Time Series Forecasting with exogenous factors: Statistical vs. Machine Learning approaches

By Cínthia Mendes Castilho

Dissertation
Master in Modeling, Data Analysis and Decision Support System

Supervised by:

Maria Eduarda da Rocha Pinto Augusto da Silva
Luís Fernando Rainho Alves Torgo

2020

"Learn from yesterday, live for today, hope for tomorrow.
The important thing is not to stop questioning."

— Albert Einstein

Biographical note

Cinthia Castilho was born in june 17th 1992, in São Paulo, Brazil, where she obtained the Bachelor degree in Applied Mathematics by University of São Paulo in 2014. In her professional life, she acquired interest in the area of Data Analytics, then in 2018 she applied to the Master in Data Analytics of University of Porto. The Master dissertation, as well the entire course, provided to her valuable knowledge in the analytics field and allowed for the comprehension of more theoretical and practical subjects regarding such area.

Professional with more than five years of experience based on Credit Risk analysis and Reporting for banks. She is currently working as Senior Consultant in the Risk Advisory area at Deloitte.

Acknowledgements

Throughout the writing of this dissertation, I have received a great deal of support and assistance.

I would first like to thank my master thesis supervisors Professor Maria Eduarda Silva and Professor Luis Torgo for their support, assistance, useful critiques, patient guidance, and whose insight and knowledge into the subject matter steered me through this research.

Nevertheless, it is luck for me to meet some friends during the master course who inspirit my effort to overcome all difficult moments. An especial thanks to Nikhil and Isabel.

I would also like to thank my dear friend, Sara, for all the support during the day by day work and for sharing valuable discussions with me.

I would like to express my deep gratitude to my family, in special to my father, who has always encouraged me to follow my goals and believe that I can get as far as I want.

Last but not the least important, I owe more than thanks to my husband Felipe, who contributed immensely to me completing my master's degree. For all the times you did everything you could to give me more time to study, for all the words of encouragement and for believing in me at all times.

Abstract

Time series present great value to several different industries, such as sales, stock market analysis, process and quality control, etc. Considering the increasing availability of information and volume of data available, the challenge of forecasting time series is becoming harder. In this sense, it is crucial to optimize forecasting models, making them more interpretable and valuable for business practitioners. This work aims to analyze the gain of considering exogenous factors as input in the forecasting models. Also, this work compares the performance of the ARIMA model with the performance of three Machine Learning algorithms. Exogenous factors can be relevant to understand the impact of some business actions. Such information brings insights into relationships between variables and allows the understanding and discussion of different scenarios. The proposed approach to integrate exogenous factors is applied to forecast three different univariate time series. The first data set regards sales in a department store, the second bike sharing demand and the third insurance quotations. The models used to produce forecasts include one traditional statistical model, ARIMA, and three Machine Learning methods, Support Vector Machine (SVM), Random Forest (RF) and Extreme gradient boosting (XGBoost).

Overall, the results indicate that considering exogenous factors as inputs in the forecasting models improves their forecasting ability for all the cases studied. Furthermore, for these particular case studies, the Machine Learning models overcome the forecasting ability of the ARIMA model.

Keywords: time series, forecasting, exogenous factors, dynamic regression

Resumo

Séries temporais apresentam grande valor para diversos setores, como vendas, análise de mercado de ações, processos e controle de qualidade, etc. Considerando a crescente disponibilidade de informações e o volume de dados, o desafio de prever séries temporais torna-se cada vez maior. Nesse sentido, é fundamental otimizar os modelos de previsão, tornando-os mais interpretáveis e assertivos para os profissionais de negocios. Este trabalho tem como objetivo analisar o ganho ao se considerar fatores exógenos como variáveis dos modelos de previsão. Adicionalmente, este trabalho compara o desempenho do modelo ARIMA com o desempenho de três algoritmos de aprendizado de máquina. Fatores exógenos podem ser relevantes para entender o impacto de algumas ações de negócios. Tais informações permitem a compreensão e discussão de diferentes cenários. A abordagem proposta para integrar fatores exógenos é aplicada em três diferentes séries temporais univariadas, incluindo vendas, demanda por partilhamento de bicicletas e cotações de seguros. Os modelos usados para gerar as previsões incluem um modelo estatístico tradicional, ARIMA, e três métodos de aprendizado de máquina, Support Vector Machine (SVM), Random Forest (RF) e Extreme gradient boosting (XGBoost).

No geral, os resultados indicam que considerar fatores exógenos nos modelos de previsão melhora a capacidade de previsão para todos os casos estudados. Adicionalmente, para esses estudos de caso específicos, os modelos de aprendizado de máquina superam a capacidade de previsão do modelo ARIMA.

Palavras-chave: séries temporais, previsão, fatores externos, regressão dinâmica

List of Abbreviations

ACF	Autocorrelation Function
CV	Cross-Validation
ML	Machine Learning
PACF	Partial Autocorrelation Function
SVM	Support Vector Machine
SVR	Support Vector Regression
RF	Random Forest
XGBoost	Extreme Gradient Boosting

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Forecasting future values of a time series is one of the most important problems that analysts face in many fields, such as finance, sales, meteorology, politics, etc. Forecasting methods consider historical data and provide estimates for future values of the time series. Since it can be applied to various practical problems in real-world, techniques to predict time series data have been a topic of increasing research activity. Many important models have been proposed in the literature for improving the accuracy and efficiency of time series modelling and forecasting. One of the most known and used time series forecasting models is the ARIMA (Autoregressive Integrated Moving Average) model. However, since the ARIMA models cannot deal with nonlinear patterns or relationships, their approximation of complex real-world problems and dynamics is not always satisfactory. On the other hand, it is possible to find complex patterns in the data dynamics using supervised Machine Learning algorithms. In this sense, several authors analyzed the application of Machine Learning for forecasting time series, among Ahmed, Atiya, Gayar, and El-Shishiny (2010), Cerqueira, Torgo, and Soares (2019) and Makridakis, Spiliotis, and Assimakopoulos (2018). The reported results do not converge to a single conclusion. The accuracy of the generated forecasts depends on the size of the series, the pre-processing methods and the presence of nonlinear components.

Furthermore, strategic planning for the future is crucial in any business to make definitive action plans. The inclusion of additional information than the past values of the target variable in forecasting models can be relevant to understand the impact of some business actions, calendar effects or external events, such as weather or strikes. Such information may explain some of the observed historical variations and then lead to more accurate forecasts. The inclusion of exogenous factors as explanatory variables lends insights into relationships between variables and allows the understanding and discussion of different scenarios. Unusual behaviour observed in the series can be caused by some specific factors, e.g., promotion events, price reduction, weather conditions, etc. If these particular events are repeated periodically, it is possible to add new features which will indicate these special events and describe the extreme or unusual values of the target variable. It is important to highlight that considering such external information into the forecasting model is different from modelling multivariate time series. Nonetheless, exogenous drivers can be determined by factors or variables outside the causal system under study. In other words, variables that affect a model without being affected by it.

Different from classic regression methods, time series models usually do not include additional information than past observations. Some existing studies in economic field consider the ARIMAX model to include such factors as explanatory variables, such as Suhartono and Prastyo (2015) and Alkali (2019). The implementation and the success of a time series model that considers external factors as inputs will require expertise, and it could be challenging to forecast all the predictors when it is necessary.

1.1 Motivation

Although it is intuitive to assume that some exogenous factors can present relevant influence on forecasting models, existing time series related works are not so common, and most of them do not compare the results with alternative time series models, see for instance Bossche and Brijs (2004). Most of the literature that considers alternative methods for time series forecasting aims to evaluate the performance of Machine Learning models on time series data. One strategy used to apply ML models to forecast time series data is to consider lagged values of the response variable as input so that the model can learn the dynamics of the series since supervised Machine Learning models do not have the ability to extrapolate patterns outside of the domain of training data. However, the application is usually made considering data that do not present exogenous factors. Makridakis, Hyndman, and Petropoulos (2019) present the state of the art of forecasting in social settings including, among other topics, the performance of Machine Learning methods on time series forecasting tasks and the added value of including exogenous variables. The authors highlight that more studies are needed to better understand the value of collecting data for exogenous variables and in which domains their inclusion into the forecasting models is likely to practically improve forecasting performance. The inclusion of additional information in forecasting models can contribute to the assertiveness of forecasts, supporting strategic business planning and guiding companies regarding production planning. The lack of a comparative study regarding the inclusion of exogenous factors to predict time series and the applications of Machine Learning models for time series forecasting have motivated this work.

1.2 Objectives

This dissertation has two main goals. The work first analyzes the performance of different forecasting models when selected exogenous factors are considered as inputs. The results are evaluated to assess the impact of additional factors when generating forecasts. Complementary, the performance of Machine Learning models is compared with the traditional ARIMA model to analyze the gain of considering such models for time series forecasting. There were several options of ML models to be applied, and three models were chosen: Support Vector Machine, Random Forest and XGboost. Such models are well-known Machine Learning algorithms and successfully performed in several prediction competitions. Monte-Carlo estimates is applied to guarantee reliable estimates of the performance. The measure considered is this work to evaluate the performance of the models is the Root Mean Squared Error (RMSE). Therefore, the RMSE of the forecasts obtained with or without exogenous factors are compared.

With these purposes, three different case studies are considered. The first case refers to daily sales in Germany's drug store called Rossmann and the goal is to predict the sales turnover for the next 30 days. Sales are affected by many factors, such as weekends, holidays and promotions. The second case present daily observations from bike rental demand in Washington, D.C. The target variable is the count of total rental bikes, and the goal is to predict the number of rentals for the next 30 days. Besides the historical usage patterns, the data combine different exogenous factors involving holidays, season and temperature. The last case studied aims to analyze the effect on monthly insurance quotations when TV advertising is disseminated, assuming that the effect may not be immediate. Hence, different lags of the external predictor are analyzed to found a correlation with the response variable and then be considered as an exogenous factor in the model.

1.3 Organization

This dissertation is divided into 5 chapters, following a logical sequence for the development of the topic addressed. Chapter 1 presents the problem to be studied. Chapter 2 introduces the theoretical foundations used. Chapter 3 describes the models applied in this work. The methodology used and the experimental studies are presented in Chapter 4. Chapter 5 concludes the work, analyzing the results obtained. The appendices contain details of the results obtained for each model regarding the datasets used.

Chapter 2

Background knowledge

This chapter introduces some essential concepts of time series analysis and forecasting, as well as supervised Machine Learning (ML). Furthermore, ML approaches to time series forecasting are described.

2.1 Time series

This section introduces the essential concepts of time series to support this dissertation. A time series is an ordered sequence of observations obtained over time. In this work we consider that the observations are obtained at equally spaced time intervals. A time series can be set as univariate or multivariate. In the first case, the series contains observations of a single variable, and in the latter, there are observations of more than one variable. In this work, only univariate time series are considered. Moreover, a time series are said in continuous time, if observations are measured at every instance of time, or discrete time if observations are measured at discrete points of time. Time series analysis is considered for many applications, such as Sales, Economics, Stock Market, Census analysis and many more. In general, time series analysis involves the following interests:

- understand the underlying process that generates the series, obtaining information about the series behaviour and looking for relevant patterns present in the data;
- forecast future observations of short, medium or long term to support business decisions.

Formally, a time series is a set of random variables indexed in time and denoted by $y_t, t = 0, \pm 1, \pm 2, ...$ In general, a collection of random variables, y_t, indexed by t, is referred to as a stochastic process. In a stochastic process, the mean, autocovariance and autocorrelation can be defined respectively as

$$\mu_t = E(y_t) \tag{2.1}$$

$$\gamma(s,t) = cov(y_s, y_t) = E[(y_s - \mu_s)(y_t - \mu_t)] \tag{2.2}$$

$$\rho(s,t) = cov(y_s, y_t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}} \tag{2.3}$$

An important concept in time series analysis is that of stationarity. There are two types of stationary processes which can be defined as (i) Strictly Stationary and (ii) Weakly Stationary.

Strictly Stationary: the probabilistic behavior of every collection of variables $(y_{t_1}, y_{t_2}, ..., y_{t_k})$ is identical to that of the time shifted set $(y_{t_1+h}, y_{t_2+h}, ..., y_{t_k+h})$. That is, for all k = 1, 2, ..., all time points $t_1, t_2, ..., t_k$, all numbers $c_1, c_2, ..., c_k$ and all time shifts $h = 0, \pm 1, \pm 2$

$$P(y_{t_1} \leq c_1, ..., y_{t_k} \leq c_k) = P(y_{t_1+h} \leq c_1, ..., y_{t_k+h} \leq c_k) \tag{2.4}$$

Weakly Stationary: time series is a finite variance process where:

1. $E[y_t] = \mu$
2. $var(y_t) = E[(y_t - \mu)^2] = \sigma^2$
3. $cov(y_s, y_t) = \gamma(|s - t|)$

In a stationary process, the statistical properties such as mean and variance do not depend upon time whereas the autocorrelation function depends only on the lag. The concept of stationarity is important for meaningful estimation of mean, autocovariance and autocorrelation from data. The autocorrelation in time series represents the correlation between observations of the same series at different lags (points in time). An important aspect of autocorrelation analysis is how it can help to discover hidden patterns in the data and support the selection of the correct forecasting methods. Consider r_1 as the measure of the relationship between y_t and y_{t-1}, r_2 as the measure of the relationship between y_t and y_{t-2} and so on. Thus, r_i can be defined as the autocorrelation function (ACF) and written as

$$r_i = \frac{\sum_{t=i+1}^{T}(y_t - \overline{y})(y_{t-i} - \overline{y})}{\sum_{t=1}^{T}(y_t - \overline{y})^2} \tag{2.5}$$

where T is the length of the time series.

The partial autocorrelation function (PACF) measures the linear correlation between y_t and y_{t+i} with the linear dependence of $(y_{t-1}, y_{t-2}, \ldots, y_{t-(i-1)})$ removed. It differs from ACF, since the autocorrelation function does not measure the relationship for other lags. Moreover, analyzing the autocorrelation function (ACF) and partial autocorrelation function (PACF) in conjunction support the selection of the appropriate ARIMA model for time series prediction.

TIME SERIES COMPONENTS

In general, four main components of a time series may be distinguished: Trend, Cyclical, Seasonal and Irregular components. A brief description of each component is given as follows based on Adhikari and Agrawal (2013).

Trend

A trend can be defined as a long term movement in a time series, representing the general tendency of a time series to increase, decrease or stagnate over a long period of time.

Cyclical

It describes the medium-term changes in the series, generated by circumstances, which repeat in cycles. A cycle's length stretches over a more extended period of time. Most of the economic and financial time series show some kind of cyclical variation.

Seasonal

Seasonal variations in a time series represent fluctuations within a year during the season. Some important factors causing seasonal variations are climate and weather conditions, traditional habits, etc. Seasonal variation is an important factor for making proper future plans.

Irregular components

Irregular variations in a time series are generated by unpredictable influences, which are not regular and also do not repeat in a unique pattern. These variations are caused by incidences such as war, strike, earthquake, revolution, etc.

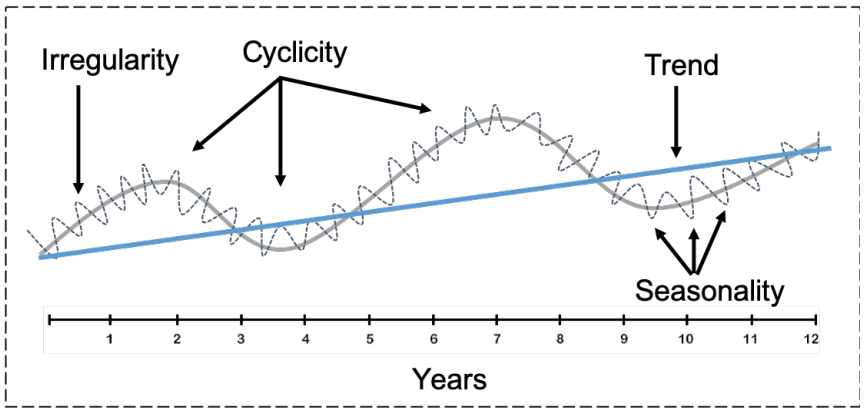

Figure 2.1: An illustration of the four main components of a time series adapted from Datavedas (2018).

Considering the effects caused by these four components, two different types of models are generally considered: Multiplicative and Additive models. The multiplicative model assumes that the elements of a time series are not independent, and they can affect one another. On the other hand, the additive model assumes independence among all the components. Thus, it is possible to write both models as follows, where T, S, C and E are respectively the trend,

seasonal, cyclical and irregular variation at time t. Note that it is possible to obtain an additive model from a multiplicative one by applying the logarithmic function to the model.

$$\text{Multiplicative Model: } y_t = T_t * S_t * C_t * E_t \tag{2.6}$$

$$\text{Additive Model: } y_t = T_t + S_t + C_t + E_t \tag{2.7}$$

2.2 ARIMA Models

ARIMA models are one of the most widely used approaches to time series forecasting. Despite the fact that the inclusion of relevant external information as predictor variables is allowed in regression models in general, ARIMA models handle the dynamics of time series data. This section introduces the AutoRegressive Integrated Moving Average model, ARIMA. Further, it describes the algorithm proposed by R. Hyndman and Athanasopoulos (2018) for automatic modelling of a time series with the best ARIMA model. The main references for this section are Chatfield (2000) and Cryer and Chan (2008).

A pth-order autoregressive process y_t, known as $AR(p)$, satisfies the equation

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t \tag{2.8}$$

where the parameters ϕ_i are such that the roots, $z_1, \ldots, z_p$ of the polynomial $\phi(z) = 1 - a_1 z - \ldots - a_p z^p$ are outside the unit circle, $|z_i| > 1$, to guarantee stationarity. The current value of y_t is a linear combination of the past p values plus a term ϵ_t that incorporates everything new in the series at time t and that is not explained by the past values. The term ϵ_t is a white noise process or shock process, that is a sequence of independent and identically distributed random variables, with zero mean and variance σ_ϵ^2. Thus, for every t, it is assumed that ϵ_t is independent of $y_{t-1}, y_{t-2}, y_{t-3} \ldots$.

A time series y_t is said to be a moving average process of order q, known as MA(q), if it is a linear combination of the last q forecast errors in a regression model, such as

$$y_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} \tag{2.9}$$

where the parameters θ_i are such that the roots, $z_1, \ldots, z_q$ of the polynomial $\theta(z) = 1 - b_1 z - \ldots - b_q z^q$ are outside the unit circle, $|z_i| > 1$, to guarantee invertibility.

It is possible to write any stationary AR(p) model as an MA(∞) model. For instance, using repeated substitution, it is possible to show this for an AR(1) model:

$$\begin{aligned} y_t &= \phi_1 y_{t-1} + \epsilon_t \\ &= \phi_1(\phi_1 y_{t-2} + \epsilon_{t-1}) + \epsilon_t \\ &= \phi^2 y_{t-2} + \phi_1 \epsilon_{t-1} + \epsilon_t \\ &= \phi^3 y_{t-3} + \phi^2 \epsilon_{t-2} + \phi_1 \epsilon_{t-1} + \epsilon_t \\ &\textit{etc.} \end{aligned} \tag{2.10}$$

Provided $-1 < \phi_1 < 1$, the value of ϕ_1^k will get smaller as k gets smaller. Therefore, it is possible to obtain a MA(∞) process as

$$y_t = \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_1^2 \epsilon_{t-2} + ... \tag{2.11}$$

Hence, it is possible to write any invertible MA(q) process as an AR(∞) process. In an invertible process, the most recent observations have higher weight than observations from the more distant past. The ARMA model is simply the mix between AR(p) and MA(q) models. The condition on the coefficients to ensure stationarity and invertibility is that the roots of the AR and MA polynomials are outside of the unit circle.

In real applications, many time series are non-stationary and then it is not possible to apply directly none of the processes described above. One alternative to handle not stationary series is to apply differencing to make them stationary:

$$y'_t = \phi_1 y'_{t-1} + \phi_2 y'_{t-2} + ... + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} + \epsilon_t \tag{2.12}$$

where y'_t is the differenced series. Consider the backward shift operator B, that can be defined as $By_t = y_{t-1}$. The backward shift operator is convenient for describing the process of differencing. The dth differences can be written as $(1-B)^d y_t$. If the original data is differenced d times before fitting an ARMA(p, q) process, then the model for the original data is an ARIMA(p, d, q) process where d denotes the number of differences taken. Thus, a non-seasonal ARIMA model can be written as

$$\phi(B)(1-B)^d y_t = \theta(B)\epsilon_t \tag{2.13}$$

where ϵ_t denotes a random process with zero mean and variance σ_z^2, B is the backshift operator, and $\phi(B)$ and $\theta(B)$ are polynomials in B of finite order p and q respectively. Table 2.1 provides examples of some special cases of ARIMA models.

Model	ARIMA orders
White noise	ARIMA(0,0,0)
Random walk	ARIMA(0,1,0) with no constant
Random walk with drift	ARIMA(0,1,0) with a constant
Autoregression	ARIMA(p,0,0)
Moving average	ARIMA(0,0,q)

Table 2.1: Examples of special cases of ARIMA models from R. Hyndman and Athanasopoulos (2018)

If the series are seasonal, with s time periods per year, then a seasonal SARIMA(p,d,q)(P,D,Q)$_s$ process can be written as

$$\Phi(B)\phi(B^s)(1-B)^d(1-B^s)^D y_t = \theta(B)\Theta(B^s)\epsilon_t \tag{2.14}$$

where Φ, Θ denote polynomials in B^s of order P, Q, respectively.

AUTO-ARIMA

When fitting autoregressive and moving average models, the main difficulty is evaluating the order of the process rather than estimating the coefficients. With ARIMA models, there is an extra problem in choosing the required order of differencing. To handle these type of issues, some formal procedures are available, such as testing for the presence of a unit root and visual analysis of the correlogram. However, there are several alternatives to automate ARIMA modelling, such as described in Hannan and Rissanen (1982) and Gomez and Maravall (2000). The method chosen to be used in this work is proposed by R. Hyndman and Athanasopoulos (2018) and will be described in the following.

The main task in automatic ARIMA forecasting is selecting an appropriate model order, that is the values of p, d, q, P, D, q . If d and D are known, it is possible to select the orders p, q, P and Q by an information criterion such as the AIC (R. Hyndman & Khandakar, 2008):

$$AIC = -2log(L) + 2(p + q + P + Q + k) \tag{2.15}$$

where L is the maximized likelihood of the model fitted to the differenced data. In order to choose d and D, the algorithm uses unit-roots tests based on a null hypothesis of no unit-root.

For non-seasonal data, an ARIMA(p, d, q) model is considered, where d is selected based on successive KPSS unit-root tests (Kwiatkowski, Phillips, Schmidt, & Shin, 1992). That is, keep testing the data for a unit root; if the test result is significant, it is necessary to test the differenced data for a unit root; and so on. The stop criteria of this procedure is obtained when the first not significant result is achieved. For seasonal data, a SARIMA(p, d, q)(P, D, Q)$_s$ model is considered, where s is the seasonal frequency and $D = 0$ or $D = 1$ depending on an extended Canova-Hansen test (Canova & Hansen, 1995). Then, after D is selected, d is selected by applying successive KPSS unit-root tests to the seasonally differenced data (if $D = 1$) or the original data (if $D = 0$). Once d and D are selected, the procedure selects the values of p, q, P and Q by minimizing the AIC. Figure 2.2 illustrates the algorithm.

In summary, this algorithm combines unit root tests and minimization of the AICc to obtain the parameters of an ARIMA model. Since this procedure selects the best parameters for a specific ARIMA model, and it is not a model itself, it will be called as Meta-ARIMA model in this work.

This algorithm is implemented in the package "Forecast" (see R. J. Hyndman & Khandakar, 2020) from Software R (R Core Team, 2013) by a function called *auto.arima*.

FORECASTING WITH ARIMA MODELS

In time series forecasting, the goal is to predict h steps-ahead, y_{t+h}, with h=1,2,..., based on the observed data. Considering the ARIMA model described in section 2.2, that is, $\phi(B)(1 - B)^d y_t = \theta(B)\epsilon_t$, R. Hyndman and Athanasopoulos (2018) describe the process of generate the predictions in three steps:

Hyndman-Khandakar algorithm for automatic ARIMA modelling

1. The number of differences $0 \leq d \leq 2$ is determined using repeated KPSS tests.
2. The values of p and q are then chosen by minimizing the AICc after differencing the data d times. Rather than considering every possible combination of p and q, the algorithm uses a stepwise search to traverse the model space.

a. Four initial model are fitted:
   - ARIMA (0,*d*,0)
   - ARIMA (2,*d*,2)
   - ARIMA (1,*d*,0)
   - ARIMA (0,*d*,1)

A constant c is included unless d=2. If d ≤ 1, an additional model is also fitted:

   - ARIMA(0,*d*,0)(0,*d*,0) without a constant.

b. The best model (with the smallest AICc value) fitted in step (a) is set to be the "current model".

c. Variations on the current model are considered:
   - Vary *p* and/or *q* from the current model by ±1;
   - Include/exclude *c* from the current model.

The best model considered so far (either the current model or one of these variations) becomes the new current model.

d. Repeat Step 2(c) until no lower AICc can be found.

Figure 2.2: Hyndman-Khandakar algorithm for automatic ARIMA modelling from R. Hyndman and Athanasopoulos (2018)

1. Expand the ARIMA equation so that y_t is on the left hand side and all other terms are on the right;
2. Rewrite the equation by replacing t with $T + h$;
3. On the right hand side of the equation, replace future observations with their forecasts, future errors with zero, and past errors with the corresponding residuals.

The process starts with $h = 1$ and then all the steps are repeated for the next points, $h = 1, 2, \ldots$. Thus, considering $\hat{y}_n(h)$ as the generated point forecast, the forecast error at the ith step-ahead is given by $e_n(i) = y_{n+i} - \hat{y}_n(i)$. The criterion to compute $\hat{y}_n(h)$ is to minimize the mean squared error, described as

$$\sum_{i=1}^{h} (\hat{y}_n(i) - y_{n+i})^2 \tag{2.16}$$

Regarding the forecasting strategy, there are two possible: the direct forecasting and the recursive forecasting. The former strategy is less efficient than the latter, which estimates only

one model and, as the name suggests, reuses it h times. Recursive, in this case, means that there is a feedback of each forecast as input back to the model to get the next prediction. ARIMA models utilize recursive forecasting.

2.3 Supervised Machine Learning

Machine Learning is a field of computer science, focusing on developing algorithms that can access data and use it to learn for themselves. In 1959, Arthur Samuel, a pioneer in the field of computer gaming and artificial intelligence, defined Machine Learning as a "Field of study that gives computers the ability to learn without being explicitly programmed" (Alpaydin, 2020). Some examples of Machine Learning applications are:

- Virtual Personal Assistants, like Siri, Alexa and Google Now.
- Email Spam and Malware Filtering.
- Product Recommendations.
- Online Fraud Detection.
- Online Transportation Networks

It is possible to divide ML taks into four types of learning: supervised, semi-supervised, unsupervised and reinforcement. This work addresses only supervised learning. In the following, a brief explanation of this method will be described with the support of Torgo (2017). The goal of a supervised learning algorithm is to find a model that relates a target variable with a set of independent variables, that can be predictors or attributes. This model can be defined as an approximation of an unknown function $y = f(x_1, \ldots, x_p)$ that describes the relationship between the target variable y and the independent variables $x_1, x_2, \ldots, x_p$. The task of supervised learning is to obtain the model parameters that optimize a certain defined criterion, for instance, minimize the prediction error of the model. This search is executed with the support of a sample of observations of the problem involved, that is, it is based on a dataset that contains examples of the learning concept to be mapped. These examples are particular instances of the variables $x_1, x_2, \ldots, x_p$. Thus, if the target variable is continuous, the task will me a regression problem and if the target variable is no nominal, the task will be a classification problem. Figure 2.3 illustrates a general scheme of a supervised learning model.

Considering data containing historical records of a target variable, the problem of time series forecasting can be formulated as a problem of supervised learning. In particular, for one-step forecasting, the t previous values of the target variable are available, and the forecasting problem can be represented in the form of a generic regression problem. In Figure 2.4, the estimate $\hat{f}$ returns the prediction of the value of the time series at time $h = t + 1$ as a function of the t previous values (Bontempi et al., 2013).

Regarding the application of ML methods to time series problems, one of the biggest challenges is that many ML algorithms cannot extrapolate patterns outside of the domain of training

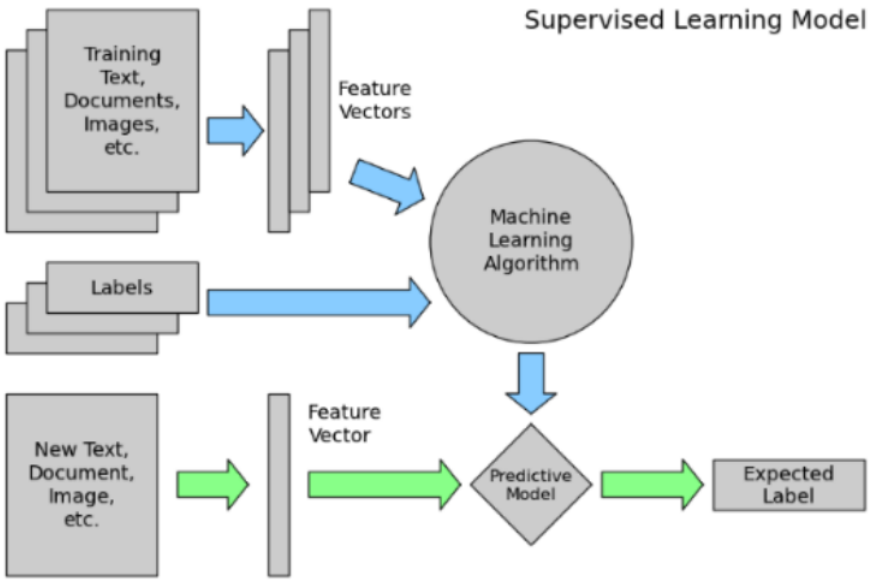

Figure 2.3: Supervised Learning Model from Nasteski (2017)

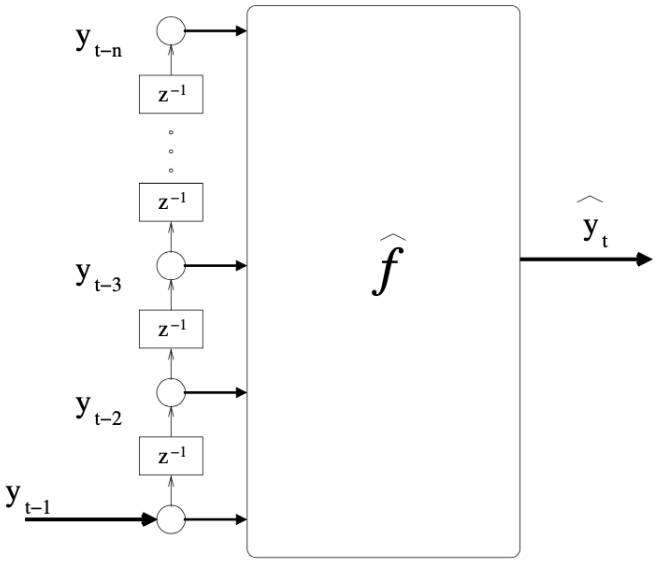

Figure 2.4: One step forecasting as supervised learning from Bontempi et al. (2013). The boxes with z^{-1} represents a unit delay operator: $y_{t-1} = z^{-1}y_t$. Note that the usual notation is $z^{-1} = B$.

data. Further, traditional models used in time series analysis have the ability to generate confidence intervals. Conversely, most Machine Learning models do not present this ability since they are not all based on statistical distributions.

Nevertheless, and as exposed above, one way to convert a time series problem into a super-

vised learning problem is using previous time steps as input variables to predict the next time step(s). This is the basic idea of time delay embedding technique (Takens, 1981). This method consists of describing the state of the dynamic system that generates the observed time series values by a set of l recently observed values. The value l is the dimension of the embedding and it is necessary to determine it. Assuming that the time dependency among the series observations is not larger than the embed size l, using such simple pre-processing in the data would ensure the application of any standard modeling method, since the time dependencies among the l past values can be modeled as relationships between the variables of the new dataset (Torgo, 2017).

In this work, to determine how many lags of the target variable should be consider as inputs of the supervised ML models, that is, the embed size l, the autocorrelation (ACF) and partial autocorrelation (PACF) functions were analyzed to identify the significant lags of the variable to be forecasted.

One contribution of using machine learning models for time series forecasting is that they allow for the inclusion of important features on the time series. In this sense, it is possible to add valuable information to data, such as Day of the week, Weekend, Holiday, Season of year, etc. This information could capture some critical component of the series. Therefore, it becomes possible to gain insights into the time variables that are influential on the data.

2.4 Machine Learning applications in Time Series Forecasting

Machine Learning models have been established themselves as contenders to classical statistical models in the area of forecasting. This section describes the existing literature regarding the application of ML methodologies to time series forecasting.

There are several existing works regarding the usage of neural networks for time series forecasting. Zhang, Patuwo, and Hu (1998a) present a survey of ANN applications in forecasting. Zhang et al. (1998a) discuss how effective are neural networks at forecasting and prediction, analyzing the performance of several studies. Zhang and Qi (2005) investigate the issue of how to effectively model time series with both seasonal and trend patterns considering neural networks. Zhao (2011) discusses benefits and problems regarding the use of neural networks in time series forecasting. Moreover, several studies compare neural networks with traditional linear models for forecasting problems, such as Sharda and Patil (1992), Swanson and White (1995) and "Approximate Nonlinear Forecasting Methods" (2006).

Ahmed et al. (2010) present a comparison study of Machine Learning models for time series forecasting. This work aims to predict one-step ahead of the monthly M3 time series competition data, series that present a variety of applications, such as micro, industry, finance, demographic, among others (Makridakis & Hibon, 2000). The models considered are multilayer perceptron, Bayesian neural networks, radial basis functions, generalized regression neural networks, K-nearest neighbour regression, CART, Support Vector Regression and Gaussian processes. In addition, the authors have tested different pre-processing methods on time series data and then analyzed the performance of the applied models, such as log transformation, deseasonalization, differencing and scaling. The authors conclude that the different Machine Learning models differ

significantly and the results can vary depending on the features of the time series, namely: the size of the sample and the level of the noise. Moreover, they observed that pre-processing can have a large effect on model performance and differencing the series usually resulted in much worse performance. Bontempi et al. (2013) present an overview of ML approaches in time series forecasting regarding three aspects: the formalization of one-step forecasting problems as supervised learning tasks, the discussion of local learning techniques as a useful tool for dealing with temporal data and the strategies for multi-step time series forecasting.

Makridakis et al. (2018) evaluate the performance of ML models over multiple forecasting horizons using the same data as Ahmed et al. (2010), the monthly series from M3 competition. The main goal of this article is to determine, empirically, if the performance of ML models exceeds the statistical ones and how the usage of these approaches could be exploited to improve forecasting accuracy. After comparing the accuracy of standard ML methods with the traditional statistical models, they found that the former are dominated over both accuracy measures used and for all forecasting horizons examined. The models applied were the same used in Ahmed et al. (2010) plus two additional methods: Recurrent Neural Network and the Long Short Term Memory network. Hence, the authors point out that other studies regarding the application of ML models to forecasting time series present a series of limitations. For instance, the methods are evaluated for short-term forecasting horizons, often one-step-ahead, not considering medium and long-term periods, and no benchmarks are used to compare the accuracy of ML methods with alternative approaches. In this sense, Makridakis et al. (2018) extend the study of Ahmed et al. (2010), including statistical methods to compare the accuracy of the ML methods, using three different approaches for obtaining 18-step-ahead forecasts and add the analysis of a computational complexity measure to determine the computational time required by each method to fit the model and obtain the forecasts. According to the results of their study, Machine Learning models need to become more accurate, requiring less computer time, and be less of a black box. The author suggests that the conclusions may be different if nonlinear components are present. In such cases, the highly flexible of ML methods could offer a significant advantage over the traditional statistical ones. Regarding pre-processing to apply ML models for time series data, Makridakis et al. (2018) affirm that the literature of ML is divided with some studies claiming that ML methods are capable of effectively modelling any type of data pattern and can, therefore, be applied to the original data (see Zhang, Patuwo, & Hu, 1998b). However, other studies concluded the opposite, claiming that without appropriate pre-processing, ML methods may become unstable, compromising the results (see Zhang & Qi, 2005).

Cerqueira, Torgo, and Soares (2019) compare traditional statistical methods with Machine Learning models for time series forecasting, controlling for sample size. The models considered are the Auto-Regressive Integrated Moving Average model (ARIMA), a seasonal random walk forecasting benchmark (Naive2), Theta (equivalent to simple exponential smoothing with drift), the exponential smoothing model (ETS) and an exponential smoothing state-space model with Box-Cox transformation, ARMA errors, trend and seasonal components (Tbats). Their results oppose Makridakis et al. (2018) conclusions, suggesting that the size of the analyzed data matters and ML methods improve the performance as the sample size grows. The authors point out that depending on the data size, the sample may not be representative of the process that generates the underlying time series. They used 90 univariate time series from several domains of application

and applied different methodologies to perform one and multi-step ahead forecasts. The pre-processing steps were similar to those of Makridakis et al. (2018). Further, they have included an analysis of the computational complexity of each method, evaluating the computational time spent by a model to complete the prequential procedure applied.

Gilliland (2019) is a recent work that reflects on the results of the M4 forecasting competition, and in particular, the impact of Machine Learning methods for time series forecasting. The author exposes that utilizing ML models in combination with traditional statistical methods or a hybrid model with exponential smoothing not only exceeded the benchmark but performed at the top. Bojer and Meldgaard (2020) provide an overview of what the forecasting community can learn from Kaggle's forecasting competitions, identifying six recent and relevant forecasting competitions and presenting the top-performing solutions. They reported strong performance of gradient boosted decision trees, increasing success of neural networks for forecasting, and a variety of techniques for adapting ML algorithms to the forecasting task.

2.5 Measures of forecasting accuracy

The accuracy of forecasts is essential in all applied fields. The more accurate are the forecasts, the more confidence users have in the model and the forecasting process. In general, forecast errors measures evaluate the quality of forecasting methods and choose the best forecasting model in determining application. Shcherbakov et al. (2013) present a survey of forecast error metrics, reviewing the commonly used forecast error measurements. According to them, forecasts measures can be divided into groups: absolute forecasting errors, measures based on percentage errors, measures based on relative errors, scaled errors, etc.

It is possible to define a forecast error as the difference between an observed value and its forecast. It is illustrated in equation (2.17), where the forecasted data is given by $(\hat{y}_{t+1}, \hat{y}_{t+2}, \ldots)$ and the observed data is given by $(y_{t+1}, y_{t+2}, \ldots)$.

$$e_{t+h} = y_{t+h} - \widehat{y}_{t+h|t} \tag{2.17}$$

There are some well-known metrics usually considered, as Root Mean Squared Error (RMSE), the Mean Absolute Percentage Error (MAPE) and the Mean Absolute Scaled Error (MASE).

Root Mean Squared Error (RMSE)

The root mean squared error is a standard measure used for scale-dependent data. When comparing forecasts of a single time series, or two distinct time series with the same scale units, the RMSE is popular as it is easier to understand and compute. It measures the accuracy of the prediction by averaging the square error value of each observed error. It is useful when measuring prediction errors in the same unit as the original series. The metric is defined as follows, where e_i is the forecast error.

$$RMSE = \sqrt{mean(e_t^2)} \tag{2.18}$$

Mean Absolute Percentage Error (MAPE)

The percentage errors are the common in forecasting domain and are calculated based on the value P_t, where $P_t = \frac{100e_t}{y_t}$. The MAPE metric belongs to this group and is described as follows:

$$MAPE = mean(|P_t|) \tag{2.19}$$

Shcherbakov et al. (2013) pointed out some disadvantages of this group of measures, such as if the observed value is equal to the forecasted value, but with opposite sign, or both of them are zero, then an error occurs when dividing by zero. These metrics are also affected by outliers as in the errors aforementioned.

Mean Absolute Scaled Error (MASE)

Scale errors were proposed by R. Hyndman and Koehler (2006) as an alternative to percentage errors to compare forecasts accuracy's across time series with different scales units. Thus, a scaled error is defined as follows:

$$q_i = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|} \tag{2.20}$$

Since both the numerator and denominator involve values on the scale of the original data, q_i is independent of the scale of the data. Then, the mean absolute scaled error (MASE) is defined as

$$MASE = mean(|q_i|). \tag{2.21}$$

In this work, the Root Mean Squared Error (RMSE) is used to evaluate the models.

Chapter 3

Forecasting models with exogenous factors

The inclusion of additional information in forecasting models can be relevant to understand the impact of some business actions, calendar effects or external events, such as weather information or e.g. strikes. Such information may explain some of the observed historical variations and then lead to more accurate forecasts. The main advantages of including exogenous factors as explanatory variables in the forecasting model can be summarized as follows:

- to lend insights into relationships between variables;
- to allow the understanding and discussion of different scenarios.

In this chapter, the forecasting models used to considered such extra information in this work are introduced.

3.1 Dynamic Regression Models

Classical regression models consider independent and identically distributed observations from the variable of interest, meaning that that the errors e_t are uncorrelated. On the other hand, time series are characterized by serial correlation and therefore the assumption of uncorrelated errors is not appropriate. Thus, to consider regression models for time series data it is necessary to allow these errors to be correlated. These type of models are known as dynamic regression models. In this section the main concepts regarding these model are presented following Shumway and Stoffer (2016) and R. Hyndman and Athanasopoulos (2018).

Consider regression models of the form

$$y_t = \beta_0 + \beta_1 x_{1,t} + ... + \beta_k x_{k,t} + \eta_t \tag{3.1}$$

where y_t is a linear function of the k predictors variables $(x_{1,t}, ..., x_{k,t})$ and η_t represents the correlated errors. In ordinary regression, η_t is assumed to be white noise.

In time series data, it is convenient to assume that the error process η_t is stationary and well modelled by an ARMA process. To illustrate that, consider a pure $AR(p)$ error, then

$$\phi(B)\eta_t = \epsilon_t \tag{3.2}$$

and $\phi(B) = 1 - \phi_1 B - \ldots - \phi_k B^p$ is the linear transformation that, when applied to the error process, generates the white noise ϵ_t. Multiplying the regression equation expressed in (3.1) by the transformation $\phi(B)$ produces

$$\phi(B)y_t = \sum_{i=1}^{k} \beta_i \phi(B) x_{t,i} + \phi(B)\eta_t \tag{3.3}$$

which is a linear regression model where the observations have been transformed so that $y_t^* = \phi(B)y_t$ is the independent variable, $x_{t,i}^* = \phi(B)x_{t,i}$ for $i = 1, 2 \ldots k$, are the independent variables, but the βs are the same as in the original model.

To model with ARMA errors, all variables must be stationary. Thus, any regression with an ARIMA error can be rewritten as a regression with an ARMA error by differencing all variables with the same differencing operator as in the ARIMA model. Considering that the original data presents an error term following an ARMA(p, q) process, the regression model can be written as in (3.1), where $\phi(B)(1 - B)^d \eta_t = \theta(B)\epsilon_t$. After differencing all variables $y_t' = \beta_0 + \beta_1 x_{1,t}' + ... + \beta_k x_{k,t}' + \eta_t'$, where $\phi(B)\eta_t = \theta(B)\epsilon_t$ and $y_t' = (1 - B)^d y_t$. In particular, considering that η_t follows an ARIMA(1,1,1) model, it is possible to write the regression model as

$$y_t = \beta_0 + \beta_1 x_{1,t} + ... + \beta_k x_{k,t} + \eta_t \Rightarrow (1 - \phi_1 B)(1 - B)\eta_t = (1 + \theta_1 B)\epsilon_t \tag{3.4}$$

and the parameters of the model are estimated by minimizing the sum of the squared ϵ_t errors.

Sometimes a change in x_t does not effect y_t instantaneously. For instance, when x_t=advertising and y_t=sales or x_t=rainfall and y_t=stream flow. These are dynamic systems with input (x_t) and output (y_t), and x_t is often a leading indicator. The model include present and past values of the explanatory variables $x_t, x_{t-1}, x_{t-1}, \ldots$ and can be written as

$$y_t = c + \vartheta_0 x_t + \vartheta_1 x_{t-1} + \ldots + \vartheta_k x_{t-k} + \eta_t \tag{3.5}$$

where η_t is an ARIMA process. It is possible to rewrite the model as $y_t = c + (\vartheta_0 + \vartheta_1 B + \ldots + \vartheta_k B^k)x_t + \eta_t$, so $y_t = c + \vartheta(B)x_t + \eta_t$. The function $\vartheta(B)$ is called the transfer function since it describes how change in x_t is transferred to y_t. The value of k can be selected using the AICc.

The *auto.arima* function provided in the package "Forecast" (R. J. Hyndman & Khandakar, 2020) and implemented in the Software R (R Core Team, 2013), will fit a regression model ARIMA with errors if the argument **xreg** is used. It is necessary to specify the predictor variables to include, then the function will select the best ARIMA model for those errors (to see more details about how this function works, see section 2.2). To consider the lag of one predictor, such information have to be included as a input variable. As with standard regression models, to obtain forecasts, it is necessary first to forecast the predictors. When the predictors are

known in the future (e.g., calendar-associated variables such as holidays, day-of-week, etc.), this is straightforward. When the future value of the predictors is unknown, it must either model them separately or use assumed future values for each one of them (R. Hyndman & Athanasopoulos, 2018).

Regarding the existing applications, Bossche and Brijs (2004) discusses the impact of various explanatory variables on traffic safety, applying a regression model with ARIMA errors (also called dynamic regression models). The impact of variables on traffic safety is measured, and at the same time, the influence of additional unknown factors is captured by the error term. The results show an important effect of weather conditions, laws and regulations on traffic safety.

3.2 Supervised Machine Learning Models

This section describes the Machine Learning models used in this work: Support Vector Machine (SVM), Random Forest (RF) and Extreme Gradient Boosting (XGBoost). In the following, a brief description of such models will be made, and some applications of these models for time series forecasting will be presented.

Support Vector Machine

Support Vector Machine (SVM) is a well-known and very important algorithm in ML field. SVM is a supervised type of ML algorithm in which, given a set of training examples, each observation as belonging to one of the many classes, an SVM training algorithm builds a model that predicts the class of the new observation. SVM has the good ability to generalize the problem, which is the goal in statistical learning (Pradhan, 2012). SVMs were developed for classification problems and later adapted for regression problems. Forecasting time series falls under SVM regression tasks. First the main concepts about this algorithm are introduced and then details for SVM regression are explained based on Torgo (2017).

The main objective of SVM in a binary classification task is to find the optimal hyperplane which linearly separates the data points in two components by maximizing the margin, that is, the distance between the vectors and the hyperplane. In many real cases, where the data points are not linearly separable, SVM uses a nonlinear mapping of the original data into a very high dimensional space where the classes can be separated linearly by a hyperplane, Figure 3.1 illustrates this process. A relevant question concerning this algorithm, is how to choose the optimal hyperplane since there is a potentially infinite number of hyperplanes that can separate the cases. The goal is to ensure that the selected hyperplane leads to higher classification accuracy. Figure 3.2 illustrates an example of the SVM structure for two separating classes.

The scheme that is presented in Figure 3.2 assumes the black solid line as the optimal hyperplane, and the two dotted lines represent other hyperplanes passing through the nearest data points to the optimal hyperplane. The distance between the vectors and the hyperplane is called margin. The goal of SVM is to maximize this margin. Thus, the hyperplane with maximum margin is called the optimal hyperplane and the closest data points are defined as support vectors. In the case of two classes, the best hyperplane is the one that maximizes the margin between the

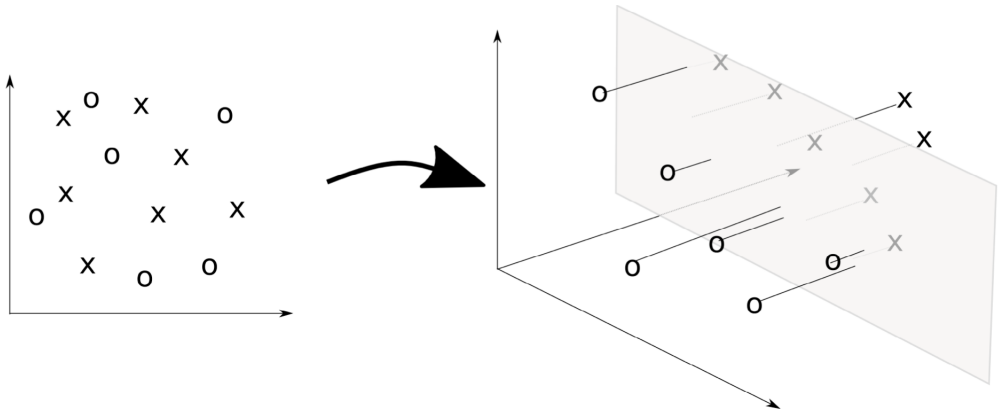

Figure 3.1: Mapping of the original data into a very high dimension space from Torgo (2017)

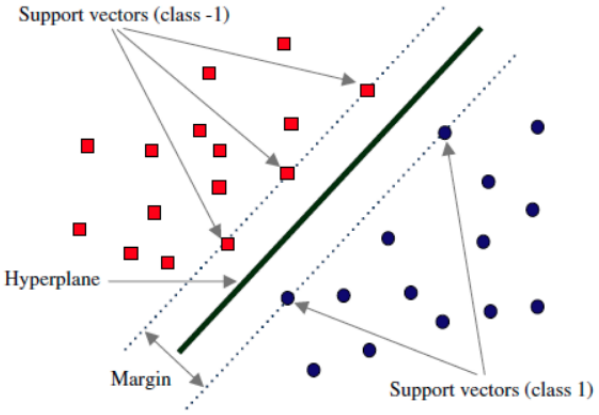

Figure 3.2: Optimal hyperplane for two classes adapted from Medjahed (2015)

points of these two classes, since it decreases the probability of confusion between the classes and allows the classifier to be able to generalize well for new observations. To determine the maximum margin hyperplane, SVMs use a quadratic optimization process.

It is possible to express the equation of the separating hyperplane as

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{3.6}$$

where $\mathbf{w}$ is a vector of coefficients, $\mathbf{x}$ is the input vector and b is the distance to the origin. Assuming that the points of one of the classes have $Y = +1$ while the others have $Y = -1$, then the separating hyperplane is described as,

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_i + b \geq 0, \forall i : y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq 0, \forall i : y_i = -1 \end{aligned} \tag{3.7}$$

Then, assuming that the separating hyperplane is in the centre of the two maximum margin hyperplanes denoted by H_1 and H_2 and that m is the distance from the maximum margin hyperplane to both them, the hyperplanes H_1 and H_2 can be represented by the following equations

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_i + b &= +m \\ \mathbf{w} \cdot \mathbf{x}_i + b &= -m \end{aligned} \tag{3.8}$$

The points in these two hyperplanes are the support vectors. It is possible to scale the variables so that $m = 1$. By definition, it is known that between H_1 and H_2 there are no instances, then

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_i + b &\geq +1, \forall i : y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_i + b &\leq -1, \forall i : y_i = -1 \end{aligned} \tag{3.9}$$

which can be re-written as,

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq +1, \forall i \tag{3.10}$$

The goal of SVMs it to maximize the margin between the hyperplanes H_1 and H_2. The problem could be resolved by minimising $\frac{1}{2}\ ||\mathbf{w}||^2$, where $||\mathbf{w}||$ is the normalized difference between their constant terms. SVMs calculate the maximum margin hyperplane by solving a dual optimization problem, that consists of maximizing the following expression,

$$\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j), \tag{3.11}$$

subject to,

$$\sum_{i=1}^{N} \lambda_i y_i = 0, \lambda_i \geq 0, 1 \geq i \geq N$$

where λ_i are the Lagrangian multipliers. Further details to support the understanding behind the SVM applied to cases of linearly separable classes are given following Burges (1998) and Torgo (2017).

Considering the problem when cases are not linearly separable, SVMs move the data into a higher dimension where the solution is feasible, i.e. where they can apply the same optimization procedure. When moving the data into a higher dimension, the goal of solving equation (3.11) in this new space will involve dot products of vectors of a much larger size than the initial data. This issue increases significantly the computational complexity of the problem. To deal with this drawback, there is the so-called kernel trick. A kernel function, $K(\cdot)$, takes as its inputs vectors in the original space and returns the dot product of the vectors in the new extended feature space. Hence, considering data $\mathbf{x}, \mathbf{z} \in X$ and a mapping from the original space into a new higher dimension space $\phi : X \rightarrow R_N$ then $K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z})$. The "trick" in the so-called

kernel trick is to replace the dot products in this high dimension on equation (3.11), by cheaper kernel calculations in the original low-dimensional space. Some usual kernel functions in SVM are Gaussian, Polynomial and Radial.

In real-world applications, it is hard to get a perfect linear separation between the classes. Thus, SVM allows misclassifications by introducing the concept of the *soft margin optimization problem*. Basically, it allows a few cases to be on the "wrong" side of the separating hyperplane, and for each of these misclassifications, a penalty is established. Therefore, the optimization problem can be re-defined considering the same as before, equation (3.11), with the addition of an extra constraint stating that $\lambda_i \leq C$, where C is the penalty given for each case misclassified for the model.

SVMs were adapted to regression problems by Vapnik (1995). In this article, the author has proposed ϵ-SV regression as a method that searches for an hyperplane whose distance to all training cases is at most ϵ. Figure 3.3 illustrates this procedure.

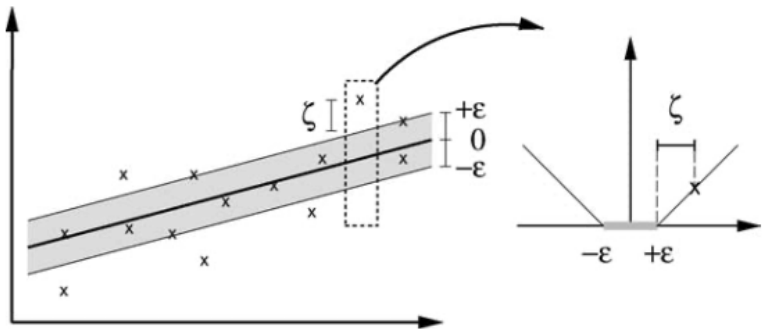

Figure 3.3: SVMs for regression from Smola and Schölkopf (2004)

Following Smola and Schölkopf (2004) and Awad and Khanna (2015), it is possible to provide a brief description of how Support Vector Regression (SVR) works. SVR uses the following error metric

$$|\xi|_\varepsilon = \begin{cases} 0 & \text{if } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases} \tag{3.12}$$

The optimization problem for SVR is similar to classification. Therefore, for the more general case where the model accepts misclassifications, SVR presents the following minimization problem,

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l}(\xi + \xi_i^*) \tag{3.13}$$

subject to,

$$\begin{cases} y_i - \mathbf{w} \cdot \mathbf{x} - b \leq \varepsilon + \xi_i \\ \mathbf{w} \cdot \mathbf{x} + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

As in SVM for classification, it is possible to use Lagrangian multipliers to obtain the function that leads to the optimization problem of maximizing

$$\begin{cases} -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i - \lambda_i^*)(\lambda_j - \lambda_j^*)(\mathbf{x}_i \cdot \mathbf{x}_j) \\ -\epsilon \sum_{i=1}^{N} (\lambda_i - \lambda_i^*) + \sum_{j=1}^{N} y_i (\lambda_i - \lambda_i^*) \end{cases} \tag{3.14}$$

Subject to,

$$\begin{cases} \sum_{i=1}^{N} (\lambda_i - \lambda_i^*) = 0 \\ \lambda_i, \lambda_i^* \in [0, C] \end{cases}$$

Several parameters can be considered when using SVM. In this work, two common and useful parameters were considered. Table 3.1 summarizes these parameters and presents a brief description of each one.

Parameter	Description
Cost	Cost of constraints violation. It represents the penalty associated with errors larger than epsilon.
Gamma	Parameter that controls the size of the acceptable boundary around the hyperplane. Increasing gamma usually increases the number of support vectors.

Table 3.1: SVM Parameters

The ability of SVM to solve nonlinear regression estimation problems while not requiring prior knowledge of the structure of the data, makes SVM successful in time series forecasting. Sapankevych and Sankar (2009) presents a survey of time series prediction using this method. Müller et al. (2006) also presents a case study applying SVM to forecast time series and finds that SVM methods work particularly well if the data are sparse, i.e. few data available in a high dimensional space. This is due to their good inherent regularization properties.

Random Forest

Tree-based methods for classification and regression tasks are presented by Breiman, Friedman, Stone, and Olshen (1984), and are widely applied in supervised learning problems. These are intuitive methods, easy to read and to interpret. Classification trees are used when the target variable is categorical, and regression trees are applied when the target variable continuous.

A tree is grown from a "root" node that contains all the observations from the initial dataset. The instances in this node are sent to one of two descendant nodes, using a split on a single

predictor variable. Figure 3.4 illustrates the general scheme of a decision tree. The split used by a tree to partition a node into its two descendants is given by considering every possible split on every predictor variable. The value given by the combination between the predictor and the split is used to partition the node. One example of splitting criteria is the sum of squared residuals for regression trees and entropy for classification trees, (Cutler, Cutler, & Stevens, 2009).

In general, the trees are grown until a stopping criterion is met. After this criterion is reached, pruning back the tree is necessary to prevent over-fitting. Once a tree has been grown and possibly pruned, the non-partitioned nodes are known as leaf nodes. For regression problems, the predicted values are obtained from the observations in a leaf node by averaging their responses. For classification problems, the predicted values are given by calculating either category proportions or the most frequent class.

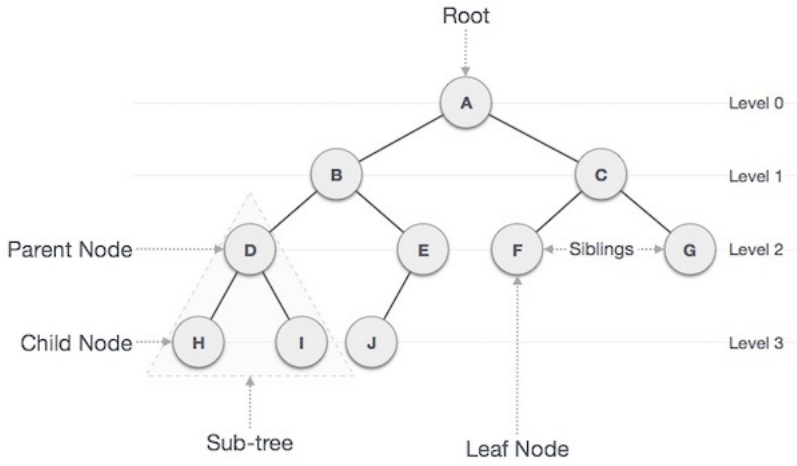

Figure 3.4: General structure of a decision tree

Tree-based ensembles combine the predictions of many different trees to give an aggregated prediction. Random Forests algorithms can be classified as one of these methods. It is a learning method that operates by constructing multiple decision trees. The final decision is made based on the majority of the trees and is chosen by the Random Forest. The main idea behind the ensemble model is that a group of weak learners come together to form a strong learner. There are two common techniques to perform ensemble decision trees: bagging and boosting. Random Forest uses the first. It creates several subsets of data from the training sample chosen randomly with replacement. Therefore, each subset of data is used to train a decision trees. As a result, the algorithm ends up with an ensemble of different models, where averaging predictions from different trees is more robust than a single decision tree. Additionally, when deciding the best test for each node of the trees, Random Forests choose this test by only considering a random subset of the predictors. This is another way of increasing the diversity among trees. The resulting trees are combined by unweighted voting if the response variable is categorical (classification) or unweighted averaging if the response variable is continuous (regression) (Cutler et al., 2012). Figure 3.5 illustrates how RF works.

A formal definition of the algorithm is given by Breiman (2001) as the following. A Random Forest is a classifier that consists of a collection of tree-structured classifiers $\{h(\mathbf{x}, \Theta_k), k =$

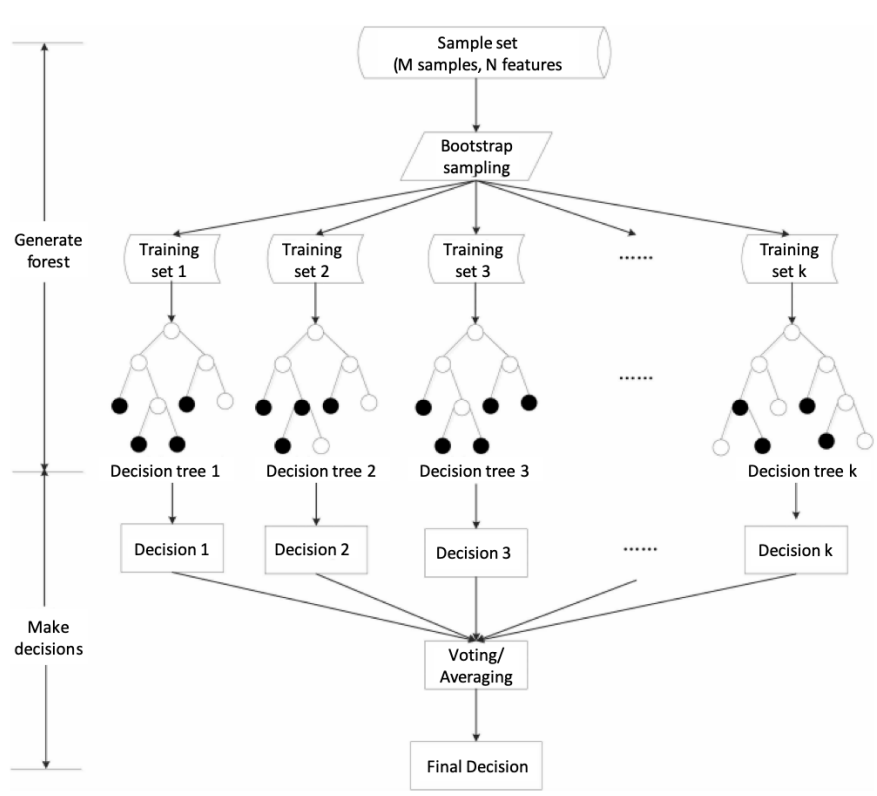

Figure 3.5: The flow chart of Random Forest adapted from Feng et al. (2018)

$1, ...\}$ where $\{\Theta_k\}$ are independent identically distributed random vectors and each tree lists a single vote for the most popular class at input x. In his article, the author presents a strong theoretical background for Random Forests.

Cutler et al. (2012) define a tree as $\hat{h}(x, \theta_j, D)$, with $D = (x_1, y_1), ..., (x_N, y_N)$, where $x_i = (x_{i,1}, ..., x_{i,p})^T$ denotes the p predictors, y_i denotes the response and θ_j denotes a particular realization of Θ_j. Each tree is fit to an independent bootstrap sample from the original data. The randomization involved in bootstrap sampling gives one part of Θ_j. Then, when splitting a node, the best split is found over a randomly selected subset of m predictor variables instead of all p predictors, independently at each node. The randomization used to sample the predictors gives the remaining part of Θ_j. Figure 3.6 shows the pseudo-code of Random Forest algorithm.

The number of the attributes used in each node is limited to some percentage of the total (known as the hyperparameter). As a result, the ensemble model does not rely too heavily on any individual feature and makes fair use of all potentially predictive features. Furthermore, Random Forest has the ability to prevent overfitting, once it adds a further element of randomness when each tree draws a random sample from the original data set.

Based on the Strong Law of Large Numbers, the generalization error for Random Forests converges to a limit as the number of trees in the forest becomes large. On average, the generalization accuracy of Random Forest does not deteriorate when more trees are included in the ensemble. The largest the number of trees considered, the most probable the ensemble has con-

Algorithm 2 Random Forests

Let $\mathscr{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ denote the training data, with $x_i = (x_{i,1}, \ldots, x_{i,p})^T$. For $j = 1$ to J:

1. Take a bootstrap sample $\mathscr{D}_j$ of size N from $\mathscr{D}$.
2. Using the bootstrap sample $\mathscr{D}_j$ as the training data, fit a tree using binary recursive partitioning:
   a. Start with all observations in a single node.
   b. Repeat the following steps recursively for each unsplit node until the stopping criterion is met:
      i. Select m predictors at random from the p available predictors.
      ii. Find the best binary split among all binary splits on the m predictors from step i.
      iii. Split the node into two descendant nodes using the split from step ii.

To make a prediction at a new point x,

- $\hat{f}(x) = \frac{1}{J}\sum_{j=1}^{J} \hat{h}_j(x)$ for regression
- $\hat{f}(x) = arg\max_y \sum_{j=1}^{J} I(\hat{h}_j(x) = y)$ for classification

where $\hat{h}_j(x)$ is the prediction of the response variable at x using the jth tree

Figure 3.6: Random Forest algorithm from Cutler et al. (2012)

verged to its asymptotic generalization error (Breiman, 2001). Despite that, in this dissertation, a different number of trees were tested as a parameter of this algorithm to analyze the performance of each ensemble model generated.

Further, it is possible to enumerate some advantages of using Random Forest to solve supervised learning problems:

- Can handle both regression and classification;
- Can be used directly for high-dimensional problems;
- Do not require tuning of many parameters;
- Have an inbuilt method of assessing generalization error;
- Provide measures of variable importance.

Regarding the application of Random Forests for time series forecasting, Kane, Price, Scotch, and Rabinowitz (2014) present an application of Random Forest for forecasting of avian influenza H5N1 outbreaks and a comparison to ARIMA. The authors founded that Random Forest provides enhanced predictive ability over the ARIMA model for the prediction of infectious disease outbreaks.

XGBoost

XGBoost stands for eXtreme Gradient Boosting. It is an algorithm that has recently been dominating applied Machine Learning and recognized competitions. It is an implementation of gradient boosted decision trees designed for speed and performance. It is a decision-tree-based ensemble ML algorithm that uses a gradient boosting framework. Different from bagging methods, boosting learners are learned sequentially with early learners fitting simple models to the observations and then analyzing the errors. Consecutive trees (random sample) are fit, and at every step, the goal is to improve the accuracy from the prior tree.

Proposed by Chen and Guestrin (2016), XGBoost is a system optimized for fast parallel tree construction and designed to be fault-tolerant under the distributed setting. The algorithm can handle tens of millions of samples on a single node and scales beyond billions of samples with distributed computing. To describe the main idea behind XGBoost, some essential concepts of gradient boosting machines will be first introduced.

In Gradient Boosting Machines (or GBMs), the learning procedure consecutively fits new models to provide a better estimate of the target variable. The main idea behind GBMs is to build the new base-learners to be maximally correlated with the negative gradient of the loss function, associated with the entire ensemble. In general, the user can decide what loss function to use. This flexibility makes the GBMs highly customizable to any particular data mining task. Moreover, the gradient boosting machines algorithms were successfully applied in several machine learning challenges, such as Johnson and Zhang (2014),Hutchinson, Liu, and Dietterich (2011), and Bissacco, Yang, and Soatto (2007). In the following, a brief introduction to the basic methodology of the algorithm is described based on Bentéjac, Csörgő, and Martínez-Muñoz (2020). Further details can be seen in Natekin and Knoll (2013).

Given a dataset $(\mathbf{x}_i, y_i)_{i=1}^N$, where $\mathbf{x}$=$(x_1, ..., x_d)$ represents the explanatory variables and y refers to the labels of the target variable, the goal of the gradient boosting is to find an approximation, $\hat{F}(\mathbf{x})$, of the function $F^*(\mathbf{x})$ which maps instances $\mathbf{x}$ to their outputs values y, by minimizing the expected value of a given loss function, $L(y, F(\mathbf{x}))$. The algorithm builds an additive approximation of $F^*(\mathbf{x})$ as a weighted sum of functions

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h_m(\mathbf{x}) \tag{3.15}$$

where ρ_m is the weight of m^{th} function, $h_m(\mathbf{x})$. These functions represent the models of the ensemble, i.e., the decision trees. The approximation is built in an iterative form. First, a constant approximation of F^* is obtained as

$$F_0(\mathbf{x}) = \arg\min_{\alpha} \sum_{i=1}^{N} L(y_i, \alpha) \tag{3.16}$$

Therefore, subsequent models are expected to minimize

$$(\rho_m, h_m(\mathbf{x})) = \arg\min_{\rho,h} \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i)) \tag{3.17}$$

Finding the solution for this optimization problem can be a hard task. Instead, each h_m can be considered as a greedy step in a gradient descent optimization for F^*. With this purpose, each model, h_m, is trained on a new dataset $D = (x_i, r_{m_i})_{i=1}^N$, where the pseudo-residuals, r_{m_i}, are calculated by

$$r_{m_i} = \left[\frac{\delta L(y_i, F(\mathbf{x}))}{\delta F(\mathbf{x})} \right]_{F(\mathbf{x})=F_{m-1}(x)} \tag{3.18}$$

The value of ρ_m is then computed by solving a linear optimization problem. Friedman (2000) states that gradient boosting algorithm can suffer from over-fitting if the iterative process is not properly regularized. Several regularization parameters are considered to control the additive process of gradient boosting. One natural way to regularize gradient boosting is to apply shrinkage to reduce each gradient descent step $F_m(\mathbf{x})=F_{m-1}(\mathbf{x}) + \upsilon\rho_m h_m(\mathbf{x})$, with $\upsilon = (0, 1]$. The value of υ is usually set to 0.1 (Bentéjac et al., 2020). Further regularization can also be achieved by limiting the complexity of the trained models. Figure 3.7 presents the general idea behind the GBM algorithm.

Algorithm 1 Friedman's Gradient Boost algorithm

Inputs:

- input data $(x, y)_{i=1}^N$
- number of iterations M
- choice of the loss-function $\Psi(y, f)$
- choice of the base-learner model $h(x, \theta)$

Algorithm:

1: initialize $\widehat{f_0}$ with a constant
2: **for** $t = 1$ to M **do**
3: compute the negative gradient $g_t(x)$
4: fit a new base-learner function $h(x, \theta_t)$
5: find the best gradient descent step-size ρ_t:
$\rho_t = \arg\min_\rho \sum_{i=1}^{N} \Psi\left[y_i, \widehat{f}_{t-1}(x_i) + \rho h(x_i, \theta_t)\right]$
6: update the function estimate:
$\widehat{f_t} \leftarrow \widehat{f}_{t-1} + \rho_t h(x, \theta_t)$
7: **end for**

Figure 3.7: Gradient Boosting algorithm from Natekin and Knoll (2013)

Regarding XGBboost, the algorithm focuses only on decision trees as base classifiers. Therefore, the following variation of the loss function is used to control the complexity of the trees

$$L(\phi) = \sum_{i=1}^{N} L(y_i, F(x_i)) + \sum_{i=1}^{M} \Omega(h_m) \tag{3.19}$$

where,

$$\Omega(h) = \gamma T + \frac{1}{2}\lambda||w||^2$$

Hence, L is the loss function that measures the quality of prediction $\hat{y}_i$, T is the number of leaves of the tree, and w are the output scores of the leaves. Ω penalizes the complexity of the model. The additional regularization term, λ, helps to smooth the final learnt weights to avoid over-fitting. Finally, the value of γ controls the minimum loss reduction gain needed to split an internal node.

Furthermore, XGBoost implements randomization techniques to reduce over-fitting and to increase training speed. These techniques included in XGBoost involve random subsamples to train individual trees and column subsampling at tree and tree node levels. The key question and the most time-consuming part in tree learning algorithms is how to find the best split. These algorithms usually enumerate all possible candidate splits and select the one with the highest gain. Therefore, it requires applying a linear scan over each sorted attribute to find the best split for each node. XGBoost focuses on reducing the computational complexity of this part since it avoids sorting the data repeatedly in every node, using a specific compressed column-based structure in which the data is stored pre-sorted. As a result, each attribute needs to be sorted only once. This structure allows the algorithm to find the best split for each attribute in parallel. Besides, instead of scanning all possible candidate splits, XGBoost implements a method based on percentiles of the dataset where only a subset of candidate splits is tested, and their gain is computed using aggregated statistics.

Several parameters can be considered when using XGBoost. Such parameters can be divided in (1) general parameters, (2) booster parameters and (3) learning task parameters. For this work, three common and useful parameters were considered. Table 3.2 summarizes these parameters and presents a brief description of each one.

Parameter	Description
Nrounds	The number of rounds for boosting.
Eta	Step size shrinkage used in update to prevents over-fitting. After each boosting step, it is possible to directly get the weights of new features, and this parameter shrinks the feature weights to make the boosting process more conservative
Gamma	Minimum loss reduction needed to make another partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be.
Max-depth	Maximum depth of a tree. Increasing this parameter will make the model more complex and more likely to overfit.

Table 3.2: XGBoost Parameters

In summary, the implementation of XGBoost offers several advanced features for model tuning, computing environments and algorithm enhancement. It is capable of performing the

main forms of gradient boosting, and it is robust enough to support fine tuning and addition of regularization parameters. Nielsen (2016) highlights the advantages of using XGBoost in Kaggle competitions.

Regarding the implementation in time series forecasting, Pan (2018) applies XGBoost algorithm to predict hourly PM2.5 concentration in China, that is one of the six pollutants analyzed to measure the Air Quality Index (AQI). The XGBoost method is compared with other ML models, such as Random Forest, multiple linear regression, decision tree and support vector machines. Although the author does not compare the results with statistical methods, they conclude that the XGBoost algorithm outperforms other data mining methods.

Chapter 4

Experimental study

This chapter presents the approach adopted in this work to address the issues raised in Chapter 1, namely: to analyze the performance of forecasting models when considering exogenous factors and to compare the forecasting performance of ML models and ARIMA time series models. Moreover, this section presents case studies and discusses the main results.

4.1 Methodology and implementation

The overall procedure developed during this work has four main steps as represented in Figure 4.1: Preliminary analysis; Data Set creation; Model fitting; Performance analysis.

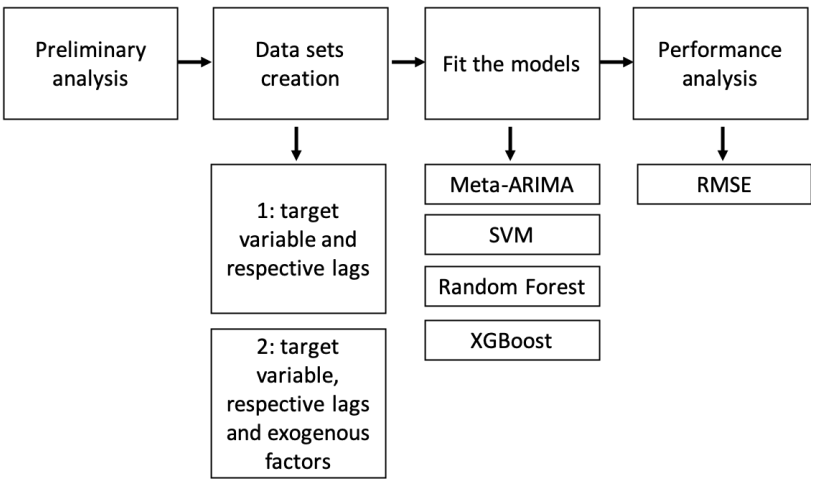

Figure 4.1: Description of the study

A short description of these steps follows.

Preliminary analysis In this step, the goal is to analyze the data and identify possible seasonalities, outliers and missing values. Furthermore, if the series present non-stationarity in variance

hindering the use of ARIMA models, then a log transformation must be considered. Such preprocessing is not applied to the data for ML models since the purpose is to uncover the potential of these models for forecasting time series with complex patterns, in particular non-linear relationships between the components trend and seasonality. In addition, the autocorrelation and partial autocorrelation functions are analyzed to determine the number of lags of the target variable should be considered as inputs in the Machine Learning models.

Data sets creation In this step, two datasets are created to work with: one that contains just the target variable and the appropriate lags and another with the additional exogenous factors selected. The purpose is to assess the gain of considering external information in time series forecasting problems as well analyze the performance of the ML models over Meta-ARIMA.

Fitting the models In this step, the models, Meta-ARIMA, Support Vector Regression, Random Forest and XGBoost are fitted to the data sets. Recall that ARIMA modelling is implemented via the *auto.arima* function minimizing AICc criterion, procedure here designated as Meta-ARIMA model. Regarding ML models, for ensuring a fair comparison with the Meta-ARIMA model, a range of values, usual in the literature, was considered for each method and are presented in Table 4.1. A brief explanation of such parameters can be found in section 3.2. The parameter combination led to twenty SVM models, three RF models and eighteen XGBoost models.The choice of the best model for each method was carried out based on RMSE.

Model	Parameter	Range
SVM	Cost	{1:10}
SVM	Gamma	{0.01,0.05}
RF	# of Trees	{200,700,1500}
XGBoost	Eta	{0.1,0.3,0.5}
XGBoost	Gamma	{0,5,10}
XGBoost	Max Tree depth	{5,10}

Table 4.1: Set of parameters considered for the ML models

Performance analysis Forecasts are evaluated with RMSE since the comparison is between approaches for the same data set. To obtain reliable estimates, a Monte Carlo experimental procedure, described in detail in section 4.2 is adopted.

Implementation All the computations carried out during this work were performed with R (R Core Team, 2013) and relevant packages namely: "performanceEstimation" (Torgo, 2014), "xgboost" (Chen et al., 2019), "e1071", (Meyer, Dimitriadou, Hornik, Weingessel, & Leisch, 2019) and "randomForest" (Liaw & Wiener, 2018). One of the essential packages is "performanceEstimation" (Torgo, 2014). This package allows for estimating the predictive performance of different approaches in predictive tasks. In the context of this package, these approaches are called

workflows. The package is generic and dynamic since it can estimate the values of any performance metric, for any workflow on different predictive tasks, such as classification, regression and time series tasks. There are several standard workflows already implemented in the package, making easier the comparison between different well-known Machine Learning algorithms and a bunch of predictive performance metrics for different tasks. According to Torgo (2014), the main goal of the standard workflows implemented in the package is to receive as input a predictive task for which training and test samples are given and to produce predictions for the given test set. These predictions will then be used to obtain the predictive metrics chosen by the user to obtain reliable estimates.

The "performanceEstimation" package allows considering two types of workflows: standard and user-defined. In this work, the standard workflow was used for Random Forest and SVM models, since these methods are configured in the package and ready for use. For XGBoost and Meta-ARIMA models, it was necessary to develop specific workflows to guarantee that all models are considering the same train and test in all samples. It was considered a Monte-Carlo experiment to obtain reliable estimates of the evaluation metric. This method was implemented by using the function *MonteCarlo* on the same package. More details about this procedure will be described in the next section.

4.2 Methods of performance estimation

The usual procedure to assess the performance of a predictive model is to compute a loss function, e.g. RMSE, for new data set. According to Cerqueira, Torgo, and Mozetic (2019), the most common approach for independent and identically distributed data is cross-validation. It is a re-sampling method used for performance evaluation and to tune model parameters commonly applied to predictive models to prevent over-fitting. In time series data, due to the dependency among observations, it is essential to preserve the temporal order and to guarantee that a model will never be tested on past data, with the risk of obtaining estimates that are not reliable. As a result, standard cross-validation procedures do not apply to time series.

Berrar (2018) provides an introduction to the most common types of cross-validation and related data re-sampling methods. Cross-validation methodology has a unique parameter k that represents the number of equal size groups that a sample will be split into. Hence, by definition, the procedure is often called k-fold cross-validation. As stated by the author, the data splitting is performed by randomly sampling cases from the learning set without replacement. Moreover, the basic idea behind this procedure is to train the model using $k-1$ subsets and then apply it to the remaining subset, called the validation set so that the performance can be measured in each iteration. This procedure is repeated until each of the k subsets has served as the validation set. The average of the k performance measurements on the k validation sets is considered as the performance of the model.

Although cross-validation is a widely used standard procedure in many evaluation processes, when it comes to time series forecasting its application is not straightforward, as a consequence of the implicit serial correlation, potential non-stationarity of the data and intrinsic order of the observations. Bergmeir, Hyndman, and Koo (2017) support that in the case of a purely autoregres-

sive model, the use of standard k-fold CV is possible as long as the models present uncorrelated errors. Cerqueira, Torgo, and Mozetic (2019) compare different variants of cross-validation and out-of-sample approaches for evaluating time series forecasting models, concluding that there are notable differences in the performance estimation under these two scenarios. The authors argue that despite empirical experiments suggest that cross-validation approaches can be applied to stationary time series, in real-world applications, where different sources of non-stationarity variation are present, the most accurate estimates are produced by out-of-sample (OOS) methods that preserve the temporal order of observations. Thus, no random re-sampling of the observations or any other procedure that may affect the time ordering of the data can be applied to time series. However, to ensure the statistical reliability of the performance estimates, some randomness should be included in the estimation of the performance measure.

With this objective, Torgo (2017) proposes to use a Monte-Carlo experiment to obtain reliable estimates performance estimates from time series data. Monte-Carlo methods rely on random sampling to obtain their results. This sampling method will be used to generate a set of R points in the series. The procedure can be described as follows:

1. Considering a series of size n, choose the number of iterations, R and the set of R points in the series;

2. For each randomly selected time point r chose the number of observations, i, of the series that will be used to train the models and the subsequent t observations that will be used to test them;

3. At the end of the R iterations, it will be R estimates for each of the evaluation metrics considered. The final estimate is given by the average of the R estimates obtained.

Each of these estimates is obtained on a randomly selected window of $Q = i+t$ observations of data, being the first i observations used for training and the remaining t observations for testing. It will ensure that the experiments always respect the time ordering of the time series data. Thus, repeating the process R times will ensure sufficient variability on the train+test conditions, which increases the reliability of the estimates. Since it is necessary to ensure that for every random point r there are i observations of data before and t observations after, some of the data from the random selection of the R points are eliminated. Figure 4.2 shows the difference between both estimation processes, cross-validation and the Monte-Carlo procedure.

4.3 Case studies

The data sets considered in this work are time series for which explanatory variables are available. The three time series under analysis present a variety of characteristics such as trend, seasonality and different sample sizes.

4.3.1 Case 1: Bike Sharing Data

As urbanization reach remarkable levels, road traffic has become a modern day issue. It is possible to highlight some adverse effects of heavy traffic, such as air pollution, safety risks,

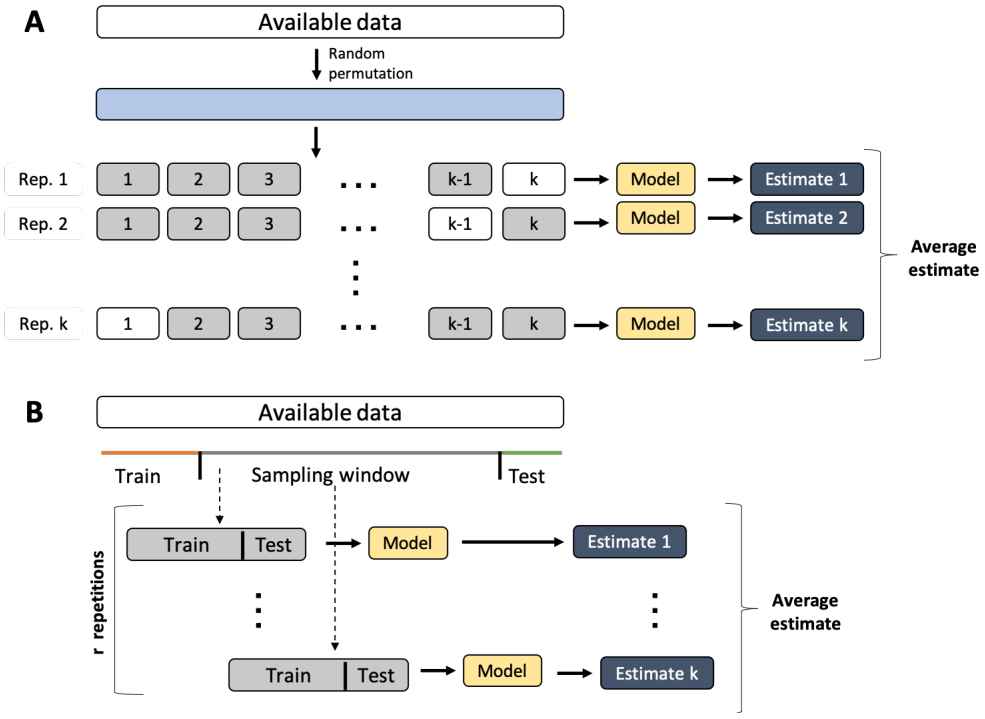

Figure 4.2: (A) k-fold cross-validation and (B) Monte-Carlo estimates adapted from from Torgo (2017)

reduction of accessibility and sustainability. Bike sharing is one alternative to this problem that has been gaining popularity in recent years. It refers to rental schemes, by which people can pick up, ride and drop off bicycles at diverse points across the city. The benefits of such systems include transport flexibility, reductions to vehicle emissions, health benefits, etc. The data generated by these systems make them attractive for general modelling analysis and research. The data characteristics turn bike sharing schemes into a virtual sensor network that can be applied for sensing mobility in the city. Therefore, it is expected that most of the significant events in the city could be detected via monitoring data from these systems.

The bike sharing dataset used in this work is from UCI Machine Learning repository, and it is public. It provides daily rental data from January/2011 to December/2012. In total, there are 731 days observed. The target variable is the daily number of rented bikes. For this work, 30 days will be forecasted, considering the historical observations of the rental bikes and exogenous drivers. Figure 4.3 represents the time series of the daily number of rented bikes from January/2011 to December/2012.

The time series presents an overall increasing trend superimposed on an annual cycle. Short-term cycles, possibly weekly cycles, may be present. The exogenous factors are constituted by continuous and categorical variables, namely:

- Season: 1-winter, 2-spring, 3-summer or 4-fall;
- Holiday: indicates if the day is a holiday;

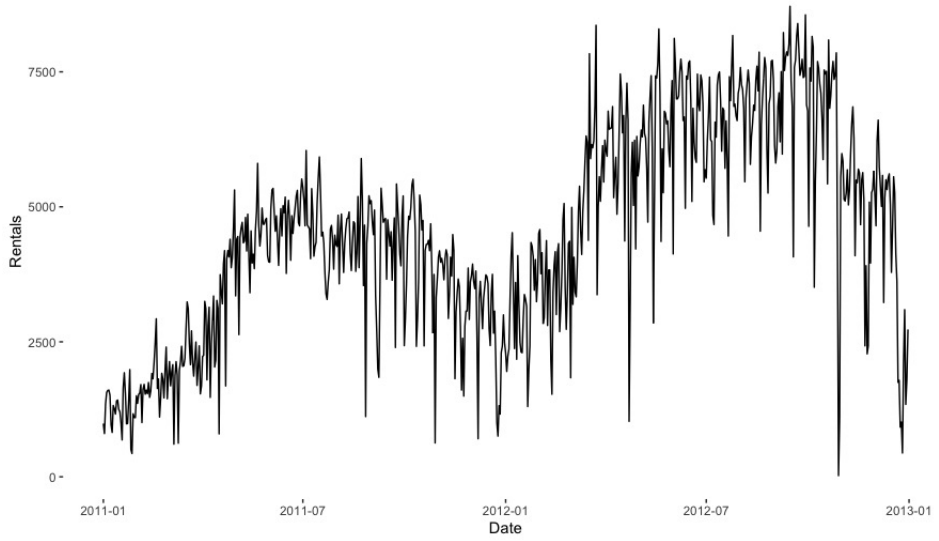

Figure 4.3: Bike Rentals series

- Weather: 1-Clear/Sunny, 2-Cloudy/Mist and 3-Rain/Snow/Fog;
- Weekend: indicates if the day is weekday or weekend, where 0=weekday, 1=Saturday and 2=Sunday;
- Temperature - Normalized in Celsius;
- Feeling temperature - Normalized in Celsius;
- Normalized humidity;
- Normalized wind speed

The data do not present missing values for any variable. When analyzing the continuous features, Figure 4.4 shows that there is a strong contemporaneous correlation between the bike rentals and the Temperature. Humidity and wind speed also present some impact on the number of rented bikes, although they are less correlated. Figure 4.5 shows the bike rentals per day of the week in 2011, and Figure 4.6 shows that bike rentals per day in 2012, where 1 represents Monday. For the both periods, all weekdays seems to present a similar range of bike rentals, although in 2012 there is an increase in the level of the rentals.

Regarding the variable Season, Figure 4.7 shows that there is more demand for bike rentals in drier seasons, such as Summer and Fall. It is possible to see that in Winter of 2012 there is a day with almost none bike rentals. Furthermore, to understand the distribution of the data regarding the variable Weather, Figure 4.8 presents an intuitive behaviour, showing that there are more bike rentals when the weather is clear or sunny.

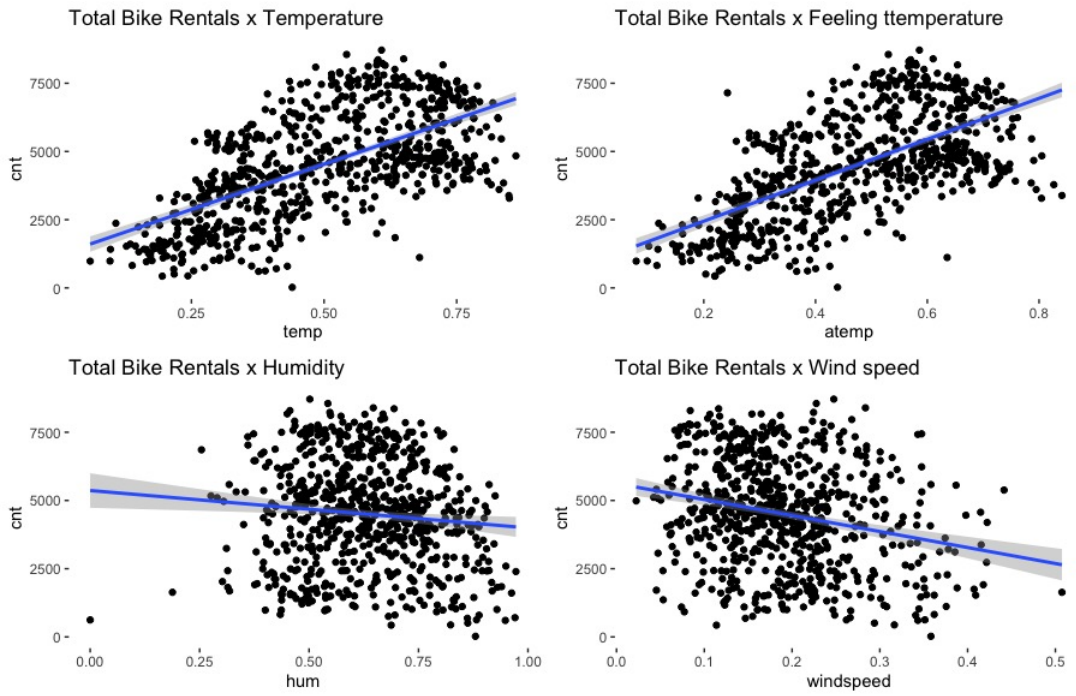

Figure 4.4: Continuous exogenous factors for Bike Sharing data

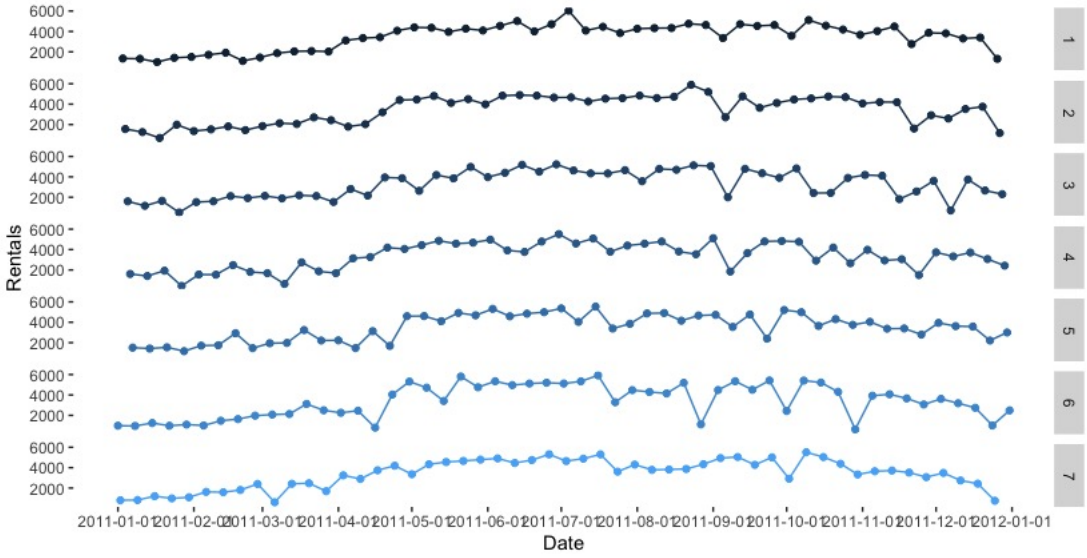

Figure 4.5: Bike Rentals per weekday in 2011, with 1 representing Monday

Regarding the characteristics of the series, Figure 4.9 shows the autocorrelation and partial autocorrelation functions of the bike rentals. The ACF is decreasing very slowly, indicating that the series is non-stationary. In the PACF plot, there is a significant correlation at the first six lags followed by correlations that are not significant. Then, the lags of the target variable up to six

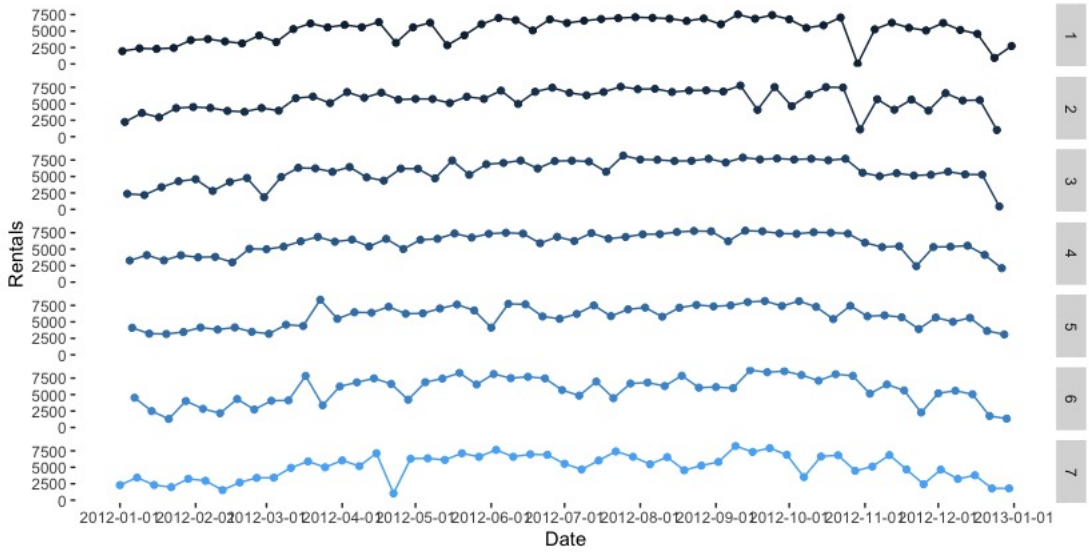

Figure 4.6: Bike Rentals per weekday in 2012, with 1 representing Monday

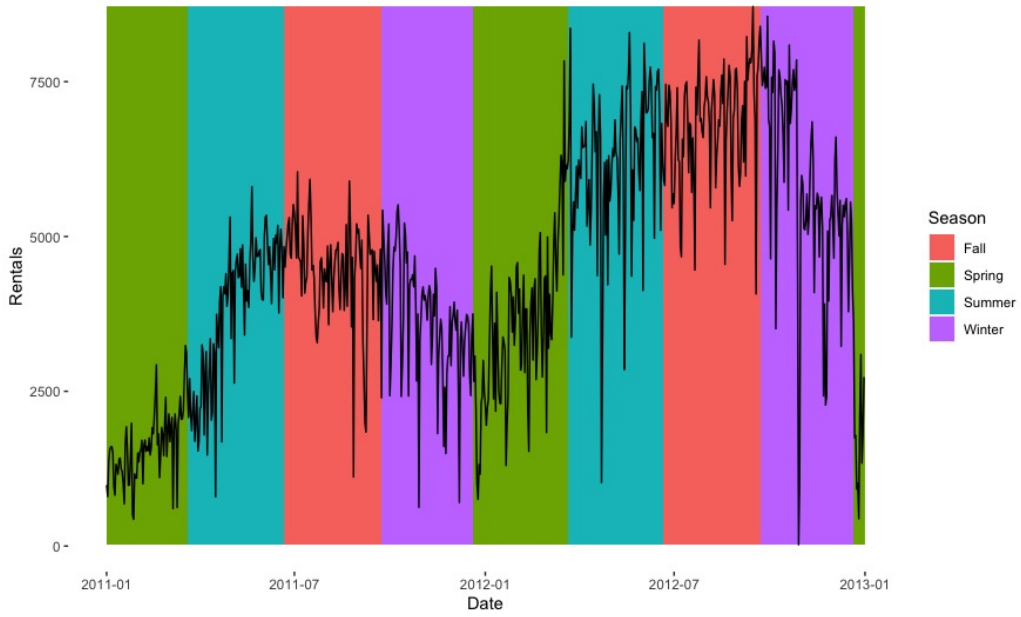

Figure 4.7: Bike Rentals per season

will be considered as inputs for the ML models.

Considering the size of the available data, $n = 731$ days, and assuming that in the businesses field a company would use the maximum information as possible to predict the future, a training sample of size $i = 500$ is considered in this work. Since the goal is to predict the next 30 days, the test sample is fixed equals $t = 30$. The number of iteration in the Monte-Carlo experiment were set to $R = 100$.

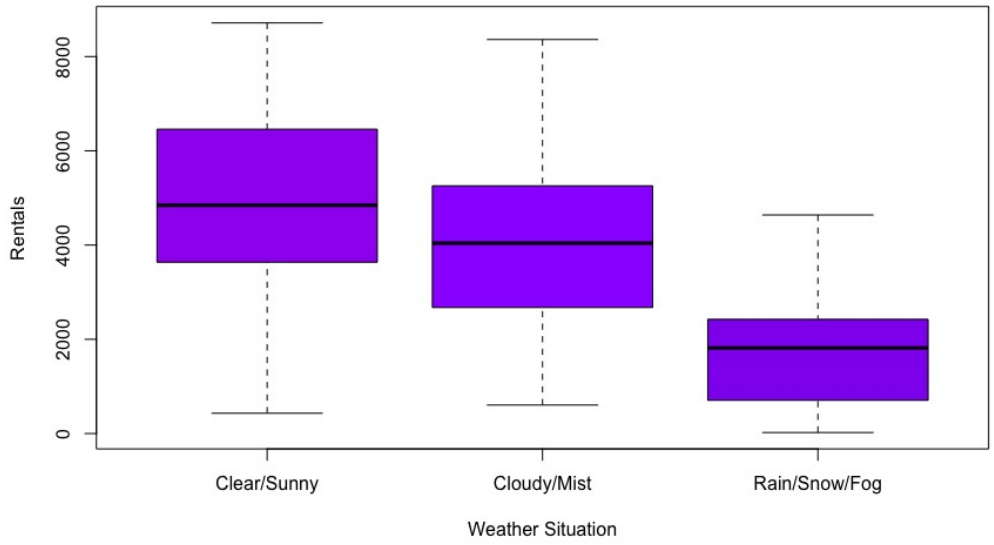

Figure 4.8: Bike Rentals per season

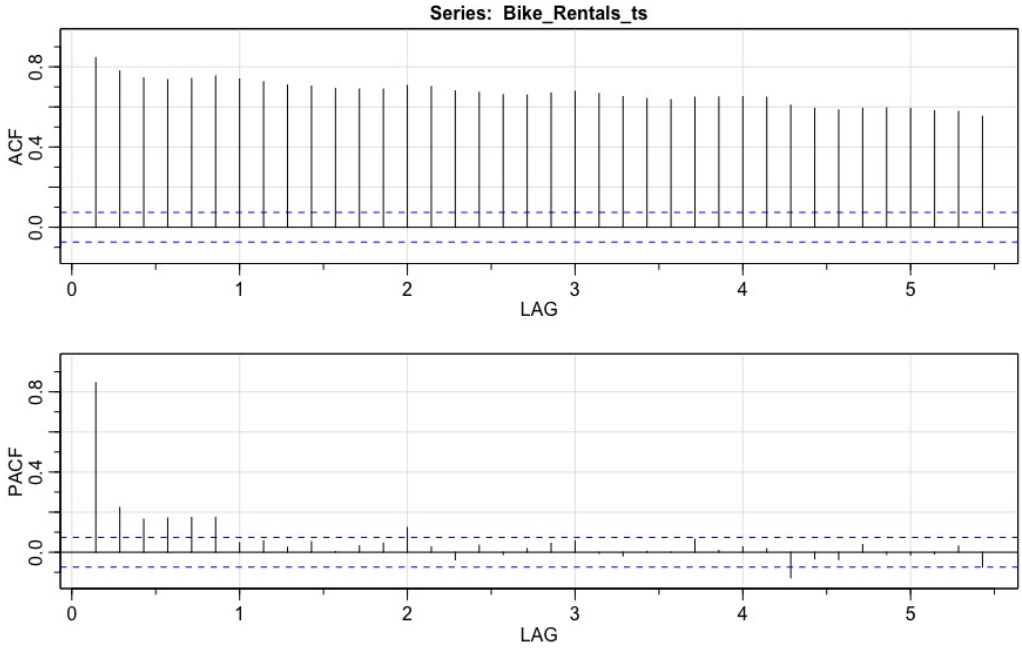

Figure 4.9: ACF and PACF of Bike Rentals

Table 4.2 summarizes the average error obtained by the best model within each class: ARIMA, SVM, RF and XGBoost.The lowest RMSEs are highlighted in bold.

	Meta-ARIMA	SVM	RF	XGboost
Without drivers	1201	1035	1154	1306
With drivers	**1129**	**854**	**938**	**940**

Table 4.2: Performance of the forecasting models data with and without exogenous information - Bike Sharing data

In general, considering additional information factors leads to forecasts with small RMSE for all the models fitted. The best accuracy is observed for the SVM model in both cases. The Meta-ARIMA model does not show a significant improvement with the inclusion of additional information. To illustrate the behaviour of the forecast obtained from each model, the data set corresponding to the first iteration of the Monte-Carlo procedure was chosen, and the results are presented in Figure 4.10. Overall, the results are in accordance with the RMSE in table 4.2 and clearly in this particular dataset the forecasts generated without the exogenous factors under estimate the number of rented bikes.

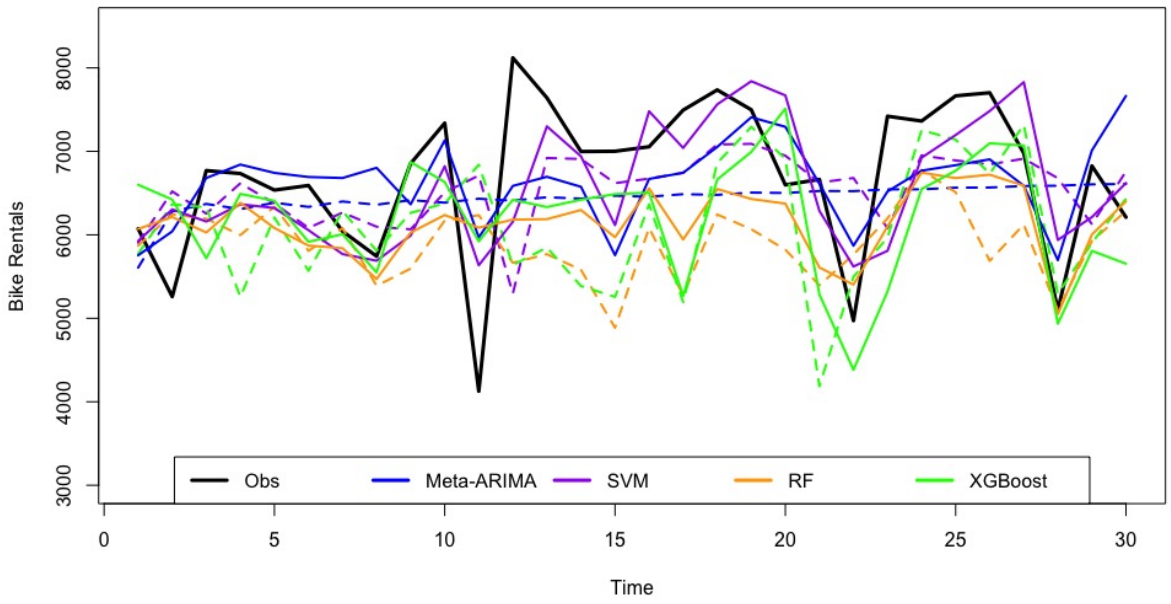

Figure 4.10: Forecasts produced by each model for the Bike Sharing dataset. The dotted lines represent the forecasts generated from the data without exogenous factors.

For the data without additional information, it is possible to see that the Meta-ARIMA model is not able to capture the long-term series pattern. After the first two steps-ahead, the forecasts are simply the mean. RF e XGboost models presented similar behaviours, and in general, they can capture the general trend of the series. Note that SVM is the model that generates the closest forecasts to the observed data. For the data considering exogenous factors, the ML models

generate better results, and the Meta-Arima model seems to capture much better the behaviour of the data. However, this model still generates the worst forecasts than others. Note that, the results can vary from iteration to iteration since the series that are being forecasted change from one iteration to the next.

The complete performance estimation results are discussed in Appendix A.1. The final ML models considered are the ones with the lowest RMSE for each method (SVM, RF and XGBoost).

4.3.2 Case 2: Sales Data

Sales forecasting is a big concern for modern business intelligence nowadays. It is an integral part of the decision-making process in any company to help in the adjustment of the strategy to improve sales or productivity in the coming future. Predicting sales can be considered as a time series problem since its behaviour strongly depends on time. It is crucial to take into account some key exogenous factors which have an impact on sales to adapt the forecast model and improve the predictions.

Thus, data from Germany's drug store called Rossmann was used to forecast sales. The data are available on the website Kaggle, and it is public. Store sales are influenced by many factors, including promotions, school and state holidays and weekends. The data do not present missing values for any variable. For this dissertation, the goal is to predict the sales turnover for the next 30 days for a selected store.

The observed period involves two years of daily data from January/2013 to July/2015. It represents 942 days in the total. Selected stores that are open on holidays and weekends were chosen to analyze the seasonality presented in the data. From these stores, one of them were randomly chosen and studied in this work. Figure 4.11 represents the sales for the selected store in 2013, showing that there is some seasonality in the data but not a clear trend.

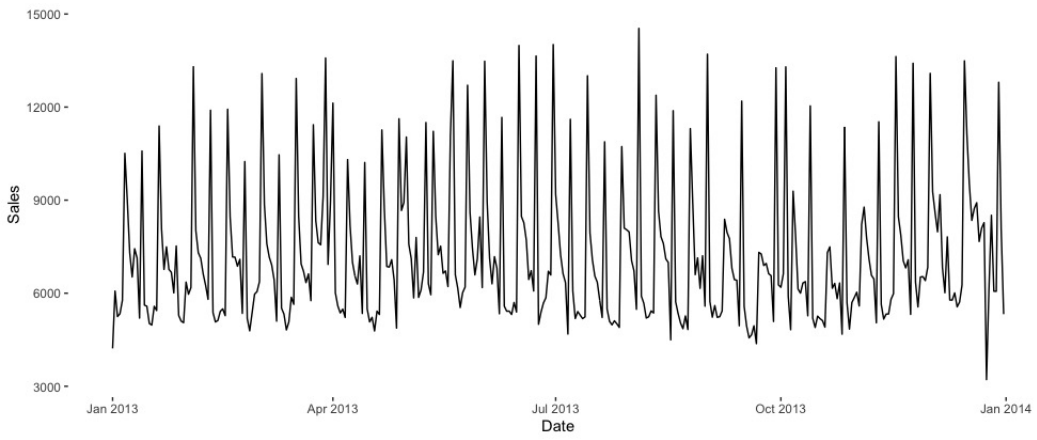

Figure 4.11: Rossmann Sales in 2013

There are four exogenous factors considered for this data, all of them categorical variables:

- Promo: indicates whether a store is running a promotion on that day;

- Weekend: indicates if the day is weekday or weekend, where 0=weekday, 1=Saturday and 2=Sunday;
- State Holiday: indicates a state holiday, where a = public holiday, b = Easter holiday, c = Christmas and 0 = None;
- School Holiday: indicates the closure of public schools.

Figure 4.12 represents the sales behaviour by weekday, showing that on Sundays (day 7), the sales baseline is greater than the sales observed in the other days. Regarding the distribution of the data when promotions are applied, Figure 4.13 shows that there are no promotions on weekends. Overall, weekdays with promotions present larger sales than those without promotions. Sunday is the day of the week with larger sales.

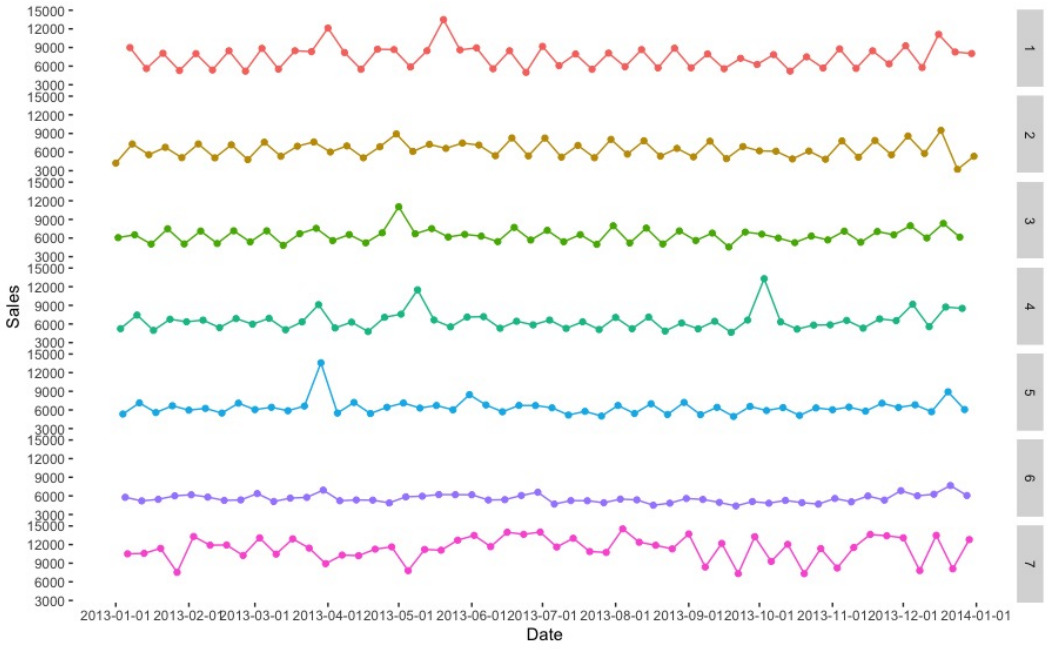

Figure 4.12: Rossmann Sales per weekday in 2013, where 1 represents Monday

Figure 4.14 represents the autocorrelation and partial autocorrelation functions for the series. The plots confirm strong weekly seasonality and hint at other seasonalities in the data. Therefore, past information about sales will be considered until lag seven as input to the Machine Learning models.

Considering the size of the available data, $n = 942$ days, and assuming that in the businesses field a company would use the maximum information as possible to predict the future, a training sample of size $i = 500$ is considered. The goal is to predict the next 30 days of sales, and then, the test sample is fixed equals $t = 30$. Regarding the Monte-Carlo experiment, $R = 100$ iterations are performed.

Table 4.3 summarizes the average error obtained by the best model within each class: ARIMA, SVM, RF and XGBoost.The lowest RMSEs are highlighted in bold.

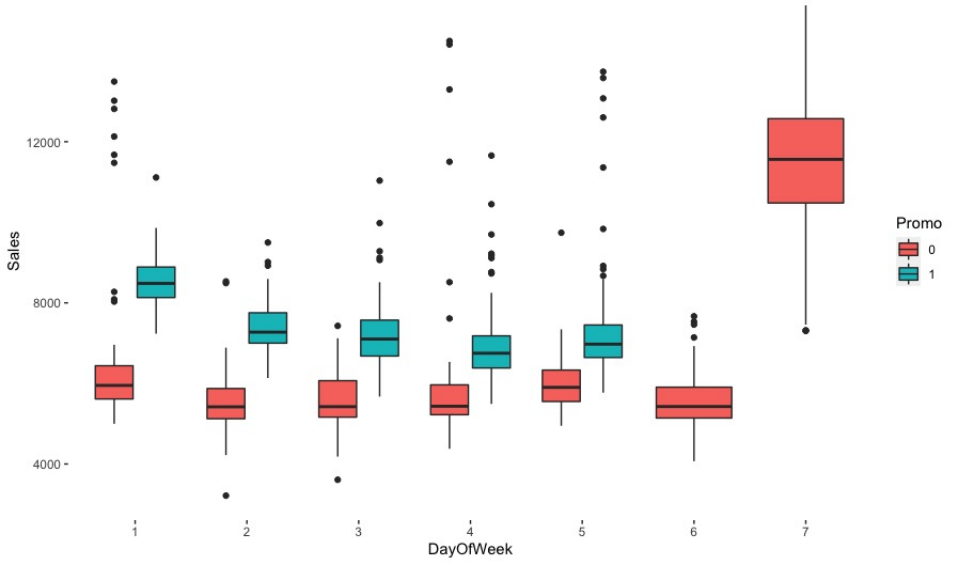

Figure 4.13: Sales distribution regarding promotions events, where 1 represents "Monday"

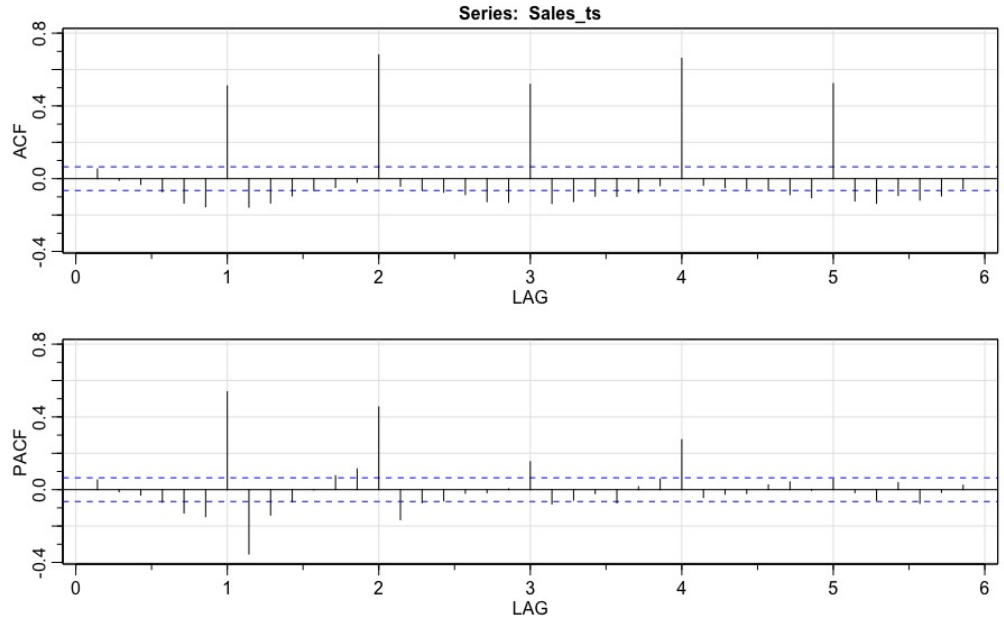

Figure 4.14: ACF and PACF of Rossmann Sales

In general, considering additional information factors leads to forecasts with small RMSE for all ML models fitted. The Meta-ARIMA model presents a slight worsening that can be related to the characteristics of the available drivers. Furthermore, all ML models have performed better than the traditional ARIMA for both cases. To illustrate the behaviour of the forecast obtained from each model, the data set corresponding to the first iteration of the Monte-Carlo procedure was chosen, and the results are presented in Figure 4.15. Overall the results are in accordance

	Meta-ARIMA	SVM	RF	XGboost
Without drivers	**1536**	1336	1475	1452
With drivers	1588	**1035**	**972**	**975**

Table 4.3: Performance of the forecasting models data with and without exogenous information - Sales data

with the RMSE in table 4.3 and clearly in this particular dataset the forecasts generated with the exogenous factors are closer to the observed data.

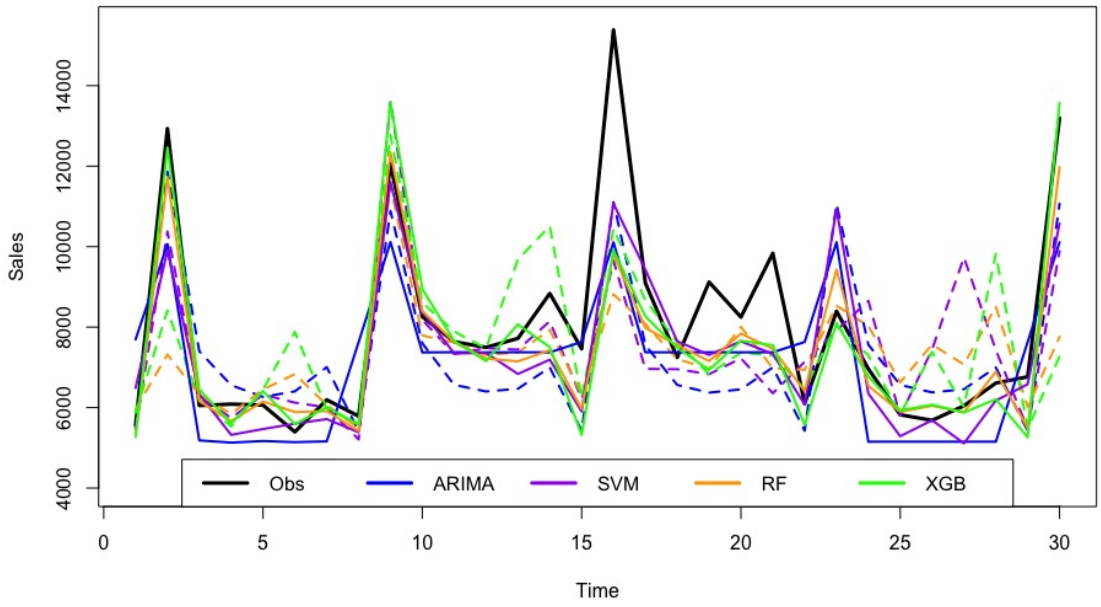

Figure 4.15: Forecasts produced by each model for the Rossmann Sales dataset. The dotted lines represent the forecasts generated from the data without exogenous factors.

When external drivers are included in the models, the ML methods can better map the pattern of the data and then generate forecasts closer to the observed data. On the other hand, the Meta-ARIMA model seems to lose the ability to detect the observed peaks when considering the additional information.

The complete performance estimation results are discussed in Appendix A.2. The final ML models considered are the ones with the lowest RMSE for each method (SVM, RF and XGBoost).

4.3.3 Case 3: Insurance Data

Considering external predictors in a regression model can present an impact that is not always immediate. For instance, if a store promotes some online marketing campaign during a period,

the effect on sales can grow as the information is spread and it may not be in the next day or next week. In these situations, it is necessary to consider the lagged effects of the predictor (more details are given in section 3.1).

To illustrate forecasting with lagged drivers, a data set concerning the effect of TV advertising on insurance quotations is considered. This case of study was inspired by the example provided by R. Hyndman and Athanasopoulos (2018). The data set is from the R package "fpp2" from R. J. Hyndman (2013), and present monthly information about insurance advertising and quotations from January/2002 to April/2005. The data do not present missing values for any variable. Figure 4.16 shows both series. It is possible to note that there is a strong correlation between them.

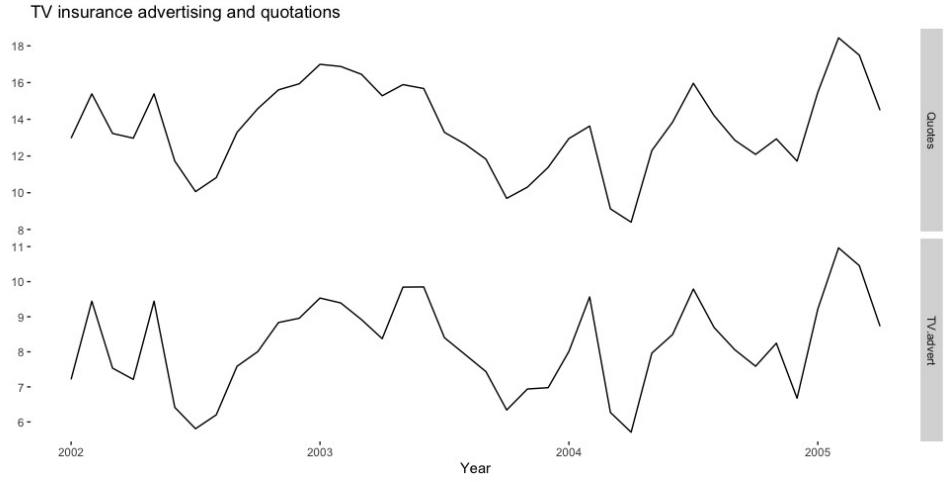

Figure 4.16: Insurance quotes and TV Advertising series

Analysis of 4.17 helps to determine the lagged relationships between quotations and advertising expenditure. It is clear that the strong correlations are contemporaneous, lag 0 , and with the previous month, lag 1. Therefore, as external predictors, only the advertising in the current month and the previous month will be considered. This result following the analysis of R. Hyndman and Athanasopoulos (2018) when analyzing the ARIMA with lowest AICc.

Besides, it is necessary to choose how many lags of the target value to consider as an input of the ML models. Considering the autocorrelation and partial autocorrelation functions at Figure 4.18, just the first lag seems to be significant. Thus, it will be used as input for the ML models. The exogenous factors considered for this data are: (i) TV advertising expenditure in the current month and (ii) TV advertising expenditure in the previous month.

There are just $n = 40$ months observed in the data. Therefore, a training set of size $i = 30$ was used, and the goal is to forecast the next three months, $t = 3$. To evaluate the performance of the models, it was performed $R = 5$ simulations.

Table 4.4 summarizes the average error obtained by the best model within each class: ARIMA, SVM, RF and XGBoost. The lowest RMSEs are highlighted in bold.

In general, considering additional information factors leads to forecasts with small RMSE for all the models fitted, being observed the best accuracy in the SVM and Meta-ARIMA models.

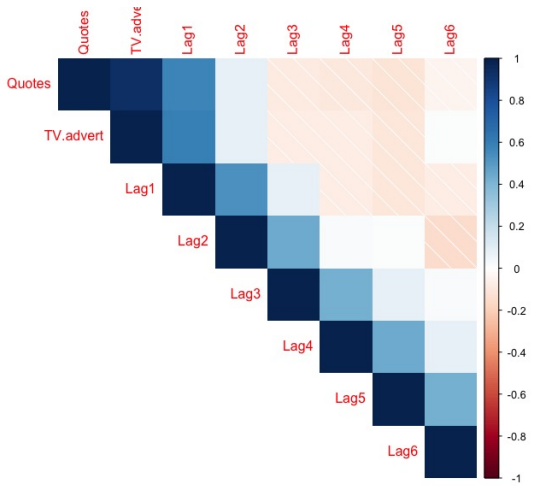

Figure 4.17: Correlation between the quotations and different lags of TV Advertising

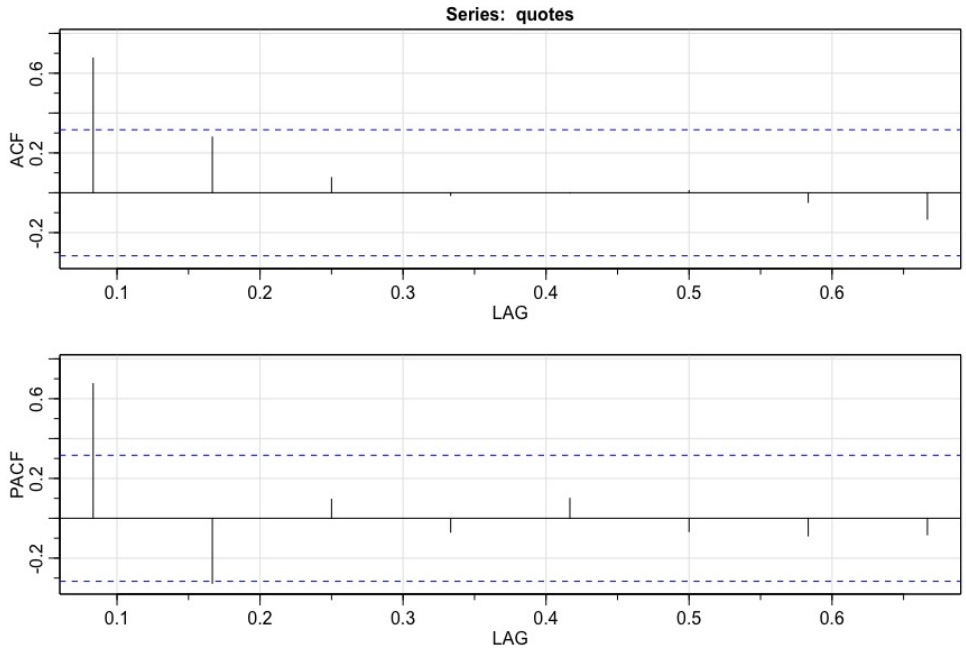

Figure 4.18: ACF and PACF of Insurance quotations

	Meta-ARIMA	SVM	RF	XGboost
Without drivers	2.65	1.98	2.76	2.22
With drivers	**0.73**	**0.60**	**1.58**	**1.33**

Table 4.4: Performance of the forecasting models data with and without exogenous information - Insurance data

To illustrate the behaviour of the forecast obtained from each model, the data set corresponding

to the first iteration of the Monte-Carlo procedure was chosen, and the results are presented in Figure 4.19. Overall the results are in accordance with the RMSE in table 4.4. Although models with exogenous factors generate better forecasts, the forecasts over-estimate the number of insurance quotations for both cases.

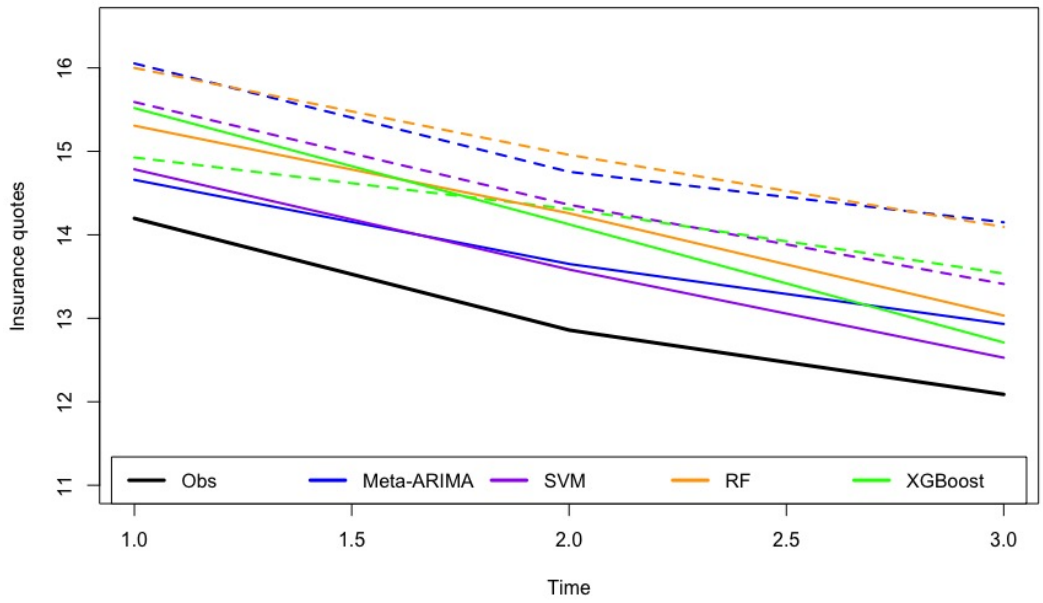

Figure 4.19: Forecasts produced by each model for the Insurance quotes dataset. The dotted lines represent the forecasts generated from the data without exogenous factors.

When external drivers are included, all the models generate forecasts closer to the observed data. However, the models SVM and Meta-ARIMA seem to capture better the dynamics especially for the second month forecasted.

The complete performance estimation results are discussed in Appendix A.3.

4.3.4 Discussion

In general, considering exogenous factors as inputs for the forecasting methods improved the performance of the models and achieved better forecasts. The exception is the Meta-ARIMA for the Sales dataset. This may be due to the characteristics of the time series as well as the ARIMA model chosen by the *auto-arima* function. A summary of the average RMSE obtained for each model for all cases studied is shown in Figure 4.20.

For the Bike Sharing and Insurance datasets, the best performance was achieved by the SVM model. However, for the latter, Meta-ARIMA model also presents a good performance as SVM. The success of the SVM model can be associated with the ability of SVM to solve non-linear regression estimation problems, making the model promising for time series forecasting. For

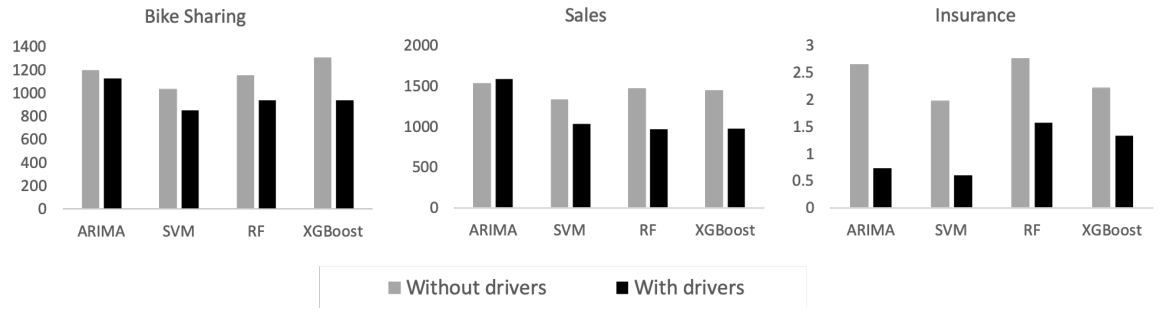

Figure 4.20: Summary RMSE comparison for data considering or not exogenous factors

the Sales dataset, the lowest errors are achieved by Random Forest and XGBoost models. The main difference between these data and the other two cases is that the drivers considered are all categorical variables.

Chapter 5

Conclusions

Generating assertive forecasts is challenging for time series data in general. The search for models that can better capture all the particularities of the data involves several studies and the use of alternative methods, such as supervised Machine Learning algorithms, has become increasingly present. However, generating forecasts that incorporate the effects of external variables on time series while comparing statistical models and ML still seems to be little explored. Including such factors in the forecasting models can be challenging since it may require the support of an expert and sometimes depend on generating the forecasts for those factors. This dissertation discussed the effect of considering exogenous factors as input variables for forecasting time series data and compared the ARIMA model with ML approaches.

As such, in this work, two different datasets with distinct characteristic and containing exogenous variables were studied, and the performance of different models was evaluated in terms of the RMSE metric. Thus, diverse models were fitted in the data, and the performances were compared. The first dataset of each case contains just information about the past values of the target variable. The second dataset includes data from the first dataset and the additional exogenous factors. Furthermore, four models were applied, one traditional time series model, ARIMA, and other three supervised Machine Learning models, Support Vector Machine, Random Forest and XGBoost. Also, a range of parameters for the Machine Learning models was tested. The conclusions refer to the models that present the lowest RMSE.

Overall, the results converge to a universal finding. The main conclusion drawn from this work indicates that considering exogenous factors as inputs for the forecasting methods improve the performance of the models and achieved better forecasts. Nevertheless, the conclusions can vary depending on the model. In the first case studied, the prediction for the number of bike rentals has shown an improvement in all applied models. The information regarding the temperature seems to be significantly correlated with the rentals. Regarding the performance of the ML models and Meta-ARIMA, the three models outperformed the former when extra information is available, and the best performance achieved by the SVM model.

For the Sales prediction, in case two, the ARIMA model used to generate the forecast does not show improvement when the external information is incorporated. In fact, it developed the worst predictions. It can be associated with the characteristics of the additional predictors considered in the model. For the other ML models tested, the results were positive when considering such

factors. Regarding the added value to the business, knowing that the incorporation of these factors generate better forecasts may lead to an improvement in the collection of data to be used in the models. As a result, knowing the indicators allows stores to better plan sales logistics.

For the third and last case, the forecasts for the insurance quotes, also have shown a general improvement on the results when considering advertising information as an input of the models. The SVM model presented the best improvement and as a consequence, the best performance. The results from the Meta-ARIMA model overcome the decision-tree based models.

Despite the results show a general improvement in the forecasts when external information is present in the dataset, the task of choosing which variables to consider as predictors strongly depend on expert analysis and may generate biased results. Further, it is not common to find time series data that contain such drivers. This work shows that including exogenous factors as predictors generates better forecasts for most of the models studied, regarding the three particular series considered. In this sense, more research on this topic is needed to expand the results and to encourage data owners to collect extra information when dealing with time series problems. Moreover, considering ML approaches to this problem seems to be valuable, bringing more precision to the forecasts.

5.1 Future Work

This work focuses on comparing the performance of different forecasting models when considering or not exogenous factors as predictors. Besides, a comparative analysis between ML models and ARIMA for this problem is also performed. In the datasets used, the future values for the exogenous factors in the forecast period were known from the beginning. However, in practice, after mapping the relationship between the response variable and this information, it may be necessary to forecast or make some assumption for these factors when generating the forecasts. For instance, when using temperature as a predictor in a forecasting model, it can be necessary to forecast values of this variable or to assume some average from past values to make the forecasts. Also, the procedure of feature engineering could add value to the interpretation of the models and allow the optimization of the selected external variables. Besides, this work focuses on generating and analysing long-term forecasts, not one-step-ahead. In this sense, a study that replicates the analysis done by including the comparison of different terms would be interesting. Then, three aspects that could be exploited in future research are:

- To extend this work by analyzing the performance of the models when the exogenous factors are unknown for the forecasted period. Therefore, compare the results with data that not contain such factors. The proposal is to analyze if, when an extra task is added to the models, similar results are achieved.

- To perform different methods of feature engineering and see if it generates more accurate forecasts and give more precisely business insights.

- To replicate the analysis done in this work by comparing long-term and short-term forecasts.

Furthermore, it would be interesting to create a framework indicating which factors could be valued depending on the applied field. It could lead to a more assertive data collection.

References

Adhikari, R., & Agrawal, R. (2013). *An introductory study on time series modeling and forecasting*. Lambert Academic Publishing. doi: 10.13140/2.1.2771.8084

Ahmed, N., Atiya, A., Gayar, N., & El-Shishiny, H. (2010). An Empirical Comparison of Machine Learning Models for Time Series Forecasting. *Econometric Reviews*, *29*, 594–621. doi: 10.1080/07474938.2010.481556

Alkali, M. A. (2019). Assessing the forecasting performance of arima and arimax models of residential prices in abuja nigeria. *Asia Proceedings of Social Sciences*, *4*(1), 4-6.

Alpaydin, E. (2020). *Introduction to machine learning, fourth edition*. The MIT Press.

Approximate nonlinear forecasting methods. (2006). In *Handbook of economics forecasting* (pp. 459–512). Elsevier. doi: https://doi.org/10.1016/S1574-0706(05)01009-8

Awad, M., & Khanna, R. (2015). Support vector regression. In *Efficient learning machines: Theories, concepts, and applications for engineers and system designers* (p. 67-80). Berkeley, CA: Apress.

Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2020, Aug). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*. Retrieved from http://dx.doi.org/10.1007/s10462-020-09896-5 doi: 10.1007/s10462-020-09896-5

Bergmeir, C., Hyndman, R., & Koo, B. (2017, 11). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics Data Analysis*, *120*. doi: 10.1016/j.csda.2017.11.003

Berrar, D. (2018). Cross-validation. *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, *1-3*, 542–545. doi: 10.1016/B978-0-12-809633-8.20349-X

Bissacco, A., Yang, M., & Soatto, S. (2007). Fast human pose estimation using appearance and motion via multi-dimensional boosting regression. In *2007 ieee conference on computer vision and pattern recognition* (p. 1-8). doi: 10.1109/CVPR.2007.383129

Bojer, C. S., & Meldgaard, J. P. (2020). Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*. doi: 10.1016/j.ijforecast.2020.07.007

Bontempi, G., Ben Taieb, S., & Le borgne, Y.-A. (2013). Machine learning strategies for time series forecasting. In M.-A. Aufaure & E. Zimányi (Eds.), *Business intelligence* (pp. 62–77). Springer. doi: 10.1007/978-3-642-36318-4_3

Bossche, F., & Brijs, T. (2004). A regression model with ARMA errors to investigate the frequency and severity of road traffic accidents. *83rd annual meeting of the Transportation Research Board*, *32*(0), 15.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. doi: https://doi.org/10.1023/A:1010933404324

Breiman, L., Friedman, J., Stone, C., & Olshen, R. (1984). *Classification and regression trees*. Taylor & Francis.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 121–167. doi: https://doi.org/10.1023/A:1009715923555

Canova, F., & Hansen, B. E. (1995). Are seasonal patterns constant over time? A test for seasonal stability. *Journal of Business and Economic Statistics*, *13*, 237–252. doi: 10.1080/07350015.1995.10524598

Cerqueira, V., Torgo, L., & Mozetic, I. (2019). Evaluating time series forecasting models: An empirical study on performance estimation methods. *CoRR*, *abs/1905.11744*. Retrieved from http://arxiv.org/abs/1905.11744

Cerqueira, V., Torgo, L., & Soares, C. (2019). Machine learning vs statistical methods for time series forecasting: Size matters. *ArXiv*, *abs/1909.13316*.

Chatfield, C. (2000). *Time-series forecasting*. CRC Press.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. doi: 10.1145/2939672.2939785

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., … Li, Y. (2019). xgboost: Extreme gradient boosting [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=xgboost (R package version 0.90.0.2)

Cryer, J., & Chan, K. (2008). *Time series analysis: With Applications in R*. Springer New York.

Cutler, A., Cutler, D. R., & Stevens, J. R. (2009). Tree-based methods. In X. Li & R. Xu (Eds.), *1–19* (pp. 1–19). Springer New York. doi: 10.1007/978-0-387-69765-9_5

Cutler, A., Cutler, D. R., & Stevens, J. R. (2012). Random forests. In C. Zhang & Y. Ma (Eds.), *Ensemble machine learning: Methods and applications* (pp. 157–175). Springer US. doi: 10.1007/978-1-4419-9326-7_5

Datavedas. (2018). *Introduction to time series data*. Retrieved from https://www.datavedas.com/introduction-to-time-series-data

Feng, W., Sui, H., Tu, J., Huang, W., & Sun, K. (2018). A novel change detection approach based on visual saliency and random forest from multi-temporal high-resolution remote-sensing images. *International Journal of Remote Sensing*, *39*(22), 7998–8021. doi: 10.1080/01431161.2018.1479794

Friedman, J. (2000, 11). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*, 1189–1232. doi: 10.1214/aos/1013203451

Gilliland, M. (2019). The value added by machine learning approaches in forecasting. *International Journal of Forecasting*, *36*, 161–166. doi: 10.1016/j.ijforecast.2019.04.016

Gomez, V., & Maravall, A. (2000). Automatic modeling methods for univariate series. In *A course in time series analysis*. Wiley Online Library. doi: 10.1002/9781118032978.CH7

Hannan, E. J., & Rissanen, J. (1982). Recursive estimation of mixed autoregressive-moving average order. *Biometrika*, *69*, 81-94. doi: 10.1093/biomet/69.1.81

Hutchinson, R. A., Liu, L.-P., & Dietterich, T. G. (2011). Incorporating boosted regression trees into ecological latent variable models. In (p. 1343–1348). AAAI Press. doi: 10.5555/2900423.2900636

Hyndman, R., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice* (2nd ed.). OTexts.

Hyndman, R., & Khandakar, Y. (2008). Automatic Time Series Forecasting: The forecast Package for R. *Journal of Statistical Software*, *26*(3), 22. doi: 10.18637/jss.v027.i03

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

Hyndman, R. J. (2013). fpp: Data for "forecasting: principles and practice" [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=fpp (R package version 0.5)

Hyndman, R. J., & Khandakar, Y. (2020). Automatic time series forecasting: the forecast package for R [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=forecast (R package version 3.0.2)

Johnson, R., & Zhang, T. (2014). Learning nonlinear functions using regularized greedy forest. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(5), 942–954. Retrieved from http://dx.doi.org/10.1109/TPAMI.2013.159 doi: 10.1109/tpami.2013.159

Kane, M., Price, N., Scotch, M., & Rabinowitz, P. (2014). Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks. *BMC bioinformatics*, *15*, 276. doi: 10.1186/1471-2105-15-276

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.

Liaw, A., & Wiener, M. (2018). Classification and regression by randomforest [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=randomForest (R package version 3.2.2)

Makridakis, S., & Hibon, M. (2000). The m3-competition: Results, conclusions and implications. *International Journal of Forecasting*, *16*, 451-476. doi: 10.1016/S0169-2070(00)00057-1

Makridakis, S., Hyndman, R., & Petropoulos, F. (2019). Forecasting in social settings: The state of the art. *International Journal of Forecasting*, *36*. doi: 10.1016/j.ijforecast.2019.05.011

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLOS ONE*, *13*(3), 1–26. doi: 10.1371/journal.pone.0194889

Medjahed, S. A. (2015). A comparative study of feature extraction methods in images classification. *International Journal of Image, Graphics and Signal Processing*, *7*, 16-23. doi: 10.5815/ijigsp.2015.03.03

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2019). e1071: Misc functions of the department of statistics, probability theory group (formerly: E1071), tu wien [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=e1071 (R package version 1.7-3)

Müller, K.-R., Smola, A., Rätsch, G., Schölkopf, B., Kohlmorgen, J., & Vapnik, V. (2006). Predicting time series with support vector machines. In *Lecture notes in computer science* (Vol. 1327, p. 999-1004). doi: 10.1007/BFb0020283

Nasteski, V. (2017). An overview of the supervised machine learning methods. *HORIZONS.B*, *4*, 51-62. doi: 10.20544/HORIZONS.B.04.1.17.P05

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, *7*, 21. doi: 10.3389/fnbot.2013.0002

Nielsen, D. (2016). *Tree boosting with xgboost - why does xgboost win "every" machine learning competition?* (Unpublished master's thesis). NTNU - Norwegian University of Science and Technology.

Pan, B. (2018). Application of XGBoost algorithm in hourly PM2.5 concentration prediction. *IOP Conference Series: Earth and Environmental Science*, *113*(1), 0–7. doi: 10.1088/1755-1315/113/1/012127

Pradhan, A. (2012). Support vector machine - A survey. *International Journal of Emerging Technology and Advanced Engineering*, *2*.

R Core Team. (2013). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from http://www.R-project.org/

Sapankevych, N. I., & Sankar, R. (2009). Time series prediction using support vector machines: A survey. *IEEE Computational Intelligence Magazine*, *4*(2), 24-38. doi: 10.1109/MCI.2009.932254

Sharda, R., & Patil, R. (1992). Connectionist approach to time series prediction: An empirical test. *Journal of Intelligent Manufacturing*, *3*, 317-323.

Shcherbakov, M., Brebels, A., Shcherbakova, N., Tyukov, A., Janovsky, T., & Kamaev, V. (2013). A survey of forecast error measures. *World Applied Sciences Journal*, *24*, 171-176. doi: 10.5829/idosi.wasj.2013.24.itmies.80032

Shumway, R. H., & Stoffer, D. S. (2016). *Time series analysis and its applications with r examples ez edition* (Green Edition ed.). Springer-Verlag New York. doi: 10.1007/978-3-319-52452-8

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. , *14*(3), 199–222. doi: 10.1023/B:STCO.0000035301.49549.88

Suhartono, M. H., Lee, & Prastyo, D. D. (2015). Two levels ARIMAX and regression models for forecasting time series data with calendar variation effects. In *American institute of physics conference series* (Vol. 1691).

Swanson, N., & White, H. (1995). A model-selection approach to assessing the information in the term structure using linear models and artificial neural networks. *Journal of Business and Economic Statistics*, *13*(3), 265–275. doi: 10.1080/07350015.1995.10524600

Takens, F. (1981). Detecting strange attractors in turbulance. In D. Rand & L. Young (Eds.), (Vol. 898, p. 366–381). Springer.

Torgo, L. (2014). *An Infra-Structure for Performance Estimation and Experimental Comparison of Predictive Models in R.*

Torgo, L. (2017). *Data mining with r: Learning with case studies, second edition* (2nd ed.). Chapman 'I&' Hall/CRC.

Vapnik, V. (1995). *The nature of statistical learning theory* (Vol. 6). Springer.

Zhang, P., Patuwo, E., & Hu, M. (1998a). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, *14*, 35-62. doi: 10.1016/S0169-2070(97)00044-7

Zhang, P., Patuwo, E., & Hu, M. (1998b). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, *14*, 35-62. doi: 10.1016/S0169-2070(97)00044-7

Zhang, P., & Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*, *160*, 501-514. doi: 10.1016/j.ejor.2003.08.037

Zhao, L. (2011). Neural networks in business time series forecasting: Benefits and problems. *Review of Business Information Systems (RBIS)*, *13*. doi: 10.19030/rbis.v13i3.4324

Appendix

A Monte-Carlo performance estimation results

A.1 Bike Sharing data

The Monte-Carlo performance estimation results for the data without exogenous factors are shown in Figure A.1. Figure A.2 represents these results for data with the additional information. The results show that, for data without external information, the performance of SVM seems to be consistency regarding the choice of parameters, being the model with best performance. However, SVM results are more sensitive to the parameters tested when exogenous factors are included. However, for the right choice of parameters SVM presents the smallest RMSE variability across samples. The final ML models considered are the ones with the lowest RMSE for each method (SVM, RF and XGBoost).

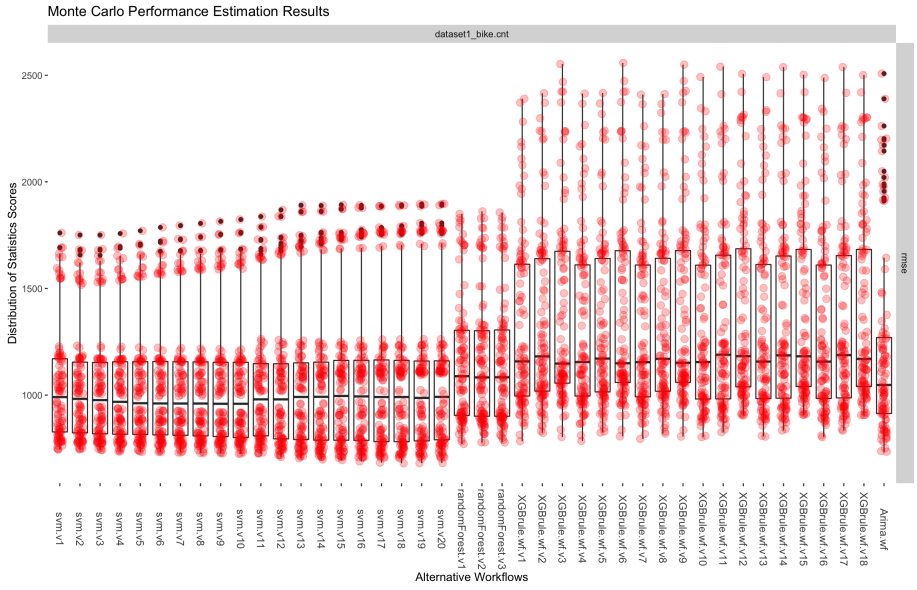

Figure A.1: Results from the Monte-Carlo procedure for the Bike Sharing series considering data without exogenous factors

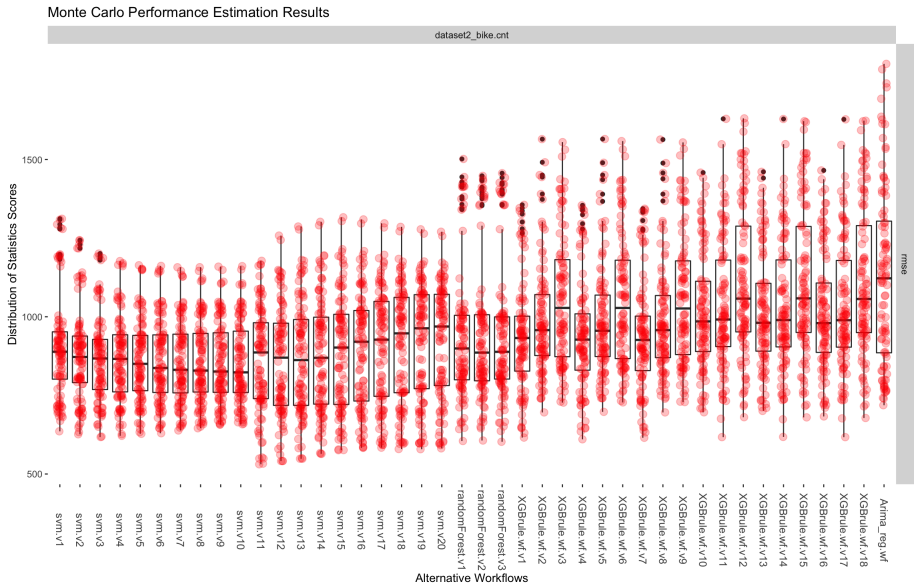

Figure A.2: Results from the Monte-Carlo procedure for the Bike Sharing series considering data with exogenous factors

A.2 Sales data

The Monte-Carlo performance estimation results for the data without exogenous factors are shown in Figure A.3. Figure A.4 shows these results for data with the additional information. The results show that the performance of SVM is sensitive to the choice of parameters for the both cases studied and RF e XGboost present similar results, being RF the model with smallest RMSE variability across samples. The final ML models considered are the ones with the lowest RMSE for each method (SVM, RF and XGBoost).

A.3 Insurance data

The Monte-Carlo performance estimation results for the data without exogenous factors are shown in Figure A.5. Figure A.6 show these results for data with the additional information. Note that, the series is too short and just five iterations were performed, so the analysis can not be conclusive. The final ML models considered are the ones with the lowest RMSE for each method (SVM, RF and XGBoost).

Figure A.3: Results from the Monte-Carlo procedure for the Rossmann Sales series considering data without exogenous factors

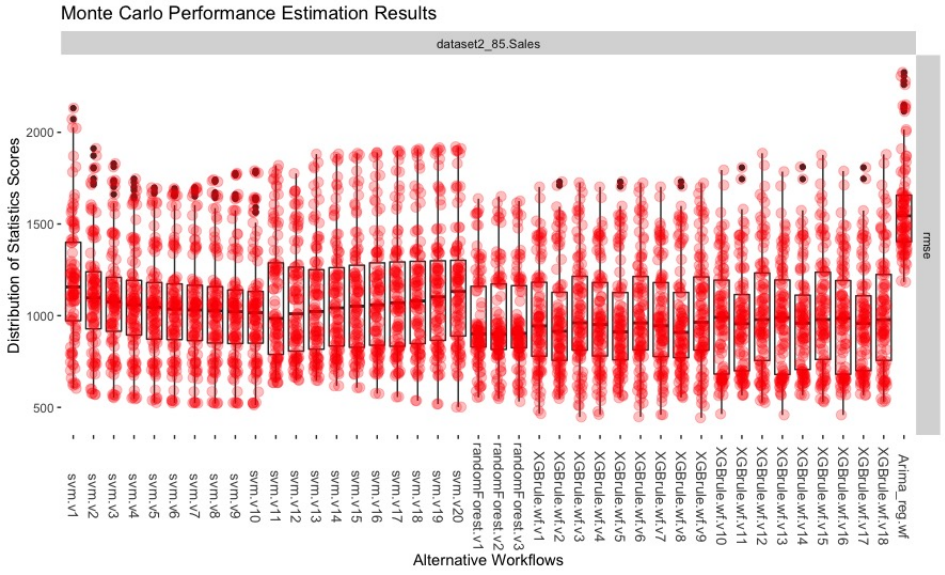

Figure A.4: Results from the Monte-Carlo procedure for the Rossmann Sales series considering data with exogenous factors

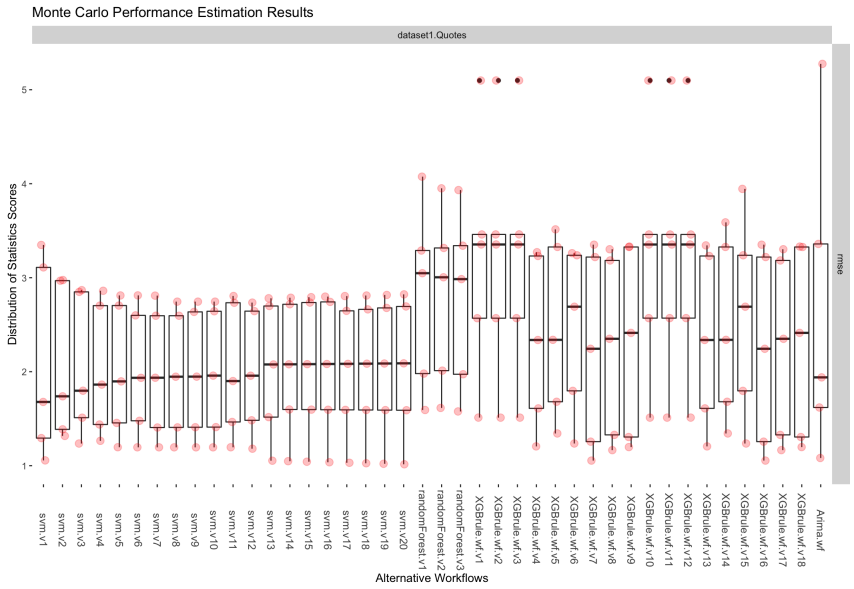

Figure A.5: Results from the Monte-Carlo procedure for the Insurance quotations series considering data without exogenous factors

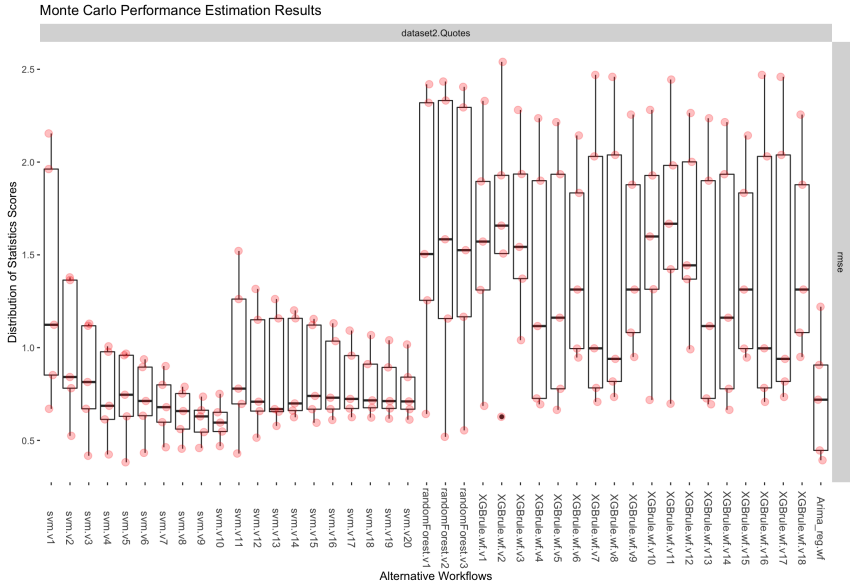

Figure A.6: Results from the Monte-Carlo procedure for the Insurance quotations series considering data with exogenous factors