Time series Forecast of Call volume in Call Centre using Statistical and Machine Learning Methods

Nicolò Baldon
baldon@kth.se

A thesis presented for the degree of:
Master in Computer Science and ICT Innovation (TIVNM)

Supervisor: Erik Fransén
Examiner: Pawel Herman
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology in Stockholm
Host Company: Teleopti
August 2019

Abstract

Time series is a collection of points gathered at regular intervals. Time series analysis explores the time correlations and tries to model it according to trend and seasonality. One of the most relevant tasks, in time series analysis, is forecasting future values, which is considered fundamental in many real-world scenarios. Nowadays, many companies forecast using hand-written models or naive statistical models. Call centers are the front end of the organization, managing the relationship with the customers. A key challenge for call centers remains the call load forecast and the optimization of the schedule. Call load indicates the number of calls a call center receives.The call load forecast is mostly exploited to schedule the staff. They are interested in the short term forecast to handle the unforeseen and to optimize the staff schedule, and in the long term forecast to hire or assign staff to other tasks. Machine learning has been applied to several fields reporting excellent results, and recently, time series forecasting problems have gained a high-interest thanks to the new recurrent network, named Long-short Term Memory. This thesis has explored the capabilities of machine learning in modeling and forecasting call load time series, characterized by a strong seasonality, both at daily and hourly scale. We compare Seasonal Artificial Neural Network (ANN) and a Long-Short Term Memory (LSTM) models with Seasonal Autoregressive Integrated Moving Average (SARIMA) model, which is one of the most common statistical method utilized by call centers. The primary metric used to evaluate the results is the Normalized Mean Squared Error (NMSE), the secondary is the Symmetric Mean Absolute Percentage Error (SMAPE), utilized to calculate the accuracy of the models. We carried out our experiments on three different datasets provided by the Teleopti. Experimental results have proven SARIMA to be more accurate in forecasting at daily scale across the three datasets. It performs better than the Seasonal ANN and the LSTM with a limited amount of data points. At hourly scale, Seasonal ANN and LSTM outperform SARIMA, showing robustness across a forecasting horizon of 160 points. Finally, SARIMA has shown no correlation between the quality of the model and the number of data points, while both SANN and LSTM improves together with the number of samples.

Keywords: Time Series Analysis, SARIMA, Seasonal ANN, LSTM, Call Center data, seasonal time series

Summanfattning

Tidsserie är en samling punkter som samlas in med jämna mellanrum. Tidsseriens analys undersöker tidskorrelationerna och försöker modellera den enligt trend och säsongsbetonade. En av de mest relevanta uppgifterna, i tidsserieranalys, är att förutse framtida värden, som anses vara grundläggande i många verkliga scenarier. Numera förutspår många företag med handskrivna modeller eller naiva statistiska modeller. Callcenter är organisationens främre del och hanterar relationen med kunderna. En viktig utmaning för callcentra är fortfarande samtalslastprognosen och optimeringen av schemat. Samtalslast indikerar antalet samtal ett callcenter tar emot. Samtalslastprognosen utnyttjas mest för att schemalägga personalen. De är intresserade av den kortsiktiga prognosen för att hantera det oförutsedda och för att optimera personalplanen och på långsiktigt prognos för att anställa eller tilldela personal till andra uppgifter. Maskininlärning har använts på flera fält som rapporterar utmärkta resultat, och nyligen har prognosproblem i tidsserier fått ett stort intresse tack vare det nya återkommande nätverket, som heter Long-short Term Memory. Den här avhandlingen har undersökt kapaciteten för maskininlärning i modellering och prognoser samtalsbelastningstidsserier, kännetecknad av en stark säsongsbetonning, både på daglig och timskala. Vi jämför modeller med säsongsmässigt artificiellt neuralt nätverk (ANN) och ett LSTM-modell (Long-Short Term Memory) med Seasonal Autoregressive Integrated Moving Average (SARIMA) -modell, som är en av de vanligaste statistiska metoderna som används av callcenter. Den primära metriken som används för att utvärdera resultaten är det normaliserade medelkvadratfelet (NMSE), det sekundära är det symmetriska genomsnittet absolut procentuellt fel (SMAPE), som används för att beräkna modellernas noggrannhet. Vi genomförde våra experiment på tre olika datasätt från Teleopti. Experimentella resultat har visat att SARIMA är mer exakt när det gäller prognoser i daglig skala över de tre datasätten. Det presterar bättre än Seasonal ANN och LSTM med en begränsad mängd datapoäng. På timskala överträffar Seasonal ANN och LSTM SARIMA och visar robusthet över en prognoshorisont på 160 poäng. SARIMA har slutligen inte visat någon korrelation mellan modellens kvalitet och antalet datapunkter, medan både SANN och LSTM förbättras tillsammans med antalet sampel.

Dedication

Dedico questo lavoro a mio padre e a mia madre che mi hanno supportato durante tutto il mio percorso, nei momenti più difficili e che hanno sempre creduto in me durante tutti questi anni.

I dedicate this work to my father and my mother, who supported me throughout my journey, in the most difficult moments and who have always believed in me during all these years.

Acknowledgements

I am sincerely grateful to:

my Master's Thesis supervisor Erik Fransén, for the patience and wise advice given during the development of this thesis. I would also like to thank my examiner Pawel Herman for taking the time to help and correct my work. Finally, I also thank you for reading this material!

Nicolò Baldon

Contents

List of Figures

List of Tables

List of Abbreviations

sMAPE	Symmetric Mean Absolute Percentage error
MAPE	Mean Absolute Percentage error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
NMSE	Normalised Mean Squared Error
ANN	artifial neural network
SANN	Seasonal Artificial Neural Network
MLP	Multilayer Perceptron
NN	Neural Network
TDNN	Time Delay Neural Network
FFNN	Feed Forward Neural Network
NARX	Non linear Autoregressive with
AIC	Aikaike information criterion
ACF	Auto-correlation Graph
PACF	Partial Autocorelation Graph
KST	Kolmogorov-Smirnov Test
SARIMA	Sesonal Autoregressive Integrated Moving Average
ARIMA	Autoregressive Integrated Moving Average
MIMO	Multiple-input-Multiple-output
LSTM	Long-short Term Memory
RNN	Recurrent Neural Network

Chapter 1

Introduction

This thesis explores the area of call load forecasting through machine learning techniques. An accurate forecast of the incoming call volume helps companies to schedule their agents better. Although, we are living in the digitalization era, call volume is still forecast using hand-written models or naive statistical models [1]. Since several years, machine learning techniques have reported better results than statistical methods as confirmed by world forecasting competitions as M3 and M4 and so on. However, machine learning is a data-driven approach and there is no certainty whether the model will work or not, and even less about the quality of the model. Several examples show how an LSTM is the good choice to forecast the price of the bitcoins but results in poor performance when it comes to forecast rainfalls. Therefore, it is hard to guess, which ML method will work before conducting a systematic experiment [2] [3] [4] [5].

This study aims at highlighting some dominant behaviors of two famous machine learning models and compare their capabilities with the method proposed by Box–Jenkins, also known as ARIMA model. ARIMA was chosen as a reference since it is still widely used in many real applications [6] [7] [8]. We want to ascertain with what accuracy it is possible to forecast call load, both at daily and hourly scale.

1.1 Motivation

Workload forecast is crucial in many different systems as healthcare, industrial, traffic forecast, energy service, retail store, and so on. Call centers are the front end of the organization since they manage the relationship with the customers, playing a fundamental role in the survival of the company itself [9]. Their final objective is to provide an optimal experience to the customer, whose satisfaction is determined through different metrics [9]. Some of the most common parameters are the number of calls, where customers hang up before being answered (abandonment ratio), the average waiting time before an agent manages to take care of the call (average speed of answer), the numbers of calls answered before a certain amount of time (service level) [10].

Modern call centers can employ thousands of agents. Each agent is trained with different skills; hence, they can accomplish just a limited set of specific tasks [11]. As it can be imagined, scheduling enough agents to cover all the tasks and to allow feasible working hours might be a hard problem to face manually. For this reason, many software tools are thought to facilitate staff management and to optimize the working schedule. Those are mainly based on two variables, the average time per call and the number of incoming calls, which is also considered as the primary variable.

Scheduling is further subdivided into long, medium and short-term. The long-term schedule aims at deciding the number of people to hire/fire, and it is usually done on average between three months and six months ahead. The medium-term scheduling assigns working time slot to the employees, and it is generally carried out a couple of weeks in advance. Last, the short-term forecasting is thought for handling the unforeseen events that may happen within a couple of days before the actual date.

Nowadays, many companies do not have the necessary competencies to implement complex forecasting models. Hence, either they forecast based on hand-made rules or using a naive statistical method or a combination of both. However, in most of cases, they find it more convenient to outsource their work to external companies named workforce management companies (WFM), like Teleopti [1].

To handle the huge variety of customers WFMs often exploit a so-called "out of bag" method. Multiple techniques are tested on the customer data set, evaluated according to a predetermined metric, and the model with the best score is picked up. Nowadays, most of those techniques are based on statistical methods or hand-written rules and they are not always capable of providing an accurate forecast, they are not able to learning effectively from the past and they provide a static forecast. Besides, scale-dependent and unreliable metrics are used to evaluate models, leading to picking up the wrong model.

As stated in [1], further research has to be carried out on the development of new models to forecast call volumes, since nowadays, statistical methods are mostly used to staffing and scheduling [12]. Moreover, there is a need to fill the gap between the industry and the research field implementing new solutions proposed in literature on real-world problems, in order to innovate the industry.

1.2 Problem statement and research questions

This project aims at investigating the capabilities of neural networks in call volume forecasting, depending on the size of the available data.

As our reference, we have chosen SARIMA, since it is widely used and deemed to be one of the best statistical models [13].

In recent research, Multilayer Perceptron architecture has resulted in good performances on time series problems, thanks to its generalization capability. For this reason we chose it as our first architecture to start with our investigation. Moreover, it is a widely studied architecture and well documented in the literature [8] [14]. In recent years, Recurrent Neural Network, in particular, Long-Short Term Memory architecture, reported excellent results so we decided to use it as a state-of-art architecture [15] [16].

The research questions are:

1. Which of the methods selected for comparison, i.e. Seasonal ANN and LSTM, deliver the most accurate performance with respect to SARIMA ?

2. How do Seasonal ANN and LSTM perform at different forecast horizons?

3. What is the relationship between the size of the training set and the quality of the models? We want to assess how much the training size influence the prediction accuracy of different models.

1.3 Scope and limitation

This thesis explores the area of call volume forecasting combined with machine learning techniques. The scope of this thesis is to analyze and test machine learning capabilities comparing them to the famous and widely used SARIMA model, as baseline approach.

The scope of this project includes:

- assessment of machine learning capabilities for call volume forecasting

- comparative analysis of different machine learning or statistical models

At the same time, this project is subject to some limitations, listed below:

1. limited training size of the data

2. limited computational power available. All these simulation are carried out on a laptop, intel(R) Core(TM) i5-3437U @ 1.90 GHz.

Chapter 2

Background

Time series analysis and forecasting are interdisciplinary fields, where computer scientist, statisticians, and economists are involved.

Time series is a particular typology of data, where the time component plays a fundamental role, being a constraint but also an additional source of information.

This chapter presents an overview of the main concepts exploited for the thesis, starting from what is a time series and how it is analyzed and forecast, to machine learning techniques concluding with a brief description of famous neural network architectures [13].

2.1 Time series

A time series can be seen as a collection of samples, taken at equally spaced time intervals, which describes specific phenomena. *Univariate* time series refers to a collection of a samples of the same variable over time. Depending on the sampling rate (hourly, daily, weekly), different patterns may emerge [13].

2.1.1 Time series analysis

A time series can be decomposed in four main patterns:

- a sloping curved, called *trend*
- a repetitive pattern over the time called *seasonality*
- an up and down fluctuation on the data which are not a real pattern, named as *cycles.*
- a random component identified as *noise*

Trend, if it is present, can be upward or downward and it models a behavior perpetuated during several periods. The seasonality is a repetitive pattern that occurs on the time series with the same frequency. There may be different seasonality according to the time-frequency. For example, it is possible to observe just a daily seasonality as well as a daily and weekly seasonality. Seasonality may be classify as *additive* or *multiplicative* A cycle is a repetitive-wavelike pattern over a period longer than a year. The last component of a time series is a random error, and it should look like white noise. To better describes a time series, the relation with external factors can be taken into account.

The purpose of time series analysis is to find a model that fits the data; in other words, it seeks the "why" behind a time series [17][13].

2.1.2 Time series forecasting

As stated by, Shmueli, Galit and Lichtendahl Jr, Kenneth C, "time series forecasting uses the information in a time series (perhaps with additional information) to forecast future values of that series" [18]. To keep in mind is that the future is completely inaccessible and it must be inferred solely from what it has already happened.

There are some major concerns about forecasting:

- How much data is available?
- How many points in the future do you want to forecast?
- Is the forecast static or can be updated frequently?
- At which sampling frequency is the forecast required?

Each of these constraints influences the capability and the choice of the technique through which you want to forecast.

In this work, we face all these constraints. We have chosen to provide a static forecast, with a limited amount of samples available, exploring till 160 points into the future. Hourly and daily scale are evaluated [13].

2.2 Error Metrics

Forecasts are difficult to evaluate with the usage of only a single metric. In the following section, we explain all the metrics explored in this work. For the sake of notation in this section:

- N indicates the total number of samples
- m indicates the seasonality
- A are the actual values whereas
- F as the predicted values

2.2.1 Normalized Mean Square Error

The normalized mean square error (NMSE) provides an estimation of the overall deviations between the prediction and the actual values. In other words, it is a normalized version of the well-known mean squared error divided by the test variance. It is computed as shown in (2.1)

$$NMSE = \frac{1}{N}\sum_{j=1}^{N}\frac{(A_j - F_j)^2}{\overline{AF}} \tag{2.1}$$

$$\overline{A} = \frac{1}{N}\sum_{j=1}^{N} A_j \tag{2.2}$$

$$\overline{F} = \frac{1}{N}\sum_{j=1}^{N} F_j \tag{2.3}$$

where, $\overline{F}$ is the mean of the predicted values and $\overline{A}$ is the mean values of the actual values. It is a balanced error measure and it is very effective for forecast evaluation. It differs from the other metrics since the deviations are summed instead of being differentiated. A low NMSE implies a well-performing model, both in time and in space. Nevertheless, if the model has a high value it does not necessarily mean that the model is unfortunate. It might be due to time or space shifting.

2.2.2 Root Mean Square Error

The root mean square error(RMSE) is commonly used to compute and evaluate the differences between the forecast values and the actual values. The formula is shown in (2.4). It is the standard deviation of the residuals. The square root is used to attenuate large errors.

$$RMSE = \sqrt{\frac{\sum_{j=1}^{N}(A_i - F_i)^2}{N}} \tag{2.4}$$

We point out that this is a scale-dependent metric used, meaning that the scale of the samples influences the RMSE values. It is not useful form an accuracy point of view, but it is very relevant from a statistical point of view.

2.2.3 Mean Absolute Error

The mean absolute error (MAE) it is simply the mean of the absolute values of the difference between two variables. It is given by (2.5).

$$MAE = \frac{\sum_{j=1}^{N}|F_j - A_j|}{N} \tag{2.5}$$

MAE is the average vertical distance with respect to the identity line. This metric is also scale-dependent. MAE is also a scale-dependant metric.

2.2.4 Mean Absolute Percentage Error

The mean absolute percentage error (MAPE) is a measure of accuracy and it is compute as shown in (2.6).

$$MAPE = \frac{100\%}{N} \sum_{j=1}^{N} \left| \frac{F_j - A_j}{A_j} \right| \tag{2.6}$$

MAPE is a valuable metric, but it has some drawbacks, as well. Singularities occur in case there are some zeros in the actual values. Besides, this metric does not have an upper limit when the forecast is too high. It is also unbalanced due to the greater penalty it put on the negative errors, so when $A_j < F_j$, then on positive errors. To solve all these issues other metrics have been developed.

2.2.5 Symmetric Mean Absolute Percentage Error

The Symmetric Mean Absolute Percentage Error(sMAPE) is a scale-independent accuracy measure and it is given by the (2.7).

$$SMAPE = \frac{100\%}{N} \sum_{j=1}^{N} \frac{|F_j - A_j|}{(|A_j| + |F_j|)/2} \tag{2.7}$$

It is the absolute difference between the forecast and the actual values divided by the sum of the absolute values of the actual and the forecast samples. It offers some interesting properties. It has lower and an upper bound, which are respectively 0% and 100%. The metric is divided by two since the upper bound would be of 200% otherwise.

2.2.6 Mean Absolute Scaled Error

The Mean Absolute Scaled Error (MASE) is a metric used to overcome some of the main issues of MAPE. It is used to evaluate forecasts values and it is data scale independent. It is computed as shown in (2.8). MASE is computed in two different ways depending on the presence of seasonality in the time series. Since all the datasets have a strong seasonality component, only the seasonality MASE is discussed below. MASE can be seen as MAE divided by the mean of the naive forecast method, which takes the values of the previous sample as the value of the next forecast.

$$MASE = \frac{\sum_{j=1}^{N} |A_j - F_j|}{\frac{N}{N-m} \sum_{j=m+1}^{N} |A_t - A_{t-m}|} \tag{2.8}$$

This metric offers some interesting properties. First of all, it is scale invariant. It shows a predictable behavior when the $F_j \longrightarrow 0$, solving the issue when the values tend to zero or when zeros appear frequently. Moreover, it is also symmetric, meaning that it penalized the positive and the negative error equally.

2.3 Statistical models

A statistical method is a mathematical model, which tries to express the relationship between two or more random variables analytically. AutoRegressive model (AR) was applied for the first time back in the 20s. Since then, many models have been proposed. We report only the most successful ones.

In 1956, Robert Goodell Brown proposed, for the first time, the exponential smoothing method, also known as Simple Exponential Smoothing (SES). SES has inspired multiple researchers, and many other models derived from SES. To be mention is the famous Holt-Winter method, proposed by Holt and Winter some years later. This class of techniques model the error, the trend, and the seasonality [19], [20]. As stated in [21]: "Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight". In the forecast community, these methods are referred to as ETS models, meaning the explicit modeling of Error, Trend and Seasonality.

The second big class of methods is born as a combination of the autoregressive and moving average methods. Autoregressive Integrated Moving Average model (ARIMA), comprises both the autoregressive part and the moving average part. The integration part allows to model time series with a trend without removing it. The ARIMA model can be estimated following the approach suggested by Box-Jenkins. A seasonal version of the ARIMA model is called Seasonal ARIMA (SARIMA), which allows to model time series with both trend and seasonalities. SARIMA includes four more parameters, increasing the complexity of the model. The first one indicates the seasonal lag, that can be inferred by looking at the ACF graph. The remaining parameters model the seasonality [22].

In the 80s, a non-linear model was proposed to model time series with non-linear patterns, known as AutoRegressive Conditional Heteroscedasticity (ARCH). This method had the purpose of modeling the change in variance over time in a time series [23].

Linear statistical methods are still, today, a valid choice to forecast data since they are simple to implement and the math behind it has deeply understood. They become very useful when the amount of data available is limited in size.

2.4 ANN Models

Artificial Neural Networks (ANN) are a type of architecture that vaguely attempts to replicate biological neural networks.

Although the idea to use neuron-like elements to solve basic mathematics dates back to the 40s, neural networks started to gain interest only in the 80s. With the introduction of the back-propagation algorithm in the early 70s, which allows the network to converge faster, combined with the increasing computational power, neural networks began to show exciting results [14].

In the last two decades, machine learning has found many applications in several fields, such as image processing, natural language processing, outperforming the current state of the art methods and in time series forecasting. To a full review of all the possible model for time series, we refer to [7].

ANNs are constituted of multiple layers, where numerous nodes compose each layer. Generally, each node is connected to all the other nodes of the successive layer. This structure is named as *fully connected*. It has to be said that, in some cases, a node might be connected to some nodes in the previous layer, and in general the architecture's structure can be modified according to the specific task. ANN approach tries to minimize a *loss function*, which is used to evaluate the quality of a candidate solution. The computed value is commonly referred to as *loss*. Lower the loss better is the model. In general, mean squared error or mean absolute error are a typical example of loss functions.

The so-called *activation function* can activate each node, that determines whether to turn on or off the node. Some examples of activation function are the *rectified function*, (ReLu) and the *sigmoid*. The most straightforward neural network always comprises the input and the output layers. This architecture may be interpreted as a weighted linear combination of all the inputs since weight is assigned to each connection. This simple architecture is well-known in literature and it is modeled by the formula given in Eq 2.9, where W is the weight matrix and x is the input vector.

$$y = Wx \tag{2.9}$$

However, this simple architecture can handle only linear problems. When a *hidden layer* is added between the input the and the output layers, the network becomes capable of modeling almost all possible linear and non-linear relationships between an input and an output under the mild condition on the activation function. It is formally stated by the universal approximation theorem [24]. Multiple architectures can be exploited and design, and the choice depends on the problem that has to be solved. In general, a Neural Network(NN) can be classifies as *static* or *dynamic*. A static NN does not contain either feedback or delays in along all the layers. A *dynamic* NN may comprise either feedback or delays or even both of them in the architecture. Thanks to the delay and feedback connection NN are empowered with a sort of *memory* which makes them suitable to learn time-varying phenomena.

For time series forecasting, there are two classes of neural network widely used nowadays. The first class is the FeedForward Neural Network, while the second one is the Recurrent Neural Network [25] [7].

2.4.1 Feedforward Neural Network

Feedforward Neural Networks (FFNN) are the oldest class of NN, proposed back in the 50s. They are the most spread network's typology, both for their fast convergence and their simple implementation. In general, they are used for supervised machine learning tasks.

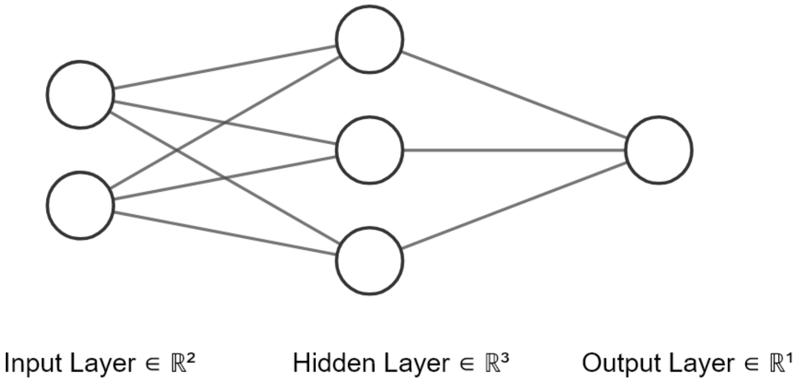


Figure 2.1: Schematic representation of a simple fully connected feedforward neural network with three layers.

The main goal is to approximate a function that maps the input into the output in the best possible way. It defines a rule $y = f(x, \theta)$, where θ are the collection of parameters the network has to learn.

They are called feedforward since the information flows from the first layer to the next ones. If a feedback connection is added from the output layer to the input layer, we obtain a recurrent neural network.

The network can be seen as a composition of different functions. An example of a simple FFNN is shown in Figure 2.1. The architecture can be scaled, adding any number of layers with any number of nodes. An optimization algorithm is used to train the network and update the weights after each epoch until the cost function has reached a minimum, which might be the global or only a local minimum.

The most common architecture used in time series forecasting (TSF) problems and belonging to the FFNN class is the MultiLayer Perceptron (MLP).

MLP comprises at least three layers, and it is normally trained with the backpropagation algorithm [26].

2.4.2 Recurrent Neural Network

Recurrent neural networks are the second class of ANN that has mostly succeeded in TSF problems, given the number of applications, where they have been used [16]. They are thought to embed the concept of 'memory' in the system. In the feedforward neural networks, we assume that all inputs (and outputs) are independent of each other. However, in some cases, it might be a strong assumption, leading to wrong results. Therefore, to consider the correlation between the inputs and the outputs, RNNs implement a way to capture the information computed so far. Each unit takes as input the state computed the previous iteration. The country contains the historical knowledge of what happened till now; to some extent, you can see it as an embryonic type of memory [27]. If we unroll the structure, we observe copies of the RNN cell, sharing the same weights, are made over time with different inputs at different time steps. Figure 2.2 represent a RNN unit and its unrolled version. As you can observe, what changes are the input and the output and the state of the cells, while the weight matrices remain unchanged.

In theory, RNN can exploit information in arbitrarily long sequences. The main issue of RNNs is that the cost functions is computed as a product of real numbers that can shrink to zero or explode to infinity. In literature, it refers to this problem as the exploding- vanishing gradient [28].
To overcome this problem, Long-short Term Memory network was proposed. It implements a *constant error carousel* avoiding the error decay to fast. The main idea, instead of computing a new state as a matrix product with the old state, it preferably calculates the difference between them. Expressivity is the same, but gradients are better behaved [29] [15].

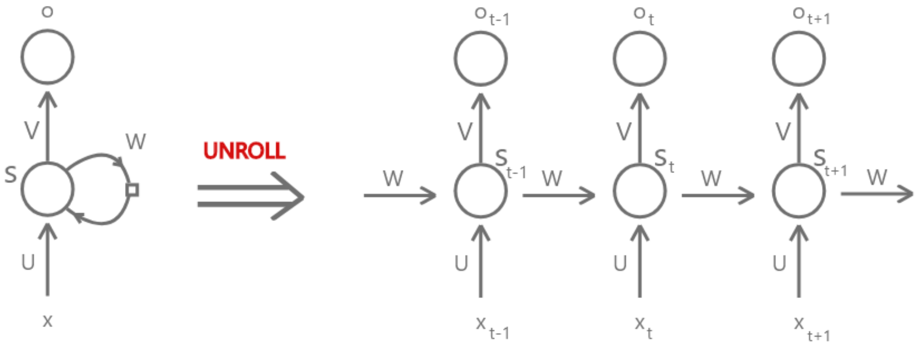


Figure 2.2: On the left the closed version of a RNN. On the right, the unrolled illustration of the same recurrent neural network. x is the input, S is the inner state of the network, V, W, and U are the weight matrices and o is the output

2.5 Related Work

In recent years call centers have sharply increased due to the explosion of the service industry. Between 60 and 80 percent of the total costs of a call center derive from staff management. Therefore, an accurate forecast can allow the company to organize its agents better. [30] [31]

However, the call volume forecast comes with some issues. Call center data are usually high dimensional. Not only the incoming calls are provided but also the duration of the call, the call type, and so on. Besides, data is sampled at high frequency, and in some cases, it is provided at 15 minutes interval. Many samples are then available, which may cause the statistical methods to break down. [32]

Furthermore, they often present intricate seasonal pattern. Due to the high sampling rate, it is possible to observe seasonality at hourly, daily, weekly, and even yearly scale. All those seasonalities have to be modeled.[33]

Call center data is also pretty sensitive to the context. The presence of outliers may due to external events, like promotion, advertising campaigns, holidays, and so on. Many techniques have been explored to handle those. [34]

In this thesis, we do not handle this problem and assume that this information is already available.

Several models are nowadays available to model seasonal data. A list of them comprises Seasonal AutoRegressive Integrated Moving Average (SARIMA) models [22], Holt-Winters (HM) models [20], Periodic AutoRegressive Moving Average (PARMA) models [35], etc. Recently, also Super Vector Machine has been applied to a time series forecast problem with a discrete success. [36] Most of them rely on deseasonalization of differencing. However, this approach has been discouraged by some researches for multiple reasons. First, it seems to be biased in the presence of many trends, and second, it leads to complicated non-linear properties, increasing the complexity of the problem itself.[37]

Therefore, researches aim at finding methods able to model the seasonality directly, without any filtering.

ANNs have shown to be extremely efficient in many different fields, gaining a lot of interest thanks to their capability and flexibility in modeling non-linear patterns. They are particularly appreciated as they are non-parametric, data-driven, and self-adaptive. [38] [39]

Regarding time series forecasting, multiple studies have been conducted highlighting different capabilities of ANNs. Some researches claim that neural networks are not able to adequately track the seasonality; others have shown the contrary. [2] [3] Notably, few studies tried to model call volumes data directly using machine learning algorithms.

Recently, an innovative structure, called Seasonal ANN (SANN) has shown promising results in forecasting seasonal time series. [40]. The main advantages are the constraints imposed on the input and the output layer, which simplifies both the modeling and the training sessions.

In recent years, also RNNs have gained popularity among the forecasting community, finding a useful application in many real-world scenarios. Among all RNNs structures, we choose LSTM, that seems to be the most appreciated. A novel approach for traffic forecasting based on LSTM is described in [15]. LSTM has been applied almost to all kind of task due to its impressive performances. [41]

Moreover, multiple forecasting techniques for SANN and LSTM are investigated. More details are provided in Section 3.5.

In this project, we proposed a comparison between statistical and neural networks models, comparing LSTM, SANN and SARIMA methods.

Chapter 3

Methods

In the following Chapter are described all the techniques exploited in this project. Section 3.2 illustrates the seasonal auto-regressive integrated moving average model, Section 3.3 explains the ANN technique, and Section 3.4 the LSTM architecture. Section 3.5 describes all the forecasting techniques probed for machine learning algorithms. In Section 3.9 are reported all the techniques exploited to set the hyperparameters utilized for the systematic experiment, which are described in Section 3.10.

3.1 Data Description

In this Section follows a general description of the dataset used for the experiments. The time series shown in Figures 3.1 and 3.2 come from three different call centers. The variable, shown in the figures, represents the incoming calls at a different time scale. Initially, all the time series were provided only at 15 minutes interval levels, but it has been aggregated on different time scales. In particular, we chose the daily and hourly scales, since they are the most useful for the customer. We summed all the intervals values for each day, obtaining a new time series, where each point represents the number of calls within one day. We repeat it for the hourly scale. At daily scale, 96 points at interval scale were averaged out to obtain one point. At hourly scale, one point is the average of 4 points at interval scale.

Incoming calls are considered a specific type of time series.They are characterized by multiple seasonalities, and in general, they deal with high frequency (hourly and daily) data. Moreover, they are subject to "special day" events, like holidays or world cup and equipment failures, such as a power cut or only a failure on collecting the number of incoming calls.

Figure 3.1a shows a daily time series belonging to the first customer, characterized by a sharp downward trend and a visible daily, weekly, and yearly seasonality. This time series has chosen to simulated an old customer. Therefore, a large amount of data is available. Additionally, to the trend and the seasonality, some other peaks are observed. Those may be outliers or a mistake during the collection of the data. However, given the amount of data, we assume that those outliers do not damage the model, hence we decided to ignore them.

Figure 3.1b shows the incoming calls belonging to the second customer, which has only a limited amount of data available. We can observe a weekly pattern and the presence of a repetitive peak, highlighting the presence of seasonality. Nothing can be said about the yearly seasonality, as data available is not enough. Moreover, there is no sign of an upward or downward trend. To be noted is the presence of multiple zeros in the time series. It is relevant for the choice of the right metric. Besides, we can observe the presence of numerous peaks and sudden fluctuations that singularize the time series. It is a typical behavior for call volumes time series since they are very sensitive to external factors. This time series simulates the scenario of a new customer.

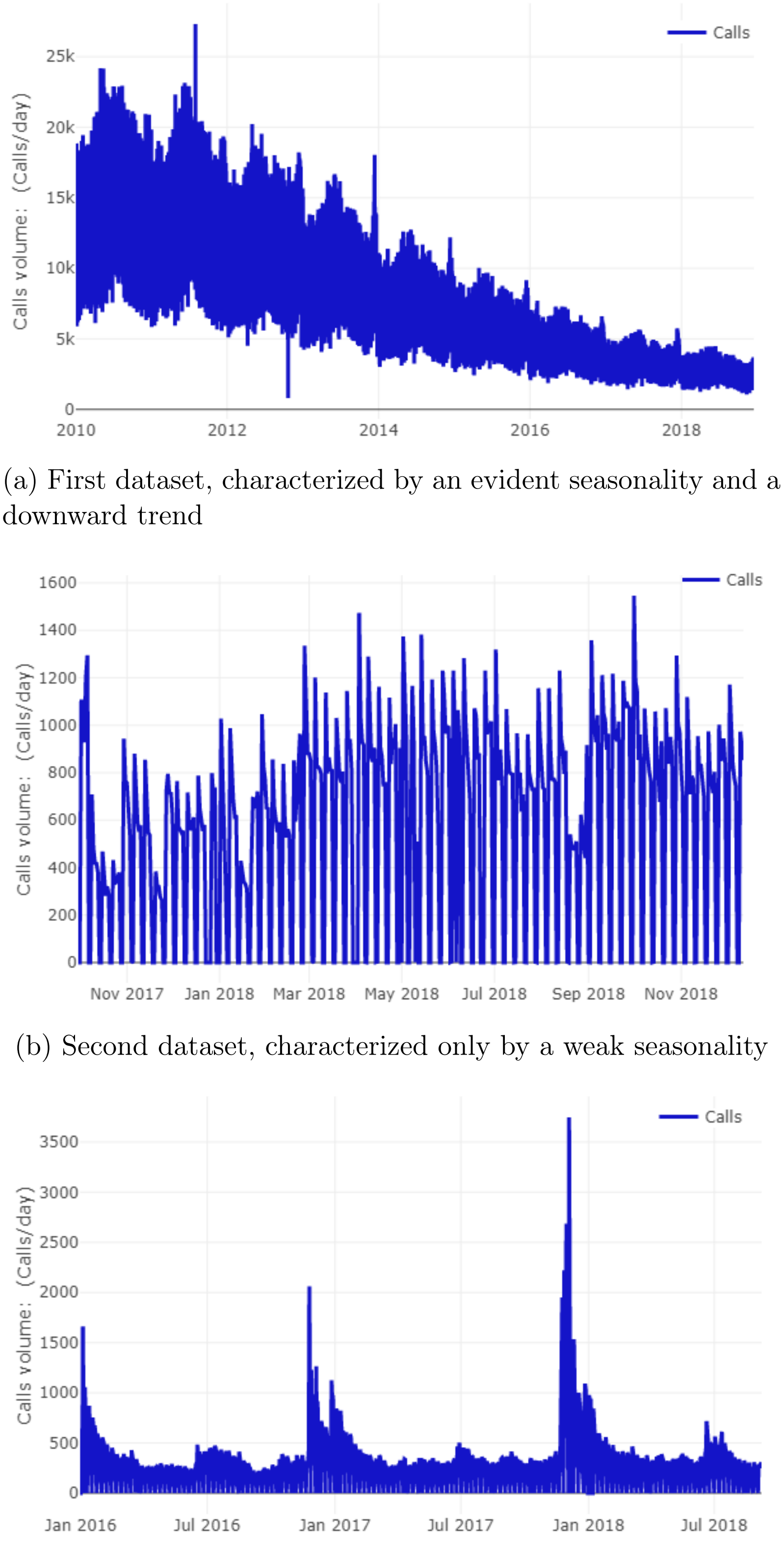
Figure 3.1c shows the daily volume of the third customer. As for the previous two scenarios, we may observe the presence of a strong seasonality both at daily, weekly, and yearly level. There is no clear evidence of a trend, but the peaks at the beginning of each year are increasing year by year.

In none of these time series, there are missing values; hence, we did not tackle the problem. We assume data to be consistent. In case there were some missing values, they should be handled separately, before applying any of the algorithm proposed in this project.

Secondly, also hourly scale is taken into account. For the sake of intelligibility, Figure 3.2 shows a window of all the series at an hourly scale. In each of them, we can observe the presence of repetitive behaviour, which indicates the presence of seasonality. The seasonality evaluation is deepened in Section 4.1, through the evaluation of ACF and PACF graphs. For the sake of completeness, a summary of the sizes for each dataset is reported in Table 3.1.

Name	**Daily points**	**Hourly points**	**Missing values**
dataset 1	3272	78525	None
dataset 2	439	10511	None
dataset 3	982	23545	None

Table 3.1: Number of samples available at daily scale, hourly scale and missing values

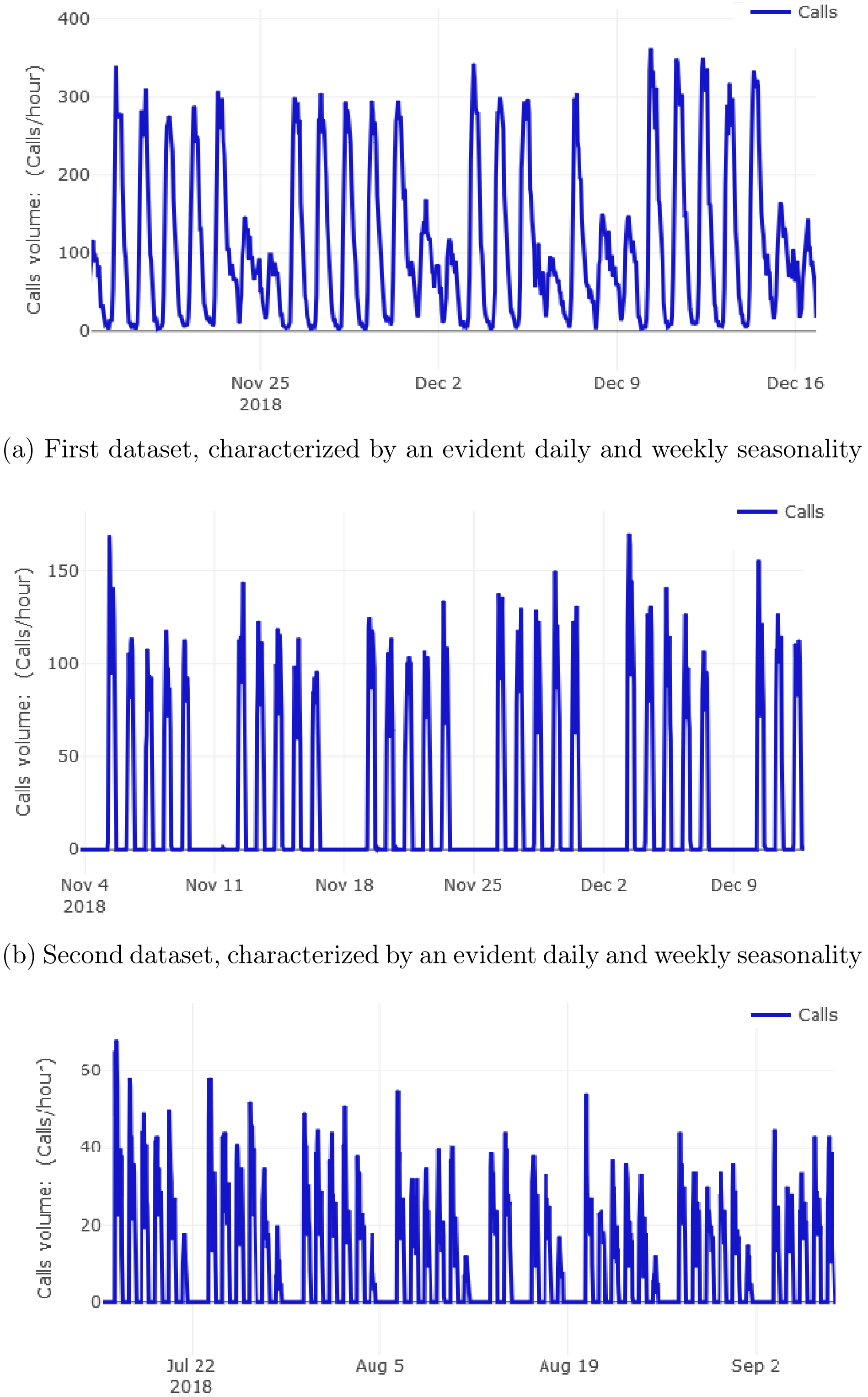


(a) First dataset, characterized by an evident seasonality and a downward trend

(b) Second dataset, characterized only by a weak seasonality

(c) Third dataset, characterized by cycles and a weekly seasonality.

Figure 3.1: A general overview of the three datasets, showing the incoming calls at daily scale



(a) First dataset, characterized by an evident daily and weekly seasonality

(b) Second dataset, characterized by an evident daily and weekly seasonality

(c) Third dataset, characterized by an evident weekly seasonality

Figure 3.2: An overview of some weeks of the three datasets at hourly scale

3.2 Seasonal ARIMA Model

In this Section, a general description of the Seasonal auto-regressive integrated moving average model (SARIMA) is provided, which goes through all the different parts comprised into the model. SARIMA is the generalization of the simpler AR model and it is commonly used to forecast uni-variate time series with a seasonal component. It comprises the auto-regressive component (AR), the differencing part (I), and the moving average component (MA).

3.2.1 Auto regressive Model (AR)

A stochastic process is defined as an auto-regressive process if it can be modeled trough 3.1:

$$X_t = \sum_{j=1}^{p} \phi_j X_{t-j} + \omega_t \tag{3.1}$$

where $\omega_t \sim N(0, \sigma^2)$. Therefore, the sample at the time t is assumed to be a weighted-linear combination of the previous values plus a noise, where the noise approximates a normal distribution. The number of past values is usually reported with the letter p, which indicates the lag of the auto-regressive problem.

If a time series is not stationary, it may lead to spurious regression, meaning that the regression equation may show significant relationship where there is not.

3.2.2 Moving Average (MA)

The moving average part of a time series is described by (3.2)

$$X_t = \omega_t + \sum_{j=1}^{q} \theta_j \omega_{t-j} \tag{3.2}$$

By definition, the observed value is a random error term plus a weighted-linear combination of previous random error terms up to a defined maximum lag, which is commonly denoted with the letter q

3.2.3 Mixed model ARMA

The combination of the previous two models define a general auto-regressive moving average process of order p and q and it can be formulated by:

$$X_t = \omega_t + \sum_{j=1}^{q} \theta_j \omega_{t-j} + \sum_{j=1}^{p} \phi_j X_{t-j} \tag{3.3}$$

A time series of order p and q is described by:

$$x_t = \beta_0 + \beta_1 X_{t-1} + \beta_2 X_{t-2} + ... + \beta_p X_{t-p} + \phi_1 \omega_{t-1} + ... + \phi_q \omega_{t-q} \tag{3.4}$$

3.2.4 SARIMA Model

The ARIMA model is described by three parameters (p, d, q), where:

- p is the trend order of the autoregressive part
- d is the trend order of the differencing part
- q is the trend order of the moving average part.

SARIMA is an extension of the ARIMA model which can handle time series with a seasonal component. The model adds four new parameters, which are:

- P is the seasonal order of the autoregressive part
- D is the seasonal order of the differencing part
- Q is the seasonal order of the moving average part.
- m the number of the time steps for a single seasonal period

The first three (P, D, Q) are identical to the ARIMA, whereas the fourth one indicates the lag of the seasonality. The model is often mentioned as $SARIMA(p, d, q)(P, D, Q)_7$.

As said before the SARIMA model can be applied just to stationary processes. The stationary of a time series is discussed in 3.9.2. The Augmented Dickey-Fuller is applied to test whether a time series is stationary, which is reported in 3.9.4.

To infer the p, P and q, Q values the ACF and the PACF graphs are explored, which are explained in 3.9.5

Once all the initial parameters are chosen, we apply a grid search to explore a meaningful part of the space of solutions. The best model is assessed through an information criterion explained in 3.9.3

Finally, to check the goodness of the model, the residuals are plotted and evaluated through the QQ plot, the ACF plot, and the normality test plot. When most of the sample autocorrelation coefficients lies within the limits $(-1.96 * \sqrt{N}, 1.96 * \sqrt{N})$ and they get close to the normal distribution, the model has fitted the data and the rest left is something close to white noise [22], which cannot be predicted with a statistical model.

SARIMA forecast

As explained in Section 3.2, SARIMA is a linear-parametric forecasting technique. Hence, it establishes a fixed rule based on linear combination of the previous samples. For the sake of completeness a simple example, for an SARIMA $(1, 1, 1)(1, 1, 1)_7$ is reported 3.6 utilizing the back-shift notation reported in 3.5 :

$$B^n y_t = y_{t-n} \tag{3.5}$$

$$(1-\phi_1 B)(1-\Phi_1 B^7)(1-B)(1-B^7)y_t = (1-\theta_1 B)(1-\Theta_1 B^7)e_t \tag{3.6}$$

where, $(1-B)$ denotes the first difference, $(1-B^n)$ denotes the seasonal difference, $(1-\phi_1 B)$, $(1-\Phi_1 B^7)$, $(1-\theta_1 B)(1-\Theta_1 B^7)$ are the $AR(1)$, $seasonalAR(1)$, $MA(1)$, $seasonalMA(1)$ respectively.

3.3 Seasonal ANN (SANN)

Multilayer Perceptron is one of the most popular and commonly used Artificial Neural Network (ANN). The simplest architecture is composed of only two layers, which are called the *input layer* and the *output layer*, respectively. This architecture, indeed, can be seen as a weighted sum of the input. Therefore, to increase the capability, some *hidden layers* can be added in between the input and the output layer.

Each layer comprises multiple nodes. Typically, each node of one layer is connected to all the nodes of the next layer, and each connection is labeled with a weight. Every node has an *activation function*, where all the weighted input of the previous layer are summed. There are multiple types of activation function, such as sigmoid, ReLu, linear, and so on.

Time delay neural network (TDNN) is a dynamic ANN and can be subdivided into *focused*TDNN and *distributed*TDNN. For this thesis, only *focused*TDNN is discussed. *Focused*TDNN have a tapped-input line, which limits the dynamic only at the input layer, as shown in Figure 3.3. They can be trained to learn sequential or time-varying patterns. For these reasons, they have found multiple applications on control system, fault detection and especially on speech recognition and prediction on financial markets.

A focusedTDNN can be seen as a *multilayer perceptron* with a tapped line at the input layer. If we assume two inputs of size k the input layer can be described as in (3.7)

$$y_1(t), y_2(t), y_1(t-1), y_2(t-1), ..., y_1(t-k), y_2(t-k) \tag{3.7}$$

,where the input is a multiple between the number of variables and the time delays of the tapped-line. TDNNs are powerful for multi-series forecasting. In case of a univariate time series, the focused TDNN becomes equal to an MLP.

Multiple solutions have been provided for univariate time series, and, recently, C. Hamzacebi proposed a model called Seasonal ANN, where the input and the output size are fixed and set equal to the seasonality of the time series, whereas the size of the hidden layer varies with the seasonality as upper-limit. The model is indicated as (s, h, s), where s is the size of the input and output layer and h is the size of the hidden layer. Moreover, h is bounded between $[1, S]$, since in the worst case, there would be s different patterns to learn. [40]

An example of the Seasonal ANN is shown in the Figure 3.3

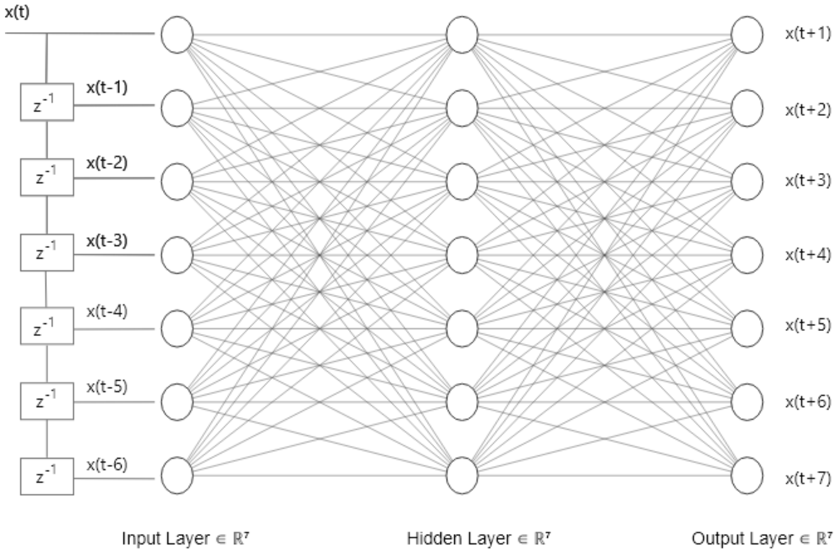


Figure 3.3: Example of an (7,7,7) Seasonal ANN architecture. The input is built through a tapped line, where the input is stored till the 7^{th} lag. The output represents the forecast of the next seven points in a consecutive order

3.4 Long-Short Term Memory (LSTM)

Recurrent neural networks (RNN) try to overcome the incapability of feed-forward neural networks to maintain information across different batches. RNNs contain feedback connections, which allow information to persist during multiple iterations.

LSTM is a special kind of RNN and it was proposed by Hochreiter back in the 90s [29]. Hochreiter designed this architecture to solve the vanishing gradient issue, which was a curse for normal recurrent neural network. [42], [43].

In recent years, it has gained a lot of attention thanks to its capability and flexibility in modeling long term dependencies. Normally, it has one input layer, one recurrent layer, one output layer.

The main idea behind LSTM relies on the *Cell State* in the recurrent layer. It carries all the information with only some linear interaction. Information is also regulated by the so-called *gates* structures, which are sigmoid functions, whose output decides how much information should go through in the next iteration. Each gate returns a value between zero and one, where zero means that "no information" pass, and one "leave the information unchanged".

LSTM is ruled by six equations. The **forget gate layer** decides what information to keep from the previous cell state.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.8}$$

Then the following two equations rule what information has to be stored. The **input layer** decides the values to update, while a **tanh layer** creates a bunch of possible new values.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.9}$$

$$\tilde{C_t} = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3.10}$$

The following equation embeds the information in the cell state. It combines the results obtained in 3.9 and 3.10.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C_t} \tag{3.11}$$

The final step is to decide what to output and the last two equations rule it.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.12}$$

$$h_t = o_t * tanh(C_t) \tag{3.13}$$

The memory cell is primarily a recurrently self-connected linear unit, called Constant Error Carousel (CEC) and the cell state is represented by the activation of the CEC.

3.5 Forecasting Techniques for Neural Networks

In contrast with the traditional linear forecasting systems as SARIMA, ANNs can forecast in different ways and each of them has some advantages and disadvantages. Below are reported the four main strategies. For the sake of notation, f and F denote the functional dependencies between past and future values, N is the number of samples used to train the model and w indicates the noisy term and H is the horizon size.

3.5.1 Recursive strategy

The recursive strategy is the most intuitive one. A neural network is trained for a single-step forecast. This model f is expressed by:

$$y_{t+1} = f(y_t, \dots, y_{t-n+1}) + w_{t+h} \tag{3.14}$$

,where the $t \in \{n, \dots, N-1\}$. The predicted value is fed to the network to forecast the successive point. This process is iterated until the predicted horizon H is reached. This strategy been applied for some real-case scenarios. However, it suffers from the cumulative error issued, since w is propagated and incremented at each iteration. [44]

3.5.2 Direct strategy

Contrary to the Recursive strategy, where only one model is trained, the Direct strategy consists of training H different models. Each of them forecasts one single point in the future. The formula is given by:

$$y_{t+h} = f_h(y_t, \dots, y_{t-n+1}) + w_{t+h} \tag{3.15}$$

where, $h \in \{1, \dots, H\}$, $t \in \{n, \dots, N - H\}$. This method might seem more robust compared to the recursive one, but it entails some drawbacks. Since each value is forecast independently by a different model, no statistical dependency between the predictions is taken into account. Last but not least, this method requires a lot of computational power and it increases the complexity [44].

3.5.3 Multiple-input-Multiple-output strategy

The methods explained in 3.5.1 and in 3.5.2 were limited to a single-step forecast. Both methods assume that the predicted values are stochastically independent. When the forecast horizon increases, the single-step forecast might be biased by the stochastic dependencies between the future values. To overcome this issue the network can be designed to forecast multiple points in the future. To deploy a multiple input multiple output (MIMO) strategy, also known as a joint strategy, the output layer of the network is set equal the size of the horizon you want to forecast. Moreover, the assumption of conditional independence between future values is neglected. The formula can describe the (MIMO) strategy:

$$[y_{t+H}, \dots, y_{t+1}] = F(y_t, \dots, y_{t-n+1}) + \mathbf{w} \tag{3.16}$$

A single multiple output model is learned, where $t \in \{n, \dots, N - H\}$, $\mathbf{F}$: $\mathbb{R}^d \rightarrow \mathbb{R}^H$, $\mathbf{w} \in \mathbb{R}^H$, which is the noise vector. This technique solves the plague of the cumulative error, seen in the 3.5.1, and it drops the assumption of conditional independence between the future values. However, it raises other problems. First of all, the complexity increases for large values of H. Moreover, the size of the output vector is constraint and cannot be modified and for large values of H it becomes unfeasible or at least not recommendable [44].

An example of a MIMO strategy is shown in Figure 3.4.

3.5.4 DirMO strategy

A combination of the MIMO and the Direct strategies is proposed in [45] [44], where the advantages of the direct strategy is combined with the MIMO strategy. Instead of forecasting the entire H with a single model, as in 3.5.3, l models are used, where each of them with m as output size, $l = \frac{H}{m}$. This strategy allows dropping the assumption of conditional independence between the predicted values, avoiding the cumulative errors, and increasing the flexibility of the forecasting horizons. Moreover, it reduces the complexity with

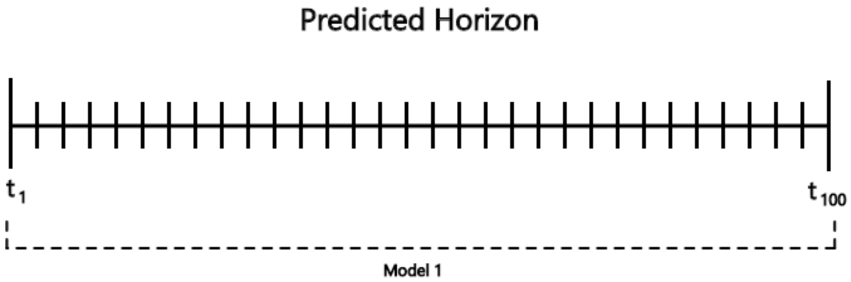


Figure 3.4: Example of MIMO strategy. Only one model is used to forecast the entire horizon. It is carried out increasing the size of the output layer until the forecasting horizon is reached.

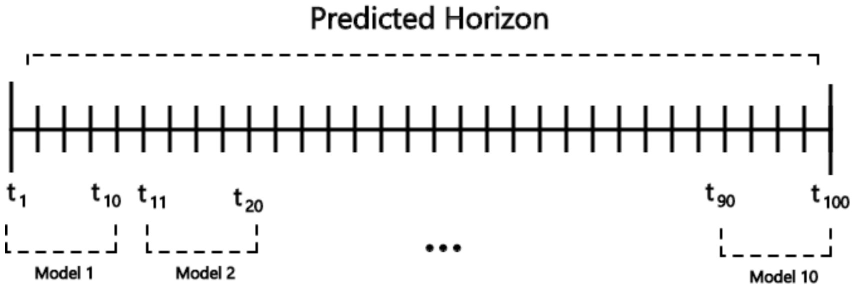


Figure 3.5: Example of DirMO strategy. The long line if the predicted horizon of 100 samples. If n is 10, each model predicts 10 points into the future until 100 is reached.

respect to the simple MIMO strategy, because it requires longer time to train a network with an output layer of size 100 than of size 10. Besides, smaller networks can be trained in parallel. The models are not trained on the same data, since the output is shifted ahead. For example, in Figure 3.5 the shift value is 10. To be explicit, the first model is trained shifting the output of only one sample ahead, the second model is trained shifting the output of 11 points ahead and so on. Hence, data changes according to the forecasting horizon. This approach is not a cascade approach, each network is independent to each other. How the data is reshaped is described in Section 3.7. Moreover, the approach suggest by [40], forces the to utilize a DirMO strategy when the forecasting horizon exceeds the number of units, since it fixes the number of units of the input and output layers of the network equal to the seasonality of the time series. An example of a DirMO strategy is shown in Figure 3.5.

3.6 Choice of the Error Metric

The evaluation of the quality of a predicting model might be arduous and it is not recommendable to judge the quality of a model based only on one metric [46]. During the years, several metrics have been used to evaluate a time series. In this work, we decided to explore six different error metrics, and we ended up choosing NMSE and sMAPE [47].

We started with the RMSE and MAE, which are quite good at evaluating the forecasting capability of the model. However, they cannot be exploited among different datasets, because they vary according to the scale of the dataset. Therefore, we discarded both of them.

We decided to explore the NMSE and MASE as possible alternatives to MAE and RMSE. Both of them judge the quality of the model but in a different way. MASE compares the current model with the naive seasonal forecast. The output gives an estimation of how better or worse is our model with respect to the naive one. For example, MASE of 0.5 means that the current forecast is two times better than the naive one. Both of them remove the scale-dependant issue; consequently, we can compare the models across different datasets. A drawback of the NMSE is the need of variance at the denominator, meaning that it cannot be calculated if only one sample is available. To overcome this issue, we have always considered a forecast with at least two samples.

Anyway, we preferred to show only the NMSE in Chapter 4, since it appears to be more reliable. Initially, to judge the accuracy of the forecast we consider MAPE, which provides a value that can be interpreted as an accuracy. However, it has a discontinuity when the actual values are zeros. Moreover, it has no upper-bound, due to the discontinuity. We decided, then, to use the symmetric MAPE. It does not present any discontinuity and if divided by 2, it can be read as an accuracy.

For the sake of completeness, we attached the results from the other metrics in Appendix A.

3.7 Preprocessing of data and setting up the data representation

Data has to be adequately reshaped, to train a neural network. In most of the cases, a supervised approach is preferred to train a neural network as suggested in [48]. Supervised learning consists of pairs of input/output examples, from which the network should be capable of mapping the relationship between input and output. Historical data can be seen as a vector of size N. This vector can be subdivided into smaller vectors of size n and then staked together, yielding to the shape of a matrix X $\in$ R^{Nxn} shown in 3.17

$$X = \begin{bmatrix} x_{N-1} & x_{N-2} & \dots & x_{N-n-1} \\ x_{N-2} & x_{N-3} & \dots & x_{N-n-2} \\ \vdots & \vdots & \vdots & \\ x_n & x_{n-1} & \dots & x_1 \end{bmatrix} \tag{3.17}$$

The output vector Y $\in$ R^{Nx1} is modeled in 3.18

$$Y = \begin{bmatrix} x_N \\ x_{N-1} \\ \vdots \\ x_{n+1} \end{bmatrix} \tag{3.18}$$

This approach simulates the natural flow of the time, where one sample is observed and stored one at a time. Once the limit is reached, the oldest value is discarded. At the beginning only one sample is available; hence, the remaining are set to zeros. This approach allow to forecast only one sample in the future. When multiple samples have to be forecast, Y become a matrix of the form $\mathbb{R}^{Nxf}$, where f are the number of points forecast.

Figure 3.6 illustrates an example of how the data have been reshaped. The interval time series is re-sized into daily time series. Each daily point, denoted with the d in the Figure, corresponds to the sum of 96 points at interval scale, while one hourly point is the sum of 4 points. The daily time series is the reshape in the matrix form, shifting the data points of one sample at the time, as described in Eq. 3.17. Each line of the matrix is then fed into the network. The samples inside the single batch are not shuffled, to avoid to lose the time dependency. The size of the matrix depends on the seasonality present in the data we are trying to model. At daily scale, we used 7, at hourly scale we used 168. In both cases, we were trying to model the weekly seasonality.

3.8 Tools

All the algorithms have been implemented in python. We exploited multiple libraries. We report only the most relevant for this thesis.

- pandas: used to handle and store dataframe
- matplotlib and plotly: utilized to plot the results
- keras with backend tensorflow: used to implement both the SANN and the LSTM
- tsmetrics: used to compute all the metrics
- statsmodel: used to implement SARIMA

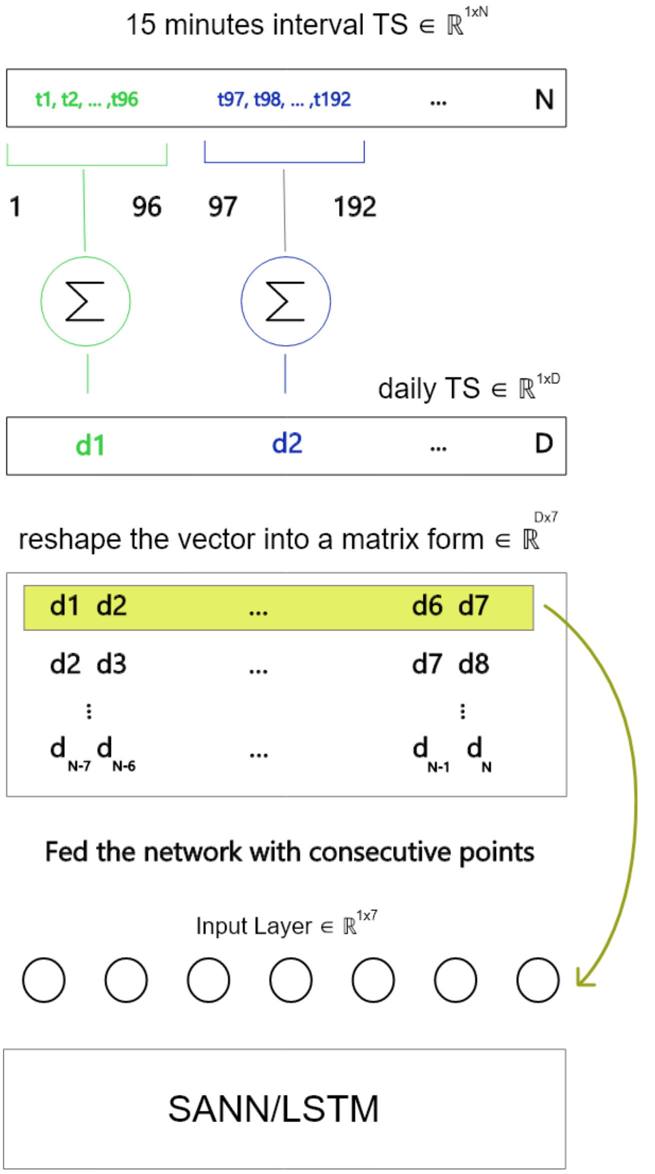
3.9 Preliminary analysis

In this chapter are reported all the techniques and concepts exploited to infer, which might be the best strategy to forecast both at daily and hourly scale. Stationarity is essential for the statistical model. KST becomes useful to demonstrate a possible future direction to improve the performances.

This Section wraps up the common steps shared among all the techniques.

3.9.1 Training, Validation, Testing set

First of all, data is split into a *training set* and *testing set.* The training set is used both for training and validating the models. The validation size is set at 5% of the training set size. All the values are summarized in Table 3.2 and in Table 3.3 at daily and hourly scale, respectively.



15 minutes interval TS ∈ ℝ^1xN
t1, t2, ... ,t96
t97, t98, ... ,t192
...
N
1
96
97
192
Σ
Σ
daily TS ∈ ℝ^1xD
d1
d2
...
D
reshape the vector into a matrix form ∈ ℝ^Dx7
d1 d2 ... d6 d7
d2 d3 ... d7 d8
d_N-7 d_N-6 ... d_N-1 d_N
Fed the network with consecutive points
Input Layer ∈ ℝ^1x7
SANN/LSTM

Figure 3.6: Full example of the pipeline followed to reshape data before feeding it into the network. t_1 to t_N are the points at interval scale. $1-96, 97-192$ indicate the indexes of the first and the last sample, comprised in a single daily value. At hourly scale they would be $1-4$, $5-8$. $d_1 to d_D$ are daily values.

The validation procedure is carried out in different ways according to the modeling technique. Samples are never shuffled during the training procedure since it would yield to a loss in the temporal correlation, which is the fundamental feature of a time series.

The size of the testing set is arbitrarily fixed at 100 points at daily scale and 160 points at hourly scale, meaning that the maximum forecasting horizon reached counts 100 and 160 points, respectively. It applies to all the modeling techniques. This value has been chosen, since they comprise both the short, medium and long term forecasting. On a daily level, 100 points are equal to 3 months of the forecast, while at hourly level 160 points are almost a week. Moreover, some dataset had few samples that it prohibits a larger testing size.

Datasets	**Training set**	**Validation set**	**Testing set**
Dataset 1	3013	158	101
Dataset 2	322	16	101
Dataset 3	837	44	101

Table 3.2: Number of time points of Training set, Validation set and Testing set at daily scale.

3.9.2 Stationarity

A time series X_t is said to be *strictly* stationary if the marginal distribution at time t is the same as any other point in time. It is summarized in Section 3.19, where $p(X_t, X_{t+k})$ does not depend on t, meaning that the first and the second order statistics of X_t are time invariant.

$$p(X_t) = p(X_{t+k}) \tag{3.19}$$

Most of the real world time series are not *strictly* stationary. Therefore, the relaxed constraint of *weakly* stationary is adopted. It requires the following three constraints to be satisfied:

- the expected values has to be approximated to a constant: $E(X_{t1}) = E(X_{t2}) = E(X_{t3}) = E(X_{tn}) = constant$
- the variance has to be approximated to a constant: $Var(X_{t1}) = Var(X_{t2}) = Var(X_{t3}) = Var(X_{tn}) = constant$

Datasets	**Training set**	**Validation set**	**Testing set**
Dataset 1	76032	3801	161
Dataset 2	10322	516	161
Dataset 3	23089	1154	161

Table 3.3: Number of time points of Training set, Validation set and Testing set at hourly scale.

- the co-variance has to be dependant on lag k: $Cov(X_{t1}, X_{1+k}) = Cov(X_{t2}, X_{2+k}) = Cov(X_{t3}, X_{3+k})$

A time series can be made stationary through the differencing operation. A time series, which becomes stationary after being differentiated once, it is said to be integrated of order one and denoted as $I(1)$

3.9.3 Akaike Information Criterion

It is a criterion to evaluate and compare statistical models and the mathematician Hirotsugu Akaike developed it. This criterion takes into account both the goodness of the model and its complexity. Akaike's Information Criterion basic formula is defined as:

$$AIC = -2(LogLikelihood) + 2K \tag{3.20}$$

,where K is the number of model parameters, and the $Log - likelihood$ measures the goodness of the fitted model. Lower is the number better is the model.

3.9.4 Augmented Dickey-Fuller test

The Augmented Dickey-Fuller test is a procedure to prove the stationary of a time series. The test was developed by the statisticians David Dickey and Wayne Fuller. It aims at verifying the null hypothesis that a unit root is present in the auto-regressive model.

Given a simple regression model as in 3.21, where ρ is a coefficient and ϵ_t is the residual, a unit root would be present if $\rho = 1$. The problem can be rewritten as in 3.22.

$$y_t = \rho y_{t-1} + \epsilon_t \tag{3.21}$$

$$\Delta y_t = \alpha + \beta y_{t-1} + \delta_1 \Delta_{y_{t-1}} + \cdots + \delta_{p-1} \Delta_{y_{t-p+1}} + \epsilon_t \tag{3.22}$$

where Δ is the first difference operator, α is a constant and β the coefficients on a time trend and p the lag order of the autoregressive process.

The basic idea behind the ADF test is that a stationary (at least trend stationary) time series tends to return to a constant value. Large values are followed by smaller values leading to negative changes and the other way around. If those changes compensate each other, then the null hypothesis can be rejected and the time series can be considered stationary. On the other hand, if a trend is present or the time series is integrated, the changes do not compensate each-others as it may be observed in a random walk process.

3.9.5 ACF and PACF graphs

Auto-correlation function is exploited to understand the linear relationship between two variables. It is usually carried out on p lags from the current one. The formula to compute the auto-correlation is shown in 3.23

$$\rho_k = Corr(Y_t, Yt_p) = \frac{Cov(Y_t, Y_{t-p})}{\sqrt{var(Y_t)var(Y_{t-p})}} \tag{3.23}$$

The partial autocorrelation function is explored to understand the relationship between the current sample and the one at lag p, when the effect of all the other time lags are removed. It is shown in 3.24

$$\begin{pmatrix} \rho(0) & \rho(1) & \cdots & \rho(k-1) \\ \rho(1) & \rho(0) & \cdots & \rho(k-2) \\ \vdots & \vdots & \vdots & \vdots \\ \rho(k-1) & \rho(k-2) & \cdots & \rho(0) \end{pmatrix} \begin{pmatrix} \phi_{k1} \\ \phi_{k2} \\ \vdots \\ \phi_{kk} \end{pmatrix} = \begin{pmatrix} \rho(1) \\ \rho(2) \\ \vdots \\ \rho(k) \end{pmatrix} \tag{3.24}$$

3.9.6 Kolmogorov-Smirnov Test

There are different ways to understand the similarity between the two distributions. The Kolmogorov-Smirnov test (KS) is a non-parametric test, which compares the shape of two distributions. Be X a causal continuous variable (CdF) and be $F(x)$ its cumulative density function, it is required to test the similarity between $F(x)$ and $F_0(x)$. $F_0(x)$ may be a reference distribution, as well as, a second empirical sample distribution. The test quantify the distance between the two distributions. The hypothesis can be mathematically written as follows:

Hypothesis 0 (H0): $H_0 : F(x) = F_0 \quad \forall x$

Hypothesis 1 (H1): $H_1 : F(x) \neq F_0 \quad for\, some\, x$

The hypothesis does not only refer to a single parameter of the variable X, but at the entire distribution. In the case two empirical distribution are compared, the KS test is given by 3.25,

$$D_{n,m} = sup_x |F_{1,n}(x) - F_{2,m}(x)| \tag{3.25}$$

and the null hypothesis is rejected at level α if:

$$D_{n,m} > c(\alpha)\sqrt{\frac{n+m}{nm}} \tag{3.26}$$

In general α is compute as :

$$c(\alpha) = \sqrt{-\frac{1}{2}ln(\alpha)} \tag{3.27}$$

This test is exploited to infer the similarity among the days of the weeks and to understand whether to model them with the same SARIMA parameters.

3.9.7 Parameters analysis for SARIMA model

Before implementing a systematic experiment, a preliminary analysis is performed to set the upper-bounds of some parameters, which is carried out only on the training set.

All the parameters have been chosen according to the ACF and PACF graphs. The seasonality parameter is quite clear for all the datasets, hence it is not validated. A significant autocorrelation is observed at lag 7^{th} at daily level and at lag 24^{th} and 168^{th} for hourly level.

Concerning the P, p and Q, q parameters, they are difficult to determine. However, in some cases, the autocorrelation decreases after the second lag. Besides, time constraints and limited computational power have played a significant role. All the values till seven have been analyzed, but the time required to fit the model grew exponentially and little improvements were observed for values greater than 2. Therefore, the upper bound for all the parameters are set at 2. Concerning the i, I parameters, they are set at 1, since all the datasets becomes stationary after the first level of integration. Stationarity has been checked with the Dickey-Fuller test. A time series is classified as stationary if its p-value is lower than 0.05. The only exception, dataset 1 at daily level requires a log transform to guarantee the stationary. This operation has been properly inverted after the prediction. The AIC criterion has been used to evaluate the models. Furthermore, the upper-bound of the data points used to fit the model is set at 300 points. It is also the more significant number of samples exploitable for the second dataset.

3.9.8 Validation of the hyper-parameters for LSTM and SANN

Neural networks require the validation of multiple parameters, which are: optimizer, loss function, activation function, size of the hidden layer, input window size, output window size, the number of epochs necessary to achieve acceptable results and, last but not least, the forecasting technique.

We tuned the all the aforementioned parameters for the LSTM and for the SANN, and at hourly and daily scale.

Time series is a particular type of data characterized by the temporal correlations. Therefore, classical K-fold cross-correlation cannot be exploited without losing the temporal dependencies among data points and consequently causing a data leakage. We preferred to implement a hold-out cross-validation approach. We split our dataset into training, and test set. Then, the training set is again split into a training subset and a validation subset. The validation subset is exploited to tune the hyperparameters. We utilized the training set to train the models and we validated on the validation set, whose values are reported in Table 3.2 and in Table 3.3.

At daily scale, we trained the models available to set the upper bounds. At hourly scale, we used only 10000 points of each dataset as a training set to speed up the process.

An example of hold-out cross-validation is shown in Figure 3.7

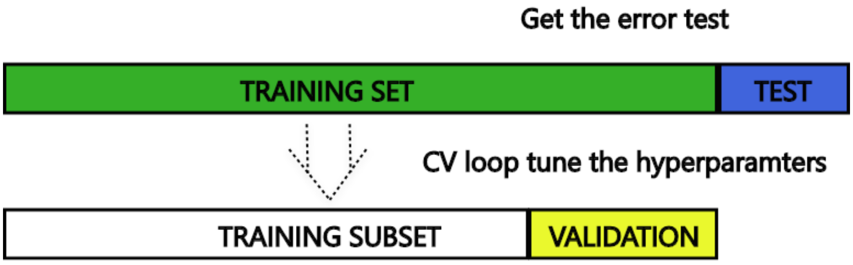


Figure 3.7: Example of hold-out cross validation. Data is split into training and test set. Then the training set is subdivided into a training subset and a validation set, used to fine tuned the hyper parameters. Validation set size is the 5% of the training set.

Seasonal ANN model

All the different forecasting techniques described in Section 3.5 have been tested. From a shallow analysis, DirMO strategy resulted in the best output. Therefore, it was applied for all the experiments at daily scale.

Seasonal ANN architecture has S and H parameters to be validated. Therefore, according to the ACF some possible values of S are selected. On the other hand, the values of H are chosen paying attention to not exceed in the number of parameters needed by the model to be trained and following the guidelines suggested by [40].

The same procedure is followed at hourly scale. MIMO strategy resulted in the best performances, so it was applied to all datasets.

The best combination is (7,7,7) model for the daily scale and (168,168,168) for the hourly model.

As far as concern the epoch step and the epoch upper bound, we have empirically observed that after 1500 and 2000 epochs the model starts to overfit the data.

LSTM Model

All the different forecasting techniques described in Section3.5 have been tested. From an early analysis, DirMO strategy results in the best output at daily scale, and MIMO at hourly scale. We applied both techniques for the experiments.

LSTM is extremely versatile and multiple networks can be designed starting from the most simple model called Vanilla.

Four different architectures have been tested listed below:

- Vanilla
- Stacked
- Bidirectional
- CNN-LSTM

Vanilla model has only one hidden layer between the input and the output layer. Stacked LSTM has two hidden layers. Bidirectional LSTM is a particular type of LSTM that removes the constraint on the future information embedded with the feedback connections. CNN-LSTM model is a combination between a convolutional neural network and the recurrent neural network. The scope of the convolutional neural network is to filter out the input and to extract the most valuable features, used to feed the LSTM.

Moreover, we tried to change the resetting time of LSTM cell. We compared a stateful vs. a stateless model. Stateful indicates that the state of the network is reset after each epoch, while stateless means the state resets after each batch.

In order to model the parameters, I have used the same approach used for the SANN. I kept the same size both for the input and for the output layer, indicated with S, and changed the size of hidden layer, indicated with H.

3.10 Systematic experiment

Once the preliminary analysis is concluded, and all the upper-bound and the parameters are chosen, we design a systematic experiment to explore the whole space of solutions comprised within the upper-bounds. We decided to perform a validation step, to chose the best model to test it on the test set. We trained on the training subset and validated on the validation set. We kept the test set fixed for the sake of simplicity. We repeat the experiments increasing the size of the training set to observe the changes into the performances. To have an idea of how much the statistical error may influence the forecast we repeated the experiments 5 times and computed the standard deviation of all of them. Among them, we pick the best model according to the validation error based on the NMSE.

3.10.1 LSTM and SANN

Both for LSTM and SANN architectures, the model with the best hyperparameters is chosen and tested. The input is scaled between -1 and 1. The model is trained until the epochs upper-bound is reached. The training starts with training steps and it is incremented by the *epoch step*. The model is evaluated through all the metrics, but the best model is chosen according to the lowest NMSE.

3.10.2 SARIMA

The systematic experiment for the SARIMA method is designed as a Grid search, meaning that all the possible combinations, within the upper-bounds, are validated and scored according to AIC criterion. The model with the best score is tested on 100 and 160 points at daily and hourly scale, respectively. This procedure is iterated, varying the input length until the upper-bound is reached. The increasing steps are 30 and 24 for daily scale and hourly scale, respectively.

3.11 Evaluation of the statistical relevance

To derive statistically relevant conclusions, we perform the Kruskal-Wallis test on the NMSE metric. For each dataset, we carry out the Kruskal-Wallis test. In other words, we evaluated the similarity among the three models within each dataset. Moreover, we performed the Kruskal-Wallis on the whole dataset concatenating the results of the three different models on the three datasets. The evaluation is carried out both at daily and hourly scale, counting eight different tests.

We wanted to assess also the difference between the models' results in terms of the forecast horizon. We split the NMSE results in two equals parts, and the first half is labeled as short term forecast and the last part as long term forecast. We evaluate the difference with the Kolmogorov-Smirnov.

Both Kolmogorov-Smirnov and Kruskal-Wallis tests do not provide any information on how large is the size of the difference. To evaluate the actual size of this difference, called "effect size", we exploited the Cohen's d and the Henges' g. More details are given in Sections 3.11.1 and 3.11.2.

These two metrics provide us quantitative values that quantify the difference between the two groups. Since we have three methods, but Cohen's d can be calculated only on two groups, we decided to compute the metric for all the possible combinations.

To exemplify the concept, assume to have three distributions, d1,d2,d3. We computed the Cohens'd for (d1,d2) (d1,d3) (d2,d3).

When the difference was relevant, then we computed the mean and we draw our conclusion based on that.

3.11.1 Kruskal-Wallis H-test for independent samples

The Kruskal-Wallis test is utilized to understand whether two or more datasets are different. In particular, we want to assess if the central tendency (mean or median) is statistically significant.

The null hypothesis (H_0) is the assumption that the datasets are instances of the same distributions. Hence, they have the same population parameters, such as mean or median.

When the null hypothesis is rejected the dataset belongs to two different populations. It suggests that the difference between the populations' parameters may be significant.

In the machine learning field, this test is used to confirm the difference in skills between different models.

The test returns a statistical value named as H and a p-value. Both of them can be used to reject the null hypothesis.

The usage of the p-values is straightforward. Once, we have set an α value we can compare it directly and:

- $p - value < \alpha \quad H_0$ is rejected
- $p - value > \alpha \quad H_0$ is not rejected

The p-value can be interpreted as the probability that the null hypothesis is verified. In this case, the p-value states the probability that the two samples are drawn from a population with the same distribution.

Some common values of α are 0.05, 0.01 and 0.001.

3.11.2 Cohen's d and Hedges' g

Both the Kolmogorov-Smirnov test and the Kruskal-Wallis test do not provide any information about the effective size of this difference. Indeed, both the p-values and the test statistic do not comment about the size of the difference. Different tests can be performed to gain some extra information on the significance of the difference.

A standard metric used to compute the difference between the mean value of groups is called Cohen's d. This metric is computes as in Eq 3.28

$$\text{Cohen's d} = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{(n_1-1)S_1^2+(n_2-1)S_2^2}{n_1+n_1-2}}} \tag{3.28}$$

$\overline{X}_1$ and $\overline{X}_2$ are the mean values of the two groups. S_1^2 and S_2^2 are, instead, the variances of the two groups, and n_1 and n_2 are the sizes of the two groups. Hence also groups with different sizes can be evaluated with the d metric. Cohen's d can also be seen as the difference between two means divided by a standard deviation for the data. This measure has become very popular since along with the metric it Cohen provided also some guidelines.

- d = 0.01 –> very small effect size
- d = 0.20 –> small effect size
- d = 0.50 –> medium effect size
- d = 0.80 –> large effect size
- d = 1.20 –> very large effect size
- d = 2.00 –> huge effect size

However, this metric was found to be statistically upwardly biased. Therefore, a new metric was developed known as Hedges' g. To correct the biased Cohen's d has to be multiplied for a constant. Hedges' g is then computed as in Eq 3.29

$$\text{Hedges' g} = \text{Cohen's d} * (1 - \frac{3}{4*(n_1-n_2)-9}) \tag{3.29}$$

Chapter 4

Results

This chapter presents an overview of the most significant results obtained from the simulations. As explained in Chapter 3 the experiments are carried out at different time scales and varying the input size.

4.1 Results of preliminary analysis

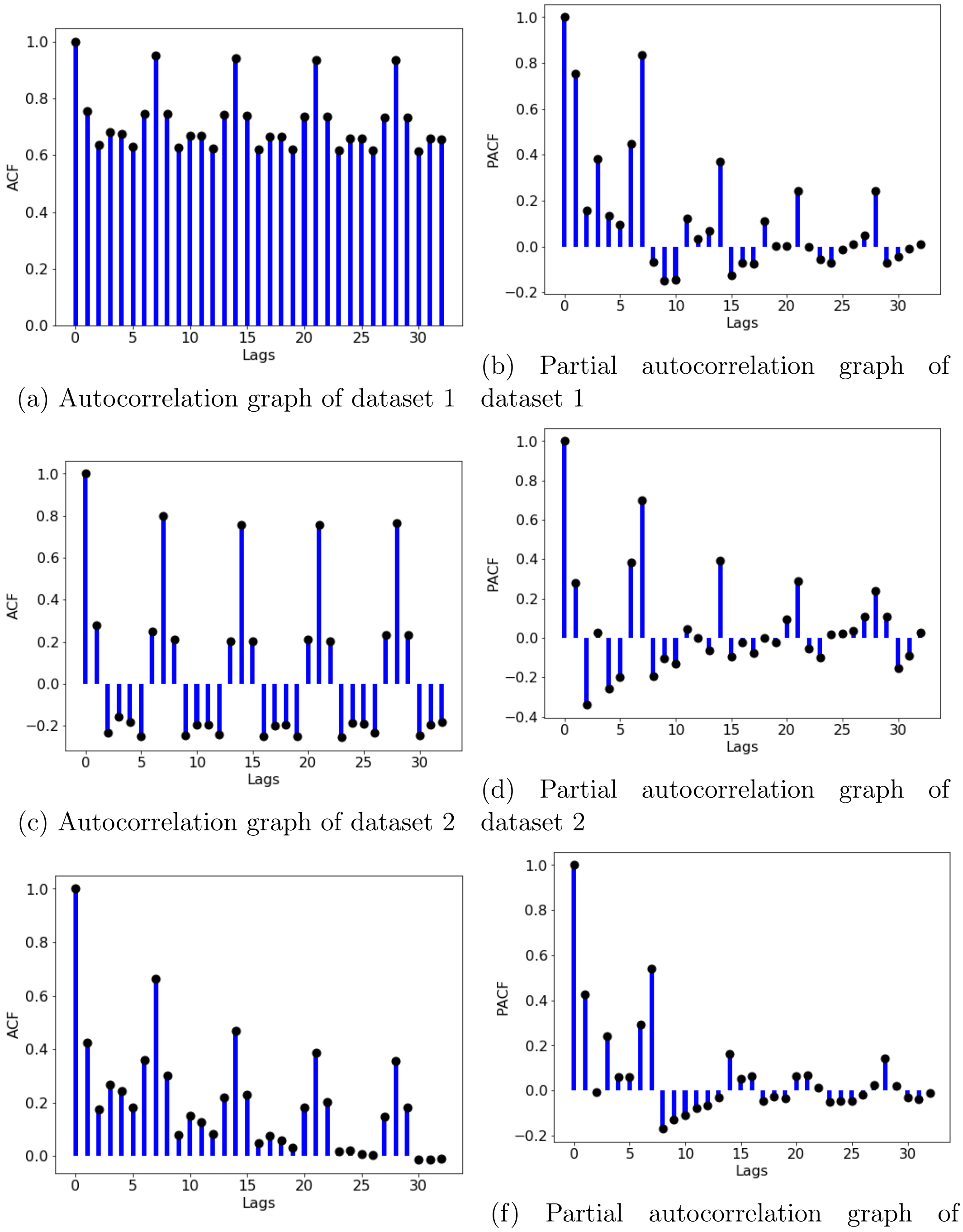
This section delves into the analysis of seasonalities and temporal correlation. Two different approaches are utilized. We exploit the ACF and PACF graphs to infer the seasonality and the KST to investigate other possible temporal correlations. It gives some information about the relevant outcomes obtained during the preliminary analysis, explained in Section 3.9.

4.1.1 ACF and PACF graphs evaluations

To better understand the seasonality, we have exploited the autocorrelation and partial autocorrelation graphs. More details about these graphs are reported in Section 3.9.5.

ACF and PACF suggest us, which lag, of the time series, is more correlated with the current one. At the daily scale, we plotted 32 days to see the correlation between the same day of two different months. Observing Figure 4.1 we may note a repetitive peak on all the autocorrelation graphs and partial autocorrelation graphs. It repeats itself precisely at the 7^{th} lag, suggesting that, all the datasets presents an incisive weekly seasonality. This sounds reasonable since, at the daily scale, a Monday is supposed to be more similar to the Monday of the last week than to the other days of the week. Moreover, ACF graphs suggest the presence of different similarity's levels among the various days. For example, in Figure 4.1a all the samples are above 0.5, indicating a high correlation between the current sample and all the others. However, it is not always true. For example, Figure 4.1c, shows that only the 7^{th} lag has a significant correlation with the current lag, while all the other stay below 0.3.

Figure 4.2 shows an overview of the autocorrelation and partial autocorrelation at hourly scale. In this case, we have plot 168 points, corresponding to the number of hours in a week. At first sight, we can immediately infer



(a) Autocorrelation graph of dataset 1

(b) Partial autocorrelation graph of dataset 1

(c) Autocorrelation graph of dataset 2

(d) Partial autocorrelation graph of dataset 2

(e) Autocorrelation graph of dataset 3

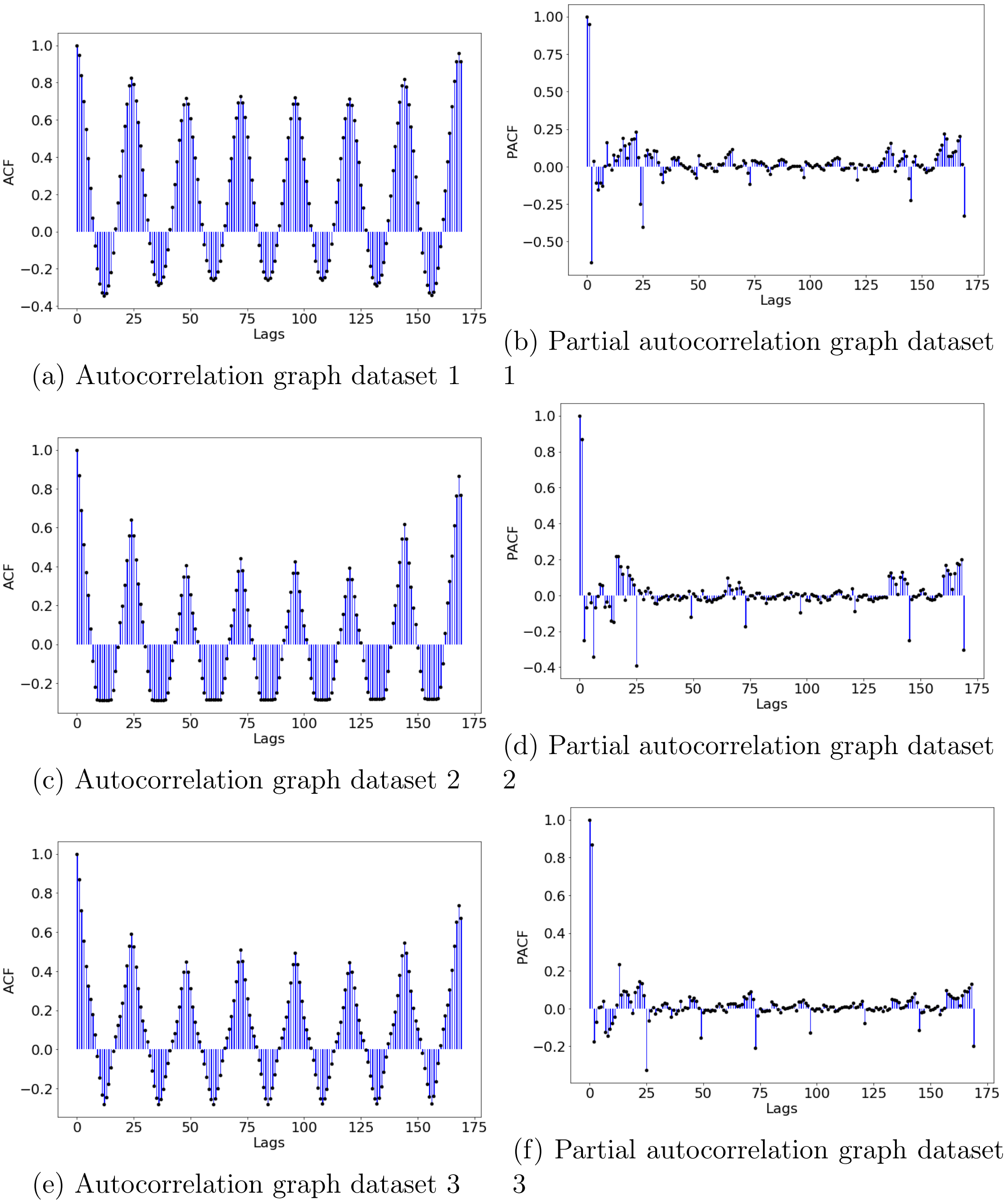
(f) Partial autocorrelation graph of dataset 3

Figure 4.1: On the left side the autocorrelation graphs and on the right the partial autocorrelation graphs of all datasets at daily scale. All the graphs are computed on a random Monday. It highlights the presence of seasonality at lag 7^{th} in each dataset.

that the partial autocorrelation graphs for all the dataset do not provide useful information. It can merely observe a repetitive peak every 24 lags. On the other side, ACFs give us valuable information. It becomes clear how our data is correlated every 24 points and every 168 points. It sounds plausible, meaning that the current hour shares a high correlation with the same hour of the previous day, but even a higher correlation with the same hour of the same day of the last week. This leads us to the conclusion that hourly data presents two evident seasonality, at daily and at weekly scale.

Moreover, the characterizing sinusoidal behavior suggests a possible correlation between the inner structure of some days, meaning that the distribution of the values within some days of the week might be very similar.

To investigate it, we decided to utilize a KS test, which is non-parametric and used to understand whether two different time series belongs to the same distribution.



(a) Autocorrelation graph dataset 1

(b) Partial autocorrelation graph dataset 1

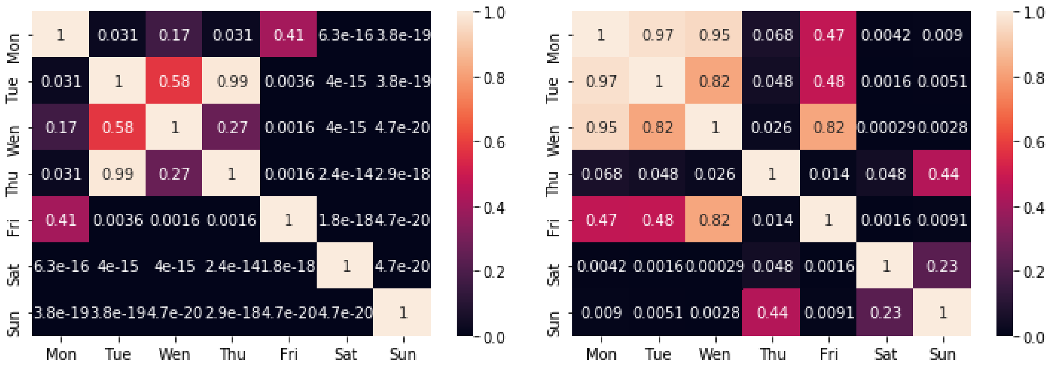
(c) Autocorrelation graph dataset 2

(d) Partial autocorrelation graph dataset 2

(e) Autocorrelation graph dataset 3

(f) Partial autocorrelation graph dataset 3

Figure 4.2: On the left side the autocorrelation graphs and on the right the partial autocorrelation graphs of all datasets at hourly scale. All the graphs are computed on a random Monday. It highlights the presence of seasonality at lag 24^{th} in each dataset.



(a) dataset 1, daily level

(b) dataset 1, hourly level

Figure 4.3: Example of an Heatmap showing the p-values of the KS test of dataset 1 at hourly and daily level. When the p-values are close to 1 the two days can be considered as instances of the same distribution.

4.1.2 Kolmogorov-Smirnov's test (KST)

This Section explains how and why we decided to exploit Kolmogorov-Smirnov's test. The test is explained in details in Section 3.9.6.

We utilize the test to evaluate the similarity among days. As we said in the previous section, ACF graphs suggest a possible correlation between days. We hypothesize that some days may be considered as instances of the same process, and then we may conclude that it would be possible to combine those days.

Splitting the dataset helps in decreasing the complexity of the system and thereby hopefully increase generalization capability.

For the sake of visualization, we report an example in Figure 4.3, showing the test performed on the first dataset, at daily and at hourly scale. We have performed the test on 300 samples, testing each day against all the others, and collecting the p-values of the KST. The diagonal is always one because each day is identical to itself. When the values are close to one, the null hypothesis is valid, and then two days can be considered instances of the same distribution. Figure 4.3a, highlights a interesting behaviour. We observe that Friday is very close to Monday and Tuesday Wednesday and Thursday have high p-values. On the other hand, we can see that weekend days presents very low p-values so they should be treated separately.

Therefore, a possible approach may be to split the dataset in 4 combinations, which would be

1. 'Mon, Fri'
2. 'Tue, Wen, Thu'
3. 'Sat'
4. 'Sun'

Scale	p	i	q	P	I	Q	Seasonality	Input size [min, max]
daily	2	1	2	2	1	2	7	[30, 300]
hourly	2	1	2	2	1	2	24	[24, 240]

Table 4.1: upper-bounds set for SARIMA's parameters after a preliminary analysis on the three datasets. The min and max amount of samples exploited for the training, and the increasing step are reported both at hourly and daily scale

A model would be trained on each combination, then, 4 different forecasts would be done. The advantage of splitting the dataset is to reduce the complexity of the problem. For a single model, it might be to complicated to model the patterns of all the seven days. A possible disadvantage of this approach is the possibility to lose the cross-correlation in the forecast since part of the dataset is not be exploited during the training. In this case, Monday and Friday are considered as instances of a distribution. Therefore, all the other days are not included in the training set. However, if for example, there was a correlation between Monday and Sunday, we would not be able to exploit it because Sundays are not included in the training set on Monday and Friday. Finally, we can observe that the behavior is different if we look at the hourly scale. Therefore, we should combine the days differently.

4.1.3 SARIMA

We exploited the ACF and PACF graphs to choose most of the parameters. The seasonality parameter is set at 7^{th} at daily level and at 24^{th} and 168^{th} for hourly level.

To set the P, p and Q, q parameters we carried out an iterative test, where it turned out that it was not feasible to perform a systematic analysis for values greater than 2. We observed that for an input size greater than 300, the improvement in the forecast was negligible.

A summary of the upper bounds chosen values can be found in Table 4.1. The minimum values were set at 0 by default.

4.1.4 Preliminary analysis results for Neural Networks

Table 4.2 summarize the optimizer, loss function and activation function tested. We fixed these parameters both for LSTM and for SANN. Among the optimizers *adam* gave slightly better performances. *MAE* was preferred to the MSE since most of the metrics derived from it. Regarding the activation function, no significant differences where noticed, hence *ReLu* is chosen.

SANN

The parameters chosen for the SANN architecture are summarized in Table 4.3.

Hyper parameter	Names
optimizer	**adam**, adamMax, RMSprop
loss function	**MAE**, MSE
activation function	sigmoid, **ReLu**

Table 4.2: Overview of the Optimizer, loss functions and activation functions tested both at the daily and at the hourly scale.

Time Scale	S	H	Max Epochs	Epoch Steps
Daily	**7**, 14, 28	3, **7**, 10	1500	25, **50**, 100
Hourly	24, 84, **168**	24, 84, **168**	2000	25, 50, **100**

Table 4.3: Hyper parameters tested separately for daily and hourly for the SANN architecture. S refers to the size of the input and output layers, while H refers to the size of the hidden layer.

The best combination is (7,7,7) model for the daily scale and (168,168,168) for the hourly model. A DirMO strategy is utilized at daily scale while a MIMO strategy is utilized at hourly scale. As far as concern the epoch step and the epoch upper-bound, we empirically tested that after 1500 and 2000 epochs the model starts to over-fit the data.

LSTM

All the parameters chose for the LSTM architecture are summarized in Table 4.4.

Time Scale	S	H	Max Epochs	Epoch Steps
Daily	7, **14**, 28	3, 7, **10**	800	25, **50**, 100
Hourly	48, 96, **168**	48, 96, **168**	400	50, **100**

Table 4.4: Hyper parameters tested separately for daily and hourly, for the LSTM architecture. S refers to the size of the input and output layers, while H refers to the size of the hidden layer.

The best combination is (7,7,7) model, with a DirMO strategy at the daily scale and (168,168,168), with a MIMO strategy at the hourly model.

We have empirically found out that *Vanilla* model was the best trade-off with respect to the quality of the forecast and the training time. However, CNN-LSTM provided the best performance.

From the initial analysis, we decided to explore the *stateless* configuration since it seems to outperform the *stateful*.

Moreover, from preliminary analysis, I have noticed that the model was not performing as expected. I suspected it was due to the noise. Therefore, I have smoothed the signal with a mean filter of size 2, removing high frequency of the signal. Anyway, this operation has not modified the true nature of the signal. It helped the network to infer the general trend.

As far as concern the epoch step and the epoch upper-bound, we empirically tested that after 1500 and 2000 epochs the model starts to over-fit the data.

4.2 Summary Set-up

Before presenting the results, a summary of the parameter used to train the neural network is provided below. Table 4.5 summarizes the parameters used both at daily scale and at hourly scale for the Seasonal ANN model and LSTM model.

Parameters	Daily scale	Hourly Scale
Optimizer	adam	adam
Loss Function	MAE	MAE
Activation Function	ReLu	ReLu
Model	(7,7,7)	(168,168,168)
Forecasting Strategy	DriMo,	MIMO

Table 4.5: Summary of the Seasonal ANN and LSTM parameters chosen after the preliminary analysis

4.3 Statistical method vs Machine learning

In the following Section, the results obtained from the simulations are presented. For each methodology, the best model is selected according to its average performance along the whole horizon. NMSE and sMAPE are chosen to compare the models since they are deemed to be some of the best metrics to judge forecasting models.

4.3.1 Day forecast

This section presents a performance overview of three methodologies explained in Section 3. For the sake of visualization and intelligibility, only one model is plotted. As stated in 2.2 multiple metrics are used to evaluate the performances of the forecasting models. However, to compare the models across the different datasets, NMSE and SMAPE appear to be the most reliable. It has been empirically observed that NMSE leads to choose the most meaningful model. The rationale behind this choice is explained in Chapter 5.

To better observe the tendency of the performance along the horizon, the metrics are computed every 10 points until the maximum of 100 is reached.

Since NMSE has no upper bound, we have chosen to set it to 1 arbitrarily. We ascertain that a model with an NMSE higher then 1 leads to a meaningless forecast.

Figure 4.4 reports NMSE and sMAPE graphs of the best model as aforementioned.

Datasets	**Input Length NMSE**			**Input Length SMAPE**		
	Sarima	**SANN**	**LSTM**	**Sarima**	**SANN**	**LSTM**
Dataset 1	300	3017	3017	90	3017	3017
Dataset 2	270	338	338	270	338	338
Dataset 3	240	874	300	240	874	874

Table 4.6: Overview of the training size of the models shown in Figure 4.4

Dataset 1	Dataset 2	Dataset 3
$(2,0,1)(1,0,1)_7$	$(2,0,2)(1,1,2)_7$	$(0,1,2)(0,1,2)_7$

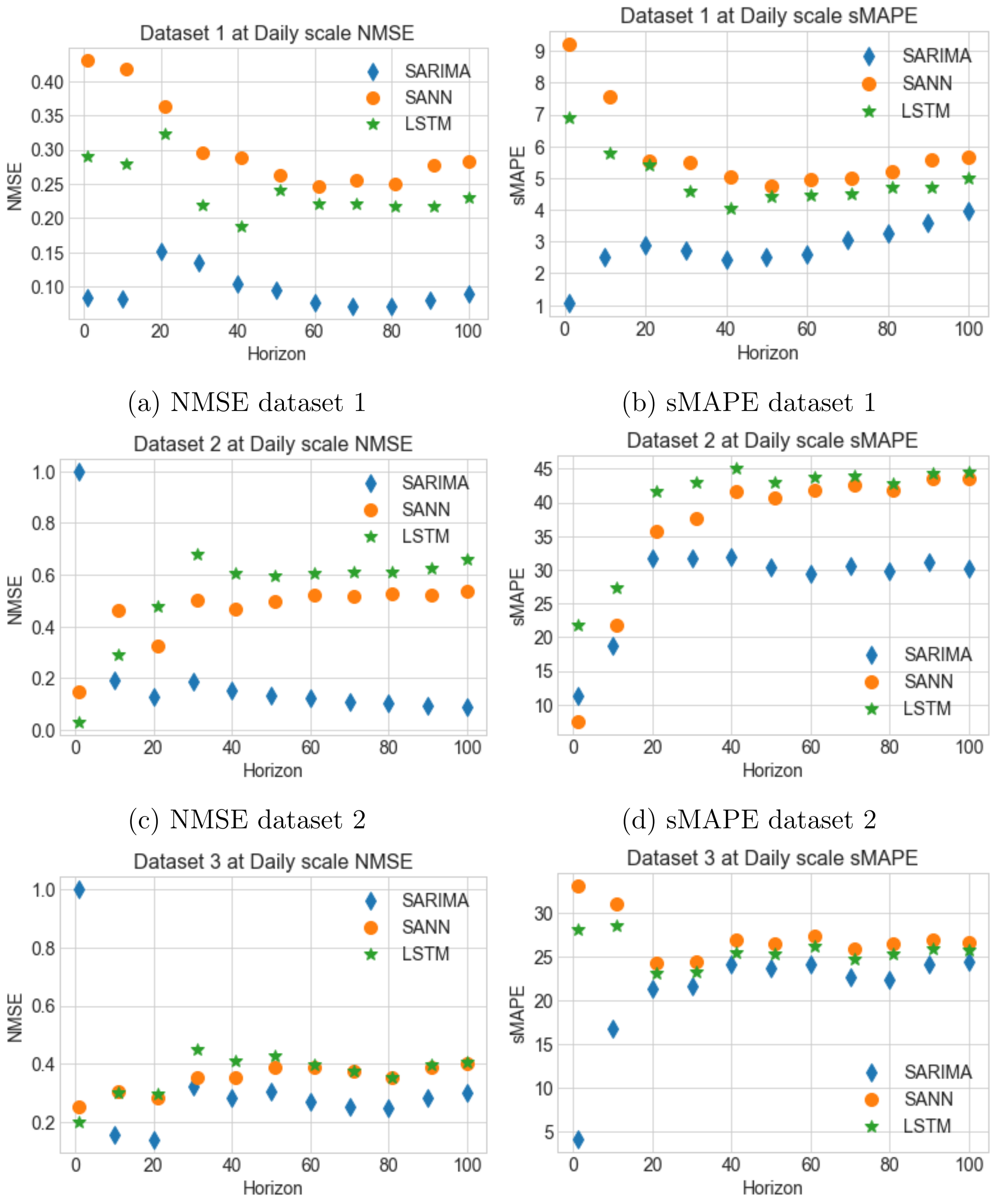
Table 4.7: Parameters of the SARIMA's models reported in Figure 4.4.

Table 4.6 reports the amount of samples used to train the models shown in Figure 4.4. We note that SANN and LSTM sizes correspond to the maximum amount of samples available, while for SARIMA those values changes.

For the sake of completeness, we have reported in Table 4.7 the SARIMA's parameters of the models shown in Figure 4.4 according to the dataset.

We note immediately that SARIMA outperforms both Seasonal ANN and LSTM. In general, it provides a more stable and reliable forecast along the whole forecast horizon. On the first data set, all algorithms perform quite good. From an accuracy point of view, the three models are above 90%, which is a good result. Regarding the second dataset, it provides the worst outcome. Even though SARIMA manages to remain above 80% of accuracy, ML techniques reach it only in the short term. After 20 points, both Seasonal ANN and LSTM performances degrade sharply till 60%. The third data set shows an interesting behavior from the metric point of view. Observing NMSE, it may be inferred that all the three models are very close to each other. One may expect similar performances also from an accuracy point of view. However, looking at the sMAPE, we observe that SARIMA accuracy is above 85% while SANN and LSTM manage to reach 75%, which is a big gap compared to the one showed by the NMSE.

To sum up, these Figures highlight how SARIMA outperforms SANN and LSTM techniques at daily scale.



(a) NMSE dataset 1

(b) sMAPE dataset 1

(c) NMSE dataset 2

(d) sMAPE dataset 2

(e) NMSE dataset 3

(f) sMAPE dataset 3

Figure 4.4: On the left the best performances of each method are reported according to the NMSE metric for each dataset. The forecast horizon is set at 160 points. The metric is computed every 10 points. On the right,NMSE is replaced with sMAPE.

4.3.2 Hourly forecast

This section presents a performance overview at hourly scale of three methodologies explained in Section 3.

To better observe the tendency of the performance along the horizon, the metrics are computed every 10 points until the maximum of 160 is reached.

Datasets	**Input Length NMSE**			**Input Length SMAPE**		
	Sarima	**SANN**	**LSTM**	**Sarima**	**SANN**	**LSTM**
Dataset 1	336	76032	30000	336	76032	30000
Dataset 2	240	10322	10322	240	10322	10322
Dataset 3	240	23089	23089	264	23089	23089

Table 4.8: Overview of the training size of the models shown in Figure 4.5

Dataset 1	Dataset 2	Dataset 3
$(2,0,1)(2,1,2)_{24}$	$(1,1,2)(0,1,2)_{24}$	$(1,0,2)(2,1,1)_{24}$

Table 4.9: Parameters of the SARIMA's models reported in Figure 4.5

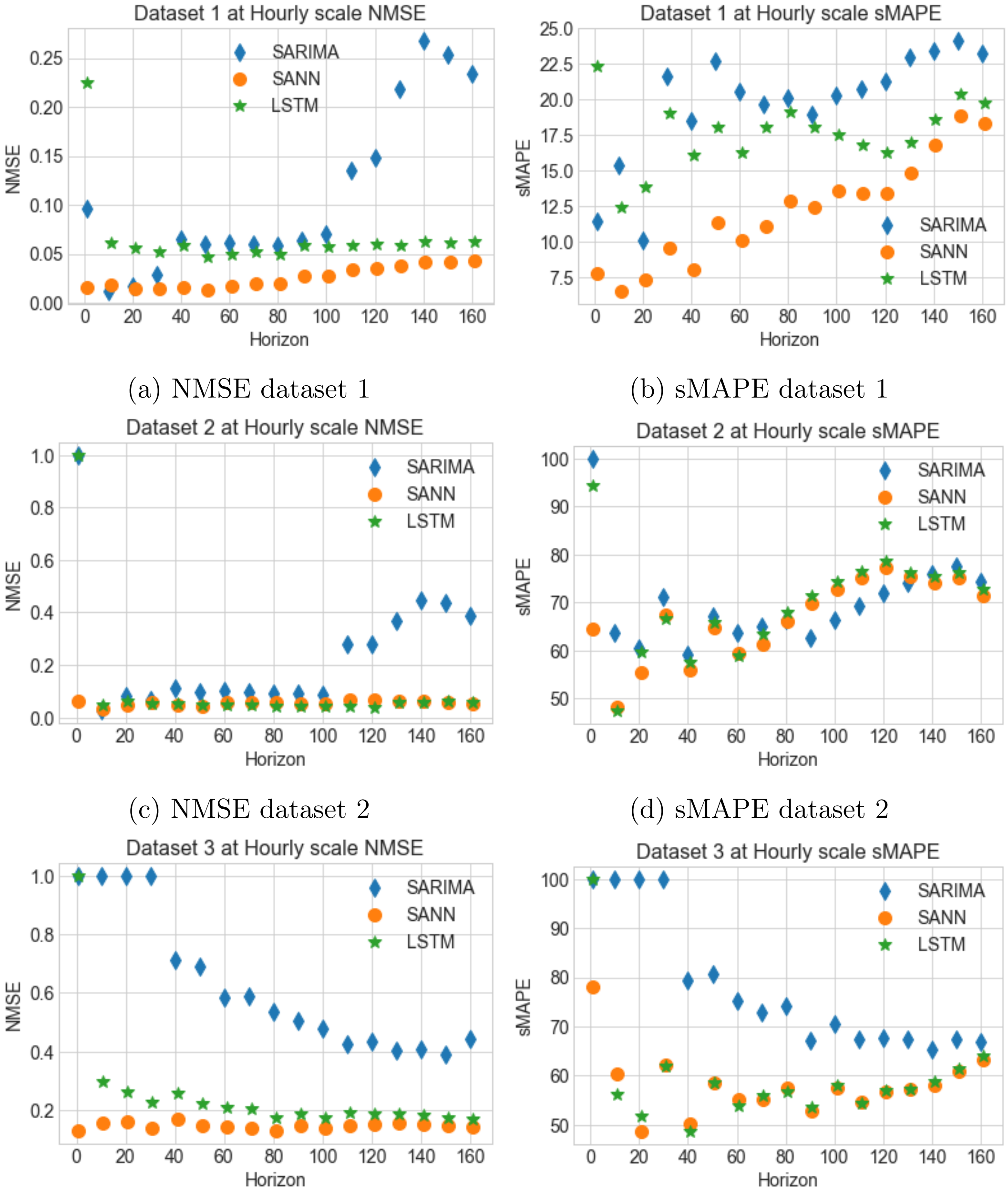
Since NMSE has no upper bound, we have chosen to set it to 1 arbitrarily. We ascertain that a model with an NMSE higher then 1 leads to a meaningless forecast.

In Figure 4.5 we reported the results of LSTM tested on the non filtered data.

Table 4.8 reports the amount of samples used to train the models shown in Figure 4.5.

We have reported in Table 4.9 the SARIMA's parameters of the models shown in Figure 4.5 according to the dataset.

Figure 4.5 reports NMSE and sMAPE graphs of the best model as aforementioned. On the hourly scale, we note that machine learning techniques perform much better compared at hourly scale. In the first dataset, NMSE remains quite stable, whereas for the SARIMA model, raises sharply after 100 points. From an accuracy point of view, SANN and LSTM are quite similar, and both of them manage to stay below 80%. We observe the same behavior also for the second dataset. After 100 points, SARIMA performances drop sharply. SANN and LSTM are quite similar both from NSME and SMAPE point of view. Also at hourly scale, the second dataset results in the worst performance, probably due to the lack of samples. The third dataset, show an exciting pattern for the SARIMA model, which decreases over time. This may due to the catch of the trend. NMSE for machine learning is quite stable. On the other side, SMAPE, highlight a drop in the accuracy after 120 points.



(a) NMSE dataset 1

(b) sMAPE dataset 1

(c) NMSE dataset 2

(d) sMAPE dataset 2

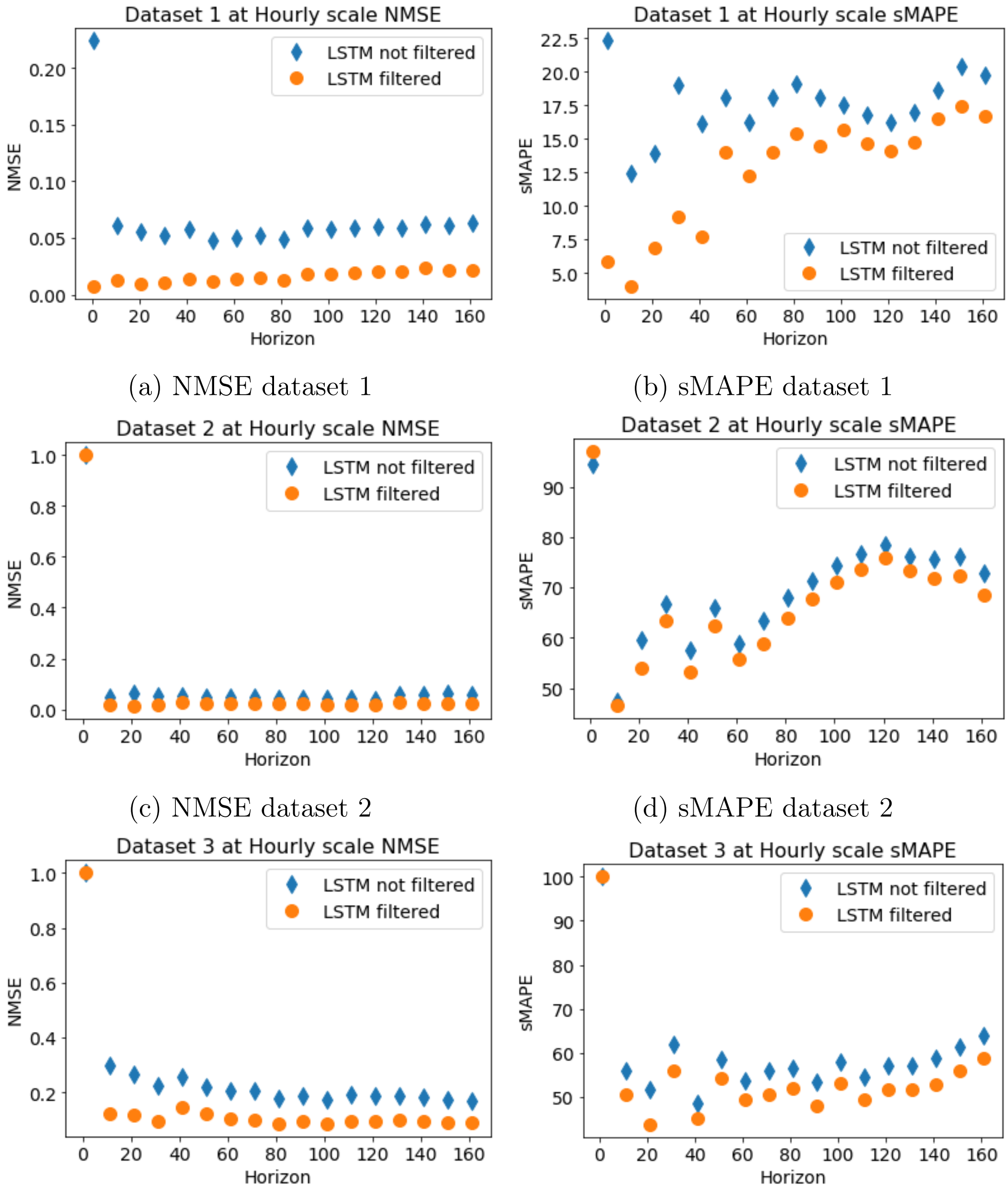
(e) NMSE dataset 3

(f) sMAPE dataset 3

Figure 4.5: On the left, the best performances of each method are reported according to the NMSE for each dataset. The forecast horizon is set at 100 points. The metric is computed every 10 points. On the right sMAPE is used instead.

4.4 LSTM filtered vs LSTM no filtered

LSTM model is found to be very sensitive to the noise. Therefore, it was necessary to filter the noise from the time series out. To filter the noise we used a simple mean filter of size 2. Here, we show the differences between the model tested on filtered signal and the model tested on the actual signal. In all the datasets, filtering the signal improve the performances.



(a) NMSE dataset 1

(b) sMAPE dataset 1

(c) NMSE dataset 2

(d) sMAPE dataset 2

(e) NMSE dataset 3

(f) sMAPE dataset 3

Figure 4.6: From top to the bottom, dataset 1 to dataset 3. The performances of the LSTM and the LSTM trained on filtered data are reported at hourly scale. NMSE and sMAPE are used as reference metrics, on the left and on the right respectively.

4.5 Input length vs models performance

Figures 4.7 and 4.8 illustrate the relationship between the size of the training set, on which model has been fit, and the corresponding performances of the model itself. Each marker indicates the average NMSE for that particular input size.

The color indicates the data set, while the height indicates the quality. The size is kept uniform. Figure 4.7 shows SARIMA models. It can be observed the complete absence of any pattern. The input size does not influence the model quality of the model. Accordingly, we can say that there is no relationship between the quality of the model and the size of the training set.

Figure 4.8 report the same result for the Seasonal ANN and for the LSTM, respectively. It is quite clear that both graph highlight a clear pattern. The quality of the model is improving following a decreasing exponential function. To be mention, is the meaning of the last label on the input axes, referred to as *max*. It indicates the maximum amount of sample available has been exploited to train the neural network. Accordingly, this value differs for each dataset, and the *max* values correspond to the sum of training size and the validation size.

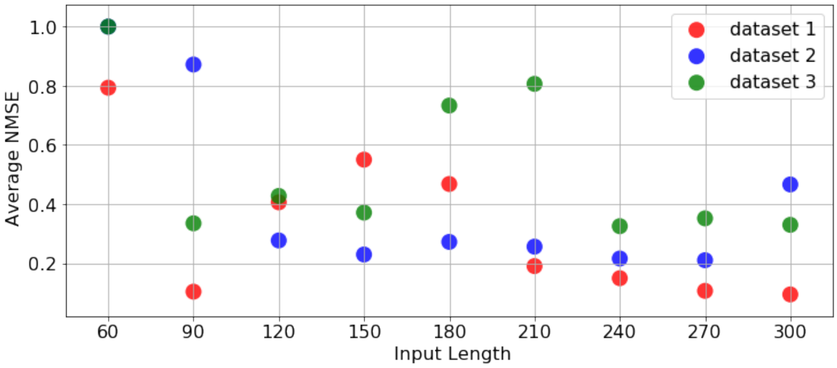
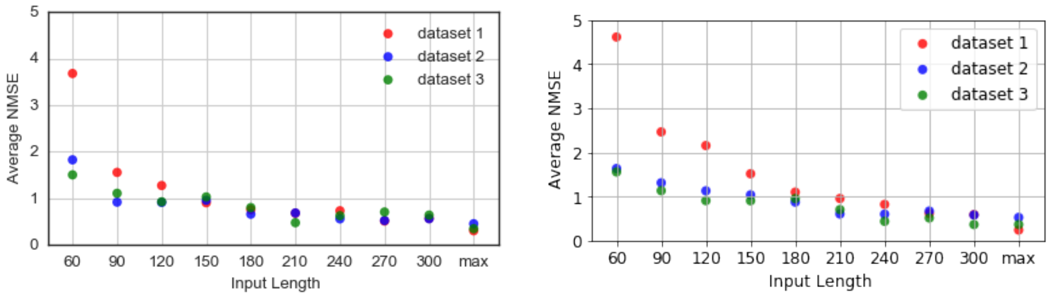


Figure 4.7: On the X axis the size of the training data, on the Y axis the average NMSE computed on 100 points forecast horizon. The average performances are reported for each dataset.

4.5.1 ML techniques

The same result is also reported for the ML techniques. In this case, since it is well known that neural network performances should improve as the dataset is larger, it has been added another input size called, meaning that all the possible sampled have been exploited during the training. This graphs are reported in Figure 4.8.



(a) SANN model

(b) LSTM model

Figure 4.8: Average performances based on NMSE metric, computed on 100 points forecast horizon, for SANN, and LSTM on the left and right, respectively. The average performances are reported for each dataset. $'max'$ indicates the maximum amount of data available, which differs for each dataset and correspond to the size of the training set plus the size of the validation set.

4.6 Statistical evaluation of the results

To assess the statistical relevance of the results we performed different tests. We chose NMSE, since it is the primary metric used in this work. NMSE was collected every 10 points in the future. Since the maximum forecast horizon utilized reaches 100 points at daily scale and 160 points at hourly scale, we have 10 points and 16 points, respectively.

We first state the null hypothesis for the Kruskal-Wallis test defined as:

- H_0 = there is no difference between d1,d2,d3
- H_1= there is difference between d1,d2,d3

For all the tests, we decided to set α equal to 0.05. We evaluated the robustness of each model computing the Kruskal-Wallis test and the KS test. We first evaluate the three different models within each dataset, comparing the NMSE of the three models (SARIMA, SANN, LSTM), then on the 3 datasets together. We observe that in the Kruskal-Wallis test the p-values are below 0.05, which is the reference threshold. Therefore, we can reject with high confidence all the null hypothesis, meaning that the 3 models follow a different distribution within each dataset and also across the three datasets. This statement is true both at daily and hourly scale. P-values are reported on Table 4.10

However, both tests do not provide any information regarding how big is the difference between the evaluated groups. To gain extra information, we decided to exploit the metrics belonging to the "d" family, in particular, Cohen's d and Hedges' g. For the sake of simplicity, in this section we report only the Cohen's d absolute values in Table 4.11. Since the metric evaluates the effect size only between two distributions, we need to compute the metric

Kruskal-Wallis	***Daily scale***	***Hourly scale***
	$p - value$	$p - value$
Dataset1	4.6×10^{-6}	1.4×10^{-6}
Dataset2	2×10^{-3}	8.2×10^{-6}
Dataset3	1.5×10^{-2}	1×10^{-10}
whole Dataset	9.3×10^{-9}	1.8×10^{-9}

Table 4.10: P-values of the Kruskal-Wallis tests of differences between SAR, SANN and LSTM. It is carried out within the single dataset and on the entire dataset, at daily and hourly scale separately.

Cohen's d	***SAR vs SANN***		***SAR vs LSTM***		***SANN vs LSTM***	
	daily	*hourly*	*daily*	*hourly*	*daily*	*hourly*
Dataset1	4.192	1.368	4.316	0.631	1.198	1.361
Dataset2	1.204	1.069	1.352	0.562	0.423	0.310
Dataset3	0.149	2.877	0.237	1.715	0.252	0.771
whole Dataset	0.941	1.166	0.857	0.730	0.044	0.472

Table 4.11: Effect sizes of parwise differences between SAR, SANN and LSTM at daily and hourly scale. The Cohen's d metrics are computed and reported. d stands for classic formulation of Cohen's d metric.

for each possible combination of the three models. Henges's g can be found in the Appendix.

These metrics are quite informative. First of all, at daily scale, we observe the presence of a substantial gap between SARIMA and SANN and LSTM. This means that the SARIMA might be much better or much worse than the other two models. At hourly scale, the gap is smaller, but still, it highlights a big difference between SANN with both SARIMA and SANN.

Afterwards, we compared the similarity between the short-term forecast and long-term forecast. We grouped the first 5 points and the last 5 points of each model from each dataset. Daily and hourly scale are evaluated separately. It turns out the KS test to be at the edge, making it hard to reject the null hypothesis. Table 4.12 summarizes the results for the horizon.

We calculated the "d" metrics of the KS test, to ascertain the presence

Kolmogorov	***Daily scale***			***Hourly scale***		
	SAR	*SANN*	*LSTM*	*SAR*	*SANN*	*LSTM*
whole Dataset	0.091	0.027	0.482	0.093	0.263	0.016

Table 4.12: P-values of the Kolmogorov test evaluating the differences between short and long horizon.

Cohen's d for KS	***Daily scale***	***Hourly scale***
SAR	0.554	0.209
SANN	0.372	0.149
LSTM	0.457	0.520

Table 4.13: Effect size for differences between short and long horizon. The measure has been computed between the short and the long forecast across the whole horizon. Cohen's d is reported.

Means	***Daily scale***			***Hourly scale***		
	SAR	*SANN*	*LSTM*	*SAR*	*SANN*	*LSTM*
Dataset1	**0.094**	0.306	0.240	0.108	**0.025**	0.066
Dataset2	**0.210**	0.456	0.525	0.239	**0.055**	0.105
Dataset3	**0.325**	0.350	0.365	0.623	**0.146**	0.253
whole Dataset	**0.210**	0.371	0.377	0.323	**0.075**	0.141
whole Dataset short	**0.274**	0.350	0.335	0.356	**0.071**	0.193
whole Dataset long	**0.156**	0.388	0.411	0.294	**0.079**	0.096

Table 4.14: Mean values of NMSE at daily and hourly scale. The mean for each method, within each dataset and across the whole dataset is calculated. The bold numbers highlight the best results.

of a relevant gap reported in Table 4.13. These values quantify the actual difference between the short and the long term.

At daily scale SARIMA and LSTM show a medium effect size, while SANN shows a small effect size. At hourly scale, the effect size is not evident. SANN is below 0.2, leading us to the conclusion that the short-term and the long-term follow the same distribution. On the other hand, the large d values indicate SARIMA's incapability in producing a dynamic forecast, and in learning the patter over time.

The d metrics have revealed the presence of a gap between the different distribution. In order to understand, if the gap is positive or negative, and consequently, to assess the model with the best performance, we computed the means of the performances, which are summarized in Table 4.14.

At daily scale, it emerges that SARIMA is the most robust method since it has the lowest mean values on all the possible scenarios. SANN and LSTM, instead, are far from SARIMA performances as highlighted from the "d" metrics and the means values. At hourly scale, SARIMA is the less robust. It resulted in the worst performances, with the highest mean. Judging the NMSE, SANN provides a more robust forecast compared to the LSTM. Anyway, the gap is small. "d" metrics are around 0.5, that can be interpreted as a slight difference. Nevertheless, when we computed the mean we realize that the difference is small but still relevant. In some cases, the SANN mean is twice lower than LSTM mean.

Chapter 5

Discussion

In this Chapter, we discuss the results presented in Chapter 4. The purpose is to examine neural networks capability in producing accurate and reliable forecast exploiting only raw data. This research does not map the patterns with a mathematical formula, but it only provides general guidelines and suggestions to build a valuable and effective model.

Below, research questions are answered.

- **Which of the methods selected for comparison, i.e. Seasonal ANN and LSTM, deliver the most accurate performance with respect to SARIMA?**

In this report, machine learning models have shown to be able, to some extent, to model and forecast the patterns of call volumes both at daily and at hourly scale. At daily scale, we observed that SARIMA outperforms both SANN and LSTM across all datasets, as shown in Figure 4.4. This outcome is due to the linearity of the patterns; indeed, daily data are ruled by a strong seasonality with a linear behavior. Besides, the limited amount of samples do not help. Neural networks are deemed to need a large amount of data to learn the patterns behind them. From an accuracy point of view, in the first dataset SARIMA reaches an sMAPE of 2, while SANN and LSTM do not go below an sMAPE of 4. Nevertheless, SARIMA outperforms ML techniques, but the gap is minimal. However, this measure is not extremely accurate. If we look at the actual forecast we may notice that the forecast is not as good as expected. It might be due to different reasons, but in this case sMAPE is not reliable. The second dataset is the one with the worse performance. It is, indeed, the most challenging scenario, where less than 400 points are available. Anyway, SARIMA reaches an overall return of 18 sMAPE, while ML techniques are above 40. In this case, if we look at the actual forecast, we observe the sMAPE correctly expresses the accuracy of the forecast. It happens the same in the third dataset, where the sMAPE values do not go below 24. If we look at the real forecast, it can be considered reasonable.

At hourly scale, SANN and LSTM perform better with respect to SARIMA. This outcome is due to the large amount of samples available, always above 10000, which allows the networks to show their potential. Not only they tracked the amplitude of the dataset almost entirely, but they are capable of catching and model both the inter-day and intra-day seasonality. Some examples are reported in Appendix A.1. Moreover, they do not require any prepossessing. Indeed, the networks are trained using raw data, which is already a significant simplification with respect to SARIMA, which needs the data to be stationary. Compare to SARIMA, both SANN and LSTM reaches higher accuracy. The only exception is the second data set, where the outcomes of all the three models are very close.

On the first dataset, it manages to stay above the 80% of accuracy. Regarding the second and the third dataset, we notice bad performances. However, if we observe the actual forecast, reported in appendix A.1, we can see that the quality of the forecast is good. This mismatch is explained by the presence of multiple zeros on the time series. In this case, when the forecast is slightly above zero the sMAPE is highly influenced. A possible solution to this problem would be to add a constant. There are others possible way to tackle this problem, but it is out of the scope of this thesis.

To be noted is that the LSTM performances were evaluated on the non-filtered data. When it comes at filtered data, LSTM outperforms SANN in all datasets. The rationale behind the filtering helps the network to figure out the general trend and to avoid to get stuck because of the noise. LSTM was proven to be sensitive to the noise data as stated in [49] [50].

Regarding the robustness of the models, we see that the NMSE at daily scale is quite stable for all the models. To evaluate the results from a statistical point of view, we carried out the Kruskal-Wallis test. It turns out that the distribution of the models are different for each dataset. The difference between each model for each dataset is computed with the Cohen's d metric. The gap between SARIMA and the network is relevant and positive, leading us to the conclusion that SARIMA has better performances across the three datasets.

On the other hand, at hourly scale, SARIMA is the less robust. THe gap with the other two models is negative, since it resulted in the worst performances, with the highest mean. Judging the NSME, SANN provides a more robust forecast compared to the LSTM. Anyway, the gap is small. "d" metrics are around 0.5, that can be interpreted as a slight difference. Besides, when computed the mean we realize that the difference is small but still relevant. In some cases, the SANN mean is twice lower than LSTM mean.

To sum up, at daily scale SARIMA has shown better performance. A decisive factor is the sample size of the dataset. With a dataset of 400 points, it is quite hard to apply a machine learning algorithms effectively. Even 3000 points do not seem to be enough to represent the problem entirely. Hence, we conclude that machine learning cannot be deployed on daily scale exploiting only raw data. Some other techniques have to be utilized or tested.

At hourly scale, SANN has shown slightly better performances with respect to SANN. Moreover, it is much faster in training, and it does not require the signal to be smooth. However, when it comes to the preprocessed signal, LSTM score better than the SANN. LSTM has been empirically found to be quite sensitive to the noise. Moreover, also the number of epoch played an important role. In some cases, the network overfits data, showing excellent results on the training set, but very poor on the validation set and testing set. Furthermore, those forecasts are exploited to arrange the schedule of large companies. Hence, the level of accuracy required is not high, since the goal is to stay above the 80% and to provide a general idea on how the workload will evolve in the next future. Nevertheless, the robustness of the model is quite relevant since it has to perform well in many different scenarios. Likely, this is the case both for SARIMA at daily scale and SANN at hourly scale. We observed that in each scenario considered those model resulted in the best performance.

- **How is machine learning performance at different forecast horizon?**

To answer this question, we decided to compare the NMSE metric on the first 5 points against the last 5 points of each model across the whole dataset. Daily and hourly scale are evaluated separately. We computed the KS test, which ends up to be at the edge, making it hard to reject the null hypothesis. In this case, we perform calculated the "d" metrics, to ascertain the presence of a relevant gap.

At daily scale SARIMA and LSTM show a medium effect size, while SANN shows a small effect size. Looking at the graphs, both SANN and LSTM do not show any trend. Both the sMAPE and the NMSE do not improve or get worse. It is because a DirMo strategy is implemented. Both LSTM and SANN can model forecast in the future without decreasing the performances since they predict just 7 points at the time. In the second dataset, SARIMA seems to be more efficient in guessing the trend of the data, while SANN and LSTM are not. Moreover, we can note that in general, the performances between the SANN and LSTM are very similar meaning that probably they are staked in a local minimum. It may happen when the data are not large enough to adequately represent the problem.

At hourly scale, the effect size is not evident. Only for SARIMA the d metric highlights a negative gap between the short-term and the long-term and the incapability of SARIMA model in producing a dynamic forecast, and in learning the patter over time. If we look at the graphs we notice how after 100 points in the future SARIMA performances decreases. Both SANN and LSTM results are quite impressive. Overall, we can affirm that the model produces the same error along the forecast horizon. It is another evidence that the models are quite robust. In this case, SANN is good given the effect size is almost negligible.

Anyway, it becomes harder and harder to forecast the future based only on the current samples. Perhaps, if we increased more the size of the network, we would observe a drop in the quality of the forecast.

Curios are the fact that LSTM and SANN forecast seem to follow the same trend with different amplitude. This is very clear looking at Figure 4.5.

- **Which is the relationship between the size of the training set and the quality of the models?**

Regarding the statistical method, SARIMA has shown no correlation between the input size and the quality of the model provided, meaning that a model fitted on 90 points may perform better than a model fitted on 300, as stated in Table 4.6. This behavior is justified by the fact that outliers can hardly influence SARIMA model. However, the handling of the outliers is out of the scope of this research. This behavior forces you to apply a grid search technique to find the best parameters. Grid search requires a long time to be performed, and it has to be carried out each time the model has to change. Besides, SARIMA has 7 parameters meaning that the number of the possible combinations may explode becoming unfeasible to explore. Moreover, it cannot exploit all the points since the grid search would require a prohibitive time when the input size raises above 400 points.

On the other side, machine learning techniques have shown to improve their performances along with the increase of input size. In Figure 4.8 we can observe how the performance follow a decreasing trend. Therefore, they may outperform statistical methods when a sufficient amount of data is available.

5.1 Comparison to the related work

In literature, there are few works about time series algorithms, which are tested on large forecasting horizons and real-world time series at the same time. For this reason, a complete comparison is difficult. In terms of architecture, our work can be compared with the one proposed by [40], where the SANN has an average MAPE of 3. However, in this case only 10 points in the future are forecast. Besides, no real-word time series were investigated. Also the work proposed by [17] reported similar results, where the SANN has an average MAPE of 7.5. Even this work has a limited forecasting horizon of 8 samples and no real-word time series were explored. In terms of evaluation metric, our work is similar to [51]. Among all the time series utilized in this work, the most similar to ours is the "Behaviour of sunspots", where MLP obtained an NMSE around 0.1, which is comparable with our results. However, the forecast horizon was limited to 4. The work proposed by [52], where neural networks and genetic algorithms are combined. The proposed architecture scores, on the well-known time series as "Passegers"and "Mackey-Glass", a sMAPE of 3.22 and 1.81. Finally, th work proposed by [53] shows very similar performance a modified LSTM architecture.

5.2 Relevant achievements

Building a Neural Network capable to forecast time series is very time consuming. The architecture is quite complex and there are many parameters that have to be tuned. The typical approach adopted is the trial and error.

The most relevant outcome of this work is the simplified architecture created by the combination of the approach suggested by [40] and the new architecture, known as LSTM. Besides, we provided a way to embed the information obtained through the temporal exploration into the network. The suggested approach, where the input and output layers' size of the LSTM are fixed, was successfully applied to three different datasets, proving better performances compared to the SANN. The constrains imposed to the size of the layer allow to speed up the process of design the network.

The second notable outcome is the large forecast horizon utilized during the tests. Most of the work available in the literature are provide results on 10 of 20 points in the future, while in this work we have reached a forecast horizon of 100 and 160 points at daily and hourly scale, respectively. Our architecture provided a reasonable forecast highlighting an high robustness of this architecture.

Finally, we performed all the experiments on real world scenario time-series, which give more credit to our statements.

5.3 Sustainability Ethical Implications of this Research

This research used anonymous data coming from different call centers. Teleopti gives its consensus to use of data to improve their forecasting capability and to explore the machine learning capabilities in modeling time series with multiple seasonalities. The forecasting aims at improving the schedule of the employees, increasing the flexibility of the company in reorganizing the schedule. We hope that this helps to enhance the average quality of work in the call centers.

5.4 Future Work

In this Section, we want to report some of the trials made that yields to poor results or that showed some improvements, but we had no time to explore giving rise to possible future works.

Regarding the statistical method, SARIMA, since it does not require a large amount of data, we tried to improve the performances would be to cluster the data. The rationale behind it resides on an intrinsic limitation of SARIMA to model only one seasonality. Therefore, we have exploited KST to infer, which days of the week followed the same distribution. Then, we rearrange the data according to the number of delivery identified. All the days that may be described by that distribution are combined. A model is fitted on each cluster, and for each of them, the forecast is computed separately and then combined at the end. This procedure increases the complexity of the system, but at the same time, it may increase both the accuracy and the possible forecast horizon. A test was carried out with the SARIMA model, and it was successful. The same approach may be applied to machine learning techniques.

Regarding ML techniques, some possible improvements have been tested. A dummy variable was added as a feature with the purpose to indicates the day of the week. We hypothesize that it may have helped the model to infer

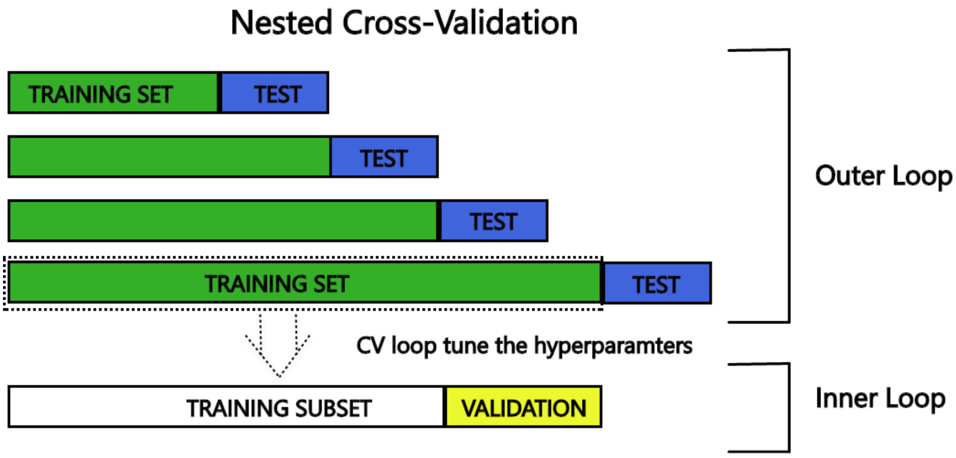


Figure 5.1: Example of how is it performed a nested cross validation

the seasonality. However, it did not help at all. The model results in worse performance than before.

A successive attempt was carried out, adding some extra feature based on a decreasing exponential function, adequately normalized. The purpose was to increase the window of the model since now it is bounded to the size of the seasonality. Adding this extra feature should provide the model with some additional information about the past. These techniques show some little improvements. However, it was much more convenient increasing the window size of the input. It would be interesting to examine more in depth to see where it the trade-off and when increasing the size of the time window becomes less convenient than adding the exponential functions.

Last but not least, it would be interesting to explore, which is the most extended horizon that machine learning techniques can forecast before collapsing in some meaningless forecast.

5.5 General Consideration and limitation's outline

Forecasting time series is considered a hard task. For a matter of time and limited computational power, it was not possible to test all the possible approaches. Moreover, a powerful computer may lead to better performance since the data could be fir entirely on the RAM.

First of all, I have only used raw values to feed the network. Any king of prepossessing techniques was applied, apart from the normalization of the data in the range between -1 and 1. No external variable was available, which limited the study only on the signal itself. Only, a hand-made external variable was validated with poor results. The investigation of the external variable is essential to, and they can lead to significant performance improvement.

As mentioned in Section 3.1, the outlier was not taken into account, since their number was quite small for all the datasets. However, in some other cases, the dataset could show the presence of multiple outliers. In that case, a strategy to detect and change should be implemented. A super vector machine might be a good candidate.

Moreover, the distribution of the data was not deeply explored.

Finally, we are aware that the validation method applied is not the most robust. Hence, a shortcoming is the arbitrary choice of the test set. The model was tested only on the last samples. We decided to implement an hold-out crossing validation process. To improve the reliability of our measures, we may have implemented nested cross-validation strategy. An example is presented in Figure 5.1. However, it has not implemented for two reasons. Firstly, at daily scale, it does not make sense since the data were not enough to model the problem. Secondly, at hourly scale models have shown consistent results after 10000 points, and only two datasets out of three have enough samples. Moreover, the simulations take too much time to be performed on a laptop as discussed in Section 1.3.

Chapter 6

Conclusions

In this thesis, we have analyzed and compared SARIMA, SANN, and LSTM performances on three different dataset, collected from different Call centers, both at daily and time interval. We concluded that on daily scale, both machine learning techniques are not able to outperform SARIMA. Most likely, it is due to the lack of samples. On hourly level, Machine learning outperforms the SARIMA in all dataset. They learned to predict not only the intraday but also the inter day seasonality, showing high flexibility across the different datasets. The performance of both SANN and LSTM are very similar, so we conclude that they are equally capable of modeling incoming call time series.

We also conclude that for SARIMA, there is no correlation between the quality of the model and the size of the training set. On the other side, we conclude that both for SANN and LSTM, there is a decreasing exponential correlation between the number of samples and the quality of the model.

Finally, we observed that machine learning is good at forecasting in the short term, also at a daily level. At hourly scale, we reached till 160 points in the future, but we conclude that the model is robust enough to allow a more extended forecast.

Machine learning have been explored only recently as a possible alternative to the classical forecasting technique as SARIMA, [4] [5]. This project provides a better understating on the capability of machine learning in forecasting call volume, where there is still little knowledge, causing a work overload in some cases due to under-staffing. Moreover, only a few projects have investigated a long forecast as 100 and 160 points in the future. The main result is the evidence that machine learning can provide valuable results on forecasting call load time series when the amount of data are sufficiently large. Consequently, it leads to better management of the agents by improving the working environment.

Appendix A

Appendix

A.1 Actual forecast

A.1.1 LSTM

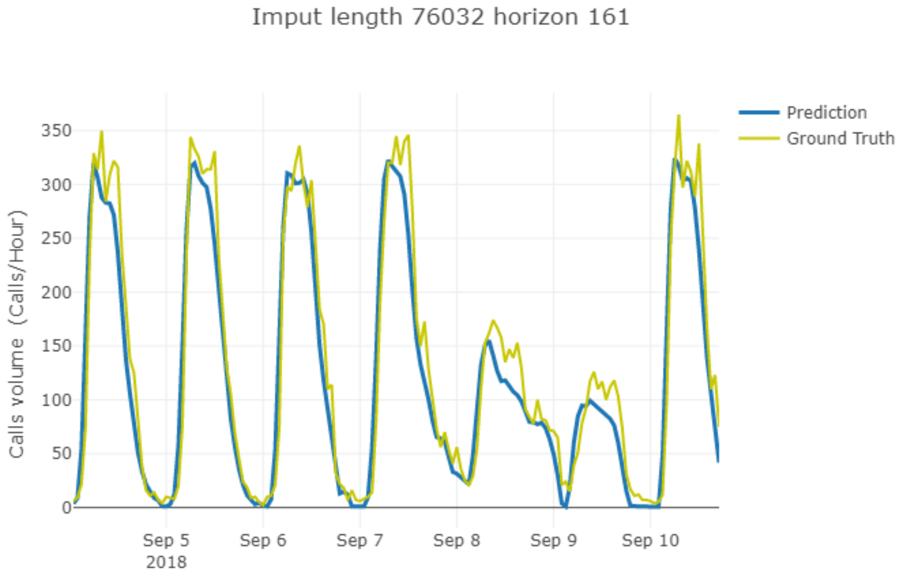


Figure A.1: Actual forecast for the LSTM on hourly scale on the dataset 1

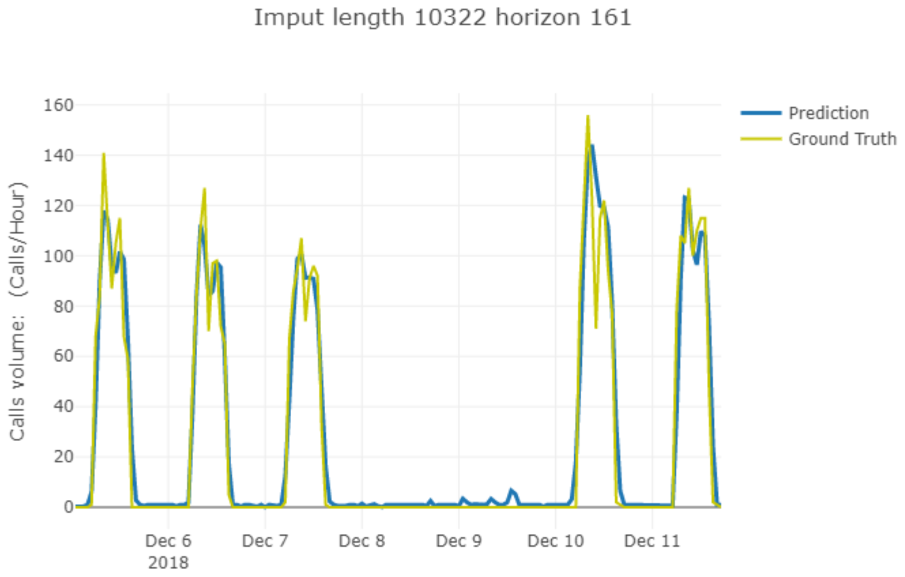


Figure A.2: Actual forecast for the LSTM on hourly scale on the dataset 2

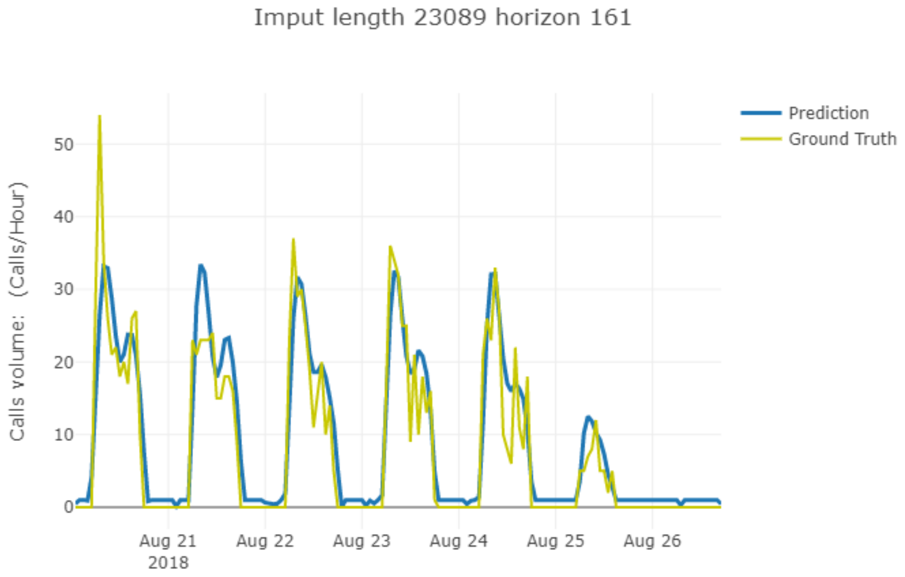


Figure A.3: Actual forecast for the LSTM on hourly scale on the dataset 3

A.1.2 SANN

A.2 Residual Example

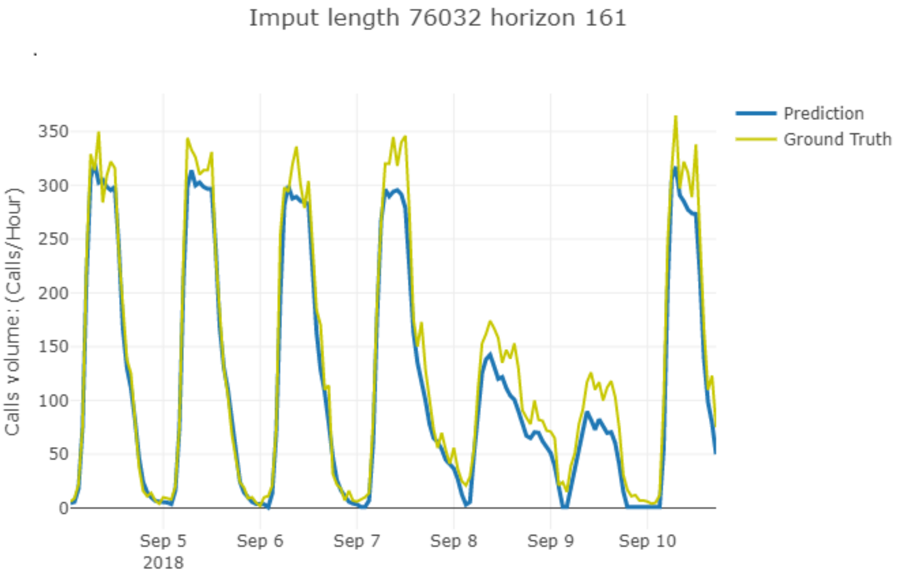


Figure A.4: Actual forecast for the SANN on hourly scale on the dataset 1

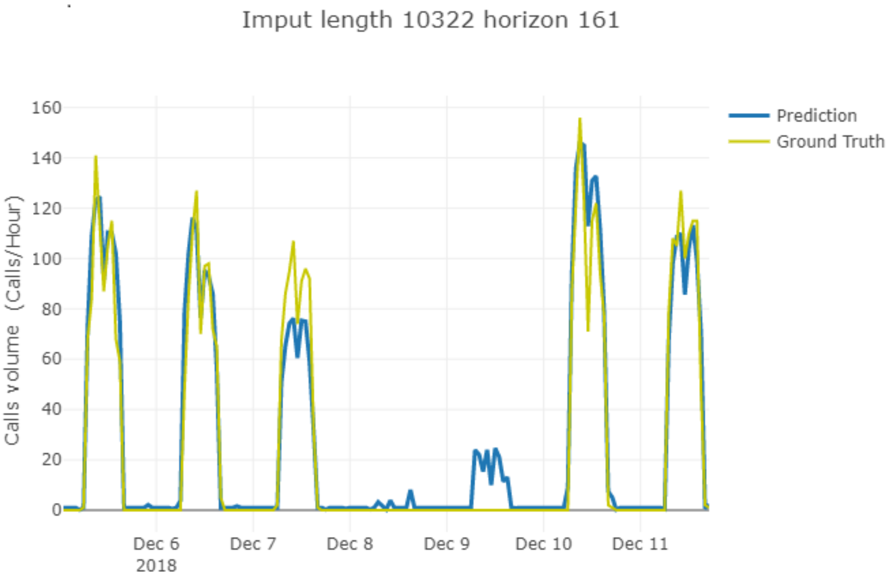


Figure A.5: Actual forecast for the SANN on hourly scale on the dataset 1

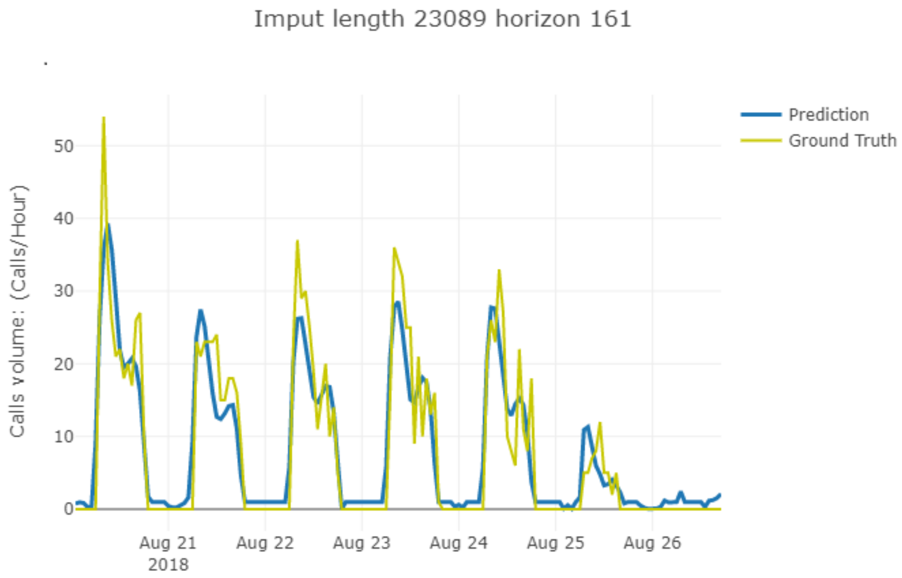


Figure A.6: Actual forecast for the LSTM on hourly scale on the dataset 1

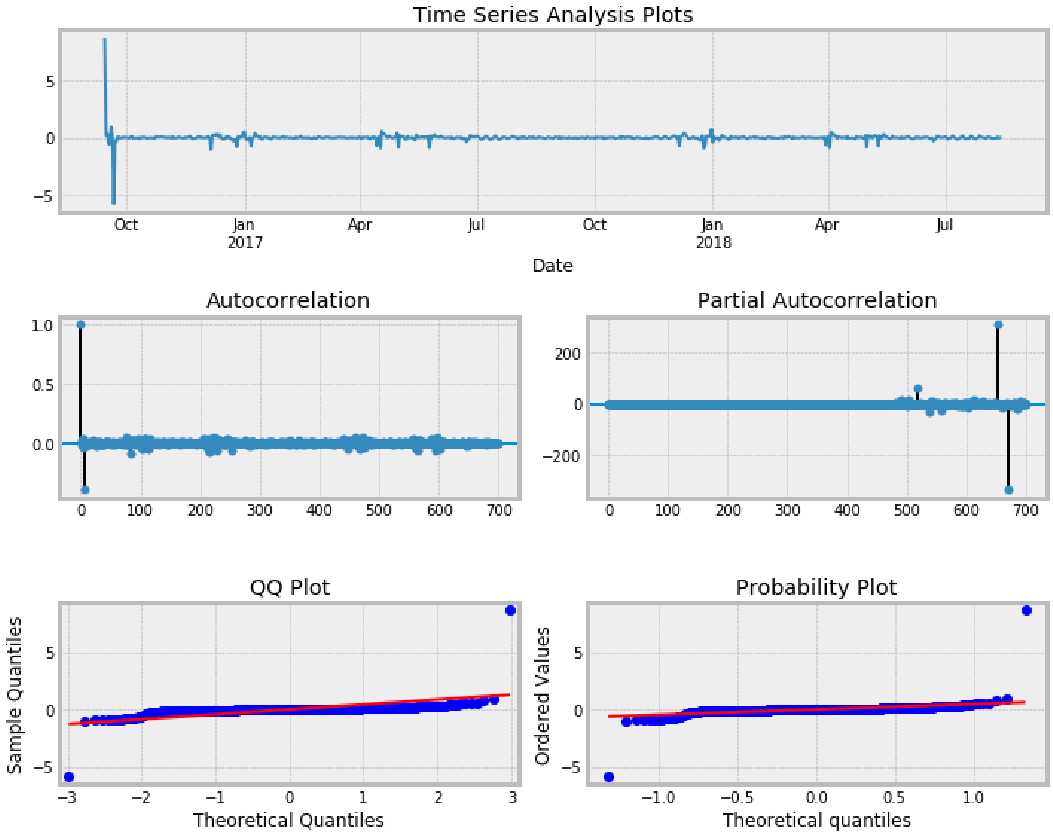
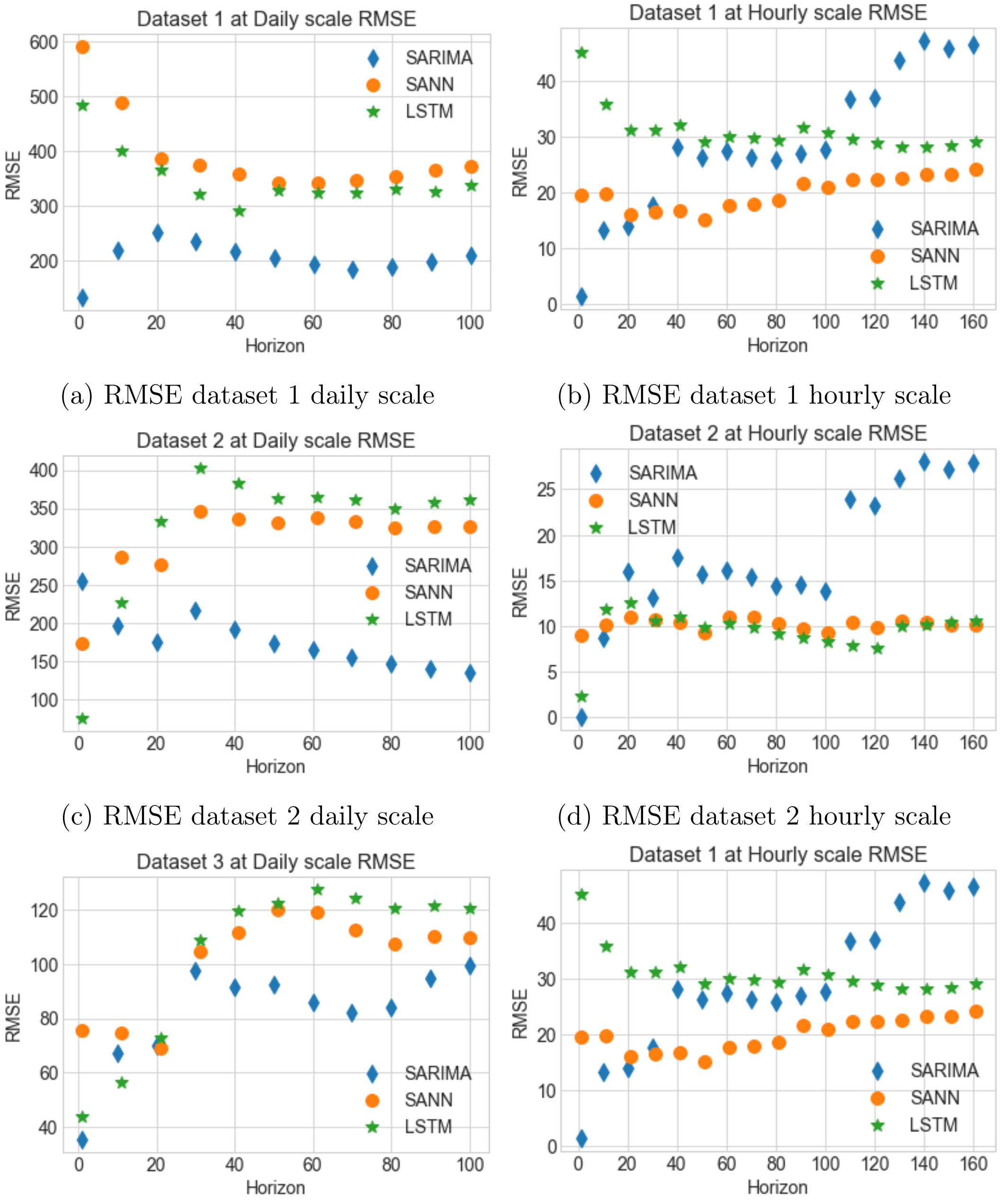


Figure A.7: Example of the residual of a SARIMA model. They follow the normal distribution therefore there is much less information left.

A.3 Error metrics overview



(a) RMSE dataset 1 daily scale

(b) RMSE dataset 1 hourly scale

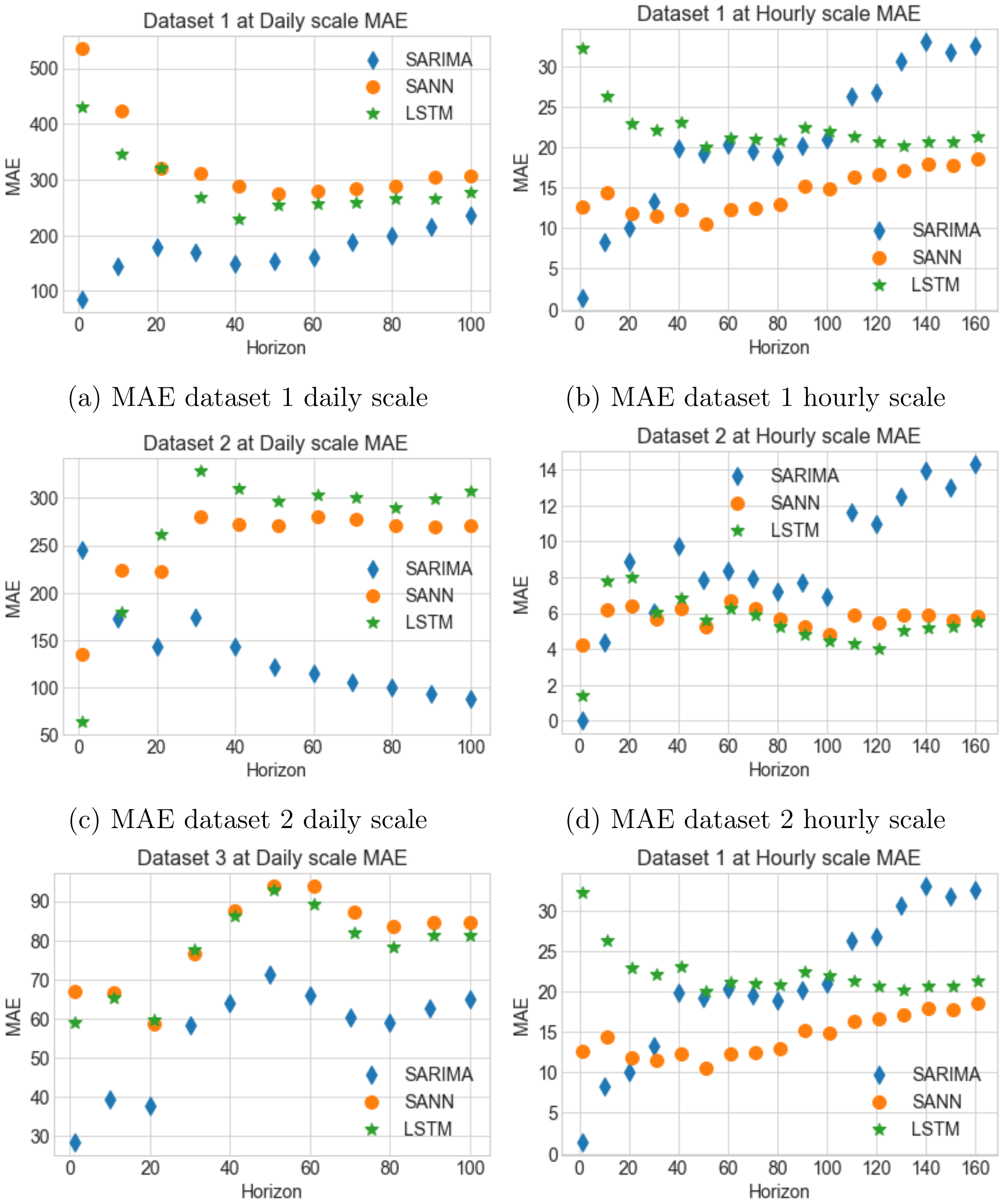
(c) RMSE dataset 2 daily scale

(d) RMSE dataset 2 hourly scale

(e) RMSE dataset 3 daily scale

(f) RMSE dataset 3 hourly scale

Figure A.8: RMSE metric summary. On the left the daily dataset, on the right the hourly dataset.



(a) MAE dataset 1 daily scale

(b) MAE dataset 1 hourly scale

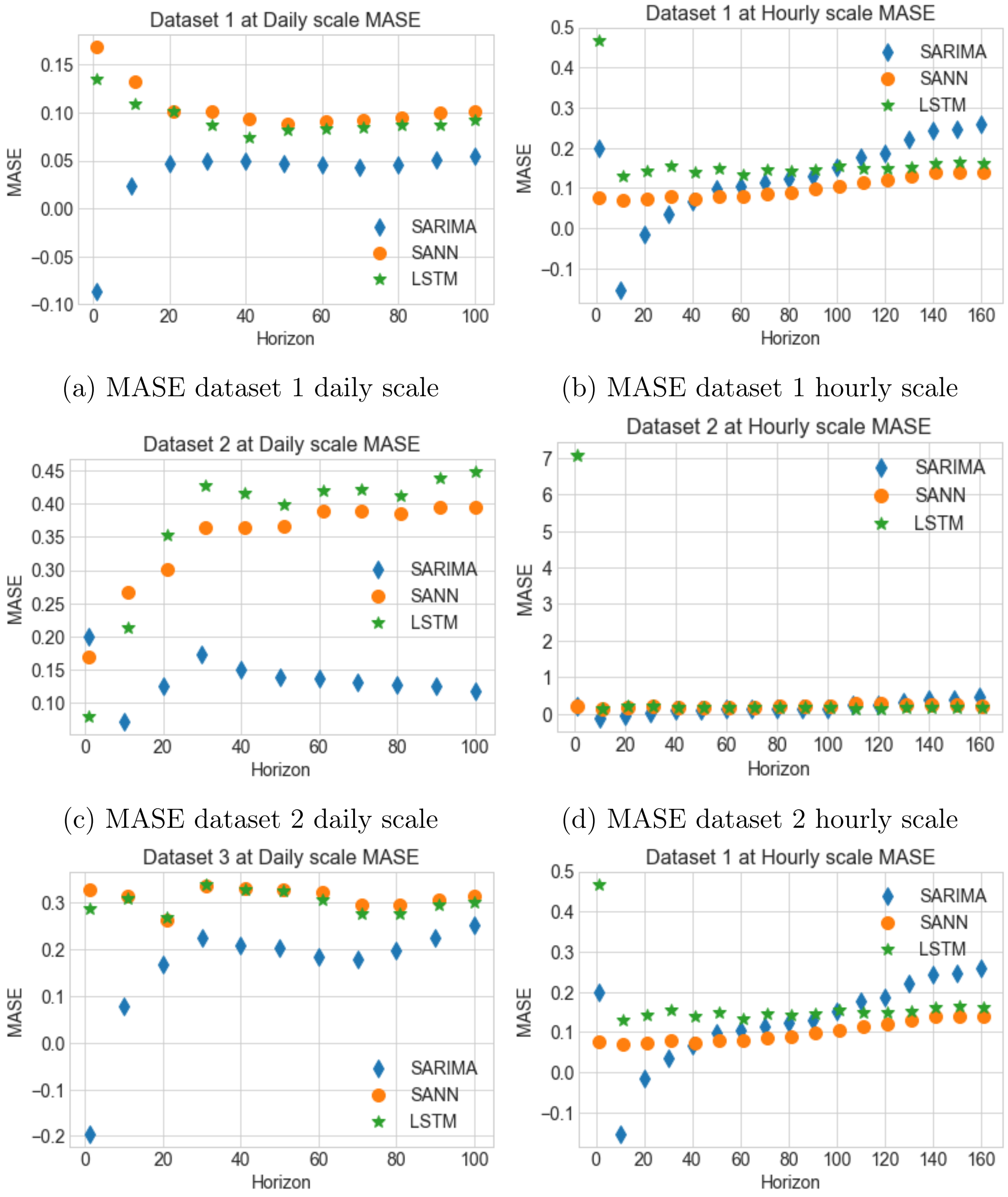
(c) MAE dataset 2 daily scale

(d) MAE dataset 2 hourly scale

(e) MAE dataset 3 daily scale

(f) MAE dataset 3 hourly scale

Figure A.9: MAE metric summary. On the left the daily dataset, on the right the hourly dataset.



(a) MASE dataset 1 daily scale

(b) MASE dataset 1 hourly scale

(c) MASE dataset 2 daily scale

(d) MASE dataset 2 hourly scale

(e) MASE dataset 3 daily scale

(f) MASE dataset 3 hourly scale

Figure A.10: MASE metric summary. On the left the daily dataset, on the right the hourly dataset.

A.4 Complete table of statistical tests

Kruskal-Wallis	***Daily scale***		***Hourly scale***	
	H	$p-value$	H	$p-value$
Dataset1	24.542537676	0.000004684	26.836305563	0.000001488
Dataset2	12.421973748	0.002007256	23.411093928	0.000008248
Dataset3	8.338356830	0.015464961	42.311913896	0.000000001
whole Dataset	32.383010183	0.000000093	35.650033930	0.000000018

Table A.1: Kruskal-Wallis test of differences between SAR, SANN and LSTM. H is the test statistic which has to be compared with the critical Chi-values. I t carried out within the single dataset and on the entire dataset, at daily and hourly scale separately.

Cohen's d	***SAR vs SANN***		***SAR vs LSTM***		***SANN vs LSTM***	
	d	g	d	g	d	g
Dataset1	4.1929	5.5905	4.3165	5.7553	1.1983	1.5977
Dataset2	1.2041	1.6055	1.3527	1.8036	0.4238	0.5650
Dataset3	0.1494	0.1992	0.2371	0.3161	0.2520	0.3360
whole Dataset	0.9419	1.2558	0.8570	1.1427	0.0449	0.0599

Table A.2: Effect sizes of parwise differences between SAR, SANN and LSTM at daily scale. The Cohen d metrics are computed and reported. d stands for classic formulation of Cohen's d metric. g stands for the unbiased version called Henges'g

Cohen's d	***SAR vs SANN***		***SAR vs LSTM***		***SANN vs LSTM***	
	d	g	d	g	d	g
Dataset1	1.3683	1.8243	0.6319	0.8425	1.3618	1.8158
Dataset2	1.0694	1.4258	0.5626	0.7501	0.3100	0.4133
Dataset3	2.8770	3.8361	1.7151	2.2868	0.7712	1.0283
whole Dataset	1.1668	1.5557	0.7306	0.9742	0.4720	0.6293

Table A.3: Effect sizes of parwise differences between SAR, SANN and LSTM at hourly scale. The Cohen d metrics are computed and reported. d stands for classic formulation of Cohen's d metric. g stands for the unbiased version called Henges'g

Kolmogorov	***Daily scale***			***Hourly scale***		
	SAR	$SANN$	$LSTM$	SAR	$SANN$	$LSTM$
whole Dataset	0.0919	0.0227	0.4828	0.0931	0.2633	0.0166

Table A.4: P-values of the Kolmogorov test evaluating the differences between short and long horizon.

Cohen's d for KS	***Daily scale***		***Hourly scale***	
	d	*h*	*d*	*h*
SAR	0.5547	0.6340	0.2092	0.2391
SANN	0.3728	0.4261	0.1499	0.1714
LSTM	0.4572	0.5225	0.5202	0.5945

Table A.5: Effect size for differences between short and long horizon. The measure has been computed between the short and the long forecast across the whole horizon. Both Cohen's d and henges'g are reported.

Means	***Daily scale***			***Hourly scale***		
	SAR	*SANN*	*LSTM*	*SAR*	*SANN*	*LSTM*
Dataset1	**0.09454**	0.30633	0.24059	0.10866	**0.02569**	0.06648
Dataset2	**0.21023**	0.45660	0.52542	0.23907	**0.05531**	0.10588
Dataset3	**0.32530**	0.35026	0.36599	0.62381	**0.14660**	0.25351
whole Dataset	**0.21002**	0.37107	0.37733	0.32384	**0.07587**	0.14196
whole Dataset short	**0.27472**	0.35020	0.33581	0.35676	**0.07164**	0.19327
whole Dataset long	**0.15611**	0.38846	0.41193	0.29459	**0.07963**	0.09634

Table A.6: Mean values of NMSE at daily and hourly scale. The mean for each method, within each dataset and across the whole dataset is calculated. The bold numbers highlight the best results.

Bibliography

[1] R. Ibrahim, H. Ye, P. L'Ecuyer, and H. Shen, "Modeling and forecasting call center arrivals: A literature survey and a case study," *International Journal of Forecasting*, vol. 32, no. 3, pp. 865–874, 2016.

[2] T. Hill, M. O'Connor, and W. Remus, "Neural network models for time series forecasts," *Management science*, vol. 42, no. 7, pp. 1082–1092, 1996.

[3] F. Tseng, G. Tzeng, and H. Yu, "Combining neural network with seasonal time series arima model to forecast the total production value of taiwan mechanical industry," *Technological Forecasting and Social Change*, vol. 65, pp. 1–17, 2000.

[4] M. Cai, M. Pipattanasomporn, and S. Rahman, "Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques," *Applied Energy*, vol. 236, pp. 1078–1088, 2019.

[5] O. Klein, F. Kanter, H. Kulbe, P. Jank, C. Denkert, G. Nebrich, W. D. Schmitt, Z. Wu, C. A. Kunze, J. Sehouli, *et al.*, "Maldi-imaging for classification of epithelial ovarian cancer histotypes from a tissue microarray using machine learning methods," *PROTEOMICS–Clinical Applications*, vol. 13, no. 1, p. 1700181, 2019.

[6] S. McNally, J. Roche, and S. Caton, "Predicting the price of bitcoin using machine learning," in *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pp. 339–343, March 2018.

[7] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv preprint arXiv:1701.01887*, 2017.

[8] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric Reviews*, vol. 29, no. 5-6, pp. 594–621, 2010.

[9] J. Ripa and R. Cheaib, "Real-time call center call monitoring and analysis," Dec. 13 2016. US Patent 9,521,258.

[10] R. A. Feinberg, I.-S. Kim, L. Hokama, K. De Ruyter, and C. Keen, "Operational determinants of caller satisfaction in the call center," *International Journal of Service Industry Management*, vol. 11, no. 2, pp. 131–141, 2000.

[11] S. Bagnara, E. Donati, and T. Schael, "Call & contact center," *Il Sole*, vol. 24, 2002.

[12] G. Koole, *Call center optimization.* Lulu. com, 2013.

[13] C. Chatfield, *The analysis of time series: an introduction.* Chapman and Hall/CRC, 2016.

[14] B. Farley and W. Clark, "Simulation of self-organizing systems by digital computer," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, pp. 76–84, Sep. 1954.

[15] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, pp. 68–75, 3 2017.

[16] A. Graves, "Supervised sequence labelling," in *Supervised sequence labelling with recurrent neural networks*, pp. 5–13, Springer, 2012.

[17] R. Adhikari and R. K. Agrawal, "Forecasting strong seasonal time series with artificial neural networks," *Journal of Scientific and Industrial Research*, vol. 71, no. 10, pp. 657–666, 2012.

[18] G. Shmueli and K. C. Lichtendahl Jr, *Practical Time Series Forecasting with R: A Hands-On Guide.* Axelrod Schnall Publishers, 2016.

[19] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control.* John Wiley & Sons, 2015.

[20] C. Chatfield and M. Yar, "Holt-winters forecasting: some practical issues," *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 37, no. 2, pp. 129–140, 1988.

[21] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice.* OTexts, 2018.

[22] B. G. E. P. . J. G. M, "time series analysis: Forecasting and control, 3rd edn," *Holden-Day, California*, vol. 24, no. 7, 1970.

[23] R. F. Engle, "Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation," *Econometrica: Journal of the Econometric Society*, pp. 987–1007, 1982.

[24] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[25] R. Rojas, *Neural networks: a systematic introduction.* Springer Science & Business Media, 2013.

[26] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," tech. rep., Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[27] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, “Learning representations by back-propagating errors,” *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[28] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[29] R. Cascade-correlation and N. S. Chunking, “1997Hochreiter_LSTM,” vol. 9, no. 8, pp. 1–32, 1997.

[30] O. Aksin, M. Armony, and V. Mehrotram, “The modern call-center: A multi-disciplinary perspective on operations management research, production and operations management,” *Special Issue on Service Operations in Honor of John Buzacott*, vol. 16, pp. 655–688, 01 2007.

[31] L. D Brown, N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn, and L. Zhao, “Statistical analysis of a telephone call center: A queueing-science perspective,” *Journal of the American Statistical Association*, vol. 100, pp. 36–50, 02 2005.

[32] N. Kourentzes and S. F. Crone, “Frequency independent automatic input variable selection for neural networks for forecasting,” in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2010.

[33] A. M. De Livera, R. Hyndman, and R. D. Snyder, “Forecasting time series with complex seasonal patterns using exponential smoothing,” *Journal of the American Statistical Association*, vol. 106, pp. 1513–1527, 01 2010.

[34] D. Barrow and N. Kourentzes, “The impact of special days in call arrivals forecasting: A neural network approach to modelling special days,” *European Journal of Operational Research*, vol. 264, no. 3, pp. 967–977, 2018.

[35] K. W. Hipel and A. I. McLeod, *Time series modelling of water resources and environmental systems*, vol. 45. Elsevier, 1994.

[36] K.-j. Kim, “Financial time series forecasting using support vector machines,” *Neurocomputing*, vol. 55, no. 1-2, pp. 307–319, 2003.

[37] E. Ghysels, C. W. Granger, and P. L. Siklos, “Is seasonal adjustment a linear or nonlinear data-filtering process?,” *Journal of Business & Economic Statistics*, vol. 14, no. 3, pp. 374–386, 1996.

[38] G. P. Zhang, “Time series forecasting using a hybrid arima and neural network model,” *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[39] J. Kamruzzaman, *Artificial neural networks in finance and manufacturing*. IGI Global, 2006.

[40] C. Hamzaçebi, “Improving artificial neural networks’ performance in seasonal time series forecasting,” *Inf. Sci.*, vol. 178, pp. 4550–4559, Dec. 2008.

[41] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” 1999.

[42] R. Dahnke, “Diplomarbeit im Fach Informatik,” *Psychiatrie und Psychotherapie*, 2008.

[43] P. F. Y. Bengio, P. Simard, “Learning long term dependencies with gradient descent is difficult,” *IEEE transasctions of neural network*, vol. 5, no. 2, 1994.

[44] Bontempi Gianluca, “Long term time series prediction with multi-input multi-output local learning,” 2008.

[45] D. M. Kline, “Methods for Multi-Step Time Series Forecasting Neural Networks,” in *Neural Networks in Business Forecasting*, pp. 226–250, IGI Global.

[46] E. Mahmoud, “Accuracy in forecasting: A survey,” *Journal of Forecasting*, vol. 3, no. 2, pp. 139–159, 1984.

[47] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky, and V. A. Kamaev, “A survey of forecast error measures,” *World Applied Sciences Journal*, vol. 24, no. 24, pp. 171–176, 2013.

[48] C. G. Atkeson, A. W. Moore, and S. Schaal, “Locally Weighted Learning,” *Artificial Intelligence Review*, vol. 11, no. 1/5, pp. 11–73, 1997.

[49] A. Rusiecki, M. Kordos, T. Kamiński, and K. Greń, “Training neural networks on noisy data,” in *International Conference on Artificial Intelligence and Soft Computing*, pp. 131–142, Springer, 2014.

[50] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, IEEE, 2017.

[51] T. Koskela, M. Lehtokangas, J. Saarinen, and K. Kaski, “Time series prediction with multilayer perceptron, fir and elman neural networks,” in *Proceedings of the World Congress on Neural Networks*, pp. 491–496, Citeseer, 1996.

[52] J. P. Donate, X. Li, G. G. Sánchez, and A. S. de Miguel, “Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm,” *Neural Computing and Applications*, vol. 22, no. 1, pp. 11–20, 2013.

[53] M. Ghiassi, H. Saidane, and D. Zimbra, “A dynamic artificial neural network model for forecasting time series events,” *International Journal of Forecasting*, vol. 21, no. 2, pp. 341–362, 2005.