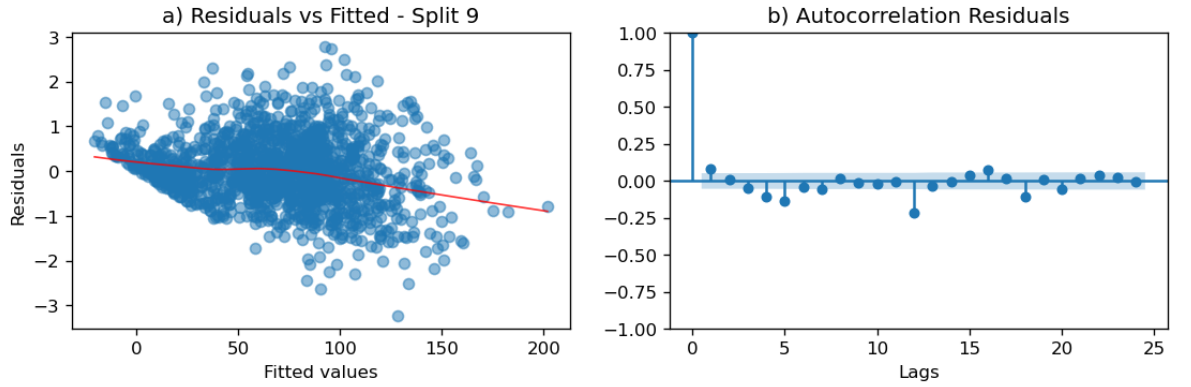


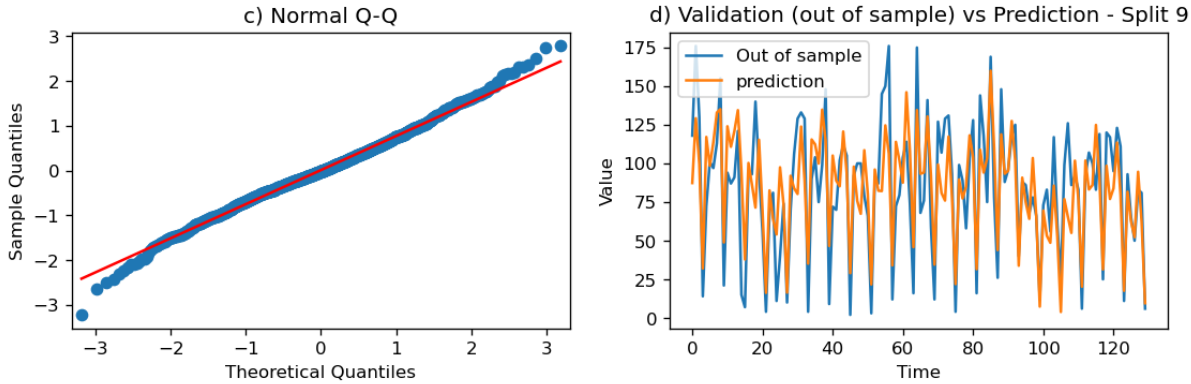

a) Residuals vs Fitted - Split 4
Residuals
Fitted values
b) Autocorrelation Residuals
Lags
c) Normal Q-Q
Sample Quantiles
Theoretical Quantiles
d) Validation (out of sample) vs Prediction - Split 4
Out of sample
prediction
Value
Time
a) Residuals vs Fitted - Split 5
Residuals
Fitted values
b) Autocorrelation Residuals
Lags
c) Normal Q-Q
Sample Quantiles
Theoretical Quantiles
d) Validation (out of sample) vs Prediction - Split 5
Out of sample
prediction
Value
Time

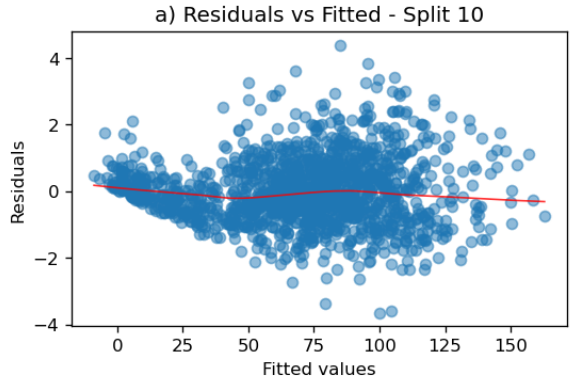

a) Residuals vs Fitted - Split 6
Residuals
Fitted values
b) Autocorrelation Residuals
Lags
c) Normal Q-Q
Sample Quantiles
Theoretical Quantiles
d) Validation (out of sample) vs Prediction - Split 6
Out of sample
prediction
Value
Time
a) Residuals vs Fitted - Split 7
Residuals
Fitted values
b) Autocorrelation Residuals
Lags
c) Normal Q-Q
Sample Quantiles
Theoretical Quantiles
d) Validation (out of sample) vs Prediction - Split 7
Out of sample
prediction
Value
Time

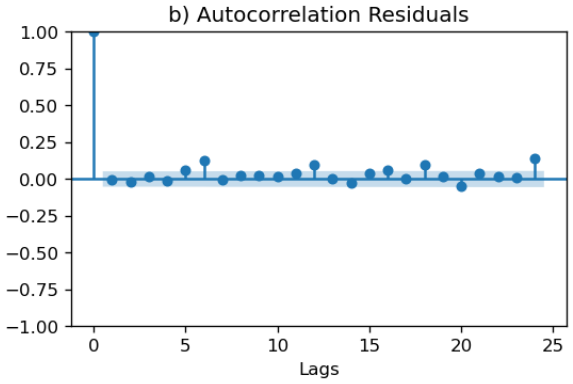

a) Residuals vs Fitted - Split 8
Residuals
Fitted values
b) Autocorrelation Residuals
Lags

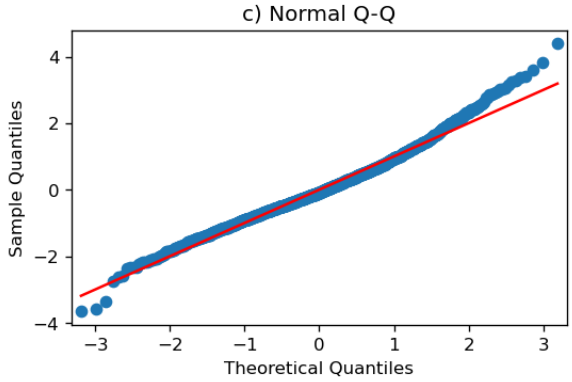

c) Normal Q-Q
Sample Quantiles
Theoretical Quantiles
d) Validation (out of sample) vs Prediction - Split 8
Out of sample
prediction
Value
Time

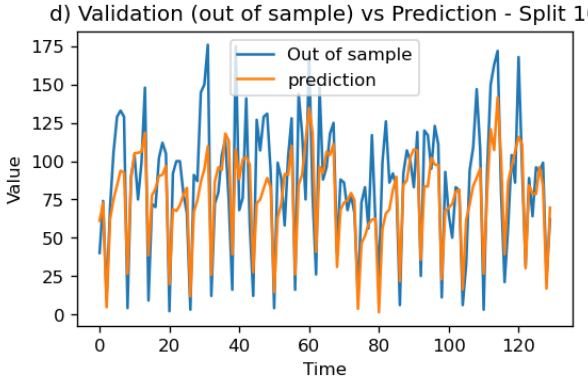

a) Residuals vs Fitted - Split 9
Residuals
Fitted values
b) Autocorrelation Residuals
Lags

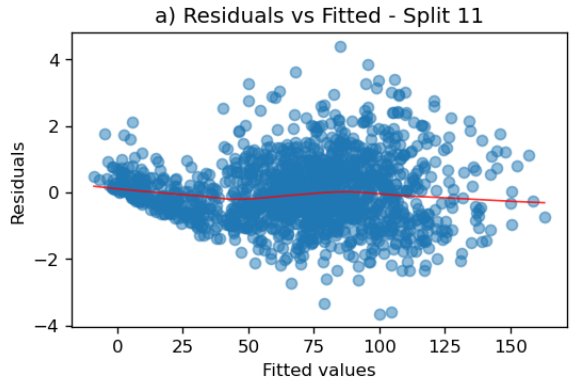

c) Normal Q-Q
Sample Quantiles
Theoretical Quantiles
d) Validation (out of sample) vs Prediction - Split 9
Out of sample
prediction
Value
Time

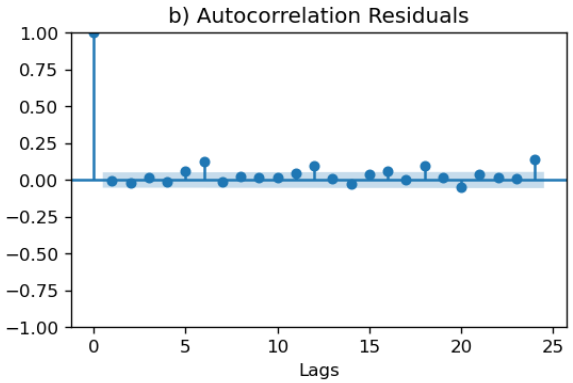

a) Residuals vs Fitted - Split 10
Residuals
Fitted values

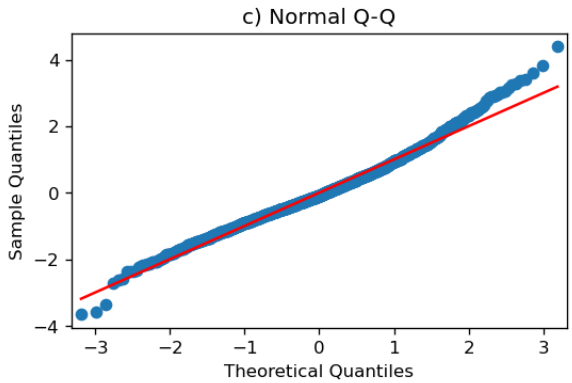

b) Autocorrelation Residuals
Lags

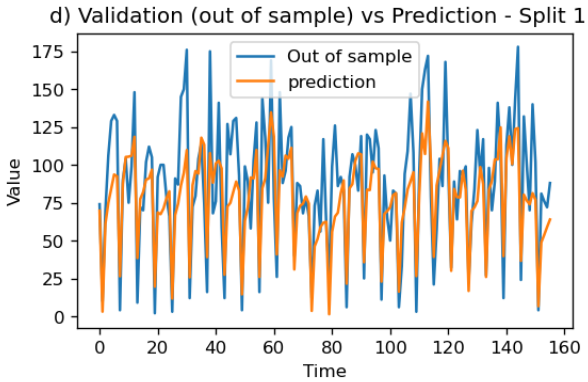

c) Normal Q-Q
Sample Quantiles
Theoretical Quantiles

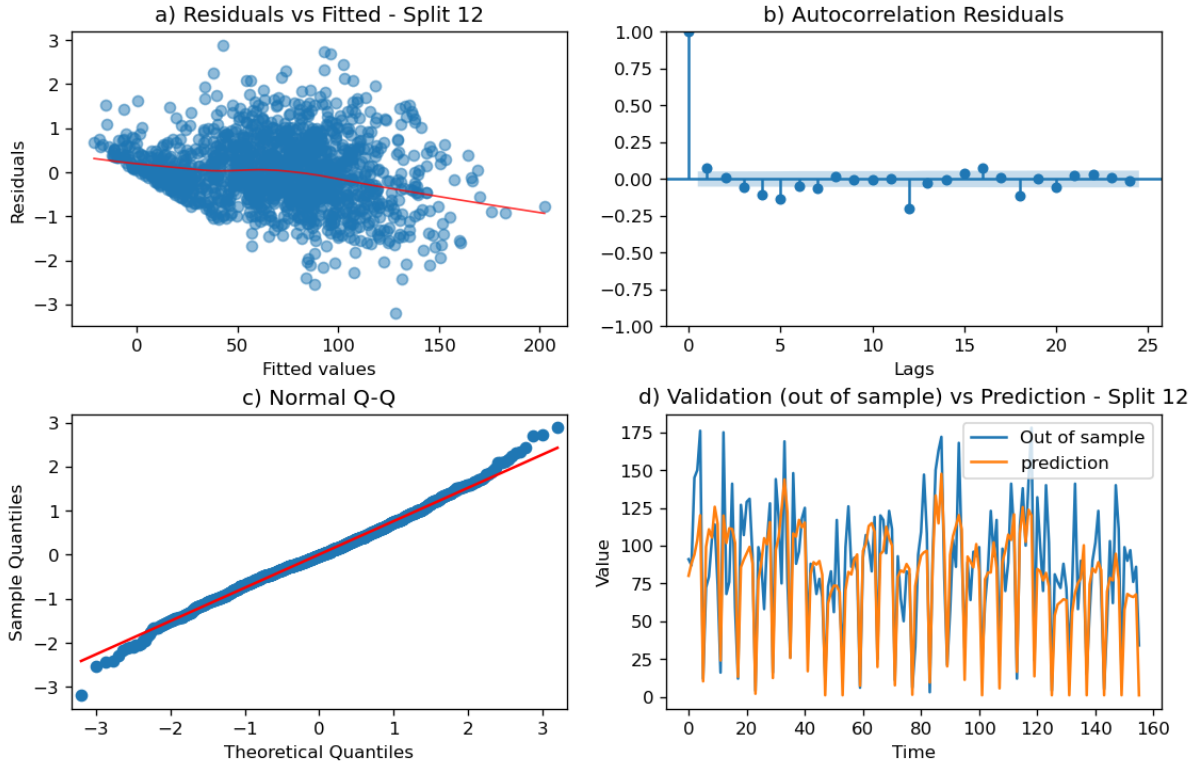

d) Validation (out of sample) vs Prediction - Split 10
Out of sample
prediction
Value
Time


a) Residuals vs Fitted - Split 11
Residuals
Fitted values

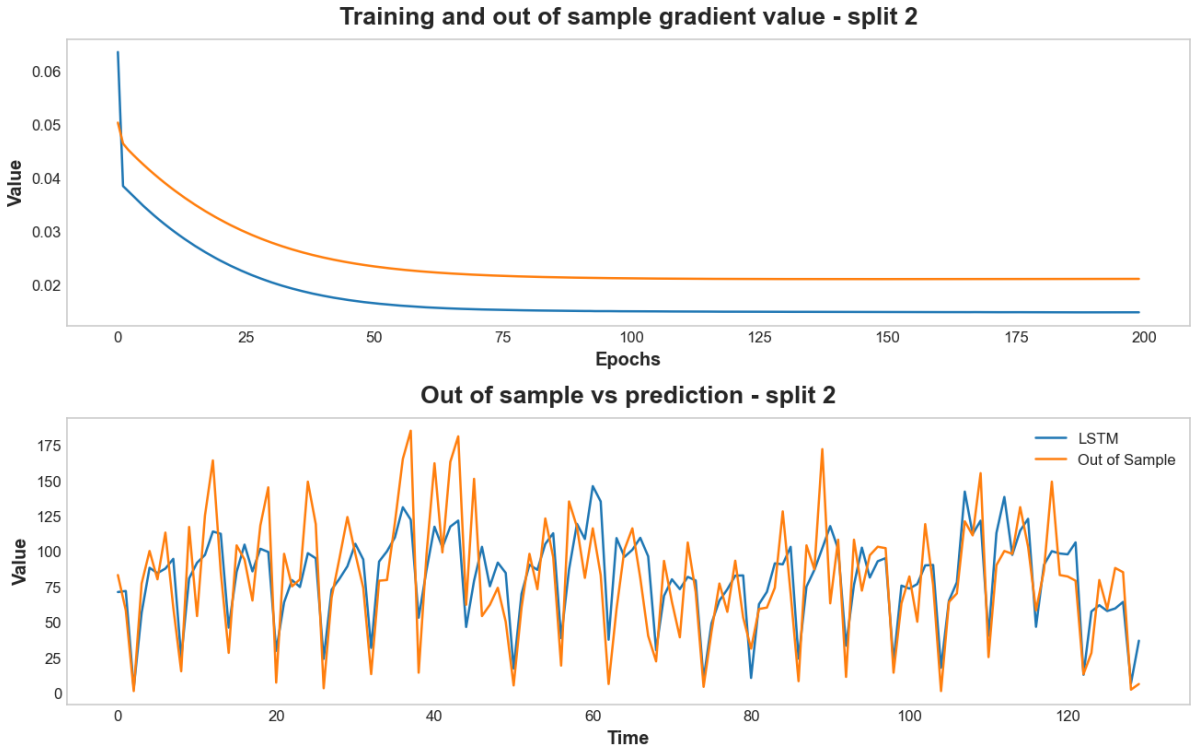

b) Autocorrelation Residuals
Lags

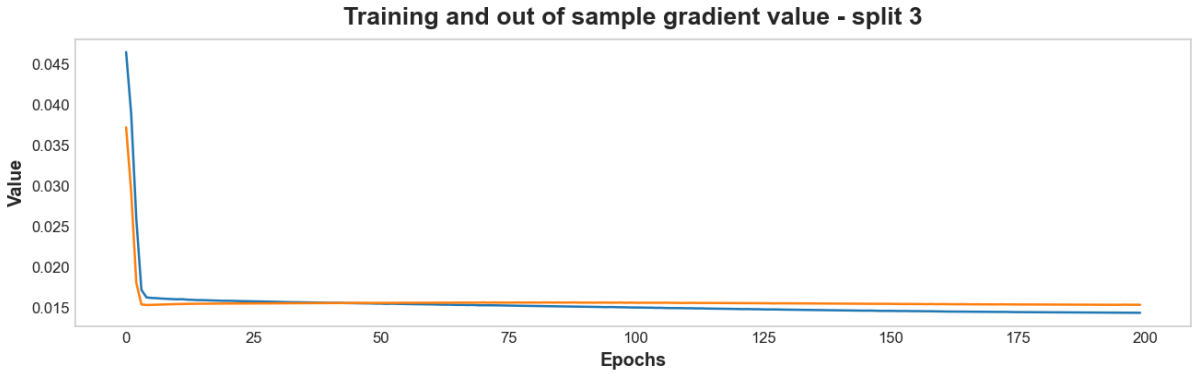

c) Normal Q-Q
Sample Quantiles
Theoretical Quantiles

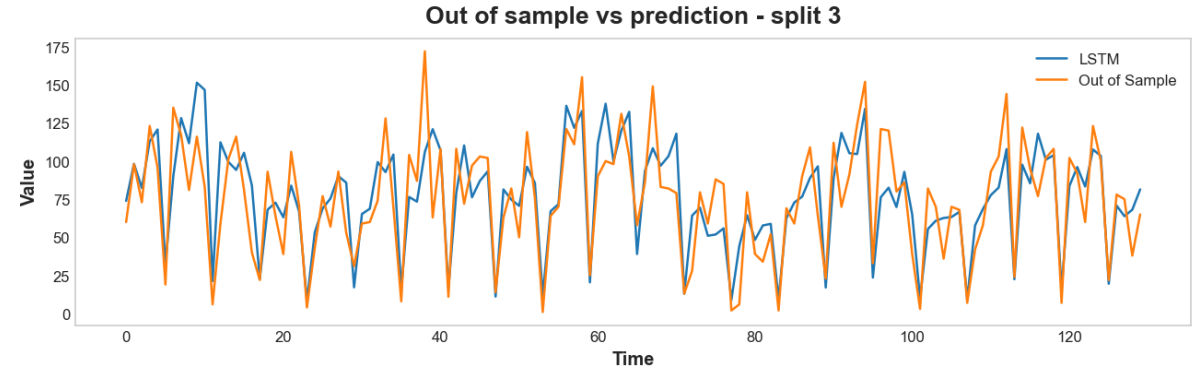

d) Validation (out of sample) vs Prediction - Split 11
Out of sample
prediction
Value
Time

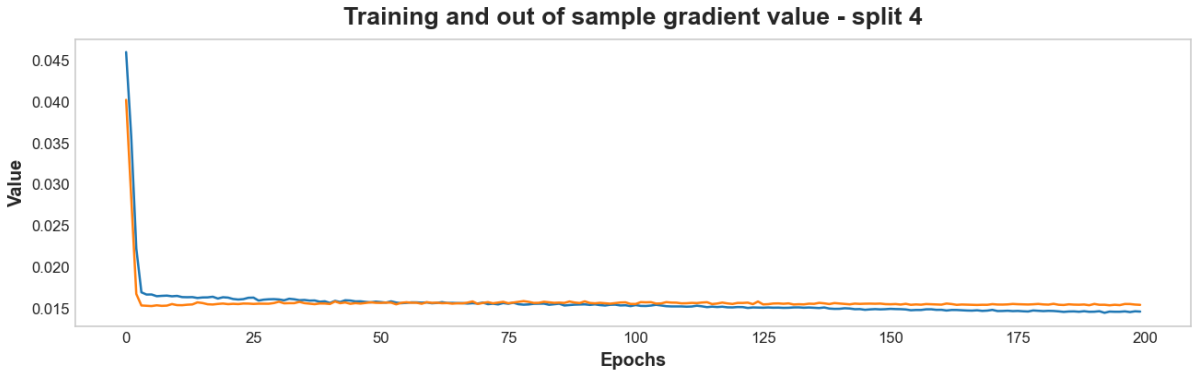


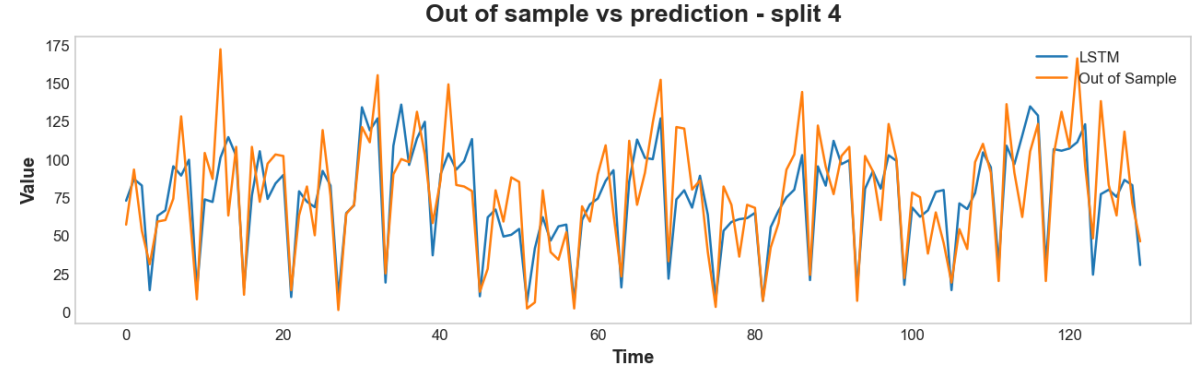
Appendix C – Training curves and validation predictions for LSTM splits

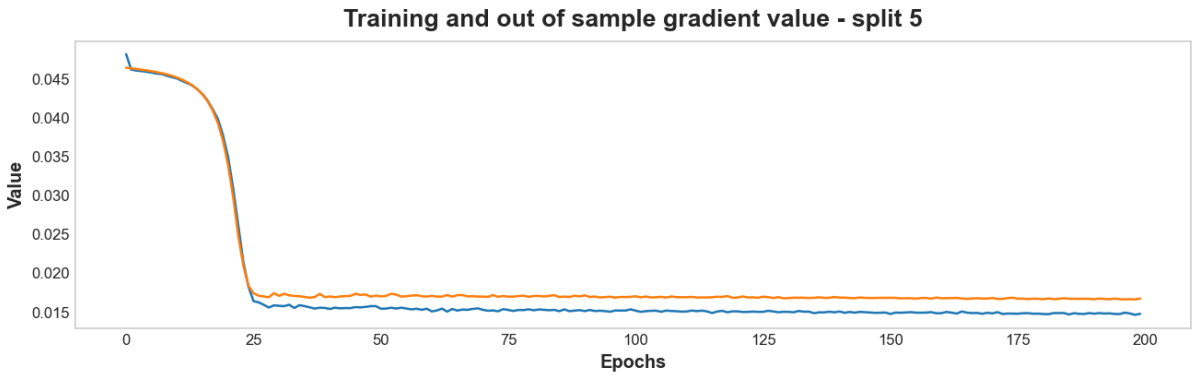


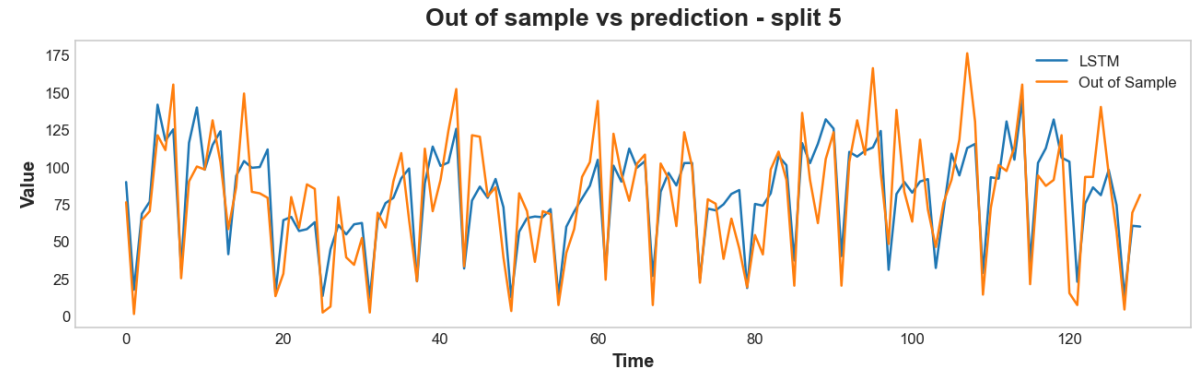

Training and out of sample gradient value - split 3
Value
Epochs
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0
25
50
75
100
125
150
175
200

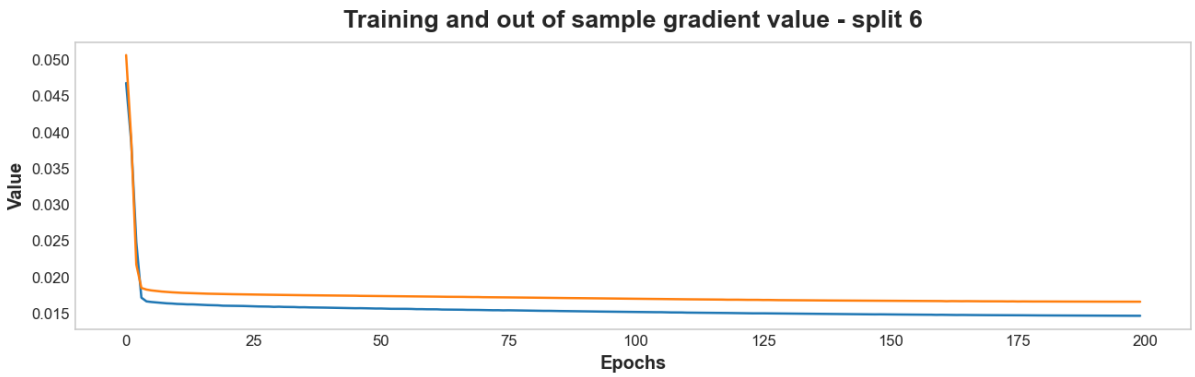

Out of sample vs prediction - split 3
LSTM
Out of Sample
Value
Time
175
150
125
100
75
50
25
0
0
20
40
60
80
100
120

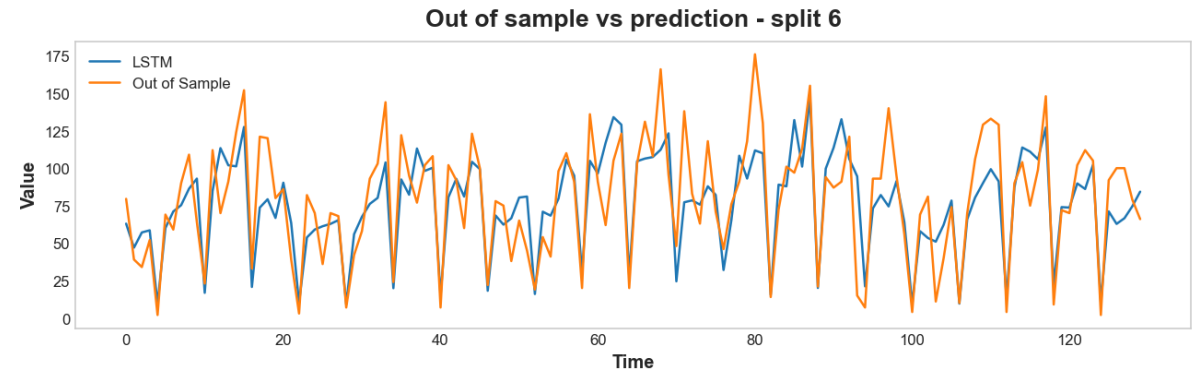

Training and out of sample gradient value - split 4
Value
Epochs
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0
25
50
75
100
125
150
175
200

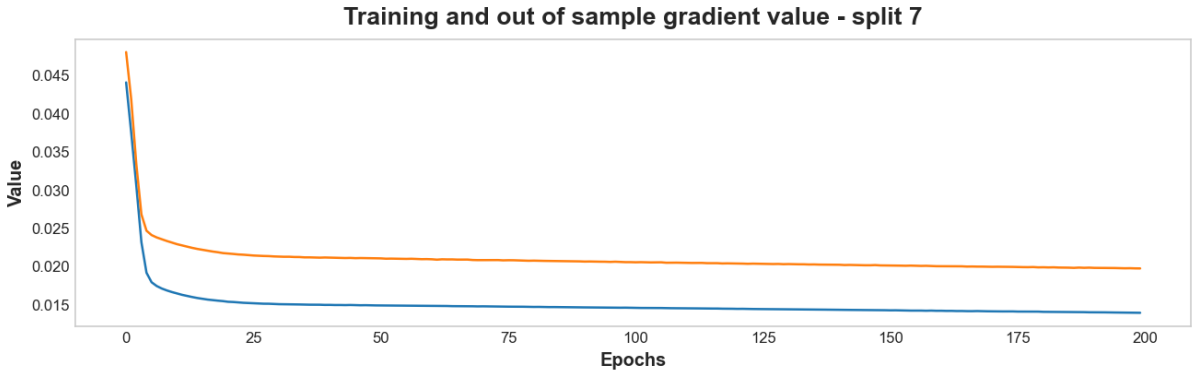

Out of sample vs prediction - split 4
LSTM
Out of Sample
Value
Time
175
150
125
100
75
50
25
0
0
20
40
60
80
100
120

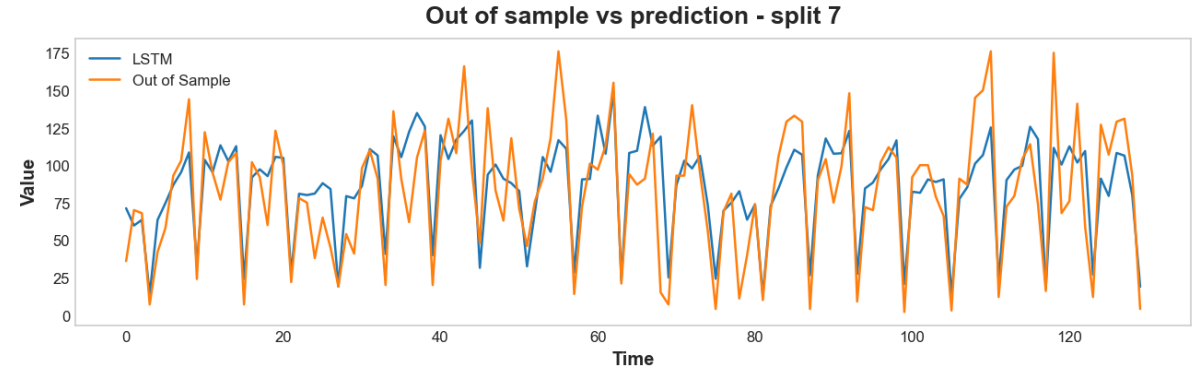

Training and out of sample gradient value - split 5
Value
Epochs
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0
25
50
75
100
125
150
175
200

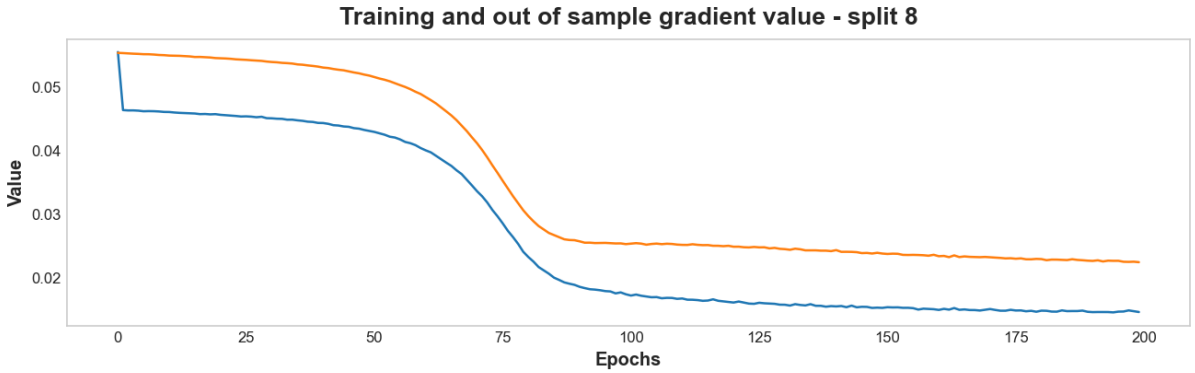

Out of sample vs prediction - split 5
LSTM
Out of Sample
Value
Time
175
150
125
100
75
50
25
0
0
20
40
60
80
100
120

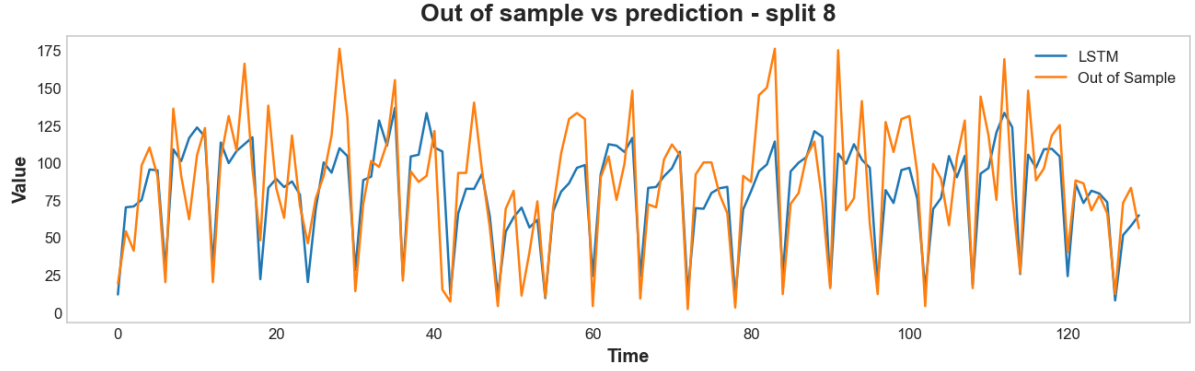

Training and out of sample gradient value - split 6
Value
Epochs
0.050
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0
25
50
75
100
125
150
175
200

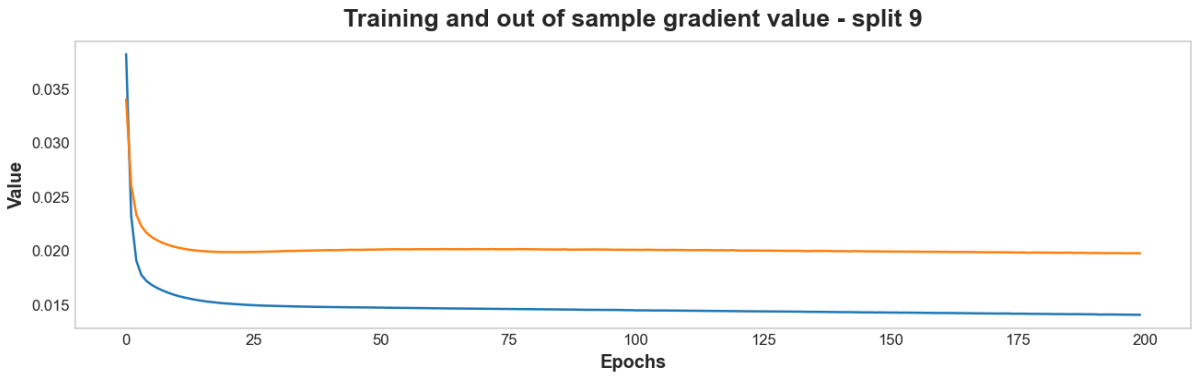

Out of sample vs prediction - split 6
LSTM
Out of Sample
Value
Time
175
150
125
100
75
50
25
0
0
20
40
60
80
100
120

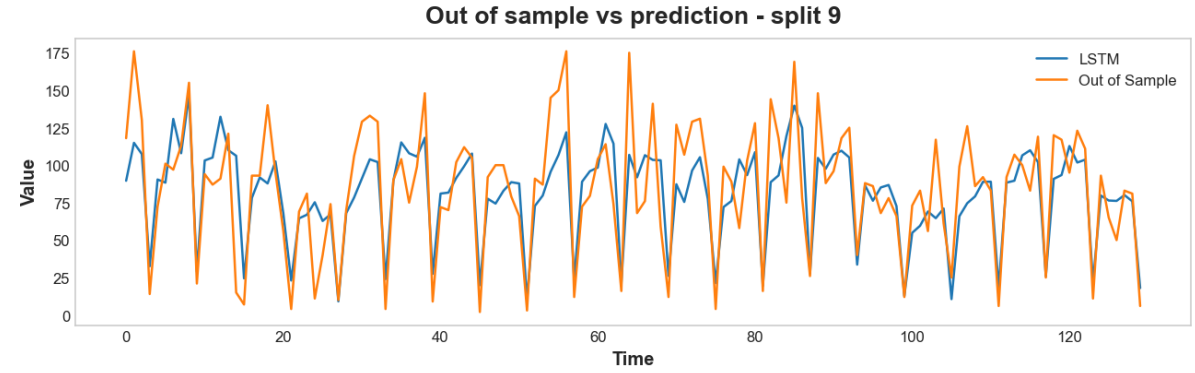

Training and out of sample gradient value - split 7
Value
Epochs
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0
25
50
75
100
125
150
175
200

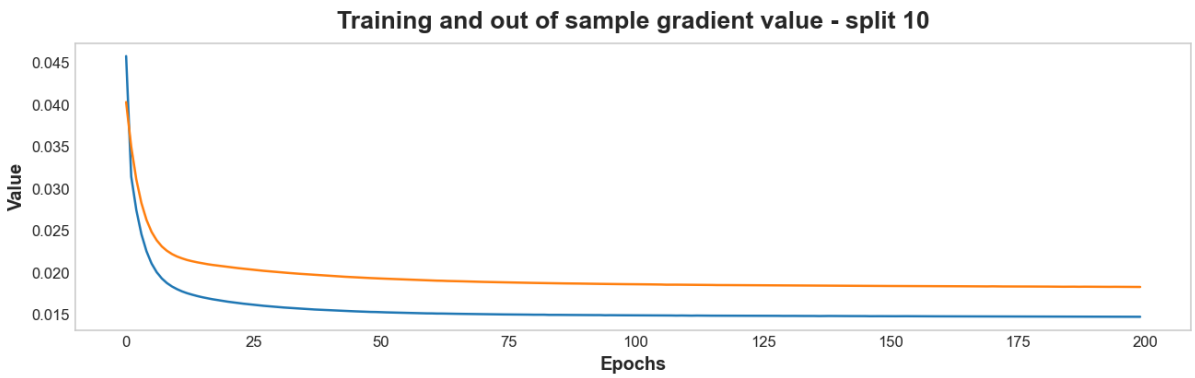

Out of sample vs prediction - split 7
LSTM
Out of Sample
Value
Time
175
150
125
100
75
50
25
0
0
20
40
60
80
100
120

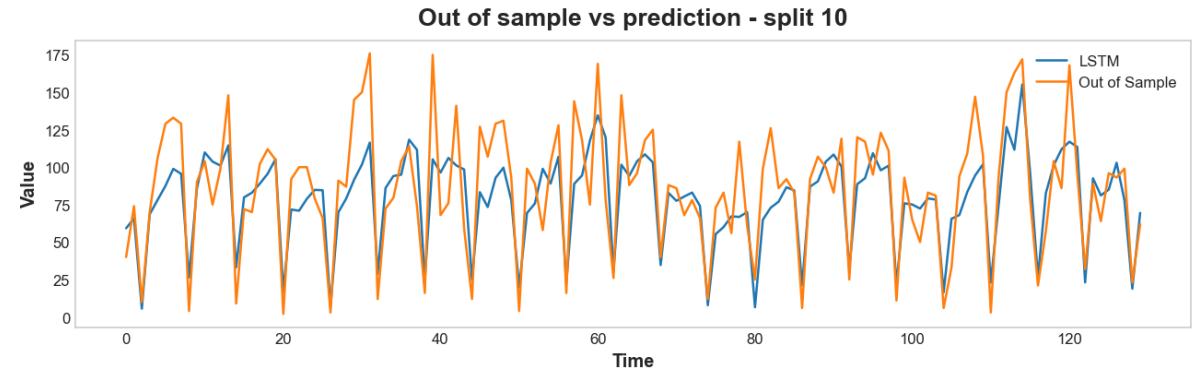

Training and out of sample gradient value - split 8
Value
Epochs
0.05
0.04
0.03
0.02
0
25
50
75
100
125
150
175
200

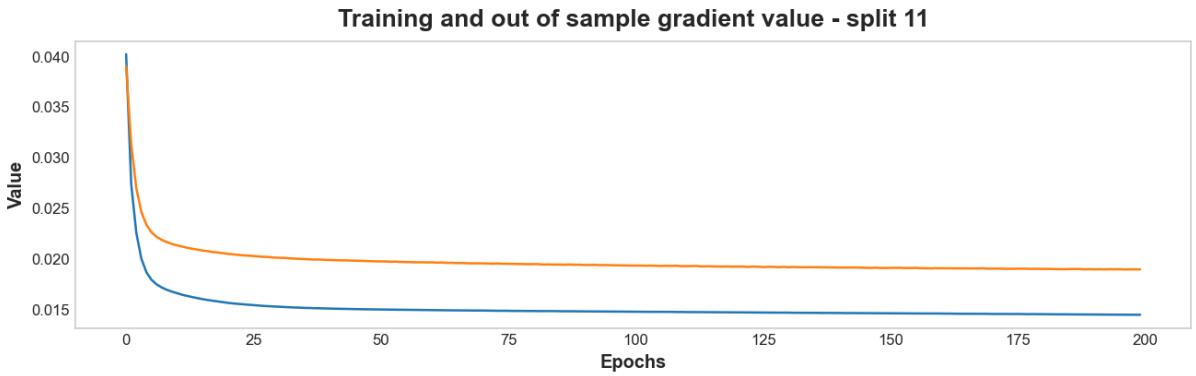

Out of sample vs prediction - split 8
LSTM
Out of Sample
Value
Time
175
150
125
100
75
50
25
0
0
20
40
60
80
100
120

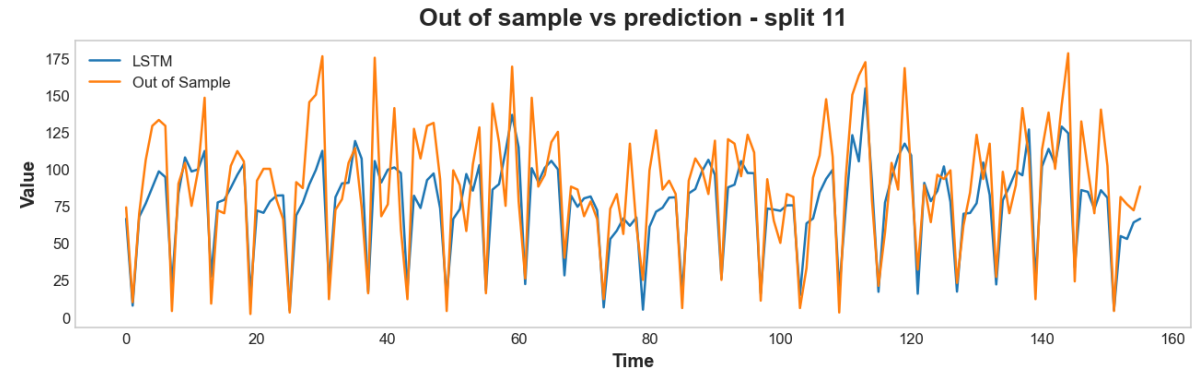

Training and out of sample gradient value - split 9
Value
Epochs
0.035
0.030
0.025
0.020
0.015
0
25
50
75
100
125
150
175
200

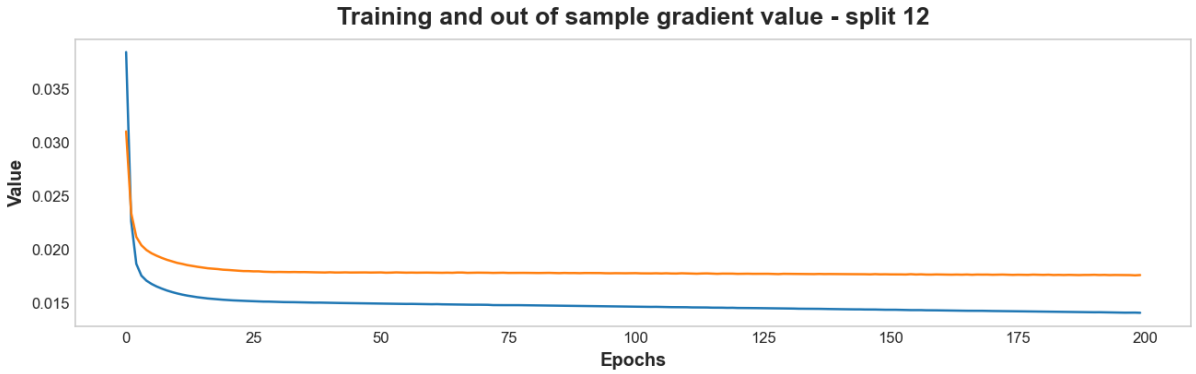

Out of sample vs prediction - split 9
Value
Time
LSTM
Out of Sample
175
150
125
100
75
50
25
0
0
20
40
60
80
100
120

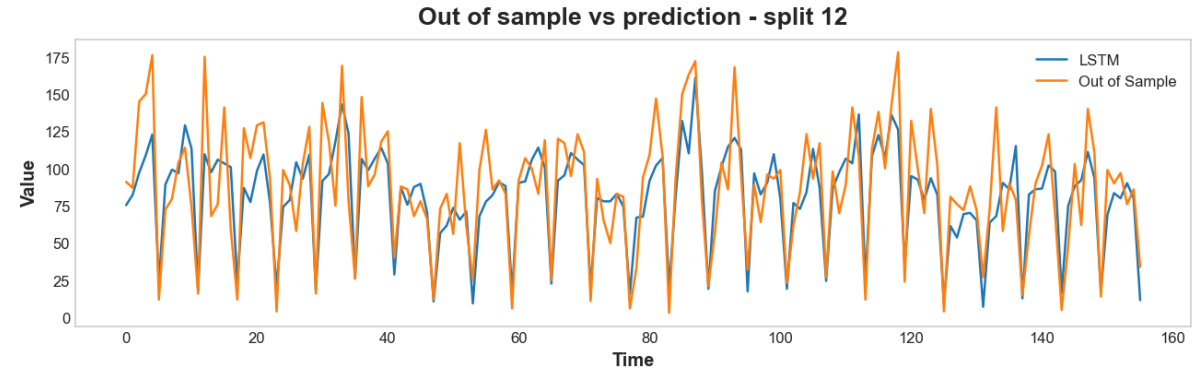

Training and out of sample gradient value - split 10
Value
Epochs
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0
25
50
75
100
125
150
175
200


Out of sample vs prediction - split 10
Value
Time
LSTM
Out of Sample
175
150
125
100
75
50
25
0
0
20
40
60
80
100
120


Training and out of sample gradient value - split 11
Value
Epochs
0.040
0.035
0.030
0.025
0.020
0.015
0
25
50
75
100
125
150
175
200


Out of sample vs prediction - split 11
LSTM
Out of Sample
Value
Time
175
150
125
100
75
50
25
0
0
20
40
60
80
100
120
140
160


Training and out of sample gradient value - split 12
Value
Epochs
0.035
0.030
0.025
0.020
0.015
0
25
50
75
100
125
150
175
200


Out of sample vs prediction - split 12
LSTM
Out of Sample
Value
Time
175
150
125
100
75
50
25
0
0
20
40
60
80
100
120
140
160