Time series forecasting: a comparison between statistical models and LSTM RNN

By

Filipe Lopes Laginha da Palma

Master dissertation in Modeling, Data Analysis
and Decision Support Systems

Supervised by:

Professor Maria Eduarda da Rocha Pinto Augusto da Silva

Faculdade de Economia
Universidade do Porto

2020

Biographical note

Filipe Lopes Laginha da Palma was born on May 26^{th} 1993, in Oporto, Portugal. Filipe enrolled in the Bachelor in Economics in 2012 at Faculdade de Economia da Universidade do Porto (FEP), accomplishing his degree in 2016. Shortly after, he joined KPMG as a consultant where he was exposed to digital transformation projects and where he started to develop a keen interest in the power of data when applied to decision making. In 2017, looking to further his interest in Data Science, Filipe enrolled in the Master in Data Analytics at FEP, with the main goal of acquiring formal knowledge of analytical techniques and tools, to use them as a complementary tool to his professional development.

More recently, Filipe has been working in Canada in a finance transformation project with the main purpose of developing and implementing a data architecture that will allow one of the country's major telecommunications company to implement automated data models to support the decision making process of all business lines.

Acknowledgements

I would like to thank my supervisor, Professor Maria Silva, for all her supervision, availability, and most important, patience. To all my work colleagues and supervisors, for all the support and comprehension during these difficult years juggling between work and classes. To my friends, for their support and encouragement. Last, to my family for all the support and for investing in my education, granting me the necessary resources to succeed in this competitive world.

Abstract

Over the past decade, the importance of having a data strategy within a company has increased considerably. Nowadays, any business decision has to be supported by data, which is one of the differentiating elements in the success of a company. The quality of existing data within an organization and the underlying accuracy of the models built with it is a determining factor in quality decision making.

Traditionally, and still with considerable weight in companies, the forecast of time series is elaborated using rudimentary and, above all, highly fallible techniques. In addition, the high consumption of hours required by these tasks represents a high opportunity cost for organizations, leading to decreases in productivity and efficiency.

The objective of this dissertation is to compare some of the most used statistical time series forecasting techniques with a deep learning, Long Short Term Memory model (LSTM). The ability of this model to store and forget information, allowing it to identify structure and patterns in the data, such as non-linearity and complexity, has increasingly called the attention of the academic community in the application to time series forecasting. Through a comparison between statistical methods and an LSTM model, this dissertation aims to contribute to the choice of more efficient and accurate time series forecasting methods, supporting decision making and contributing to the increase in business productivity.

Keywords: LSTM, time series, forecasting, statistical methods

Resumo

Ao longo da última decada, a importância da existência de uma estratégia de dados no seio de uma empresa aumentou consideravelmente. Hoje em dia, qualquer decisão de negócio tem de ser suportada em dados, sendo este um dos elementos diferenciadores no sucesso de uma empresa. A qualidade dos dados existentes no seio de uma organização e a subjacente precisão dos modelos com eles construídos é um fator determinante na tomada de decisão.

Tradicionalmente, e ainda com um peso considerável nas empresas, a previsão de séries temporais é elaborada recorrendo a técnicas rudimentares e, acima de tudo, altamente falíveis. Adicionalmente, o elevado consumo de horas requerido por estas tarefas representa um elevado custo de oportunidade para as organizações, conduzindo a quebras de produtividade e eficiência.

O objectivo desta dissertação é comparar algumas das técnicas estatísticas de previsão de séries temporais mais utilizadas com um modelo de *deep leraning*, *Long Short Term Memory* (LSTM). A capacidade deste modelo de guardar e esquecer informação, permitindo-lhe identificar estrutura e padrões nos dados como a não linearidade e complexidade, tem cada vez mais chamado a atenção da comunidade académica na aplicação a previsão de séries temporais. Através de uma comparação entre métodos estatísticos e um modelo LSTM esta dissertação pretende contribuir para a escolha de métodos de previsão de series temporais mais eficientes e precisos, servindo de suporte à tomada de decisão e contribuindo para o aumento da produtividade empresarial.

Palavras-chave: LSTM, séries temporais, métodos estatísticos, previsão

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Forecasting has always been of utmost importance to any business. Everyday, managers make decisions without knowing what the future holds. They define the inventory requisites without knowing the volume of sales they will face, they set orders to buy new equipment despite demand behaving in a unpredictable way, and make investments with uncertainty about what profits they will return. Nevertheless, managers are always trying to make better choices amid such uncertainty, trying to minimize the error rate of their decisions. The ability to provide accurate estimates is the main purpose of forecasting and that makes it a major priority for any company.

From my professional experience as a management consultant I am often faced with the problems created by inaccurate forecasting. From scheduling staff in a call centre to defining the stock requirements to stock inventory, the slightest deviation from the actual needs can result in huge operational constraints. At the end of the day, forecasting errors always result in two major problems that any company wants to avoid: 1) lower productivity, from having to redo inaccurate forecasts and, worst of all, 2) higher costs or lower revenues.

The main purpose of this dissertation is to study more recent forecasting techniques, namely Long Short Term Memory (LSTM), that show promising results when comparing to more traditional statistical forecasting methods. By providing a comparison between statistical forecasting methods and LSTM models I hope to contribute to a better decision making process for any manager that relies on forecasting techniques.

1.2 Problem definition

Companies face numerous decisions everyday. However, most companies still rely on traditional methods that are highly resource consuming and more prone to mistakes, resulting in large amounts of effort and, most of the time, highly inaccurate predictions. Therefore, it is of the utmost importance to any business leader to have accurate and less time consuming methods that allows them to have a better resource allocation.

The purpose of this dissertation is to provide a comparison between some frequently used statistical forecasting methods, such as SARIMA and Holt Winters, and machine learning methods, specifically Long Short Term Memory. The main assumption of forecasting something is that the thing that it is trying to be forecasted is unknown and therefore it can be treated as a random variable. As an example, the sales volume for next week can assume a large range of values and, until the end of next week it is not possible to know the actual values. Hence, until the event actually happens the next week sales volume is a random quantity. Most of forecasting situations, the further ahead one tries to predict it the more uncertain it will be. When applying forecasting methods, we are trying to predict the middle range of viable values the random variable can assume.

This dissertation will focus on three datasets providing different case scenarios: 1) monthly prediction of disembarking passengers at an airport, 2) daily temperature prediction and, 3) weekly sales prediction for a retailer company. The datasets were carefully selected to provide a more in depth comparison by guarantying different data patterns, such as trend, seasonality and time frequency.

Statistical forecasting methods take into account all previous observations to provide the set of values the random variable could take along with their relative probabilities, also known as the probability distribution, more specifically forecast distributions. The first part of this dissertation will then centre on analyzing the accuracy from statistical methods in order to provide a point of comparison with a more recent method, LSTM. LSTM is an artificial recurrent neural network architecture (RNN) used in the field of deep learning. One great advantage when comparing to statistical methods is that they are capable of storing information over a period of time, in other words, they have a memory capacity. When dealing with time series data this characteristic is extremely useful, letting the user decide which information will be stored and discarded. Despite not being a novelty in time series prediction they are still quite recent and there is not much literature oriented to business problems, comparing the performance of this technique to more classic methods.

Lastly, the dissertation will try to provide an answer to which forecasting techniques should be used depending on the problem and the stand point of statistical methods when compared to a machine learning approach.

1.3 Thesis overview

The next section will be focused on the literature review of statistical time series models: ARIMA, SARIMA, Holt-Winters on section 2.1, LSTM on section 2.2 and evaluation metrics on section 2.3. Afterwards, we present the case studies, followed by data description and preparation as so the experimental setup for each dataset. Finally, we present the main conclusions of this dissertation.

Chapter 2

Literature review

The following section provides an overview of the literature surrounding the models that will be applied in this dissertation as also some general guidelines on how they should be applied. Additionally, the section ends with an analysis on which evaluation measure should be considered in order to properly compare statistical models with an LSTM (deep learning model).

2.1 Statistical time series models

2.1.1 ARIMA

Autoregressive Integrated Moving Average models, ARIMA, were popularized by Box and Jenkins (1970) and are a generalization of the Auto Regressive Moving Average model (ARMA). Before diving into what is an ARIMA model, it is important to first introduce the concept of stationary, to discuss autoregressive moving average models and time series differencing.

As defined by Kwiatkowski, Phillips, Schmidt, and Shin (1992), a time series whose properties are not dependent on the time at which the series is observed is considered to be a stationary time series. Therefore, any time series that exhibits a trend or the data is seasonal is not stationary since both of these factors interfere with the value of the time series at different times. Normally, a stationary time series does not have any patterns that are predictable in the long-term. Hence, a plot of a stationary time series should exhibit data around an horizontal line with a constant variance. Stationarity allows the meaningful estimation of statistical measures such as the mean, the variance and the serial correlation or auto-correlation.

The statistical models for time series are all developed under stationarity conditions. The simplest model linear models are the ARMA models.

According to Shumway and S.Stoffer (2017), AutoRegressive (AR) models forecast the variable of interest using a linear combination of past values of the variable, suggesting a regression of the variable against itself. The mathematical formulation of an autoregressive model of order p is

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t,$$

with ε_t representing white noise, a sequence of independent and identically normally distributed random variables. Highly useful to handle a great variety of different time series patterns, autoregressive models of order p are commonly referred as $AR(p)$ models. The values of the parameters $\phi_1, \ldots, \phi_p$ affect the time series patterns from the model and ε_t, the variance of the error term, changes the series' scale. The parameters ϕ_j are restricted so that the model is stationary. These restrictions are the following for

- an $AR(1)$ model: $-1 < \phi_1 < 1$
- an $AR(2)$ model: $-1 < \phi_2 < 1, \phi_1 + \phi_2 < 1, \phi_2 - \phi_1 < 1$

For $p \geq 3$ these restrictions are easily expressed in terms of the roots of the polynomial $1 - \phi_1 z - \phi_2 z^2 - \ldots - \phi_p z^p$ which must be in modulus greater than 1.

Instead of relying on past values of the forecast variable in a regression, a moving average model (MA) counts on errors of the past forecast in a model similar to a regression as mentioned in Box and Jenkins (2015).

$$y_t = c + \varepsilon_t \ + \theta_1 \varepsilon_{t-1} \ + \theta_2 \varepsilon_{t-2} \ + ... + \theta_q \varepsilon_{t-q}$$

,

with ε_t representing white noise. The above model is known as an $MA(q)$ model, meaning a moving average model of order q. MA models are allways stationary. Any stationary $AR(p)$ model can be written as an $MA(\infty)$ model. If certain constraints on the MA parameters are applied the reverse holds and the model is called invertible. Hence, it is possible to write any invertible $MA(q)$ process as an $AR(\infty)$. The invertibility constraints applied to MA models are identical to the constraints applied to stationarity models, namely the roots of the polynomial $1 + \theta_1 z + \theta_2 z^2 + \ldots + \theta_q z^q$ must be in modulus greater than 1.

Hyndman and Athanasopoulos (2019) proclaim differencing is a well known method to make stationary a non-stationary time series dataset. The differenced series, as the name suggests, is the difference between consecutive observations in the original series and it can be formulated as follows

$$y'_t = y_t - y_{t-1}$$

The series that is differenced will only have $T - 1$ values because computing a difference y'_1 for the first observation is simply not viable. For seasonal time series a seasonal differencing may be considered, which is the change between an observation and the previous one from the same season.

$$y'_t = y_t - y_{t-m}$$

,

where m is equal to the number of seasons. This process of subtracting the observation after a lag of m periods is known as "lag-m differences". In more extreme cases it can be useful to use a first difference and a seasonal difference to obtan a stationary data. This way, the data

is transformed in the first phase, using logarithms and after that the seasonal differences are computed.

When combining autoregression, a moving average model and differencing we get a non-seasonal ARIMA Model, where the "I" stands for Integration, the opposite of differencing. The mathematical formulation of an ARIMA model is as follows

$$y'_t = c + \phi_1 y'_{t-1} + ... + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} + \varepsilon_t,$$

where y'_t stands for the differenced series. This ARIMA model is called an $ARIMA(p, d, q)$ model where:

- p is the order of the autoregressive part,
- d represents the degree of the first differencing involved, and
- q stands for the order of the moving average part

To be able to model an accurate ARIMA is important to understand the behaviour of the model. Hyndman and Athanasopoulos (2019) provide a clear view of the different behaviours of an ARIMA model depending on the values of the constant c. This particular constant has a key effect on the long-term forecasts:

- if $c = 0$ and $d = 0$, the long-term forecast tends to zero
- if $c = 0$ and $d = 1$, the long-term forecast tends to a non-zero constant
- if $c = 0$ and $d = 2$, the long-term forecast follows a straight line
- if $c \neq 0$ and $d = 0$, the long-term forecast tends to the data's mean
- if $c \neq 0$ and $d = 1$, the long-term forecast follows a straight line
- if $c \neq 0$ and $d = 2$, the long-term forecast follows a quadratic trend

As mentioned above, the value of d has also an effect on the prediction intervals - the higher, the faster the size increase of the prediction intervals.

At last, the value of p also assumes a relevant role if the data exhibits cycles. In order to get cyclic forecasts, it is mandatory to have $p \geq 2$, and other conditions on the parameters.

Hyndman and Athanasopoulos (2019) outline in their book some special cases of ARIMA models that may need closer attention when interpreting the results and can be seen in table 2.1.

One of the most challenging parts of computing an accurate ARIMA model is selecting the appropriate values for p, d and q. Usually, simple observation of the time plot is not often enough to arrive to any viable conclusion. Therefore, as evidenced by Brockwell and Davis (2016) the ACF and PACF plot, showing the correlation and partial autocorrelation plots, can provide valuable insights to help choose the correct ARIMA parameters. In fact, if the data belongs to an $ARIMA(p, d, 0)$ or $ARIMA(0, d, q)$, the ACF and PACF plot play an important role finding

White noise	ARIMA(0,0,0)
Random walk	ARIMA(0,1,0) with no constant
Random walk with drift	ARIMA(0,1,0) with constant
Autoregression	ARIMA(p,0,0)
Moving Average	ARIMA(0,0,q)

Table 2.1: Special cases of ARIMA model

the correct value of q and p. On the contrary, if both the parameters are positive, the plots are not helpful finding viable values. According to Nielsen (2019) if the ACF plot of the differenced data is exponentially decaying or sinusoidal and there is a significant spike at lag p in the PACF, but none beyond lag p then the data suggests an $ARIMA(p, d, 0)$. On the other hand, if the PACF is sinusoidal or exponentially decaying and there is a significant spike at lag q in the ACF, but none beyond lag p then the data suggests an $ARIMA(0, d, q)$.

At last, for helping choose the correct order of an ARIMA model, the Akaike's Information Criteria (AIC) introduced by Akaike (1973) plays an important role for selecting predictor for regression. The mathematical formulation behind is as follows

$$AIC = -2log(L) + 2(p + q + k + 1)$$

where L represents the likelihood of the data, $k = 1$ if $c \neq 0$ and $k = 0$ if $c = 0$.
In particuarly, for ARIMA models, the AIC is as follows

$$AICc = AIC + \frac{2(p + q + k + 1)(p + q + k + 2)}{T - p - q - k - 2}$$

and the Bayesian Information Criterion can be written as

$$BIC = AIC + [log(T) - 2](p + q + k + 1)$$

To obtain good models an ARIMA model should minimise the BIC, AIC or AICc. According to Hyndman and Athanasopoulos (2019) the AICc should be preferred over the other two, and, therefore, it will be the measure used in this dissertation. The AICc will be mainly used to select the values of p and q since it is not a an appropriate measure to help select the order of differencing d. Since the differencing alters the data on which the likelihood is calculated, turning incomparable the AICc values between models that have different orders of differencing. Hence, the ACF and PACF are used to choose d and after the AICc helps selecting p and q.

There are numerous ways an ARIMA model can be fitted to a set of time series data. For the purpose of this dissertation the approach suggested by Hyndman and Athanasopoulos (2019) will be preferred and can be summarized as follows:

1. Plot the data to look for atypical observations

2. If needed, stabilise the variance by transforming the data

3. If the data is non-stationary take first differences of the data until the data is stationary
4. Observe the ACF/PACF plots to understand if it is an $ARIMA(p, d, 0)$ or an $ARIMA(0, d, q)$
5. Test the selected model and rely on the AICc to look for a better one
6. Compute the residuals from the chosen model by plotting the ACF of the residuals and performing a portmanteau test of the residuals. If the plot is not similar to white noise then a modified model should be testes
7. Calculate forecasts once the residuals look like white noise.

2.1.2 SARIMA

Autoregressive Integrated Moving Average is one of the methods mainly used for forecasting univariate time series data. Despite the method being capable of handling data with a trend, it is not able to support time series with a seasonal component.

The Seasonal Autoregressive Integrated Moving Average model, SARIMA, builds upon the ARIMA model as the name may suggest. It includes the same parameters p, q and d but also an extra set of parameters to take care of the seasonality of the time series.

$$ARIMA(p, d, q)(P, D, Q)_m$$

Uppercase notation is used for the seasonal component and lowercase for the non-seasonal component of the model. The meaning of the parameters for the non-seasonal part is the same as the ones on the ARIMA described above, nevertheless, for convenience, an extended recap is provided:

- p order of the autoregressive model - number of lagged terms described in the AR equation
- d number of differences required to make the time series stationary
- q order of the moving average model - number of lagged terms describer in the MA equation
- P order of the seasonal autoregressive model
- D number of seasonal differences applied to the time series
- Q order of the seasonal moving average model
- m seasonality of the model. As an example, if the seasonality of the time series is quarterly then $m = 4$, which means that the pattern repeats once very four months

According to Cowpertwait and Metcalfe (2009) a seasonal ARIMA model relies on differencing at a lag equal to the number of seasons (m) to remove additive seasonal effects. As with lag 1 differecing to remove a trend, the lag m differencing introduces a moving average term. At

lag m, moving average terms (Q) and autoregressive terms (P) are included. Using the backward shift operator the $SARIMA(1,1,1)(1,1,1)_4$ can be expressed as follows

$$(1-\phi_1 B)(1-\Phi_1 B^4)(1-B)(1-B^4)y_t = (1+\theta_1 B)(1+\Theta_1 B^4)\varepsilon_t$$

Generally, the SARIMA model is non-stationary, however when the roots of the characteristica equation exceed unity in absolute value and $D = d = 0$ the outcome model is stationary.

As the ARIMA model, seasonal ARIMA models accept a wide number of parameters and terms combinations. Hence, when choosing the parameters a trial and error approach should be preferred using the AICc as a criterion to choose the best-fitting model.

The ACF and PACF also play an important role for estimating the parameters for the SARIMA and as suggested by Hyndman and Athanasopoulos (2019) should be used to observe the seasonal part of the AR or MA model. As an example, an $ARIMA(0,0,0)(1,0,0)_{12}$ will evidence a seasonal lags with exponential decay in the ACF and a PACF with a single significant spike at lag 12. In fact, the procedure behind modelling the SARIMA is very similar to the one used for the non-seasonal data, with the excpetion that it is necessary to choose the seasonal MA and AR terms and the non-seasonal components of the model.

Mills (2019) suggests in his book a practical approach on how to identify a seasonal model that will be used as guideline for his dissertation:

1. Plot the time series data to look for trend and seasonality features
2. Perform differencing if necessary. When seasonality exists but no trend, seasonality will show in the ACF by tapering at multiples of m
    - If the ACF/PACF plots show a linear trend without seasonality a first difference should be taken. In case the trend is curved, then, before differencing, the data should be transformed.
    - In case there is seasonality and trend, the data should undergo seasonal difference and after the trend should be re-evaluated. In case the trend remains, then first differences should be taken.
    - If no obvious seasonality or trend exists it is not neessary to take any differences
3. In case differencing was performed, have a look at the ACF and PACF plots
    - Non-seasonal terms: the first lags should be examined to evaluate non-seasonal terms. Spikes at low lags in the ACF suggest non-seasonal MA terms. The same in the PACF suggest non-seasonal AR terms.
    - Seasonal terms: the lags that are multiples of m should be examined for the existence of any pattern. As an example, the lags 12,24,36 should be examined for monthly data.
4. Based on the conclusions from step 3, estimate the model that may be appropriate
5. At last, examine the residuals of the ACF to validate the model accuracy.

2.1.3 Holt-Winters

The Holt-Winters method introduced by Holt (1957) and Winters (1960), while working in the School of Industrial Administration at Carnegie Institute of Technology, relies on exponentially weighted moving averages to update the estimates of the seasonally adjusted mean (also known as the level), slope, and seasonals.

There are two variations to this method that differ in the nature of the seasonal component and will be further explored. The additive method:

$$\begin{aligned}\hat{y}_{t+h|t} &= l_t + hb_t + s_{t+h-m(k+1)} \\ l_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \\ b_t &= \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1} \\ s_t &= \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}\end{aligned}$$

where k is the integer part of $(h-1)/m$, which guarantees that the estimates of the seasonal indices utilized for forecasting come from the final year of the sample. The l_t stands for the level equation, which exhibits a weighted average between the seasonally adjusted observation $(y_t - s_{t-m})$ and the non-seasonal forecast $(l_{t-1} + b_{t-1})$ for time t. The b_t, which represents the trend, has an equation similar to the Holt´s linear method. The s_t, representing the seasonal component, has an equation that exhibits a weighted average between the current seasonal index, $(y_t - l_{t-1} - b_{t-1})$ and the seasonal index of the same season last year. The parameters α, $\beta*$ and γ are the smoothing parameters for l_t, b_t and s_t respectively. Additionally, m is used to denote the seasonality frequency, i.e., number of seasons in a year.

Additionally, the multiplicative method, a method preferred when the seasonal variations change proportional to the level of the series, the seasonal component is expresses in percentages and the series is seasonally adjusted by dividing through by the seasonal component. The multiplicative method can be expressed the following way

$$\begin{aligned}\hat{y}_{t+h|t} &= (l_t + hb_t)s_{t+h-m(k+1)} \\ l_t &= \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1}) \\ b_t &= \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1} \\ s_t &= \gamma\frac{y_t}{(l_{t-1} - b_{t-1})} + (1-\gamma)s_{t-m}\end{aligned}$$

Usually, it is possible by looking at the data to understand if the multiplicative or additive method should be applied. Nevertheless, forecasting both methods in R is a simple process and can provide a clearer view if one method should be preferred over the other. At last, for improving the results of the Holt-Winters method Hyndman and Athanasopoulos (2019) suggest damping, which is both possible with the multiplicative and additive Holt-Winters' methods.

2.2 LSTM

Learning to store information over extended time intervals via recurrent backpropagation takes a long time, mainly because of the decaying error backflow being insufficient. When reviewing Hochreiter (1991) analysis of this problem, Hochreiter and Schmidhuber (1997) introduced a novel gradient-based method "Long Short-Term Memory", commonly know as LSTM.

Traditionally, neural networks are not able to store information about past events to help make prediction about current and future events. By adding loops to the networks, recurrent neural networks address this issue, allowing information to persist. Recurrent neural networks, RNN, can be perceived as a set of multiple copies of the same network, where each transmits a message to a successor. LSTM, arises as a special type of RNN which performs better than the standard version for many tasks.

One of the major popularized benefits of RNNs is the possibility to connect past information with the present task. However, the gap between the required information and the point where it is needed may become very large. In theory, RNNs should be perfectly capable of handling this long-term dependencies. Nevertheless, as shown by Hochreiter (1991) and Bengio, Simard, and Frasconi (1994) there are significant problems with this theory

LSTMs are conceived to avoid the long-term dependency problem and work remarkably well with numerous problems, one of them being time series forecasting that will be explored further on. RNNs have the form of a chain of repeating modules of a neural network. Like the RNN, LSTM also has this chain structure, however, the repeating module is conceived differently. Instead of relying on a single neural network layer, LSTM work with four neurons that interact in a particular way. The differentiating point is the cell state, which goes straight down the entire chain, having few minor linear interactions. By making use of the structures known as gates, LSTM has the capability to add or remove information to the cell state. Made of a pointwise multiplication operation and a sigmoid neural net layer, gates are a way to decide if information should come through or not. The sigmoid layer is a number between zero and one, that provides information of how much of each component passes. Zero stands for "nothing passes" and one for "everything passes". LSTM relly on three different gates to regulate information flow in a cell:

- Forget gate - which decides what information is kept and what information is thrown away.
- Input gate - which updates the cell state.
- Output gate - which decides what the next hidden state should be.

The forget gate marks the first step in a LSTM model, deciding which information is kept and which is thrown away. To make this decision, the model analyzes h_{t-1} and x_t and outputs a number between 0 and 1 for each number in the cell state C_{t-1}, where h represents an output value, x an input value and C cell state. A 1 means that the information is kept and a 0 that the information is forgotten. The next step, decides on what new information is stored in the cell state and is processed in two parts. At first, the input gate layer makes a decision on which values are updated or not. And after, a hyperbolic tangent layer builds a vector of new candidate values, $\hat{C}_t$, that could be added to the state. Next, the old cell state, C_{t-1} is updated giving place to the

new cell state C_t. By multiplying the old state by f_t, forgetting function, the model forget what was decided to forget in the previous steps. Afterwards, the new candidates values, scaled by how much it was decided to update each state value is added (i), $i_t * \hat{C}_t$. Last, it is decided what is outputed. The output is based on a filtered version of the cell state. First, an output gate decides what parts of the cell are outputed and then the cell states goes through tan ad is multiplied by the output of the output gate, in order to only output the parts desired.

The above description is the standard behaviour of a normal LSTM, however, there are many different variations proposed in the literature. Gers and Schmidhuber (2000) introduced a widely used LSTM variant that adds peephole connections which lets the gate layers look at the cell state. Another common variation is to rely on coupled forget and input gates. Instead of making separate decision on what should be added and forgotten, those decisions are made together. The Gated Recurrent Unit, GRU, a variation of the LSTM presented by Cho et al. (2014) combines the forget and input gates into a single update gate. Additionally, the cell and hidden state are merged among other changes. The model results in a simpler LSTM version and has been gaining popularity in recent years. The Depth Gated RNNs by Yao, Cohn, Vylomova, Duh, and Dyer (2015) presents an extension of LSTM to use a depth gate to connect memory cells of adjacent layers, introducing a linear dependence between lower and upper recurrent units. Koutník, Greff, Gomez, and Schmidhuber (2014) presents a Clockwork RNN, an alternative way to deal with long-term dependencies. As seen, there are numerous variations to the standard LSTM model and is not easy to select the best one. In fact, according to Greff, Srivastava, Koutník, Steunebrink, and Schmidhuber (2015) in their comparison study all the variations are very similar in the produced results. Nevertheless, the study conducted by Jozefowicz, Zaremba, and Sutskever (2015) suggests that some types of LSTMs work better on certain tasks.

As shown by Elsworth and Güttel (2020), LSTM provide great results when applied to forecasting time series data. However, one of the main problems when working with LSTMs is finding the right hyperparameters and even understanding what is going on behind the scenes. When choosing the right amount of nodes and layers there is no definite rule. A common formula that can work as a starting point is as follows:

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o)}$$

Where the number of input neurons is represented by N_i, the number of output neurons N_o, the number of samples in the training data N and α is the scaling factor that can vary between 2 and 10. In case the model is not very demanding, other rules can be put into place to find the number of nodes. Nevertheless, despite being easier to apply they do not guarantee optimal results. Concerning the hidden layer number of nodes, generally, 2 layers are more than capable to detect features. By adding more layers, despite having the chance to improve the model it can also make it more complicated to train. As standard, 1 hidden layer is more than enough for simple models and 2 layers can deal with most of the complex features that may arise with a difficult problem. To prevent model overfitting, a dropout layer should be added to an LSTM model. This way the randomly selected neurons are ignored in the training phase, reducing the level of sensitivity of individual neurons' weights. Last, a final layer should be added, the activation layer, which can be inside the density layer. By splitting the activation layer from the density layer it

becomes possible to return the reduced output of the density layer. Another common problem is what activation layer to choose, which may vary depending on the application chosen. Normally, the softmax activation layer works quite well guarantying accurate results, allowing the model to perceive the outputs as probabilities.

Some of the commonly manipulated parameters to improve the performance of the model are batch size and epochs. The batch size coordinates the number of samples to work through before updating the internal model parameters. As an example, a batch size of 50 takes first 50 samples from the training dataset and trains a network. After, it takes another batch of 50 samples and trains a new network again. This procedure is repeated until propagated over the networks of all samples. If all training samples create one batch, it is called batch gradient descent. If it is the composed of one sample only, then it is called stochastic gradient descent. In case it is greater than one sample but less than the size of the training dataset then it is called mini-batch gradient descent. For this last batch type, popular sizes advised in literature are 32, 64 and 128 as noted by Bengio (2012). Additionally, the epoch, which is made of one or more batches, decides the number of times the learning algorithm works through the entire training dataset. One epoch results in one backward pass and one forward pass of all the training dataset. To decide on what should be the number of epochs it is important to first calculate the RMSE of train and test data for each number of epochs selected. This way, overfitting can be prevented. After finding the optimal number of epochs the tests can be repeated with different neuron numbers. Normally, the epochs' number is high, varying between hundreds and thousands, granting the learning algorithm to run until the model's error has been minimized. Anzanello and Fogliatto (2011) suggest common epochs numbers to be 10, 100, 500 and larger.

At last, the right configuration of an LSTM model should be performed using a trial and error approach, relying on the Root Mean Square Deviation, RMSE, to access the model's accuracy.

2.3 Evaluation metrics

To evaluate a forecast accuracy it is of the utmost importance to use genuine forecasts. Hence, the residuals' size can not be considered as a credible indication of how large forecast errors are. To determine the accuracy of forecasts the performance of the model needs to evaluated under new data that was not previously used to fit the model. To overcome this problem, and has mentioned by Hyndman and Athanasopoulos (2019), it is standard practice to distinguish the data set in two partitions, a training and a test dataset, being the first one used to estimate the parameters of the forecasting method that will then be used on the test data so that the accuracy can be evaluated. Since the data in the test dataset is not used to train the forecasting model, it is capable of providing a good indication of how well the model will perform with new data.

Usually, the size of the test set ranges between 20 and 30 percent of the total data. Nevertheless, when deciding the size of the test dataset it should never be considered a size smaller than the forecast horizon required. Some notes should be considered when using train and test datasets to evaluate forecast accuracy:

- The fact that the model fits the training data well does not mean the it can perform valuable forecasts

- By selecting a model with enough parameters it is always possible to obtain a perfect fit
- Over-fitting a model or not identifying a pattern in the data are equally bad outcomes

The training dataset can also be called ”in-sample data” and the test dataset ”out-of-sample data” or ”hold-out set”. However, in this dissertation ”training” and ”test” data will be the names used.

Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), scale-dependent measures, are two of the most used metrics to measure accuracy for continuous variables. MAE measures the average magnitude of the errors in a set of predictions, not taking into account the actual direction. It is calculated by considering the average of the absolute differences between prediction and actual observations. RMSE is a quadratic scoring that measures the average magnitude of the error. It's the square root of the average of squared differences between prediction and actual observation. The RMSE has the advantage of penalizing large errors more. However, MAE grants the user a simpler interpretation. The MAE is more popular since it is easier to interpret and compute. A forecast method that minimizes the RMSE will lead to forecasts of the mean, while minimizing the MAE will lead to forecasts if the median. Despite being not so easy as the MAE to interpret, the RMSE is also commonly used and will be the preferred measure to evaluated models in this dissertation since it gives a relatively high weight to large errors.

Nevertheless, as noticed by Hyndman and Koehler (2006) there many possible measures to evaluate the accuracy of univariate time series forecasts. There is not a one measure fits it all and different measures should be considered and preferred depending on the problem at hands.

Chapter 3

Case studies

During this dissertation the importance of forecasting has been highlighted numerous times. Good forecasts capture the patterns and relationships which exist in historical data, avoiding reproducing past events that will not occur again. ARIMA and Holt-Winters models are among the most common used techniques. More recently, machine learning as been making solid steps into the forecasting world with such models, such as LSTM, exhibiting promising results, more accurate when compared to more traditional models. Therefore, by performing a comparison between some of the most popular statistical techniques and LSTM, this dissertation pretends to enlighten the reader on which model to apply to predict future values.

This section is divided between the two datasets being each subsection divided between statistical models and LSTM. The statistical forecasting models were made in R (2020) using the *forecasting* package from Hyndmam (2008), which provides methods to display and analyze time series forecasts. Each dataset was converted into a time series function using the *ts()* function of the *forecast* package and adapting to the proper frequency and dates for each dataset. Additionally. each dataset was divided into a train and test subset, with 70% and 30% of the observations, respectively, allowing to evaluate the accuracy of the models tested.

The datasets were selected taking into consideration different criteria that would allow a fair comparison between the different methods to be applied. Therefore, the datasets should respect the following characteristics: different frequency, trend and seasonality.

3.1 Airport dataset

The first dataset was extracted from *Instituto Nacional de Estatística* database and the underlying data concerns the number of disembarked passengers in Francisco Sá Carneiro Airport between January 2004 and December 2019. The data is monthly and is composed by 2 variables (month and number of passengers) and 192 observations. As seen in figure 3.1 the data follows a clear increasing trend accentuated more recently with a peak in the volume of passengers between 2015 and 2019.

Additionally, and as shown in figure 3.2, which allows to observe the data from each season overlapped, there is a notorious seasonality in the dataset. In fact, the seasonal plot allows the underlying seasonal pattern to be seen more clearly, evidencing a jump in the number of disem-

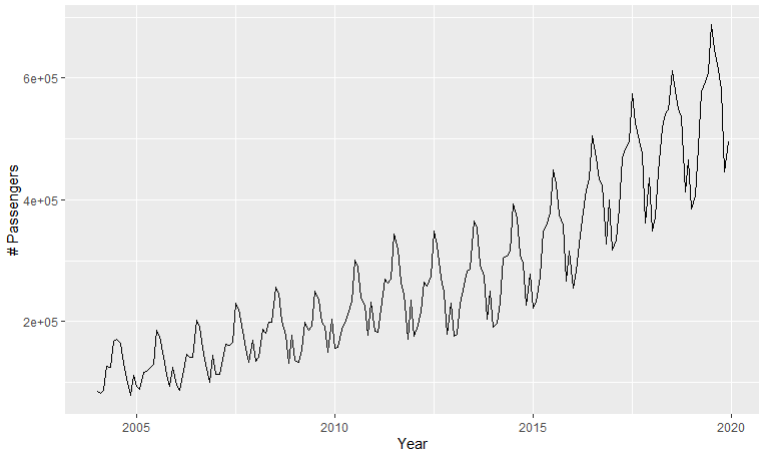

Figure 3.1: Disembarked passengers: Porto´s airport, Portugal (2004 - 2019)

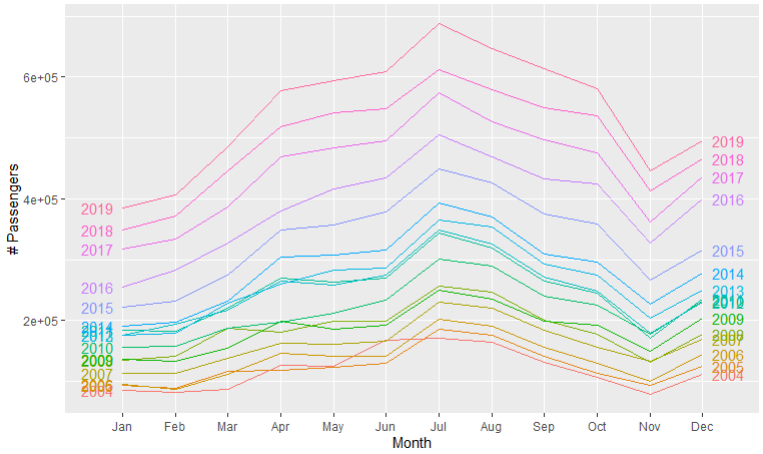

Figure 3.2: Seasonal plot airport dataset

barked passengers between the summer months (June, July and August). The graph also shows an upward trend with a peak in July, followed by a downward trend until November and ends with a spike in sales in December, probably due to end of year vacations travelling.

At last, the autocorrelation function exhibited in figure 3.4 evidences correlations significantly different from zero, with the bars considerably above the dashed blue lines. Additionally, the slow decrease in the ACF as the lags increase is due to the upward trend as seen in figure 3.1. The ”scalloped” shape is due the seasonality.

3.1.1 ARIMA

ARIMA models, an acronym for AutoRegressive Integrated Moving Average, are one of the most used approaches to time series forecasting, aiming to describe autocorrelations that may

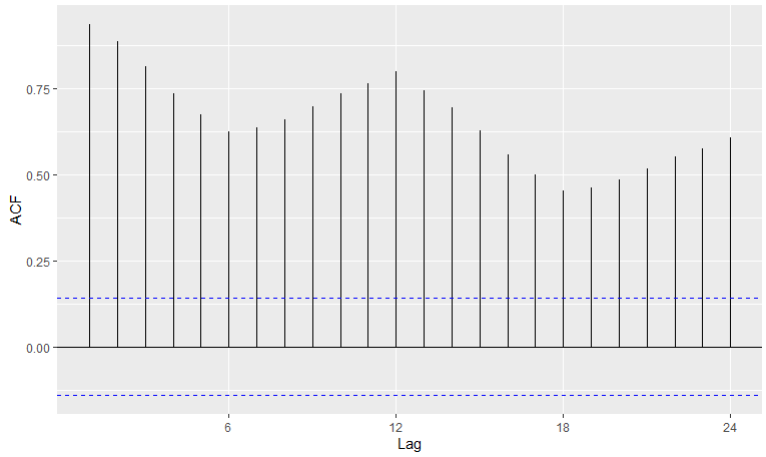

Figure 3.3: Autocorrelation function for airport data

exist in the data. As shown in Hyndman and Khandakar (2008) the *auto.arima()* function in R, that combines minimisation of the AICc, unit root tests and MLE to obtain the model, provides very accurate results when comparing to manual selected ARIMAs. Nevertheless, an automated process can always be a little dangerous since it does not allow to try different parameters.

As previously shown, the ACF plot exhibits the autocorrelations which measures the relationship between y_t and y_{t-k} for different values of k. If a correlation between y_t and y_{t-k} exists, then y_{t-1} and y_{t-2} also are correlated. Although, then y_t and y_{t-2} may also be correlated, since both are linked to y_{t-1}, preferably than because of new information in y_{t-2} that could possibly be useful for forecasting y_t.

Partial autocorrelation can be used to get over this problem. PACF measures the relationship between y_t and y_{t-k} after taking the results of lags $1, 2, 3, 4, 5...., k-1$. Therefore the first autocorrelation is similar to the first partial autocorrelation, since there is not anything in between them that needs to be removed. The last coefficient in an AR model can be used to estimate each partial autocorrelation. Precisely, α_k, the kth partial autocorrelation coefficient, is identical to the estimate of ϕ_k in an AR(k) model. This means that exist more optimized algorithms for calculating α_k than fitting all of these autoregressions, although they offer identical outcomes.

To fit an ARIMA model to the *airport* dataset described in the subchapters above it is important to first have a look at the time series plot and ACF and PACF plots as shown in figure 3.4.

The ACF is decaying with all lags being significant. In the PACF, there is one significant spike followed by no significant spikes (apart from the third spike slightly significant) and two other significant spikes until lag 12 suggesting an AR(1) model. Therefore an initial candidate model would be an ARIMA$(3, 1, 1)$. The results from fitting the proposed model along some couple variations can be seen in table 3.1.

As it can be seen from observation of the table above the ARIMA$(3, 1, 2)$ is the model that has the smaller AICc value, therefore the best ARIMA model. The ACF plof of the residuals in figure 3.5 shows that most of the autocorrelations are outside the threshold limits, showing that

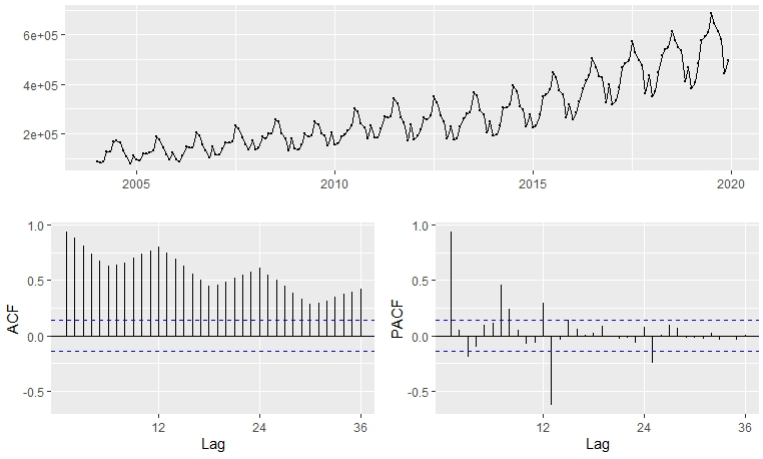

Figure 3.4: Time plot and ACF and PACF plots for the airport dataset

ARIMA Model	AICc
(3,1,1)	3167.81
(3,1,2)	3119.86
(3,1,3)	3165.21
(4,1,2)	3167.32
(3,2,2)	3170.23

Table 3.1: Datasets description

the residuals are not behaving like white noise. One of the reasons for this is that the *airport* dataset is highly seasonal and has an upward trend and a seasonal ARIMA model should be used instead. To overcome this, and since seasonal ARIMAs will be used in the next subchapter, some techniques can be used to adjust the data before applying the ARIMA model. Decomposition is mainly used for studying time series data but it can also be useful for forecasting. Taking into account an additive decomposition, the decomposed time series is expressed in the following way

$$y_t = \hat{S}_t + \hat{A}_t$$

where $\hat{A}_t = \hat{T}_t\hat{R}_t$. In order to forecast the decomposed time series, the seasonally adjusted component $\hat{A}_t$ is forecasted separately from the seasonal component $\hat{S}_t$.

Seasonal and Trend decomposition using Loess, STL, developed by Cleveland, Cleveland, McRae, and Terpenning (1990) is a flexible method for decomposing time series. The $mstl()$ function in R gives an automated STL decomposing that is capable of providing an acceptable balance between allowing the seasonality to change as time goes by and overfitting it. To address the residuals problem described above, the airport dataset was seasonally adjusted using the $mstl()$ and $seasdj()$function as it can be seen in figure 3.6 By applying the same ARIMA(3,1,2) model to the seasonal adjusted data the ACF plot of the residuals is now showing that all autocorrelations are inside the threshold limits, therefore, evidencing that the residuals are behaving like white

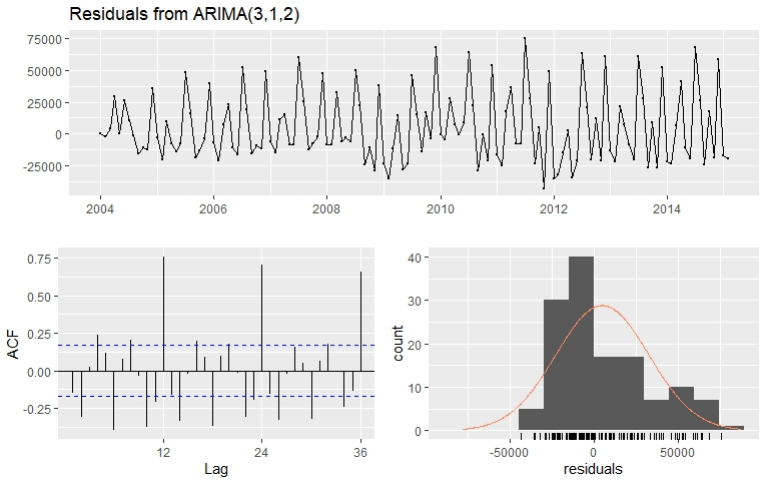

Figure 3.5: Residual plots for the ARIMA(3,1,2) model

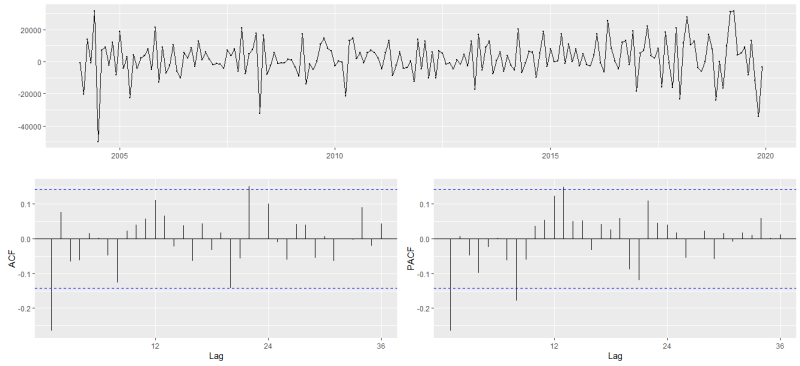

Figure 3.6: Seasonally adjusted plots for the airport dataset

noise exhibited in figure 3.7. Additionally, the large p-value resulting from the portmanteau test (0.8523) also suggests that the residuals are white noise.

The forecasts from the seasonal adjusted data can be seen in figure 3.8

At last, using the $forecast()$ function applied to the $mstl()$ the upper and lower limits of the prediction intervals on the seasonally adjusted date are "reseasonalised" by adding in the forecasts of the seasonal component. The final forecasts applied in the test dataset are exhibited in figure 3.9

The residuals from this model are shown in figure 3.10, which exhibits none significantly spikes. The model passes the Ljung-Box test altough with a p-value of 0.117. The model is appropriate for forecasting but given the seasonality of the data a seasonal ARIMA may be deemed more accurate.

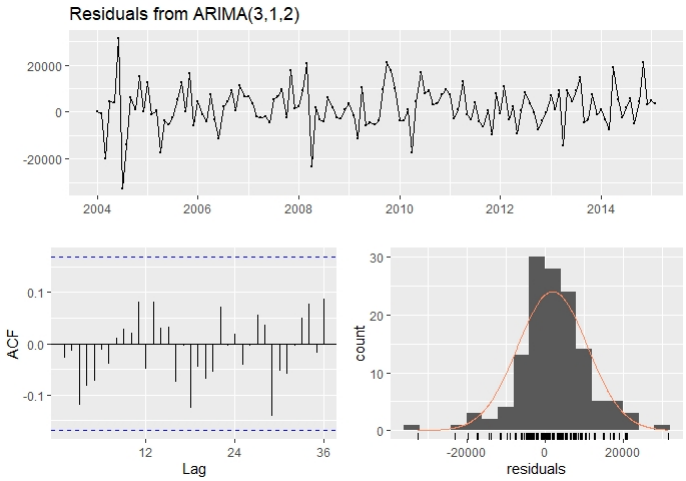

Figure 3.7: Residuals from the seasonally adjusted plots for the airport dataset

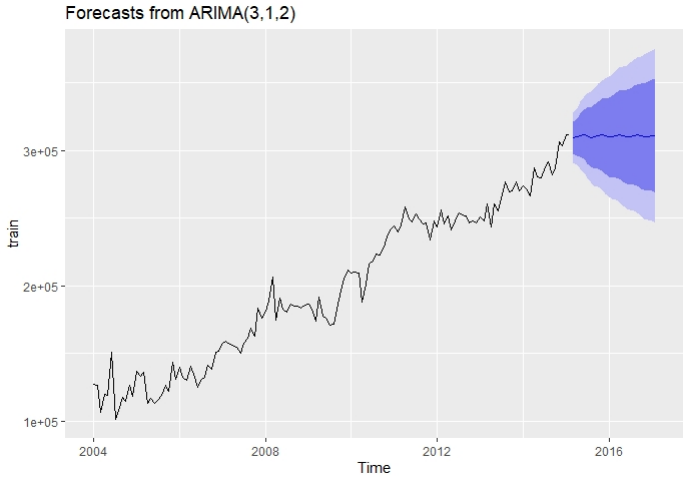

Figure 3.8: Forecasts for the seasonally adjusted airport dataset

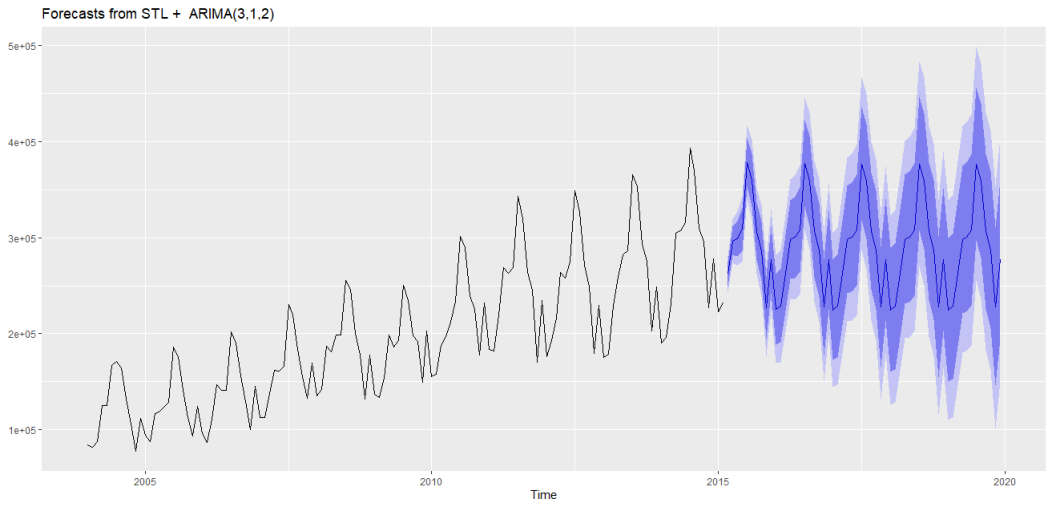

Figure 3.9: Forecasts of the airport dataset based on an ARIMA forecast of the seasonally adjusted data and a seasonal arima forecast of the seasonal component, after an STL decomposition of the data.

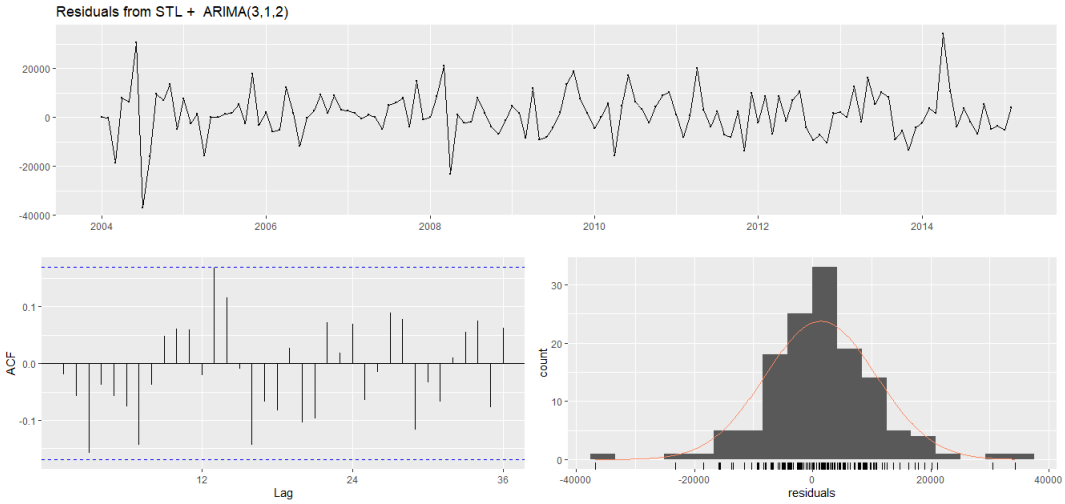

Figure 3.10: Residuals from the $STL + ARIMA(3, 1, 2)$ model applied to the airport data.

3.1.2 SARIMA

ARIMA models have the ability to model a broad array of seasonal data. By including seasonal terms in the ARIMA models, the ARIMA is able to accommodate seasonal data, that as seen in previous subchapters may improve the accuracy of the models. The model can be expressed as follows:

$$ARIMA(p,d,q)(P,D,Q)_m$$

where m stands for the number of observations per year. The terms of the seasonal part of the ARIMA are identical to the non-seasonal part, except they involve backshifts of the seasonal period,

Returning to first dataset presented in this dissertation, the airport dataset, from observation of figure 3.13 with the ACF and PAC for the seasonally adjusted data, it is not possible to derive any candidate model from looking at the pattern in the ACF. However, when observing the PACF pattern we see that there are two significant spikes in the non-seasonal lags. This may be indicative of a seasonal AR(3) term. In the seasonal lags there are also two spikes suggesting a AR(2) term. Therefore, the analysis indicates that an $ARIMA(2,0,0)(2,1,1)$ may be a viable model. By testing this model against some variations we get the following results present in table 3.2

SARIMA Model	AICc
(2,0,0)(2,1,0)	2641.77
(2,0,0)(0,1,0)	2666.89
(2,0,0)(2,1,1)	2632.69
(1,1,0)(1,1,0)	2620.74
(2,0,1)(2,1,1)	2634.93
(2,0,1)(0,1,0)	2668.62

Table 3.2: AICc of the different SARIMA for the airport dataset

Of the models presented, the one that minimizes the AICc is the $ARIMA(1,1,0)(1,1,0)$. From observation of figure 3.11 there are no spikes in the ACF and with a p-value of 0.17 it passes the Ljung-Box test deeming the model appropriate to forecast the dataset.

The results of the model when applied to the test dataset can be seen in figure 3.12. The model is able to capture the seasonal pattern, altough the upward trend of the data is not fully captured suggesting that the model could be further improve with a technique to capture the trend such as drift.

In fact, since the data exhibits a variation that increases with the level of the series, a transformation could be a viable way to deal with that trend. A useful transformation that includes both power transformations and logarithm is Box-Cox. Using the $Box.Cox.lambda()$ function in R we can see that a good value of λ is 0.2428875, which makes the size of the seasonal variation roughly the same across the whole series. Applying a seasonal arima model with the same parameters as above with a box-cox transformation we get the following results

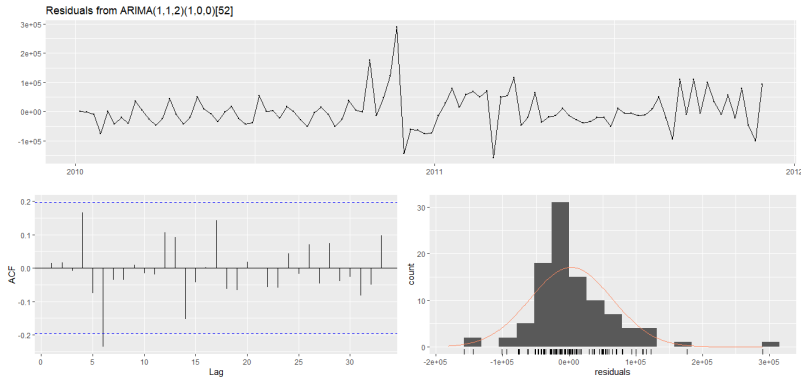

Figure 3.11: Residuals from the $ARIMA(1,1,0)(1,1,0)_{12}$ model applied to the airport data.

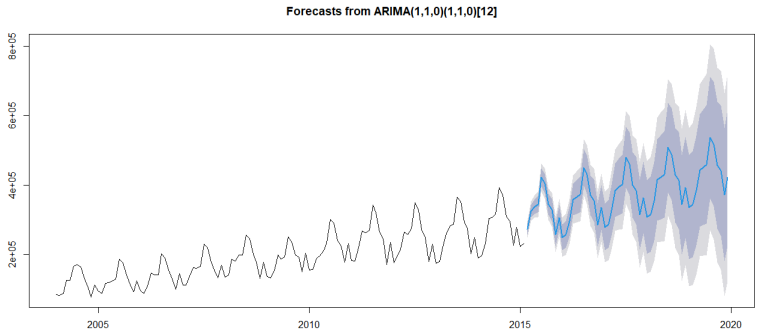

Figure 3.12: Forecast from the $ARIMA(1,1,0)(1,1,0)_{12}$ model applied to the airport data.

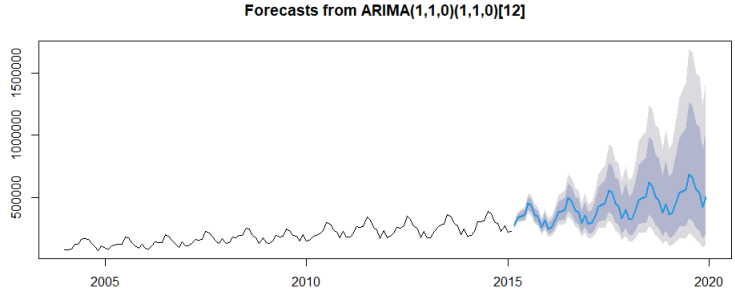

Figure 3.13: Forecast from the $ARIMA(1,1,0)(1,1,0)_{12}$ model with a Box-Cox transformation applied to the airport data.

Additionally, with a p-value of 0.20 it passes the Ljung-Box test although having a slightly significant spike as can be seen in the figure 3.14. It is also important to notice that the model has an AICc of just 402.47, a considerable improvement when compared to the seasonal ARIMAs without Box-Cox transformation.

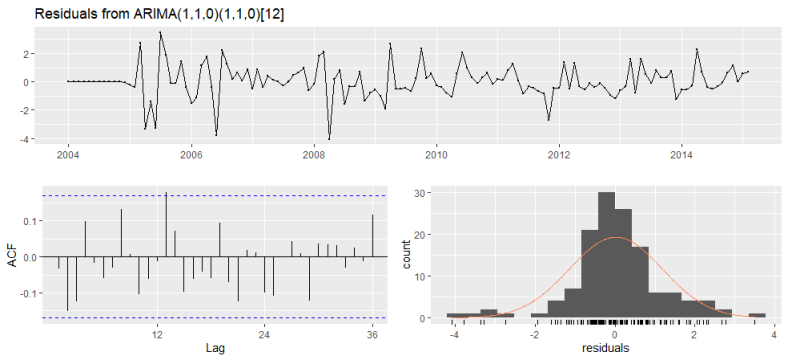

Figure 3.14: Residuals from the $ARIMA(1,1,0)(1,1,0)_{12}$ model with a Box-Cox transformation applied to the airport data.

3.1.3 Holt-Winters

Holt (1957) and Winters (1960) broadened Holt's seasonality capture method. The seasonal method Holt-Winters is composed by three smoothing equations and a forecast - one for the seasonal component s_t, one for the level l_t and another one for the trend b_t, with the respective smoothing parameters γ, α and β^*. m is used to indicate the frequency of the seasonality, as also presented in the notation for the SARIMA model.

The Holt-Winters´seasonal method can be used with two variations that impact the seasonal component. To choose one is important to first have a look at the data and identify the general characteristics in terms of trend and seasonality. When the seasonal variations do not change considerably throughout the series the additive method is preferred. On the other hand, when the seasonal variations are changing proportional to the series' level the multiplicative method should be considered instead. The additive method expresses the seasonal component in absolute terms taking into consideration the scale of the series observed, and in the level equation the series is seasonally adjusted by removing the seasonal component. Each year, the seasonal component will sum up to near zero. The multiplicative method, expresses the seasonal component in relative terms and the series is seasonally adjusted by dividing trough by the seasonal component. Each year, the seasonal component will add up to near m.

By applying both the multiplicative and additive seasonality Holt-Winters' method to forecast the airport dataset we get the results that can be seen in figure 3.15

Since both methods have equal number of parameters that need to be estimated it is possible to compare the training RMSE from both models. This exercise will be done later on the evaluation and performance metrics subchapter. However, by looking at the plot, the multiplicative method seems to capture better the patterns in the dataset. In fact, this result is somehow expected since the seasonal variation of the airport dataset increases as the level of the series increases. The forecasts' plot also reflects the above mentioned. The forecasts that result from the multiplicative seasonality method evidence an increasing and much larger seasonal variation as the level of the forecasts increases when compared to the forecasts resulting from the additive seasonality.

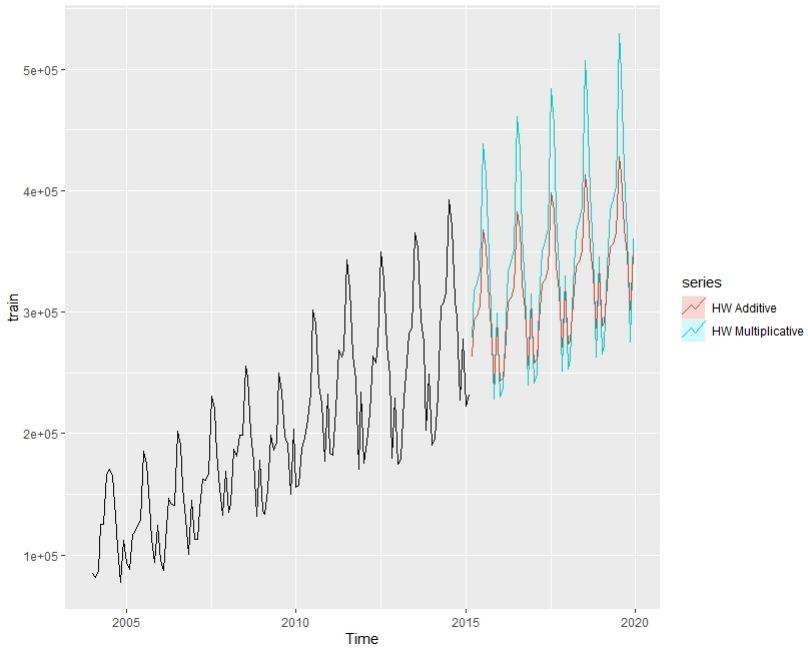

Figure 3.15: Forecasting airport dataset using the Holt-Winters method with both additive and multiplicative seasonality.

3.1.4 LSTM

There are not a lot of literature of LSTM applications to forecasting problems and understanding how to configure the neural network is not the easiest task since the lack of good theory. To apply the LSTM to any of the datasets proposed in this dissertation it is crucial to first apply three major transformation steps:

1. Transform the data into stationary, using a lag = 1 differencing to eliminate the increasing trend

2. Transform the data into a supervised learning problem. That is, organizing the data into an input and output patterns so that the observation at the previous step can be used as input to forecast the observation at the current time step.

3. Transform the observation to have a specific scale. Rescale the data to values that fit between -1 and 1 so that they can meet the the default hyperbolic tangent activation function of the LSTM model.

The LSTM model was developed using Python, particularly, the TensorFlow and Keras packages. Taking advantage of Keras, by initializing the model as $sequential()$ it is possible to simply stack multiple layers on top of each other. There is not a mandatory rule on how much nodes, hidden neurons and layers should be used when configuring and LSTM model. Therefore, the process should follow a trial and error approach. However, there are some guidelines that can be

used as a rule of thumb to give an initial hint about the parameters' configuration. For example, 2 layers have shown significant evidence to be good enough to detect more complex features for the hidden layers. To test the right hyperparameters configuration for the LSTM for each dataset a diagnostic approach will be preferred. The trial error experimentation will manipulate the following parameters:

- Batch - which defines the number of samples to work trough before updating the internal node parameters. A Batch can be perceived as a for-loop that iterates over one or more samples making predictions. At the end of the batch, the predictions are compared to the output variables that were expected, calculating an error. Taking the error, the algorithm is then updated and used to improve the model.

- Epoch - set the number of times that the learning algorithm works over the training dataset. One epoch means that each sample in the training data has had the chance to update the internal node parameters. An epoch is made of one or more batches. As an example, an epoch that has one batch is called the batch gradient descent learning algorithm.

  As it can bee seen the parameters selection that works best when predicting the airport dataset is 8 hidden layers, 100 epochs and a batch size of 3. The forecasts can be observed in figure 3.16, plotting the results against the testing dataset. Table 3.3 shows the results of different LSTM configurations experimented on the airport dataset

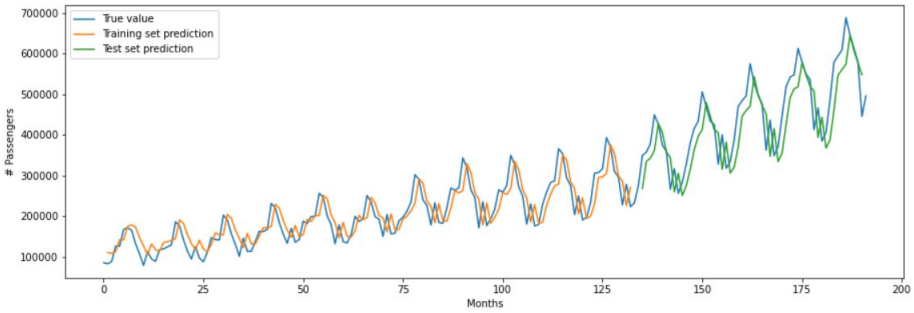

Figure 3.16: LSTM forecast of the airport dataset.

.

Hidden nodes	Epoch	Batch Size	Test RMSE
8	100	3	62622.86
8	500	3	79627.80
8	1000	3	90708.7
4	100	3	63532.37
2	100	3	67276.66
10	100	3	66247.57
8	100	10	68316.98
8	100	20	96616.69
8	100	4	70865.55

Table 3.3: AICc of the different SARIMA for the airport dataset

3.2 Sales dataset

The last dataset used in this dissertation was extracted from Kaggle and is part of a competition that the retailer company Walmart launched a few years ago as part of their recruitment process. Given the length of the data, it was slightly altered to meet a more approachable size that would allow to test the techniques proposed in this dissertation. The dataset contains 143 observations corresponding to the weekly sales of a given Walmart store between February 2010 and October 2012. As it can be seen in figure 3.17 the dataset exhibits a slight upward and cyclical trend with clear peaks around the final months of each year.

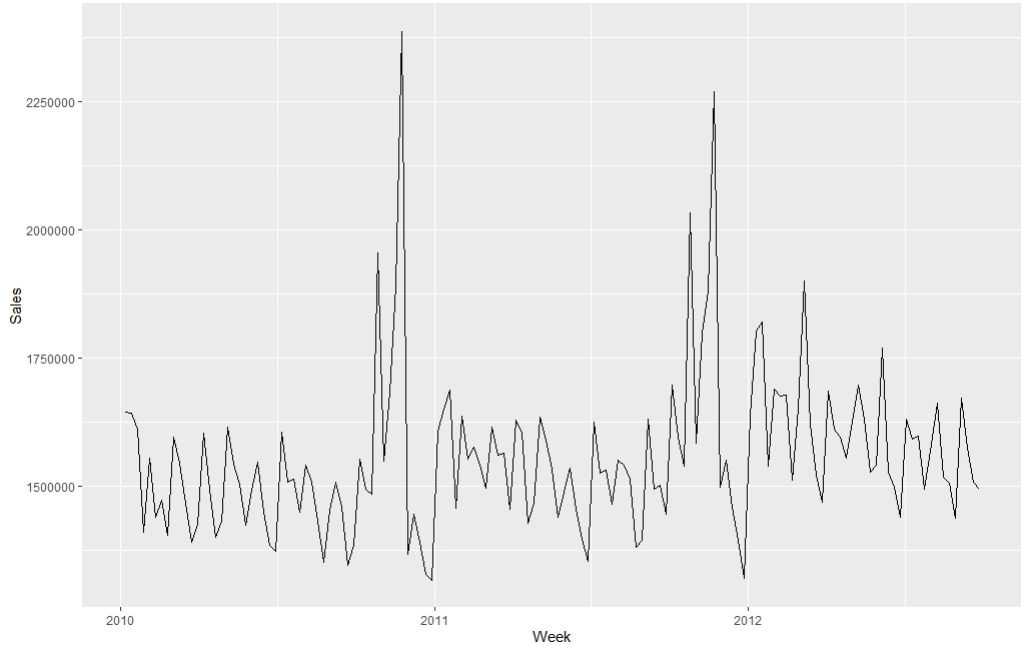

Figure 3.17: Weekly sales Walmart Store (2010-2012)

In fact, figure 3.18 clearly exhibits the cyclical trend with the data showing the same behaviour each week year over year with the slightly difference that the sales values tend to increase as time goes by. The graph also allows to see more clearly which weeks drive the sales volume up during the year, with the weeks previous to Christmas seeing a spike in sales each year.

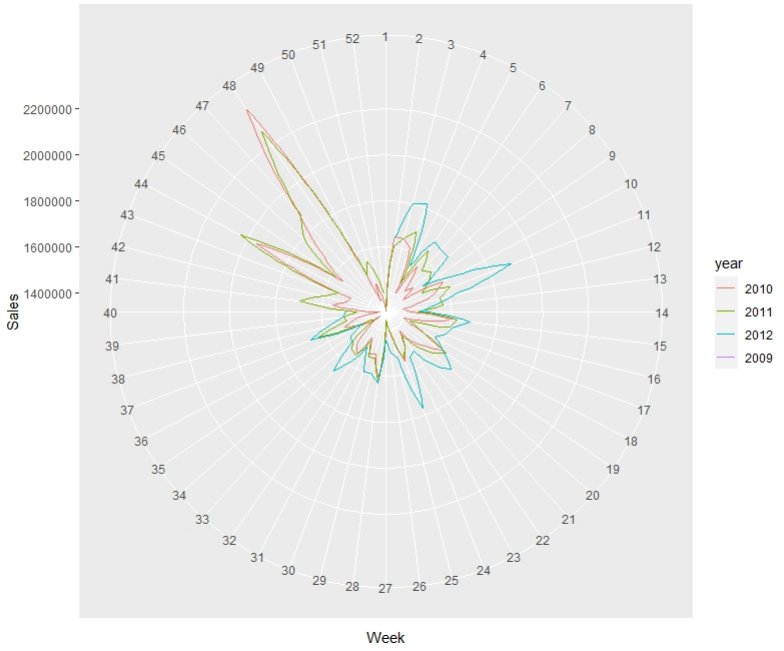

Figure 3.18: Seasonal graph for sales dataset

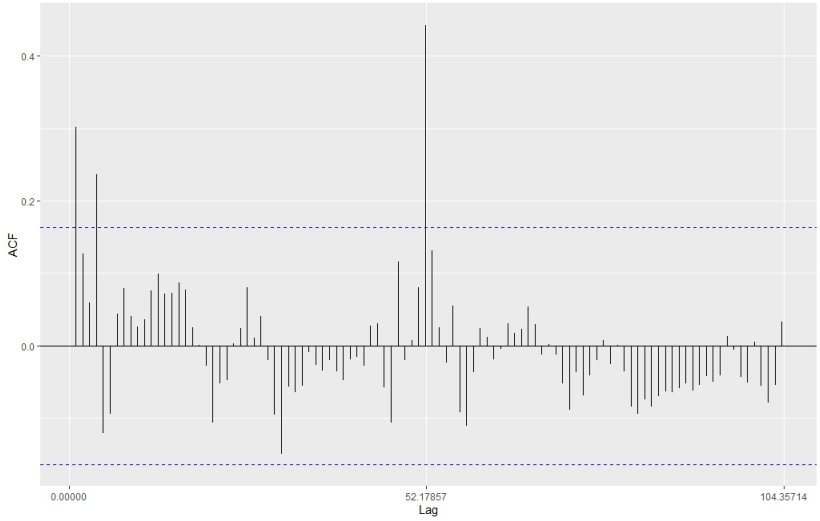

Figure 3.19: ACF plot sales dataset

At last, by analyzing the autocorrelation graph in figure 3.19 we can see that every peak in sales is followed by a sharp decline in the following weeks, contrasted by the positive and negative lags. Contrary to what as observed in the other graphs the ACF graph does not suggest that a trend exists in the data with the lags alternating between positive and negative ones. However, and as previously mentioned, the ACF graph shows a seasonal pattern with the autocorrelation being larger for the season lags than for other lags, at multiples of the seasonal frequency (a year).

3.2.1 ARIMA

As shown in the first dataset, before fitting an ARIMA model it is important to observe the ACF and PACF plots since they provide important clues regarding the parameters of the ARIMA. As seen in the section above, the sales dataset exhibits a seasonal pattern that can also be observed on the ACF plot with peaks in the graph around each seasonal lag. Therefore, the data exploration analysis suggests that an ARIMA model without any seasonal component may not be capable of capturing the pattern of the data. However, for testing purposes and model comparison a non seasonal ARIMA will be considered. Given the fact that the dataset does not have a large amount

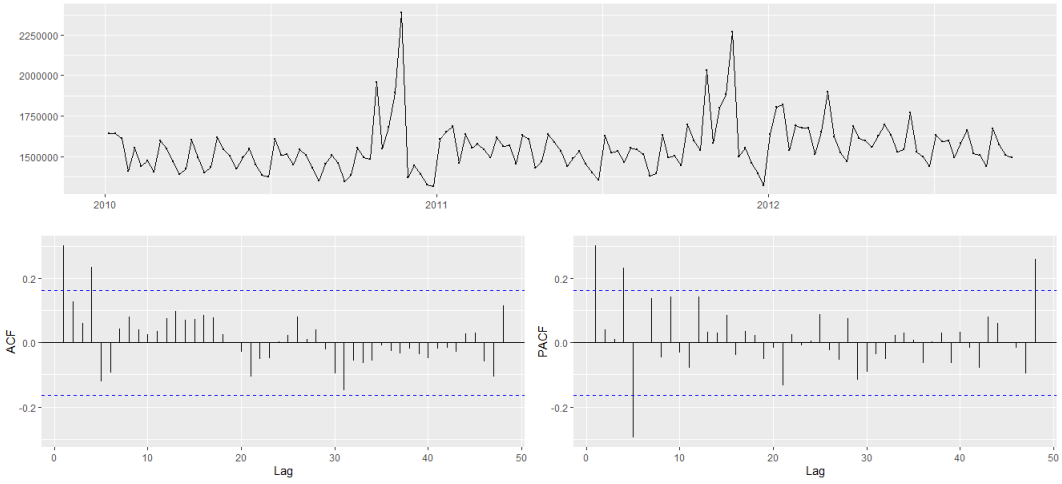

Figure 3.20: Time, ACF and PACF plots for the sales dataset

of observations, the training dataset will be comprised of 80% of the observations in order to have 2 complete years to train the model. From looking at the ACF and PACF plot an $AR(3)$ may be a viable candidate. A time plot of an $ARIMA(3, 0, 2)$ can be seen in figure 3.21. The model is not capable of capturing any data pattern assuming a mean value for all the observations after the first ones. The residuals plot with a p-value of 0.67 and with no significant spikes evidences that a mean model may be capable of providing acceptable estimations for the model. However, since the model does not capture any pattern, any significant spike in the dataset will not be captured. Therefore, other models should be explored. As did in the Airport dataset, a seasonal and trend decomposition using loess is one of the many ways to address the seasonal component of a dataset. In fact, this particular method allows to address any type of seasonality, allowing the seasonal component to change over time. Fitting an $ARIMA(3, 1, 2)$ with an STL

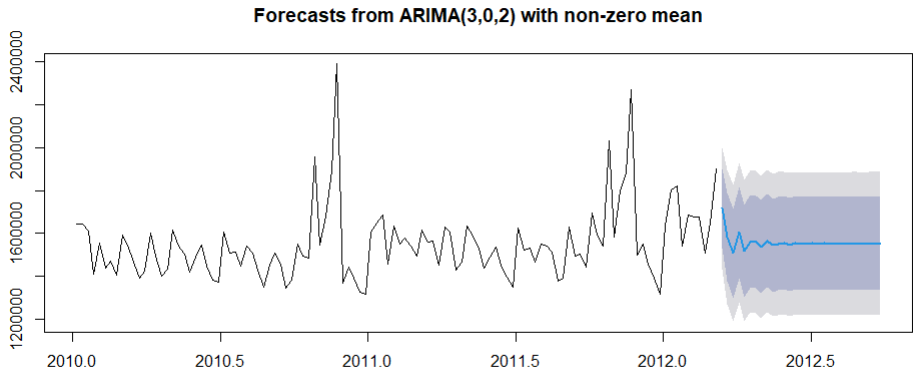

Figure 3.21: Time plot $ARIMA(3, 0, 2)$ for the sales dataset

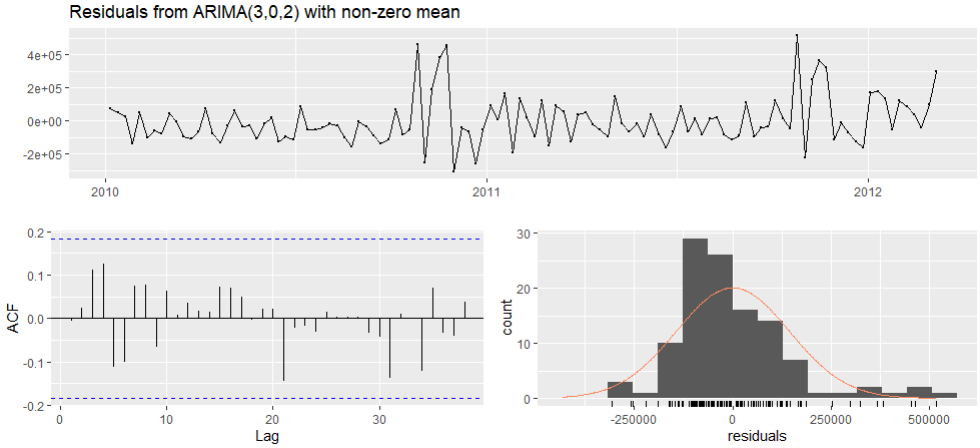

Figure 3.22: Residuals plot for the $ARIMA(3, 0, 2)$ for the sales dataset

decomposition give us a clearer pattern when compared to a simpler ARIMA model. At last, by analyzing the model residuals there is no significant spike in the ACF plot suggesting that the model provides good estimates when forecasting the dataset. This fact is supported by the large p-value resulting from the portmanteau test (0.7618), suggesting that the residuals are white noise.

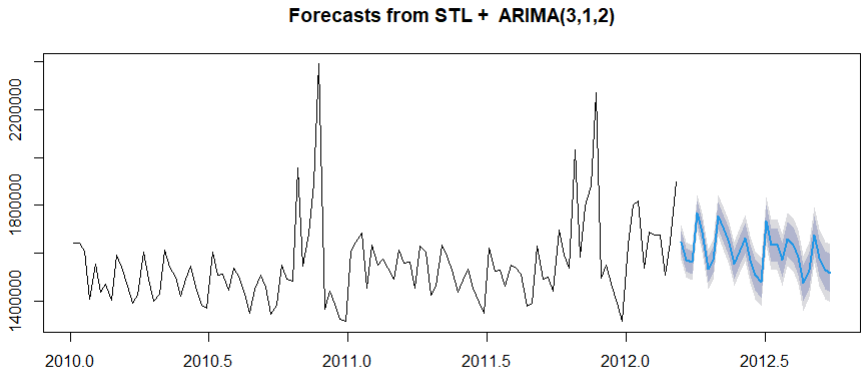

Figure 3.23: $STL + ARIMA(3, 1, 2)$ plot for the sales dataset

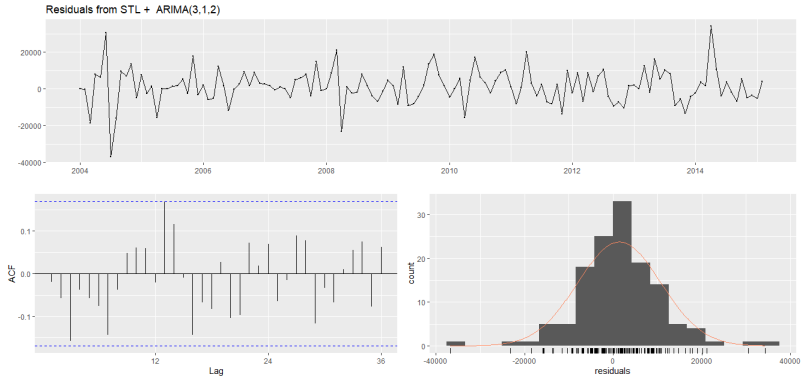

Figure 3.24: $STL + ARIMA(3, 1, 2)$ residuals plot for the sales dataset

3.2.2 SARIMA

As evidenced by the time series plots, the data exhibits a seasonal pattern that needs to be taken into consideration making a non seasonal ARIMA a bad candidate predicting future values. Therefore a seasonal ARIMA should be considered to take into account the seasonality of the data. The analysis of the ACF and PACF plots suggests an $ARIMA(1, 1, 2)(1, 0, 0)$ as a viable model. By testing this model against a couple of variations we get the following results present in table 3.4.

Of the models presented, the one that minimizes the AICc is the ARIMA(1, 1, 2)(1, 0, 0). From observation of figure 3.25 there is one significant spike in the ACF suggesting that the model could have a better accuracy. Altough, with a p-value of 0.27 it passes the Ljung-Box test deeming the model appropriate to forecast the dataset.

The results of the model when applied to the test dataset can be seen in figure 3.26. The model is able to capture the seasonal pattern.

SARIMA Model	AICc
(1,1,2)(1,0,0)	2578.02
(1,0,2)(1,0,0)	2605.89
(1,1,3)(1,0,0)	2579.37
(1,1,2)(0,0,0)	2666.42
(1,1,2)(0,0,1)	2624.31
(1,2,2)(0,0,1)	2608.35

Table 3.4: SARIMA AICc results for the sales dataset

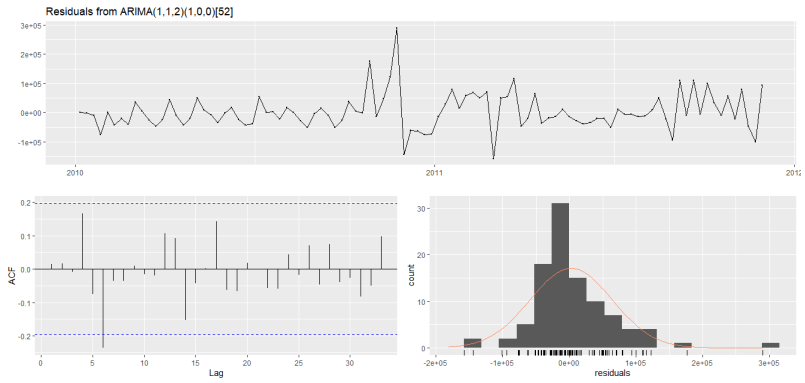

Figure 3.25: Residuals from the $ARIMA(1,1,2)(1,0,0)_{52}$ model applied to the sales data.

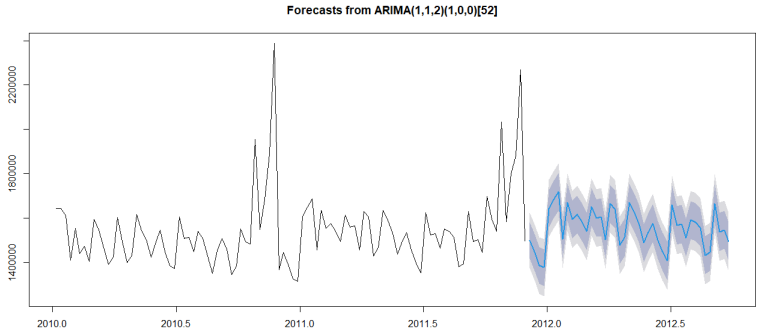

Figure 3.26: Forecast from the $ARIMA(1,1,2)(1,0,0)_{52}$ model applied to the sales data.

3.2.3 Fourier series

Not every forecasting method is appropriate to deal with weekly data. The Holt-Winters method is not capable of handling weekly data and, therefore, an alternative method should be considered. As mentioned by Livera, Hyndman, and Snyder (2010) a Fourier series where the seasonal pattern is modelled using Fourier terms with short-term time series dynamics allowed provides an accurate way to deal with weekly data. The below model explains the mathematical formulation behind the model:

$$y_t = \alpha + \sum_{k=1}^{K}[\alpha_k sin(2\pi kt/m) + \beta_k cos(2\pi kt/m)] + N_t$$

,

where N_t is an ARIMA process. The value of K can be chosen by minimizing the AICc.
A Fourier series has several advantages such as:

- permits any length seasonality;
- in case the data has more than one seasonal period it is possible to include Fourier terms of different frequencies;
- the seasonal pattern is smooth for small values of K
- short-term dynamics are easily handled with a simple ARMA error

Nevertheless, when compared to a SARIMA model it has one disadvantage which is that the seasonality is assumed to be fixed, not allowing the pattern to change over time.

The different combinations of the Fourier series parameters and results can be seen in the following table:

SARIMA Model	K	AICc
(1,1,2)(0,0,1)	4	2618.49
(1,1,2)(0,0,1)	6	2595.99
(1,1,2)(0,0,1)	8	2593.22
(1,1,2)(0,0,1)	8	2569.7
(1,1,2)(0,0,1)	17	2531.18
(1,2,1)(0,0,1)	4	2637.65

Table 3.5: AICc of the different Fourier regressions for the sales dataset

As it can be seen in figure 3.27 there is one significant spike which can evidence that the model may not be the most accurate one to forecast the proposed dataset

The forecast can be seen in figure 3.28

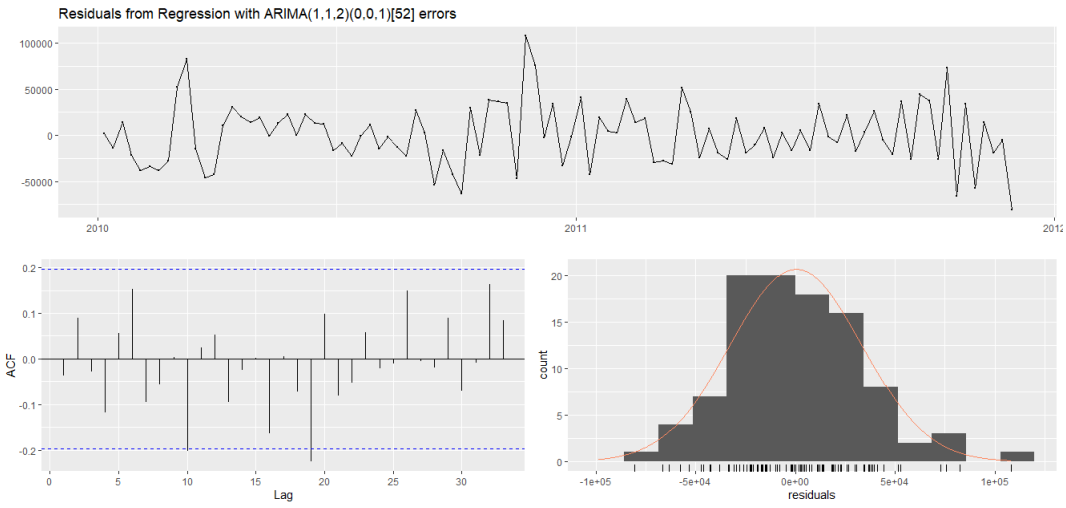

Figure 3.27: Residuals from Regression with ARIMA(1, 1, 2)(0, 0, 1)52 model applied to the sales data.

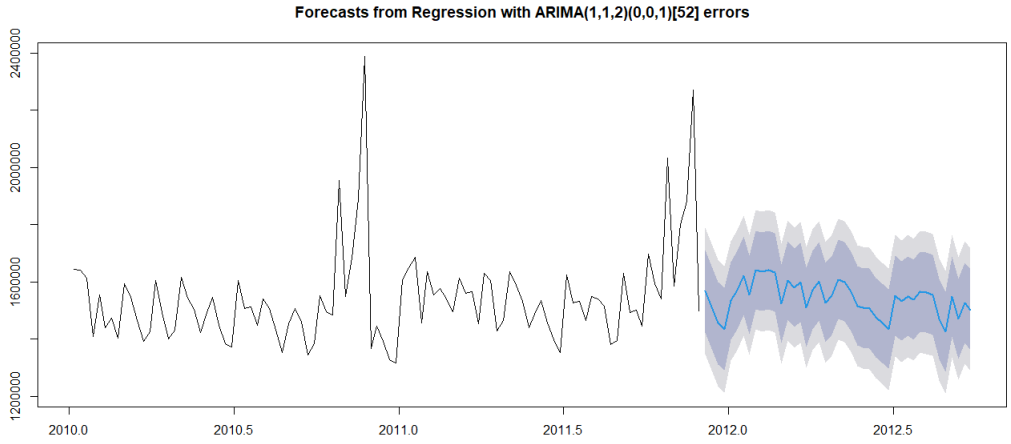

Figure 3.28: Forecast from the Regression with ARIMA(1, 1, 2)(0, 0, 1)52 model applied to the sales data.

3.2.4 LSTM

As evidenced in the dataset above to configure the proper LSTM model different combination of parameters should be considered selecting the one that minimizes the RMSE.

The following table shows the results of different LSTM configurations experimented on the sales dataset.

As it can bee seen the parameters selection that works best when predicting the sales dataset is 4 hidden layers, 200 epochs and a batch size of 1. The forecasts can be observed in figure 3.29, plotting the results against the testing dataset

Hidden nodes	Epoch	Batch Size	Test RMSE
4	100	1	114607.70
8	500	5	116627.80
8	10000	5	148566.27
8	1000	1	116296.01
8	100	10	115726.85
16	100	5	115055.6
4	100	10	114492.62
16	100	10	115775.58
4	200	1	90441.40

Table 3.6: LSTM results for the airport dataset

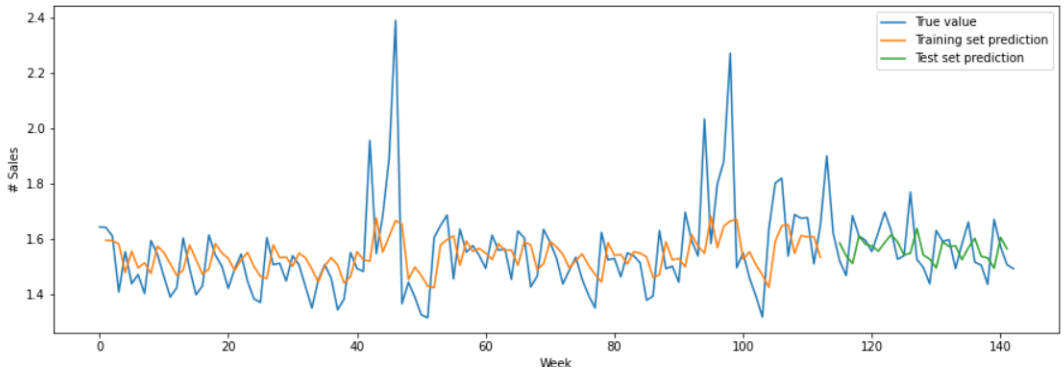

Figure 3.29: LSTM forecast of the sales dataset.

3.3 Comparison and evaluation

As mentioned in the literature review section of this dissertation the selected measure to evaluate the model's accuracy is the Root Mean Square Error (RMSE). The following section explores the different model accuracy for each case study according to the RMSE measure.

3.3.1 Airport dataset

By analyzing the results present in table 3.7 it is clear that the LSTM provides the most accurate model to forecast the airport dataset when compared to simple variations of the ARIMA and Holt-Winters. Nevertheless, given the predictable pattern of this dataset, it could be further transformed to allow for a better fit of the seasonal ARIMA model. Hence, when applying a box-cox transformation to the dataset and training a seasonal ARIMA model over the transformed data we achieve a RMSE considerably lower than other forecasting technique used for this dataset. In fact, the ARIMA models offer a better control of the model than the LSTM models, allowing to better adjust the model to the dataset by considering additional transformations to increase its fitness.

The described results can be seen by looking at the figures with the forecast plots which

evidence a better fit for the SARIMA with Box Cox transformation when compared to other statistical models and LSTM.

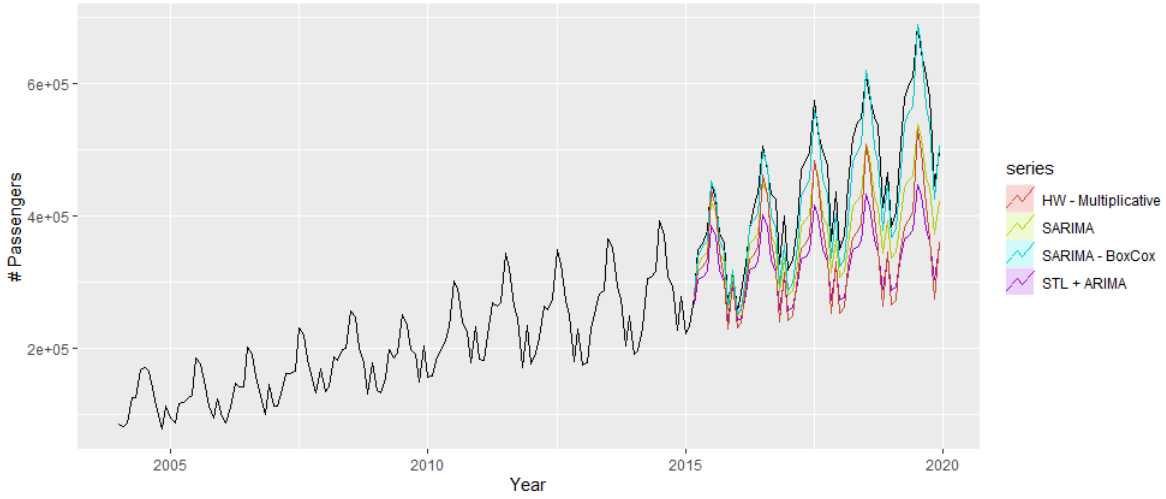

Figure 3.30: Statiscal forecasting plots applied to the airport dataset.

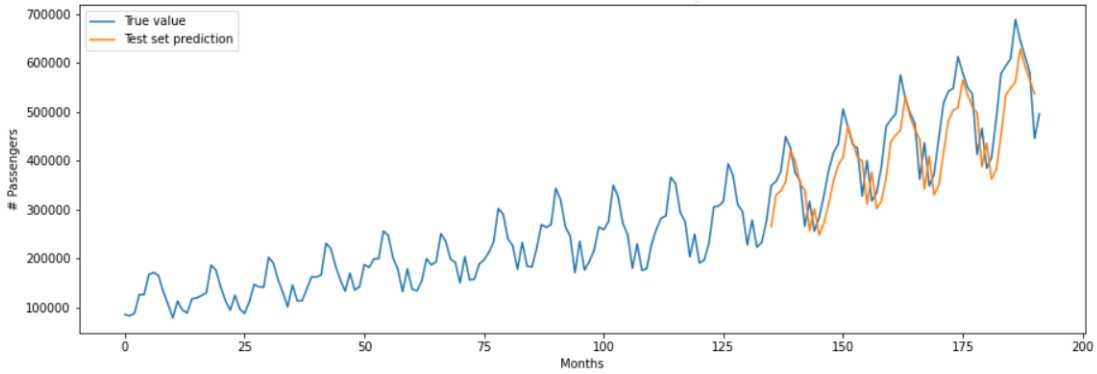

Figure 3.31: LSTM plot applied to the airport dataset.

Airport dataset

Model	Training RMSE	Test RMSE
STL+ARIMA(3,1,2)	9472.63	177274.32
SARIMA(1,1,0)(1,1,0)	11142.25	81293.83
SARIMA(1,1,0)(1,1,0) + BoxCox	11149.86	29767.48
HW Multiplicative	10413.35	115830.25
LSTM	36054.32	62622.86

Table 3.7: Forecast models results for the airport dataset

3.3.2 Sales dataset

As in the Airport dataset, the SARIMA model is the one that better forecasts the data surpassing the LSTM and other statistical models. However, given the unclear pattern of the data neither model is capable of fitting the data with low RMSEs, which would result in significant errors when using for sales prediction in a real case scenario.

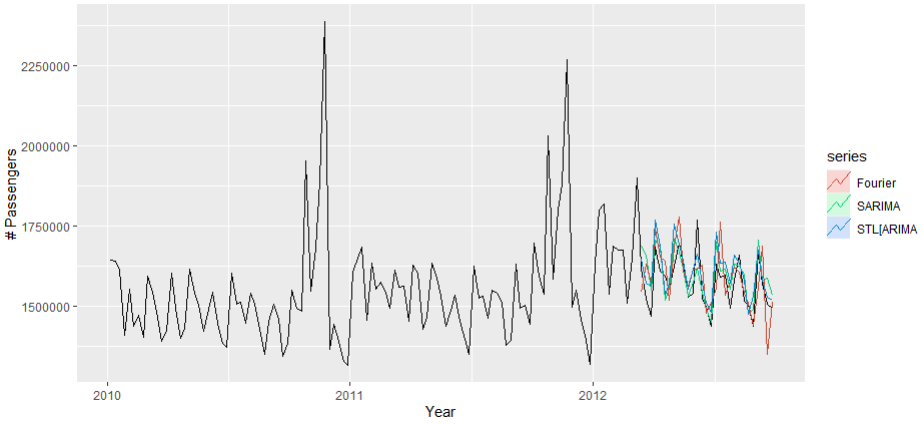

Figure 3.32: Statiscal forecasting plots applied to the sales dataset.

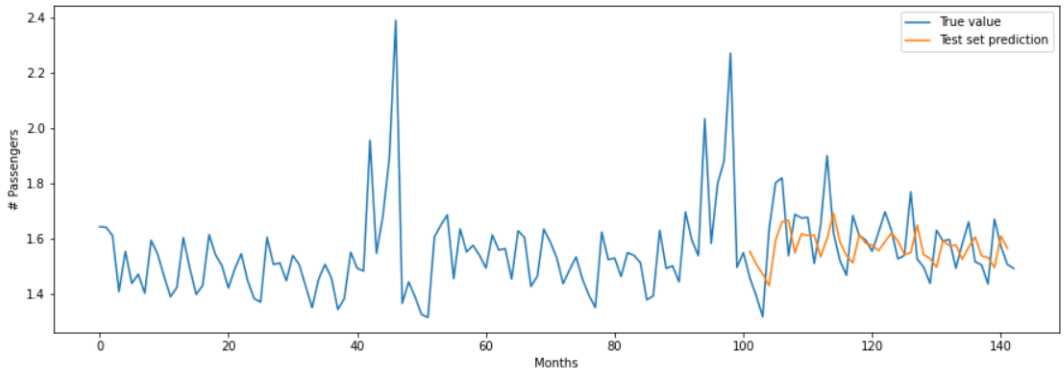

Figure 3.33: LSTM plot applied to the sales dataset.

Sales dataset

Model	Training RMSE	Test RMSE
STL+ ARIMA(3,1,2)	35689.91	61833.33
SARIMA(1,1,2)(1,0,0)	65517.27	61246.93
Fourier(1,1,2)(0,0,1)	38330.63	82499.43
LSTM	152811.30	90441.40

Table 3.8: Forecast models results for the sales dataset

Chapter 4

Final remarks

The main purpose of this dissertation was to compare some of the most commonly used statistical approaches to time series forecasting against a deep learning method that has been gaining popularity in recent years, Long Short Term Memory. To ensure a proper comparison between models, allowing to clearly highlight some of the main advantages and disadvantages of each model, two different datasets were selected.

The airport dataset, the first case study of this dissertation, evidenced a clearly seasonal pattern and an upward trend over a long period of time. When comparing some variations of the ARIMA model and Holt-Winters with the LSTM, the latter yields better results, suggesting that when applied to long term modelling the LSTM may be deemed as a more accurate model. However, when performing additional transformations in the Seasonal ARIMA model this model is capable of surpassing the RMSE of the LSTM. In fact, LSTM models are appropriate to process, predict and classify time series data given time lags of unknown duration. Relative insensitivity to gap length gives an advantage to LSTM over ARIMA models especially when trying to predict long periods of time such as the exercise done in the airport dataset, which considers a five years period. Nevertheless, as evidenced in the first dataset, if the data exhibits a predictable pattern over the length of the time series there are some additional transformations that can be applied to ARIMA models that make this models equal or better to LSTM models when forecasting long term.

The second dataset, was selected due to its more short term nature, with a small amount of data and not so clear patterns. When trying to perform short term forecasts with small amounts of historical data, LSTM models fail to capture the data's pattern, evidencing large RMSE differences when compared with statitiscal models such as a seasonal ARIMA.

In fact, evidenced by the case studies under analysis in this dissertation, statistical models return clearly better results in forecasting short term, whereas LSTM can be more competitive for long term modeling. Additionally, LSTM is without a doubt more complicated and demands a higher level of knowledge and understanding to train and it is not clear that they can exceed the performance of ARIMA models in every situation. Simpler to implement, easier to understand and with no parameter tuning required, ARIMA models offer a quick to run method to forecast time series data that should never be disregarded when forecasting data. Nevertheless, neural networks offer the capability to understand nonlinear relationships with arbitrarily defined but

fixed number of inputs. Recurrent neural networks in particular, can understand what relevant observations it has seen previously and determine its relevancy in forecasting due to its learn and forget characteristics. Since LSTM models are capable of understanding better long term correlations, they can model complex multivariate sequences without having to clarify any time window.

As mentioned on the abstract of this dissertation, one of its main purposes is to contribute to improve the quality of the decision making process within organizations. Therefore, when deciding which method to apply to forecasting problems some decision points need to be taken into consideration, such as, the understandability, accuracy and simplicity of the model. Hence, as evidenced in this dissertation, deep learning methods do not seem to stand to their hype when applied to univariate time series forecasting. Classical methods such as ARIMA and Holt-Winters outperform or do not lag much behind deep learning methods for one-step forecasting on univariate datasets. Hence, despite the recent hype with deep learning they should never be immediately seen as the go to model to forecast time series data. Given the easibility to apply and understandability of most of the statistical methods they can provide quick wins to any company looking to improve its forecasting techniques. At last, even when considering more advanced models, such as deep learning ones, statistical models provide powerful benchmarks that can help improve any technique chosen.

Bibliography

Akaike, H. (1973). *Information theory and an extension of the maximum likelihood principle. in b. n. petrov, f. csaki (eds.), proceedings of the 2nd international symposium on information theory (pp. 267-281). budapest: Akademiai kiado.*

Anzanello, M., & Fogliatto, F. (2011, 09). Learning curve models and applications: literature review and research directions. international journal of industrial ergonomics, 41, 573-583. *International Journal of Industrial Ergonomics*, *41*, 573-583. doi: 10.1016/j.ergon.2011.05.001

Bengio, Y. (2012, 06). Practical recommendations for gradient-based training of deep architectures. *Arxiv*.

Bengio, Y., Simard, P., & Frasconi, P. (1994, 02). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, *5*, 157-66. doi: 10.1109/72.279181

Box, G., & Jenkins, G. C. R. G. M. L., Gwilym M. (2015). *Time series analysis: Forecasting and control (5th ed).hoboken, new jersey: John wiley sons.*

Box, G., & Jenkins, G. M. (1970). *Time series analysis: Forecasting and control.* Holden-Day.

Brockwell, P. J., & Davis, R. A. (2016). *Introduction to time series and forecasting (3rd ed). new york, usa: Springer.*

Cho, K., van Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., & Bengio, Y. (2014, 06). Learning phrase representations using rnn encoder-decoder for statistical machine translation.
doi: 10.3115/v1/D14-1179

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). Automatic time series forecasting: The forecast package for r. *Journal of Official Statistics*, *6*(1), 3–33.

Cowpertwait, P., & Metcalfe, A. (2009, 01). Introductory time series with r.. doi: 10.1007/978-0-387-88698-5

Elsworth, S., & Güttel, S. (2020, 03). Time series forecasting using lstm networks: A symbolic approach.

Gers, F., & Schmidhuber, J. (2000, 02). Recurrent nets that time and count. In (Vol. 3, p. 189 - 194 vol.3). doi: 10.1109/IJCNN.2000.861302

Greff, K., Srivastava, R., Koutník, J., Steunebrink, B., & Schmidhuber, J. (2015, 03). Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, *28*. doi: 10.1109/TNNLS.2016.2582924

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen, technical university munich, institute of computer science.

Hochreiter, S., & Schmidhuber, J. (1997, 12). Long short-term memory. *Neural computation*, *9*, 1735-80. doi: 10.1162/neco.1997.9.8.1735

Holt, C. (1957). Forecasting seasonals and trends by exponentially weighted averages (o.n.r. memorandum no. 52). carnegie institute of technology, pittsburgh usa.

Hyndmam, R. J. (2008). Forecasting functions for time series, r package version 8.12, url https://cran.r-project.org/web/packages/forecast/index.html.

Hyndman, R., & Athanasopoulos, G. (2019). Forecasting: principles and practice, 3rd edition, otexts: Melbourne, australia.

Hyndman, R., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software, Articles*, *27*(3), 1–22. doi: 10.18637/jss.v027.i03

Hyndman, R., & Koehler, A. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679 – 688. doi: 10.1016/j.ijforecast.2006.03.001

Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). An empirical exploration of recurrent network architectures.

Koutník, J., Greff, K., Gomez, F., & Schmidhuber, J. (2014, 02). A clockwork rnn. *31st International Conference on Machine Learning, ICML 2014*, *5*.

Kwiatkowski, D., Phillips, P., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, *54*(1-3), 159-178. Retrieved from https://EconPapers.repec.org/RePEc:eee:econom:v:54:y:1992:i:1-3:p:159-178

Livera, A., Hyndman, R., & Snyder, R. (2010, 01). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, *106*, 1513-1527. doi: 10.1198/jasa.2011.tm09771

Mills, T. C. (2019). Applied time series analysis: A practical guide to modeling and forecasting..

Nielsen, A. (2019). *Practical time series analysis. o'reilly media, inc.*

R. (2020). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from https://www.R-project.org/

Shumway, R. H., & S.Stoffer, D. (2017). Time series analysis and its applications, 4rd edition.

Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management Science*, *6*(3), 324-342. Retrieved from https://doi.org/10.1287/mnsc.6.3.324 doi: 10.1287/mnsc.6.3.324

Yao, K., Cohn, T., Vylomova, K., Duh, K., & Dyer, C. (2015, 08). Depth-gated recurrent neural networks.