Universidade de Aveiro
2021

Alina Lysenko

Previsão de consumos de água com recurso a séries temporais

Water demand forecasting using time series

Universidade de Aveiro
2021

Alina Lysenko

Previsão de consumos de água com recurso a séries temporais

Water demand forecasting using time series

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizada sob orientação científica de Professor Doutor António Gil D'Orey de Andrade Campos, Professor Auxiliar Com Agregação do Departamento de Engenharia Mecânica da Universidade de Aveiro.

Esta dissertação teve o apoio dos projetos UIDB/00481/2020 e UIDP/00481/2020 - Fundação para a Ciência e a Tecnologia; e CENTRO-01-0145 FEDER-022083 - Programa Operacional Regional do Centro (Centro2020), através do Portugal 2020 e do Fundo Europeu de Desenvolvimento Regional.

O júri / The jury

Agradecimentos / Acknowledgements

I start by leaving a special thank you to my parents, who always believed in me. For the effort they put into bringing me to this wonderful country that is Portugal and ensuring a quality education that would otherwise not be possible. To my sister for her words of motivation. To my classmates Pedro Teixeira, Tiago Delgado and Tiago Crispim who always showed concern for my progress, your intervention had a great impact on me. To my boyfriend and to his family, for giving me endless support and patience. To all the professors of the department for sharing your wisdom and showing willingness to help. A big thank you to my adviser Gil Campos for being patient, available and concerned about me, for ensuring all the tools to finish this work. To everyone who contributed directly or indirectly to this stage of my education.

Palavras-chave

Sistemas de abastecimento de água; previsão de consumos de água; séries temporais; métodos de previsão; ARIMA; modelo heurístico de previsão

Resumo

A área de otimização dos processos tem vindo a ter um interesse crescente por parte das empresas. Para uma empresa de gestão de abastecimento de água, a otimização de distribuição de água significa coordenar, de forma eficiente, os processos de recolha, tratamento e distribuição de água num único processo. Contudo, para planear de forma eficiente o funcionamento dos sistemas de abastecimento de água é essencial prever as demandas de água. A tarefa de previsão é possível se algumas variáveis forem conhecidas previamente, como os dias de chuva que influenciam diretamente o processo de recolha de água. Tal como as condições meteorológicas, existem outras variáveis que afetam o sucesso de uma boa previsão de demanda de água, como os dias de férias. Um elevando número de variáveis requer modelos de previsão cada vez mais sofisticados que requerem maior capacidade de processamento de informação. Esses modelos nem sempre encontram uma solucão razoável e, na prática, as variáveis necessárias para treinar um modelo robusto nem sempre estão disponíveis. Essa dificuldade é superada com a implementação de modelos de previsão baseados somente em séries-temporais. São computacionalmente mais simples e não requerem dados de entidades externas às empresas de abastecimento de água. O foco deste trabalho é analisar, implementar e testar algoritmos de previsão baseados em séries temporais aplicadas à demanda de água, tendo como variável de entrada apenas o histórico de consumo de água. A primeira etapa deste trabalho consistiu em estudar o histórico do consumo de água na região de Penacova para traçar padrões temporais de consumo. A segunda etapa trata da implementação de um modelo clássico de previsão ARIMA e da implementação de um modelo heurístico de Bakker. Ambos os modelos foram comparados e diferentes vantagens e desvantagens foram analisadas.

Keywords Water supply system (WSS); water demand forecasting; time series; forecasting methods; ARIMA; heuristic forecasting model

Abstract

Process optimization has been an area of growing interest for companies. For a water supply management company, water distribution optimization means coordinating efficiently the water harvesting, treatment and distribution stages in a single process. However, to plan efficiently the operation of water supply systems it is mandatory to forecast the water demands. The forecasting task is possible if some variables are previously known, such as the rainy days that directly influence the harvest stage. As well as weather conditions, there are other variables that affect the success of a good water demand forecast, such as holidays. The large number of variables requires increasingly sophisticated forecasting models that require greater information processing capacity. These models do not always find acceptable solution, and, in practice, the variables needed to train a robust model are not always available. This difficulty is overcome by the implementation of forecast models based solely on the time-series. These are computationally simpler and do not require data from entities outside the Water Supply companies. The focus of this work is to analyze, implement and test forecasting algorithms based on time series applied to water demand, having as input variables only the history of water consumption. The first stage of this work consisted of studying the history of water consumption in the Penacova area to trace temporal patterns in the data. The second stage deals with the implementation of a classical ARIMA prediction model and implementation of a Bakker heuristic model. Both models were compared and different advantages and disadvantages were analyzed.

Contents

List of Tables

List of Figures

Part I

Framework

Chapter 1

Introduction

1.1 Context and objectives

Nowadays, due to computational advances, the behaviour of several systems can be predicted. The main objective of predicting the behaviours of a system is to act in advance. When the prediction of a system is related to a company, the common advantages are events scheduling, optimisation of processes and incidents prevention.

Water Supply and Distributions system are objective of several studies due to their high complexity. Such systems consist of four main phases that are described by water collection, water treatment and storage, water transport in the network and distribution network [1]. Water supply companies are responsible for the distribution of good quality water to the consumers. The good quality water is obtained by satisfaction of legislative norms as well as by ensuring the certain demand pressure.

To design a Water Supply System (WSS), water supply companies consider the system implementation and distribution costs, in order to minimise them. The common distribution costs are associated to operation control, maintenance of equipment and troubleshoot, for example water losses caused by pipe burst. The operational control includes procedures such pump control scheduling, valves control, starting a filtration process, etc. It is important to consider that while a system is active, there will be wear that needs to be compensated by maintaining procedures and, in unforeseen cases, water losses may occur. Companies started to invest in automation of a water supply system by installing supervisory control and data acquisition (SCADA) equipment. In fact, the procedure of collecting data is essential to extract water consumption patterns and to find the relation between this patterns and external factors as weather, day of the week, time of the year (e.g. vacations and holidays), past consumptions and geographic location. Considering the prediction advantages, studies were carried out on the application of traditional forecasting techniques as well as more recent techniques based on Artificial Neural Network (ANN). Besides that, some authors combine methods statistically with each other. The main differences, advantages, disadvantages and relevant work review will be addressed in the State-of-art review section.

Water supply companies can take advantage by predicting the water demand of consumers. The optimisation of scheduling processes related with water demand is an advantage to highlight. So, the work developed in this thesis consists in the implementation and analysis of different forecasting techniques to the new case study. The application of the traditional methods in another case study, although it seems redundant, is necessary to validate and evaluate the effectiveness and robustness of the implemented techniques. In [2] is referred that several researched works based on Machine Learning (ML) forecasting does not make a side-by-side comparison with traditional (classic) forecasting methods which may compromise the effectiveness of conclusions.

Chapter 2

State-of-the-art Review

2.1 Generalities concerning water demand forecasting

The prediction based on time-series is called time-series forecasting. Water demand forecasting can be categorized, in terms of forecast horizon, as short-term, medium-term or long-term which represents forecasting objectives. In [3] different scales for this categorization is discussed. In [4] a Dynamic Artificial Neural Network model was developed and applied on different forecasting time scales. A more recent work of mid-term water demand forecasting is presented in [5] and [6]. In order to clarify the approach of the developed work, in the short-term scales is considered time ranges as weekly, daily, hourly and even time frames portioned in minutes forecasting. In the other hand, time ranges of months to decades are included to medium/long-term scale [7]. The main purposes of long-term water demand forecasting are network planning, system design and future management of resources. Short-term water demand forecasting is required for schedule processes, for water quality insurance and for network failures detection [8].

In general, water demand forecasting models uses previous water consumptions data to find consumption patterns. It is possible to include other variables that may influence consumption such weather or price of water to increase forecasting accuracy. In fact, it is preferable to use only easily collectable data. The main concern about using is related to error propagation. For example, weather as input is not practical for water supply companies. The collection of weather related data from another source is certainly accompanied with gaps and some noises. Methods that use one variable as input are commonly related to time series models and are called, for that reason, univariate time series models [9]. Recently, large number of studies uses classical methods (e.g. ARIMA, exponential smoothing, naïve model) for comparison and validation of developed methods purposes [4; 7; 10; 11].

From another point of view, it is possible to identify two basic techniques of water demand forecasting. Both techniques uses past demand data as input but the main difference is that (i) the first search relationship between present and past data demand (such seasonality or patterns) and the (ii) second is based on mathematical formulation, which includes external variables data (such as demographic factors or weather) [8]. Despite this, another group or forecasting methods should be highlighted: hybrid models, which results from combination of different singular models [3]. In [11], [12] and [13], it is proven that hybrid models forecasting outperform the singular models.

Water demand forecasting is object of several studies. Historically, the most common adopted techniques were based on time-series (e.g. exponential smoothing and ARIMA) analysis and regression analysis due to of their implementation simplicity. However, in scientific community, Machine Learning (ML) algorithms gained a special attention because of their learning capability [2]. This capability can be used for forecasting purposes as well as water demand forecasting. The most usual ML algorithm is Artificial Neural Network (ANN). It is relevant to clarify that regression analysis also makes part of ML algorithms category. Some of the referred examples of regression analysis in [7] and [8] are project pursuit regression (PPR), multivariate adaptive regression splines (MARS), support vector regression (SVR), multiple linear regression (MLR) and others. Large number of papers in the literature implement ANN algorithms in parallel with other ones (time-series and/or regression analysis) for comparison purposes [4; 7; 8; 10; 14; 15; 16; 17]. The reported studies show that forecasting ANN based models outperform the other ones. Despite these conclusions in [8], the best forecasting results are obtained from a Support Vector Regression (SVR) model over an ANN model.

Forecasting models based on ML algorithms are computationally demanding and, therefore, it is important to carefully choose input variables for more accurate output. The common example is taken from [14] where the occurrence of rainfall (binary input) showed better weekly forecasting results than rainfall amount for City of Kanpur (India). Contradictorily, in [15] a similar study for City of Ottawa (Canada) showed more accurate weekly forecast for variable of rainfall amount. From these analogies it is possible to highlight the biggest drawback of ML algorithms: high computational cost, high dependency of external data sources for accurate results and lack of robustness itself.

Focusing on ML algorithms, in [18] it is proposed an abductive network approach for electric load short-term forecasting (hourly and daily). The focus of [18] is to introduce a solution to overcome ML limitations. The used approach is inductive self-organizing group method of data handling based on Group Method of Dada Handling (GMDH) theory. Comparing GMDH approach to neural network algorithms, the pointed-out advantages are faster model development and faster convergence during it synthesis, automatic selection of proper input variables and automatic model structure assemble. In [19] a similar study applied to wind speed forecast is presented. In the conclusions, an improvement of the forecast performance and the avoidances of unnecessary inputs, such as effects of noise and errors, are reported. In the forecasting approach, the basis of GMDH method consists of generation of forecasting models by testing a set of models-candidates to choose the more accurate one(s) considering an established criterion. Most of the GMDH algorithms use polynomial mathematical expressions as reference. This method searches for the most significant system inputs based on the system output analysis during the model synthesis [20]. Other studies, such as [9], [16] and [21], suggests fully adaptive methods based on data-driven and/or moving window programming. This approach seems to be useful because of their independency from external data sources (e.g. weather). In the other hand, this type of models requires a lower amount of data, where only the more recent data is relevant, and set of calibration parameters are updated quickly in comparison with conventional time-series models. Although the majority of the studies suggest an hourly or daily short-term forecast, in [21] it is described a model that forecasts water demand for each 15 minutes step in the next 48 hours without weather input. The advantage of this forecast refinement is that it is possible to make a more detailed control, which means that knowing the exact time to switch pump adds optimization margin in process control.

2.2 Water demand forecasting and network failures

In water supply systems (WSS), occurrence of network failures is also a topic of concern. The main reasons that can lead to water distribution network failure is pipe bursts. Indeed, the occurrence of pipe bursts is an expected event due to the high wear that these suffer. The biggest difficulty of WSS companies are about failure time and location of the occurrence in the network identification. Usually, the identification of network failures are reported by consumers when they realize that water pressure is lower [21]. Until the report, water losses occur. In this context, it is proposed to use water demand forecasting for the water demand anomalies. These anomalies can be associated with water losses from pipe bursts.

Part II

Methodology and Implementation

Chapter 3

Selected forecasting techniques

In time-series forecasting there is a varied number of approaches that must be selected carefully. Despite this it is necessary to apply a few approaches to select the most accurate one(s). Considering the wide availability of forecasting resources and the forecaster experience, the choice of the most accurate method is not always evident. In this community it is common to use specific methods for comparison and validation purposes.

In this chapter three time-series forecasting models and selected forecasting accuracy methods will be presented. Before proceeding with the models, it is necessary to recognize the difference between choosing approaches in an industrial or in an academic environment. Thus, the valued features for each one differs slightly. The accuracy is the most important feature however for the same amount of data, it may be preferable to choose a known method that has a short implementation and processing time over a complex method with higher accuracy, for example. In the industrial environment, it matters to use an optimized forecasting method, which will rely on a utility function or on developers' experience. Meanwhile in academic environment, the interest is testing new approaches and draw conclusions to contribute to the forecasting field community.

Hence, Autoregressive Integrated Moving Average (ARIMA) is a known model, usually, applied on case studies by default where data is not stationary. In the opposition to the previous analytical time-series model, the adaptative Bakker's model (heuristic) that shows interesting results for six different water supply zones in the Netherlands but has no application in southern regions case-studies.

Before proceeding with the implementation of the forecasting model, forecasters have a wide choice of solutions. So, it is convenient to have a measure to evaluate the performance of the available models. This evaluation is made through the estimation of the model's accuracy. Despite the simplicity of estimating and reading the accuracy results, this task must be done with caution as the accuracy measurements are not the same for most of the forecasting models. Note that even with the same measures of accuracy, models cannot be compared side by side if case studies are very different in terms, for example, of the amount or origin of data. In [22], accuracy measures are categorized into three main groups: scale dependent errors, percentage errors and scaled errors. Each error group is best suited for a certain type of case study. Therefore, percentage errors are preferable in most of the cases, however, there are studies where this measure is not ideal. For example, in situations where one intends to evaluate different forecast models but using the same dataset, measures based on scaled-dependent errors should be preferred [23].

3.1 Seasonal autoregressive integrated moving average model

Autoregressive Integrated Moving Average is a model commonly used in time-series forecasting by default. Apart from the model being combination of autoregressive (AR) and moving average (MA) parameters, it also offers a tool to make time-series being stationary, an integrated parameter (I).

In time-series forecasting, AR models suggests that the forecasted value is a result of an autocorrelation between past values (also called lags) and can be written as [22]:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + e_t, \tag{3.1}$$

where y_t , c and e_t are the forecasted value at time t, a constant and an error at time t (also called white noise), respectively, and ϕ_1, ϕ_2, ..., ϕ_p are regression parameters until order p.

MA models are commonly used to trace trends in dataset, so it is a very simple and useful tool to predict long term trends. Since the data set must be stationary (no trend), this approach covers a hidden pattern that may exist between a datapoint in a time t and the error in the previous time $t-1$ by autocorrelation [24]. A moving average process is given as [22]:

$$y_t = c + \theta_1 y_{t-1} + \theta_2 y_{t-2} + \ldots + \theta_p y_{t-q} + e_t, \tag{3.2}$$

where y_t , c and e_t are the forecasted value at time t, a constant and an error at time t, respectively, and θ_1, θ_2, ..., θ_q are moving average parameters until order q.

Thus, ARIMA model is usually referred as ARIMA (p,d,q) where p, d and q represent autoregressive, integrated and moving average orders, respectively. In fact, ARIMA is an extended version of Autoregressive Moving Average (ARMA) but combined with differencing step of time-series ("integrated" parameter) and it is useful when data shows to be non-stationary. ARMA model is represented as [22] :

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \theta_1 y_{t-1} + \theta_2 y_{t-2} + \ldots + \theta_p y_{t-q} + e_t, \tag{3.3}$$

where y_t, c and e_t are the forecasted value at time t, a constant and an error at time t, respectively, ϕ_1, ϕ_2, ..., ϕ_p are regression parameters and θ_1, θ_2, ..., θ_q are moving average parameters.

For ARIMA(p,d,q) model the approach is almost the same [22]:

$$y'_t = c + \phi_1 y'_{t-1} + \phi_2 y'_{t-2} + \ldots + \phi_p y'_{t-p} + \theta_1 y'_{t-1} + \theta_2 y'_{t-2} + \ldots + \theta_p y'_{t-q} + e_t, \tag{3.4}$$

where y'_t represents the differenced series. Parameter d represents number of times the time series is differenced until it gets stationary.

Before running ARIMA model, it is necessary to guarantee that time-series is stationary. In time-series analysis it is important establish statistical accuracy all over data sample, which means that some properties must not change over the time. In [22], a stationary series is described as roughly horizontal (mean is constant), with a constant variance and without patterns predictable in the long-term. Hereupon, ARIMA model offers a tool that makes a time series stationary, an integrated parameter. Thus, a series must be differenced enough times to appear stationary.

The most common and practical way to estimate AR and MA parameters is through a visual analysis of the autocorrelation (ACF) and partial autocorrelation (PACF) functions. In [25] ACF and PACF are deeply explained. Table 3.1 summarizes the parameters definition through the visual analysis of ACF and PACF plots.

Table 3.1: AR and MA parameters identification based on ACF and PACF plots. Adapted from [25]

ACFs	PACFs	Model
Exponential decay and/or damped sinusoid	Cuts off after lag p	AR (p)
Cuts off after lag q	Exponential decay and/or damped sinusoid	MA (q)
Exponential decay and/or damped sinusoid	Exponential decay and/or damped sinusoid	ARMA (p, q)

There are time series that have a seasonal behavior. For these cases the seasonal differencing is applied. However, the simple seasonal differencing does not guarantee a complete elimination of seasonal features [25], such as ACF and PACF between seasonal lags, often referred to as m. For this reason, Seasonal Autoregressive Integrated Moving Average model is generally referred as SARIMA$(p,d,q)(P,D,Q)_m$.

The complete implementation of SARIMA model can be found in Appendix B. The Python module used to implement SARIMA model it is called statsmodels. Note that data pre-processing step was applied followed by resample function to set time-steps from 10 minutes to 30 minutes time-step. This resampling makes the 24 hours seasonality implicit in parameter $m = 48$.

3.2 Adaptive Bakker's model

Adaptive Bakker's model is a heuristic model since its methodology is based on empirical rules. This means that some coefficients are set by default. The basis of the model is to forecast water demand for the next 48 hours with 15-min time steps using data from the last 48 hours as input. However, in this study, 20-min time steps were used because data were collected in 10-min time steps and were reshaped in 20-min time steps. This model consists of three steps: 1. Average water demand estimation for the next 48 hours; 2. Average water demand estimation for each time-step for the next 48 hours; 3. Extra sprinkle water demand estimation for each time-step for the next 48 hours, when applicable.

The first step can be divided into two sub-steps. Firstly, it is calculated an array of corrected water demand ($Q_{\text{corr},t}$) based on previous 48 hours hence the factor correspond to the previous two days. This is done by dividing the collected data (Q_t) by the typical day of the week factor ($f_{dotw,typ,i}$):

$$Q_{\text{corr}}, t = \frac{Q_t}{f_{dotw,typ,i}}. \tag{3.5}$$

Then, the average water demand value for the next 48 hours, ($Q_{\text{forc,corr,avg}}$) is, basically, the mean of total number of t steps. In this case, the total number of 20-min.

steps in 48 hours:

$$Q_{\text{forc,corr,avg}} = C_1 \cdot \left(\frac{1}{72} \sum_{t=-72}^{t=0} Q_{\text{corr}}, t \right) + C_2 \cdot \left(\frac{1}{144} \sum_{t=-144}^{t=-73} Q_{\text{corr}}, t \right), \tag{3.6}$$

where C_1 and C_2 are set at 0.85 and 0.15, respectively which means that the water demand values from last 24 hours are heavier than the older one.

In step two is calculated an average water demand for each time-step for the next 48 hours and it comes in the form of an array ($Q_{\text{forc,norm},t}$) . To achieve this result, the average water demand value ($Q_{\text{forc,corr,avg}}$) is multiplied by the typical day of the week factor ($f_{dow,typ,i}$) and by the typical 20-min. time step factor ($f_{qtr,typ,i,j}$):

$$Q_{\text{forc,norm,t}} = Q_{\text{forc,corr,avg}} \cdot f_{dotw,typ,i} \cdot f_{qtr,typ,i,j}. \tag{3.7}$$

The third step is an extra step for those cases where extra water demand occurs. The identification is based on fitting the normal demand curve on the collected water demand curve. A more detailed explanation about the identification and calculation of sprinkle water demand (Q_{sprinkle}) can be found in [21]. The final equation can be written as:

$$Q_{\text{forc,tot,t}} = Q_{\text{forc,norm,t}} + Q_{\text{forc,sprink,t}}, \tag{3.8}$$

where $Q_{\text{forc,tot},t}$ and $Q_{\text{forc,sprink},t}$ are total water demand forecast and average sprinkle demand forecast, respectively. However, to achieve $Q_{\text{forc,sprink},t}$ from Q_{sprinkle} the methodology is quite de same as described for $Q_{\text{forc,norm},t}$.

The calculation of typical 20-min. time step factor ($f_{qtr,typ,i,j}$) is done in two steps. Firstly, it is calculated factor for the normal demand ($f_{qtr,i,j}$) for each t. Secondly, is done the mean of all stored factors where n refers to number of previous stored arrays of factors. Both are calculated as

$$f_{qtr,i,j} = \frac{Q_t}{\frac{1}{144} \sum_{t=-144}^{t=0} Q_t}, \quad \text{and} \tag{3.9}$$

$$f_{qtr,typ,ti,j} = \frac{1}{n} \sum_{i=1}^{i=n} f_{qtr,\{typ=ti\},j}. \tag{3.10}$$

The typical day of the week factor ($f_{dotw,typ,i}$) is obtained as follows and m represent a number of last observations of that type of water demand:

$$f_{dow,typ,ti} = \frac{\frac{1}{m} \sum_{i=1}^{i=m} Q_{avg,\{typ=ti\},i}}{\frac{1}{m \cdot 7} \sum_{i=1}^{i=m \cdot 7} Q_{avg,all,i}}. \tag{3.11}$$

From [21], n and m are set 5 and 10 respectively. This type of approach resembles to rolling window forecasting since factors are estimated from last water demand observations. The advantage is that factors adjust as the season changes quickly enough.

The calculation of $Q_{\text{forc,sprink,t}}$ part is done by the same way as $Q_{\text{forc,norm,t}}$ but it is not described since this part is not applicable in case study of this work as justified in Section 4.2 (Data Analysis section). More detailed information can be found in [21].

The complete implementation of Bakkers heuristic model can be found in Appendix C. The code is splitted in two parts: 1. training stage, where the typical factors are estimated, both $f_{qtr,i,j}$ and $f_{dow,typ,ti}$ based on water demand observations from previous weeks; 2. forecasting stage, where actual forecast for next 48 hours occurrs based on past 48 hours of water demand consumpion. The forecasted values are obtained by setting input variables: input_day, input_month, input_year and n_days_fore. In the Appendix C it is forecasted one week of water demand starting at 23rd December 2019.

3.3 Selected forecasting accuracy methods

The performance measures chosen for this work and its datasets are one percentage and one scale-dependent errors which are: The means Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE), respectively. Since the purpose of the case study was to evaluate different forecasting models in the same dataset, the choice of a scaled error measure make sense. In addition, percentage errors were also calculated for comparisons outside of this case study. These errors are calculated according to the following equations:

$$MAPE = \frac{\frac{1}{n}\sum_{i=1}^{i=n} |y_i - \widehat{y_i}|}{\overline{y}} \cdot 100\%, \quad \text{and} \tag{3.12}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{i=n} |y_i - \widehat{y_i}|\,, \tag{3.13}$$

where y_i is the measured value and $\widehat{y_i}$ is the forecasted value.

The accuracy measurement was made for both water demand per 24 hours and per 30 and 20 minutes time-step, for SARIMA and for heuristic Bakker's model, respectively. For the 24 hours evaluation, an average of the values of the 20 minutes (and 30 minutes) time-step was taken. For the 20 minutes time step each time step forecast is compared to the measured value.

Chapter 4

Data pre-processing

In most of the cases it is not possible to use raw data as input in forecasting model. Due to the nature of temporal data, it is usual to find contaminations as result of unexpected interventions. In data science this occurrence is called anomaly or outlier. Despite this there is a subtle difference between both. A simple way to differentiate both terms it is assuming that when an observation deviates from dataset in the model training stage, that observation is outlier; when the same occurrence is observed in dataset that was not used in training stage it is called anomaly. It is common to use both terms as meaning the same, which is unexpected events in the dataset. These terms will be used in this document interchangeably.

The main objective of identifying anomalies in dataset is to treat them. The amount of treatment will affect the quality of the data and, consequently, of the calibration of the model in the training stage. In [26], it was concluded that anomalies have the biggest impact on forecast at the forecast origin. So, it becomes clear the need of anomaly detection in forecasting applications. In summary, there are two types of anomaly detection approaches: feature-based, which evaluate data features, such as mean and standard deviation, and model-based which compares the predicted values with the real ones. Both approaches consist in establishing a threshold value (for data features and for data itself, respectively) then each point from the real dataset is analyzed and whenever its value fall outside the threshold it is considered as an anomaly.

The data pre-processing stage is not the same for all data types, but it commonly involves two main steps: (i) outliers and missing data identification and (ii) treatment of identified values.

4.1 Outliers detection

Before proceeding to outliers' detection, it is necessary to analyze entire data to understand how much processing is required. This is done by plotting raw data. Figure 4.1 shows data with seasonality of 24 hours, so the plotting is done in that time interval. For easier analysis, the dataset was split into each day of the week (dotw) and the last plot corresponds to the whole dataset. Figure 4.1 is an example of a common dataset before pre-processing where outliers are clear.

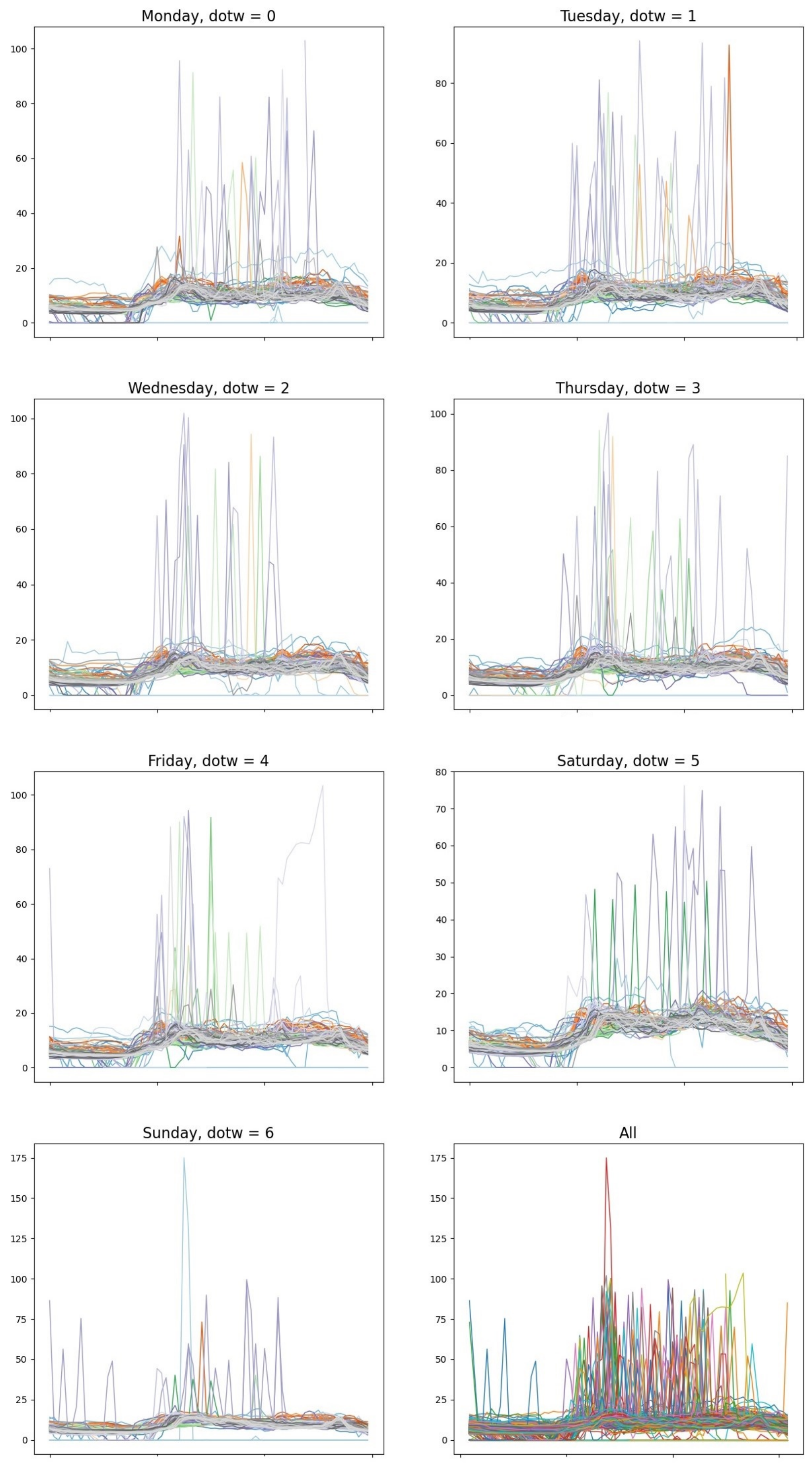


Figure 4.1: Raw time series plot for each day of the week (dotw) and for whole dataset of Espinheira

The first step of data cleaning is focused on removal of obvious outliers' points. This allows to establish more stricter threshold values in the next steps of data cleaning process.

Isolation Forest is an unsupervised method, which means that it works by itself to discover data features and/or data patterns. The main advantage of unsupervised methods it that they make possible to work with random variables and/or features. The main reason of choosing this method for this study is that it is very efficient in identification of very deviating points of whole dataset, and it is quite fast [27]. The model is based on decision trees, and it assumes that the process of isolating an anomaly point requires less partitions than the process of isolating a regular point as shows Figure 4.2. The model isolates every data point and calculates its anomaly score. Anomaly score translates the ease of isolating a point. For instances, if anomaly score is 0.5, it is a regular point, and, if anomaly score is 1, it is an anomaly point.

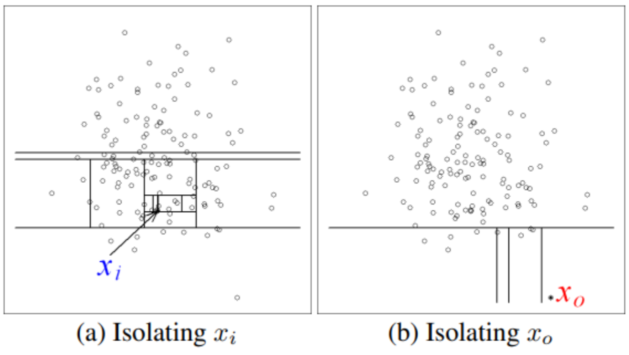


Figure 4.2: Schematic representation of isolation of points through partitions, (a) a regular point x_i requires twelve random partitions to be isolated. (b) an anomaly point x_0 requires four random partitions to be isolated [27]

For the next step it was used a very popular outlier detection method for seasonal and trendy data: Seasonal and Trend decomposition of Loess, commonly called STL decomposition. The objective of this method is to decompose a time series into three parts: seasonal, trend and residuals. The procedure consists of iterated cycle which, in turn, include two loops: the inner and outer loops. At each cycle, the inner loop applies seasonal and trend smoothing by this order and then, the outer loop estimates the irregular component, residuals, using seasonal and trend estimations from inner loop [28]. For this case study, the number of observations in each cycle of the seasonal component was set to number of points for a single day, which means the seasonality is daily. Since the decomposition is done, another amount of anomaly points can be identified. Thus, every residual point that falls out of established thresholds are set as outlier. The lower threshold is 2 times of standard deviation of residuals and the upper threshold is 3 times of standard deviation of residuals. Figure 4.3 shows a typical STL decomposition.

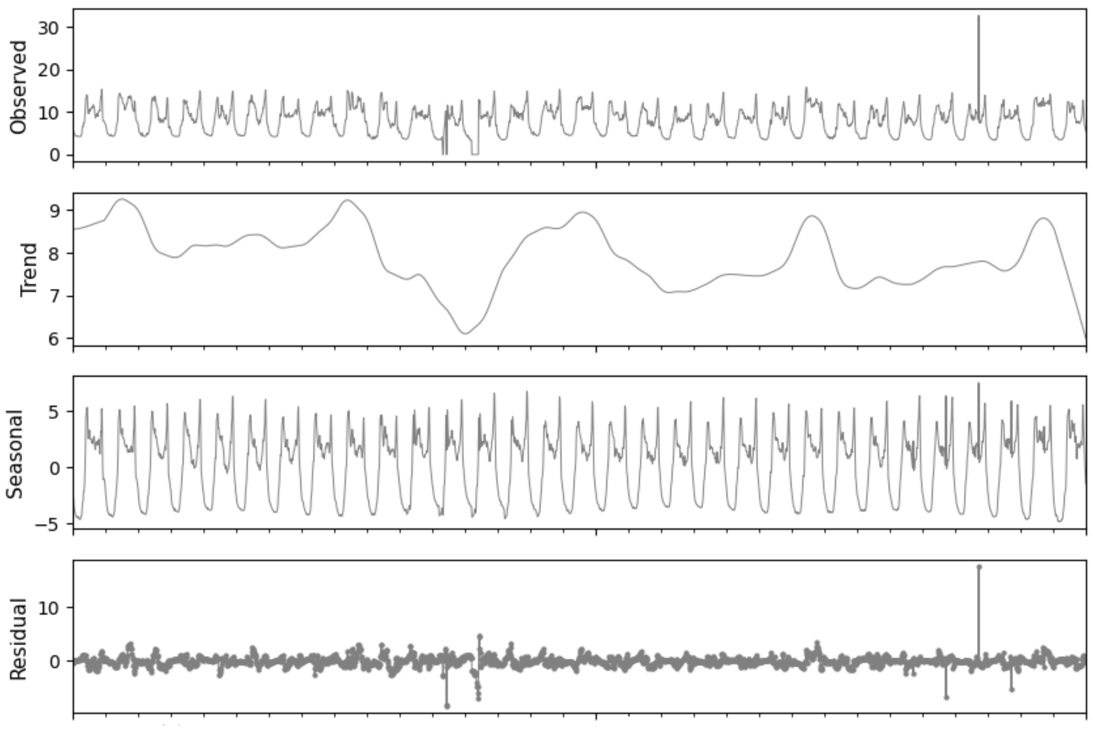


Figure 4.3: Example of time series STL decomposition

The last data cleaning step consists of identifying days where the percentage of missing data is greater than 10%. Therefore, if a day has at least 10% of missing data, the whole demand of that day is set to zero. This allows to fill missing values in rest of the days using linear and polynomial interpolation with no significant daily demand pattern changes.

After identifying the outliers, the number of missing values increases. At this point, it is possible to fill or simply discard the missing data sections. In the present study both approaches were applied. In small missing data sections, up to 7 consecutive missing datapoints (out of 72 datapoints per day), data was filled using linear interpolation. In the rest of the cases the data was discarded, which means it was not considered in training stages of the forecasting procedure. The reason for this approach is that the amount of available data is not large enough to apply a more robust solution, such as filling sections with predicted values. Figure 4.13 shows the final plot for pre-processed dataset.

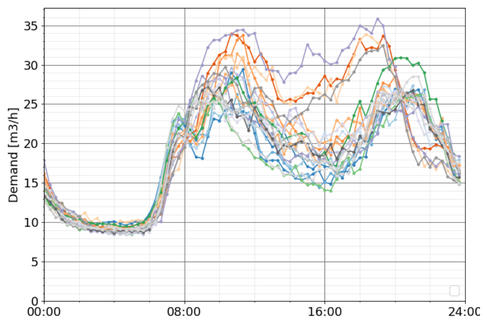
The complete code referring to the outliers can be found in the Appendix A. First, the variables referring to holydays and deviating days were defined, then a detailed breackdown of the time variable (in date, time, day of the week and other auxiliar variables). Outliers cleaning is the next step and it is commented throughout the code. At the end, the data is ready to be labeled by the type of the day (note "LABELING days of the week" and "LABELING DAY TYPES" code sections).

4.2 Data analysis

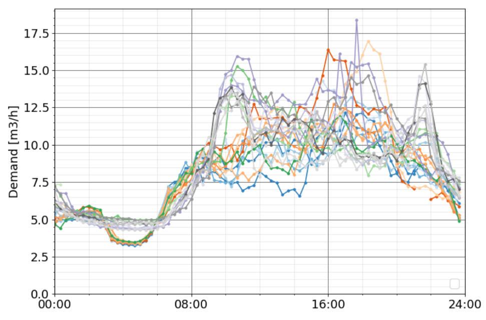
Data analysis is an important step in time series forecasting mainly when dataset shows pattern in very large scale. A study of patterns was done to understand the relationship between seasons and to make a more accurate interpretation of the accuracy of the models. It was made a patterns study suggested on [21] since Bakkers heuristic forecasting model was applied. In order to adapt the categorizations made in [21], the following types of the day was discern: seven types for each day of the week, four types for primary school holidays (Christmas holidays, Carnival holidays, Easter holidays and Summer holidays) and five types for individual deviating days (New Year's Day, Good Friday, day after Ascension Day, day after New Year's Day and Liberation Day).

In Figures 4.11, 4.12 and 4.13, the existence of patterns between the days of the week are clear for the demands of Albarqueira, Aveleira and Espinheira. However, this figures represents days of the week for the whole datasets, which means that other day types are included here, for example Christmas holidays. The next figures only show two datasets for the easier interpretation of the results. The reason for the number of the days being different for two datasets are related to the pre-processing step of dataset.

Figures 4.4, 4.5, 4.6 and 4.7 show the four types for primary school holidays. In Figure 4.4, for both cases, the lack of pattern consistency is notable, especially in 4.4a two patterns are visible (working days and Saturdays) although these datasets were set to the same type of day.



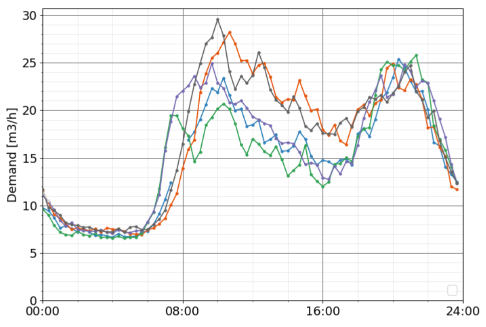
(a) Aveleira



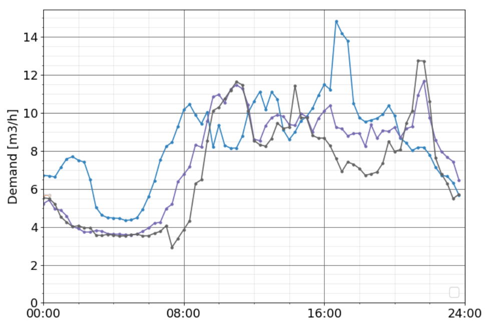
(b) Espinheira

Figure 4.4: Christmas holidays for two datasets

For Carnival and Easter holidays, the day patterns are almost the same as shows Figure 4.5 and Figure 4.6, respectively.

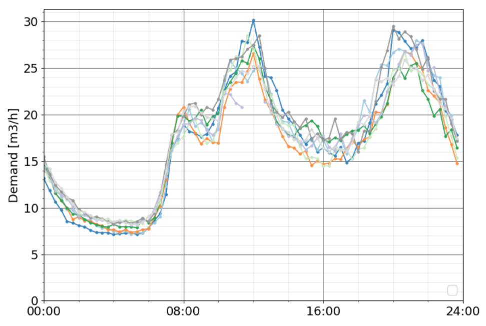


(a) Aveleira

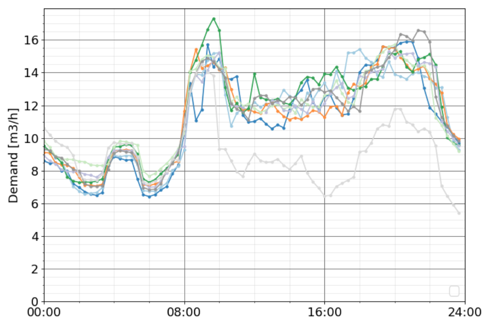


(b) Espinheira

Figure 4.5: Carnival holidays for two datasets



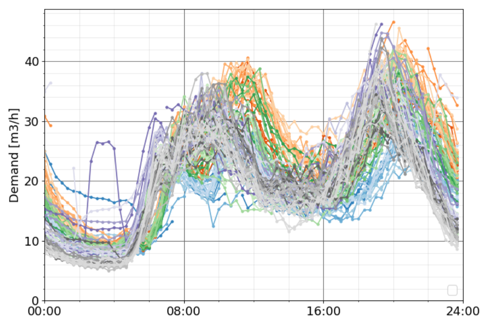
(a) Aveleira



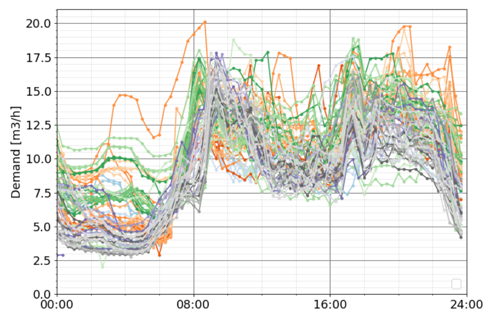
(b) Espinheira

Figure 4.6: Easter holidays for two datasets

The results of plotting summer holidays are shown in Figure 4.7. It is noted that in 4.7a, the patterns are more obvious than in 4.7b. These observations have recurrences in the forecasting accuracy for Summer holidays months.



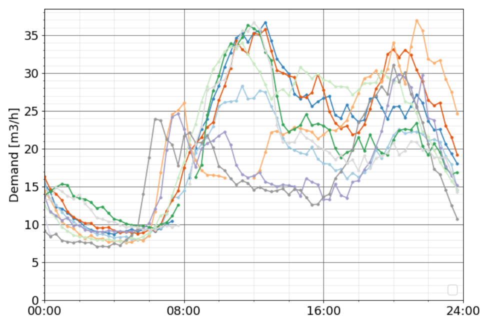
(a) Aveleira



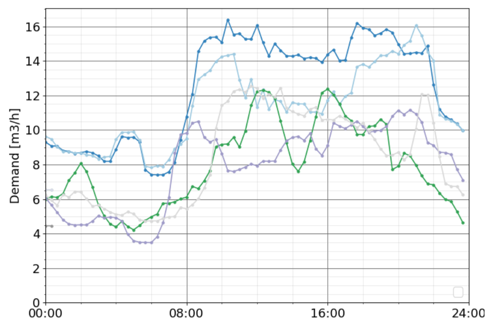
(b) Espinheira

Figure 4.7: Summer holidays for two datasets

The next type of days is Individual Deviating Days that have an unique labeling for each one. In Figure 4.8, in general, the demand curves have a different behaviour from each other, specially in 4.10b.



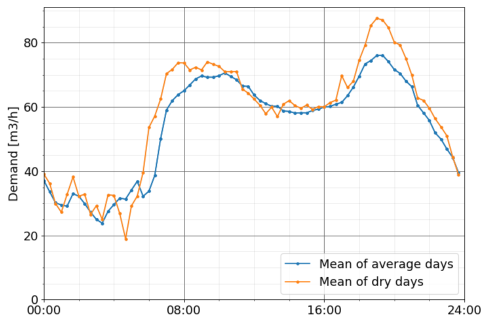
(a) Aveleira



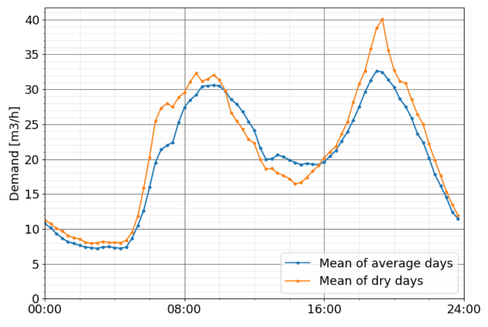
(b) Espinheira

Figure 4.8: Individual deviating days for two datasets

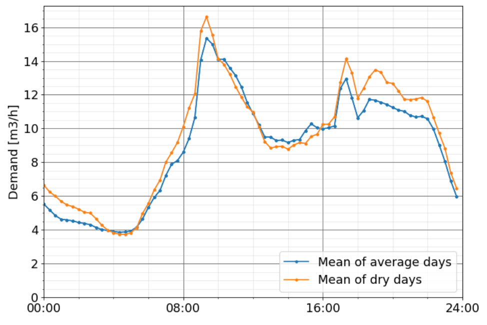
In [21], the last type of the days is related to weather conditions, which are not known in advance. It was made a study to understand the reality of water demand in dry days in comparison with mild weather days weather days. So, it was selected the months of July, August and September to plot the average of water demand in mild weather days along with average of water demand in dry days, represented in Figure 4.9. The dry days considered were those with more than 5 degrees Celsius above the moving average with period of 30 of the summer months. The choice of 5 degrees Celsius as threshold is related to selecting sunny days that presents average day temperature way higher than days before and after.



(a) Albarqueira



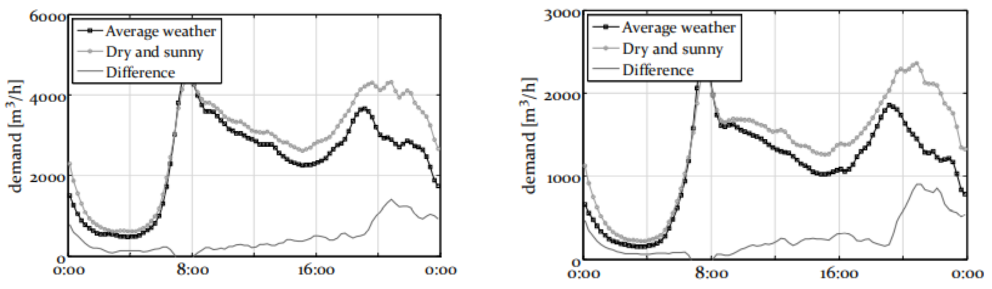
(b) Aveleira



(c) Espinheira

Figure 4.9: Average water demand pattern for mild and dry weather in summer months (Mean of average days refers to average of mild weather days)

The analysis of Figure 4.9 allows us to conclude that there are no significant pattern variations to justify the labeling dry days separately from average weather days, as it is done in [21] where the similar study justify this need. In Figure 4.10 the water demand is much higher in dry days, especially between 18:00h and 0:00h, so it makes sense to apply an extra demand forecasting parameter called sprinkle water demand forecast.



(a) Rhine area (b) Almere area

Figure 4.10: Deviating water demand pattern during mild and dry weather [21]

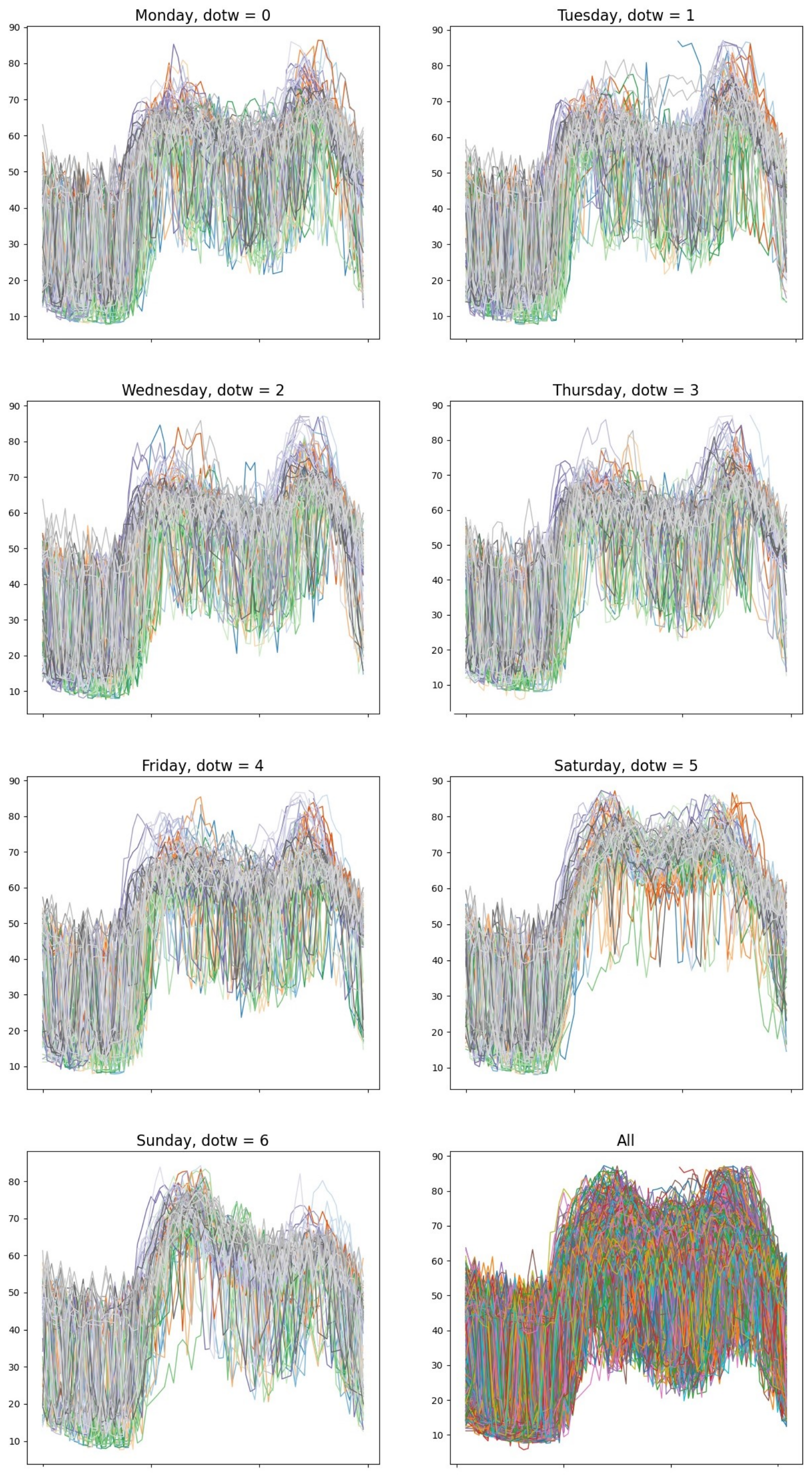


Figure 4.11: Pre-processed time series plot for each day of the week (dotw) and for whole data of Albarqueira

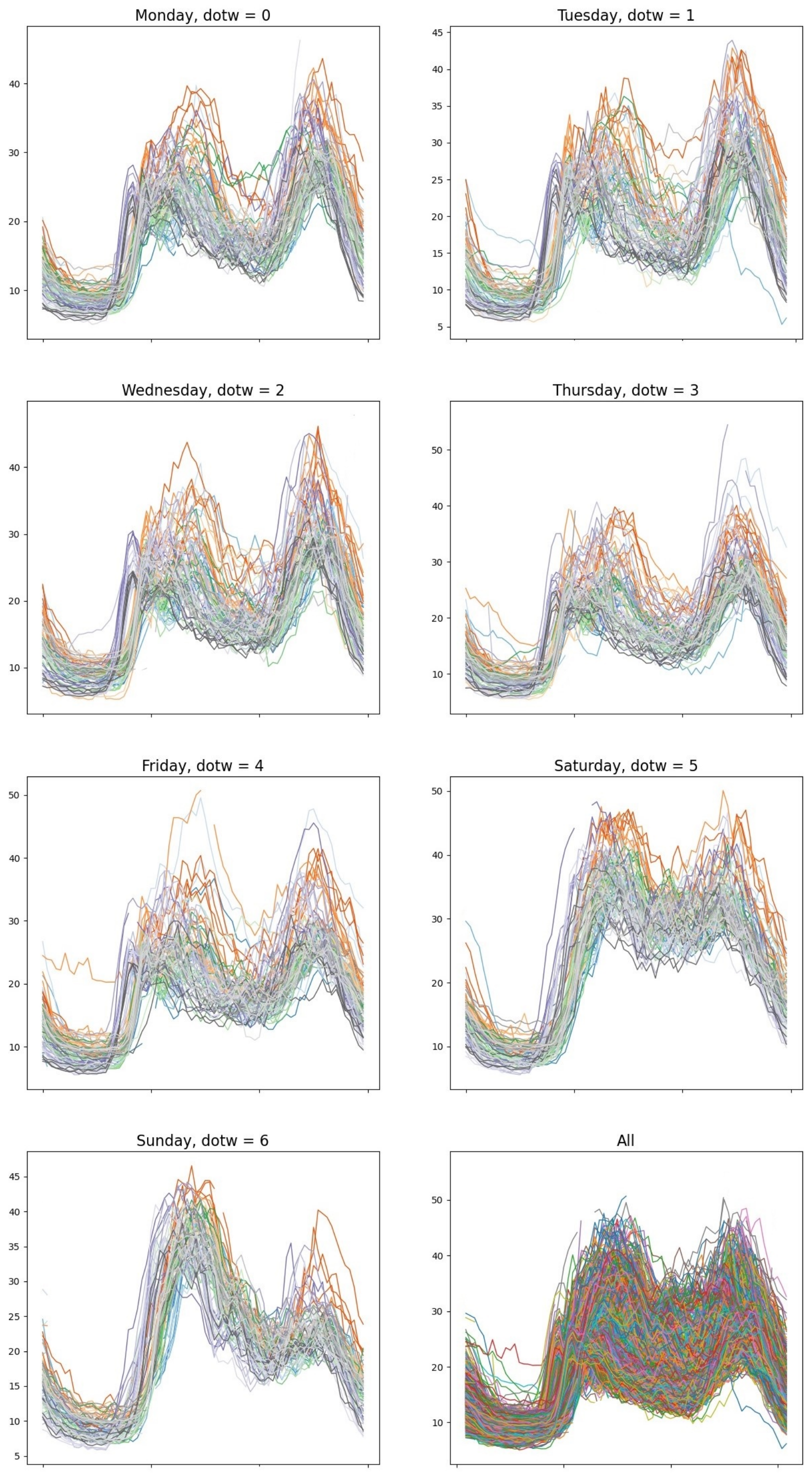


Figure 4.12: Pre-processed time series plot for each day of the week (dotw) and for whole data of Aveleira

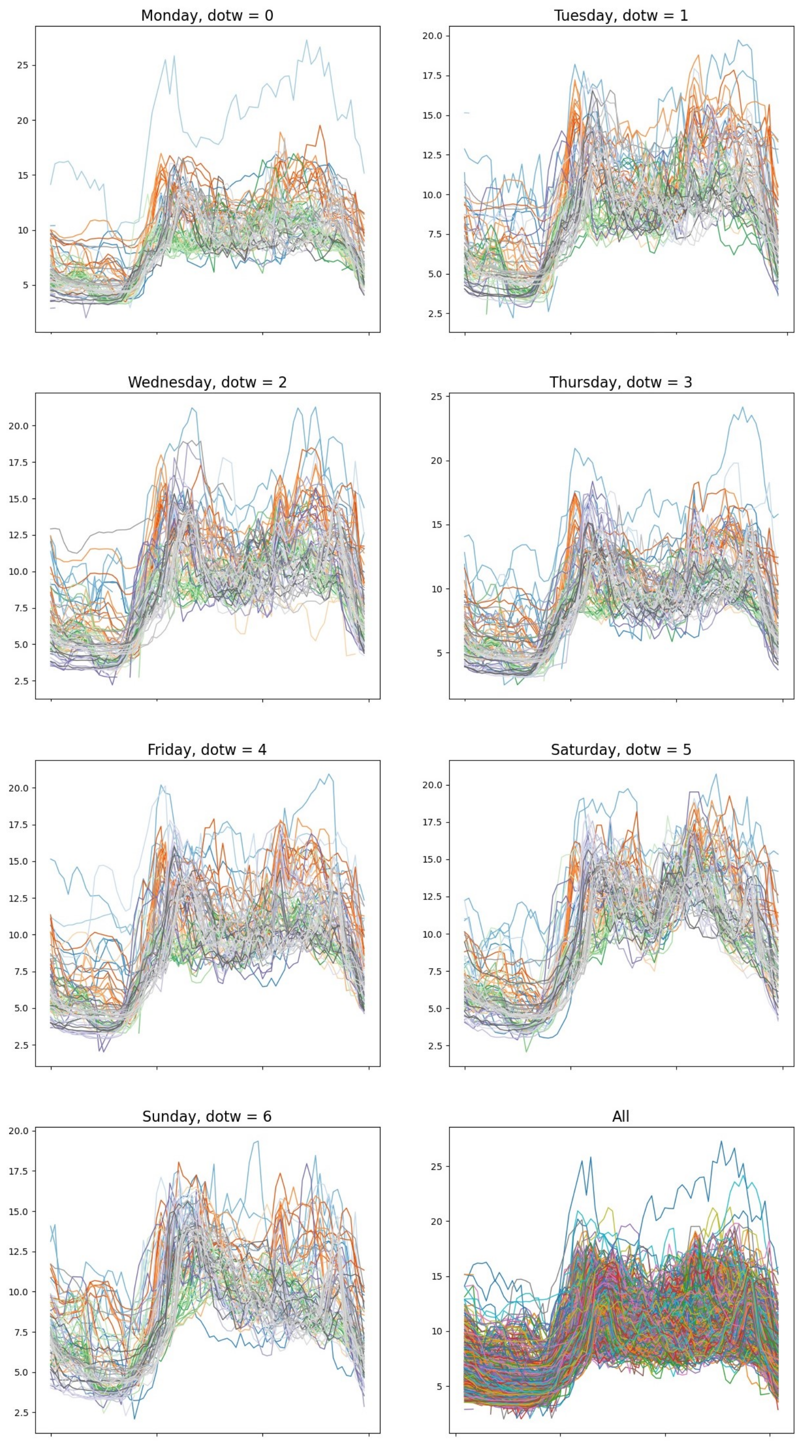


Figure 4.13: Pre-processed time series plot for each day of the week (dotw) and for whole data of Espinheira

4.3 Portuguese case study description

Water demand forecast has, as main input, the data of previous demand observations. When forecast methods are time-series based it is important to identifying patterns and understand its source for better data treatment, described in section of data pre-processing, and accurate results analysis.

The analysis is split in two parts: firstly it is presented general characteristics of studied areas and then, data analysis of each one.

Data used in the current study refers to three different portuguese places located in the center of Portugal. This area is characterized by a warm mediterranean climate and the average daily temperatures are 10.3 ºC and 21.0 ºC for winter and summer months, respectively. These locations are small villages and they are about 5 kilometers apart from each other.The localization is presented in figure 4.14. The used data refer to two years of consumption, from march of 2018 to march of 2020, with 10min time steps, resulting in 105120 values for each case study. The data has several outliers and chunks of missing datapoints which was treated in Section 4.1. Table 4.1 describe the average water demand consumption per day for each area. It is observed that there is a water consumption relationship between three datasets: Albarqueira and Aveleira have a factor 5.3 and 2.1 larger average water consumption of Espinheira, respectively. Note that the number of inhabitants or area itself are not related to the amount of water consumption of the location since this information is not known.

Table 4.1: Average demand per hour for each area

Case study	Average demand [m^3/day]	Relative factor
Albarqueira	7054.79	5.3
Aveleira	2825.02	2.1
Espinheira	1323.86	1

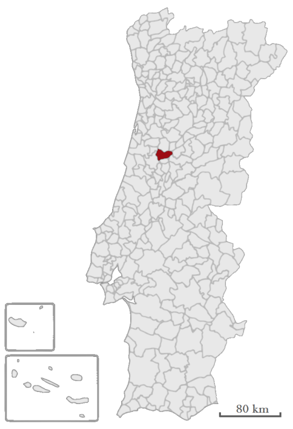


Figure 4.14: Location of the investigated areas in Portugal

Part III

Results, discussion and conclusions

Chapter 5

Results

5.1 Accuracy of the ARIMA

The results obtained for each method and time interval are presented in this section and its tables. For ARIMA the forecasting, 3 months of data were considered for training step and, from then on, a one week water demand forecast was made. The analysis of data allowed to conclude the best ARIMA model fit is $(1, 0, 1)(1, 1, 0, 48)$, where 48 is related do daily seasonality (according to reshaping the data to 30 minutes time step). Table 5.1 describes accuracy for 24 hours of ARIMA forecasting and it shows that the MAPE error varies between approximately 4% and 16%. The area with the highest water demand, Albarqueira, has the smallest error of 4.81% and Aveleira the highest error of 15.56%. Table 5.2 lists the results for 30 minutes time step accuracy for three datasets and, overall it is much higher than in table 5.1 ranging between 18% and 23%.

Table 5.1: Performance of ARIMA model per 24 hours for 1 week of forecasting

Area	MAE [m^3]	MAPE [%]
Albarqueira	115.64	4.81
Aveleira	129.79	15.63
Espinheira	38.49	9.92

Table 5.2: Performance of ARIMA model per 30 minutes for 1 week of forecasting

Area	MAE [m^3]	MAPE [%]
Albarqueira	10	20.22
Aveleira	3.9	22.83
Espinheira	1.46	18.26

Figure 5.1, 5.2 and 5.3 show graphic results for one week forecasting using ARIMA model for Albarqueira, Aveleira and Espinheira, respectively. Forecasting has daily seasonal pattern and, for each following day, the forecasting pattern remains the same and it causes low accuracies described above. For Albarqueira scenario, 30 minutes time steps forecasting shows a large deviant result from original data (20 %). However, for 24 hours accuracy, the result is much lower (around 5 %) because 24 hours accuracy is

calculated by averaging the 30 minutes forecasting values. For Aveleira it is seen that both 30 minutes and 24 hours forecasting values are very deviant (16% and 23%). And, for Espinheira, forecast and original demands have less devaint values out of three cases (10% and 18%) and seem to have more accurate pattern to the original demand. Note that, in this context, original demand refers to measured demand.

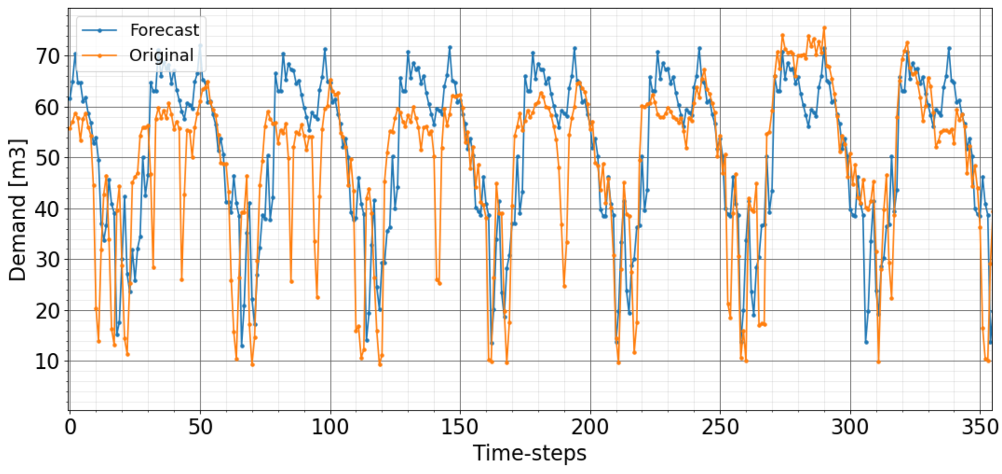


Figure 5.1: Results for one week of forecasting using ARIMA model for Albarqueira

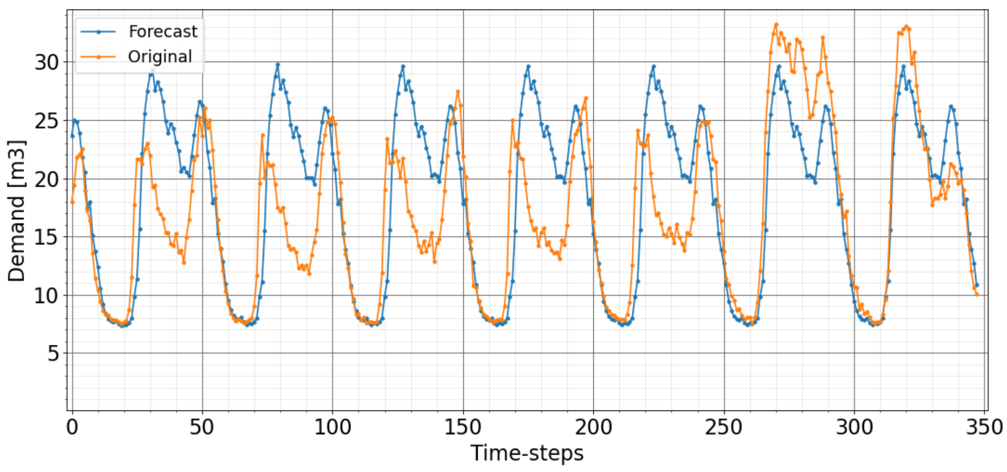


Figure 5.2: Results for one week of forecasting using ARIMA model for Aveleira

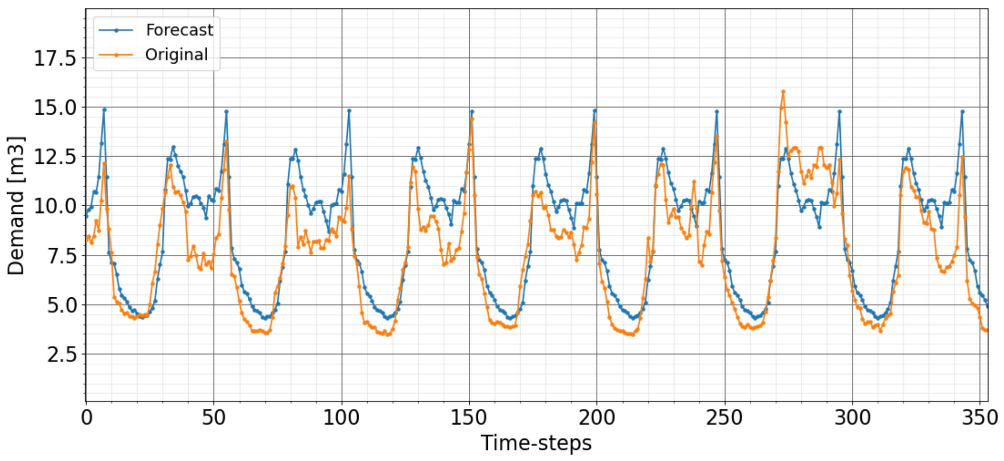


Figure 5.3: Results for one week of forecasting using ARIMA model for Espinheira

5.2 Accuracy of the Bakkers heuristic model

For heuristic model, since the data has 2 years of water demand, datasets were split equally in two datasets: training and testing. Although this approach is not the common one, it was necessary to designate, at least, one year of dataset for training purposes and the rest for testing purposes because there are days labeled as unique. An unique day means that water demand pattern is specific to himself and occurs once a year. The results of accuracy are shown in Tables 5.3 and 5.4. These tables are related to the first 24 hours forecast despite bakkers' heuristic model calculates forecast for the next 48 hours. In general, the accuracy for 24 hours forecasting is much better than for 20 minutes step forecasting and the reason for that is that the daily water demand consumption is less volatile. The analysis of Table 5.3 shows that MAPE is lower for Albarqueira, which in the area where the average water demand is the highest. In the other hand, analysis of Table 5.4 shows that, for 20 minutes step forecast, Albarqueira has the biggest MAPE: 26.42%.

Table 5.3: Performance of the Bakkers' heuristic model per 24 hours for 1 year of forecasting

Area	MAE [m^3]	MAPE [%]
Albarqueira	200.32	5.57
Aveleira	110.20	8.07
Espinheira	39.49	6.29

Figures 5.4, 5.5 and 5.6 show graphic results for all case-studies. It is visible the adaption of forecasting model to the type of the day and the graphic results match the results presented in Tables 5.3 and 5.4.

Table 5.4: Performance of the Bakkers' heuristic model per 20 minutes for 1 year of forecasting

Area	MAE [m^3]	MAPE [%]
Albarqueira	9.32	26.42
Aveleira	2.33	12.58
Espinheira	1.12	14.00

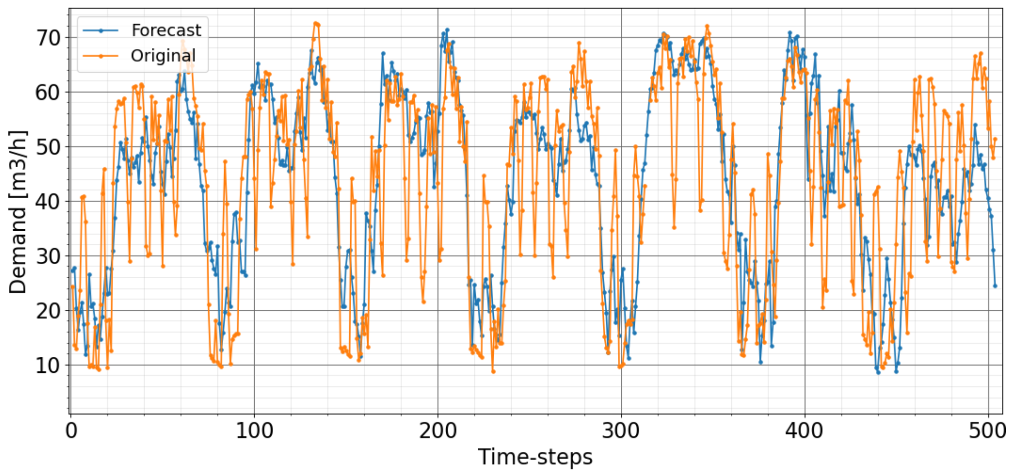


Figure 5.4: Results for one week of forecasting using bakkers' heuristic model for Albarqueira

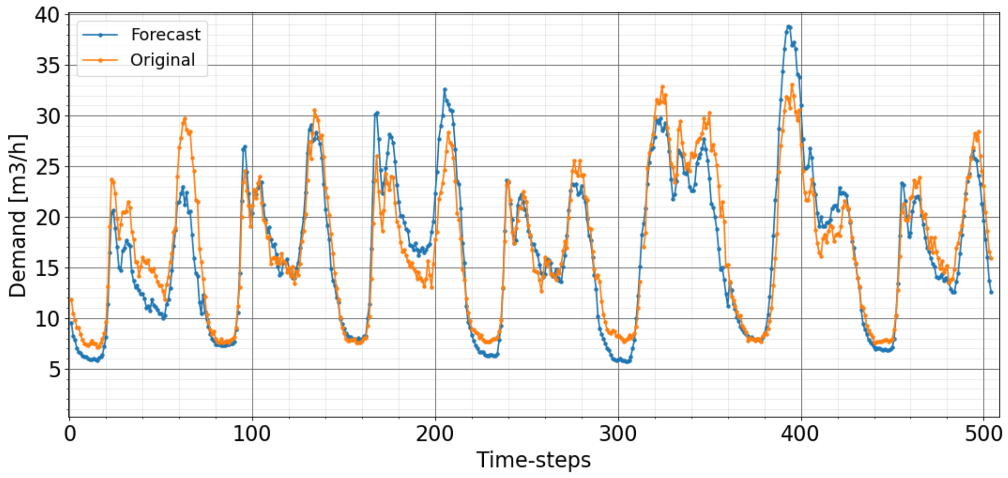


Figure 5.5: Results for one week of forecasting using bakkers' heuristic model for Aveleira

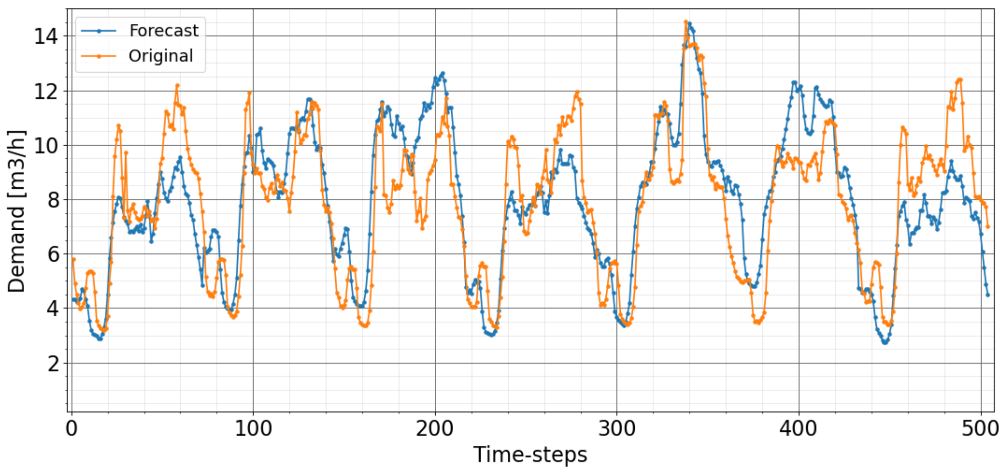


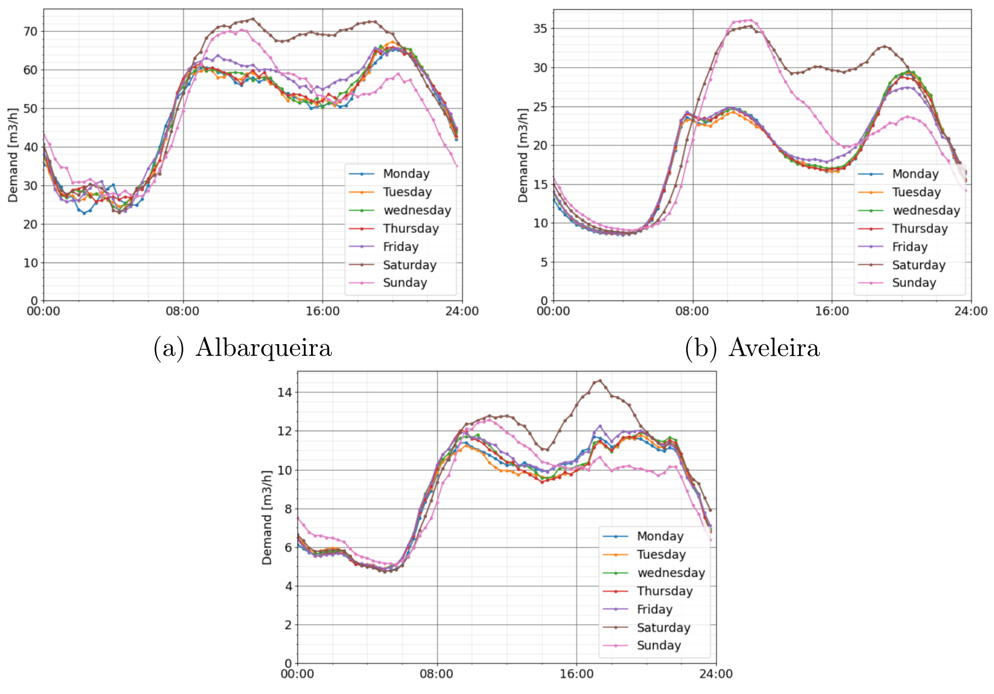
Figure 5.6: Results for one week of forecasting using bakkers' heuristic model for Espinheira

5.3 Results discussion

This section discusses the accuracy of the results. Overall, the accuracy of 24 hours forecasting is significantly better than accuracy of smaller time steps forecasting. However, there is not a clear linear relationship between both accuracies. It is observed that the location with higher water consumes (Albarqueira) has better accuracies for both ARIMA and heuristic bakkers' forecasting models for 24 hours forecast. For ARIMA scenario, in 24 hours forecast, the location Aveleira has around 3 times larger MAPE error (15.63%) than Albarqueira (4.81%) and Espinheira has almost 2 times larger MAPE (9.92%) error than Albarqueira, despite Aveleira has an average water consumption between these two locations.

Accuracy result for Aveleira seems to be different from an expected result but it can be explained by Aveleira having a very divergent water consumption pattern between the types of days. Figure 5.7 shows the average water demand pattern for whole dataset for each type of the day, in this case, each day of the week. For example, the water consumption pattern on Saturday and Sunday is plainly different from the consumption pattern of the rest of weekdays, as shown in Figure 5.7b.

Albarqueira has an interesting precision error of 4.18% because it presents a consistent daily water demand average over time. For 30 minutes forecast, the accuracy errors vary between 18% and 23%. Albarqueira is the location with the highest error, as expected. The reason for a poor accuracy is a peculiarity of the data that have peaks and troughs throughout the daily consumption history (visible in Figure 5.4, 5.1 and 5.4), it had a high volatility through the day. Espinheira is the location with lowest error. The reason for this is not clear since, from data analysis (Figure 4.13), the patterns are not consistent; despite this, the possible reason might be related to the choice of random data chunk for evaluation which benefited the accuracy in this regard.



(a) Albarqueira (b) Aveleira

(c) Espinheira

Figure 5.7: Average pattern of water demand for each day of the week

For bakkers' heuristic method scenario, in 24 hours forecast, the accuracy error results are ranged between 5% and 8% and these seem to have no linear relation with the average water demand of the listed locations, however this conclusion must be confirmed with a greater number of locations. For 20 minutes forecast, the MAPE error is higher for Albarqueira with the value of 26.42% which is an expected result due to the particularity of the Albarqueira data (Figure 4.11), as mentioned in the analysis of the ARIMA results. Aveleira and Espinheira locations have 12.58% and 14.00% MAPE errors, respectively. The reason Aveleira's MAPE is quite better is that the datapoint from this location are consistent and smooth through all dataset (Figure 4.12).

Figure 5.8 and 5.9 represent trend of the MAPE for 24 hour time step forecast and 20 minutes time step forecast over 11 months of analysis, respectively. The missing month has been discarded because of high lack of data. The first graph shows that Albarqueira has highest MAPE in July, around 8%, and lowest MAPE in October, around 4% and it is visible a decreasing error trend. For Aveleira and Espinheira the month of December stands out due to the sudden rise of the error that may be related to the labeling of the data as different type of the day in whole Christmas holydays (see Figure 4.4).

The second graph (Figure 5.9) shows that for summer months (June, July and August) Albarqueira present lower MAPE. It is noteworthy that Albarqueira MAPE is almost double of the MAPE value for Aveleira, which is almost the same relationship that is visible in Table 4.1 represented as relative factor of average water demand per day between this both areas. Aveleira and Espinheira have both the maximum MAPE value in May that remains relatively constant throughout the year for Aveleira and decreases throughout the year for Espinheira.

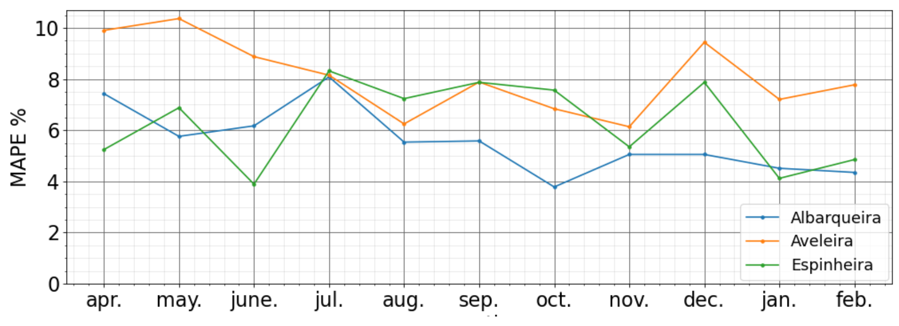


Figure 5.8: Trend of MAPE for the 24 hours step forecasting over 11 months of analysis

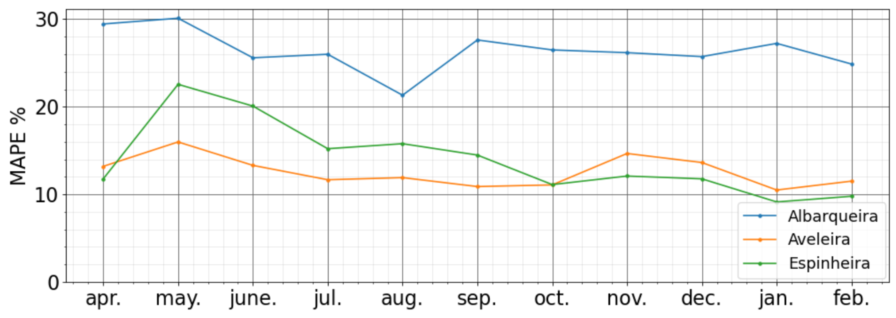


Figure 5.9: Trend of MAPE for the 20 minutes step forecasting over 11 months of analysis

Figues 5.10 and 5.11 show results for one week of forecasting in Christmas holidays and there is clear that days 24 and 25 of December are not properly forecasted for both cases. Although, it is visible that overall, the fit of the model is not appropriate, specially in Figure 5.10 .

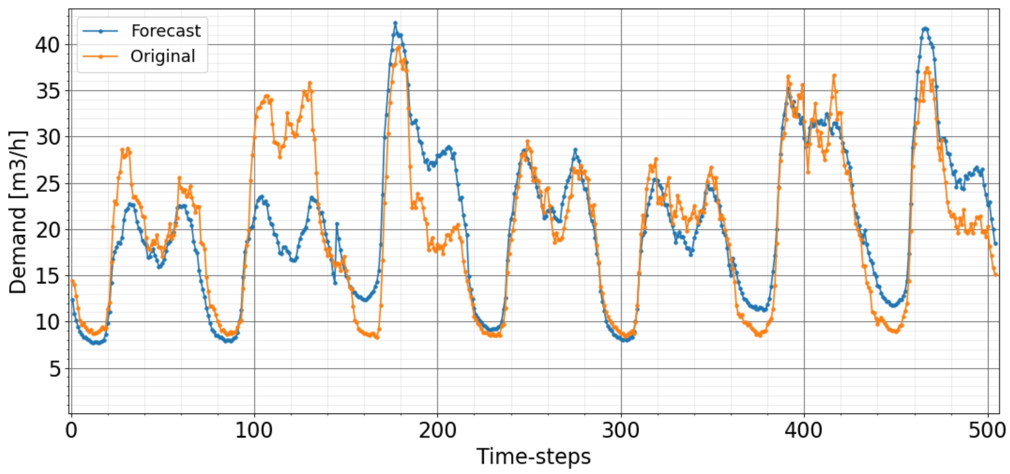


Figure 5.10: Results for one week of forecasting, from 23 and 29 December, using bakkers' heuristic model for Aveleira

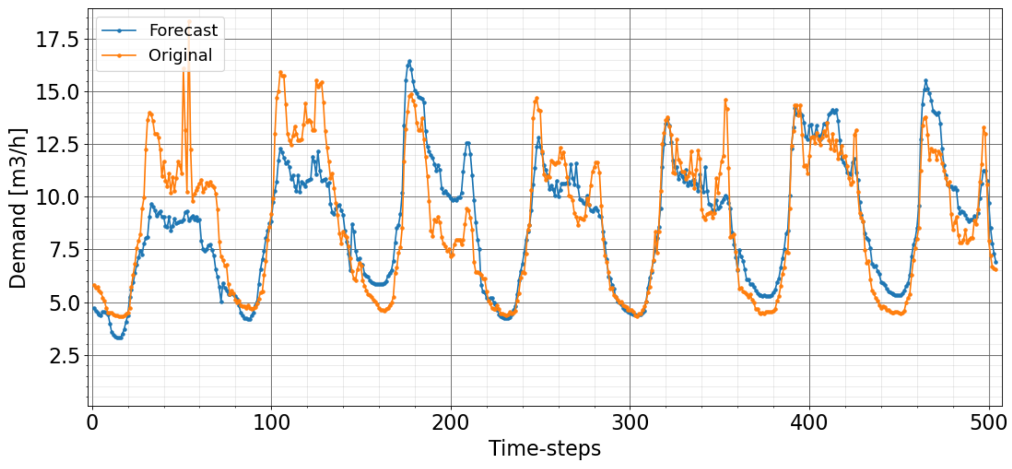


Figure 5.11: Results for one week of forecasting, from 23 and 29 december, using bakkers' heuristic model for Espinheira

Chapter 6

Conclusion and final remarks

Through the literature review and the work developed in this thesis it is possible to conclude that the domain of forecasting algorithms provides enriching case studies. The purpose is not to run case studies randomly but it is to draw conclusions about each one to contribute to the improvement of the field of forecasting time series algorithms. The case study used refer to city Penacova, located in central region of Portugal. The main conclusions of this work are:

1. the amount of data history influences the quality of data labeling;
2. forecasting water demand with small time step interval only makes sense if this information adds practical value to the water distribution process;
3. in the found related works, the pre-processing step is hardly ever explicit despite the importance it represent in accuracy of the forecasting models;
4. it is necessary to test more similar case studies to draw more robust conclusions about the effectiveness of the forecasting models;
5. the heuristic model presents accuracy results that should be taken into account for further related work;
6. it is possible to improve the accuracy of the results if the analysis of the data is done in greater detail (more data is needed);
7. forecasts based exclusively in the variable time are highly valuable because of the independence of variables that are difficult to obtain;
8. for better analysis, the Nash–Sutcliffe model efficiency coefficient, also called R^2 factor, would be very valuable for a more detailed model analysis, as its robustness.

The objectives of this work were partially achieved. Three forecast models were proposed but ARIMA and Bakkers' heuristic model were carried through to the end, leaving GMDH model out.

The obtained results, especially for heuristic model, have interesting values leaving room for improvements. It is suggested to test heuristic model with more data and consider making a different classification of the type of the days from those suggested by the author.

For ARIMA model, the results are not that interesting. The reason for that is because the model itself does not allow to detail the seasonality of the data. For that pupose it is suggested to use a more reigning version of the ARIMA model: Exponential smoothing state space model with Box-Cox transformation, ARMA errors, Trend and Seasonal components, also called TBATS model. This model allows to include seasonality at various scales, for example, daily, weekly and even yearly seasonality.

Thus, both models are not comparable side by side despite this goal. ARIMA was tested on a randomly chosen one-week forecast meanwhile heuristic was tested for a whole year. It would be possible to make the forecast for a whole year using ARIMA model in rolling window scenario for better accuracy results, where the model is fitted with new parameters every step of rolling window. Although, this solution is not viable at all due to the computational resources consumption. Bakkers' heuristic turned out to be a more efficient model because it provides for the updating of parameters throughout the forecast cycle, having the rolling windows approach and it does not need a lot of data to estimate starting parameters.

The main contribution of this work stands out in the testing an efficient and simple to implement heuristic model that has not been tested in a mediterranean climate and in low water consumption areas such as Albarqueira, Aveleira and Espinheira.

Bibliography

[1] Swamee, Prabhata K., and Ashok K. Sharma. (2008) *Design of water supply pipe networks.* John Wiley and Sons.

[2] Makridakis, S., Spiliotis, E., Assimakopoulos, V. (2018) Statistical and Machine Learning forecasting methods: Concerns and ways forward *PLOS ONE* 13.3.

[3] Donkor, E.A., Mazzuchi, T.A., Soyer, R. and Roberson, J.A (2014) Urban water demand forecasting: review of methods and models, *Journal of Water Resources Planning and Management*, Vol. 140, Issue 2

[4] Ghiassi, M., Zimbra, D. K., and Saidane, H. (2008) Urban water demand forecasting with a dynamic artificial neural network model. *Journal of Water Resources Planning and Management* 134.2:138-146.

[5] Kofinas, D. , Mellios, N. , Papageorgiou, E. and Laspidou, C. (2014) Urban Water Demand Forecasting for the Island of Skiathos. *Procedia Engineering*, 89: 1023-1030.

[6] Karamaziotis, P. I., Raptis, A., Nikolopoulos, K., Litsiou, K. and Assimakopoulos, V. (2020) An empirical investigation of water consumption forecasting methods. *International Journal of Forecasting*, Elsevier, 36.2: 588-606.

[7] Coelho, B., and Andrade-Campos, A. G. (2019) Short-term forecasting of hourly water demands-a Portuguese case study. *International Journal of Water* 13.2:173-207.

[8] Herrera, M., Torgo, L., Izquierdo, J. and Pérez-García, R. (2010) Predictive models for forecasting hourly urban water demand. *Journal of Hydrology* 387.1-2: 141-150.

[9] Candelieri, A., Soldi, D. , and Archetti, F. (2015) Short-term forecasting of hourly water consumption by using automatic metering readers data. *Procedia Engineering* 119.1: 844-853.

[10] Pulido-Calvo, I., Roldán, J., López-Luque, R., and Gutiérrez-Estrada, J. C. (2003) Demand forecasting for irrigation water distribution systems. *Journal of Irrigation and Drainage Engineering* 129.6: 422-431.

[11] Tiwari, M. K., and Adamowski, J. (2013) Urban water demand forecasting and uncertainty assessment using ensemble wavelet-bootstrap-neural network models. *Water Resources Research* 49.10: 6486-6507.

[12] Wang, X., Sun, Y., Song, L. and Mei, C. (2009) An eco-environmental water demand based model for optimising water resources using hybrid genetic simulated annealing algorithms. Part I. Model development. *Journal of environmental management* 90.8: 2628-2635.

[13] Wang, Y., Ocampo-Martínez, C., Puig, V. and Quevedo, J. (2014) Gaussian-process-based demand forecasting for predictive control of drinking water networks. *International Conference on Critical Information Infrastructures Security.* Springer, Cham.

[14] Jain, A., Kumar Varshney, A. and Chandra Joshi, U. (2001) Short-Term Water Demand Forecast Modelling at IIT Kanpur Using Artificial Neural Networks. *Water Resources Management* 15: 299–321.

[15] Bougadis, J., Adamowski, K., and Diduch, R. (2005) Short-term municipal water demand forecasting. *Hydrological Processes: An International Journal* 19.1:137-148.

[16] Pacchin, E., Alvisi, S. and Franchini, M. (2017) A short-term water demand forecasting model using a moving window on previously observed data. *Water* 9.3: 172.

[17] Pacchin, E., Gagliardi, F., Alvisi, S. and Franchini, M. (2019) A Comparison of Short-Term Water Demand Forecasting Models. *Water Resources Management.* 33.4: 1481-1497.

[18] Abdel-Aal, R. E. (2004) Short-term hourly load forecasting using abductive networks. *IEEE Transactions on Power Systems* 19.1: 164-173.

[19] Abdel-Aal, R. E., Elhadidy, M. A. and Shaahid, S. M. (2009) Modeling and forecasting the mean hourly wind speed time series using GMDH-based abductive networks. *Renewable Energy* 34.7: 1686-1699.

[20] National Institute for Strategic Studies, International Center for Information Technologies and Systems of the National Academy of Sciences of Ukraine (2021). *GMDH.* https://www.gmdh.net

[21] Bakker, M. (2014) *Optimised Control and Pipe Burst Detection by Water Demand Forecasting*, PhD thesis, TU Delft, Delft University of Technology, Delft, Netherlands.

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

[23] Coelho, B. (2016) *Energy Efficiency of Water Supply Systems Using Optimisation Techniques and Micro-Hydroturbines*, PhD thesis, Mechanical Engineering Department, University of Aveiro, Portugal.

[24] Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M. Z., Barrow, D. K., Bergmeir, C., ... and Ziel, F. (2020) *Forecasting: theory and practice.*

[25] Montgomery, D. C., Jennings, C. L., and Kulahci, M. (2008) *Introduction to time series analysis and forecasting.* John Wiley and Sons.

[26] Chen, C. and Liu, L. M. (1993) Forecasting time series with outliers. *Journal of Forecasting* 12.1: 13–35.

[27] Liu, F. T., Ting, K. M., and Zhou, Z. H. (2008) Isolation forest. *2008 eighth ieee international conference on data mining.* 413-422.

[28] Theodosiou, M. (2011) Forecasting monthly and quarterly time series using STL decomposition. *International Journal of Forecasting.* 24.7:1178-1195.

[29] Bakker, M., Vreeburg, J., Van Schagen, K. and Rietveld, L. (2013). A fully adaptive forecasting model for short-term drinking water demand. *Environmental Modelling and Software*, 48, 141–151.

[30] Bakker, M., Van Duist, H., Van Schagen, K., Vreeburg, J. and Rietveld, L. (2014). Improving the performance of water demand forecasting models by using weather input. *Procedia Engineering*, 70, 93–102.

Appendix A

Pre-processing code

```
import pandas as pd
import numpy as np
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.ensemble import IsolationForest
from dateutil import parser


#Time constats
one_day_hours = 24                         # amount of hours in one day
hour_minuts = 60                           # amount of minuts in one hour
week_days = 7                              # amout of type days in one week
hour_steps = 10                            # original time step
week_time_interval_original = int((hour_steps/hour_steps)
                                * one_day_hours * week_days)

# list of timezones 2018 and 2019
tz_mar_18 = '2018-03-25'
tz_out_18 = '2018-10-28'
tz_mar_19 = '2018-03-31'
tz_out_19 = '2018-10-27'

# national holidays 2018
jan = '2019-01-01'                 # 2019
abr0 = '2018-04-01'
abr1 = '2018-04-25'
mai0 = '2018-05-01'
mai1 = '2018-05-31'
jun = '2018-06-10'
ago = '2018-08-15'
out = '2018-10-05'
nov = '2018-11-01'
dez0 = '2018-12-01'
dez1 = '2018-12-08'
dez2 ='2018-12-25'

# national holidays 2019
jan_1 = '2020-01-01'               # 2020
abr0_1 = '2019-04-19'              # sexta feira santa
abr1_1 = '2019-04-21'              # Pascoa
abr2_1 = '2019-04-25'              # dia da liberdade
mai0_1 = '2019-05-01'              # dia do trabalhador
jun0_1 = '2019-06-10'              # dia de portugal
jun1_1 = '2019-06-20'              # corpo de deus
ago_1 = '2019-08-15'               # assuncao da n. senhora
out_1 = '2019-10-05'               # implantacao da republica
nov_1 = '2019-11-01'               # dia de todos os santos
dez0_1 = '2019-12-01'              # restauracao da independencia
dez1_1 = '2019-12-08'              # dia da imaculada conceicao
dez2_1 ='2019-12-25'               # natal

# individual deviating days 2018
new_year = '2019-01-01'            # ano novo
good_friday = '2018-03-30'         # sexta feira santa
asc_day_after = '2018-04-02'       # dia seguinte do feriado municipal
libertation_day = '2018-04-25'     # dia da liberdade
asc_day_after1 = '2018-06-25'      # dia do feriado municipal

# individual deviating days 2019
new_year_1 = '2020-01-01'
good_friday_1 = '2019-04-19'
asc_day_after_1 = '2019-04-23'
libertation_day_1 = '2019-04-25'
asc_day_after1_1 = '2019-06-25'

'Function␣for␣data␣treatment'
```

```
def data_treatment(loc):
    # feriados municipais das regioes
    if loc == 'Aveleira':
        feriado = '2018-06-24'
        feriado1 = '2019-06-24'
        lim_min = 5
        interpolate_lim = 4
        i_method='polynomial'
    elif loc == 'Albarqueira':
        feriado = '2018-04-07'
        feriado1 = '2019-04-07'
        lim_min = 0.6
        interpolate_lim = 4
        i_method='linear'
    elif loc == 'Espinheira':
        feriado = '2018-07-17'
        feriado1 = '2019-07-17'
        lim_min = 2
        interpolate_lim = 4
        i_method='linear'

    dir='C:/Users/AlinaLysenko/Documents/Universidade/Tese␣de␣mestrado/Data/Demand␣Scubic'
    index_date = pd.date_range(start = '16/3/2018␣17:50', end = '16/03/2020␣17:50',
                freq = '10min')

    def parser(x):
        return pd.datetime.strptime(x,"%Y-%m-%dT%H:%M:%S%z")

    series = pd.read_csv(dir + '/PE_' + loc + '.csv',
                header=0, parse_dates=[0], index_col=None, sep=";",
                usecols = ['Time', loc + '␣(PE)'],
                squeeze=True, date_parser=parser)

    df_date0 = series.set_index(pd.to_datetime(series['Time'],utc=True))

    "␣Resampling␣from␣10␣minutes␣to␣20␣minutes␣time-step"
    xdd = df_date0.resample("20T").mean()
    df_date0 = xdd

    "␣Add␣relevand␣columns"
    df_date0_aux = (df_date0.index).strftime("%Y-%m-%d␣%H:%M:%S")
    df_date0['Date␣and␣Time'] = [x[0:16] for x in df_date0_aux]
    df_date0['Date'] = [x[0:10] for x in
                        (df_date0.index).strftime("%Y-%m-%d␣%H:%M:%S")]
    df_date0['Time'] = [x[11:16] for x in
                        (df_date0.index).strftime("%Y-%m-%d␣%H:%M:%S")]
    df_date0['Date'] = (pd.to_datetime(df_date0['Date'], format='%Y-%m-%d'))
    df_date0['Date␣and␣Time'] = (pd.to_datetime(df_date0['Date␣and␣Time']))
    df_date0['valid_days'] = 1
    df_date0['Day␣of␣the␣week'] = (df_date0['Date']).dt.day_name()
    df_date0['dow'] = (df_date0['Date']).dt.dayofweek
    df_date0['Type'] = df_date0['dow']

    df_date0.rename(columns={loc + '␣(PE)':'Demand'}, inplace = True)
    df_date0.index.names = ['idx']

    "␣Drop␣firsts␣and␣lasts␣rows␣"
    df_date0 = df_date0.drop(df_date0.index[0:19])
    df_date0.drop(df_date0.tail(54).index, inplace=True)

    "␣Drop␣rows␣where␣timezome␣exist"
    for tz_idx in [tz_mar_18, tz_out_18, tz_mar_19, tz_out_19]:
        tz_var = df_date0[ df_date0['Date'] == tz_idx].index
        df_date0.drop(tz_var , inplace=True)

    """ OUTLIERS_____________________________________________________
        order:
                1st - isolation forest to clear more obvious outliers
                2nd - clear all data below very low value (0.6 or 5)
                3rd - STL decomposition
                4th - clear data where  available data is less than 90%
                5th - set all zeros to NaN
    """
    # (1)-----------------------------------------------------------------
    df_date0 = df_date0.fillna(0)
    model=IsolationForest( contamination=float(0.005), n_estimators=10)
    x=(df_date0['Demand'].values)
    xx=pd.DataFrame(x)
    model.fit(df_date0[['Demand']])
    df_date0['scores']=model.decision_function(df_date0[['Demand']])
    df_date0['anomaly']=model.predict(df_date0[['Demand']])
    df_date0.loc[df_date0['anomaly']==-1, 'Demand'] = np.nan

    # (2)-----------------------------------------------------------------
    df_date0.loc[df_date0['Demand']<lim_min, 'Demand'] = np.nan

    # (3)-----------------------------------------------------------------
    df_date0 = df_date0.fillna(0) # bc SDT doesnt work well with NaN values
    result = seasonal_decompose(df_date0['Demand'], model='additive',period=72)

    seasonal, trend, resid = result.seasonal, result.trend, result.resid
    resid_mu = resid.mean()
    resid_dev = resid.std()
    lower = resid_mu - 2* resid_dev
    upper = resid_mu + 3* resid_dev
    anomalies = df_date0[(resid < lower) | (resid > upper)]
    for inx,row in anomalies.iterrows():
         i_ann = inx
         df_date0['Demand'].loc[i_ann] = 0

    # (4)-----------------------------------------------------------------
    c_index = []
    for c in range(0,len(df_date0),72):
        df_head = df_date0.iloc[c:c+72]
        df_zeros = (df_head['Demand'] == 0).sum()
        df_nan = df_head['Demand'].isna().sum()
        z_n = df_zeros + df_nan
        if z_n > 7:
            c_index.append(c)

    df_date0 = df_date0.reset_index()
    for inx,row in df_date0.iterrows():
        i = inx
        if i in c_index:
            df_date0.at[i:i+71,['Demand','valid_days']] = 0
    df_date0 = df_date0.set_index('idx')

    # (5)-----------------------------------------------------------------
    df_date0.loc[df_date0['Demand'] == 0, 'Demand'] = np.nan

    "␣Interpolate␣missing␣values␣"
    df_date0 = df_date0.interpolate(method=i_method, axis=0,
                limit = interpolate_lim  ,order=3  ,directions='forward')

    "␣LABELING␣days␣of␣the␣week"
    monday = df_date0.loc[df_date0['dow']==0]
    tuesday = df_date0.loc[df_date0['dow']==1]
    wednesday = df_date0.loc[df_date0['dow']==2]
    thursday = df_date0.loc[df_date0['dow']==3]
    friday = df_date0.loc[df_date0['dow']==4]
    saturday = df_date0.loc[df_date0['dow']==5]
    sunday = df_date0.loc[df_date0['dow']==6]

    # feriados municipais das regioes
    if loc == 'Aveleira':
        feriado = '2018-06-24'
        feriado1 = '2019-06-24'
    elif loc == 'Albarqueira':
        feriado = '2018-04-07'
        feriado1 = '2019-04-07'
    elif loc == 'Espinheira':
        feriado = '2018-07-17'
        feriado1 = '2019-07-17'

    d_18= {'National␣holidays':[jan, abr0,mai0,mai1,jun,ago,nov,dez0,
                              dez1,dez2,feriado]}
    d_19 = {'National␣holidays':[jan_1, abr1_1, abr2_1,mai0_1,jun0_1,
                              jun1_1,ago_1,out_1, nov_1,dez0_1,
                              dez1_1,dez2_1,feriado1]}
    dd_18= {'Individual␣deviating␣days':[new_year,good_friday,asc_day_after,
                                      asc_day_after1, libertation_day]}
    dd_19 = {'Individual␣deviating␣days':[new_year_1,good_friday_1,
                                      asc_day_after_1, asc_day_after1_1,
                                      libertation_day_1]}

    """ LABELING DAY TYPES__________________________________________
        order:
                1 - set 'Type' to dotw
                2 - set 4 holydays (chrismas, carnaval, easter, summer)
                3 - set all saturdays and all sunday
                4 - set all national holydays to sunday
                5 - set individual deviating days
    ______________________________________________________________
    """

    # (1) already done when i created 'type' column ----------------

    # (2) holydays-------------------------------------------------
    df_date0.loc['2018-12-17':'2019-01-02','Type'] = 10 # 2019
    df_date0.loc['2019-12-17':'2020-01-06','Type'] = 10 # 2020

    df_date0.loc['2019-03-04':'2019-03-06','Type'] = 11 # 2019
    df_date0.loc['2020-02-24':'2020-02-26','Type'] = 11 # 2020

    df_date0.loc['2018-03-26':'2018-04-06','Type'] = 12 # easter

    df_date0.loc['2018-06-16':'2018-09-12','Type'] = 13 # 2018
    df_date0.loc['2019-06-24':'2019-09-09','Type'] = 13 # 2019

    # (3) saturdays and sundays------------------------------------
    df_date0.loc[df_date0.dow==5, 'Type'] = 5
    df_date0.loc[df_date0.dow==6, 'Type'] = 6

    # (4) national holydays ------------------------------------------------
    for k in range(len(d_18['National␣holidays'])):
        df_date0.loc[d_18['National␣holidays'][k],'Type'] = 6
    for k in range(len(d_19['National␣holidays'])):
        df_date0.loc[d_19['National␣holidays'][k],'Type'] = 6

    # (5) individual deviating days ----------------------------------------
    for g in range(len(dd_18['Individual␣deviating␣days'])):
        df_date0.loc[dd_18['Individual␣deviating␣days'][g],'Type'] = 90
    for g in range(len(dd_19['Individual␣deviating␣days'])):
        df_date0.loc[dd_19['Individual␣deviating␣days'][g],'Type'] = 90
    return df_date0
```

Appendix B

Implementation of ARIMA model

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX
import fun_labeled_data as lbldata

loc='Aveleira'
df_date = lbldata.data_treatment(loc)
df_date.set_index('Date')
data_sample=2*24*60    # 2 months for ARIMA model fitting

# Resample foe 30 min
df3 = df_date.resample('30T').mean()

# select a sample of data (from tail)
df_sample=df3.tail(data_sample)

# data splitting
l1=len(df_sample)
l_75=int(75*l1/100)
train = df_sample[:l_75]
test = df_sample[l_75:]

# sarima implementation
model = SARIMAX(train, order=(1,0,1), seasonal_order= (1,1,0,48),
         enforce_stationarity=False, enforce_invertibility=False)
model_fit=model.fit()
forecast = model_fit.get_forecast(steps=48*7)    # forecast 1 week

# plot results
ax=df_sample.plot()
ax1 = forecast.predicted_mean.plot(ax=ax, label='forecast')
plt.show()
```

Appendix C

Implementation of heuristic model

```
from os import system
import matplotlib.ticker as ticker
import sys
import numpy as np
import time
import pandas as pd
import datetime
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter
from datetime import datetime, timedelta
import plot_template as plt_temp
import fun_labeled_data as lbldata
import bakk_func as bakk

loc='Aveleira'

# DATA TREATMENT

df_date0 = lbldata.data_treatment(loc)
df_date0.set_index('Date')

"␣Time␣constats␣"
one_day_hours = 24        # amount of hours in one day
hour_minuts = 60          # amount of minuts in one hour
week_days = 8             # amout of type days in one week
hour_steps = 20           # original timesteps (10min from SCUBIC)
week_time_interval_original = int((hour_steps/hour_steps) *
                                one_day_hours * week_days)
"""
Bakkers model for water demand forecasting is made of three steps
    1st -   the average water demand for the next 48 hours is forecasted
    2nd -   the normal water demands for the individual 15-min.
            time steps are forecasted
    3rd -   if applicable, extra sprinkle water demands for the individual
            15-min. time steps are forecasted

"""
final = []
original = []
accuracy_step = []
n_days_fore = int(7)      # how many days to forecast next
input_day = 23            # today (day to predict)
input_month = 12          # month
input_year = 2019         # year
fst_day =  datetime(2018, 3, 18)



"""
                         #Training set
--------------------------------------------------------------------
"""
for f in range(n_days_fore):     # next days to forecast
        tic = time.clock()
        today = datetime(input_year,input_month,input_day)
        if f != 0:
                today = (today + timedelta(days=f))
        today_str = (datetime(input_year,input_month,input_day)).strftime('%Y-%m-%d')
        yesterday = (today + timedelta(days=-1))         # 1 day before
        beforeyesterday = (today + timedelta(days=-2))  # 2 days befor
        tomorrow = (today + timedelta(days=1))          # day after
        aftertomorrow = (today + timedelta(days=2))

        df_yesterday = df_date0.loc[yesterday]
        df_beforeyesterday = df_date0.loc[beforeyesterday]
        df_today = df_date0.loc[today]
        df_tomorrow = df_date0.loc[tomorrow]
        df_aftertomorrow = df_date0.loc[aftertomorrow]
        total_n_days = int((today - fst_day).days+1)

        weeks = (total_n_days + f)/7
        n_days = 0
        m_prio = 10
        n_prio = 5

        bakker_minuts_steps = 20
        original_minuts_steps = 10

        # function to return constants that depends of minuts_steps
        n_i, n_steps, m_prio_aux, n_prio_aux = lbldata.fun_time_const(bakker_minuts_steps,
                                                total_n_days, weeks, n_days, n_prio, m_prio)
        Q_t = df_date0[fst_day:(yesterday.date()).strftime("%Y-%m-%d")]
        Q_t_previous = df_date0.loc[(beforeyesterday.date()).strftime("%m/%d/%Y"):
                                                (yesterday.date()).strftime("%m/%d/%Y")]

        # INIT EMPTY ARRAYS
        f_qtr = []                          # last Q_t
        f_qtr_global_td = []                # last stored Q_t (m/n)
        f_qtr_global_tm = []
        f_qtr_typ = []
        f_dotw_typ_prev = []
        f_dotw_typ_next = []
        f_qtr_typ_global = []
        Q_corr_t = []
        Q_forc_norm_f = []
        j_dotw = []                         # array with dotw of last 48 hours
        j_dow_global_td = []                # array with dotw of last 5 week
        j_dow_global_tm = []
        j_type = []
        j_type_global = []


        "␣FACTOR␣of␣Typical␣15-min␣step:␣f_qtr␣--------------------------------------------"

        for i in range(len(Q_t_previous)):

                j = Q_t_previous['Type'].values[i]
                s = bakk.f_qtr_fun(Q_t_previous,i,n_steps)
                f_qtr.append(s)
                j_dotw.append(j)
        Q_t_previous_dotw = np.unique(Q_t_previous['dow'].values)

        "␣Selecionar␣apenas␣os␣tipos␣relevantes␣com␣determinado␣n"
        ttoday = df_today['Type']
        ttomorrow = df_tomorrow['Type']

        Q_t_type_td = pd.DataFrame(columns=['Idx','Type','Demand'])
        Q_t_type_tm = pd.DataFrame(columns=['Idx','Type','Demand'])
        count_n_td = 0
        count_n_tm = 0
        c_td = 0
        c_tm = 0
        nnnn=int(n_steps*n_prio)

        # Selecting Q_t_type for today
        for index,row in Q_t[::-1].iterrows():
                if row['valid_days']==1:
                        line_type = row['Type']
                        line_demand = row['Demand']
                        line_idx = index
                        count_n_td += 1
                        if line_type == ttoday and c_td < nnnn:
                                c_td += 1
                                line = {'Idx': [line_idx], 'Type': [line_type],
                                        'Demand': [line_demand]}
                                df_line = pd.DataFrame(line)
                                Q_t_type_td = Q_t_type_td.append({'Idx': line_idx,
                                        'Type': line_type, 'Demand': line_demand},
                                        ignore_index=True)

        # Selecting Q_t_type for tommorow
        for index,row in Q_t[::-1].iterrows():
                if row['valid_days']==1:
                        line_type = row['Type']
                        line_demand = row['Demand']
                        line_idx = index
                        count_n_tm +=1
                        if line_type == ttomorrow and c_tm < nnnn:
                                c_tm +=1
                                line = {'Idx': [line_idx], 'Type': [line_type],
                                        'Demand': [line_demand]}
                                Q_t_type_tm=Q_t_type_tm.append({'Idx': line_idx,
                                'Type': line_type, 'Demand': line_demand},ignore_index=True)

        # split arrays in days
```

```
Q_t_split_td = np.split(Q_t_type_td, int(len(Q_t_type_td))/int(n_steps))
print('Q_t_type_tm', Q_t_type_tm, len(Q_t_type_tm),n_steps)
Q_t_split_tm = np.split(Q_t_type_tm, int(len(Q_t_type_tm))/int(n_steps))

# for today
for jj in range(len(Q_t_split_td)):
    Q_t_jj = Q_t_split_td[jj]
    for ii in range(len(Q_t_jj)):
        l = bakk.f_qtr_fun(Q_t_jj,ii,n_steps)
        f_qtr_global_td.append(l)

# for tomorrow
for jj in range(len(Q_t_split_tm)):
    Q_t_jj = Q_t_split_tm[jj]
    for ii in range(len(Q_t_jj)):
        l = bakk.f_qtr_fun(Q_t_jj,ii,n_steps)
        f_qtr_global_tm.append(l)

# geting f_qrt_type for today and for tomorrow
for xx in [f_qtr_global_td[::-1], f_qtr_global_tm[::-1]]:
    for yy in range(n_steps):
        j = xx[yy::n_steps]
        l = bakk.f_qtr_typ_fun(j)
        f_qtr_typ_global.append(l)


"␣FACTOR␣of␣Typical␣day␣of␣the␣week:␣f_dow␣------------------------------------------------"

yyesterday = df_yesterday['Type']
bbeforeyesterday = df_beforeyesterday['Type']

for tod in [yyesterday,bbeforeyesterday]:
    t =  bakk.f_dotw_typ_fun(Q_t[::-1],tod,n_steps)
    f_dotw_typ_prev.append(t)
f_dotw_typ_prev[::-1]

for tod in [ttoday,ttomorrow]:
    tt =  bakk.f_dotw_typ_fun(Q_t[::-1],tod,n_steps)
    f_dotw_typ_next.append(tt)

"␣STEP␣1␣"
n_row, _ = Q_t_previous.shape
steps = n_row/2
C1 = 0.85
C2 = 0.15
count_idx = 0
for index,row in Q_t_previous.iterrows():
    count_idx +=1
    Q_t_val = row['Demand']
    if count_idx <= steps:
        Q_corr_t_val = Q_t_val/ f_dotw_typ_prev[0]
    else:
        Q_corr_t_val = Q_t_val/ f_dotw_typ_prev[1]
    Q_corr_t.append(Q_corr_t_val)

C1_aux = C1 * (1 / steps * np.nansum(Q_corr_t[int(len(Q_corr_t)/2):len(Q_corr_t)]))
C2_aux = C2 * (1 / steps * np.nansum(Q_corr_t[0:int(len(Q_corr_t)/2)]))
Q_forc_corr_avg = C1_aux + C2_aux

"␣STEP␣2␣"
count_idx_1 = 0
for t in range(n_steps*2):
    count_idx_1 +=1
    if t <= n_steps:
        Q_forc_norm_val = Q_forc_corr_avg*f_dotw_typ_next[0]*f_qtr_typ_global[int(t)]
    else:
        Q_forc_norm_val = Q_forc_corr_avg*f_dotw_typ_next[1]*f_qtr_typ_global[int(t)]
    Q_forc_norm_f.append(Q_forc_norm_val)

fst_day_aux = pd.to_datetime(Q_t_previous['Date'].tail(1).values[0])

add_day = timedelta(days=1)
add_week = timedelta(days=7)

measured = df_date0[(fst_day_aux+add_day).strftime("%Y-%m-%d"):
        (fst_day_aux+add_day+add_day).strftime("%Y-%m-%d")]    # aka original data
forecasted = Q_forc_norm_f
```