M.SC. THESIS

InnoEnergy MSc Energy for Smart Cities
Màster Universitari en Enginyeria de l'Energia

Energy Demand Forecasting in Smart Buildings

REPORT

Autor:	Álvaro Picatoste
Acadèmic Director:	Andreas Sumper
Company:	COMSA Industrial
Industry Director:	Albert Cot
Sessions:	September 2017

ETSEIB

ETSEIB

Disclaimer

This study has been carried out by the student in collaboration with the company. The company has solely provided access to the data for the analysis but has had no influence on their treatment nor process. All the information and results shown in the report are the outcome of the student's work. The company refuses all responsibilities for mistakes in the data collection nor it analysis and does not confirm that the results are legit nor represent the reality.

ETSEIB

Abstract

Energy demand forecasting has become a relevant subject in the energy management field. Different techniques are being currently applied to forecast the energy demand for different time horizons and for diverse types of loads. Some of them are based in complex Machine Learning (ML) algorithms, which maps the energy consumption to a set of influence parameters or inputs, such as the historical data consumption, the weather or other variables, making it possible to predict the energy demand.

Important management decisions from different stakeholders in the Energy sector are based on these predictions and, therefore, it is important to rigorously assess the performance of these predictive models. A specific methodology is presented in this dissertation through its application over a real-building case-study in which energy demand predictions are being carried out by a ML model. All the steps in the evaluation process are explained and exemplified, including the data gathering, evaluation period selection, data preprocess with special emphasis in the data abnormalities an its relation to the process dynamics and, finally, the data process itself. The accuracy of the model and the main parameters of influence are evaluated through four different metrics and data visualizations, based mainly in box-and-whisker plots.

Several anomalies when predicting energy consumption in a disaggregated load (single building) have been found in the study. By removing them the stability of the case-study model is around 88%. The metrics yield a MAPE (Mean Absolute Percentage Error) of 18.05% and a MBPE (Mean Biased Percentage Error) of -4.67%. While being values within the literature range they show a poor accuracy. Nevertheless, there is space for improvement and by re-training, refining and calibrating the model it will be possible to improve its performance. The day of the week, the working calendar and the hour of the day showed to have a strong influence over the error metrics analyzed.

Other alernative Machine Learnings methodologies have been applied to the same dataset and their performance have been analyzed. Artificial Neural Network, k-Nearest Neighbors and Random Forest based models have been compared after training with more than 1-year hourly Energy Consumption data and other influence variables. The Random Forest achieved the best accuracy when re-trained, showing a MAPE below 10%. The importance of passing a detailed working calendar to the model, using accurate weather variables forecasts and defining an adequate re-training strategy have been proved to improve model accuracy.

CONTENT

ETSEIB

List of figures

List of tables

Glossary

AI	*Artificial Intelligence*
AMI	*Advanced Metering Infrastructure*
ANN	*Artificial Neural Network*
APE	*Absolute Percentage Error*
BBL	*Building Base Load*
BPE	*Biased Percentage Error*
DR	*Demand Response*
EDF	*Energy Demand Forecasting*
EVs	*Electrical Vehicles*
FFNN	*Feed Forward Neural Network*
k-NN	*k Nearest Neighbors*
LTLF	*Long Term Load Forecasting*
MAPE	*Mean Absolute Percentage Error*
MBPE	*Mean Biased Percentage Error*
ML	*Machine Learning*
MLR	*Multivariate Lineal Regression*
MRE	*Mean Relative Error*
MTLF	*Mid Term Load Forecasting*
NA	*Missed Value*
RMSE	*Root Mean Square Error*
R^2	*Coefficient of determination*
RF	*Random Forest*

SaaS: Software as a Service

SD: Standard Deviation

SBs: Smart Buildings

SGs: Smart Grids.

STLF: Short Term Load Forecasting

WBE: Whole Building Electrical

ZV: Zero Value

1 Preface

1.1 Origin of the project

This master thesis project is born in the R&D Energy & Systems department of the firm where the student has completed his master internship. Within the company, the focus of his work was mainly on writing project proposals for the European Union founding program Horizon2020. One of the projects in which the firm participated in the previous years had led to the creation of a new company. This company, to which this document will refer as "*THESTARTUP" has* developed a product for improving the energy management in tertiary buildings. It is formed by a suite of application with different specific objectives and functionalities. One of these, to which this document will refer as "*PREDICTION",* produces energy demand forecasts that predicts hourly energy consumption from 24h to 48h ahead in Smart Buildings. As these forecasts have been running in some pilot buildings for a long time now, it is of importance to be able to assess its performance. The firm and the student agreed to start a project to design a methodology that allows this evaluation in a simple and scalable way. *PREDICTION* is intended to be installed in a wide set of different buildings to which each performance should be assessed. With a proper assessment corrective measures could be suggested to improve the model. In addition, alternatives to the technology behind the *PREDICTION* model should be considered to improve the accuracy given the case the evaluation shows a deficient performance of the current implementation.

1.2 Motivation

Energy systems are facing several challenges in the last decades that requires an optimized management of the system. From renewable distributed generation to building energy consumption, it is crucial to be able to anticipate the energy needs and resources of the grid. The system must work to fit the energy requirements and knowing what these will be in advance will optimize their performance (Wahid and Kim, 2016). Utilities and retailers will take advantage of being able to forecast the needs of the power systems at network and distribution level. Accurate load forecast enables demand management and energy efficiency initiatives that will optimize the power flow and avoid expensive electricity network infrastructure investment (Yildiz, Bilbao and Sproul, 2017). In addition, at users' level, predictive models enable mechanism for the building users to make a smarter energy usage through peak shifting, smart contracts or demand response.

On the other hand, although all these agents will benefit from energy (Demand or generation) forecast models, they are hardly understood due to its complexity. It is common that these models are developed by third parties, with not precise specifications on how they work. As a result, it becomes essential to be able to assess their performance and to understand how well they are making the predictions.

Therefore, the motivation behind undertaking this dissertation comes from two sides. On one hand, to contribute to a subject that will have an impact in the upcoming years in the management of the different energy systems. On the other hand, from that need of assessing others` models when applied to propietary systems, help in its understanding and being able to study alternatives.

2 Objective

The objective of this thesis can be split into two main lines. The first objective of this master thesis is to design a methodology, through the analysis of a real case-study, to assess the accuracy of energy demand forecast models being implemented in Smart Buildings. In addition to a general performance study, the methodology aims to reveal which circumstances most affect the accuracy. The second objective is to compare , by using the designed methodology in the first part, alternative Machine Learning models when applied to the same Energy demand forecasting problem.

2.1 Scope of the project

This project aims to develop a methodology to assess forecast for Smart Buildings at individual level and to give an overview of current implementations in the field of Machine Learning (ML) applied to energy predictions.

In this master thesis there is no algorithm development. The main models currently applied will be analyzed but the algorithmic behind them will remain as a, what is commonly referred in the field, *black box*. This study will remain more practical than theoretical and no complex mathematical equations will be shown. To assist the reader that is willing to go deeper in this sense, a set of references will be pointed out through-out the document.

The scope will focus in whole building electrical consumption (WBE). Another well-known and related application of ML in Smart Buildings is the fault/diagnosis of energy building detection. In this document, this topic is not tackled and no analysis has been performed in this line. The energy consumption in the document refers solely to the WBE and no specific systems, like climate or lighting, will be considered.

Another criterion that will not be considered throughout the report is the computational time for carrying out the models. As the dataset with which the models will be working is small this is not of importance at this level and it is not part of the scope of the project. Nevertheless, when working in bigger environments, like in building clusters, it is parameter that should be considered in the analysis.

3 Introduction

In this section the topic which this projects tackles, that is, Energy Demand Forecasting models and their accuracy, is presented. The importance of precise enough demand forecasting and the different methodologies being applied are explained in 3.1. More specifically, main Machine Learning (ML) models are shortly described in 3.2. Furthermore, a brief literature review and previous work on the topic is presented in 3.3. The expected contribution of this master thesis to the field is advanced in 3.4. The tools that have been used in the dissertation are described in 3.5. Finally, in 3.6, the structure of the remaining dissertation is described before going onto the next section.

3.1 Energy Demand Forecasting in Buildings

In the last decades there have been a lot of significant changes in the Electric systems. New players, such as EVs, Smart Customers, Renewable Energies, and the development of information and communications technologies (ICTs) have led to the creation of innovative concepts that completely implies changes in the management of the electrical grids. In this new scenario forecasting models, both for generation and demand, play a crucial role to harness energy at the lowest cost, and to guarantee optimal quality and reliability of the energy supply. For all these recent concepts, such as *Smart Grids (SGs), Distributed Generation (DG)*, *Virtual Power Plants (VPPs)*, *Demand Response (DR), Microgrids (MGs)* and *Smart Buildings (SBs),* it is extremely valuable to be able to forecast the demand and generation within their domains (Hernandez et al., 2014).

Accurate Energy Demand Forecasting (EDF) models will improve the real-time and long-term performance of power systems (Boroojeni et al., 2017). At the building level, deployment of measurement and intelligence devices enables an advanced metering infrastructure (AMI) and data is now more available and easy to collect. Regarding the building owners and facility managers, they can benefit from anticipating the energy demand of their assets in many ways: Demand side management and Demand Response for control of residential and industrial loads, optimization of O&M operations, cheaper energy procurement, energy-efficient building control strategies, etc.

Energy Demand Forecasting methodologies

Energy consumption in buildings is mainly due to heating/cooling load, hot water and electricity consumption. At the same time, there are several factors that have a strong and demonstrated influenced over it, such as the weather conditions, the use of the building, construction materials, occupancy and schedule, etc. Consequently, Energy demand presents many *non-linearities* which need to be detected and later transferred into equations that model it (Hernandez et al., 2014).

There are different techniques for load forecasting. They can be divided into five categories such as engineering methods, statistical methods, gray-box modeling methods, machine learning and artificial intelligence methods (Fan, Xiao and Wang, 2014).

The engineering methods are based in physical equations considering the systems in which the energy system of a building is based. Taking into consideration the thermal dynamics of the building and details of the machinery of the building systems (HVAC, lighting, etc.) and using complex equations the energy consumption can be modelled. The main difficulties of this technique are the difficulty of gathering all the data needed for the equations in a reliable way and the complexity of the equations that model the system. Software simulation tools, as the well-known EnergyPlus, are commonly used to model the energy consumption behavior of the building. Another drawback of this technologies is that tis tools required high skilled man labor to develop proper analysis. Nevertheless, these models, when properly implemented, are effective and accurate, and can represent closely to the actual energy consumption of the building. Once the model has been defined, given the new inputs it will be possible to accurately predict the energy consumption under the new conditions.

On the other hand, statistical models correlate the energy consumption with some influencing variables. In this case the main challenge to design an accurate model is to gather enough historical data. Although there is no need of complex physical equation and knowledge about the equipment and systems in the building, the models are completely based on the data gathered to use as inputs. The main advantages of these models are their relative simplicity to the engineering models. They accuracy is limited. Between this method it is worth mentioning the Auto-Regressive model with eXtra inputs (ARX), the Auto Regressive Integrated Moving Average (ARIMA) and the ARIMA with eXternal inputs (ARIMAX). One of the main drawbacks of these methods is error produced when abrupt changes in environments or sociological variables (e.g. changes of the weather, type of the day, etc.) occurs.

Grey-box modeling combines both previous methodologies partially. These models integrate qualitative and quantitative knowledge. Regarding EDF they include a simplified model of the engineering systems involved in the energy system of the building and other data of some influence variables such as weather, solar irradiation, cloud clover, etc. When some parameter of the engineering method is unknown, the data-base methods are used to fill in that missed information (Hauth, 2008).

The machine learning and Artificial Intelligence methods can be grouped together as *back-box* models. For these models, opposite to the engineering methods, there is not differential equations modelling the behavior of the energy system. They are all data-driven based and can include complex algorithms that find relationship between desired output and some influence variables. Within this group the most widely used one is the Artificial Neural Networks (ANNs), Supper Vector Machines (SVM), Random Forest (RF), etc. The main drawback of this models is their interpretability.

Forecast time horizons

When referring to prediction models it is relevant to always specify the time horizon of the predictions. Which methods to apply and the data inputs and preprocess is highly influenced by the time horizon. Furthermore, the accuracy and error tolerance can vary substantially depending on the horizon desired. Researchers agrees on three main categories when referring to load forecast time horizon (Raza and Khosravi, 2015):

- Long term load forecast (LTLF): It refers to horizons from 1 year to 10 years ahead. Typically used for long term power system planning (e.g. assets utilities plan).
- Medium term load forecast (MTLF): Horizons from 1 Month to 1 year ahead. Aiming to the efficient operation and maintenance of the power system.
- Short term load forecast (STLF): For horizons from 1h or 1 day to 1 week ahead. Used mainly for adjust generation and demand. This category has a strong influence on optimum unit commitment, control of spinning reserves, evaluation of sales/purchase contracts between various companies, etc. The focus of this study is in this category.

3.2 Machine Learning applied to Energy Demand Forecasting

As it has been already mentioned the Energy System physical approaches involves complex physical differential equations. In contrast to that, machine learning model derives functional dependencies between an output and certain inputs, directly from the observations. (Heinermann and Kramer, 2016). Compared to physical models, ML are much easier to be implement although it is harder to identify the optimum parameters and they require data over a prolonged period for a proper training process. (Zhou et al., 2008)

Tom Mitchell, a recognized eminence in the topic, defines machine learning by defining learning problem as follows: “A computer program is said to learn from experience E, with respect to some task T, and some performance measure P, if its performance on T as measured by P improves with experience E”. Following this definition and applying to the case of this study the learning problem includes the task T (to predict the energy consumption 24 and 48 hours ahead), which performance is measured by P (the error metrics defined in 4.3.1), and that is improving with experience E (each prediction when compared to the final real consumption, so every hour predicted and then checked can be assumed to be a gained experience).

With ML it is possible to model the energy behavior of a building while not knowing the technical specifications but just considering the historical data and related variables. The development of this models is heavily based in the amount and quality of data available, the capabilities of the algorithm utilized and computing time to learn from this data. As main drawback, this modelling strategy needs data available and there is a cost associated to the acquisition and deployment of the sensors that will provide the data. Some authors argue that it is nowadays not feasibly to heavily instrument a building cost-effectively (Edwards, New and Parker, 2012). Nevertheless, sensor developing and costs are dropping with the same transistor density sound every 18 months as defined by Moore’s Law (Schaller, 1997). In addition, accuracy depends on the target variable to predict and when predicting hourly whole building electrical (WBE) consumption, it is easier to obtain satisfactory results thatn when focusing in more specific variables.

3.3 Prediction models’ evaluation

To design a good forecast model, the load data should be closely analyzed and dynamics should be clearly understood (Raza and Khosravi, 2015). In order to assess their performance, the same should stands. In the literature it is agreed that there is no single metrics that can be universally used to evaluate the performance of forecast models.(Mehdiyev et al., 2016).

In the accuracy measures field there are many several types of metrics used. There are scale-dependent measures, such as the Mean Absolute Error (MAE), an easy interpretable metric to assess the error. This metrics are easily biased in the presence of extremely large outliers. Even a single large error can sometimes dominate the results (Chen, Twycross and Garibaldi, 2017). On the other hand, there are Percentage-based measures that are scale-independent, such as the Mean Absolute Percentage Error (MAPE). Also, scale-independent, the Relative-based measures are frequently used, as the Mean Relative Absolute Error (MRAE), that compares the performance of a model to a reference model for similar conditions. The selection of what kind of measure use depends on a comprehensive analysis of the data and the specifics of the model being implemented.

When referring to Energy Demand forecasting the most common accuracy measures relates to statistical error metrics such as MAPE or Mean Biased Error to name some examples. Nevertheless, the metrics changes between authors and there is not official reference in the sector neither. There are competitions on predictive models around the globe with relative frequency and these usually serve as a reference for the metrics and evaluation methodologies. For example, in the energy sector one of the most famous one is the Global Energy Forecasting Competition held in 2012 and in 2014 and that used the Weighted Root Mean Square as their evaluation method. Some other important competitions on the forecasting field are the M-Competitions, using the Mean Squared Error, or the Indoor Positioning and Indoor Navigation Competition (IPIN 2015 in its last session), using the RMSE.

All this variability in the used metrics stresses the need for a framework to evaluate he forecasts methods. This master thesis also aims to contribute, where possible, with the design of a methodology that gather all the important parameters to consider when evaluating the performance of an EDF model.

3.4 Literature Review and previous work

Forecasting the energy consumption has been studied since the mid-50s, when (Gillies, Bernholtz and Sandiford, 1956) presented his approach to predict peak load based on curve trends comparing the weather predictions to peak loads. In 1970 a STLF model for state estimation in a power system, based on regression algorithms, was presented by (Toyoda, Chen and Inoue, 1970). In the 80s several several authors introduced some non-liner models based on ANN. One of the main reference for this models can be found in (Hopfield, 1982), which gave its name to an specific model, the Hopfield Netowrk. Since then many authors have worked with these models, that ended up being very popular between the EDF researchers. (Raza and Khosravi, 2015) offers an extend analysis of different ANN models applied to electric load forecasting. Since those early studies on the EDF there has been a lot of authors using different models to predict energy demand with different time horizons and for different kind of loads. One of the best collection of load forecasting models can be found in (Hernandez et al., 2014). In their research the authors enlist over 70 previous real case-studies and applications of electric power demand forecasting, describing the model used, predictors, forecasts, horizons, application areas, error metrics and other information.

Most of the literature deals with aggregated loads such as electricity demand at district level, region level or even country level. There are less studies on disaggregated loads, specially at single building level. (Fan, Xiao and Wang, 2014) studied the next-day energy consumption of the tallest building in Hong King by using 8 different ML. (Edwards, New and Parker, 2012) analysis the energy consumption in residential building to make next-hour energy predictions by seven different ML models.(Massana et al., 2015) research shows predictions of STLF to non-residential buildings in the University of Girona. (Boroojeni et al., 2017) makes STLF predictions on the region level (New York) for different time scales using time series. (Yildiz, Bilbao and Sproul, 2017) predicts the electricity load of a University building in Wales using Multivariate Linear Regression and other NN techniques.

3.5 This master thesis contribution

This master thesis will contribute to the EDF field in diverse ways. Most of the work done so far it is referred to grids or microgrids consumption, meaning that the energy analysis has been performed considering a certain number of aggregated loads (building clusters, microgrids, regions, etc.). This dissertation is focused in a single building. Although this single “load” could be broken down into smaller pieces (lighting systems, floors, etc.) it represents a disaggregated level that has not frequently studied in the literature. This master thesis is a unique real case-study with actual data on the consumption of a middle size building. It hopes to serve as an example for small size loads and as a reference for other buildings studies.

The evaluation methodology proposed aims to provide some guidelines in the assessment of prediction models when they are applied to electric loads at building level. Furthermore, it pretends, by comparing certain ML models, to facilitate the decision of choosing an appropriate forecasting method. Most of the literature, when referring to the accuracy of predictions models, sticks to some statistical metrics but do not really deepen in comprehensive analysis into the physical meaning of the results. This study hopes to help understanding the dynamics of load curve of a tertiary building and its relationship with the predictions. It pretends to set the base knowledge for those non- experts in the energy field and help them assessing their models when applied to EDF.

Furthermore, and up to a lesser extent, this master thesis aims to apply some of the ML methodologies less frequently found in the literature. A version of a k-NN model will be study for EDF in the case-study. Lastly, this dissertation can be used as an updated literature review on the topic.

3.6 Tools used in the project

The analysis performed in this dissertation have been carried out using the statistical programming language known as *R* through the friendly environment *RStudio*. R was created by Ross Ihaka and Robert Gentleman in 1995 and since then, it has gained popularity between data scientist around the world. It is an open source advanced statistical language and its ease to support extensions and the vast community using it has led to a great diversity of available packages. These packages are collections of R functions, data, and complied code that can be installed in R with one single line. Some the packages that have been frequently used in this master thesis are:

> *Dplyr:* for data manipulation, written and maintained by Hadley Wickham. It provides some great, easy-to-use functions that are very handy when performing exploratory data analysis and manipulation (Kodali, 2017).
>
> *Zoo*: consists of the methods for totally ordered indexed observations. It aims at performing calculations containing irregular time series of numeric vectors, matrices & factors. (Sunku, 2017)
>
> *Caret*: maintained by Max Kuhn, is the go-to package in the R community for predictive modeling and supervised learning. This widely used package provides a consistent interface to all of R's most powerful machine learning facilities. (Blog, 2017)
>
> *Ggplot2:* Although is not one of its strengths R has good data visualization options though this package, which has been used for all the graphs presented in this report.

Although there are other good programming languages that are also popular in the *data analytics* field, R has been chosen over all of them. The main reason is the vast community that supports it and that has published all these available package, automating different task, saving time and making it easy to perform more complex models. Given the time given and the nature of the project it is more suitable to reach an enough appropriate programming level than with other programming languages. Furthermore, the computing time for the dataset involved in this project it is not important. If the project dealt with bigger dataset another fastest languages should be considered as the time for computing the models would exponentially increase and R is a slow language. Part of the scripts and code (written in R) will be added as ANNEX to this report.

3.7 Report structure

After this first introduction to the topic and the intended contribution of this dissertation the rest of the document structures as follows: In Section 4, the first part of the master thesis is described and the methodology proposed is at the same time applied over a real case-study. System description, data inputs and preprocess, data process, error metrics and evaluation of the results are discussed in this section. In Section 5, and second part of the document, three models are used to predict the energy consumption in the same case-study conditions than the first part. Data inputs, feature selection and results evaluations are presented. Later, in section 6, the results obtained from section 4 and 5 are commented and compared and some final conclusions are given. To close the study some recommendations of future work and acknowledgment for the accomplishment of this study are presented in sections 7 and 8 respectively.

As the different ANNEXs accounts for a considerable share of the document a brief description of each one is given also at this point. ANNEX I shows the specifications of the devices from the AMI. ANNEX II collects the graphical representations of the dataset of each individual month for the whole evaluation period (see Figure 5) to facilitate the visualization of the data and data cleanness process to the reader. ANNEX III shows some results for the selected model in part II. Lastly, ANNEX IV and ANNEX V explains written code for this master thesis and gathers samples of all the different scripts used in the evaluation of the case-study.

4 PART I: Methodology for the assessment of EDF models through a case-study.

This section refers to the objective I of the project, that is, to the design of a methodology to evaluate the performance of forecasting models applied to Energy Demand in disaggregated tertiary buildings. Through a case-study the methodology is applied and all the steps in the process are explained. First, a brief description of the system in which the methodology is tested is given, including the building and the hardware description. Second, the data gathering and the pre-process applied is shown. Third, an analysis is performed with a specific set of error metrics. Lastly, the results and discussion on the methodology and the results are discussed and summarized.

4.1 Description of the case-study system

The case-study in which the proposed evaluation methodology is applied as an example is based in the energy demand of an office building. In this building THESTARTUP has installed the product PREDICTIONS making it possible to forecast the energy consumption of the building 24h and 48h ahead. Through a cloud software the user (facility manager or building owner) can request the forecast up to 2 days ahead. Details on the building and the hardware and software involved in the process are given in this section.

4.1.1 Building characteristics

This COMSA headquarters office building is placed near the center of Barcelona (Spain), in the street Avinguda de Roma no. 25-27. It has a total surface of 2626 square meters distributed along its seven floors. The façade is covered by PV panels and the building is exclusively fed by electricity. The building’s air conditioning system is composed by three outdoors variable refrigerant flow units and fourteen VRF interior units, all from Mitsubishi; the total cooling power is 574 kW and the heating power is 640 kW.

Figure 1. Left: Emplacement of COMSA Headquarters in Barcelona and main entrance. Right: 3D Google view of the building.

4.1.2 Hardware and software infrastructure

This study does not pretend to go deep in the details of the measuring structure as it is out of the scope of the project (its focuses rather in analyzing the data). Nevertheless, the key device from which the data that will be used is being gathered will be explained in this section. All the advanced metering infrastructure (AMI) relies on the gateway and datalogger, a model DL172 from the company SenNet. This model is designed to monitor many electric loads in a compact format. It incorporates a radio frequency module (868 MHz) as well as RS232 and RS485 ports. It includes 6 three-phase electricity meters, configurable also as 18 single-phase or as a combination. The internal electricity meters allow monitoring power, energy (active, reactive and apparent), power factor, current, voltage, and frequency. It GPRS/GSM/3G connectivity allows it to send the data through a router connected to the Ethernet. Specifications of the device can be found in the ANNEX I.

This device has been installed in the building and it is measuring different energy parameters of each of the floors of the buildings. It has been running fully operative since 14th February 2017. Nonetheless, as for the shake of this study more data is needed, this study will consider just the general consumption data of the building, that is, the WBE. Since 2014, the meter installed have been measuring the total electric consumption and the specific of the 4th floor, where the department of R&D in Energy is placed. There is available reliable data of energy consumption of *WBE* since 1st January 2015. Despite having such an amount of available consumption data, PREDICTIONS started being operative later and reliable data from the application is just available from May 2016.

The software over the AMI is a cloud Software as a Service (SaaS) that can be accessed from any common browser (e.g. Google Chrome). It Is named DexCell and its property of the Spanish energy monitoring company *DEXMA Tech*. This Energy manager software can integrate a wide range of metering hardware and acts as a platform in which other application regarding energy management in buildings can be installed. That is the case of PREDICTIONS that can be incorporated to the software through the platform market place (in an equivalent way as applications for smart phones are sold through Play Store or Apple Store in Android and Apple respectively).

4.1.3 PREDICTION Energy demand forecast model

Once PREDICTION has been incorporated to the Energy manager DexCell it is possible to configure it and start using it, being therefore possible to predict the building's hourly electricity consumption up to 48h ahead and then compare it with real-time consumption.

PREDICTION application works through a proprietary machine-learning algorithm. This algorithm has been developed by a third party of the European Consortium that led to the creation of THESTARTUP. It must be clarified at this point that THESTARTUP and the author of this dissertation have not specific information on what kind of Machine Learning algorithm PREDICTION is based on, nor on the specific inputs that it is using. It is protected by Intellectual Property (IP) and the technology behind it will remain as a *black box.* Nevertheless, this lack of information will not affect the performance of the assessment methodology as it is a "evaluation" model and it does not consider *how* the results has been obtained but the results themselves.

This application models the behavior of the building's consumption based on historical data of energy use and its correlation with historical data of weather (outdoor temperature). Then, it estimates the evolution of the building's consumption in the near future taking the weather forecast available from a third-party service as an input. Once it has been installed and initiated, it learns the behavior of the building for a minimum of 6 week, after which it starts producing hourly demand electricity forecast with a 24h span for the building. A simplified schema of process is represented in Figure 2.

The training data refers to the data that allows the algorithm to learn how the building behaves. It involves the historical data, which is the measurement data of previous periods that represent the behavior of the building for specific weather external conditions. It also considers the building data, that is, static data of the building that can influence its behavior, including the location, the use (office building), data about the infrastructure, the working calendar, occupancy, etc.

The *Using* data tells the algorithm what predictions the user is asking for and provides some valuable information. It comprises the magnitude to be forecasted (WBE in our case), the forecast time horizon and the weather forecast (imported from third parties, in this case a company called Wunderground).

The outputs of PREDICTION are the energy demand forecast. The model will display the variable after the application has processed the inputs and the algorithm.

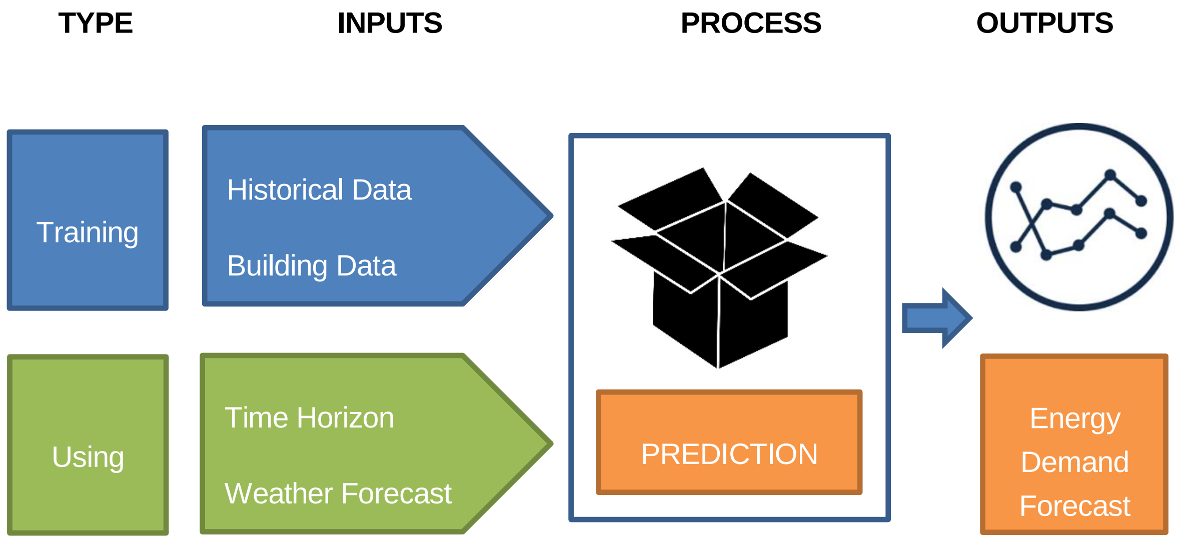
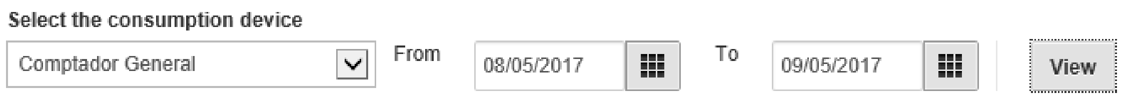


Figure 2. General process behind the PREDICTION application

This application, still at early-to-mid stage development has a simple user-friendly interface helped by the intuitive interface of the DexCell. After a quick configuration of main parameters of the building in which it will work upon, the user can request a forecast as shown in Figure 3.



Select the consumption device				
Comptador General	From	08/05/2017	To	09/05/2017

Figure 3. PREDICTION interface

The user can set the consumption device, in this case “Comptador general” or WBE, which will be studied. This consumption is what the energy retailer is providing to the building and the variable on which the app will launch the prediction. Then, the desired period must be selected, being just currently possible to select up to 2 days in advance from the actual date. It is also possible to check and retrieve the forecast of any period in the past, as shown in

Figure 4. In this figure, predictions made the 7th of May for the two upcoming days are shown together with now known real-consumpton (blue).

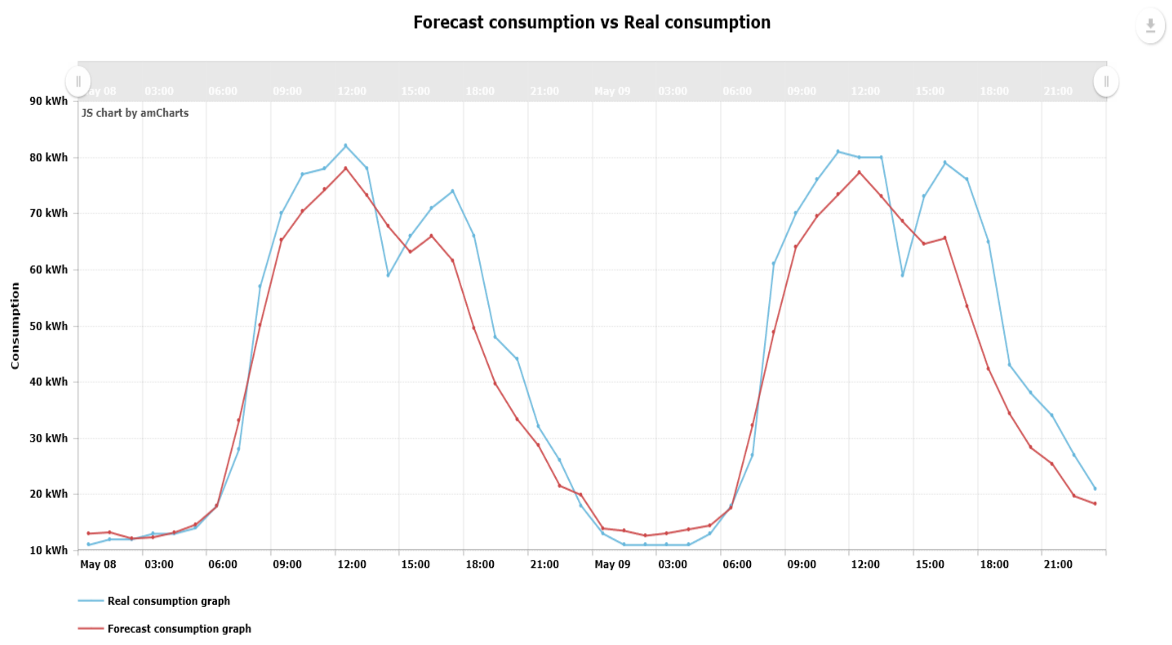


Figure 4. PREDICTION output appearance, once the real-consumption has been obtained.

4.2 Data selection, pre-process and cleanness

One of the first steps when working on a data analysis project, prior to the analysis itself, is always related to the data pre-process. It the data analytics field, it is said that a thorough data-preprocess must be carried to perform an accurate analysis (FAMILI et al., 1997). In this section, the data preprocess followed in this methodology is explained to detail, from the data selection to data cleanness and data visualization.

4.2.1 **Data Inputs and Period selection**

Both the AMI and the PREDICITION app have been set up and running in the building just for the last two years. There have been some changes in the configuration since they were first installed, influencing the quality of the data generation and the forecasts as these are dependent from the input data.

At the moment of writing these words the available data in the building comprises the following parameters:

- Total electric consumption of each floor of the building.
- Total electric consumption of climate installations of each floor
- The difference between the two-previous parameter, which is called “rest”. All the equipment that is consuming electricity in each floor but not included in the indoor climate operations is considered within this category. This could include lightning, computers, printers. etc.

Nevertheless, all this measures dates from February 2017, having just available total energy consumption of the whole building and the specific for 4th floor for previous periods. Due to the brief period logged for this specific variable this dissertation will consider as the variable to predict be the total electric consumption of the building (WBE). In future work it will be interesting, once a large period data has been logged, to perform the analysis over the other variables.

Regarding the period selected to evaluate the accuracy of the PREDICTION forecast model, it should be selected by taking into consideration the reliability of the data, both from consumption and from the forecast outputs (energy demand forecasts). As it was pointed in the previous section, from data consumption there is quality data since 2015 and from predictions from May 2016. Therefore, as same period for both is needed in order to make the comparison, the starting point is fixed in May 2016. The end of the period has been selected to be the latest possible in order to be able to have as much data as possible to perform the analysis. The end of the period is the 1st of June 2017.

The granularity of the measures selected is hourly. It has been commented already that in February 2017 there was an important update of the AMI and, apart from adding different parameters to the metering infrastructure, the granularity of the measures was also updated. Since then each measure is recorded every 15 minutes. Nevertheless, as it was explained for the variable to predict, the period is too short to be considered and the granularity will be the one available for the extended period, that is, the measures for every hour.

The period in which the assessment on the PREDICTION forecasts will be carried out is therefore from 1st May 2016 to 31th May 2017. The dataset contains 9504 instances, one per each hour of the 396 days this period comprises. A graphical representation of the timeline regarding the deployment of the AMI and PREDICTION and evaluation period can be shown in Figure 5.

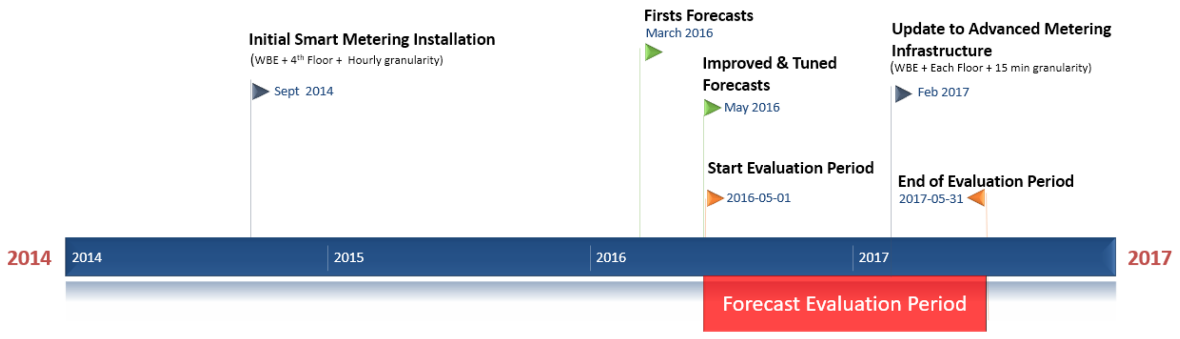
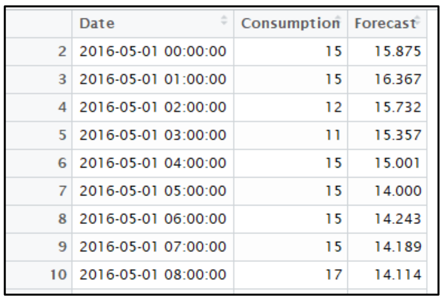

Initial Smart Metering Installation
(WBE + 4th Floor + Hourly granularity)
Sept 2014
Firsts Forecasts
March 2016
Improved & Tuned Forecasts
May 2016
Start Evaluation Period
2016-05-01
Update to Advanced Metering Infrastructure
(WBE + Each Floor + 15 min granularity)
Feb 2017
End of Evaluation Period
2017-05-31
2014
2015
2016
2017
Forecast Evaluation Period

Figure 5. Metering and Forecast timeline for COMSA Headquarters Building

Once that period of analysis has been decided, the associated data is requested and downloaded from the DexCell software in the format *Comma Separated Value* or *.CSV,* a usual format to work with data. As it was explained in the section 3.6, using R programming language it is loaded and set up for its analysis.



	Date	Consumption	Forecast
2	2016-05-01 00:00:00	15	15.875
3	2016-05-01 01:00:00	15	16.367
4	2016-05-01 02:00:00	12	15.732
5	2016-05-01 03:00:00	11	15.357
6	2016-05-01 04:00:00	15	15.001
7	2016-05-01 05:00:00	15	14.000
8	2016-05-01 06:00:00	15	14.243
9	2016-05-01 07:00:00	15	14.189
10	2016-05-01 08:00:00	17	14.114

Figure 6. First 10 instances of the dataset of the evaluation period once data is loaded in RStudio environment. Units in kWh.

In Figure 6 it is shown how the dataset has been arranged prior to any data-preprocess or cleanness. The first column contains the timestamp of the instance, the second one is the real consumption (WBE) for that time and the "Forecast" column represents the output of PREDICTION. For example, in the first instance of the evaluation period, the WBE was 15 kWh while the PREDICTION output (which was obtained 24 to 48h before) was 15.875 kWh.

4.2.2 **Data visualization**

Data visualization is a crucial phase in data analytics as it allows to obtain the first insights about what the predictive model is doing. By plotting the real vs forecasted energy consumption, it is possible to draw a first gross idea on how well a model is behaving. The dataset, containing 9503 instances, is split into months and plotted using ggplot2 package. The thirteen months of the evaluation period graphs are shown in ANNEX II. As an example, Figure 7 represents the real-consumption and the forecasted values for each hour of the first month of the evaluation period, that is, May 2016.

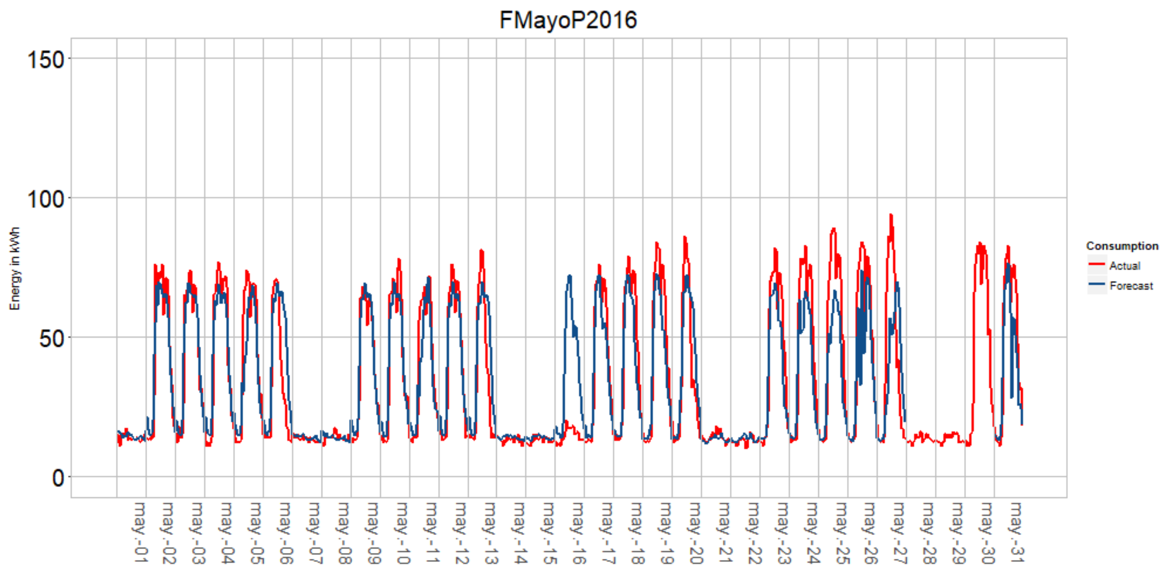


Figure 7. May 2016 representation of the PREDICTION output (Forecast, in Blue) and the real WBE consumption (Actual, in Red).

With the aid of the data visualization it is possible to detect some of the most common errors the model is incurring in. In the example of the Figure 7 there are two of them, one, detected on the 16th of May 2016 is related to working calendar and the other, on May 28th to May 30th where the model did not work at all. These errors, once identified their typology, will be detected in the whole dataset by implementing an algorithm in the R script. The different abnormalities be further explained in the following sections.

4.2.3 **Data Dynamics**

It is convenient at this point to explain the shape of the graphs and its relation to the process they are representing. Using Figure 7 as example the key features behind the dynamics of the energy consumption in an office building will be explain in this section.

There different patterns are depict in Figure 7:

- *Weekly pattern:* There is a 7 days period cycle that repeats almost through the whole evaluation period. This cycle its composed by 5 days of high consumption, associated to the days of the week (Monday to Friday) followed by 2 days of super low consumption, corresponding to weekends (Saturday and Sunday). In an office building most activity takes places during the weekdays corresponding to the official working days. This pattern can be altered by local or national holidays. In Figure 7 four whole weekly cycles can be observed, being the third one altered the 16th of May as it was local holiday and that Monday the building remained empty (it will be explained later why it was not forecasted by the algorithm).

- *Daily pattern:*
  - *Weekday*: Along the 24h of any of those 5 high energy consumptions days in a row another consumption pattern is observed. Every working day, energy consumptions sky-rockets from 7 to 9 in the morning when employees arrive in the building and start their working related activities. Then, it remains flat and lowers down and then another peak at round 15 takes place to go down again to meet the building base load at around 19, when all the employees leave the building and working activity for the day ceases.
  - *Weekend:* In a standard weekend day the consumption remains practically flat as there are not activities in it. It is not 0 as any building has some systems that keeps working even when it Is not a working day. This minimum consumption sets what will be referred to as the Building Base Load (BBL), that is, the minimum level of energy consumption fo the building.

Apart from the daily seasonality, with a period of 24h, and the weekly seasonality, with a period of 7 days, there's a third seasonality in the energy consumption curve in office buildings. With a period of 12 months, the consumption follow a pattern in which it is higher in summer and winter due to location's weather conditions and smoother in autumn and spring. All this seasonality will have an impact in the ML models and must be considered.

Regarding the dynamics might be useful to highlight the values of the top consumption and bottom consumption of the building in the evaluation period. The bottom consumption, value we will assign to the BBL for this study, is 8 kWh. The maximum consumption in this period is, on the other hand, is 152 kWh and was recorded the 12th of September 2016. Given the range between the BBL and the maximum consumption, all the graphs in this report will be reduced to the range 0 to 150 kWh in the ordinates axis to facilitate the visualization.

4.2.4 Anomalies & Data Cleanness

In this section the most important anomalies observed in the dataset are enlisted and detailed. In this study, anomalies refer to deviations in predicted energy consumption that describes consumptions that do not seem reasonable and do not correspond with the energy system dynamics of the building. In the methodology proposed in this master thesis there are six types of characteristic errors, some of them common to other processes and some specific to the energy consumption dynamics in a tertiary office building.

Missing Values

Missing Values (NAs) are the Achilles' Heel of data analysis. Every data-logging process ends up with a higher or lower number of "unregistered" values or *Missing Values*. NAs can be the results of mistakes from various sources, which are many times untraceable. There is a whole science behind *Missing Data Theory* and how to fill in these values. Imputation refers to replacing missing data with a ''best guess'', which can be obtained from modelling the relationship between the missing and the observed data (Perera, 2008). One form of imputation and the one this methodology will follow is "ignore" the values in the analysis. The result is a reduced dataset that has no missing data and can be analyzed by any conventional method. This strategy is commonly known in the social sciences as *listwise deletion* or *casewise deletion*, but also goes by the name of *complete case analysis*. (Allison, 2009). In the dataset analyzed in this dissertation, there have been found NAs both in the real energy consumption and in the forecasted energy consumption.

Energy Consumption missing values

There have been found some NAs for certain hours in the dataset in the real-consumption column. Nevertheless, they represent just 0.29% of the sample and instead of *casewise deletion* they have been filled in with average of energy consumption similar periods. Concretely, there are 4 missed values for 4 hours in a row during the hardware update on Feb 2017 and the 30th of June 2016, for which there is no value at all. This low percentage in NAs in the real-consumption dataset confirms a robust AMI.

Forecast missing values

In the forecasted values data there are more NAs than in the real-consumption data. During the one-year and one month evaluation period there have been found extended periods in which PREDICTION did not work at all, probably due to some technical or connection issue. This kind of error seems to be more technical related than algorithmic related. A total of 4.29% of the sample contains NAs. The cases found are: January 2017, 1st to 12th; October 2016, 28th and 30th; and May 2016, 28th to 30th. An example of this anomaly can be found in the Figure 7. As it has been explained, this NAs are imputed by ignoring them, but for the shake of the comparison analysis the corresponding values of the real-consumption data for the dates of the corresponding instances is set also to NA, so they do not affect in the error metrics and these instances are totally neglected.

Outliers predictions

Like Missing Values Theory, there is a whole science behind the Outlier detection and treatment. The definition given by (Hawkins, 2014) of an outlier yields: "*An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism."* Outliers are also referred to as abnormalities, discordant, deviants, or anomalies in the data mining and statistics literature.(Aggarwal, n.d.).

According to the previous definitions what this dissertation refers to as *anomaly* is also known in a general way as *outlier* in the literature. Therefore, all the deviations explained in this section could fall within this *outlier* definition. Nevertheless, in our analysis the concept *Outlier* will refer solely to those deviations that, given its deviation from the average energy consumption of the building, falls beyond a pre-stablished range.

In these proposed data cleanness process, the range has been set wide. All those predicted values that exceeds by 100% the maximum energy consumption in the evaluation period, that is, 148 kWh, will be labeled as an *Outlier*. The process to deal with these anomalies will be to convert the corresponding forecast and real-consumption values to NAs and thus, leave them out of the accuracy analysis. A 0.21% of the sample have been categorized as *Outliers.* An example is shown in Figure 8, where normal patterns cannot be differentiated due to the scale of the axis.

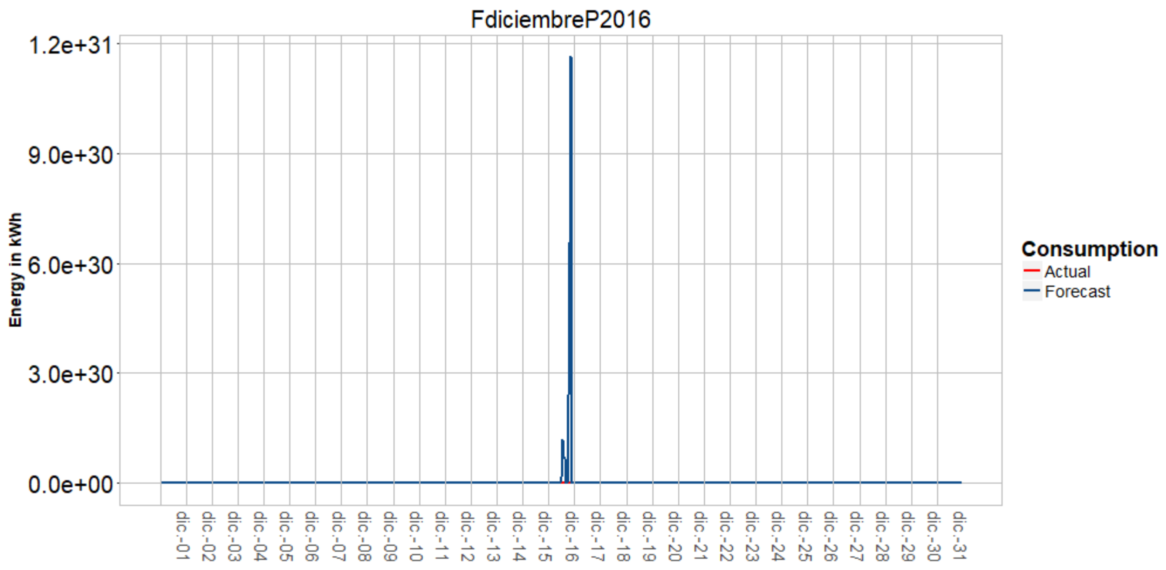


Figure 8. Example of outliers in the PREDICTION output in December 2016.

Zero consumption predictions (ZVs)

It has been observed in the data visualization process that some output values of PREDICTION model yield 0 kWh energy consumption predictions. While NAs do not have to contradict the dynamics of the energy consumption, ZVs has no physical sense when representing a building's energy consumption. Every tertiary building has a minimum energy consumption, what has been already defined as the Building Base Load. There are systems that are always working and consuming energy (e.g. Emergency light systems). Furthermore, as it has been pointed out in section 4.2.3 the minimum consumption recorder in the evaluation period has been 8 kWh and, therefore, it does not seem reasonable that PREDICTION forecasts values lower than that.

The methodology to assess the accuracy of the PREDICTION model will convert this data points into NAs so they are imputed in the same way (by ignoration). Any of the metrics that will be explained in the following sections would be heavily affected if the values that have been predicted as 0 kWh are kept the same. Therefore, they must be removed from the statistical metrics explained in 4.3.1. A total of 1.68% of the sample contains ZVs and will be set to NAs.

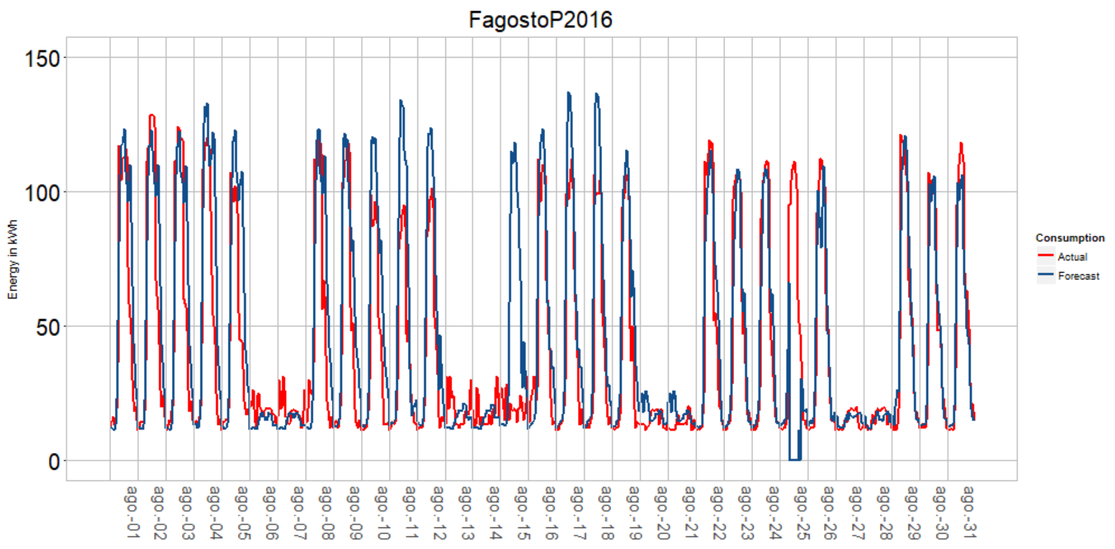


Figure 9. Example for Zero consumption prediction on August 25th, 2016.

Below Building's base load predictions

Another anomaly detected in a first place by data visualization and then added to the code for automatic detecto is predicted values below the BBL. As it was defined before BBL represents the energy consumption of the building when it is empty or almost empty. There is no physical justification to predict values below BBL and, therefore, they are treated as deviation from normal behavior.

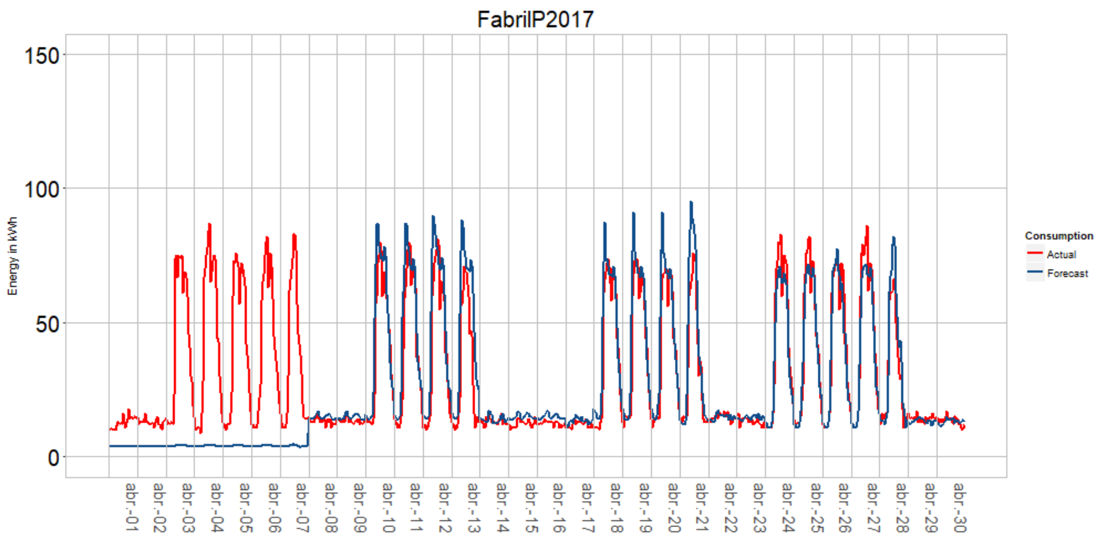


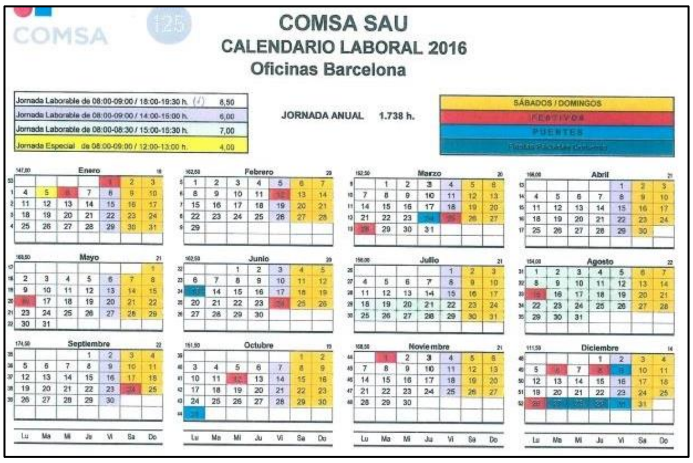
Figure 10. Example of anomaly in the data. Energy forecasted values below the BBL from 1st to 7th April 2016.

The treatment for this kind of deviation from standard results will be similar to the one explained for the ZVs and outlier deviations. First, the forecasted values below the BBL will be converted to NAs. Then, as the rest of the NAs they will be ignored and not considered for the metrics analysis so they do not affect to the accuracy of the model. A total of 4.01% of the sample falls within this category,

Deviations from working calendar

Working calendars define yearly working/holidays dates set by the regional government and company. They include all the weekends and national and regional official holidays. Furthermore, each company defines its own distribution of holidays and schedules. For example, some companies cease their activity the last week of December and some do shorter working days in the summer. All this information should be included for a good energy forecasting as the energy consumption is extremely different one working day than one holiday or weekend as it was explained in 4.2.3.

Due to the location of the building the working calendar of the company that make uses of the building should include all the official holidays for Spanish territory and specifically for the state of Catalunya and the city of Barcelona. Furthermore, it should include special holidays marked by the company and the own policy regarding working days and schedule distribution.



COMSA SAU
CALENDARIO LABORAL 2016
Oficinas Barcelona

Jornada Laborable de 08:00-09:00 / 18:00-19:30 h.	8,50
Jornada Laborable de 08:00-09:00 / 14:00-15:00 h.	6,00
Jornada Laborable de 08:00-08:30 / 15:00-15:30 h.	7,00
Jornada Especial de 08:00-09:00 / 12:00-13:00 h.	4,00

JORNADA ANUAL 1.738 h.

🟧 SÁBADOS / DOMINGOS
🟥 FESTIVOS
🟦 PUENTES
🟦 [illegible]

147,00 Enero 18

	Lu	Ma	Mi	Ju	Vi	Sa	Do
53					1	2	3
1	4	5	6	7	8	9	10
2	11	12	13	14	15	16	17
3	18	19	20	21	22	23	24
4	25	26	27	28	29	30	31

162,50 Febrero 20

	Lu	Ma	Mi	Ju	Vi	Sa	Do
5	1	2	3	4	5	6	7
6	8	9	10	11	12	13	14
7	15	16	17	18	19	20	21
8	22	23	24	25	26	27	28
9	29						

152,50 Marzo 20

	Lu	Ma	Mi	Ju	Vi	Sa	Do
9		1	2	3	4	5	6
10	7	8	9	10	11	12	13
11	14	15	16	17	18	19	20
12	21	22	23	24	25	26	27
13	28	29	30	31			

168,00 Abril 21

	Lu	Ma	Mi	Ju	Vi	Sa	Do
13					1	2	3
14	4	5	6	7	8	9	10
15	11	12	13	14	15	16	17
16	18	19	20	21	22	23	24
17	25	26	27	28	29	30	

168,50 Mayo 21

	Lu	Ma	Mi	Ju	Vi	Sa	Do
17							1
18	2	3	4	5	6	7	8
19	9	10	11	12	13	14	15
20	16	17	18	19	20	21	22
21	23	24	25	26	27	28	29
22	30	31					

162,50 Junio 20

	Lu	Ma	Mi	Ju	Vi	Sa	Do
22			1	2	3	4	5
23	6	7	8	9	10	11	12
24	13	14	15	16	17	18	19
25	20	21	22	23	24	25	26
26	27	28	29	30			

156,00 Julio 21

	Lu	Ma	Mi	Ju	Vi	Sa	Do
26					1	2	3
27	4	5	6	7	8	9	10
28	11	12	13	14	15	16	17
29	18	19	20	21	22	23	24
30	25	26	27	28	29	30	31

154,00 Agosto 22

	Lu	Ma	Mi	Ju	Vi	Sa	Do
31	1	2	3	4	5	6	7
32	8	9	10	11	12	13	14
33	15	16	17	18	19	20	21
34	22	23	24	25	26	27	28
35	29	30	31				

174,50 Septiembre 22

	Lu	Ma	Mi	Ju	Vi	Sa	Do
35				1	2	3	4
36	5	6	7	8	9	10	11
37	12	13	14	15	16	17	18
38	19	20	21	22	23	24	25
39	26	27	28	29	30		

151,50 Octubre 19

	Lu	Ma	Mi	Ju	Vi	Sa	Do
39						1	2
40	3	4	5	6	7	8	9
41	10	11	12	13	14	15	16
42	17	18	19	20	21	22	23
43	24	25	26	27	28	29	30
44	31						

168,50 Noviembre 21

	Lu	Ma	Mi	Ju	Vi	Sa	Do
44		1	2	3	4	5	6
45	7	8	9	10	11	12	13
46	14	15	16	17	18	19	20
47	21	22	23	24	25	26	27
48	28	29	30				

111,50 Diciembre 14

	Lu	Ma	Mi	Ju	Vi	Sa	Do
48				1	2	3	4
49	5	6	7	8	9	10	11
50	12	13	14	15	16	17	18
51	19	20	21	22	23	24	25
52	26	27	28	29	30	31	

Figure 11. Working Calendar of the company for 2016.

In the dataset studied there haven some errors observed regarding the working day/holidays distribution. In Figure 7 it was already mentioned that the 16th of May the model predicted a normal working day as it was Monday but, as it can be pointed out in the official working calendar in Figure 11, it as holiday and the building had no activity that could led to energy consumption. There are cases in which the opposite occurs and no activity is predicted as if the model considered a specific day holiday but it was a normal working day and, therefore, there was energy consumption over the BBL. An example of this deviation has been detected the 17th of October 2016 and it is shown in Figure 12.

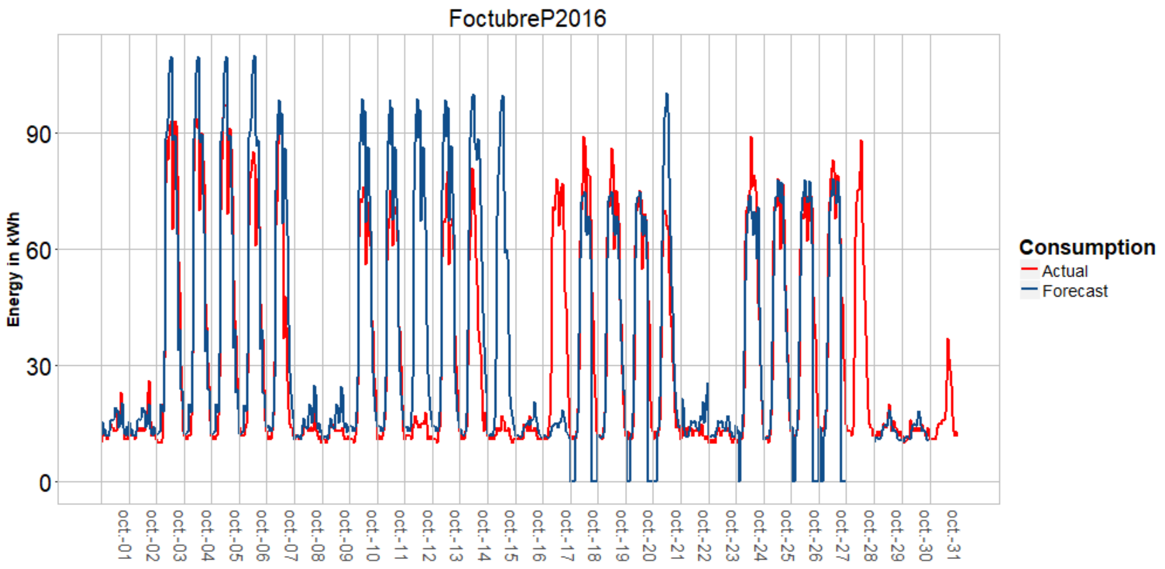


Figure 12. PREDICTION forecast 17th of October 2016 as holiday or weekend

Strange dynamics behavior

Another detected anomaly showing strange behavior have been identified in the outputs of the PREDICTION model. For example, sudden changes in the energy consumption were predicted in November 2016 (See Figure 13). This behavior violates the dynamics of energy consumption in buildings as these usually follow smooth variations and do not have such sudden changes.

Another example of strange dynamics behavior that can be observed also in the November graph is identified by the presence of negative predictions. Although the building is equipped with solar PV panels that could generate energy and this fact could be represented as *negative* Energy consumption (given the PV generated more than the energy consumption and the excess would be sold to the grid), it is not the case and the predictions is probably a mistake in the algorithm. As the renewable energy is not being considered and we are considering the WBE, it is physically impossible to have negative energy consumptions. This non-justifiable behave of the model output that should be corrected (i.e. imposing a condition such as Output > 0).

The total amount of this strange deviations accounts for 0.60% of the sample and will be set to NAs and deleted from the analysis dataset.

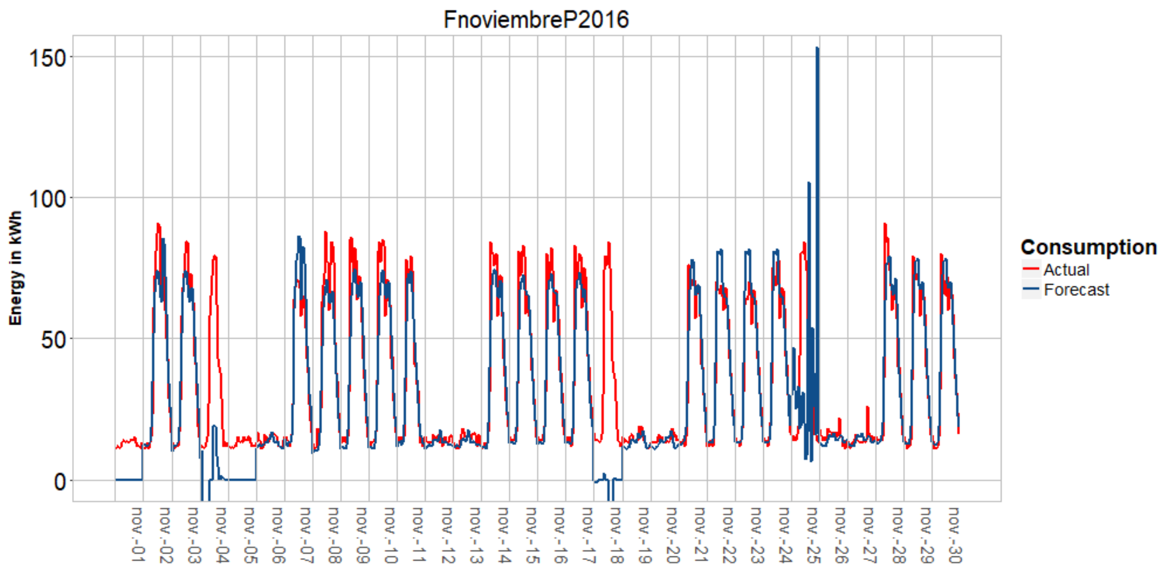


Figure 13. Strange Energy Consumption behavior predicted in November 2016

Other considerations

In the literature, as it has been already mentioned, anomalies or *outliers* in data analysis refers to observations that differs from other observations so much that arouse suspicious about its nature. In many cases, it is responsibility of the analyst to set the boundaries on what would be a suspicious instance suitable to be categorized as an anomaly. The analyst should, based on his understanding about the processes that the data is representing and his experience, determine if one instance should or should not be included in the anomaly group. In this master thesis, the *anomalies detection* has not been as thorough as to detect smooth anomalies in the energy consumption data. Therefore, in the final dataset that will be used for the analysis, there might remain some small anomalies in the energy consumption of the building that could slightly affect the accuracy. Bacause of the frequency and magnitude of this anomalies, they have been assumed to have weak influence in the results and it has been decided to not deal with them.

As an example of this kind of anomalies Figure 14 shows the energy consumption of the 4th floor of the COMSA Headquarters two weeks of July 2017. During this period, it can be shown how on the Saturday 8th and on the Sunday 16th there is energy consumption. This consumption is due to the student that decided to go to his desk on weekend to progress on the work on this dissertation. This example illustrates a stochastic event that would be impossible to predict and will affect the metrics. Nevertheless, its impact will be neglectable in this case. This kind of errors should never be attributed to the model itself but to the raw input data.

For a more thorough analysis on the anomalies there are existing algorithms in the literature. For example, (Fan, Xiao and Wang, 2014) apply Generalized extreme studentized deviate (GESD) to their dataset to identify abnormalities in the building energy consumption.

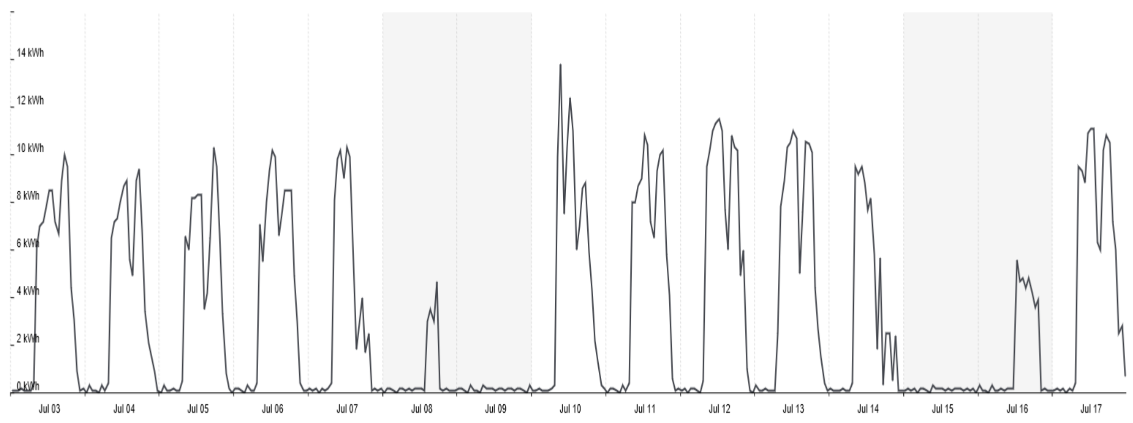


Figure 14. Stochastics events in the energy consumption. Source: DexCell

Anomalies Summary

All the above-mentioned deviations are gathered in Table 1 and its main characteristics are given. Adding all of the instances that have being categorized as an anomaly they represent 12% of all the instances contained in the data set. These corrupted instances are converted to NAs and therefore, they will be ignored in the final dataset. From 9503 instances, 1141 are discarded and 8362 will configure the dataset that will be used in the statistical analysis.

The value of *missing data* in the original datasets accounts for a total of a 12%. It means that, for one year and disregarding its accuracy, 1051 hours (put of 8760) the model is not given any output (or a totally corrupted one). This implies that the grade of *reliability* is considerably low and that corrective measures should be implemented to increase it. According to the company interest in the forecast, any value of NAs or corrupted data should remain below 5% to be acceptable.

As there is no information on how PREDICTION models work, it is difficult to assess and find reason why these anomalies are happening. Some of them will be easily corrigible while others might show more difficulties. For example, direct found NAs instances are probably due to technical fails with the hardware involved in the systems, errors in the communications or in the data servers can lead to fill an instance with an NA. A reviewed and updated hardware and software infrastructure will probably reduce the % of this type of anomaly. Another deviation that could be effortless corrected is the one related to the working calendar. An updated and adequate calendar will automatically troubleshoot this deviations for the better. As it has been mentioned it should include all national, regional and local holidays as well as the specific companies' policies regarding schedule and holidays. It has been observed in the dataset that in 2017 there is no deviation like this, probably due to an update in the calendar which the forecast model is working with.

Table 1. Anomalies detection summary

Type of Anomaly	**Explanation**	**% of sample**	**Corrective measure**
Missed Values in Energy Consumption	Metering infrastructure fails to measure and record building energy consumption	0.29	Instances filled in with averaged similar data.
Missed Values in Energy forecasts	Forecast application fails and no output is given by the prediction model	4.29	Instances Ignored and deleted from dataset.
Zero Value building predictions (ZVs)	Forecast predicts that WBE will be 0 kWh.	1.68	Instances are converted to Missed Values and then Ignored and deleted from data set.
Below BBL predictions	Forecast predicts that building energy consumption will be lower than the BBL.	4.01	Instances are converted to Missed Values and then Ignored and deleted from data set.
Outliers	Forecast predicts too high or too low WBE consumption.	0.21	Instances are converted to Missed Values and then Ignored and deleted from data set.
Working Calendar	Forecast missed to predict correctly a holiday.	1.26	Instances are converted to Missed Values and then Ignored and deleted from data set.
Strange Behavior	Forecasts predicts WBE Consumptions that have no physical logic.	0.60	Instances are converted to Missed Values and then Ignored and deleted from data set.
TOTAL	Percentage of the sample that contains anomalies.	**12.06**	Instances ignored in the error metrics.

4.3 Data Analysis and Error assessment

After completing the data cleanness process explained in the previous section the dataset is ready to continue with the model statistical accuracy analysis. First, the statistical metrics that will be used are explained. Second, they will be applied to the dataset and the results obtained will be interpreted and discussed.

4.3.1 Statistical Error metrics

In this section the statistical metrics on which the accuracy will be assessed are explained and their equations are shown. They have been chosen by carefully studying the literature in the subject and selecting the most common and suitable parameters. This way, it will be possible to better understand how well PREDICTION model is working and compare it with other similar dissertations in the literature.

It is convenient to highlight again the importance of constantly keep track of the dynamics of the processes associated to the data that is being analyzed. In our case, there are some advantages that will make the analysis easier. The dynamics of energy consumption in a tertiary building are constrained within an interval. This interval is set by the minimum, what we have defined as the BBL, and a maximum, which would be the maximum power consumption of a building given all the devices were turned on at the same. In our building, these values have historically oscillated between 5-8 kWh to 150-200 kW. Thanks to the data cleanness process, we have ensured that all the predictions remain within this range.

Regarding evaluation methodologies for forecasting models, it is also common to use a “naïve” method, which provides a benchmark. A set of relative error metrics could be calculated to compare a determined model to an accepted reference in the subject. This study does not consider any reference error and will sticks to absolute errors calculations.

Mean Absolute Percentage Error (MAPE)

The error metric that is most widely used in the literature is *Mean Absolute Percentage Error (MAPE)* due to its advantages of scale-independency and interpretability (Kim and Kim, 2016). It is defined by the equation Ec. 1.

$$MAPE = \frac{1}{N} * \sum_{i=1}^{N} \frac{|pred(i) - real(i)|}{|real(i)|} * 100 \qquad [Ec\ 1]$$

Where:

- N refers to the number of data points
- pred(i): Value of the Energy Demand Forecast for the instance *i*
- *real(i): Value of the real energy consumption for the instance i*

The main disadvantage of this metric is that it could yield infinite or undefined values when the actual values are zero or close to zero (Kim and Kim, 2016). As for the dynamics of the process this will never be the case (due to the BBL) it has been chosen as the main estimator of the accuracy for energy demand forecast in tertiary buildings.

Root Mean Square Error (RMSE)

The Root Mean Square Error (RMSE) and Mean Square Error (MSE) are also widely used due to their theoretical relevance in statistical modeling (Hyndman and Koehler, 2006). This study will focus just in the RMSE. Contrary to MAPE, this metric is scale-dependent and it results in values with the same units of the measurements (Fan, Xiao and Wang, 2014). It is sensitive to outliers but as we have put the data through a thorough data cleanness process there will not be any outlier that can bias this metric.

$$RMSE = \sqrt{\frac{1}{N} * \sum_{i=1}^{N} (pred(i) - real(i))^2} \qquad [Ec\ 2]$$

Where:

- N refers to the number of data points
- pred(i): Value of the Energy Demand Forecast for the instance *i*
- *real(i): Value of the real energy consumption for the instance i*

RMSE is commonly used for describing uncertainty. It is function of the Mean Absolute Error and the distribution of error magnitudes. It penalizes the larger error terms. There is controversy in which error metrics is better between MAPE and RMSE (Yildiz, Bilbao and Sproul, 2017) In this dissertation both of them will be calculated.

Mean Biased Percentage Error (MBPE)

This study aims also to detect any systematic data in the model and Mean Biased Error is a common and suitable method for this purpose. Like MAPE, MBPE it is not dimensional. It is a measure of the overall bias error. This metric establishes how likely a model is to over-estimate or under-estimate the actual energy consumption (Edwards, New and Parker, 2012). Consequently, the closer this value is to 0 the less “biased” the model is, meaning that the model does not favor any trend in its prediction.

$$MBE = \frac{1}{N} * \sum_{i=1}^{N} \frac{(real(i) - pred(i))}{real(i)} * 100 \qquad [Ec\ 3]$$

Where:

- N refers to the number of data points
- pred(i): Value of the Energy Demand Forecast for the instance *i*
- *real(i): Value of the real energy consumption for the instance i*

Coefficient of Determination (R^2)

The statistic that is used as a measure of goodness of fit is the adjusted coefficient of determination, denoted by R^2 adj (Hyde and Hodnett, 1997).

$$R^2_{adj} = 1 - (1 - R^2)(\frac{n-1}{n-k}) \qquad [Ec\ 4]$$

Where:

- *n* refers to the number of data points
- *k* is the number of predictors
- *R^2* is the coefficient of determination

In data analysis where the number of predictors is high with respect to the number of data points it makes sense to refer to the adjusted coefficient of determination. However, in our case the number of predictors will always remain low (never more than 20) compared to the number of data points (thousands). Consequently, in this study it will be calculate just the coefficient of determination, with no adjustment made.

$$R^2 = 1 - \frac{SSE}{SST} \qquad [Ec\ 5]$$

Where:

- SSE: Sum of Squared errors.

$$SSE = \sum_{i=1}^{N} (\widehat{pred(\imath)} - real(i))^2 \qquad [Ec\ 6]$$

Where:

$\widehat{pred(\iota)} \rightarrow$ output vale of instance i
$real(i) \rightarrow$ real consumption

- SST: Regression Identity.

$$SST = SSE + SSR \qquad [Ec\ 7]$$

- SSR: Sum of squared regression. Variation in the predicted values.

$$SSR = \sum_{i=1}^{N} (\widehat{pred(\iota)} - \overline{pred})^2 \qquad [Ec\ 8]$$

Where:

$\widehat{pred(\iota)} \rightarrow$ output vale of instance i
$\overline{pred} \rightarrow$ output mean

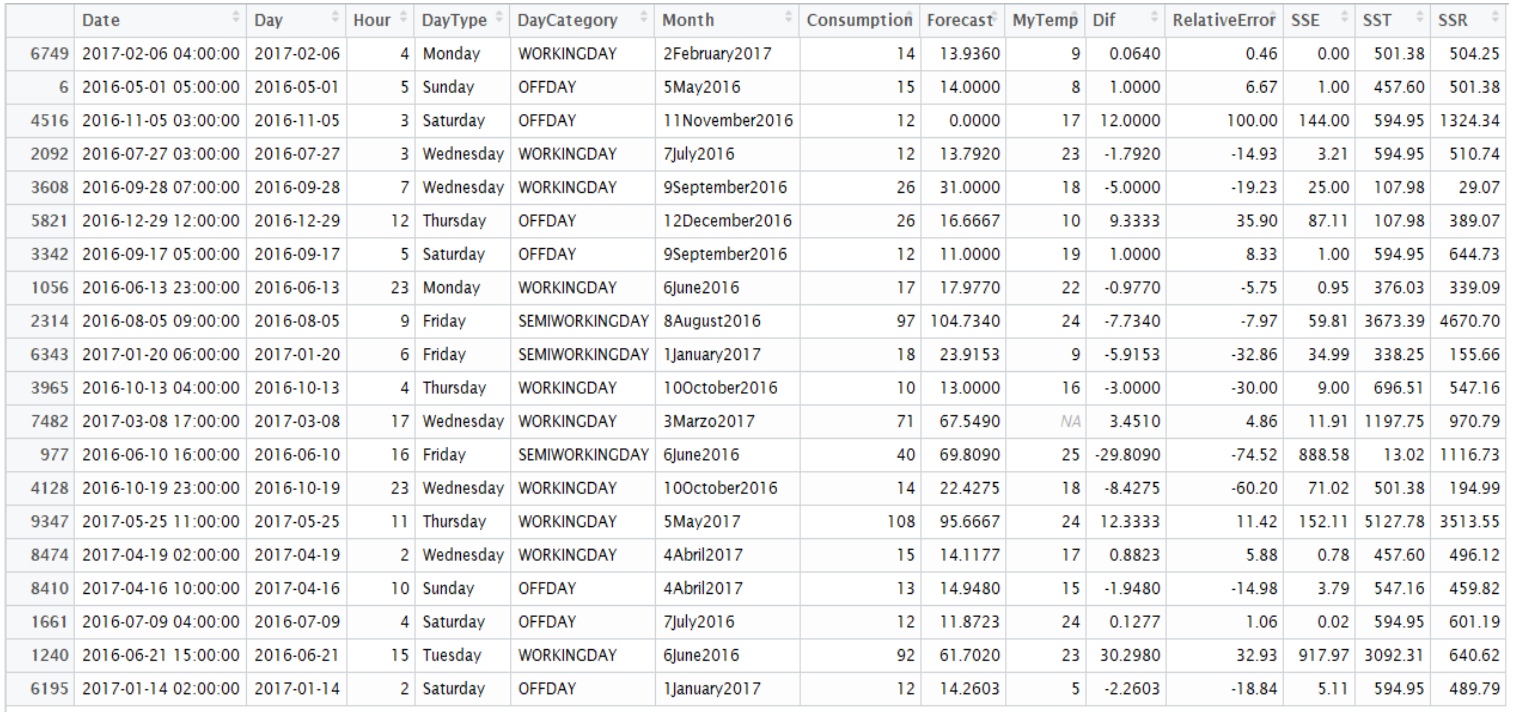
The coefficient of determination will measure the wellness of the fit by the trained model.

4.3.2 **Dataset evaluation**

A random 20 instances sample of the dataset over which the metrics are going to be calculated is shown in Figure 15. In order to accomplish the data-cleanness process some new features have been created with respect to the data frame shown in Figure 6. Furthermore, other attributes have been added for other purposes out of the scope of this first statistical analysis.

Features as *DayType or DayCategory* were used to find the deviations in the instances that were wrong categorized. *Month* attribute was used to break down and name the graphs shown throughout the document. *MyTemp* attribute has been added for a future analysis on the influence of the weather over the performance of the model, this will be further explained in the section 5.

Dif feature is simply the difference between the predicted value and the real consumption for a given timestamp. *RelativeError (or BPE)* is the percentage the prediction has deviated from the actual consumption. If the algorithm has predictd a lower consumption than the actual consumption this number will be positive. Therefore, if a BPE is negative it means that for that instances, the algorithm has overpredicted the consumption, giving a higher value than the real one. It will be used to assess if the model is overal biased through the MBPE. *SST, SSE* and *SSR* represents the parameters that are needed for calculating the coefficient of determination, as explained in 4.3.1.



	Date	Day	Hour	DayType	DayCategory	Month	Consumption	Forecast	MyTemp	Dif	RelativeError	SSE	SST	SSR
6749	2017-02-06 04:00:00	2017-02-06	4	Monday	WORKINGDAY	2February2017	14	13.9360	9	0.0640	0.46	0.00	501.38	504.25
6	2016-05-01 05:00:00	2016-05-01	5	Sunday	OFFDAY	5May2016	15	14.0000	8	1.0000	6.67	1.00	457.60	501.38
4516	2016-11-05 03:00:00	2016-11-05	3	Saturday	OFFDAY	11November2016	12	0.0000	17	12.0000	100.00	144.00	594.95	1324.34
2092	2016-07-27 03:00:00	2016-07-27	3	Wednesday	WORKINGDAY	7July2016	12	13.7920	23	-1.7920	-14.93	3.21	594.95	510.74
3608	2016-09-28 07:00:00	2016-09-28	7	Wednesday	WORKINGDAY	9September2016	26	31.0000	18	-5.0000	-19.23	25.00	107.98	29.07
5821	2016-12-29 12:00:00	2016-12-29	12	Thursday	OFFDAY	12December2016	26	16.6667	10	9.3333	35.90	87.11	107.98	389.07
3342	2016-09-17 05:00:00	2016-09-17	5	Saturday	OFFDAY	9September2016	12	11.0000	19	1.0000	8.33	1.00	594.95	644.73
1056	2016-06-13 23:00:00	2016-06-13	23	Monday	WORKINGDAY	6June2016	17	17.9770	22	-0.9770	-5.75	0.95	376.03	339.09
2314	2016-08-05 09:00:00	2016-08-05	9	Friday	SEMIWORKINGDAY	8August2016	97	104.7340	24	-7.7340	-7.97	59.81	3673.39	4670.70
6343	2017-01-20 06:00:00	2017-01-20	6	Friday	SEMIWORKINGDAY	1January2017	18	23.9153	9	-5.9153	-32.86	34.99	338.25	155.66
3965	2016-10-13 04:00:00	2016-10-13	4	Thursday	WORKINGDAY	10October2016	10	13.0000	16	-3.0000	-30.00	9.00	696.51	547.16
7482	2017-03-08 17:00:00	2017-03-08	17	Wednesday	WORKINGDAY	3Marzo2017	71	67.5490	NA	3.4510	4.86	11.91	1197.75	970.79
977	2016-06-10 16:00:00	2016-06-10	16	Friday	SEMIWORKINGDAY	6June2016	40	69.8090	25	-29.8090	-74.52	888.58	13.02	1116.73
4128	2016-10-19 23:00:00	2016-10-19	23	Wednesday	WORKINGDAY	10October2016	14	22.4275	18	-8.4275	-60.20	71.02	501.38	194.99
9347	2017-05-25 11:00:00	2017-05-25	11	Thursday	WORKINGDAY	5May2017	108	95.6667	24	12.3333	11.42	152.11	5127.78	3513.55
8474	2017-04-19 02:00:00	2017-04-19	2	Wednesday	WORKINGDAY	4Abril2017	15	14.1177	17	0.8823	5.88	0.78	457.60	496.12
8410	2017-04-16 10:00:00	2017-04-16	10	Sunday	OFFDAY	4Abril2017	13	14.9480	15	-1.9480	-14.98	3.79	547.16	459.82
1661	2016-07-09 04:00:00	2016-07-09	4	Saturday	OFFDAY	7July2016	12	11.8723	24	0.1277	1.06	0.02	594.95	601.19
1240	2016-06-21 15:00:00	2016-06-21	15	Tuesday	WORKINGDAY	6June2016	92	61.7020	23	30.2980	32.93	917.97	3092.31	640.62
6195	2017-01-14 02:00:00	2017-01-14	2	Saturday	OFFDAY	1January2017	12	14.2603	5	-2.2603	-18.84	5.11	594.95	489.79

Figure 15. Random Sample of 20 instances from the dataset after the data cleanness.

4.3.3 Results and discussions Part I

In this section the accuracy of the forecast model will be assessed. The dataset has been prepared for its statistical analysis and four different error metrics have been defined. Using the statistical R programming language, as explained in 3.6, the metrics are applied to the dataset (See Figure 15) to assess how well the predictive model is forecasting the energy consumption in the study-case building. To understand the importance of the data preprocess the results will display all the metrics after applying each of the deleting each of the abnormalities categories explained in 4.2.4.

Table 2. Error metrics after each step of the data cleanness process

	MAE (kWh)	MAPE (%)	MBPE (%)	RMSE (kWh)	R^2	NAs (%)
Error metrics calculated directly from the **original dataset.**	>1e+20	>1e+20	>1e+20	>1e+20	>1e+20	4,29
After cleaning the biggest **outliers** (those which are higher than 200% of the maximum energy consumption in the evaluation period).	8,45	24,95	-1,68	15,86	0,78	4,5
Once removed the energy consumptions predicted as 0 kWh **(ZVs).**	8,09	23,6	-3,51	15,17	0,8	6,19
Afterwards instances containing Energy predictions **BBL** are removed.	7,13	21,03	-7,28	13,17	0,84	10,19
Subsequently instances with errors in the **working calendar** are removed.	6,76	18,36	-4,48	12,14	0,86	11,45
Last step in the data cleanness process is to remove **strange behaviors** that do not respect the energy dynamics.	6,58	18,02	-4,67	11,67	0,87	12,05

Table 2 shows what have been mentioned throughout the whole document: the importance of keeping track of the dynamics of the systems being analyzed. If the accuracy analysis were carried out directly to the dataset, the results would be the ones in the first row, that is, with a 4,29% of missed values and values over 10^{20} in all the other metrics. This is due to some irrational predictions that forecasted building energy consumptions thousands of times higher than the total world energy consumption (predictions over 10^{24} GWh). As it was explained in the MAPE defiition, this outliers have a huge impact in this metrics. Once these astronomic values are removed, a more logical dataset remains and the metrics can be reasonably analyzed as they yield within acceptable and interpretable ranges. Nevertheless, the dataset still includes all the others anomalies that represents fails in the model and should be considered before giving insights on how accurate it is predicting the energy consumption.

As the data cleansing process progresses it can be seen in the metrics how the number of misseing values, or NAs, is increasing while the model improves its accuracy. Every step leaves out a certain number of instances that will account as timestamps in which the model is considered to not work at all. This parameter can be interpreted as the *reliability* or *stability* of the model, that is, how many time during the year it works. Given these assumptions, the results show that during the evaluation period the model did not work 12.05% of the time or, what is the same, 1145 out of 9503 hours.

Extracting the values from the last row in Table 2, that is, those metrics obtained once the data pre-process have been completed, the conclusive results for the model accuracy are summarized in Table 3.

Table 3. Conclusive results for the model accuracy

MAE (kWh)	6,58
MAPE (%)	18,02
MBPE (%)	-4,67
RMSE (kWh)	11,67
R^2	0,87
NAs (%)	12,05

All the errors improve towards their final values, once the dataset is ready for the statistical analysis. At this point, the accuracy can be measured more legitimately. The main metric in this study, the Mean Absolute Percentage Error, defined in 4.3.1, results in a value of 18.02%. It means that, on average, the model deviates around 18% from the real consumption. The results are not satisfactory as the COMPANY has set a goal of having a MAPE below 10%. Comparing to some results in the literature this value is still high, although some researcher has obtained similar values using their models. (Yildiz, Bilbao and Sproul, 2017) obtained values of 15.43% for a single building using single regression models, although it improved this results by using multiple regression models up to around 7%. (Edwards, New and Parker, 2012) show results for independent house buildings of MAPEs ranging from 16% to 21.33%. On the other hand, (Fan, Xiao and Wang, 2014) show results in cases which different models resulted in MAPEs below 5%.

MAPE values for disaggregated loads, as the one it is considered in this document, are usually higher than for aggregated loads as grids, microgrids or building clusters. The time frame and time horizon studied also have strong influence on forecast accuracies. The longer the time frame for which the predictions are being made, the more likely is to achieve lower MAPE values. For example, (Hernandez et al., 2014) shows in their survey on Electric Power Demand Forecasting, cases in which values of MAPEs below 2% when the aim is to predict monthly energy consumption over a big region. More examples and MAPE results for different models, application areas and time horizons can be found in his survey.

Regarding the Mean Biased Percentage Error (MBPE), its negative value means that the model is biased and, on average, predicts more energy consumption than what the real-consumption finally is. The closes this value is to 0, the better. A value of 0 would mean that the model, despite sometimes predicting higher consumptions and others lower consumptions, on the long term it does not have a *built-in* deviation. This metric is less common in the literature and, therefore, harder to compare. Nevertheless, (Yildiz, Bilbao and Sproul, 2017) shows similar absolute values for this metrics in the study of an University building in Wales. In their study, divided into seasons, the MBE for ranges from -4.54% in Autumn to 5.17% in Summer.

The other two metrics, that is RMSE and R^2, are harder to understand and interpret, specially when just small pieces of information of what the model is doing is avaialble. Both have been calculated in this study to facilitate and to enable the comparison for future works in the subject. Their values fall within reasonable ranges that fits with what have been found in the references. More about this metrics will be commented when how the model builds the predicionts, in the models analzyed in the Part II of the dissertation,

Influence of particular variables in the model accuracy

The values given in Table 3.

Table 3 refers to all the instances in the pre-processed dataset afther the pre-process. It does not consider any particularity or category. There are many parameters of interest that can shed light on when the model's accuracy is higher or lower and what can have stronger influence on it. In this section, representations will be shown to specifically study how well the model works regarding the day of the week, the day category (Offday, workday or special schedule day), the season and the hour of the day.

All of visualizations are based on the box-and-whisker plot (BOXPLOT) representation, introduced by statistician John W. Tukey 1977 (Tukey, 1977). This kind of representation comprises many statistical information in one single graph. A diagram explaining all the parameters represented is shown in Figure 16.

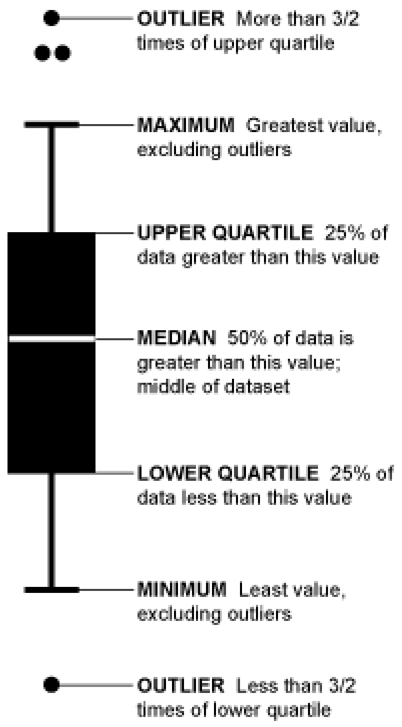

OUTLIER More than 3/2 times of upper quartile
MAXIMUM Greatest value, excluding outliers
UPPER QUARTILE 25% of data greater than this value
MEDIAN 50% of data is greater than this value; middle of dataset
LOWER QUARTILE 25% of data less than this value
MINIMUM Least value, excluding outliers
OUTLIER Less than 3/2 times of lower quartile

Figure 16. Box-and-whisker diagram Plot Explanation (Yau, 2017).

In *Figure 17* the Absolute Percentage Error and Mean Absolute Percentage Error have been represented for each day of the week. From Monday to Wednesday it is almost constant. In Thursdays the MAPE is slightly higher where as in Friday is notably higher. Normally the three first day of the week are constant regarding work activity while Thursday and Fridays have more variability. Specially Fridays, when the schedule is modified and many employees takes this day as holidays to enjoy a long weekend out of the office. On weekends the MAPE is considerably lower than the rest of the days of the week and lower than the overall value. As the weekends the office usually remains off the variability in the energy consumption is lower (around the BBL) and the model tends to have better energy consumption predictions.

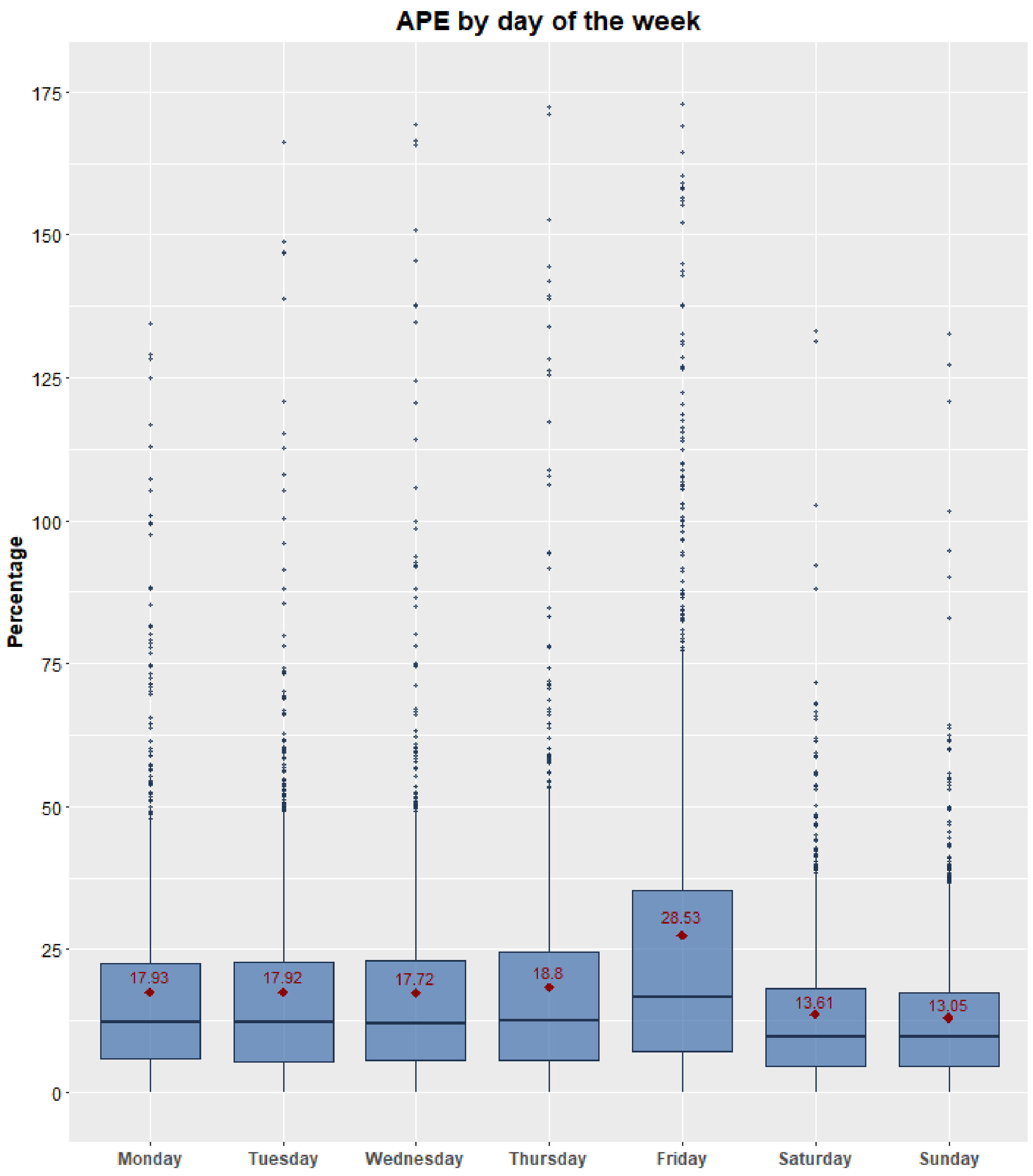


Figure 17. APE and MAPE (red diamond) by day of the week.

The previous analysis is confirmed when looking to Figure 18. Monday to Wednesday the model behaves similarly. It is biased towards predicting more energy consumption that the actual one, as it was already derived from the overall MBPE shown in Table *3.*

Table *3*. The same happens but in a higher quantity on Thursday and Fridays. As there are probably less workers these days and the working activity decreases comparing it to the early days of the week, the algorithm overpredicts the energy consumptions. Surprisingly, the model is more biased on weekends than at the beginning of the week. The most typical deviation for a normal weekend from the normal energy consumption is the one explained in *Figure 14*. If an employee goes to the office on a weekend usually the energy consumption will be higher than expected and thus, the BPE of the instances where he is in the office, lower. If a auto eggresive parameter (e.g.the load 1 or 2 days before at the same time) is used as an inputo for the model and has high influence in the model it could affect the predictions on weekends.This variables consider the activity and values recorded during the week and might overpredict the energy consumption in Saturdays and Sundays, when the activity drastically decreases.

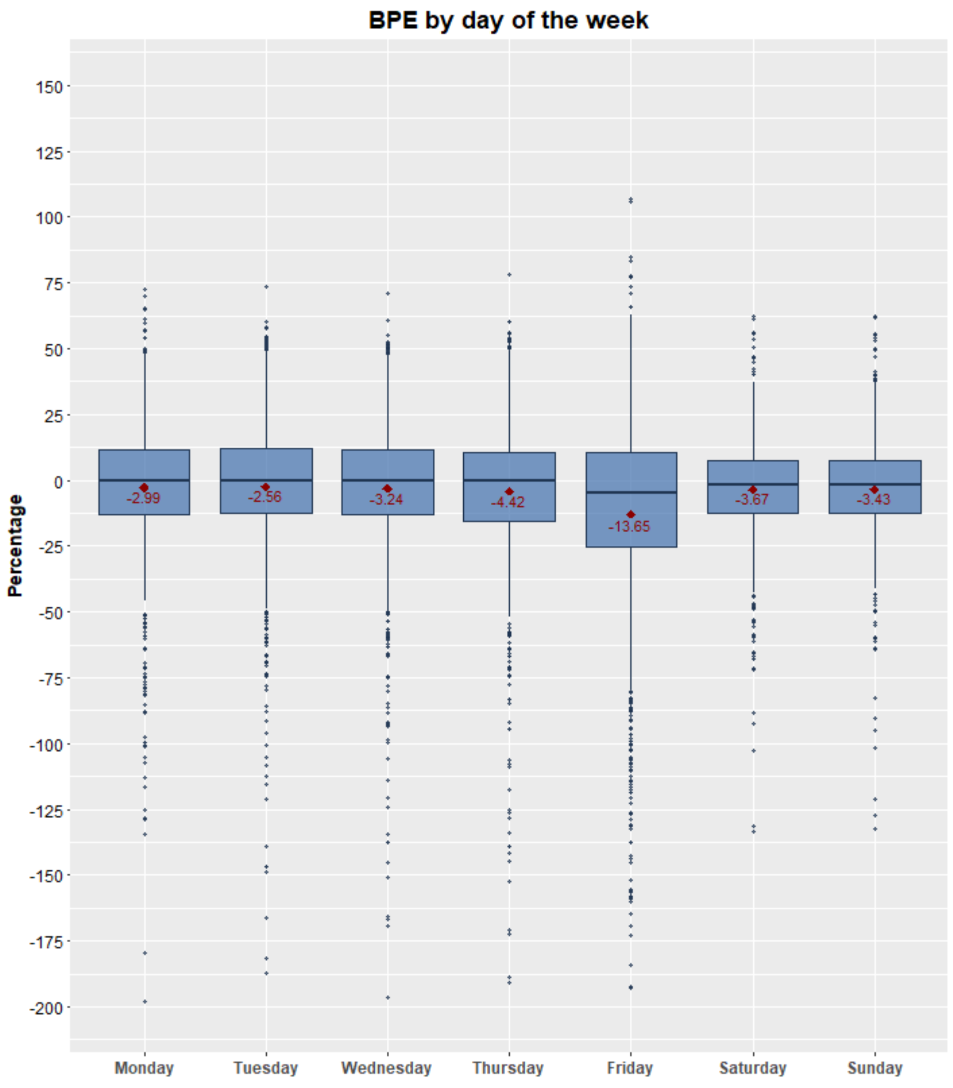


Figure 18. BPE and MBPE (red diamond) by day of the week.

Regarding the day category, in this study there three different classes: WORKINGDAY for days with normal schedule (09h to 19h approx.), SEMIWORKINGDAY for those days with special schedule (08h to 15h approx.) and HOLIDAYS (or OFFDAYs) for weekends and official holidays. On the first one, the metrics are similar to the overall results. Nevertheless, on SEMIWORKING days, that is, Fridays, days before official holidays and a whole month in summer, the model miss predictions by much large values. This difference is explained with the variability associated to these days, where employees have more freedom regarding the schedule and the activity is lower than normal working days. HOLIDAYS behaviors reflect similar numbers than the ones from Saturdays and Sundays as it was expected. It is likely that PREDICTION model is not using the same labelling category thant we have used or at least not with the same calendar. If a day that is "labelled" by Semiworking day it is a normal working day, the MAPE will be high as the working activity in the office differs from one day no another. Information on how the model has labeled the days tegarding this issue should be known to better understand what this metrics are represnting.

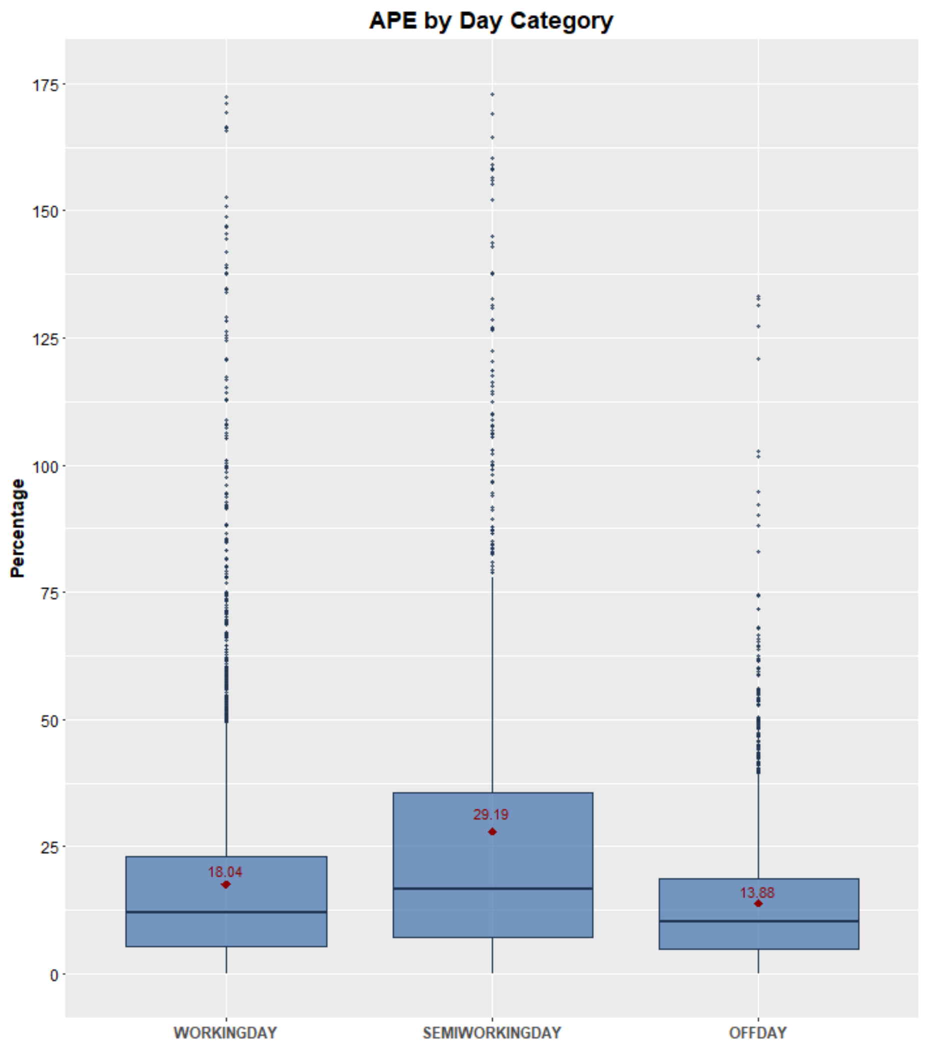


Figure 19. APE and MAPE (red diamond) by day category.

Figure 20 shows once again that the model is biased. Especially those SEMIWORKING days in which the activity usually decreases. On OFFDAYs the model is less biased than the rest of classes. Comparing its -2.58 % MBPE to the one given in Figure 18 for Saturdays and weekends, at around -3.5 % it means than during holidays days in the middle of the week the algorithm is doing better than on weekends (which are including under the OFFDAY category). The effect of this isolated holidays is positive to the overall category metric.

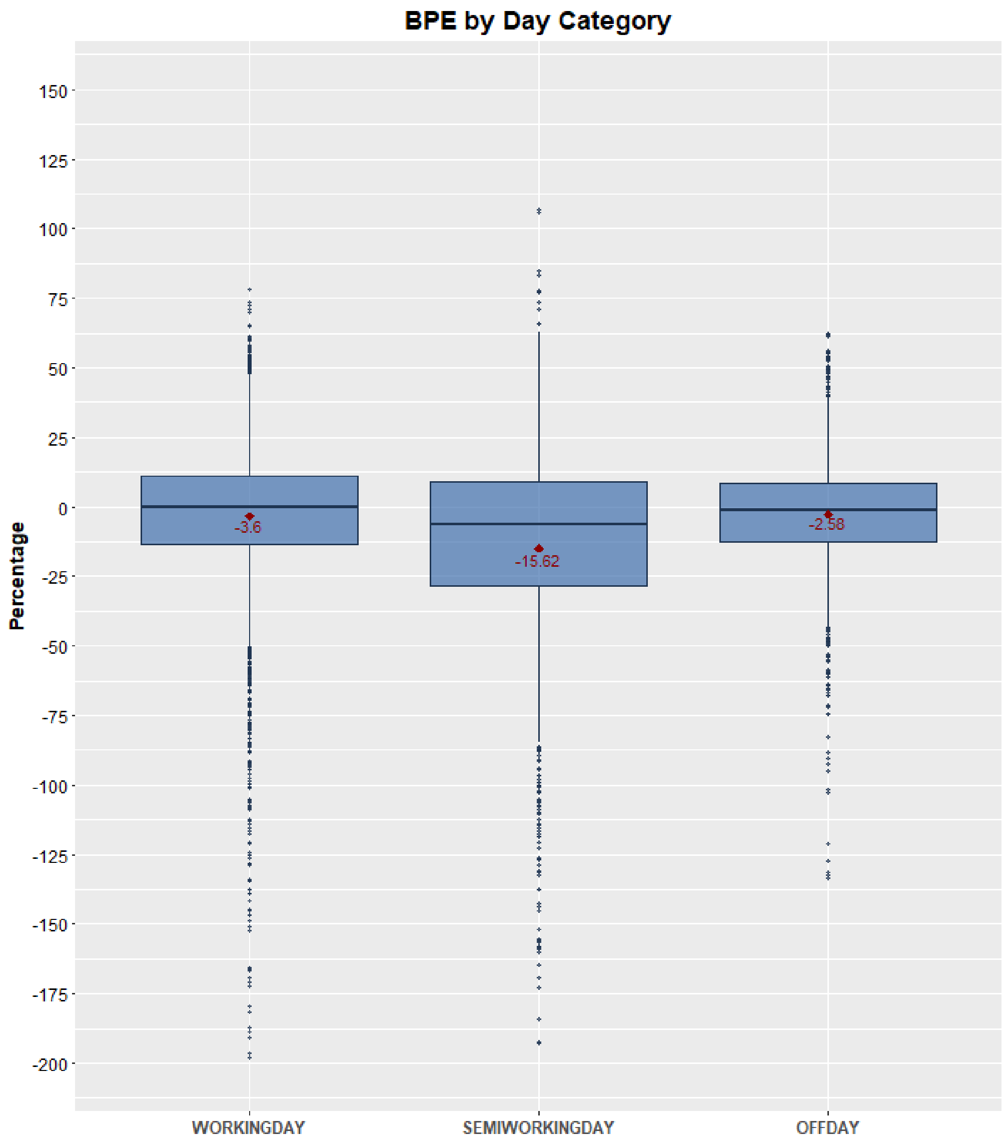


Figure 20. BPE and MBPE (red diamond) by day category.

Another parameter that can lead to important insights about the behavior of the model is its accuracy according to the annual season. Figure 21 shows the APE and MAPE in each of the four seasons. While Spring, Autumn and Winter behaves similarly, in summer the MAPE drastically increases. This result aligns with what it has already been discussed for the day of the week and the day category. In August the company sets the special schedule and all of the week days are categorized as SEMIWORKINGDAYS. In this days, activity in the building tends to be lower and more stochastic, as employees organizes themselves differently from the rest of the year. It is likely that the model does not take this modifications into consideration and predicts more consumption than it should, given also greater biased results as shown in Figure 21. The addition of the variable desribing the building occupancy, that could include the specific employees calendar would help improving the accuracy in this season.

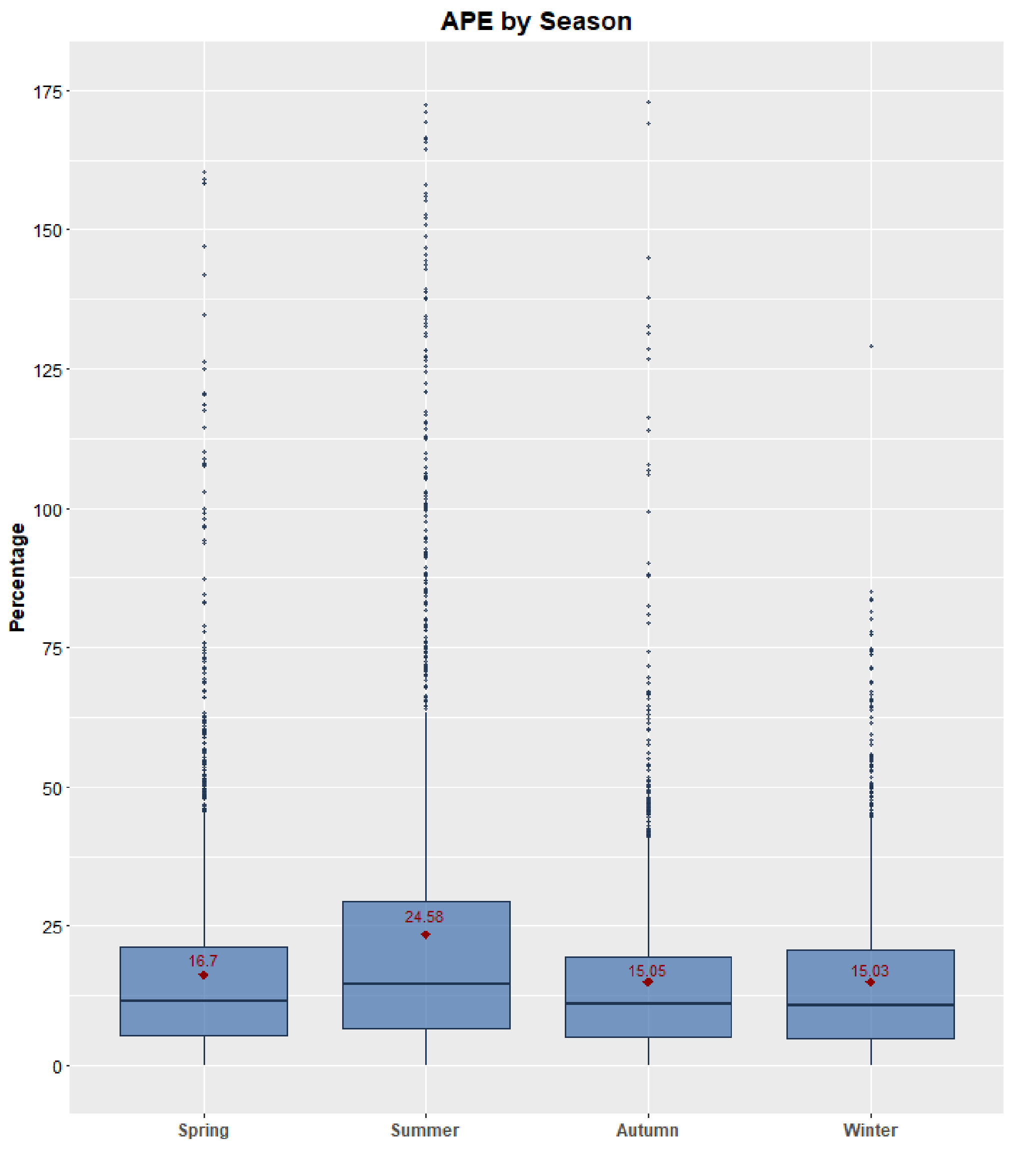


Figure 21. APE and MAPE (red diamond) by season.

It is difficult by looking to this graph ig the temperature has an important impact in the accuracy of the model. Although the temperature varies from one season to another, the variation on the metrics are probably due to the activity in the building and the working schedule and labeling. For future jobs, this schedule influence will be removed to isolate the effect of the temperature on the building.

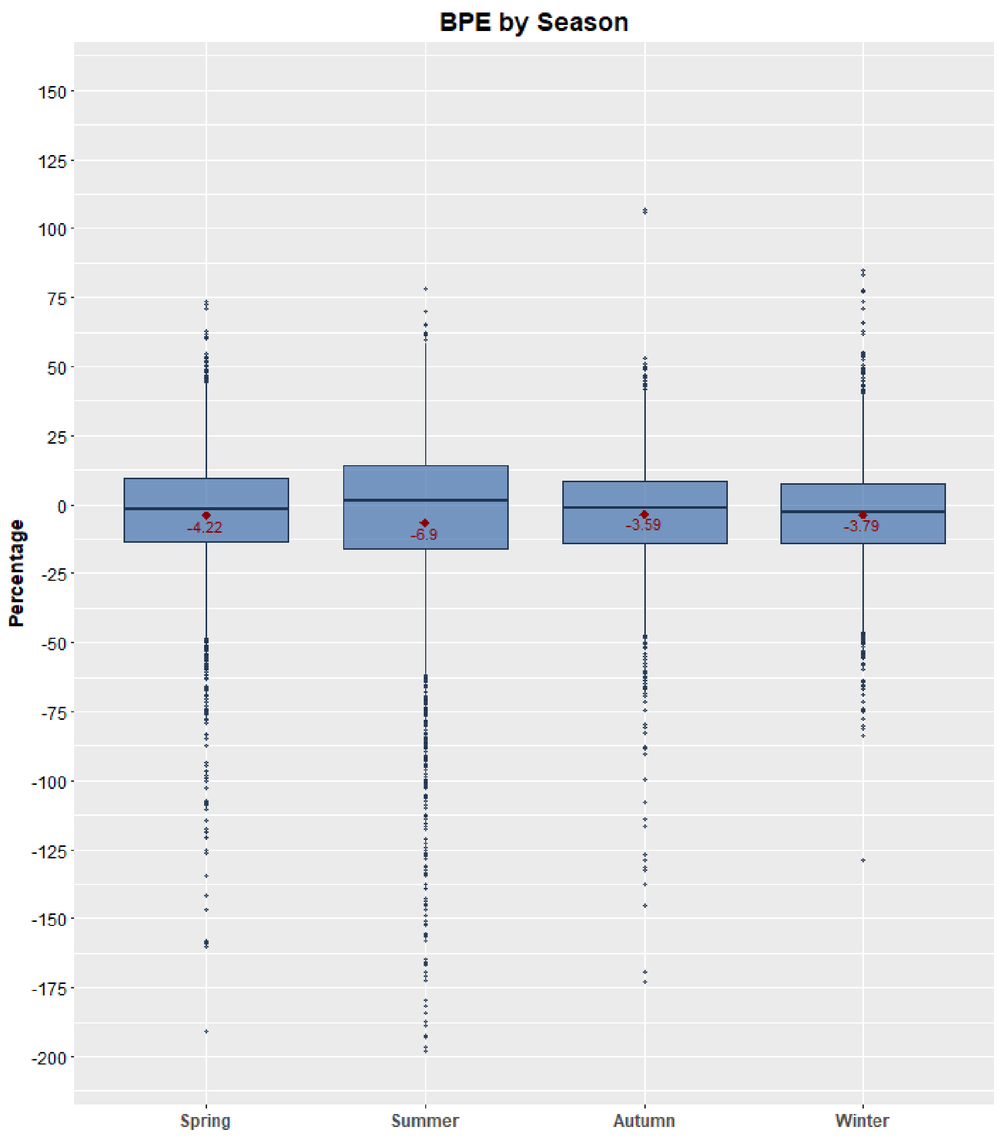


Figure 22. BPE and MBPE (red diamond) by season.

The last parameter of influence that will be study in this analysis is the hour of the day in which the prediction takes place. The APE and MAPE for each hour of the day during the whole evaluation period is shown in Figure 23. It is convenient to recall Figure 4, in which the daily energy consumption pattern of the building is shown for two consecutive days. By comparing both shapes a direct relationship can be instantly derived regarding energy consumption and the time of the day. The bigger the variation of energy consumption is, the bigger the model deviates. Early in the morning, when the working activity starts for the day, the derivative of the energy consumption, that is, how fast it is change, it's at its maximum. Before 8 A.M. all the energy systems in the buildings are fired up and fast energy consumption increases takes place. In this period the algorithm reaches its top MAPE from 07:00 to 08:00 given energy consumption deviated an average of 27.6% to the actual one. This is repeated later in the day when, again, the derivative of the energy consumption in the buildings increases (this time negatively) when the employees start to leave the building. Again, values up to 26.1% of MAPE are reached in this period, specifically at 20:00h in the evening. These values differ notably from the overall MAPE of 18.01% and special attention should be given to improve these predictions as they have a key role in the final error metrics. The parameter of occupancy and specific employees schedule habits would probably decrease this critical times and help the model make better predicions.

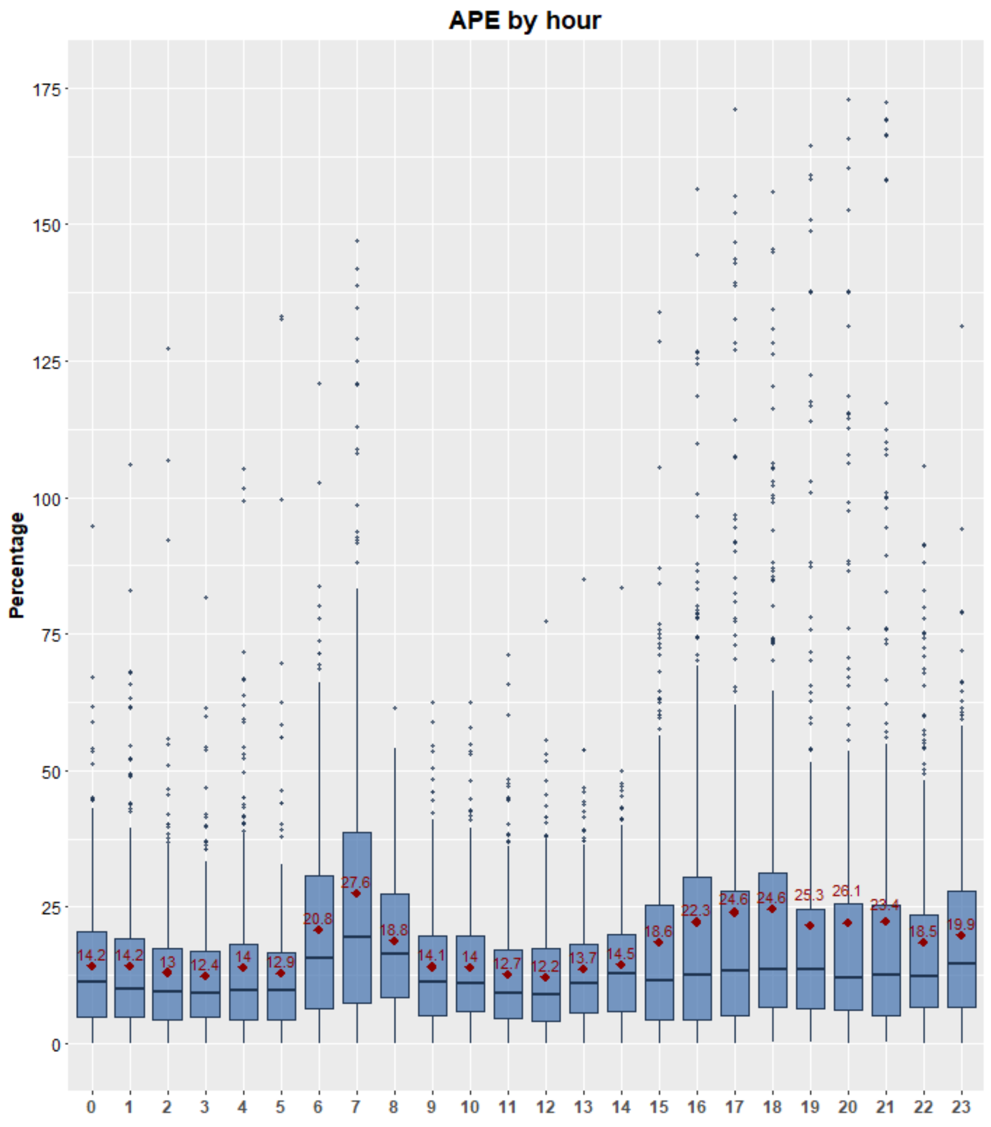


Figure 23. APE and MAPE (red diamond) by hour of the day.

The BPE also reflects the pattern explained for APE and MAPE regarding the hour of the day. The most biased results are obtained at 07:00 and 19:00, concurring with the times where the energy consumption variation is maximum. It is also convenient to highlight how in the flat periods of the energy consumption curve of a day, the model, although still having a considerable MAPE above 10%, it has a much lower MBPE. This reflects that in flat periods the model is not that biased and that the overall MBPE of almost – 5% comes probably from the periods of maximum variation in the energy consumption, where the model is clearly given over predictions.

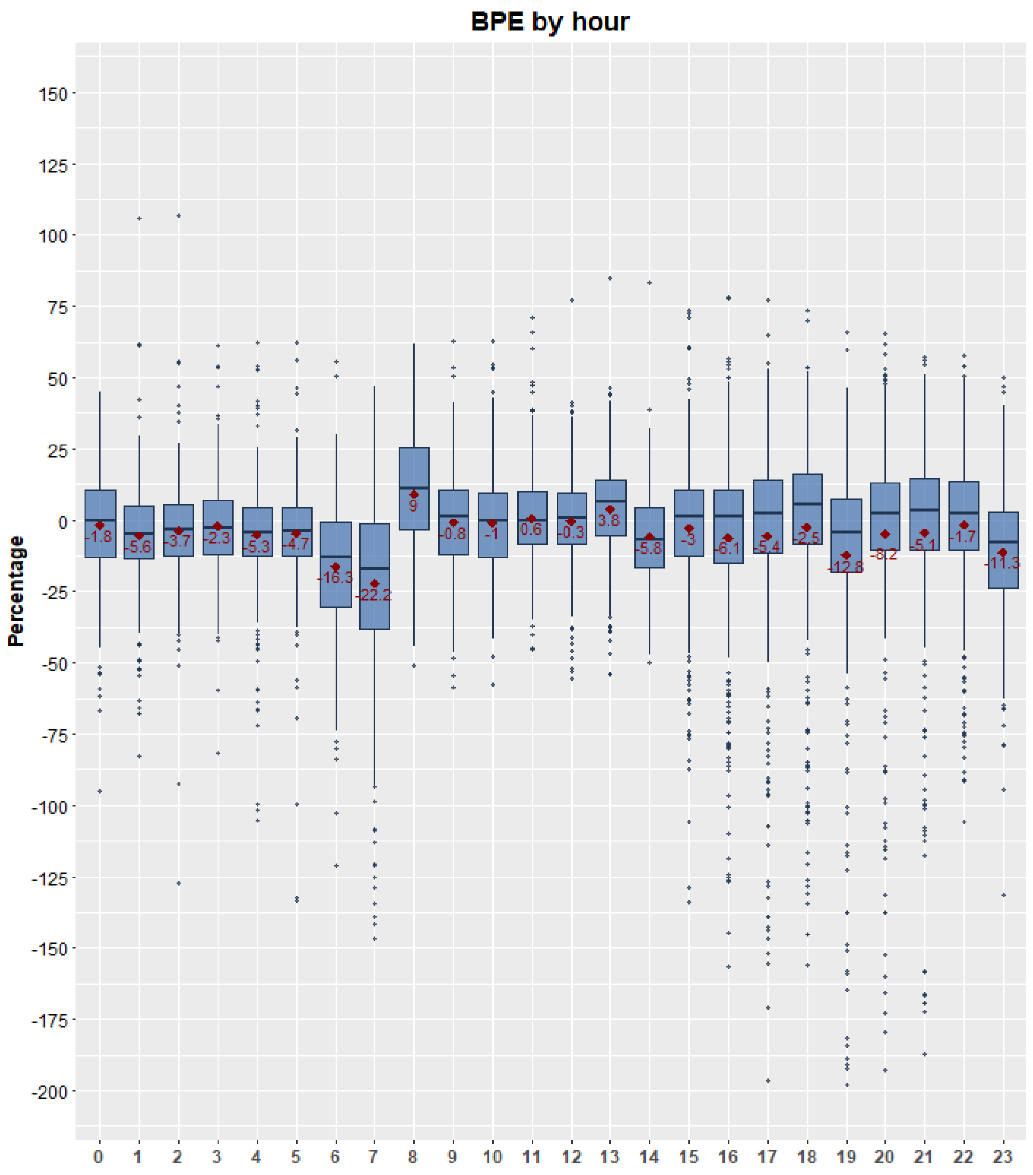


Figure 24. BPE and MBPE (red diamond) by hour of the day.

Leaving the BOXPLOT representations another two graphs are shown to complete the results evaluation. The first one, Figure 25, shows the APE for each instance chronologically arranged. There are two wide blank spaces in February 2017 and April 2017 due to anomalies in the predictions output, as seen in Figure 59, Figure 63 and Figure 65. Apart from that, the cloud of points in the first part of the sample seems to represent a higher APE values than the second part of the sample, where the points are more concentrated. This fact could evidence improvements in the accuracy and performance of the algorithm along the evaluation period as more data for training the model was available and incorporated to the model. The importance of re-training the model will be explain in Part II but in this case, once again, there is no confirmed information on what retraining strategy the model is applying and no conclusions can be drawn. Related to improvements in the model, it has been known during the development of this dissertation that the working calendar for 2017 was updated at some point because in 2017 (second half of the sample) there are not big anomalies due to working calendar issues. No Holidays were forecasted as normal working days and viceversea.

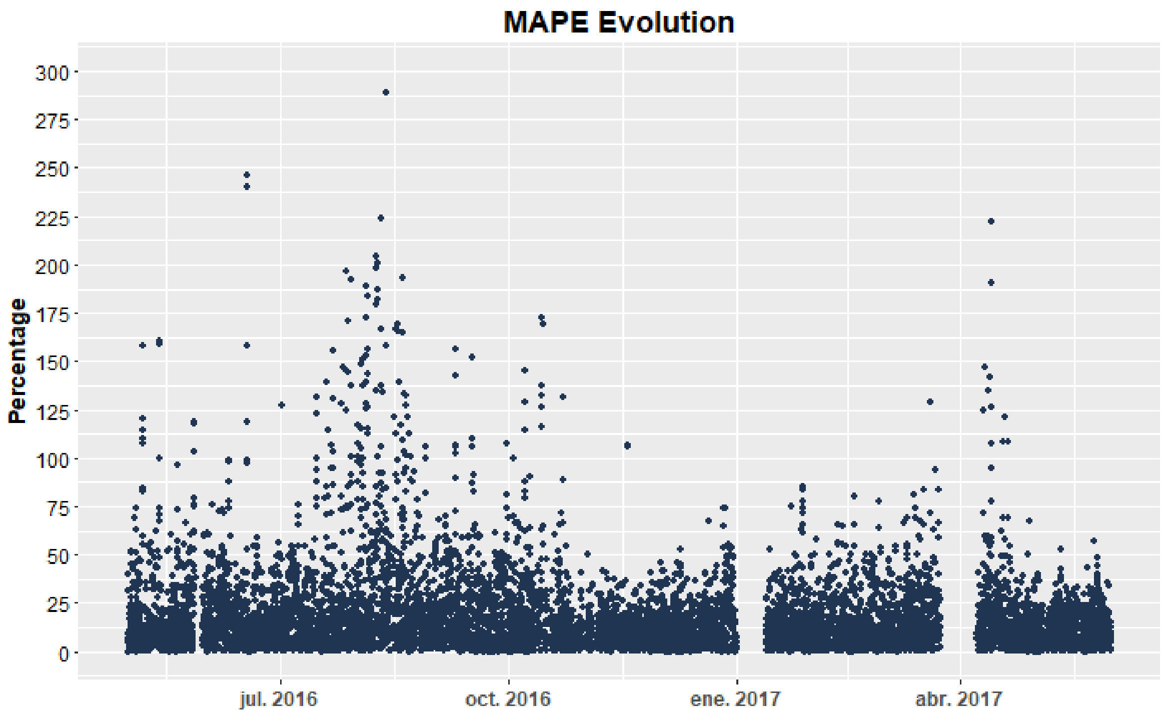


Figure 25. Each instance APE for all the evaluation period

The second graph, shown in Figure 26, represents a histogram of the BPE. In a perfect model it should look like a perfect Gaussian distribution, centered in the 0. Nevertheless, it shows an asymmetric behavior that aligns along with the MBPE indicating a biased model. It is of importance to point out from this figure that there are almost none instances where the real consumption was 100% higher than the prediction whereas there are several occasions in which the model predicted energy consumption 100%, 200% and almost even 300% higher than the real consumption. Ignoring this difference in the important deviations, for the smallest ones the shape of the histogram is similar to the Gaussian distribution. Despite being somehow biased, the distribution of upper and lower deviation seems to be even, and therefore model looks to be working within acceptable ranges. In order to improve the model the points where the model is deviating more should be study. A methodology to study error metrics must understand what the model is doing and have access to the specific inputs of the model for those instances in which it perms worse. As there is no access to this data this error metric analysis will be further developled in Part II, when propietary models will be carried out and all the information is known. Basically, the process would imply to cut the *tails* of the BPE distribution and look into the instances that are filtered out byt the cut. In Figure 26 a good range would be to cut instances over 100% of APE and study their typology.

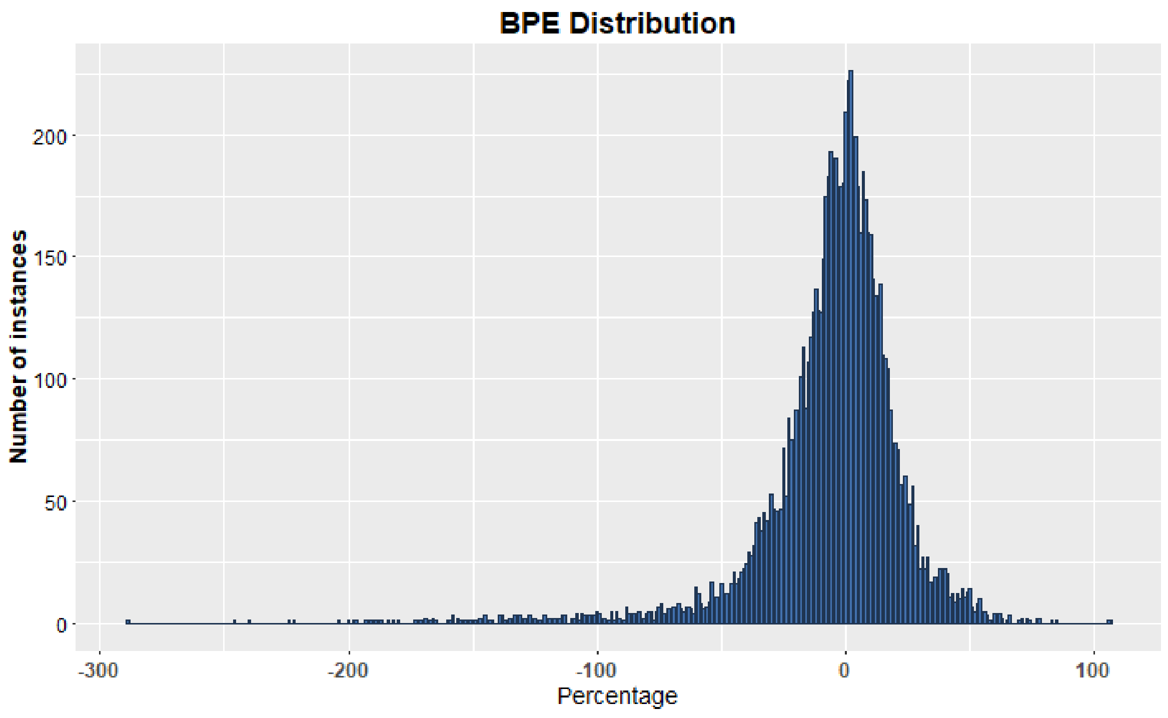


Figure 26. BPE Histogram

4.4 Evaluation methodology summary

The evaluation methodology followed is summarized in this section through a process schematic shown in Figure 27.

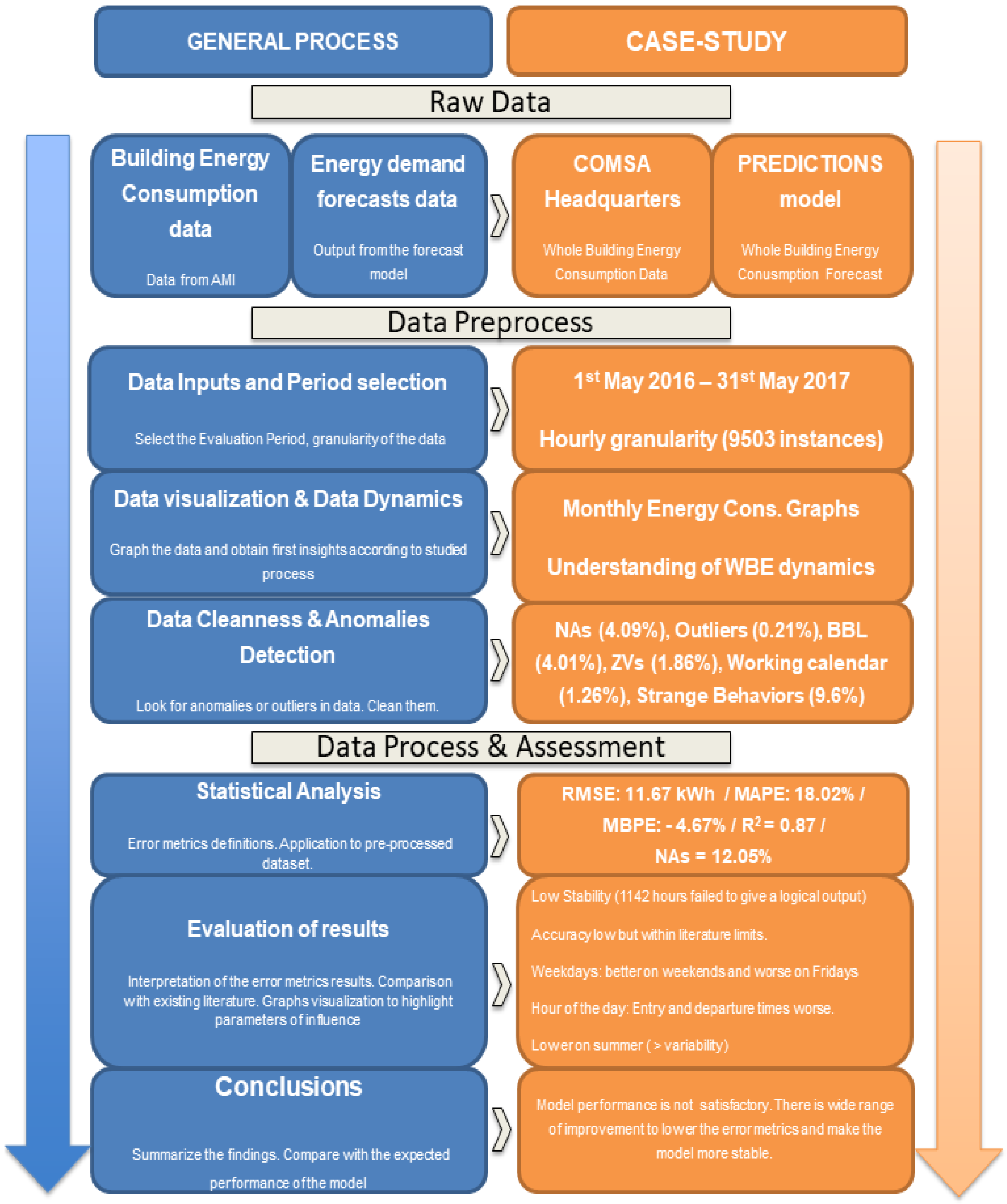

GENERAL PROCESS
CASE-STUDY
Raw Data
Building Energy Consumption data
Data from AMI
Energy demand forecasts data
Output from the forecast model
COMSA Headquarters
Whole Building Energy Consumption Data
PREDICTIONS model
Whole Building Energy Consumption Forecast
Data Preprocess
Data Inputs and Period selection
Select the Evaluation Period, granularity of the data
1st May 2016 – 31st May 2017
Hourly granularity (9503 instances)
Data visualization & Data Dynamics
Graph the data and obtain first insights according to studied process
Monthly Energy Cons. Graphs
Understanding of WBE dynamics
Data Cleanness & Anomalies Detection
Look for anomalies or outliers in data. Clean them.
NAs (4.09%), Outliers (0.21%), BBL (4.01%), ZVs (1.86%), Working calendar (1.26%), Strange Behaviors (9.6%)
Data Process & Assessment
Statistical Analysis
Error metrics definitions. Application to pre-processed dataset.
RMSE: 11.67 kWh / MAPE: 18.02% / MBPE: - 4.67% / R² = 0.87 / NAs = 12.05%
Evaluation of results
Interpretation of the error metrics results. Comparison with existing literature. Graphs visualization to highlight parameters of influence
Low Stability (1142 hours failed to give a logical output)
Accuracy low but within literature limits.
Weekdays: better on weekends and worse on Fridays
Hour of the day: Entry and departure times worse.
Lower on summer (> variability)
Conclusions
Summarize the findings. Compare with the expected performance of the model
Model performance is not satisfactory. There is wide range of improvement to lower the error metrics and make the model more stable.

Figure 27. Schematic of the methodology and application to the case-study

4.5 Conclusions Part I

In this dissertation an evaluation methodology to assess the performance of EDF models in disaggregated loads (at building level) has been proposed through the analysis of a case-study. The ML model currently forecasting the energy demand (up to 48h ahead) in the COMSA Headquarter office building has been analyzed as an application example.

The evaluation period was set from 1st May 2016 to 31st May 2017. A total of 9503 instances with measured hourly WBE consumption and forecasted demand comprises the dataset to be analyzed. A rigorous data preprocess was performed to eliminate abnormalities in the data. NAs, Zero and below the BBL predictions, outliers, deviations in the working calendar and strange behavior have been removed. The final stability of the model is 87.96%. Four different statistical metrics have been used for the analysis. Their overall results yield a MAPE of 18.01%, a MBPE of – 4.67 %, a RMSE of 11.65 kWh and a R^2 of 0.87.

The results show an unsatisfactory accuracy and stability for the model. Nevertheless, this model is relatively new and with proper calibration, refine and re-train, it is possible that it will perform better. Different parameters have been found to have an important influence over the accuracy of the model and re-considering the inputs better metrics could be achieved. In general, performance is worse those days where the working schedule is not strict, that is, Fridays, days prior to holidays or special summer schedule. Regarding the seasonal effect in the accuracy, no conclusions can be drawn out of this first evaluation. The model performs different depending on the hour of the day, obtaining the worse during the start and the end of the working session, when the working activity has more variability. Most of this differences regarding the above-mentioned parameters are related to the working activity in the building. Those days when the activity tends to have less variations, that is, working days from Monday to Wednesday, the model performs better. The flatter the energy consumption variations are in the building, the better the model performs. It is likely that an occupancy parameter will improve the performance of the model in this high variability periods. This parameter, apart from being immediately related to the energy consumption in the building, it helps tracking the working activity of the company, which will add valuable information to the model. Habits on the arrival and departure time, as well as holdays personal calendar, of the company personnel would increase the model performance.

The methodology has been defined and applied to a real case, giving valuable insights about its performance and the parameters that most affect the model performance. Nevertheless, it is an early stage and further development must be performed. In Section 7 guidelines to further develope the methodology will be provided.

5 PART II: Evaluation of other Machine Learning alternatives

In the second part of this dissertation other models, also based in ML algorithms, will be applied to the available dataset of the energy consumption of the COMSA Headquarters office building. The objective is to obtain the best energy predictions 24h and 48h and compare them to the results of current model being used by PREDICTION.

5.1 Methodology

As it has been explained in 3.2, there is a wide variety of ML models being currently studied and implemented to make energy demand predictions. From all that spectrum of algorithms this thesis focuses on three different options. First, Artificial Neural Network as they are the most widely methodology used in load forecasting. Second on the Random Forest because it is known that PREDICTION works under this algorithms they are tremendousl powerfull. As third option, it has been decied to work with k-Nearest Neighbors algorithm, a simple and potent model that is hardly found on the literature. All these models will work in the same conditions to be able to compare them. Therefore, the data inputs and preparations will be the same in all cases aswell as the training and testing time periods.

5.1.1 Data Preparation

Data-process is the process that takes the raw data from the building consumption (reference known outputs) and the influence parameters that will be consider in the study and assign the proper format to optimize the model performance.

Data inputs

The influence parameters are those variables that have an effect on the variable to predict and thus, are the candidates to be inputs of the forecast model. There is not a universal agreement for input selection, demonstrable influence and experience and expertise on the field must select the final group of predictors to be used (Raza and Khosravi, 2015).

In the energy-field, there are several factors that have an effect on the final energy consumption of a particular building. A large list of parameters that have, to a greater or lesser extent, an impact on Energy Demand Forecasting models' outputs can be made. It could be made including parameters relatives to the equipment (technical characteristics, time of usage, efficiency, etc.), to the users (e.g.: how likely are they to make a smart usage of the systems) to the weather, to the time of the year, to the location, to the building characteristics, etc.

In this master thesis there have been considered seven different influence parameters. In the first part of the thesis it was demonstrated that there are some variables that strongly affects the results of the model. Apart from these, other variables are set as influence parameters by experience in the field and by other literature studies that have proved them to be good energy demand predictors. These variables are related to weather conditions and to previous loads in the building (autoregressive character). The final selection of parameters to be included in the models is set as follows:

- **Scheduling parameters:** Conclusions of part I aligns with literature bearing out that the effect of scheduling is strong in the energy consumption of a tertiary building. Within this group we differentiate:

    - *DayType:* Weekday/weekend. Separated by Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday.
    - *DayCategory:* WorkingDay/Semiworking Day/Holiday
    - *Hour*: of the day
    - *Month:* Month of the year

  In the literature there are different options to deal with scheduling. One of them is by using different models (e.g. build a specific model por each day of the week or for each hour of the day) or by using an indicator variable (dummy variable). As (Papalexopoulos and Hesterberg, 1990) did in their analysis, this study will consider the categorical variables by creating dummy variables, that is, splitting every parameter in all the categories that it contains, and assigning the value 1 or 0 when corresponds.

- **Weather Parameters:** *Ambient temperature* is proved to be have a strong relationship with the energy consumption of a building (Kissock, Reddy and Claridge, 1998). Regarding weather variables this will be only one that will take part in the regression analysis. Although it is also proven that humidity and other parameters also have influence over the electric consumption in buildings, they effect is almost neglectable with respect to the ambient temperature. (Neto and Fiorelli, 2008) showed how the addition of relative humidity and solar irradiation parameters slightly improved the accuracy of their model. As these parameters are at the moment not available for the building location it has been decided to not include them as data inputs for the models.

- **Historical consumption predictors:** Energy consumption follows time series data structures, meaning that it represents data points taken over time in which the next value has some kind of relationship with the previous one. There are trends, correlations and seasonal factors that should be accounted for. In the energy consumption there are three different kind of seasonality's: daily pattern, weekly pattern and annual pattern. The Auto-regressive (AR) parameters to be considered in this study are the following ones:

  - Load at the same time the previous day ($L_{(D-1)}$)
  - Same day and same hour but one week before ($L_{(W-1)}$)

All the influence parameters considered in this case-study and their characteristics are summarized in Table 4.

Table 4. Influence Parameters to be considered as Inputs to the models

Variable	**Type**	**Value**	**Description**
Day Type	Categorical	Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday	Week of the Day
Day Category	Categorical	WorkingDay, SemiWorkingDay, Holiday	Category regarding working calendar. Semiworking day: Reduced schedule (Summer and days before HOLIDAY).
Hour	Categorical	[1-24]	Hour of the day
Month	Categorical	[1-12]	Month of the year
Temperature	Numeric	[-1,32]	Exterior Temperature. In ºC
Load$_{(D-1)}$	Numeric	[8, 152]	In kWh
Load$_{(W-1)}$	Numeric	[8,152]	In kWh

Data pre-process

Once the feature selection has been fixed, the process to set the proper format to the raw data from the building consumption (known outputs) and the influenceable parameters is performed. The way in which the data is passed to the model has a strong effect in its outcome and, therefore, it must be converted to optimize the model performance. The steps followed to prepare the data are explained in this section.

Feature scale

The input variables have different forms and scales. As Table 4 shows, there are categorical variables and numerical variables. Between the categorical variables there are variables with different number of categories, and between numerical parameters, the rangers and scales can notably differ.

The algorithms need these parameters to be reduced to the same scale so it computes the outcome in less time and better precision. Two processes are used in this study to properly scale the data inputs. For categorical values, they are converted using *dummy* variables, that is, creating one sub-variable per category and assigning the value 1 or 0 accordingly to the original value. An example of the process of converting categorical variables to *dummy* variables is shown in Table 5. In this table, the *DayType* variable for three random instances is converted to seven different *dummy* variables (one per day of the week) and a 1 is assigned to the one that corresponds to the weekday, assigning 0 to the rest. Obviously, there can not appear more than one 1 in the same instance in the group of *dummy* variables representing the same influence parameter. The same process is applied to *DayCategory (*three *dummy* variables), *Hour* (24 *dummy* variables) and *Month (*12 *dummy* variables). After the process, the 4 categorical variables have been converted to 46 binomials 1/0 variables.

Table 5. Example of converting categorical variable into a Dummy Variable

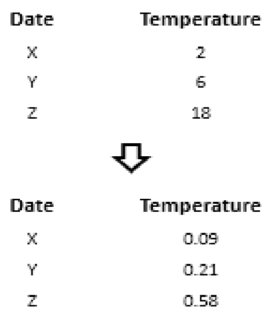
Date	**DayType**
X	Mon
Y	Tue
Z	Wed

⇩

Date	**DayType _Mon**	**DayType _Tue**	**DayType _Wed**	**DayType _Thu**	**DayType _Fri**	**DayType _Sat**	**DayType _Sun**
X	1	0	0	0	0	0	0
Y	0	1	0	0	0	0	0
Z	0	0	1	0	0	0	0

Regarding the numeric variable, they must be also rescaled to the same scale, that is, to have values between 0 and 1. The range of numerical values must be in the same scale.

Table 6. Normalization of the numeric variables



Date	Temperature
X	2
Y	6
Z	18

⇩

Date	Temperature
X	0.09
Y	0.21
Z	0.58

Once all the feature scaling has been accomplished the data inputs can be passed to the model, as shown in Figure 29.

Training and Testing period.

In supervised learning, as it is the case in this Machine Learning problem, there are, at least, two datasets. The first one, called training set, contains instances with input data together with known output of the variable to predict. In our case, the training set is comprised by rows with the values of the influence parameters explained in the previous section and the corresponding energy building consumption associated to those conditions (what would be the output of the forecast model). This dataset has been prepared so the model can learn or train and map the relationship between that output and the inputs.

In order to estimate how well a model performs it is needed another dataset to present to the model, but with the output *hidden.* Once the model reads the inputs data and apply the corresponding algorithm it yields a prediction. As the correct output for the corresponding input is still known, you can study how well the model has mapped the output to the inputs by an accuracy study. Through the testing set you understand if your model is good enough to perform predictions over data which output is unknown and trust the result.

In order for the model to be well trained and in order its performances to be reliable the selection of the size (number of samples) of these dataset is important. As rule of thumb, it is common to select, from all the available data, 70% for the training set and 30% for the test set.

It was described in 4.1 that the energy consumption in the building of the case-study has been measured with enough guaranties since 1st January 2015. Nonetheless, the PREDICTION model just started working from 1st May 2106 and that is why the evaluation period in Part I started on this date. In this part, the training set will gather all the available data prior to the start of this evaluation period and the testing set will be the same of the evaluation period itself. That is:

- Training set: 1st January 2015 to 1st May 2016. It comprises 11661 instances of hours representing a period of 486 day or 1 year and 4 months.
- Test set: 1st May 2016 to 1st June 2017 It comprises 9503 hours instances or hours representing a period of 396 days or 1 year and 1 month.

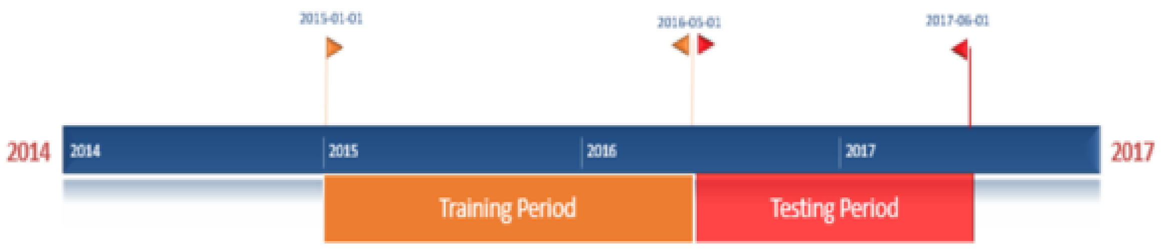


Figure 28. Training and Testing period for the ML model comparison

In this case, the rule of thumb 70/30 is not applied because of the nature of the data and the problem tackled. The purpose of the study is to compare different models that has available some data (training set) to train with and must predict over a certain period from which the output is *unknown* (testing set). The study simulates that models take all the data available before the testing period and will make periods over it without being re-trained.

As it has been mentioned throughout the PART I there is little information on how PREDICTION forecasting model works. There is no data on how the training and test are made. It is only mentioned that the minimum training period for the model to work is one month and the recommend is 6 months. This study assumes that the mode has trained with a certain amount of data prior to the time it started working properly (1st May 2016) and it has not been trained with new data. In order to be able to compare the accuracy, the three different models explained in the following sections will take the same dataset for training and test.

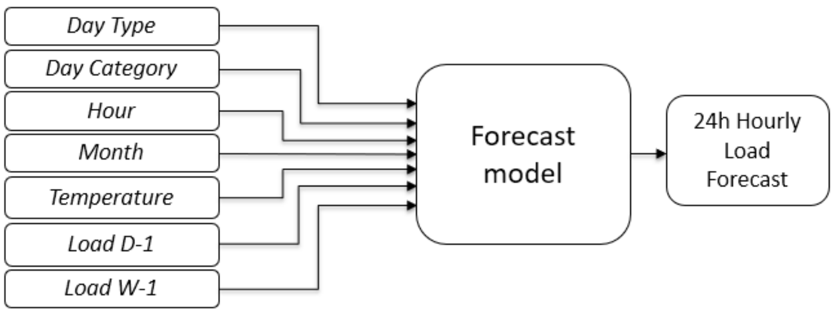


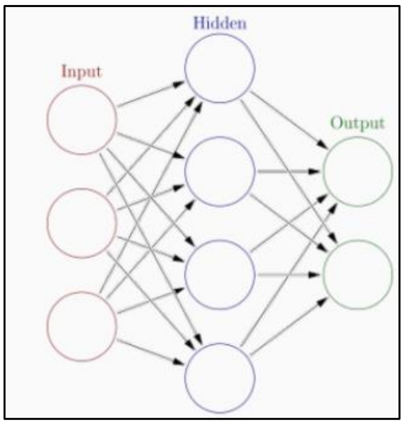
Figure 29. Data Inputs to the model (no dummy variables representation)

5.1.2 Model 1: Artificial Neural Networks

As it was described in 3.2 ANN are known as *black-model* algorithms. it is really complex to understand what is actually happening with the inputs and how are these are treated within the models. Neural Networks extracts the nonlinear relationship between input variables using training process of the network. The network can learn the future output behavior of load demand by using pattern reorganization functions (Raza and Khosravi, 2015).

The ANN applied to the case-study is based in a feed-forward neural network (FFNN) with a single hidden layer. It was first developed by Brian Ripley and Willian Venables for the language S. (Venables and Ripley, 2002) and then adapted to R and added to the CRAN repository in February 2016 (Ripley, 2017).

Feed-forward networks are the simplest ANN as there is not feedback loop. Information flows just in one direction, from the inputs to the outputs. Through supervised learning this network tries to minimize the Mean Square Error between the targeted and network output vales. The MSE is used to update the weights and bias values of network iteratively until an acceptable value of the error is reached.



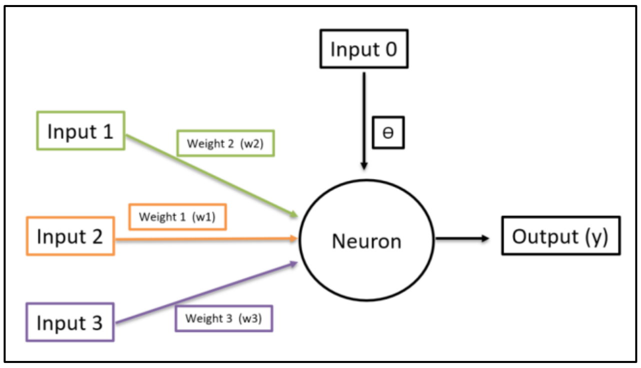


Figure 30. Left: General structure of a FFNN. Right: Neuron Structure

$$y = \theta + w1 \cdot x1 + y2 \cdot x2 + \cdots + wn \cdot xn \qquad [Ec.9]$$

Basic NN network, with 3 layers and 4 neurons in the hidden layers, and the process occurring in a single neuron is represented in Figure 30 and ecuation [Ec.9] . Neural Networks can map really complex relationship between the inputs and the outputs and it is a high powerful algorithm. Main drawbacks of ANN are that they are hard to interpret and they usually imply a high computational cost.

The FFNN configured in this study is based in the Ripley and Venables model, and uses just one single layer of neurons. The parameters of the model configured has been:

- 6 Nodes in the Hidden layer.
- 0.1 Decay. This parameter represents weight regularization in the NN.
- 1000 Max. Iteration. The process will stop if it has not converged if this number of iterations is reached.

In Figure 31.It has been plot one ANN model used in the analysis. The first column represents the inputs that passes the information to each of the 6 nodes of the hidden layers through the corresponding weights. The results of the model are gathered together with the rest of the models in Table 7. GLOBAL Error metrics for the models for given training/testing conditionsTable 7.

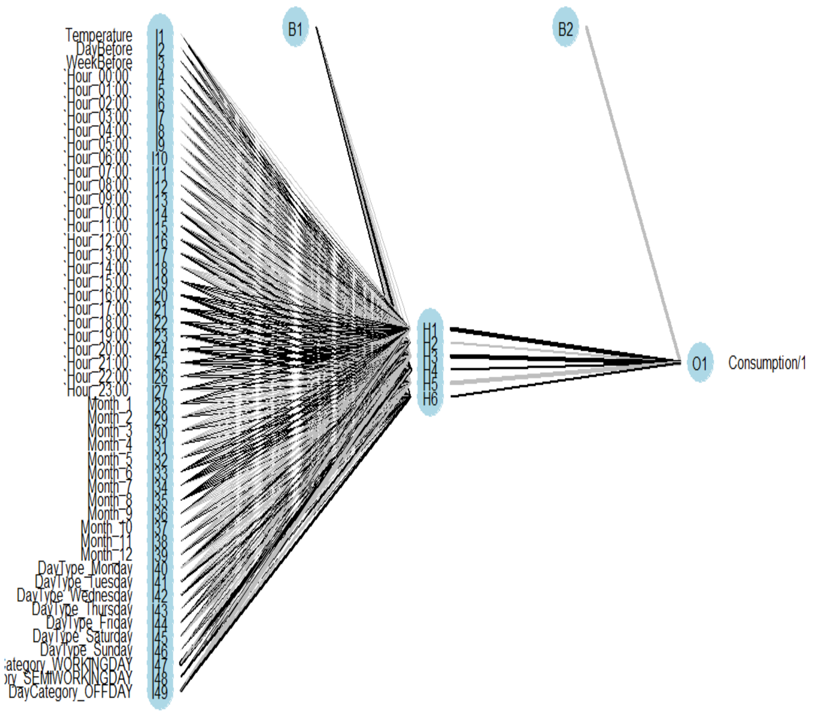


Figure 31. Representation of the Neural Network model with their three nodes levels (input, hidden layer and output)

The ANN network used in this study is a simple network, there are dozens of possible configurations and parametrization of the models. It is out of the scope of this dissertation to deepeen in the ANN field. To mention some of the current models, there are researchers working on Recurrent Neural Networks, Markov Chain, Hopfield Chain, Deep Belief Networks, Deep Convolutional Networks, etc. In Future work it will be proposed to use and optimize the models created in this study to improve the predictions accuracy. For further understanding of the ANN and its application to Load forecasting (Raza and Khosravi, 2015) published an extended study.

5.1.3 Model 2: k- Nearest Neighbors

k-Nearest Neighbor is a simple non-parametric algorithm frequently used for classification problems. It bases the predictions on a similarity measure. It can be used also for regression, to predict the output the algorithm looks for the nearest neighbors' points which pair (input, output) is known and, depending on the number of neighbors considered, assign it an output value. If the algorithm is configured to use just one neighbor, then it would assign to the input the value of the output of the closest neighbor (measured by certain distance criteria). When using higher number of neighbors, the value assigned would be the average.

A simple example is shown in Figure 32 to understand the technique. The known data points are those for inputs 1, 1.8, 3.2, 5 and 5.9 (the blue dots). The output (Y-axis value) for input 4 is unknown. Using k-NN with 1 neighbor it would assign to input 4 the output of the closest known data point, that is, x = 3.2. It would then be assigned the value 6. When k = 2 it would also consider the next closest neighbor, that is, input 5. The result algorithm would average the output value of these two inputs. The value assigned to input 4 would be average between 6 (output for input 3.2) and 3 (output for input 5). Analogously, with k = 3 the output assigned to 4 would be 3.7.

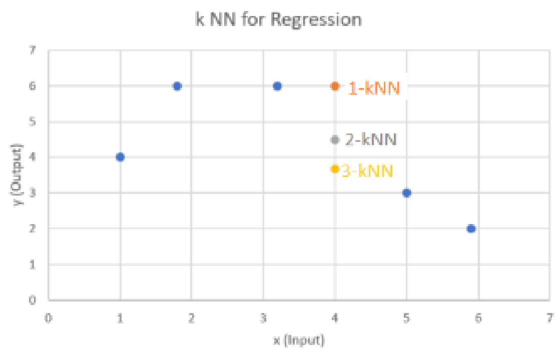


Figure 32. kNN regression. (Lavrenko, 2017)

The k-NN regression model used in this dissertation was created by Shengqiao Li and it's part of CRAN package FNN, updated to the repository in July 2013 (Li, 2017). It uses the Euclidean distances and the parameters selected in this model is k = *2* for 2 neighbors (Adding neighbors yield worse accuracy results) and the algorithm to use for distance computations is based on brute-force method. The results obtained for this methodology in this study are shown in Table 7, together with the rest of models applied.

5.1.4 Model 1: Random Forest (RF)

Random forest is an extension of traditional decision tree method. It is considered as an ensembled method as it combines multiple decision trees to produce powerful models. Each of the trees is trained independently, using a random sample of the data (Bradley and Amde, 2015). The ultimate results are obtained by weighting each tree's output as shown in Figure 33.

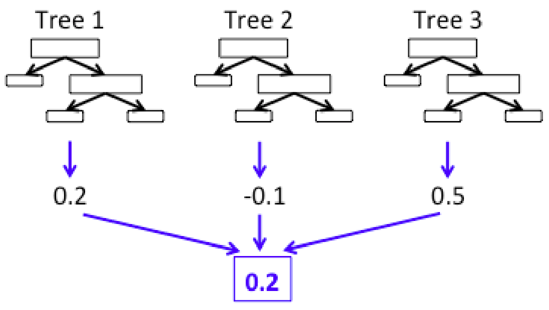


Figure 33. Ensemble Method. Example for regression(Bradley and Amde, 2015)

The Random Forest based method used in this case-study was first written in Fortran by Leo Breiman and Adele Cutler (Stat.berkeley.edu, 2017). The version used in this study is the adaptation to R language programming, written by Andy Liaw and Matthew Wiener (Cran.r-project.org, 2017). The model has been initiated with 1000 trees but the errors stabilize with a lower amount as seen in Figure 34 and the tree number was lowered to 500. The final results are gathered in Table 7.

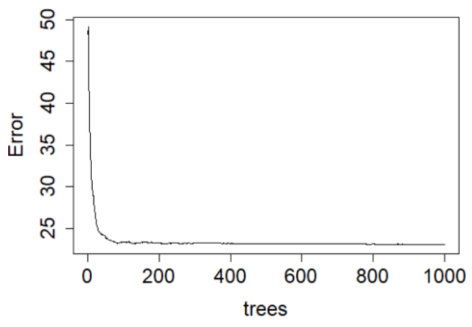


Figure 34. Error vs Number of Trees in RF model

5.2 Results and Discussion of Part II

Once the data is set to be passed to the models and these have been trained in the training dataset (See Figure 28) the models' performances are analyzed in the testing dataset. Results are summarized and findings are discussed in this section.

5.2.1 Model Comparison and Model Selection

The error metrics explained in Part I are applied to the models explained in the previous sections, which have been described in the porevious section. Selected training and testing periods (See Figure 28) are the same for all of them. Results are summarized in Table 7. Random Forest is clearly the model with best performance, overcoming the rest of the models in all the metrics. It has lower MAPE, it less Biased than the k-NN and the ANN model and its RMS and coefficient of determination are better.

Table 7. GLOBAL Error metrics for the models for given training/testing conditions

	MAPE %	**MBPE (%)**	**RMSE (kWh)**	**R^2**
k-NN	19.55	-13.08	8.19	0.93
ANN	16.21	- 10.21	5.42	0.97
Random Forest	13.87	- 8.35	5.49	0.97

Although it was mentioned in the scope of the project that the computation time would be not considered in this analysis, it must be mentioned that while having the best metrics and performance, the Random Forest is a much slower method compared to the other ones. The computional time consumed order of the methods increases with it complexity and for determined applications could definitely be a determined factor when selection a specific method.

Table 8. Computing time scale for each model measured in a personal computer (i7, Windows, 8GB RAM)

Model	**Computing Time (s)**
k-NN	3 - 8
ANN	300 - 600
Random Forest	800 - 1400

As the computing time is not taking into consideration in this study the best model selection is easy. Random Forest outperforms his competitors and will be selected to carry out the rest of the analysis. All the studies shown in the following sections have been confirmed by its implementation with the other methods (ANN and kNN), obtaining analogue results. Nonetheless, with the purpose of keeping the report cleaner and more readable, their results have been omitted and just the ones corresponding to RF models will be presented.

5.2.2 Influence of Working Calendar

During the elaboration of the current dissertation, different issues regarding the working calendar have arisen. In order to prove the importance of including a detailed working calendar in the data inputs to the model, Table 9 shows the different metrics for models with no working calendar at all, model with detailed working calendar, and PREDICION model, which working calendar remains unknowns.

Table 9. Influence of the building Working Calendar

	Working Calendar	Test Period	MAPE %	MBPE (%)	RMSE (kWh)	R^2
RF Option 1	No working calendar is considered. Just Fridays as Semiworking Days and Sat/Sun as Offdays.	2016-06-01 to 2017-06-01	19.78	-12.13	9,51	0.90
RF Option 2	Corrected Working Calendar to Company Specifications. Summer calendar included plus more Semiworking Days before holidays.	2016-06-01 to 2017-06-01	13.87	- 8.35	5.49	0.97
PREDICTION	Unkown. Seems to be wrong in 2016 and corrected for 2017. Not consider holidays to the local level but to regional level.	2016-06-01 to 2017-06-01	18.01	-4.67	11. 67	0.86

It has been observed the importance of properly labelling the instances according to its *DayCategory*. Thanks to the error analysis it has been possible to identify and properly corrected many of these labels. In many instances where the APE was too high the reason behind that deviations was a wrong category on the *DayCategory* variable. As an example of this, the k-NN model was evaluated and the instances with APE bigger than 100% were carefully analyzed. These points have been represented in Figure 35, which represents the output predictions of a k-NN model filtered by APE bigger than 100% for a testing period between 1st January and 1st June 2017.

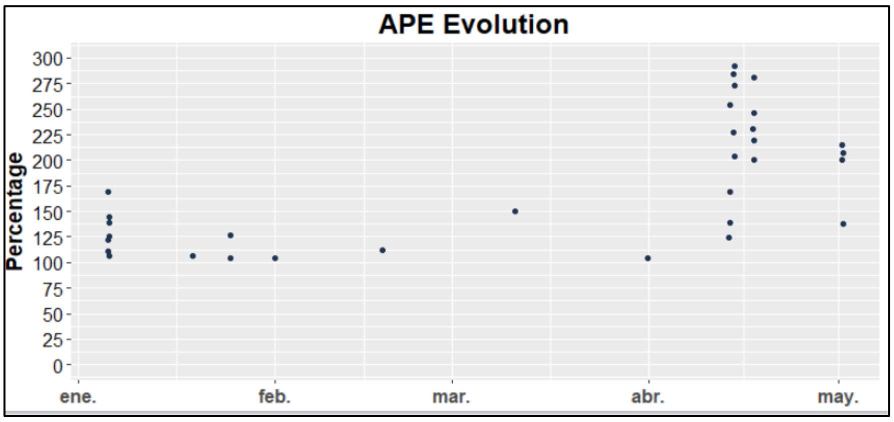
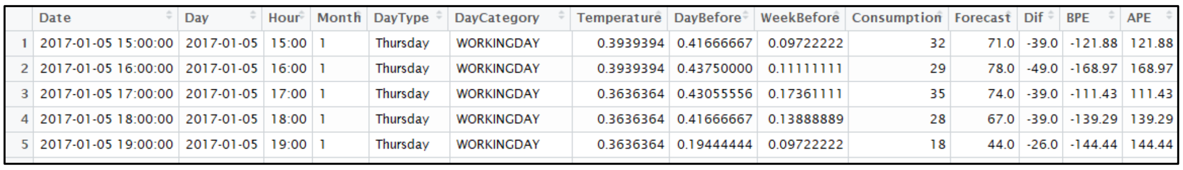


Figure 35. Data Points filtered with APE > 100% for a k-NN model

The days corresponding to the instances appearing on this filtered representation of the output were cross-checked in the dataset and many of them were wrongly labeled. For example, the 5th of January 2017 appears as *WorkingDay* but its consumptions correspond more with a *SemiWorkingDay.* In a normal WorkingDay, at 17:00 the consumptions usually exceed 70 kWh but in this case, it is just 35 kWh (See Figure 36)



	Date	Day	Hour	Month	DayType	DayCategory	Temperature	DayBefore	WeekBefore	Consumption	Forecast	Dif	BPE	APE
1	2017-01-05 15:00:00	2017-01-05	15:00	1	Thursday	WORKINGDAY	0.3939394	0.41666667	0.09722222	32	71.0	-39.0	-121.88	121.88
2	2017-01-05 16:00:00	2017-01-05	16:00	1	Thursday	WORKINGDAY	0.3939394	0.43750000	0.11111111	29	78.0	-49.0	-168.97	168.97
3	2017-01-05 17:00:00	2017-01-05	17:00	1	Thursday	WORKINGDAY	0.3636364	0.43055556	0.17361111	35	74.0	-39.0	-111.43	111.43
4	2017-01-05 18:00:00	2017-01-05	18:00	1	Thursday	WORKINGDAY	0.3636364	0.41666667	0.13888889	28	67.0	-39.0	-139.29	139.29
5	2017-01-05 19:00:00	2017-01-05	19:00	1	Thursday	WORKINGDAY	0.3636364	0.19444444	0.09722222	18	44.0	-26.0	-144.44	144.44

Figure 36. 1st January 2017 was marked as WorkingDay but it should be SemiWorkingDay.

After similar analysis more than 12 days were found to be marked as categories they do not belong to. These days had a remarked influence over the final results as it can be derived from Table 9. Decision on which days are SemiWorkingDay or which days are holidays might depends solely on the specific company, and it Is crucial for accurate predictions to have an updated and detailed knowledge of all of them.

5.2.3 Influence the re-training strategy

In ML the models are trained using existing data. Usually, in order for the model to make accurate predictions the training data must be updated and the model must be re-trained with the new available data. First, because the models will make predictions within a similar distribution of the original training data. Data distributions might drift over time and re-training the model with adequate frequency will keep track of this variations and add them to the predictions distribution. Second, model's performance can degrade and to keep the accuracy level or increase it (if there is margin to do so).

There are different strategies of re-training a ML model. *Online learning* updates the model automatically with every new instance or set of instances. It is not a *total* update as it is mainly based on updating some parameters of the model, as the weights, whereas other important parameters will remain the same until a thorough train is performed. In *offline learning*, the model starts with a sample or a certain amount of available data until the performance it is satisfactory to start making predictions. The model will keep working as it has been configured in this first train and will remain the same until a new training process takes place, no specific times or set of instances are given for this to happen. Lastly and in between these two methodologies, there is a *batch learning*. In this kind of training strategy, a batch structure is configured (can be by number of instances, by time or by specific dates). The batch its passed to the algorithm and it updates its parameters after consuming all the comprised instances. The selection of one of these strategies depends on the data, on a comprehensive analysis of what it is representing and on the software and hardware system capabilities.

In the case of the single office building energy demand forecast there two batches will be configured to show impact of re-training in the model performance. The original model, which training set and evaluation period was defined in Figure 28, will be compared to the performance of the same model after being re-trained with new data. The first update will be simulated to be the 1st of September 2016, corresponding to the first quarter of the evaluation period, and the second one the 1st of January of 2017, corresponding to the half of the evaluation period.

The results shown in Table 10 illustrate an improvement of the performances as the model is training with more instances. If there training model is not retrained in the whole 2016, the predictions that will come out of the model will be worse than if we re-training it by adding batches of new samples to the training dataset.

Table 10. Influence of the re-training strategy

	Re-training Date	Test Period	MAPE %	MBPE (%)	RMSE (kWh)	R^2
RF Option 1	No Update	2017-01-01 to 2017-06-01	12.46	-7.76	4.53	0.97
RF Option 2 (1 batch)	2016-09-01	2017-01-01 to 2017-06-01	11.20	-6.19	4.46	0.97
RF Option 3 (2 batches)	2016-09-01 & 2017-01-01	2017-01-01 to 2017-06-01	10.65	-5.44	4.44	0.97
PREDICTION	Unknown	2017-01-01 to 2017-06-01	14.80	-5.54	8.70	0.90

The three models have the same base training period but different re-training strategies. A timeline is shown in Figure 37 to clarify how the different options have been trained. The best results are obtained for the option 3, that is, the one that has gone through the process of being re-trained after two batches periods. All the metrics have improved except for the coefficient of determination. This improvement must be carefully interpreted as it can lead to some mistakes. The base training data-set contains one year and a half of hourly energy consumption. For the purpose of energy forecasting it is not a *large* dataset, it just contains 9503 samples. There are not even two samples of each season included in the sample. This implies, that as long as we add new instances, the better the trainer will be per definition. Given that the training sample would have included five years of energy-consumption it is more than probable that the metrics would not have improved in the same amount that the ones shown in Figure 37. But in that case, it still would be recommended to re-train the model for output distribution optimization. In this case-study due to the short training available data, re-training strategy should aim to update the model with relative frequency. Better metrics will be obtained and the distribution of the predictions will be updated.

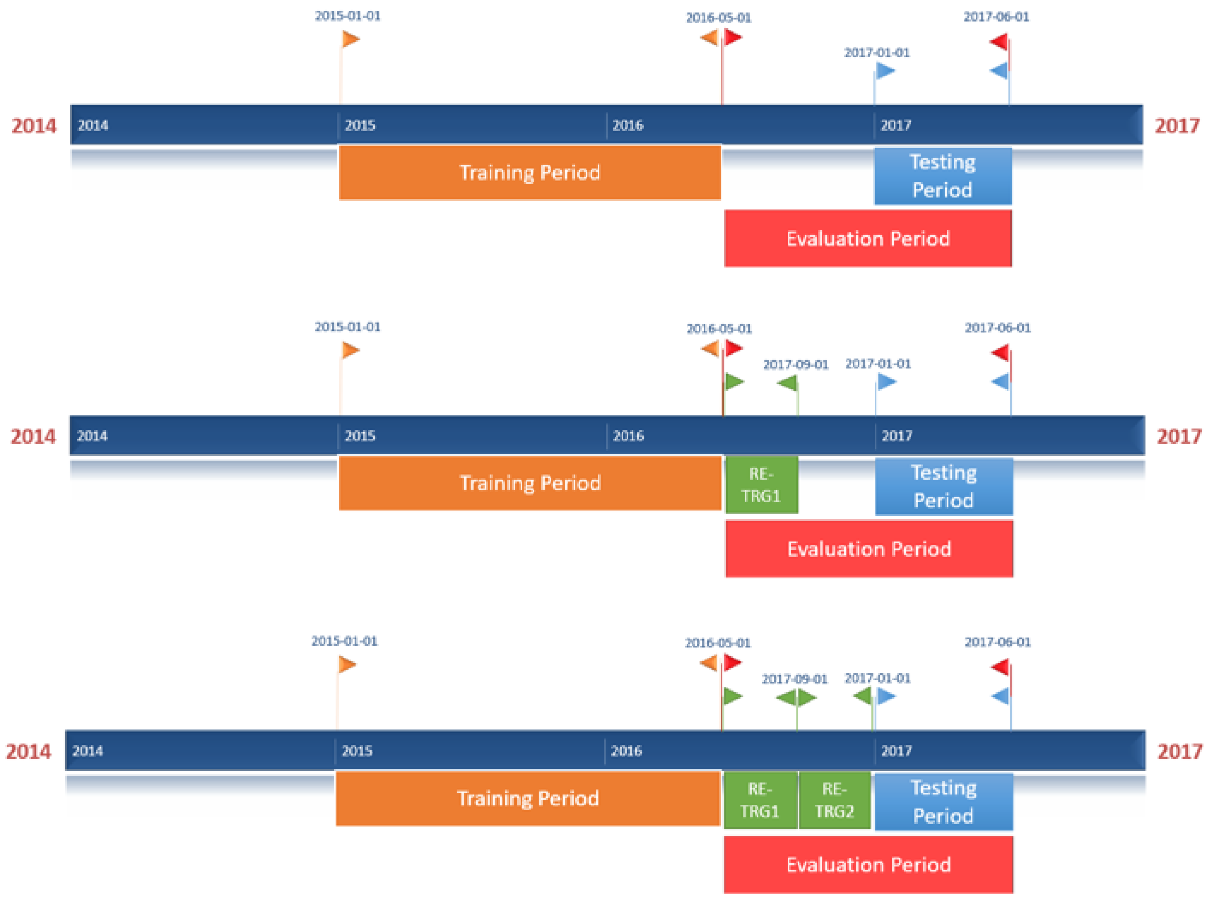


Figure 37. Timeline for different training strategies for the ML model.

5.2.4 **Influence of the Temperature forecast accuracy**

For testing the models shown in this dissertation the temperature has been treated as a “known” input. When these models are applied in real cases, the temperature that they will have available will be a “forecasted” temperature. It is important to understand the influence of this difference as it will have an impact on the accuracy of the EDF models.

PREDICTION model uses as inputs the temperature imported from a third party. This is a *forecasted* temperature and, by definition, it will have an error to a greater or lesser extent. In this study this fact has been considering by comparing the same model (RandomForest with same training and resting period) when it uses different temperature inputs. One model uses the actual temperature and the other two will use another simulation of temperatures forecasting. Table 11 illustrate the results for these different temperature inputs.

Table 11. Influence of the Temperature Input Accuracy

	T input	Train Period	Test Period	MAPE %	MBPE (%)	RMSE (kWh)	R^2
RF Option 1	Actual	2015-01-01 to 2017-01-01	2017-01-01 to 2017-06-01	10.65	- 5.44	4.53	0.97
RF Option 2	Simulation of accurate Forecast	2015-01-01 to 2017-01-01	2017-01-01 to 2017-06-01	10.95	- 5.84	4.62	0.97
RF Option 3	Simulation of bad forecast	2015-01-01 to 2017-01-01	2017-01-01 to 2017-06-01	12.16	-7.40	5.09	0.97

Option 1 model uses the actual Temperature to make the predictions. Option 2 and Option 3 used a *simulated* forecasted temperature. From the original actual temperatures another vector has been created to introduce some error in the temperatures used as inputs in the model. In the second option, each instance's temperature was modified by multiplying the actual corresponding temperature by a random coefficient ranging between [0.8 to 1.2]. For option 3 the range was amplified to [0.5 to 1.5] to introduce more error in the simulated *forecasted* temperature. The MAE in both new samples was 1.3 and 3.35 ºC respectively. Ideally, a more accurate simulation of a *forecasted temperatures* would better represent the influence of this parameter in the energy demand prediction. Nevertheless, to understand as a proof of concept, model 2 and model 3 should illustrate the mentioned influence.

The more accurate the temperature input that the model uses to give a prediction, the more accurate this will be. This implies that the model must rely on the temperature predictions of a third party. Table 12 represents the state of the art in temperature forecast according to ForecastWatch (ForecastWatch, 2017). The best prediction 1-3 Days out has an RMS of 3.3 ºF, which translated to ºC could mean that deviation from 1 to 2ºC can be commonly encountered when predicting the temperature one to three days ahead. When considering hiring a third party to use their forecast as inputs to the model, a reliable performance from their predictions should be aimed as will have strong influence over the model performance itself.

Table 12. RMS in Degrees Fahrenheit for U.S. Locations (ForecastWatch, 2017)

	1-3 Day Out	4-6 Day Out	1-9 Day out
Global Weather Online	3.347	5.071	7.046
AccuWeather	4.094	6.035	6.213
World Weather Online	5.099	7.115	7.159

All the results of the three modes are considered with perfect temperature forecast, which in reality is nearly impossible. Nevertheless, the purpose of these forecast is 24h to 48h, and therefore, there error in the forecast is minimized assuming that the temperature forecast is updated and given to the model. PREDICTION forecast model on the other hand has used forecasted temperatures in their prediction but no vector has been provided to make more legit comparisons.

5.2.5 Final Machine Learning Model Metrics

In this section the Random Forest model have been re-trained with the whole available data to date, that is, from 1st of January 2015 to 1st June 2017. An accurate working calendar has also been provided to the model.

Table 13. Error metrics for the proposed RF model. Training Set 70%. Test 30%. Period: 01/01/2015 – 01/06/2017

MAE (kWh)	2.88
MAPE (%)	9.33
MBPE (%)	-1.57
RMSE (kWh)	4.71
R^2	0,98

The final performance with all the data available shows how the model is learning and for the first time the main metric studied, MAPE, goes below 10%. The model is also more 0 centered as the MBPE shows the lowest possible deviation seen in the dissertation. The improvement in the metrics as the training data was added to the model leads to the conclusion that in order to predict the Energy Demand in disaggregated loads (building level) the minimum period of available data for the model should not be lower than 2 years.

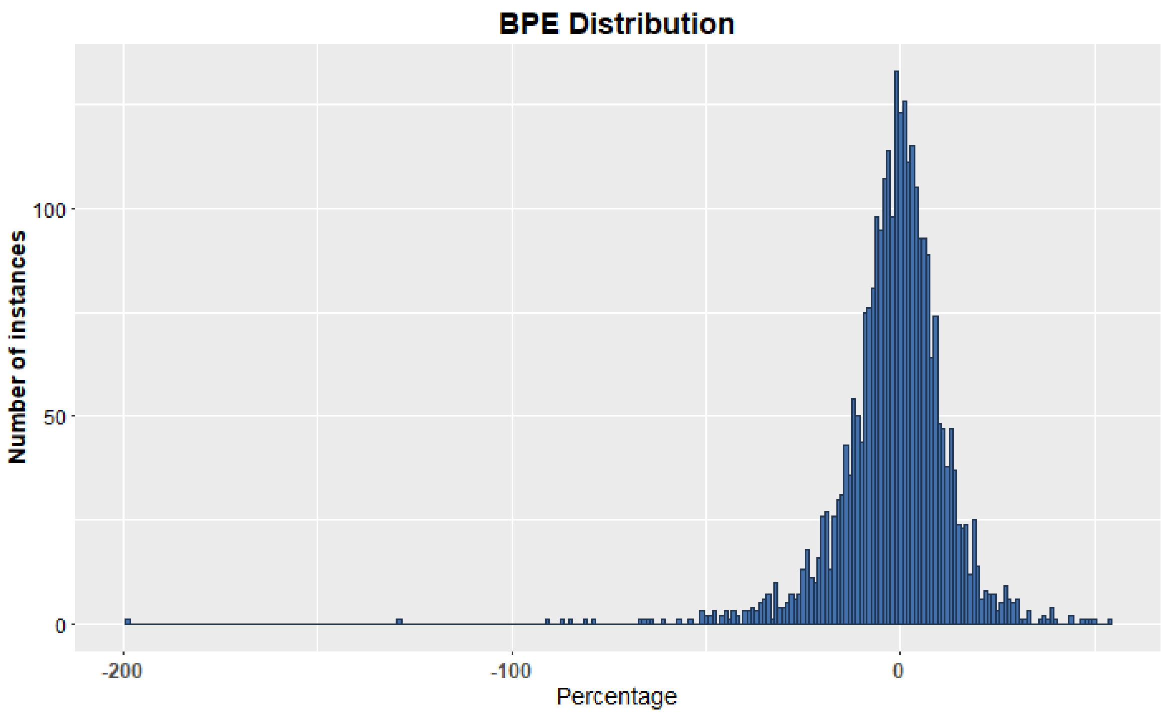
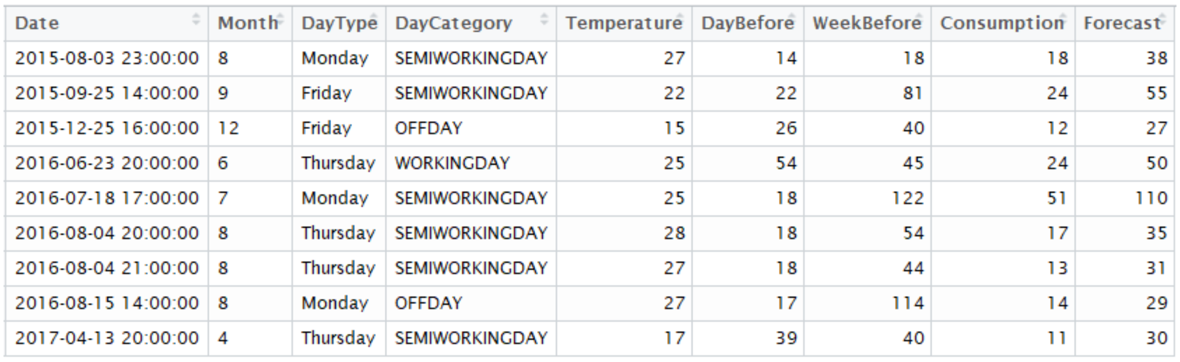


Figure 38. BPE Distribution for Final RF model

By filtering out the errors it is derived that there is room to improve the model. Some off the main error explanations come from the fact that when the algorithm checks the WeekBefore load (load same day of the previous week at the same time) it does not compute the category of that day from the previous week. For example, in Figure 39, at 14:00 of Friday 2015-09-25 the algorithm get confused because one day before was holiday and input variable Load 1 day before passes and abnormal value it would have been *WorkingDay.*

There are instances that have been wrong predicted and no reason has been found to cause the deviation. For example, in the first instance in Figure 39 there was an Energy demand prediction of 38 kWh at 23:00h in the night. There is nothing that might lead to such high value. Looking to the corresponding historic parameters they seem normal as represent low energy consumption in the night.

Many of these big deviations are also due to outliers in the consumption of the building. For example, in 2016-08-04, the week before there were two data points with abnormal energy consumption. At 20:00 of a Thursday this value should be lower and therefore, mistakes in the prediction would cannot be attributed to the algorithm.



Date	Month	DayType	DayCategory	Temperature	DayBefore	WeekBefore	Consumption	Forecast
2015-08-03 23:00:00	8	Monday	SEMIWORKINGDAY	27	14	18	18	38
2015-09-25 14:00:00	9	Friday	SEMIWORKINGDAY	22	22	81	24	55
2015-12-25 16:00:00	12	Friday	OFFDAY	15	26	40	12	27
2016-06-23 20:00:00	6	Thursday	WORKINGDAY	25	54	45	24	50
2016-07-18 17:00:00	7	Monday	SEMIWORKINGDAY	25	18	122	51	110
2016-08-04 20:00:00	8	Thursday	SEMIWORKINGDAY	28	18	54	17	35
2016-08-04 21:00:00	8	Thursday	SEMIWORKINGDAY	27	18	44	13	31
2016-08-15 14:00:00	8	Monday	OFFDAY	27	17	114	14	29
2017-04-13 20:00:00	4	Thursday	SEMIWORKINGDAY	17	39	40	11	30

Figure 39. Biggest deviations (APE > 100) for the final ML model

5.2.6 **Comparison with PREDICTION Model**

With the RF model re-trained as shown in previous section a comparison with the current PREDICTION model being implemented is carried out for the month of June 2017. The metrics results are shown in Table 14 and a representation of the two models predictions compared to the actual consumption is presented in Figure 40.

Table 14. JUNE 2017 comparison. RF vs PREDICTION

	MAPE %	**MBPE (%)**	**RMSE (kWh)**	**R^2**
Random Forest	11.77	- 3.10	6.40	0.97
PREDICTION	14.44	5.51	12.74	0.88

As in the rest of the analysis the RF outperforms the PREDICTION model in all the metrics. Furthermore, PREDICTION model fails on the 5th day of the month as it yields 0 value predictions. As it was explained in Part I of this thesis, these values were removed from the dataset before proceeding with the evaluation and the metrics do not consider them. The stability in this case is 97%.

When comparing the two models it must be considered that PREDICTION Models is an *Actual* implementation whereas RF is a *simulation.* PREDICTION is a software hardware integrated solution in which there can be several causes that makes the models fails. RF is a software model working solely over a dataset.

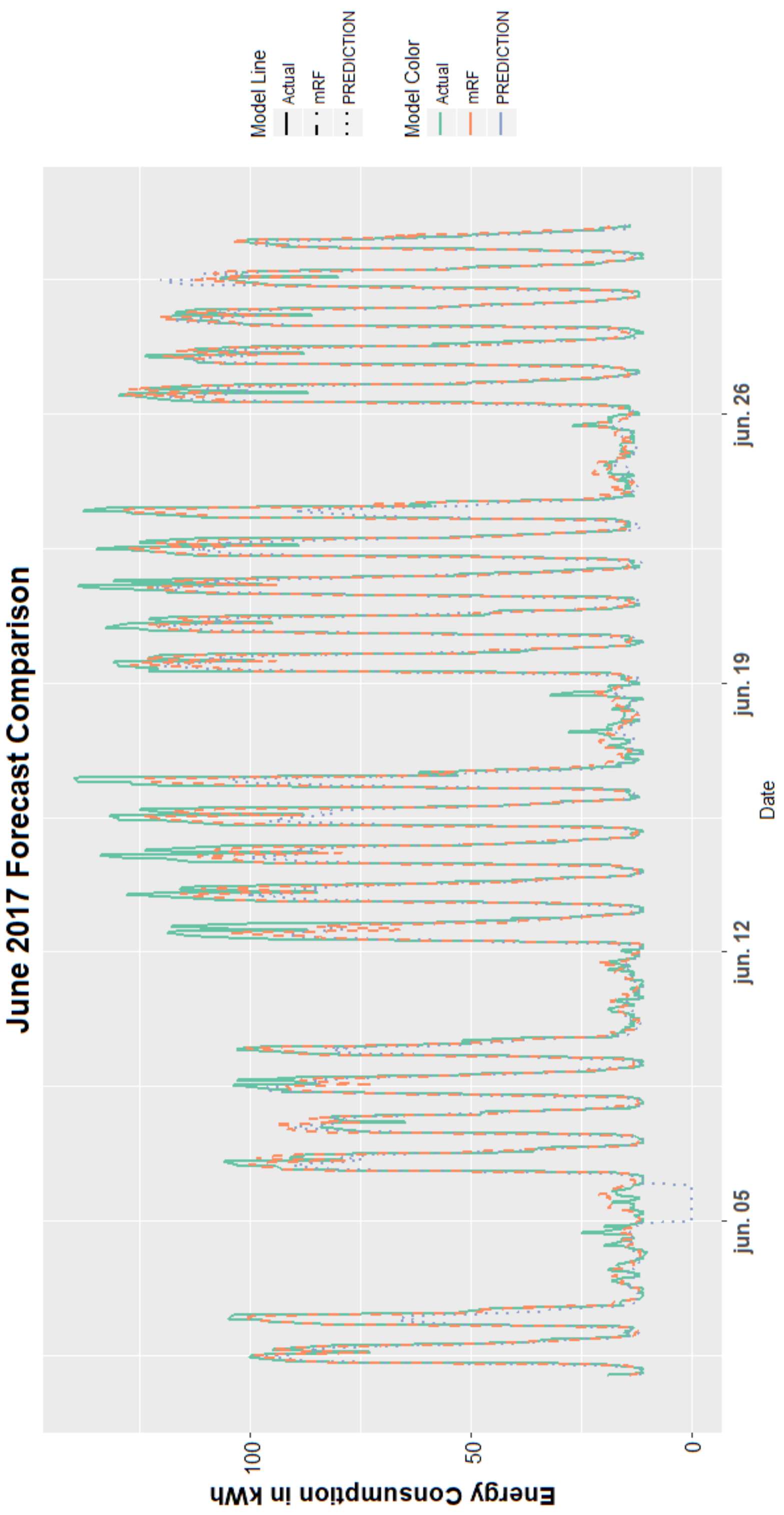


Figure 40. RF model vs PREDICTION. June 2017

5.3 ML methodology Summary

Analogously to Part I, a schematic of the process applied in the Part II is shown in Figure 41. A general description of an Energy Demand Forecast process is given. Steps are broken own and associated to corresponding application to the case-study of COMSA Headquarters.

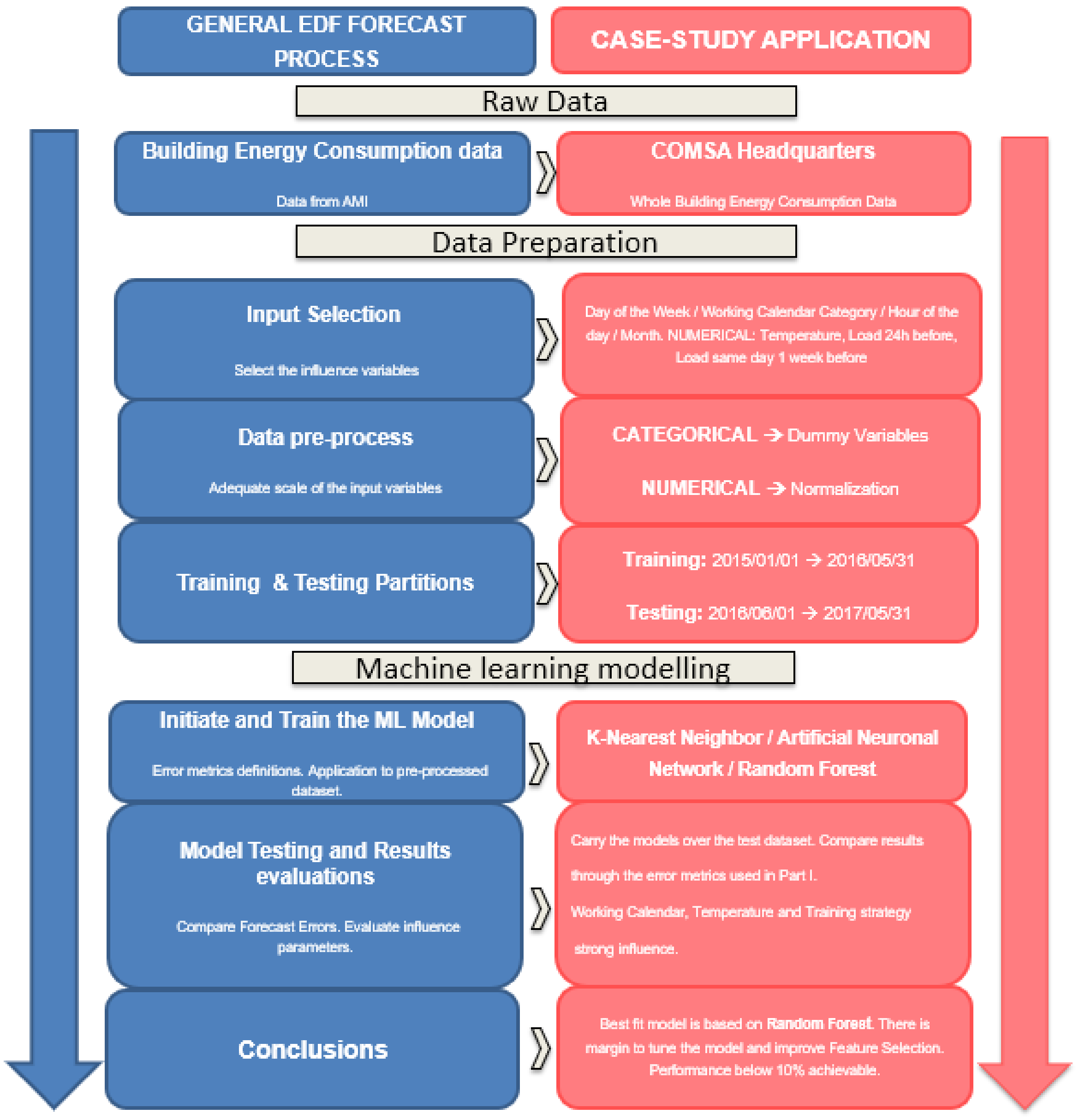

GENERAL EDF FORECAST PROCESS
CASE-STUDY APPLICATION
Raw Data
Building Energy Consumption data
Data from AMI
COMSA Headquarters
Whole Building Energy Consumption Data
Data Preparation
Input Selection
Select the influence variables
Day of the Week / Working Calendar Category / Hour of the day / Month. NUMERICAL: Temperature, Load 24h before, Load same day 1 week before
Data pre-process
Adequate scale of the input variables
CATEGORICAL → Dummy Variables
NUMERICAL → Normalization
Training & Testing Partitions
Training: 2015/01/01 → 2016/05/31
Testing: 2016/06/01 → 2017/05/31
Machine learning modelling
Initiate and Train the ML Model
Error metrics definitions. Application to pre-processed dataset.
K-Nearest Neighbor / Artificial Neuronal Network / Random Forest
Model Testing and Results evaluations
Compare Forecast Errors. Evaluate influence parameters.
Carry the models over the test dataset. Compare results through the error metrics used in Part I. Working Calendar, Temperature and Training strategy strong influence.
Conclusions
Best fit model is based on Random Forest. There is margin to tune the model and improve Feature Selection. Performance below 10% achievable.

Figure 41. Schematic of the Forecast Process Followed

5.4 Conclusions Part II

In the second part of this report three different Machine Learning algorithms have been applied to predict the Energy demand of the COMSA Headquarters office building. K-Nearest Neighbors, Artificial Neural Network and Random Forest based models in their regression versions have been trained and tested over the Energy consumption dataset from the building. With a granularity of hourly predictions and forecast time horizons from 24h to 48h the models were trained with 1 year and a half data points and tested in the following 1 year and 1 month. The results yield that the most suitable model is the model based in Random Forest algorithm (MAPE of 13.87%), then the ANN (MAPE of 16.21%) and lastly the K-NN (MAPE of 19.55%).

The error analysis helped detecting wrong labelled instances regarding the influence parameter *DayCategory,* which indicates if the corresponding day is normal working day, holiday or it has special reduced schedule. By correcting these instances and properly labeling them according to the specific building working a notable improvement in the performance of the model was achieved. The RF model was tested with no holidays at all but weekends and then, all the holidays and special days were included. The model showed an improvement with the accurate working calendar, from 19.78% to the above-mentioned 13.87%.

Once the calendar was corrected, two batches of new instances representing periods of approximately 3 months each, were used to retrain the RF model. The updated models showed improvements in all the metrics, reducing the MAPE by 1.26% and 1.81% respectively. The re-training strategy is an important parameter of any ML forecast system as it can improve the accuracy of the model and keeping a more precise output distribution.

All the prediction models incorporate the temperature as an important parameter of influence. In the previous test, the temperature used to predict the Energy consumption was not a forecasted one but the actual on. To simulate the real input that these models would use, the temperature input vector was modified to introduce some randomness. The accuracy of the models dropped when their temperature inputs are not perfect. The studies show that temperature forecast will affect the prediction model performance by 1 to 3%. Therefore, if the model imports the weather forecast from a third party, a good track of its performance should be considered, as they have direct influence over the results.

Finally, through the error analysis, the biggest errors (APE > 100%) were studied. No reasonable explanation has been found but to assign the cause of the error to strange behavior in the forecast or to abnormalities in the energy consumption data.

6 Final Conclusions

This master thesis has worked on two main lines regarding the EDF in disaggregated loads. In the first part, a methodology to evaluate the performance of EDF models was presented. The evaluation process was demonstrated over a forecast model currently making energy predictions in a office building placed in Barcelona. In the second part, three different ML models were studied as EDF alternatives for the same building. With the historical data and other influence variables these models were trained and tested over the same time periods.

The assessment of an EDF model is proposed as a data analytics process which requires a comprehensive understanding of dynamics associated to energy consumption of an office building. By data and expertise the field different abdormalities typologies were found: Zero and below the BBL predictions, extreme outliers (even with negative values), working calendar contradictions, etc. These abnormalities are assumed to represent instability in the model and are not computed in the accuracy analysis. With a 12.04% of invalid instances, final metrics of the model are MAPE of 18.01%, a MBPE of – 4.67 %, a RMSE of 11.65 kWh and a R^2 of 0.87. By studying the abnormalities, it is possible to improve the model. First way would be to include an accurate working calendar of the company using the building. The biggest deviations were detected in the periods where the variability of the working activity in high, that is, in entrance and departure times and in special schedule days. To help reduce this variability, the addition of an *occupancy* input would play a crucial role. Furthermore, a thorough analysis of outliers in the consumption data could improve the metrics as these intrinsic abnormalities can lead to deviations that should not be attributed to the model but to the input data.

To study alternative ML models to work over the same EDF problem three different methods, k-NN, ANN and RF, have been tested. RF was selected as the most suitable alternative. Its analysis, in similar conditions to the model studied in the PART I, resulted in MAPE of 13.87%, a MBPE of – 8.35 %, a RMSE of 5.49 kWh and a R^2 of 0.97. It has been demonstrated that these metrics can be improved in two separate ways: correcting the company working calendar and re-training the model with new data. Other optimization measures are related to training period (a minimum of 2 years period is recommended), feature selection and model parametrization. For weather variables used as inputs it is important that their values are accurate because they directly affect the performance of the model. With all the implementations the model reached MAPE of 9.33%, a MBPE of – 1.57 %, a RMSE of 4.71 kWh and a R^2 of 0.98. Nevertheless, there have been found instances which errors cannot be explained, they are mainly based on strange behavior of the forecast model or in abnormalities in the input data that should be further investigated.

7 Future work

This master thesis proposes a methodology for Energy Demand Forecasting model evaluation. The first results achieve its objective of measuring the model accuracy and finding when they tend to deviate more. Nevertheless, the methodology is in an early stage state and there is room for improvement.

As for the methodology a more careful analysis on smooth outliers embedded in the input that should be included in the analysis. Furthermire, the metthodology must incorporate an implementation to study the influence of the weather variable over the accuracy of the forecast model. The interest point would be on how the algorithm reacts to sudden change in the weather conditions, such as cold or hot spells. In the near future the objective is to improve, debug and clean the code to make it a fully scalable to allow the comparison of any forecast of any building in an analogous way. This will allow comparison between performances of the same model in different buildings, or comparisons of different models over the same building. Medium-Term wise the idea is to integrate the solution in a software platform and, through a friendly User Interface, enable fast comparisons of the available portfolio of building and models.

Regarding the application of particular configured ML algorithms there is even more space to improvement. In this master thesis *default* models have been applied and tested but there is room for optimization. From the feature selection point of view, adding the occupancy parameters is the first priority but other parameters will probably lead to better metrics, as increasing the number of weather variables like humidity or solar irradiation. Another Autoregressive parameter could also be added and tried in the model, such us the average load of the previous day, previous week or previous month. Regarding model optimization it can be done by configuring model parameters, as the number of hidden layers in the ANN model or testing other Neural Network based models. Designing an automated training batch strategy is also important as the retraining has been proved to improve the accuracy and adequate the output model distribution. In addition, split the models into categories, as for example creating one model for weekdays and another for weekends could increase the accuracy. Furthermore, combining the different ML technologies models into what is known in the fiend as Ensemble models (e.g. RF) must be investigated as weighing the models can lead to improvement in the dataset.

Finally, it will be interesting to scale up the predictions and study the effect of aggregated Loads by combining different buildings into clusters and study and predict their energy demand as a whole.

Acknowledgement

The ideation of this master thesis started seven months ago and now, close to its completion, it is time to write a thankful message to all of you that helped me in this long journey.

I would like first to thank my family because they have always been supportive and I would have not started this master and this project without their help. They have provided me with experience and advice, and they have always been a reference on how to tackle any problem I have faced, not just pure technical. Thanks to Antonio, Rosa, Fer, Richi, Isa, María, David and, of course, Olivia.

Thanks to Innoenergy that has given me the opportunity to enroll in this master. They allowed to me to live an enrichment 2 years' experience. Thanks because I have been able to meet many friends (some more redheaded and special than others), extend my network and travel thousands of kilometers around Europe.

At the same time, this work could have never been completed without the help of COMSA Industrial and its R&D in Energy and Systems department. Special thanks to Albert and Merche for offering me the opportunity to work with their data and for given me time and advice to proceed with the dissertation.

On the academic side I would like to thank to CITCEA for their support as the Innoenergy representatives at UPC. Thanks to Ingrid and Andreas for their time during this year at the university.

References

Aggarwal, C. (n.d.). *Outlier Analysis*.

Allison, P. (2009). *Missing data*. Thousand Oaks, Calif.: Sage Publications.

Blog, D. (2017). *3 Reasons to Learn Caret*. [online] R-bloggers. Available at: https://www.r-bloggers.com/3-reasons-to-learn-caret/ [Accessed 27 Aug. 2017].

Boroojeni, K., Amini, M., Bahrami, S., Iyengar, S., Sarwat, A. and Karabasoglu, O. (2017). A novel multi-time-scale modeling for electric power demand forecasting: From short-term to medium-term horizon. *Electric Power Systems Research*, 142, pp.58-73.

Bradley, J. and Amde, M. (2015). *Random Forests and Boosting in MLlib*. [online] Databricks. Available at: https://databricks.com/blog/2015/01/21/random-forests-and-boosting-in-mllib.html [Accessed 22 Jul. 2017].

Chambers, J., Eddy, W. and Härdle, W. (2002). *Modern Applied Statistics with S*. New York: Springer New York.

Chen, C., Twycross, J. and Garibaldi, J. (2017). A new accuracy measure based on bounded relative error for time series forecasting. *PLOS ONE*, 12(3), p.e0174202.

Donders, A., van der Heijden, G., Stijnen, T. and Moons, K. (2006). Review: A gentle introduction to imputation of missing values. *Journal of Clinical Epidemiology*, 59(10), pp.1087-1091.

Edwards, R., New, J. and Parker, L. (2012). Predicting future hourly residential electrical consumption: A machine learning case study. *Energy and Buildings*, 49, pp.591-603.

FAMILI, A., SHEN, W., WEBER, R. and SIMOUDIS, E. (1997). Data preprocessing and intelligent data analysis. *Intelligent Data Analysis*, 1(1-4), pp.3-23.

Fan, C., Xiao, F. and Wang, S. (2014). Development of prediction models for next-day building energy consumption and peak power demand using data mining techniques. *Applied Energy*, 127, pp.1-10.

Forbes.com. (2017). *Forbes Welcome*. [online] Available at: https://www.forbes.com/sites/bernardmarr/2016/12/06/what-is-the-difference-between-artificial-intelligence-and-machine-learning/#1a0f3e812742 [Accessed 4 Aug. 2017].

ForecastWatch (2017). *Six-Month Analysis of High Temperature Forecasts*. [online] Available at: http://www.forecastwatch.com/static/Six_Month_High_Temperature_Study_2013.pdf [Accessed 6 Jul. 2017].

Gillies, D., Bernholtz, B. and Sandiford, P. (1956). A New Approach to Forecasting Daily Peak Loads [includes discussion]. *Transactions of the American Institute of Electrical Engineers. Part III: Power Apparatus and Systems*, 75(3).

Hauth, J. (2008). *Grey-Box Modelling for Nonlinear Systems*. phD. Technische Universität Kaiserslautern.

Hawkins, D. (2014). *Identification of outliers*. [Place of publication not identified]: Springer.

Heinermann, J. and Kramer, O. (2016). Machine learning ensembles for wind power prediction. *Renewable Energy*, 89, pp.671-679.

Hernandez, L., Baladrón, C., Aguiar, J., Carro, B., Sanchez-Esguevillas, A., Loret, J. and Massana, J. (2014). A survey on Electric Power Demand Forecasting: Future Trends in Smart Grids, Microgrids and Smart Buildings. *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, 16(3).

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8), pp.2554-2558.

Hyde, O. and Hodnett, P. (1997). An adaptable automated procedure for short-term electricity load forecasting. *IEEE Transactions on Power Systems*, 12(1), pp.84-94.

Hyndman, R. and Koehler, A. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), pp.679-688.

Kim, S. and Kim, H. (2016). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, 32(3), pp.669-679.

Kissock, J., Reddy, T. and Claridge, D. (1998). Ambient-Temperature Regression Analysis for Estimating Retrofit Savings in Commercial Buildings. *Journal of Solar Energy Engineering*, 120(3), p.168.

Kodali, T. (2017). *Data Manipulation with dplyr*. [online] R-bloggers. Available at: https://www.r-bloggers.com/data-manipulation-with-dplyr/ [Accessed 27 Aug. 2017].

Lavrenko, V. (2017). *kNN.8 Nearest-neighbor regression example*. [online] YouTube. Available at: https://www.youtube.com/watch?v=3lp5CmSwrHI [Accessed 13 Sep. 2017].

Li, S. (2017). *CRAN - Package FNN*. [online] Cran.r-project.org. Available at: https://cran.r-project.org/web/packages/FNN/index.html [Accessed 10 Sep. 2017].

Liaw, A. and Wiener, M. (2017). *CRAN - Package randomForest*. [online] Cran.r-project.org. Available at: https://cran.r-project.org/web/packages/randomForest/index.html [Accessed 10 Sep. 2017].

Lora, A., Santos, J., Exposito, A., Ramos, J. and Santos, J. (2007). Electricity Market Price Forecasting Based on Weighted Nearest Neighbors Techniques. *IEEE Transactions on Power Systems*, 22(3), pp.1294-1301.

Massana, J., Pous, C., Burgas, L., Melendez, J. and Colomer, J. (2015). Short-term load forecasting in a non-residential building contrasting models and attributes. *Energy and Buildings*, 92, pp.322-330.

Mehdiyev, N., Enke, D., Fettke, P. and Loos, P. (2016). Evaluating Forecasting Methods by Considering Different Accuracy Measures. *Procedia Computer Science*, 95, pp.264-271.

Neto, A. and Fiorelli, F. (2008). Comparison between detailed model simulation and artificial neural network for forecasting building energy consumption. *Energy and Buildings*, 40(12), pp.2169-2176.

Papalexopoulos, A. and Hesterberg, T. (1990). A regression-based approach to short-term system load forecasting. *IEEE Transactions on Power Systems*, 5(4), pp.1535-1547.

Perera, R. (2008). Research methods journal club: a gentle introduction to imputation of missing values. *Evidence-Based Medicine*, 13(1), pp.5-5.

Raza, M. and Khosravi, A. (2015). A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings. *Renewable and Sustainable Energy Reviews*, 50, pp.1352-1372.

Ripley, B. (2017). *Feed-Forward Neural Networks and Multinomial Log-Linear Models [R package nnet version 7.3-12]*. [online] Cran.r-project.org. Available at: https://cran.r-project.org/web/packages/nnet/index.html [Accessed 10 Sep. 2017].

Schaller, R. (1997). Moore's law: past, present and future. *IEEE Spectrum*, 34(6), pp.52-59.

Schlittgen, R. (2008). Robert H. Shumway and David S. Stoffer: Time series analysis and its applications with R examples, 2nd edn. *AStA Advances in Statistical Analysis*, 92(2), pp.233-234.

Stat.berkeley.edu. (2017). *Random forests - classification description*. [online] Available at: https://www.stat.berkeley.edu/~breiman/RandomForests/reg_home.htm [Accessed 10 Sep. 2017].

Sunku, S. (2017). *zoo time series exercises*. [online] R-bloggers. Available at: https://www.r-bloggers.com/zoo-time-series-exercises/ [Accessed 27 Aug. 2017].

Toyoda, J., Chen, M. and Inoue, Y. (1970). An Application of State Estimation to Short-Term Load Forecasting, Part I: Forecasting Modeling. *IEEE Transactions on Power Apparatus and Systems*, PAS-89(7), pp.1678-1682.

Treiber, N., Heinermann, J. and Kramer, O. (2013). Aggregation of features for wind energy prediction with support vector regression and nearest neighbors. *European Conference on Machine Learning, DARE Workshop.*, pp.22-30.

Tukey, J. (1977). Exploratory Data Analysis. *Biometrics*, 33(4), p.768.

Venables, W. and Ripley, B. (2002). *Modern applied statistics with S-Plus*. New York: Springer.

Wahid, F. and Kim, D. (2016). A Prediction Approach for Demand Analysis of Energy Consumption Using K-Nearest Neighbor in Residential Buildings. *International Journal of Smart Home*, 10(2), pp.97-108.

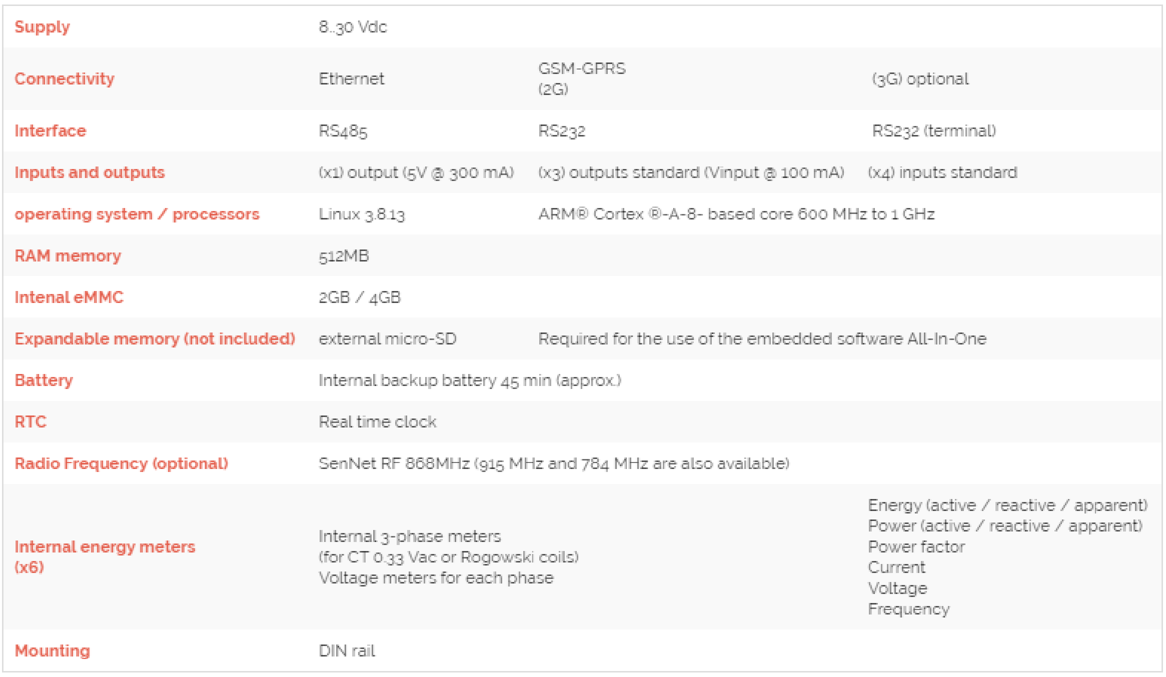
Yau, N. (2017). *How to Read and Use a Box-and-Whisker Plot*. [online] FlowingData. Available at: https://flowingdata.com/2008/02/15/how-to-read-and-use-a-box-and-whisker-plot/ [Accessed 1 Sep. 2017].

Yildiz, B., Bilbao, J. and Sproul, A. (2017). A review and analysis of regression and machine learning models on commercial building electricity load forecasting. *Renewable and Sustainable Energy Reviews*, 73, pp.1104-1122.

Zhao, H. and Magoulès, F. (2012). A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews*, 16(6), pp.3586-3592.

Zhou, Q., Wang, S., Xu, X. and Xiao, F. (2008). A grey-box model of next-day building thermal load prediction for energy-efficient control. *International Journal of Energy Research*, 32(15), pp.1418-1431.

ANNEX I: Hardware and Software specs.



Supply	8..30 Vdc		
Connectivity	Ethernet	GSM-GPRS (2G)	(3G) optional
Interface	RS485	RS232	RS232 (terminal)
Inputs and outputs	(x1) output (5V @ 300 mA)	(x3) outputs standard (Vinput @ 100 mA)	(x4) inputs standard
operating system / processors	Linux 3.8.13	ARM® Cortex ®-A-8- based core 600 MHz to 1 GHz	
RAM memory	512MB		
Intenal eMMC	2GB / 4GB		
Expandable memory (not included)	external micro-SD	Required for the use of the embedded software All-In-One	
Battery	Internal backup battery 45 min (approx.)		
RTC	Real time clock		
Radio Frequency (optional)	SenNet RF 868MHz (915 MHz and 784 MHz are also available)		
Internal energy meters (x6)	Internal 3-phase meters (for CT 0.33 Vac or Rogowski coils) Voltage meters for each phase		Energy (active / reactive / apparent) Power (active / reactive / apparent) Power factor Current Voltage Frequency
Mounting	DIN rail		

Figure 42. DL172 Datalogger specifications

ANNEX II: Evaluation Period Monthly raw data visualization

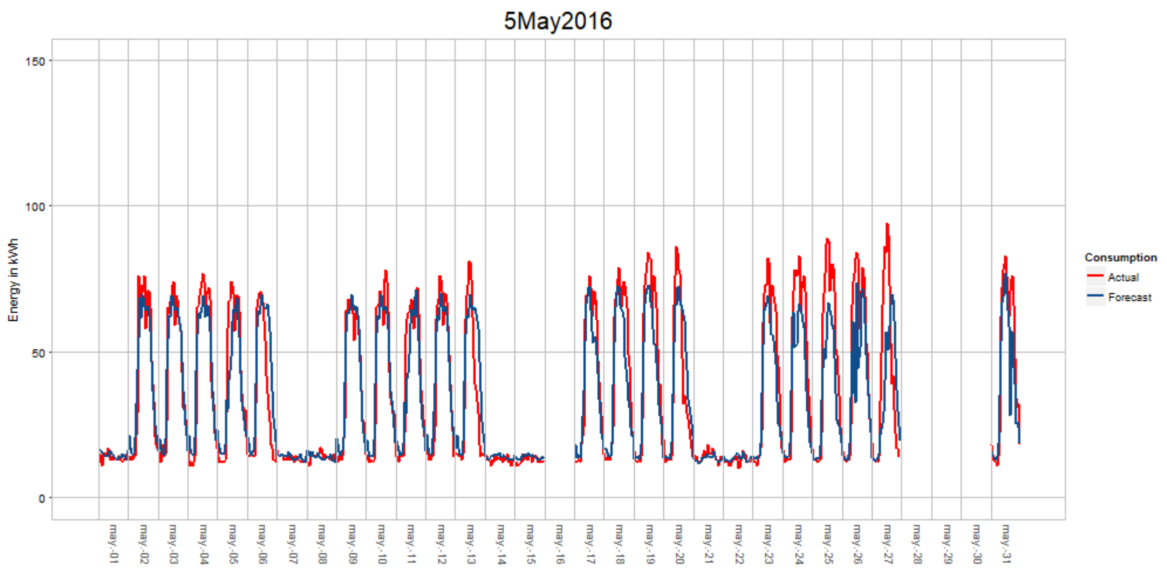


Figure 43. May 2016 Consumption before data cleanness process

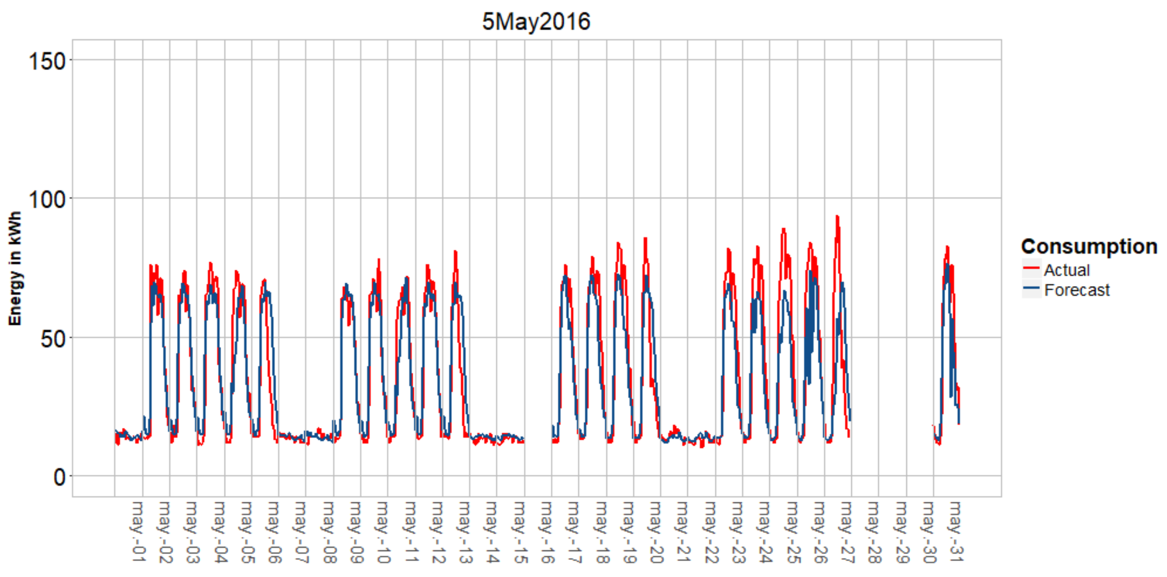


Figure 44. May 2016 Consumption after data cleanness process.

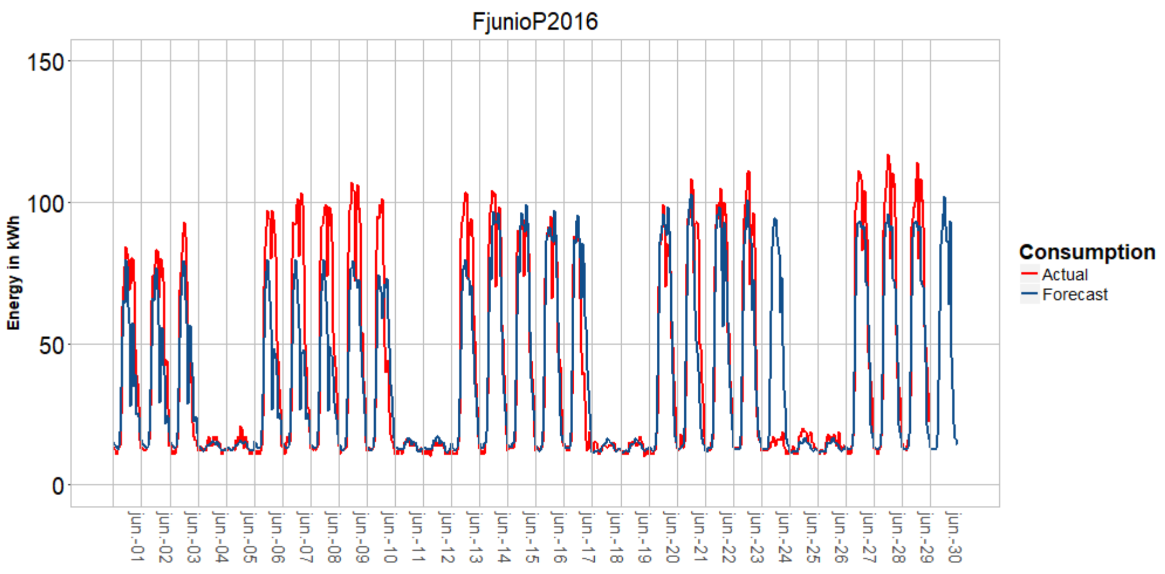


Figure 45. June 2016 Consumption before data cleanness process

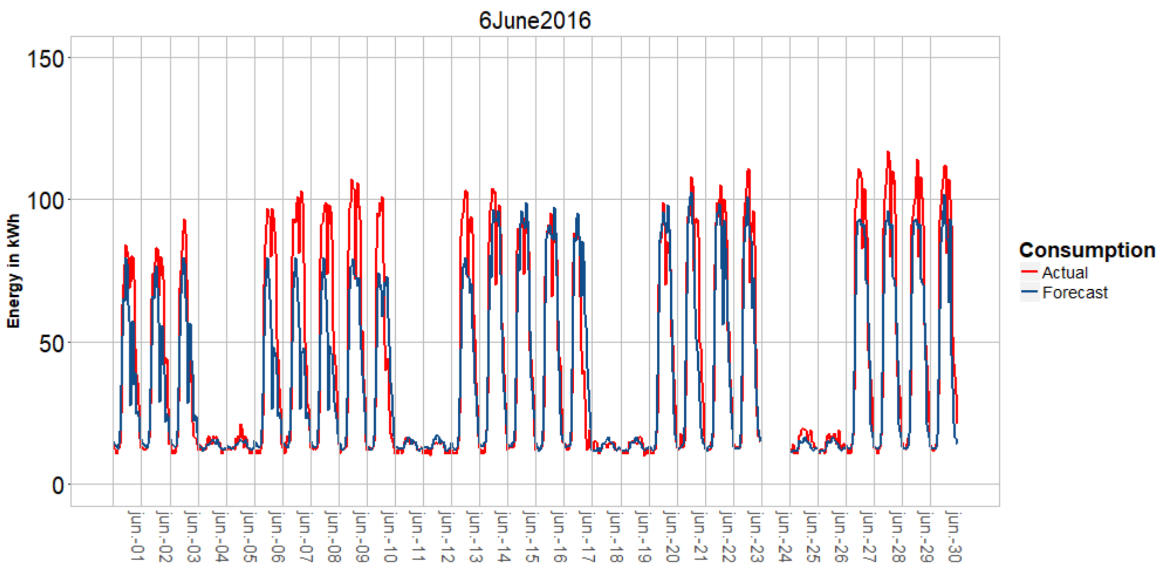


Figure 46. June 2016 Consumption after data cleanness process.

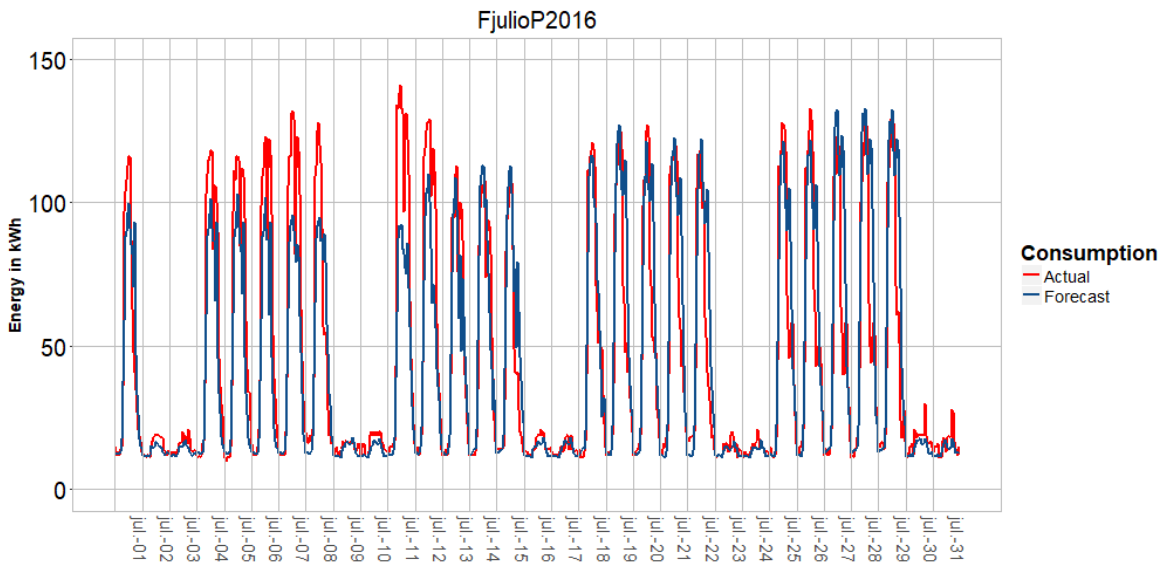


Figure 47. July 2016 Consumption before data cleanness process

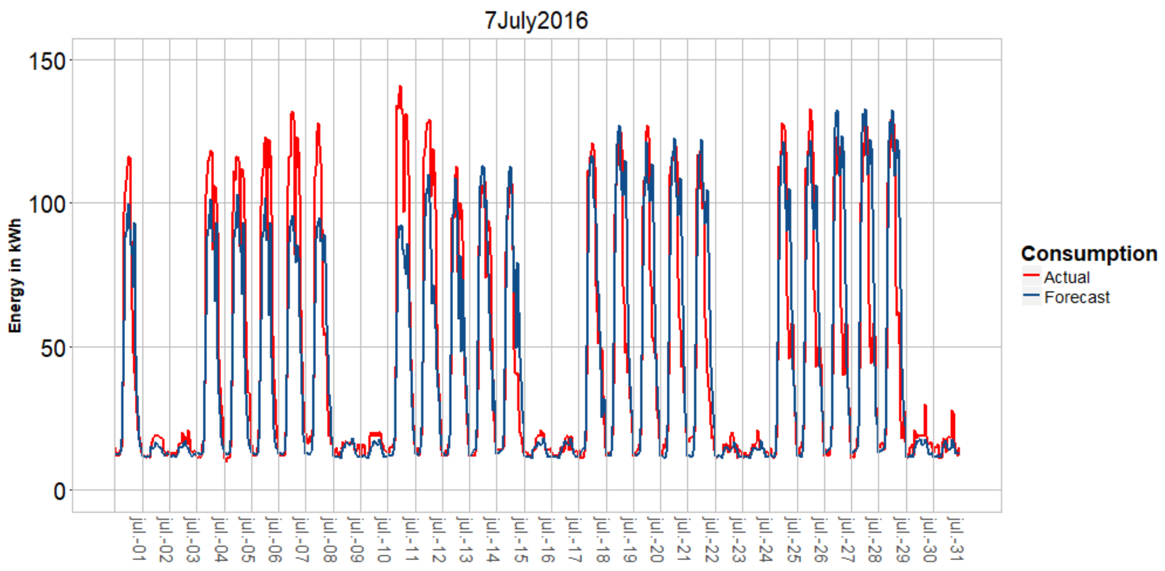


Figure 48. July 2016 Consumption after data cleanness process.

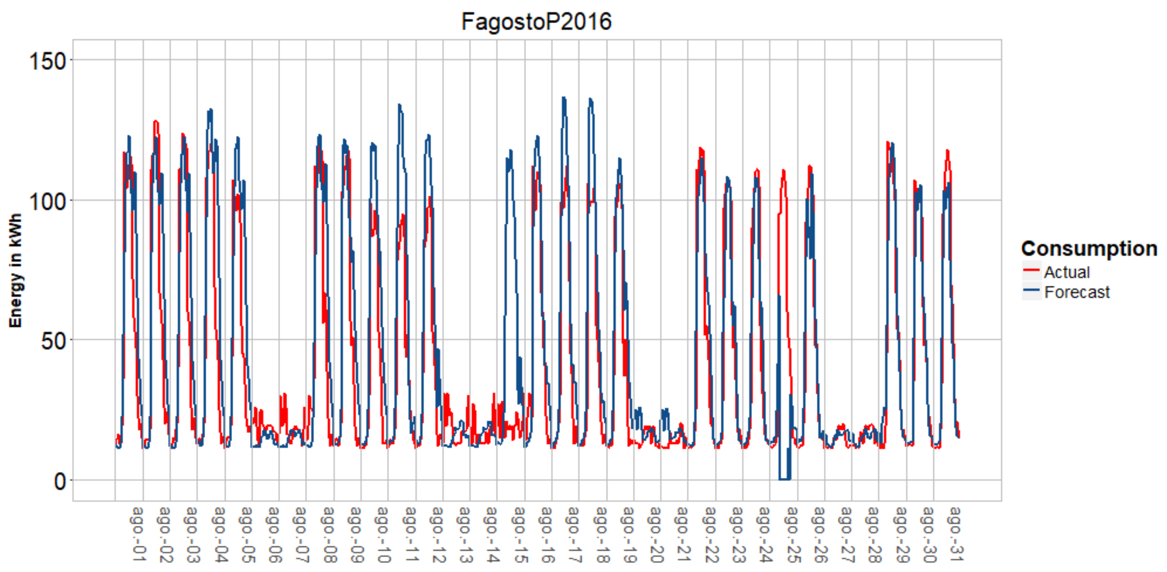


Figure 49. August 2016 Consumption before data cleanness process

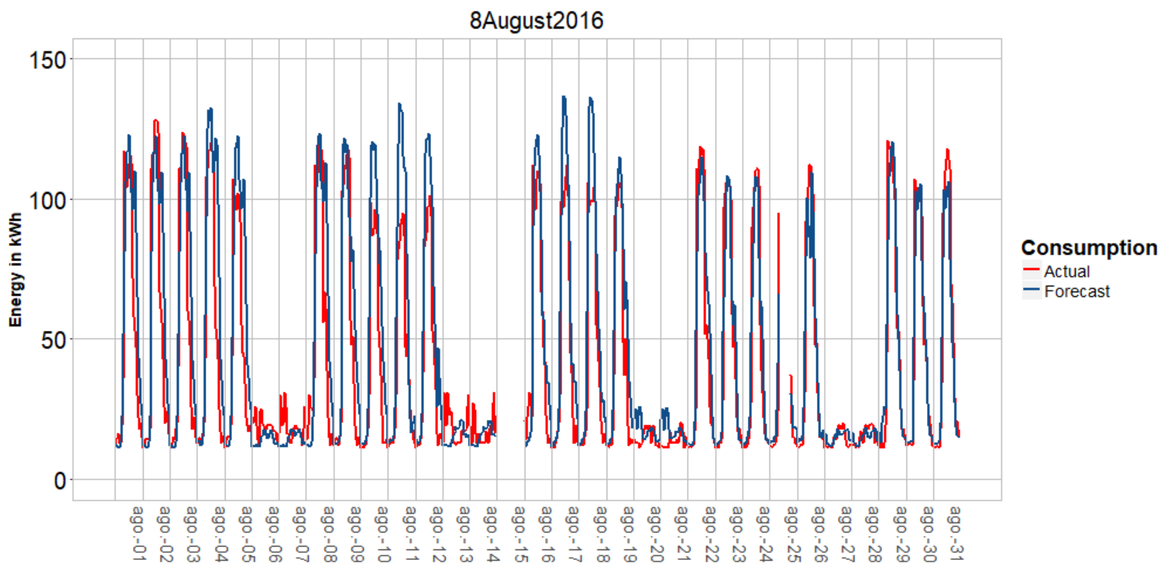


Figure 50. August 2016 Consumption after data cleanness process.

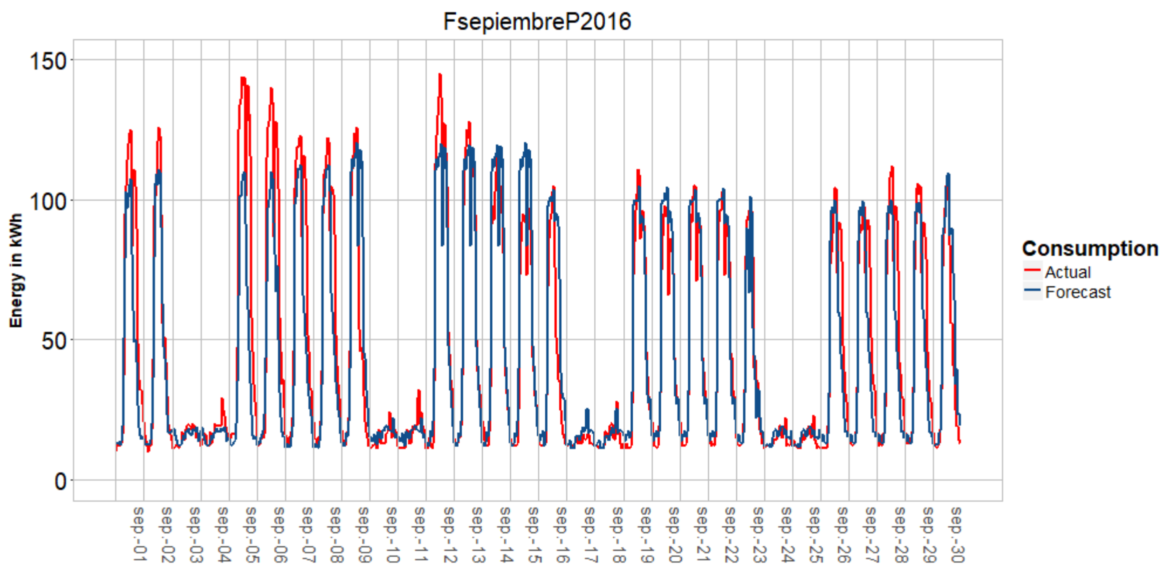


Figure 51. September 2016 Consumption before data cleanness process

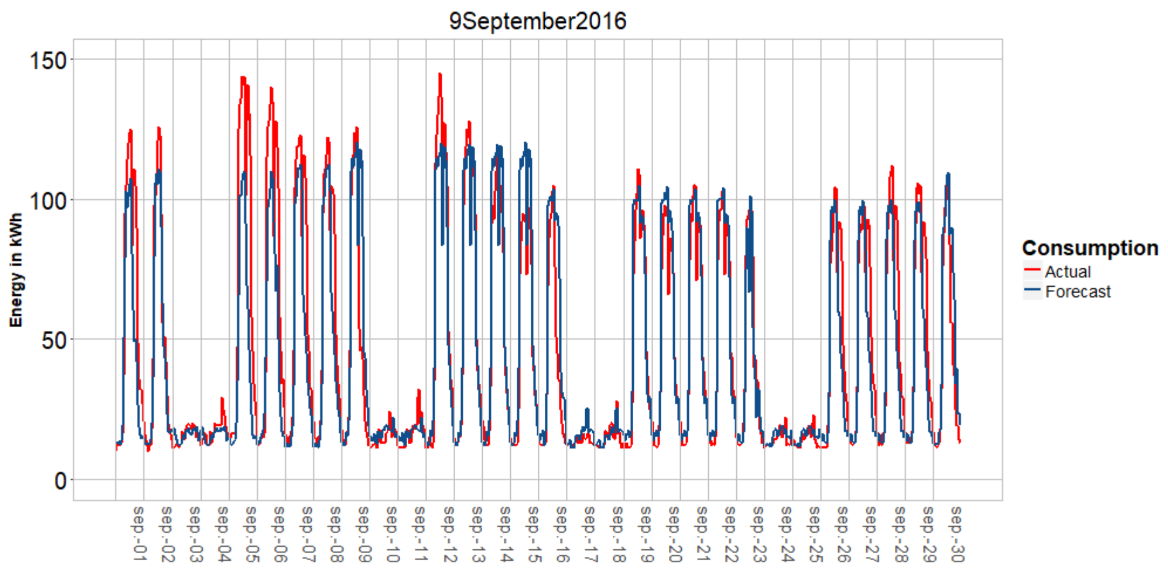


Figure 52. September 2016 Consumption after data cleanness process.

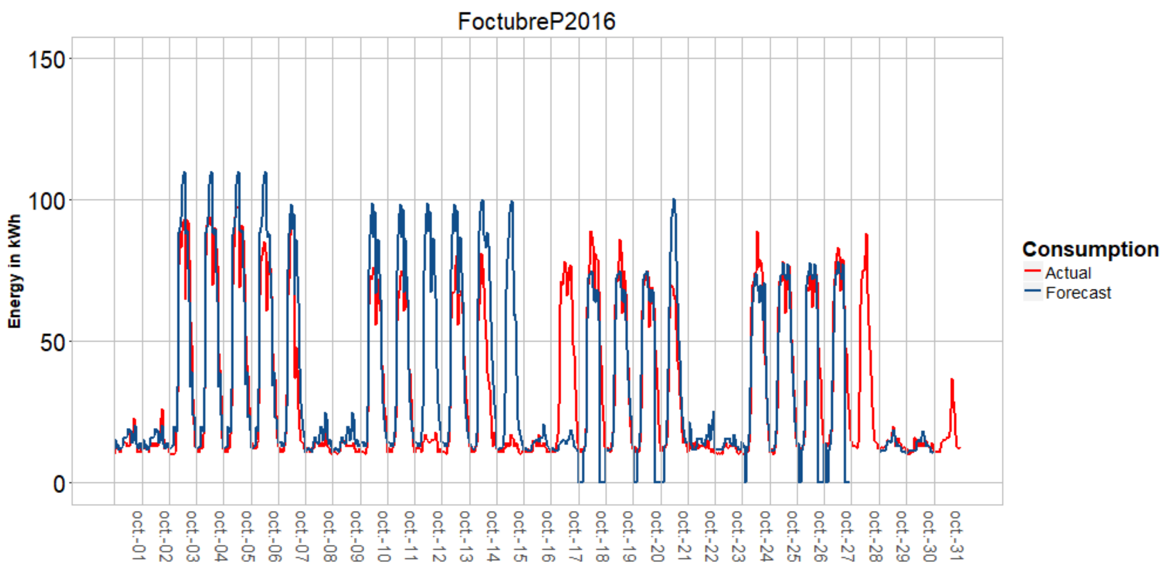


Figure 53. October 2016 Consumption before data cleanness process

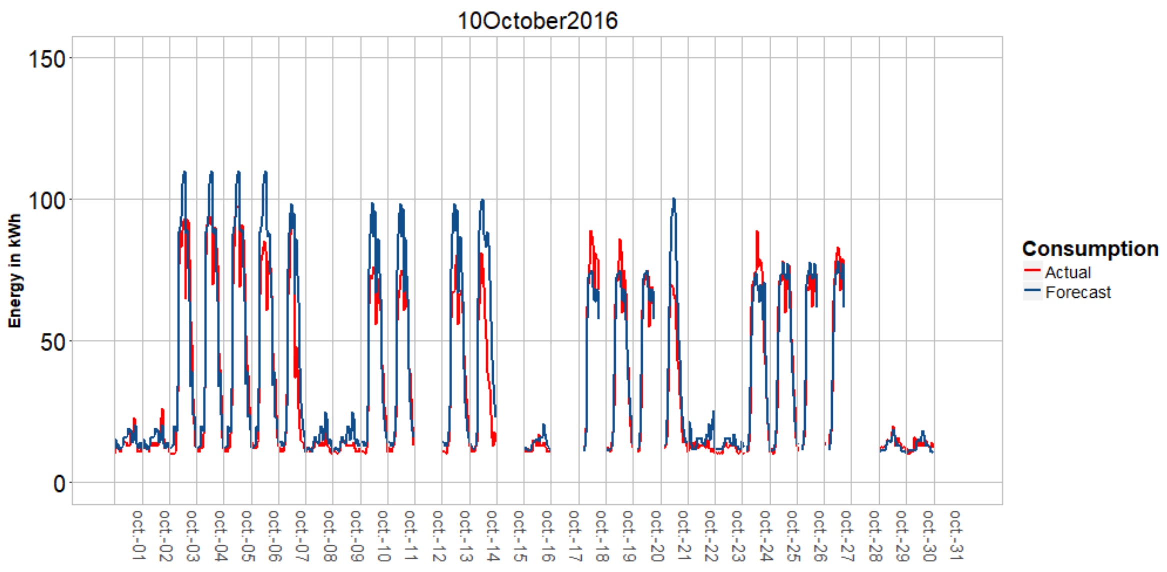


Figure 54. October 2016 Consumption after data cleanness process.

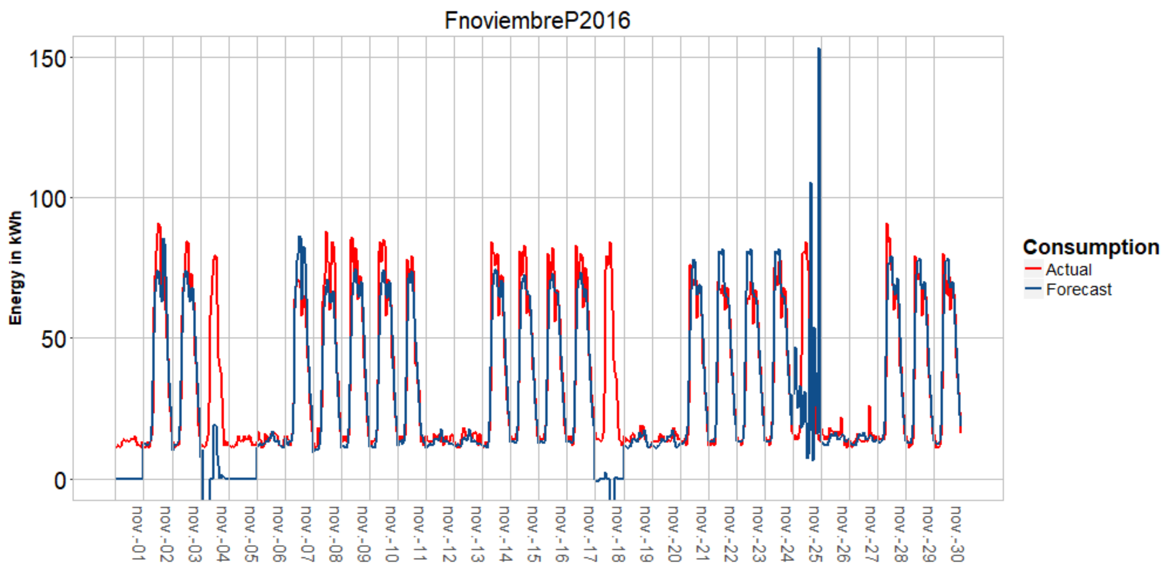


Figure 55. November 2016 Consumption before data cleanness process

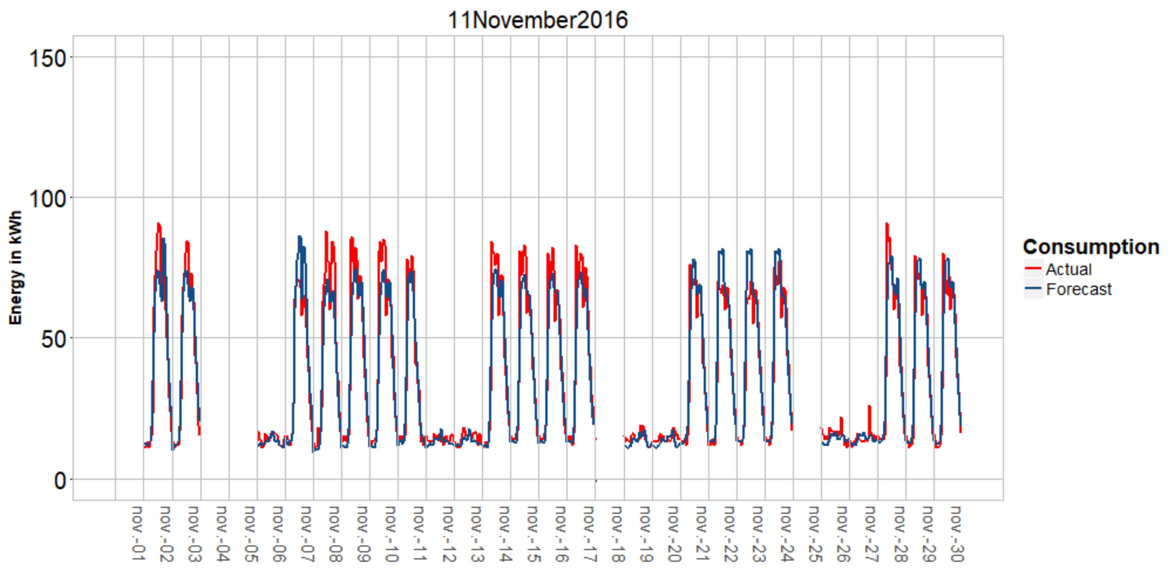


Figure 56. November 2016 Consumption after data cleanness process.

ETSEIB

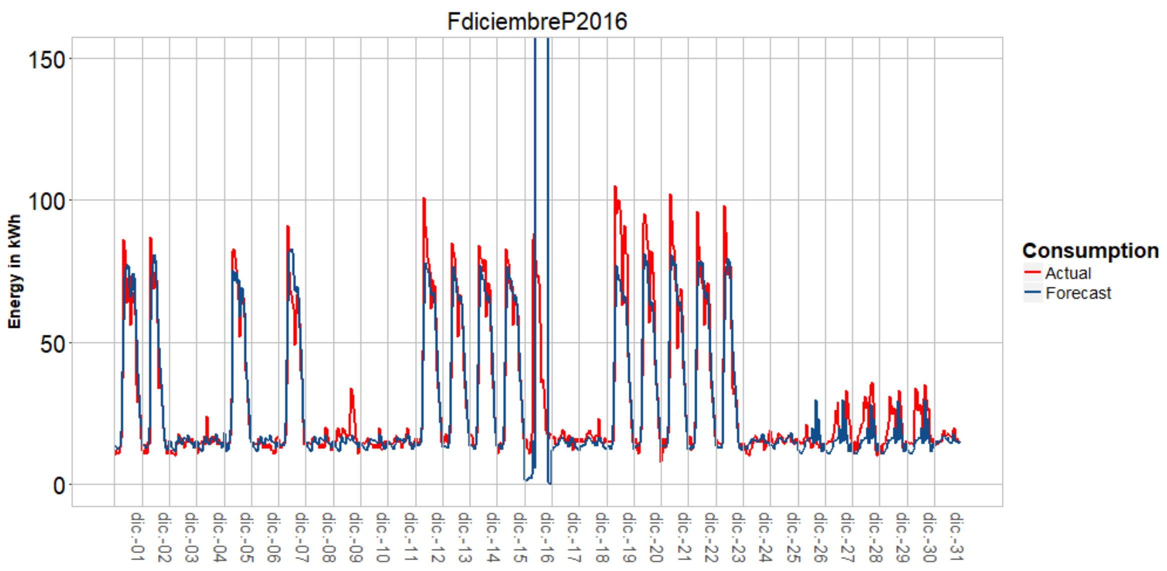


Figure 57. December 2016 Consumption before data cleanness process

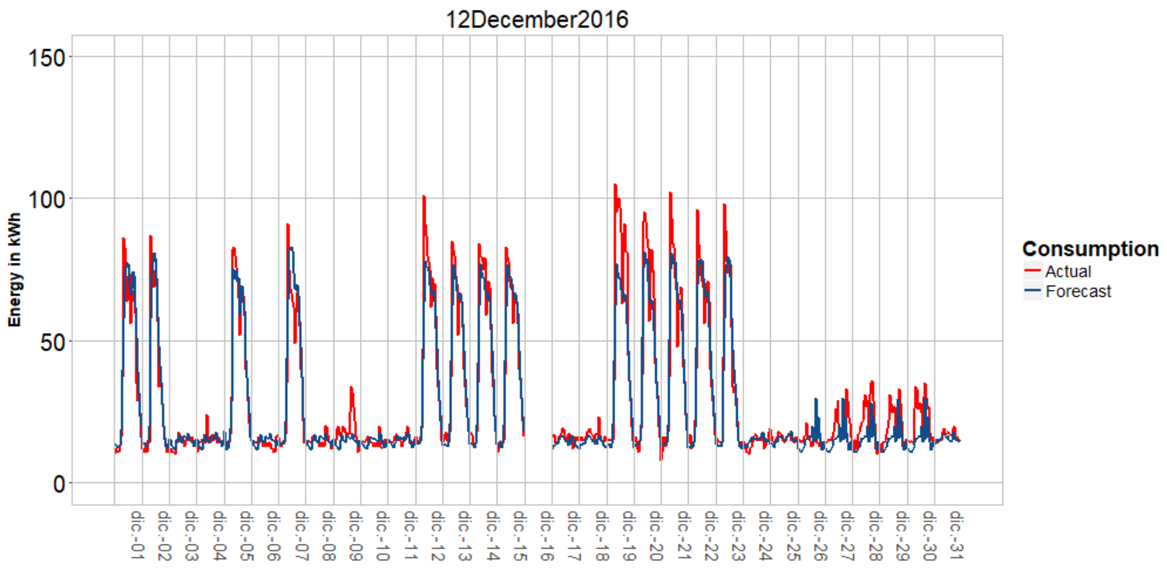


Figure 58. December 2016 Consumption after data cleanness process.

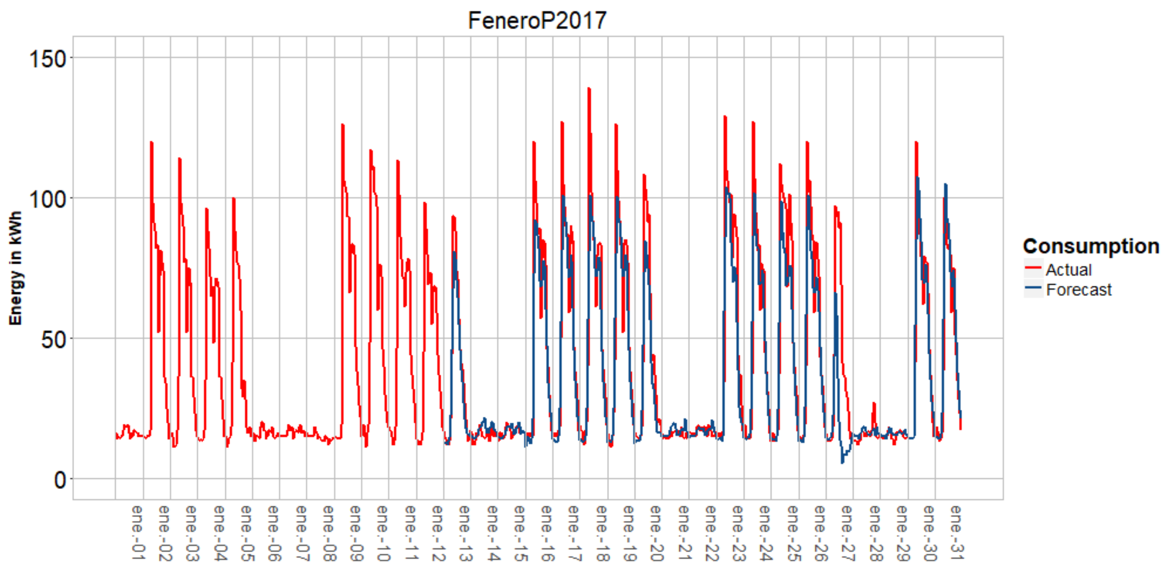


Figure 59. January 2017 Consumption before data cleanness process

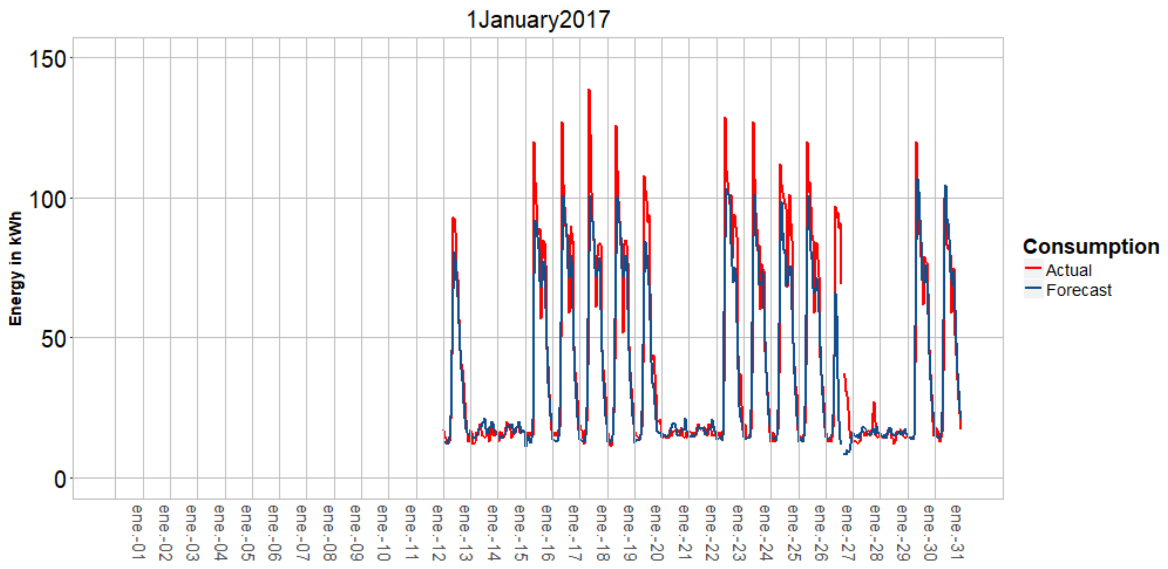


Figure 60. January 2017 Consumption after data cleanness process.

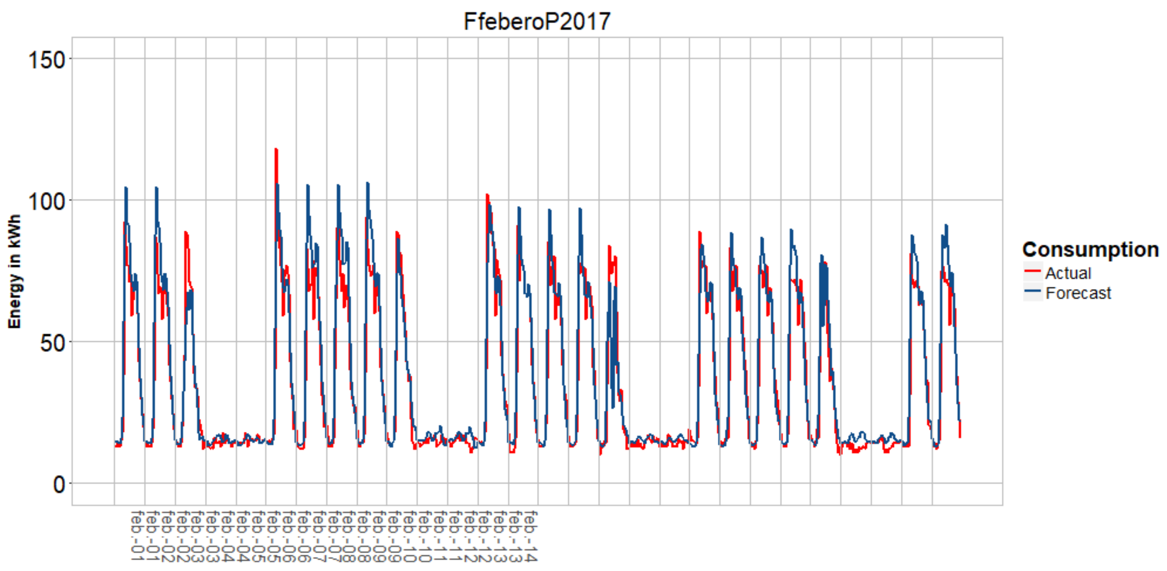


Figure 61. February 2017 Consumption before data cleanness process

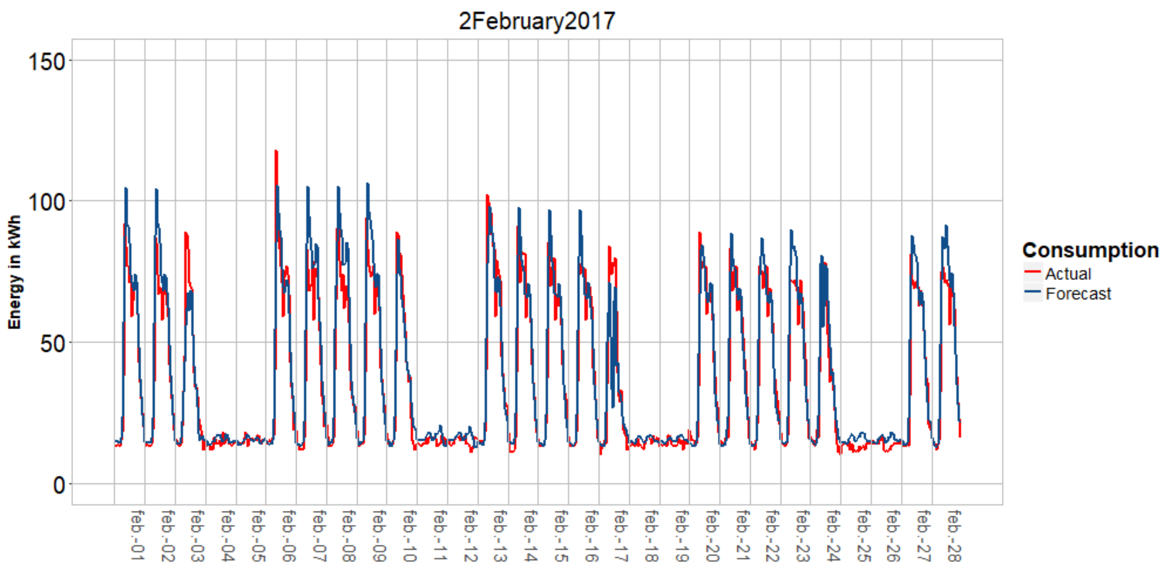


Figure 62. February 2017 Consumption after data cleanness process.

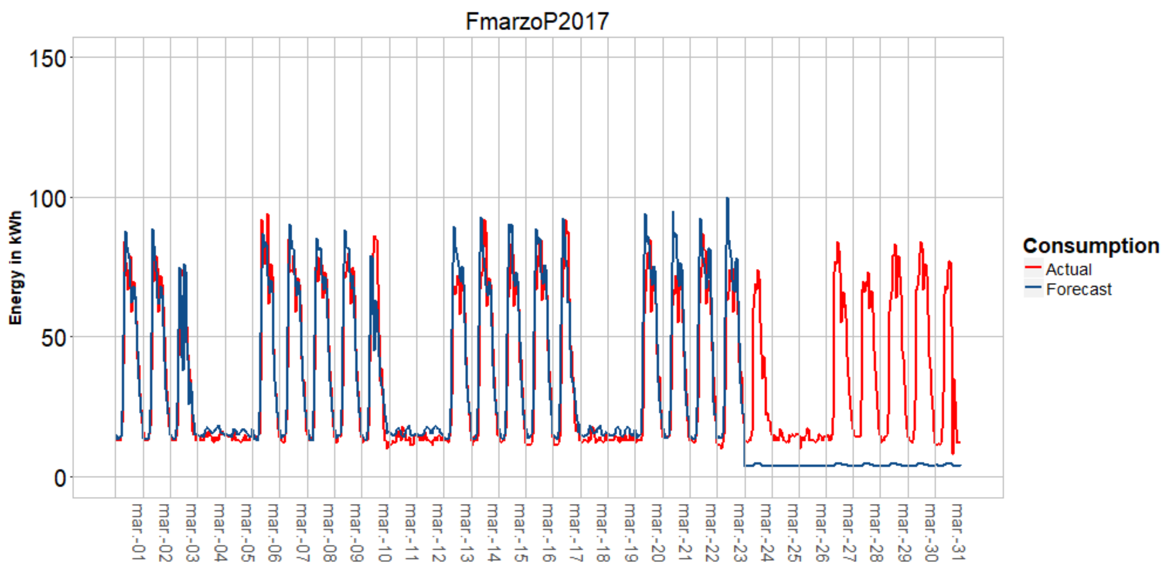


Figure 63. March 2017 Consumption before data cleanness process

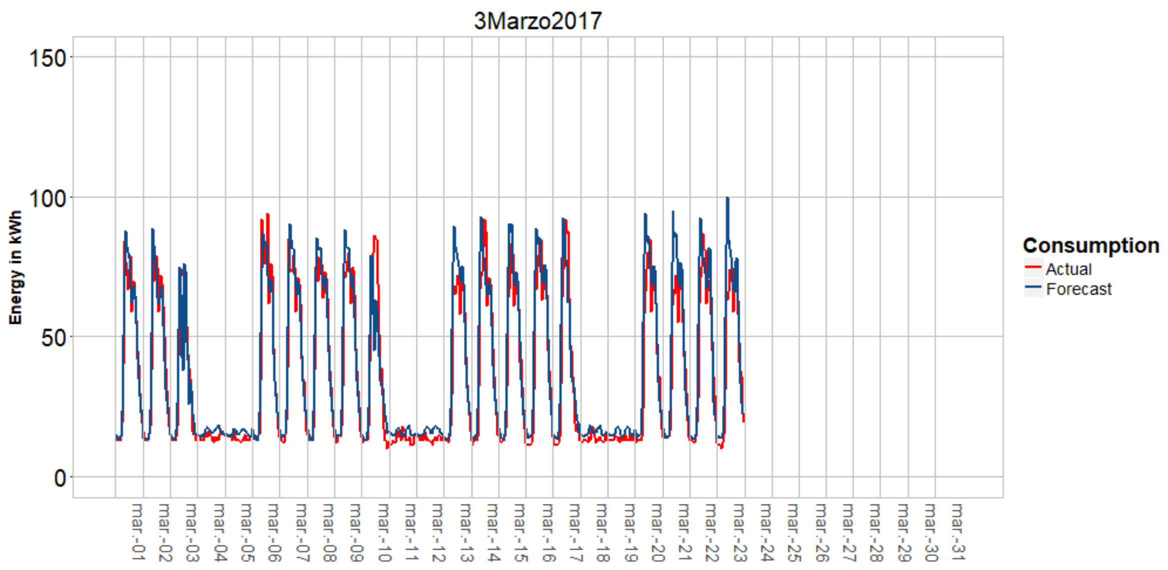


Figure 64. March 2017 Consumption after data cleanness process.

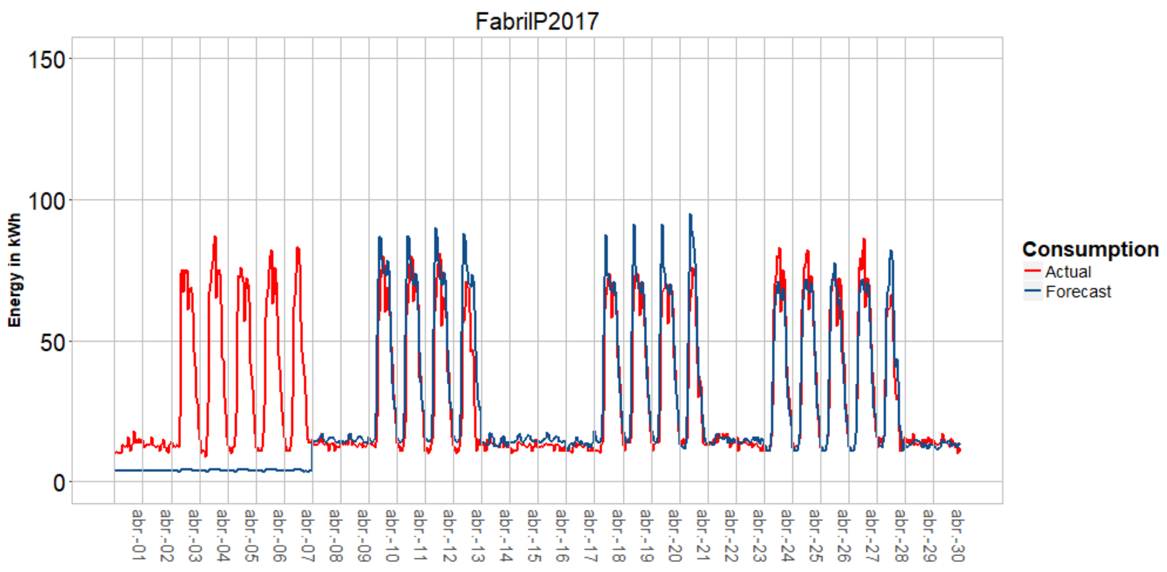


Figure 65. April 2017 Consumption before data cleanness process

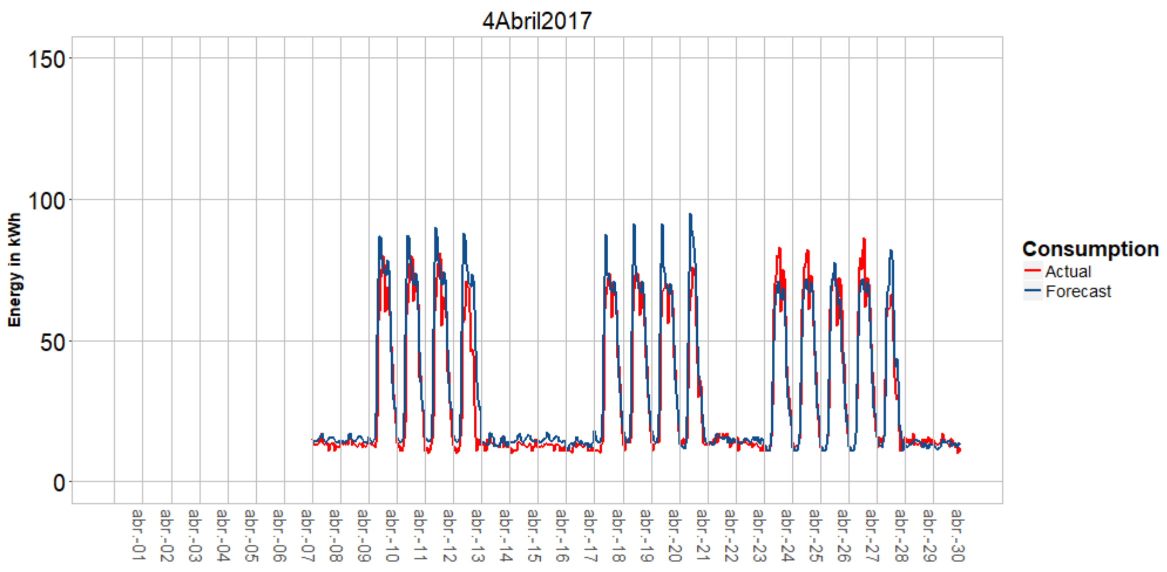


Figure 66. April 2017 Consumption after data cleanness process.

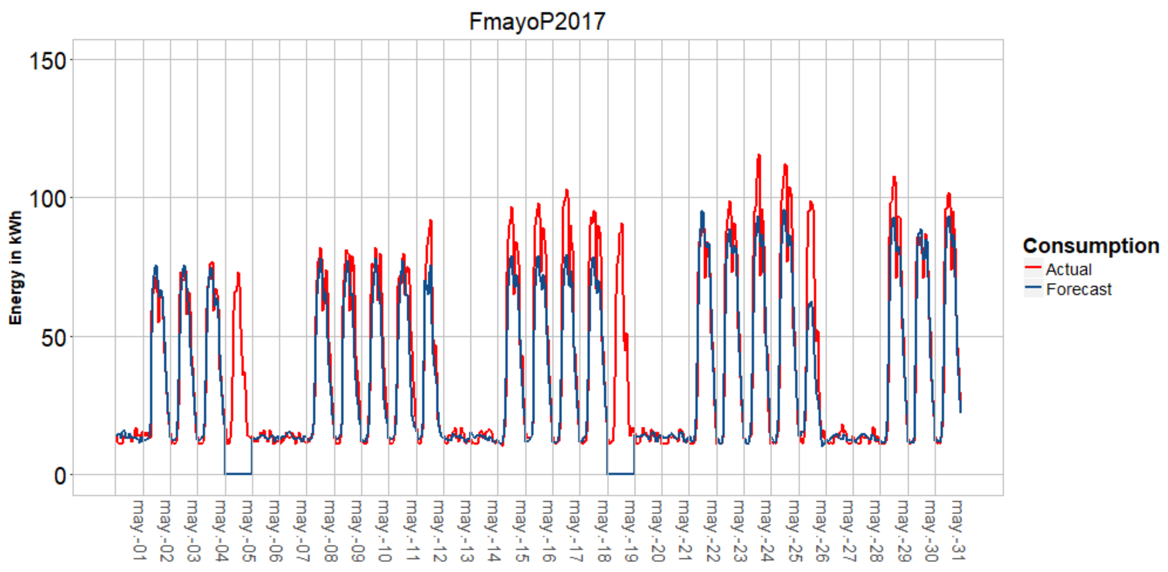


Figure 67. May 2017 Consumption before data cleanness process

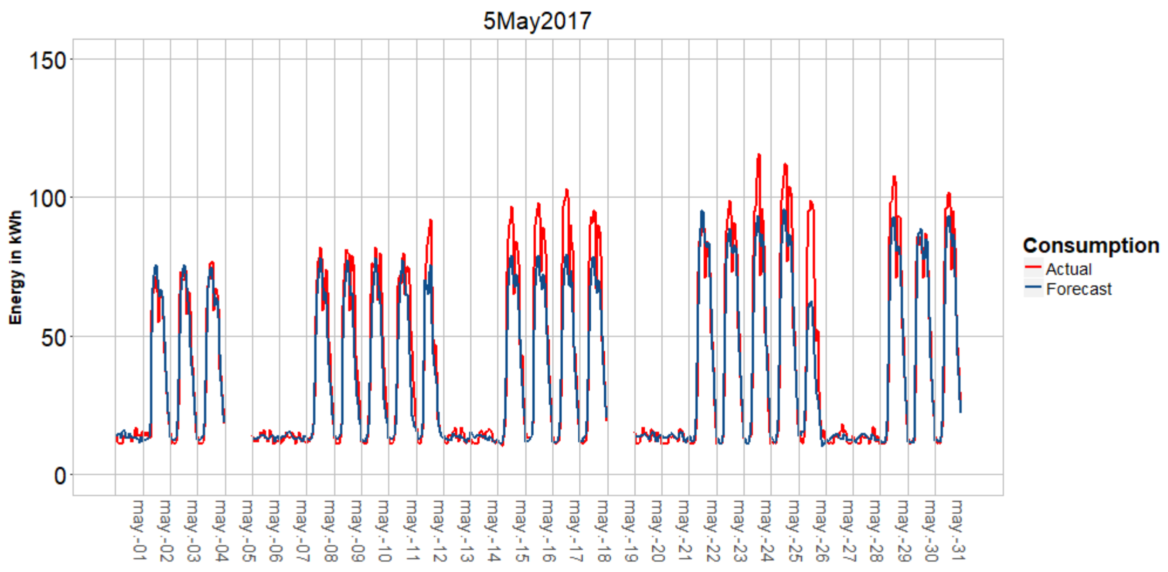


Figure 68. May 2017 Consumption after data cleanness process.

ANNEX III: Random Forest Model Graphics

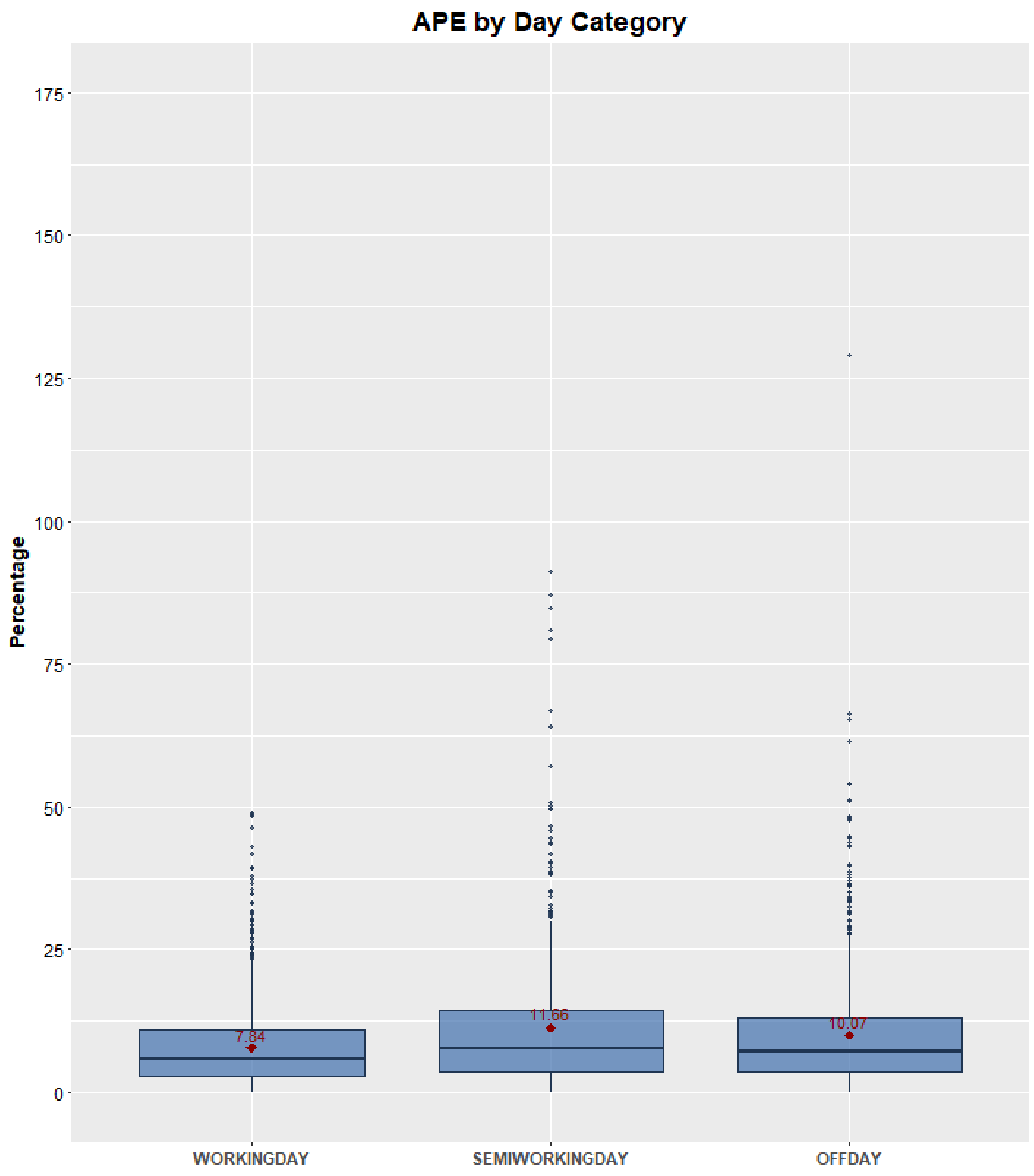


Figure 69. APE by Day Category of final RF model

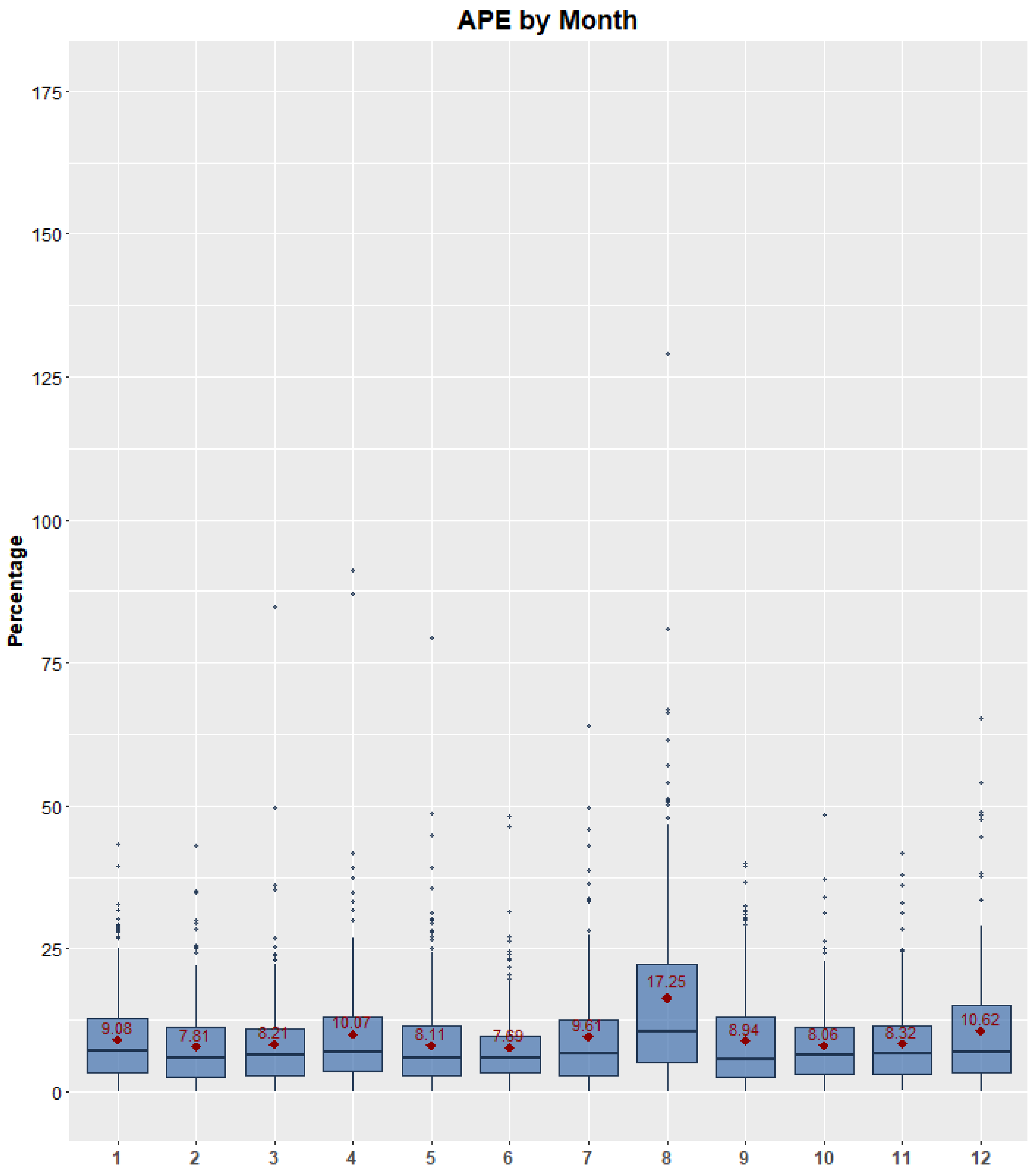


Figure 70. APE by Month of final Part II RF model

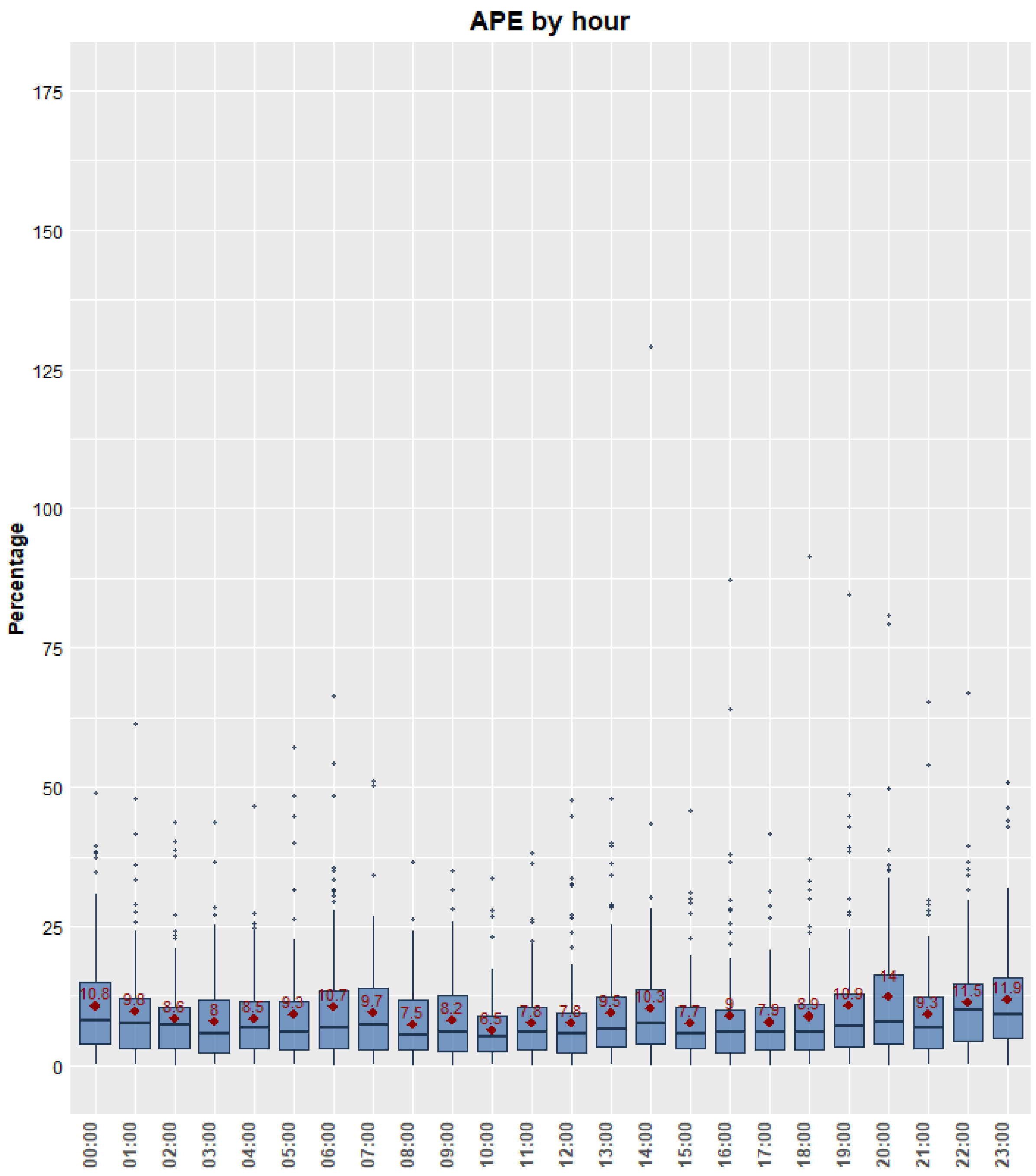


Figure 71. APE by Hour of final RF model

ANNEX IV: PART I R Code Explanation

The code written for the data analysis performed in the PART I of this dissertation is explained in this annex in detail. As it has been described in section 3.6, the programming language used has been R through the integrated development environment (IDE) RStudio.

The program developed to accomplish the analysis uses as inputs the data from the building energy consumption, obtained from the advanced metering infrastructure, and from the Energy Demand forecast, obtained as an output of the forecast model. Through DexCell Energy Manager software these datasets can be downloaded in one single one in *.CSV* format.

After running the code, the programs return two outputs. First, it returns monthly graphs of the energy consumption and energy demand forecast before and after the data cleanness process (See ANNEX II). Second, it returns the statistical metrics explained in 4.3.1, gathered in a table Furthermore, it draws and stores in the local disk all the visualizations shown in 4.3.3.

The program is structured in different scripts. The principal is mainv1.R and carries the overall process explained through the dissertation . *This script sources other scripts that carries individual task such as: tesisfunctions.R, workingcalindar.R, weathertotal.R. Specific objective of each script is explained in the following table.*

All the codes are available at the GitHUB student profile. For access or help please contact through apicatoste@gmail.com.

Table 15. R Scripts descriptions

Script	**Task**	**Lines**
Main.R	It is the main and acts as the *scheleton* of the program. From it all the other scripts are sourced.	502
Tesisfunctions.R	Includes all the functions written for the projects. These are meant for pre-process CSV files, plotting graphs, obtaining metric errors, etc.	330
Workingcalendar.R	Sets the official working calendar for years 2015 to 2017.	32
WeatherBCN.R	Deals with all the weather variables involved in the project. Not used for Part I.	114
results.R	Gather the metrics results and	167

Main script code

```r
#Opening all the libraries that will be needed

library(gridExtra)

library(lattice)

library(grid)

library(readr)

library(readxl)

library(lubridate)

library(dplyr)

library(tidyr)

library(chron)

library(ggplot2)

library(forecast)

library(plyr)

library(scales)

library(reshape2)

library(tseries)

library(zoo)

library(WriteXLS)

#Here is where all the CSVs by month should be moved before running the code

getwd()

setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")

source("tesisfunctions.R")
```

```
#Declare the list and vectors that will be used in the code

mydataframes <- list()

myforecasts <- list()

myForeGraphs <- list()

myCorrectedForeGraphs <- list()

myTempGraphs <- list()

###### SET UP THE CONSUMPTION DATASETS####

   setwd("C:/Users/Alvaro/Dropbox/TESIS/R  tesis/Datos/Consumo/CSV/")  #Here
is where all the CSVs by month should be moved before running the code

   files <- list.files(pattern = "*.csv")

  #This  is  a  bucle  for  to  read  and  pre-preprocess  with  the  function
setupdataset all the datasets in the same folder (and in CSV format). Together

  #with the pre-process I create a vector called mymonth to store all the
names of the datasets for later purposes. Also a list is created to have

  #???all the dataset in order.

    for ( i in 1:length(files) ) {

        this                                                                <-
substr(paste(gsub('.csv','',paste(files[i]))),3,nchar(paste(files[i]))-7)

        mydataframe <-  setupdataset(files[i])

        mydataframes[[this]]  = mydataframe  #Ordeno  mis  dataframes  en  una
lista y les doy sus correspondientes nombres.

        rm(mydataframe,i, this)

      }

  #In order to save all the graphs to my working directoy this for structure
is used

    for ( i in 1:length(mydataframes) ) {

        setwd("C:/Users/Alvaro/Dropbox/TESIS/R
tesis/graficas/ConsumoOriginal/")
```

```
        mymonth <- mydataframes[[i]]

        mesname <- names(mydataframes)[i]

        png(paste(mesname, ".png", sep=""), width = 1000, height = 500)

        mygraph <- graphmydataframe(i,mydataframes)

        print(mygraph)

        dev.off()

        rm(mesname,i, mymonth, mygraph)

      }

###### SET UP THE FORECAST DATASET ####

      setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/Datos/forecast/CSV/")

   filesfore <- list.files(pattern = "*.csv")

#This is a bucle for to read and pre-preprocess with the function setupmyforecast all the datasets in the same folder (and in CSV format). Together

#with the pre-process I create a vector called mymonthFORE to store all the names of the datasets for later purposes. Also a list is created to have

#???all the dataset in order.

    for ( i in 1:length(filesfore) ) {

            this                                                  <-
substr(paste(gsub('.csv','',paste(filesfore[i]))),3,nchar(paste(filesfore[i
])))

            myforecast <- setupmyforecast(filesfore[i])

            myforecasts[[this]] = myforecast #Ordeno mis dataframes en una lista y les doy sus correspondientes nombres.

            rm(myforecast,i, this)

          }
```

```
#In order to save all the graphs to my working directoy this for structure
is used

    for ( i in 1:length(myforecasts) ) {

        setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/Forecasts/")

        mymonth <- myforecasts[[i]]

        mesname <- names(myforecasts)[i]

        png(paste(mesname, ".png", sep=""), width = 1000, height = 500)

        mygraph <- graphmyforecast(i, myforecasts)

        print(mygraph)

        dev.off()

        rm(mesname,i, mymonth, mygraph)

    } # This bucle stores all the graphs in PNG in the folder FORECASTS in
the environment

    for ( i in 1:length(myforecasts) ) {

       this <- names(myforecasts)[i]

       mygraph <-  graphmyforecast(i, myforecasts)

       myForeGraphs[[this]] = mygraph

       rm(mygraph, this)

    } # This bucle creates a list storing all the graphs in variables.

###### CLEANING THE MISSING VALUES IN CONSUMPTION ######

  ### FROM CONSUMPTION DATASFRAMES

       #I attach through rbind command all the dataset to have all the anual
consumption in one single dataframe. In order for the final dataset I order
it by date.

            AnualConsumption <- NULL

            for (i in 1:length(mydataframes)){
                AnualConsumption         <-         rbind(AnualConsumption,
mydataframes[[i]])

                rm(i)

              }

            AnualConsumption                                              <-
AnualConsumption[order(AnualConsumption[,1]),]

            NAsCons.0                                                     <-
length(which(is.na(AnualConsumption$Consumption)))/length(AnualConsumption$
Consumption)*100

      #Once I have detected the missing Values I think of the best way to
fill-in this data:

        #There are 4 hours in FEBRUARY 2017 in which there are no consumptin
data due to hardware update. In order to fill in this NAs the average energy
consumption from the previous day and the

        #day after are going to be forced.

            detach("package:plyr")

            fixfebruary <- mydataframes[[2]] %>%

                filter(Date >= "2017-02-13 11:00" & Date < "2017-02-13 15:00"
| Date >= "2017-02-15 11:00" & Date < "2017-02-15 15:00") %>%

                group_by(Hour) %>%

                summarise(avg = mean(Consumption))

            indextofixFEB <- which(is.na(mydataframes[[2]]$Consumption))

            mydataframes[[2]]$Consumption[indextofixFEB]                 <-
round(fixfebruary$avg, digits = 1)

            library(plyr)

        #The   30th   of   June   from   00:00   to   23:00   is   just   missing
values.Substitute NAs in the corresponding dataframe that belongs to the
list of dataframes we created (If we modified)
```

```
            detach("package:plyr")

            fixjune <-  mydataframes[[6]] %>%

              filter((Date  >=  "2016-06-27  00:00"  &  Date  <="2016-06-29
23:00")) %>%

              group_by(Hour) %>%

              summarise(avg = mean(Consumption))

             indextofixJUN <- which(is.na(mydataframes[[6]]$Consumption))

             mydataframes[[6]]$Consumption[indextofixJUN]              <-
round(fixjune$avg, digits = 1)

            library(plyr)

        #Check the Values have been corrected and there are no more NAs in
the Anual Consumption

            AnualConsumption <- NULL

            for (i in 1:length(mydataframes)){

              AnualConsumption         <-         rbind(AnualConsumption,
mydataframes[[i]])

              rm(i)

             }

            AnualConsumption                                            <-
AnualConsumption[order(AnualConsumption[,1]),]

            NAsCons.1                                                   <-
length(which(is.na(AnualConsumption$Consumption)))/length(AnualConsumption$
Consumption)*100

    ### CLEANING THE MISSING VALUES IN CONSUMPTION FORECASTING

        #Same  process  that  the  previous  one  is  folllowed,  the  part  of
consumption  from  the  Forecast  Dataset  is  the  same  than  COMPTADOR  from
Consumption dataset

           AnualForecast <- NULL

           for (i in 1:length(myforecasts)){
            AnualForecast <- rbind(AnualForecast, myforecasts[[i]])

           }

          AnualForecast <- AnualForecast[order(AnualForecast[,1]),]

          NAsForCONS0                                                  <-
length(which(is.na(AnualForecast$Consumption)))/length(AnualForecast$Consum
ption)*100

          myforecasts[[2]]$Consumption[indextofixJUN] <- round(fixjune$avg,
digits = 1)

          indextofixFEBfore <- which(is.na(myforecasts[[10]]$Consumption))

          myforecasts[[10]]$Consumption[indextofixFEBfore]             <-
round(fixfebruary$avg, digits = 1)

          AnualForecast <- NULL

          for (i in 1:length(myforecasts)){

            AnualForecast <- rbind(AnualForecast, myforecasts[[i]])

          }

          AnualForecast <- AnualForecast[order(AnualForecast[,1]),]

          NAsForCONS1                                                  <-
length(which(is.na(AnualForecast$Consumption)))/length(AnualForecast$Consum
ption)*100

          # Borro las variables temporales para el proceso de remplazar los
NAs

          rm(fixfebruary,    fixjune,    indextofixFEB,    indextofixJUN,
indextofixFEBfore, i)

###### PRE-PROCESS MISSING VALUES, 0 VALUES, OUTLAYERS, STRANGE DEVIATION IN
FORECASTING ######

#  In  order  to  avoid  modifying  original  dataset  another  dataset  from  the
original one is created (Dataset name = Comparison).

 # we prepare the dataset for Analytics Purposes.
```

```
          Comparison <- AnualForecast

        #  Comparison$Forecast <- round(Comparison$Forecast, digits = 0)

          Comparison <- Comparison[order(Comparison[,1]),]

          row.names(Comparison) <- c(1:nrow(Comparison))

          Comparison$Day      <-      as.POSIXct(paste(day(Comparison$Date),
month(Comparison$Date),   year(Comparison$Date),   sep   =   "-"),   format   =
"%d-%m-%Y")

          Comparison$DayType <- weekdays(Comparison$Date)

          Comparison$DayType <- mapvalues(Comparison$DayType,

                                          from   =   c("lunes",   "martes",
"miércoles", "jueves", "viernes", "sábado", "domingo"),

                                          to   =   c("Monday",   "Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday" , "Sunday"))

          Comparison$Month <- month(Comparison$Date)

          Comparison$Month<- mapvalues(Comparison$Month,

                                       from  =  c("1",  "2",  "3",  "4",  "5",
"6", "7", "8", "9", "10", "11", "12"),

                                       to   =   c("1January",   "2February",
"3Marzo",  "4Abril",  "5May",  "6June"  ,  "7July",  "8August",  "9September",
"10October", "11November", "12December"))

          Comparison$Month          <-            paste(Comparison$Month,
year(Comparison$Date), sep = "")

          Comparison$Dif <- Comparison$Consumption - Comparison$Forecast

          Comparison$RelativeError                                        <-
round(Comparison$Dif/Comparison$Consumption*100, digits = 2)

          Comparison$SSE      <-      round((Comparison$Consumption       -
Comparison$Forecast)^2, digits = 2)

          Comparison$SST      <-      round((Comparison$Consumption       -
mean(Comparison$Consumption, na.rm = TRUE))^2, digits = 2)

          Comparison$SSR       <-       round((Comparison$Forecast        -
mean(Comparison$Consumption, na.rm = TRUE))^2, digits = 2)

          Comparison$Hour <- hour(Comparison$Date)

          Comparison <- Comparison[c(1,4,12,5,6,2,3,7,8,9,10,11)]

         # View(Comparison[sample(nrow(Comparison), 20), ]) Generar muestra
randomo del dataset. Just for presentation purposes.

     #We  add  the  Temperature  to  each  Hour.  So  the  code  in  Script
"weatherBCNclean" must be excuted before.

           setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")

          source("weathertotal.R")

          for (i in 1:nrow(Comparison)) {

              if(Comparison$Date[i] %in% temperatures$Superdate) {

               Comparison$MyTemp[i]                                      <-
temperatures$Temp[which(temperatures$Superdate == Comparison$Date[i])]

            } else {

              Comparison$MyTemp[i] <- NA

            }

          }

             View(Comparison)

         #Categorical Day Set

            library(plyr)

               Comparison$DayCategory <- Comparison$DayType

               Comparison$DayCategory <- mapvalues(Comparison$DayCategory,

                          from   =   c("Monday",   "Tuesday",   "Wednesday",
"Thursday", "Friday", "Saturday" , "Sunday"),

                          to  =  c("WORKINGDAY", "WORKINGDAY", "WORKINGDAY",
"WORKINGDAY", "SEMIWORKINGDAY", "OFFDAY" , "OFFDAY"))

                Comparison                                           <-
Comparison[c(1,2,3,4,14,5,6,7,13,8,9,10,11,12)]

          #Setting more specifically the holidays. Set the WORKING CALENDAR
for the company and check what is happening these days

                 setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")

                source("workingcalendar.R")

                 for (i in 1:nrow(Comparison)) {

                  if (Comparison$Day[i] %in% holidaysTOTAL) {

                    Comparison$DayCategory[i] <- "OFFDAY"

                  }

                }

              #Before Applying all the modifications to the dataset (cleaning
it) we create a copy.. This copy will be the argument of a function that
will table out all the errors in

              #different points of the cleaning process.

               ComparisonforErrors <- Comparison

             #
ComparisonforErrors$Consumption[which(is.na(ComparisonforErrors$Forecast))]
<- NA

              ComparisonRAW <- Comparison

              length(which(is.na(ComparisonRAW$Consumption)))

              length(which(is.na(ComparisonRAW$Forecast)))

  ### 1-. Looking for Missing Values in Forecast

        NAsForORIG                                                   <-
length(which(is.na(Comparison$Forecast)))/length(Comparison$Forecast)*100

  ### 2 -. Looking for HUGE Outlayers in the prediction. From the graphs we
know that biggest ones are in Nov and Dec.
```

```
        # The criteria used to erase the biggest outliers that affect the
accuracy is to assume all those forecast values in which $Dif is over 90% of
Maxium

        # consumption over the whole period analized.

        differlimit    <- 0.90*max(Comparison$Consumption)

        toEraseOutlier <- which(abs(Comparison$Dif) > differlimit)

        NAsForOutliers                                                   <-
length(toEraseOutlier)/length(Comparison$Consumption)*100

        Comparison[toEraseOutlier,]

        rm(differlimit)

        #Convert Outliers to NAs

        Comparison$Forecast[toEraseOutlier] <- NA

 ### 3-. Looking for zero value forecasts.

        toEraseZero <- which(Comparison$Forecast == 0)

        NAsForEqual0 <- length(toEraseZero)/length(Comparison$Forecast)*100

        Comparison[toEraseZero,]

        #Convert Zero Value Forecasts to NAs

        Comparison$Forecast[toEraseZero]    <- NA

         ### 4-. Looking for Values below  building's energy base load to
convert them into NAs to be ignored in the accuracy analysis.

        #First we calculate the buildings base load energy consumption by
analyzing what is happenning on saturdays and sundays.

     weekends <- Comparison %>% filter(DayType == "Saturday" | DayType ==
"DOMINGO")

        myrange <- 0.6

        toEraseUnderBL    <-    which(Comparison$Forecast    >    0     &
Comparison$Forecast < (myrange*min(weekends$Consumption)))

        NAsForUnderBS                                                       <-
length(toEraseUnderBL)/length(Comparison$Forecast)*100

        Comparison[toEraseUnderBL,]

        rm(myrange)

        #Convert Below BBS Forecasts to NAs

        Comparison$Forecast[toEraseUnderBL] <- NA

   ### 5-. Obtaning WHICH HOLIDAYS the algorithm has predicted consumption

          # Find which days, during the working hours, the forecastings has
been, on average, greater than forecastlimit. This way we can obtain the
days that were

        # holidays but a normal prediction was made.

                detach("package:plyr")

               detectHOLIDAY <- Comparison %>%

                  filter(hours(Comparison$Date)        >=        07          &
hours(Comparison$Date) < 19) #Filter by workingHours

               detectHOLIDAY <- detectHOLIDAY %>%

                  group_by(Day) %>%

                  summarise(avgcon = mean(Consumption, na.rm = TRUE),

                            avgfor = mean(Forecast, na.rm =TRUE))

               detectHOLIDAY$avgDIF      <-      detectHOLIDAY$avgfor      -
detectHOLIDAY$avgcon

               forecastlimit <- 40

               overforecastlimit     <-     which(detectHOLIDAY$avgDIF     >
forecastlimit)           #Posicion del vector que nos dirá los días con la
AVG > forecastlimit

               myfailHolidaysdates <- detectHOLIDAY[overforecastlimit,]$Day
#Días detectados como Holidays en los que se ha predecido consumo
```

```
               toEraseHoliday       <-       which(Comparison$Day       %in%
myfailHolidaysdates)             #Filas del Dataset que ocupan todos estos
días.

               NAsForHOLIDAY                                             <-
length(toEraseHoliday)/length(Comparison$Forecast)*100

               library(plyr)

               rm(detectHOLIDAY, myfailHolidaysdates, forecastlimit)

         # Convert HOLIDAYS to NAs

               Comparison$Forecast[toEraseHoliday] <- NA

    ### 6-.  Obtaning Days with Strange behaviour by VISUAL exploration

        #16 DECEMBER 2016 , #27 ENERO 2017 ???, #4 NOVEMBER 2016, #17 OCTOBER
2016



               strangeforecast    <-    as.POSIXct(c("2016-12-16","2016-11-
04","2016-11-25", "2016-10-17"))

               toEraseSTRANGE  <- which((Comparison$Day %in% strangeforecast
& !is.na(Comparison$Forecast ))) #!is.na is needed in order to obtain the
proper

#length of the elements to be erased

               NAsForSTRANGE                                             <-
length(toEraseSTRANGE)/length(Comparison$Forecast)*100

       # Convert STRANGE to NAs

               Comparison$Forecast[toEraseSTRANGE] <- NA

  #  We  have  converted  all  the  incongruence  into  NAs.  (NAs  per  se,  Big
Outlayers, 0 forecasts & below baseload forecast). We delete the consumption
data

  #from the same times that there is no forecast prediction as these data do
not provide any value to the analysis. Also the total NAs in the end is
```

```
  #calculated before to have a reference of how ofthen the predictions fails.

              NAsForTOT                                                    <-
length(which(is.na(Comparison$Forecast)))/length(Comparison$Forecast)*100

              NAsForTOTsum <- NAsForORIG + NAsForOutliers + NAsForEqual0 +
NAsForUnderBS + NAsForHOLIDAY + NAsForSTRANGE

              #Convert all the Consumption associated to a NA in Forecast
into NA and UPDATE the DIF and RelativeError, SSE,SST,SSR Columns

             Comparison$Consumption[which(is.na(Comparison$Forecast))]    <-
NA

              Comparison$Dif        <-        Comparison$Forecast          -
Comparison$Consumption

              Comparison$RelativeError                                     <-
round(Comparison$Dif/Comparison$Consumption*100, digits = 2)

              Comparison$SSE     <-     round((Comparison$Consumption      -
Comparison$Forecast)^2, digits = 2)

              Comparison$SST     <-     round((Comparison$Consumption      -
mean(Comparison$Consumption, na.rm = TRUE))^2, digits = 2)

              Comparison$SSR      <-      round((Comparison$Forecast       -
mean(Comparison$Consumption, na.rm = TRUE))^2, digits = 2)

rm(toEraseHoliday,toEraseOutlier,toEraseSTRANGE,toEraseUnderBL,toEraseZero,
holidays2016fd,  holidays2017fd,  weekends,  overforecastlimit,  NAsCons.0,
NAsCons.1, NAsForCONS0, NAsForCONS1)

##### FIRST ACCURACY ANALYSIS #####

#Once the Dataset has been cleaned and big outlayers and 0 measures have
turned into NAs a study of the accuracy is performed below. For that purpose

 #another PREPROCESS is followed to facilitate the analysis.

  # In order to be able to plot Monthly consumptions to compare month
performance. The Anual Dataframe its split into months.


        monthlyForecast <- split.data.frame(Comparison, Comparison$Month)

        length(monthlyForecast)
        monthlyForecast[[14]] <- as.data.frame(Comparison)

             #In order to save all the graphs to my working directoy this
for structure is used

        for ( i in 1:length(monthlyForecast) ) {

          setwd("C:/Users/Alvaro/Dropbox/TESIS/R
tesis/graficas/Corregidas/")

          mymonth <- monthlyForecast[[i]]

          mesname <- names(monthlyForecast)[i]

          png(paste(mesname, ".png", sep=""), width = 1000, height = 500)

          mygraph <- graphmyforecast(i,monthlyForecast)

          print(mygraph)

          dev.off()

          rm(mesname,i, mymonth, mygraph)

        }

        for ( i in 1:length(monthlyForecast) ) {

          this <- names(monthlyForecast)[i]

          mygraph <-  graphmyforecast(i, monthlyForecast)

          myCorrectedForeGraphs[[this]] = mygraph

          rm(mygraph, this)

           } # This bucle creates a list storing all the graphs in variables.

#We calculate the key performance indicators : MAE, RelativeError, R2 and
RMSE. For that purpose the function "myerrorcalculations" is written with
the dataframe with the data as an argument. Also

#function cleanMYdataset carries the process over a copy of the Comparison
Dataframe to which it applies the cleaning process step by step and stores
the error metrics in a table.
```

```
        setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")

        source("tesisfunctions.R")

        ErrorsTable <- as.data.frame(cleanMYdataset(ComparisonforErrors))

        ErrorsTable <- format(ErrorsTable, scientific=FALSE)

        ErrorsTable[1, 1:5] <- ">1e+20"

        ErrorsTable

        ComparisonforErrorsWork       <-       ComparisonforErrors        %>%
filter(DayCategory != "OFFDAY")

        ErrorsTableWorkDay                                                 <-
as.data.frame(cleanMYdataset(ComparisonforErrorsWork))

        ErrorsTableWorkDay <- format(ErrorsTableWorkDay, scientific=FALSE)

        ErrorsTableWorkDay[1, 1:5] <- ">1e+20"

        ErrorsTableWorkDay

        ComparisonforErrorsWeekend <- ComparisonforErrors %>% filter(DayType
== "Sunday" | DayType == "Saturday")

        ErrorsTableWeekEnd                                                 <-
as.data.frame(cleanMYdataset(ComparisonforErrorsWeekend))

        ErrorsTableWeekEnd <- format(ErrorsTableWeekEnd, scientific=FALSE)

        ErrorsTableWeekEnd[1, 1:5] <- ">1e+20"

        ErrorsTableWeekEnd

         ##### PLOTTING THE ERROR ######

       # ANother script for this purposes has been created

##### DEALING WITH TEMERATURES ######

            for ( i in 1:length(monthlyForecast) ) {

              setwd("C:/Users/Alvaro/Dropbox/TESIS/R
tesis/graficas/Temperature/")

              mymonth <- monthlyForecast[[i]]
```

```
      mesname <- names(monthlyForecast)[i]

      png(paste(mesname, ".png", sep=""), width = 1000, height = 500)

      mygraph <- graphmytemp(i,monthlyForecast)

      print(mygraph)

      dev.off()

      rm(mesname,i, mymonth, mygraph)

    }

    for ( i in 1:length(monthlyForecast) ) {

      this <- names(monthlyForecast)[i]

      mygraph <-  graphmytemp(i, monthlyForecast)

      myTempGraphs[[this]] = mygraph

      rm(mygraph, this)

    } # This bucle creates a list storing all the graphs in variables.

    #Plotting  temperature  togheter  with  the  consumption  and
forecasteing of each month

    for ( i in 1:length(monthlyForecast) ) {

      setwd("C:/Users/Alvaro/Dropbox/TESIS/R
tesis/graficas/TempConsumJuntos/")

      mesname <- names(monthlyForecast)[i]

      png(paste(mesname, ".png", sep=""), width = 1000, height = 1000)

      mygraph                                                    <-
grid.arrange(myTempGraphs[[i]],myCorrectedForeGraphs[[i]], ncol=1)

      print(mygraph)

      dev.off()
```

```
            rm(mesname,i, mygraph)

          }
```

Functions script code

```
#This is function to pre-process my dataset and leave them in the same format
directly from the CSV given by DeXCell.

      setupdataset <- function(n,...){

      myworkingd <- getwd()

      setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/Datos/Consumo/CSV/")

      thisfile <- paste(n)

      dataframe <- read.csv(file = thisfile, sep = ";", header = FALSE)[-
c(1:15),c(1,2,12,18)]

      names(dataframe) = c("Day", "Hour", "Consumption", "Planta4")

      dataframe$Day<-  as.POSIXct(as.character(dataframe$Day),  format  =
"%d/%m/%Y")

      dataframe$Hour <- as.character.POSIXt(dataframe$Hour) #Por defecto, la
columna Hour tiene 34 levels de factors. Lo reduzco a 24.

      dataframe$Hour <- as.factor(dataframe$Hour)

      dataframe$Consumption             <-             as.numeric(gsub(",",
"." ,dataframe$Consumption))

      dataframe$Planta4 <-as.numeric(gsub(",", "." ,dataframe$Planta4))

      row.names(dataframe) <- c(1:nrow(dataframe))

      dataframe$Date <- paste(dataframe$Day, dataframe$Hour)

      dataframe <- dataframe[,c(1,2,5,3,4)]

      dataframe$Date <- as.POSIXct(as.character(dataframe$Date), format =
"%Y-%m-%d %H:%M")

      setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")
      return(dataframe)

    }

#This is function to pre-process my dataset and leave them in the same
format.

setupmyforecast <- function(n,...){

      myfile <- paste(n)

      dataframe <- read.csv(file = myfile, sep = ",", header = FALSE)[-
c(1),c(1,2,3)]

      names(dataframe) = c("Date", "Consumption", "Forecast")

      dataframe$Date <- as.POSIXct(as.character(dataframe$Date), format =
"%Y-%m-%d %H:%M")

      dataframe$Consumption                                              <-
as.numeric(as.character(dataframe$Consumption))

      dataframe$Forecast <- as.numeric(as.character(dataframe$Forecast))

      return(dataframe)

    }

#Graphics original plots

  #In order to plot the data of interest, in this case the energy consumptio
versus the date, I create a function to set-up the design of the plot:

  #same with dataframes list. The funcion argument is a number that it is
related with the list mydataframes. To check properly the dataframe you

  #want to plot you have to first know which position of the list are u
interested in.



      graphmydataframe <- function(x, onelist, ...){

               dates            <-            c(date(onelist[[x]]$Date[1]),
date(onelist[[x]]$Date[nrow(onelist[[x]])]))
```

```
        selection <- onelist[[x]]$Date[which(hour(onelist[[x]]$Date) == 00)]

        ggplot(data = onelist[[x]], aes(x = Date)) +

          geom_line(aes(y = Consumption, colour = "General"), size = 0.8) +

          geom_line(aes(y = Planta4, colour = "Planta 4"), size = 0.8) +

          geom_vline(xintercept = as.numeric(selection), linetype=1,
colour="gray", na.rm = TRUE, size = 0.1) +

          ggtitle(names(onelist)[x]) +

          labs(x="", y= "Energy in kWh") +

          scale_colour_manual(name = "Consumption", values=c("red",
"darkolivegreen4")) +

          scale_x_datetime(labels = date_format("%b-%d"),
breaks=pretty_breaks(n = 32)(as_datetime(dates))) +

          coord_cartesian(ylim=c(0, 150))+

          theme(panel.grid.minor = element_blank(),

                panel.grid.major = element_line(colour = "gray", size =
0.1),

                panel.grid.major.x = element_blank(),

                panel.background = element_blank(),

                axis.text.y = element_text(colour="black", size = 10),

                axis.text.x = element_text(size = 9, angle = 270, vjust = -
1.2),

                axis.ticks.x=element_blank(),

                plot.title = element_text(hjust = 0.5, size = 20, vjust =
1),

                panel.border = element_rect(colour = "gray", fill= NA,
size=0.1),

                legend.background = element_rect(fill = "white", colour =
NA),
                legend.text = element_text(size = rel(0.8), color="black" ),

                legend.text.align = NULL,

                legend.title = element_text(size = rel(0.8), face = "bold",
hjust = 0, color="black"),

                legend.title.align = NULL)

      }

       graphmyforecast <- function(x, onelist, ...){

        dates              <-               c(date(onelist[[x]]$Date[1]),
date(onelist[[x]]$Date[nrow(onelist[[x]])]))

        selection <- onelist[[x]]$Date[which(hour(onelist[[x]]$Date) == 00)]

        ggplot(data = onelist[[x]], aes(x = Date)) +

          geom_line(aes(y = Consumption, colour = "Actual"), size = 0.8) +

          geom_line(aes(y = Forecast, colour = "Forecast"), size = 0.8) +

          geom_vline(xintercept   =   as.numeric(selection),   linetype=1,
colour="gray", na.rm = TRUE, size = 0.1) +

          ggtitle(names(onelist)[x]) +

          labs(x="", y= "Energy in kWh") +

          scale_colour_manual(name    =    "Consumption",    values=c("red",
"dodgerblue4"))+

          scale_x_datetime(labels          =          date_format("%b-%d"),
breaks=pretty_breaks(n = 32)(as_datetime(dates)))  +

          #coord_cartesian(ylim=c(0, 150))+

          theme(panel.grid.minor = element_blank(),

                panel.grid.major  =  element_line(colour  =  "gray",  size  =
0.1),

                panel.grid.major.x = element_blank(),
                panel.background = element_blank(),

                axis.text.y = element_text(colour="black", size = 20),

                axis.text.x = element_text(size = 15, angle = 270, vjust =
-1.2),

                axis.title = element_text(size=14,face="bold"),

                axis.ticks.x=element_blank(),

                plot.title = element_text(hjust = 0.5, size = 20, vjust =
1),

                panel.border  =  element_rect(colour  =  "gray",  fill=  NA,
size=0.1),

                legend.background = element_rect(fill = "white", colour =
NA),

                legend.text = element_text(size = 14, color="black" ),

                legend.text.align = NULL,

                legend.title = element_text(size = 18, face = "bold", hjust
= 0, color="black"),

                legend.title.align = NULL)

      }


      graphmytemp <- function(x, onelist, ...){

        dates               <-              c(date(onelist[[x]]$Date[1]),
date(onelist[[x]]$Date[nrow(onelist[[x]])]))

        selection <- onelist[[x]]$Date[which(hour(onelist[[x]]$Date) == 00)]

        ggplot(data = onelist[[x]], aes(x = Date)) +

          geom_line(aes(y = MyTemp, colour = "Temperat"), size = 0.8) +

          geom_vline(xintercept    =    as.numeric(selection),    linetype=1,
colour="gray", na.rm = TRUE, size = 0.1) +

          ggtitle(names(onelist)[x]) +
          labs(x="", y= "Temperature in (°C)") +

          scale_colour_manual(name = "Tª", values=c("darkgreen"))+

          scale_x_datetime(labels           =           date_format("%b-%d"),
breaks=pretty_breaks(n = 32)(as_datetime(dates))  +

          coord_cartesian(ylim=c(0, 40))+

          theme(panel.grid.minor = element_blank(),

                panel.grid.major  =  element_line(colour  =  "gray",  size  =
0.1),

                panel.grid.major.x = element_blank(),

                panel.background = element_blank(),

                axis.text.y = element_text(colour="black", size = 10),

                axis.text.x = element_text(size = 9, angle = 270, vjust = -
1.2),

                axis.ticks.x=element_blank(),

                plot.title = element_text(hjust = 0.5, size = 20, vjust =
1),

                panel.border  =  element_rect(colour  =  "gray",  fill=  NA,
size=0.1))


      }

      graphmyconsumption <- function(x, onelist, ...){

                dates            <-            c(date(onelist[[x]]$Date[1]),
date(onelist[[x]]$Date[nrow(onelist[[x]])]))

        selection <- onelist[[x]]$Date[which(hour(onelist[[x]]$Date) == 00)]

        ggplot(data = onelist[[x]], aes(x = Date)) +

          geom_line(aes(y = Consumption, colour = "General"), size = 0.8) +
          geom_vline(xintercept = as.numeric(selection), linetype=1,
colour="gray", na.rm = TRUE, size = 0.1) +

          ggtitle(names(onelist)[x]) +

          labs(x="", y= "Energy in kWh") +

          scale_colour_manual(name = "Consumption", values=c("red")) +

          scale_x_datetime(labels = date_format("%b-%d"),
breaks=pretty_breaks(n = 32)(as_datetime(dates))) +

          coord_cartesian(ylim=c(0, 150))+

          theme(panel.grid.minor = element_blank(),

                panel.grid.major = element_line(colour = "gray", size =
0.1),

                panel.grid.major.x = element_blank(),

                panel.background = element_blank(),

                axis.text.y = element_text(colour="black", size = 10),

                axis.text.x = element_text(size = 9, angle = 270, vjust = -
1.2),

                axis.ticks.x=element_blank(),

                plot.title = element_text(hjust = 0.5, size = 20, vjust =
1),

                panel.border = element_rect(colour = "gray", fill= NA,
size=0.1),

                legend.background = element_rect(fill = "white", colour =
NA),

                legend.text = element_text(size = rel(0.8), color="black" ),

                legend.text.align = NULL,

                legend.title = element_text(size = rel(0.8), face = "bold",
hjust = 0, color="black"),

                legend.title.align = NULL)
```

```
      }

      graphmysinglelonggraph <- function(x, onelist, ...){

        dates <- c(date(onelist[[x]]$Date[1]),
date(onelist[[x]]$Date[nrow(onelist[[x]])]))

        ggplot(data = onelist[[x]], aes(x = Date)) +

          geom_line(aes(y = Consumption, colour = "General"), size = 0.8) +

          ggtitle(names(onelist)[x]) +

          labs(x="", y= "Energy in kWh") +

          scale_x_datetime(labels = date_format("%b-%d"),
breaks=pretty_breaks(n = 32)(as_datetime(dates))) +

          coord_cartesian(ylim=c(0, 150))+

          theme(panel.grid.minor = element_blank(),

                panel.grid.major = element_line(colour = "gray", size =
0.1),

                panel.grid.major.x = element_blank(),

                panel.background = element_blank(),

                axis.text.y = element_text(colour="black", size = 10),

                axis.text.x = element_text(size = 9, angle = 270, vjust = -
1.2),

                axis.ticks.x=element_blank(),

                plot.title = element_text(hjust = 0.5, size = 20, vjust =
1),

                panel.border = element_rect(colour = "gray", fill= NA,
size=0.1))

      }

      graphmygraph <- function(dataframe, ...){
        dates                    <-                    c(date(dataframe$Date[1]),
date(dataframe$Date[nrow(dataframe)]))

        selection <- dataframe$Date[which(hour(dataframe$Date) == 00)]

        ggplot(data = dataframe, aes(x = Date)) +

          geom_line(aes(y = Consumption, colour = "Actual"), size = 0.8) +

          geom_line(aes(y = Forecast, colour = "Forecast"), size = 0.8) +

          geom_vline(xintercept    =    as.numeric(selection),    linetype=1,
colour="gray", na.rm = TRUE, size = 0.1) +

          ggtitle("Data") +

          labs(x="", y= "Energy in kWh") +

          scale_colour_manual(name    =    "Consumption",    values=c("red",
"dodgerblue4"))+

          scale_x_datetime(labels          =          date_format("%b-%d"),
breaks=pretty_breaks(n = 32)(as_datetime(dates)))  +

          coord_cartesian(ylim=c(0, 150))+

          theme(panel.grid.minor = element_blank(),

                panel.grid.major  =  element_line(colour  =  "gray",  size  =
0.1),

                panel.grid.major.x = element_blank(),

                panel.background = element_blank(),

                axis.text.y = element_text(colour="black", size = 10),

                axis.text.x = element_text(size = 9, angle = 270, vjust = -
1.2),

                axis.ticks.x=element_blank(),

                plot.title = element_text(hjust = 0.5, size = 20, vjust =
1),

                panel.border  =  element_rect(colour  =  "gray",  fill=  NA,
size=0.1),
                legend.background = element_rect(fill = "white", colour =
NA),

                legend.text = element_text(size = rel(0.8), color="black" ),

                legend.text.align = NULL,

                legend.title = element_text(size = rel(0.8), face = "bold",
hjust = 0, color="black"),

                legend.title.align = NULL)

      }

#Functions to obtain error metrics

      #We calculate the key performance indicators : MAE, RelativeError, R2
and RMSE. For that purpose the function "myerrorcalculations" is written
with the dataframe with the data as an argument. Also

      #function cleanMYdataset carries the process over a copy of the
Comparison Dataframe to which it applies the cleaning process step by step
and stores the error metrics in a table.

      myerrorcalculationsBIS <- function(dataframe,...){

        #Update the dataframe to obtain the error metrics

        dataframe$Consumption[which(is.na(dataframe$Forecast))]    <-    NA
#Both Vectors need to have the same NAs in order for the R squared to be
calculated

        dataframe$Dif <- dataframe$Consumption - dataframe$Forecast

        dataframe$BPE   <-   round(dataframe$Dif/dataframe$Consumption*100,
digits = 2)

        dataframe$APE                                                    <-
round(abs(dataframe$Dif)/abs(dataframe$Consumption), digits = 2)

        dataframe$SSE     <-      round((     dataframe$Consumption      -
dataframe$Forecast)^2, digits = 2)

        dataframe$SSR      <-       round((      dataframe$Forecast       -
mean( dataframe$Forecast, na.rm = TRUE))^2, digits = 2)
```

```
        dataframe$SST <- dataframe$SSE + dataframe$SSR



        #Mean absolute error (MAPE). Is the same as Mean() o Aritmetic
average.

     errorMAE <- round( sum(abs(dataframe$Dif), na.rm =
TRUE)/(length(which(!is.na(dataframe$Dif)))) , digits = 2)

        #Mean absolute error (MAPE). Is the same as Mean() o Aritmetic
average.

        errorMAPE <-
round( sum(abs(dataframe$Dif)/dataframe$Consumption*100, na.rm =
TRUE)/(length(which(!is.na(dataframe$Dif)))) , digits = 2)

        #Mean Biased Error

        errorMBPE <- round ( mean(dataframe$BPE, na.rm = TRUE) , digits = 2)

        #Root Mean Square (RMSE)

        RMSe <- round( sqrt(sum((dataframe$Dif)^2, na.rm =
TRUE)/length(which(!is.na(dataframe$Dif)))) , digits = 2)

        #R2

        rsquaredGEN <- round( (1 - sum(dataframe$SSE, na.rm =
TRUE)/sum(dataframe$SST, na.rm = TRUE)) , digits = 2)

        # Gathering together all the errors into a single error vector



        NAsForTOT <-
length(which(is.na(dataframe$Forecast)))/length(dataframe$Forecast)*100

        errorsvectorBIS <- round( c(errorMAE,errorMAPE, errorMBPE, RMSe,
rsquaredGEN, NAsForTOT) , digits = 2)

        names(errorsvectorBIS) <- c("MAE (kWh)", "MAPE (%)", "MBPE (%)",
"RMSE (kWh)", "R2", "NAs(%)" )

        return(errorsvectorBIS)

      }
      cleanMYdatasetBIS <- function(dataframe,...) {

        #------------- RAW

        metricasRAW <- myerrorcalculationsBIS(dataframe) # Obtain errors in
RAW dataset

        #------------- OUTLIERS

        differlimit    <- 0.9*max(dataframe$Consumption)

        toEraseOutlier <- which(abs(dataframe$Dif) > differlimit)

        print(toEraseOutlier)

        dataframe$Forecast[toEraseOutlier] <- NA

        metricasOUTLIER <- myerrorcalculationsBIS(dataframe) # Obtain errors
after first modification (1-. Set Outliers to NAs)

        metricasOUTLIER

        #------------- ZERO VALUES

        toEraseZero <- which(dataframe$Forecast == 0)

        dataframe$Forecast[toEraseZero]    <- NA

        metricasZERO <- myerrorcalculationsBIS(dataframe) #  Obtain errors
after second modification (2-. Set Zero Values to NAs)

        #------------- BELOW BASE LOAD

        weekends <- dataframe %>% filter(DayType == "Saturday" | DayType ==
"DOMINGO")

        myrange <- 0.6

        toEraseUnderBL <- which(dataframe$Forecast > 0 & dataframe$Forecast
< (myrange*min(dataframe$Consumption)))

        dataframe$Forecast[toEraseUnderBL] <- NA

        metricasBASELOAD  <-  myerrorcalculationsBIS(dataframe)  #  Obtain
errors after third modification (3-. Set UnderBase Values to NAs)
```

```
        #------------- HOLIDAYS

        detach("package:plyr")

        detectHOLIDAY <- dataframe %>%

          filter(hours(dataframe$Date) >= 07 & hours(dataframe$Date) < 19)
#Filter by workingHours

        detectHOLIDAY <- detectHOLIDAY %>%

          group_by(Day) %>%

          summarise(avgcon = mean(Consumption, na.rm = TRUE),

                    avgfor = mean(Forecast, na.rm =TRUE))

        detectHOLIDAY$avgDIF <- detectHOLIDAY$avgfor - detectHOLIDAY$avgcon

        forecastlimit <- 40

        overforecastlimit  <-  which(detectHOLIDAY$avgDIF  >  forecastlimit)
#Posicion del vector que nos dirá los días con la AVG > forecastlimit

        myfailHolidaysdates     <-     detectHOLIDAY[overforecastlimit,]$Day
#Días detectados como Holidays en los que se ha predecido consumo

        toEraseHoliday  <-  which(dataframe$Day  %in%  myfailHolidaysdates)
#Filas del Dataset que ocupan todos estos días.

        library(plyr)

        dataframe$Forecast[toEraseHoliday] <- NA

        metricasHOLIDAYS  <-  myerrorcalculationsBIS(dataframe)  #  Obtain
errors after third modification (4-. Set HOLIDAYS to NAs)

        #------------- STRANGE

        strangeforecast <- as.POSIXct(c("2016-12-16","2016-11-04","2016-11-
25", "2016-10-17"))

        toEraseSTRANGE     <-   which((dataframe$Day   %in%   strangeforecast
& !is.na(dataframe$Forecast ))) #!is.na is needed in order to obtain the
proper

        #length of the elements to be erased
        dataframe$Forecast[toEraseSTRANGE] <- NA

        metricasSTRANGE <- myerrorcalculationsBIS(dataframe) # Obtain errors
after third modification (5-. Set STRANGE to NAs)

        #------------- ALLTOGTHER

        metricasRAW <- as.data.frame(metricasRAW)

        metricasOUTLIER <- as.data.frame(metricasOUTLIER)

        metricasZERO <- as.data.frame(metricasZERO)

        metricasBASELOAD <- as.data.frame(metricasBASELOAD)

        metricasHOLIDAYS <- as.data.frame(metricasHOLIDAYS)

        metricasSTRANGE <- as.data.frame(metricasSTRANGE)

        metricasFINAL <- metricasSTRANGE

        metricasERRORbis               <-               cbind(metricasRAW,
metricasOUTLIER,metricasZERO,      metricasBASELOAD,      metricasHOLIDAYS,
metricasFINAL)

        names(metricasERRORbis) <- c("Raw Data", "wo/Outliers", "wo/ZVs",
"wo/BelowBBL", "wo/HOLIDAYS", "wo/StrangeB")

        metricasERRORbis <- round(t(metricasERRORbis), digits = 2)

        return(metricasERRORbis)

              }
```

workingcalendar code

```
holidays2015fd <- c("2015-01-01", "2015-01-06",
                    "2015-04-03", "2015-04-06",
                    "2015-05-01",
                    "2015-06-01", "2015-06-24",
                    "2015-08-15",
                    "2015-09-11", "2015-09-24",
                    "2015-10-12",
                          "2015-12-07","2015-12-08","2015-12-25", "2015-12-28",
"2015-12-29", "2015-12-30", "2015-12-31")
      holidays2015fd <- as.POSIXct(holidays2015fd)

      priorholidays2015fd <- c("2015-01-05",
                                    "2015-04-02",
                                    "2015-12-24")
      priorholidays2015fd <- as.POSIXct(priorholidays2015fd)

      #7th of december they did PUENTE as the last week of DECEMBER
      #30th of April not included because once checked the consumption looks like
normal working day
      #31 may 2015 sunday
      #23rd june 2015, wednesday but they worked full time
      #14th august was already friday, semiworkindday
      #10th, 23rd september they worked full time
      #11th, october sunday

      holidays2016fd <- c("2016-01-01",
                          "2016-01-06",
                          "2016-02-12",
                          "2016-03-25",
                          "2016-03-28",
                          "2016-05-16",
                          "2016-06-24",
                          "2016-08-15",
                          "2016-10-12", "2016-10-31", #15 Octobe is Saturday
                          "2016-11-01",
                          "2016-12-06", "2016-12-08", "2016-12-09","2016-12-
26","2016-12-27", "2016-12-28", "2016-12-29", "2016-12-30")
      holidays2016fd <- as.POSIXct(holidays2016fd)

      priorholidays2016fd <- c("2016-01-05")
      priorholidays2016fd <- as.POSIXct(priorholidays2016fd)


      #1st January , 6th January, -> From semiwroking day to OFFDAY
      #12th February -> From semiwroking day to OFFDAY
      #25th,March -> From semiwroking day to OFFDAY
      #28th March -> From workingday to OFFDAY

      holidays2017fd <- c("2017-01-06",
                          "2017-04-14", "2017-04-17",
                              "2017-05-01",
                              "2017-06-05",
                              "2017-08-15",
                              "2017-09-11", "2017-09-25",
                              "2017-10-12",
                              "2017-11-01",
                              "2017-12-06", "2017-12-07", "2017-12-08", "2017-12-
25","2017-12-26", "2017-12-27", "2017-12-28", "2017-12-29")
      holidays2017fd <- as.POSIXct(holidays2017fd)

      priorholidays2017fd <- c("2017-01-05",
                                 "2017-04-13")
      priorholidays2017fd <- as.POSIXct(priorholidays2017fd)


      #1st January , 6th January, -> From semiwroking day to OFFDAY
      #12th February -> From semiwroking day to OFFDAY
      #25th,March -> From semiwroking day to OFFDAY
      #28th March -> From workingday to OFFDAY

      holidaysTOTAL <- c(holidays2015fd,holidays2016fd, holidays2017fd)
      priorholidaysTOTAL <- c(priorholidays2015fd,priorholidays2016fd,
priorholidays2017fd)
      View(totalconsumption)
       semiholidays2015fd <- c("2015-07-20",
                                "2015-07-21",
                                "2015-07-22",
                                "2015-07-23",
                                "2015-07-27",
                                "2015-07-28",
                                "2015-07-29",
                                "2015-07-30",
                                "2015-08-03",
                                "2015-08-04",
                                "2015-08-05",
                                "2015-08-06",
                                "2015-08-10",
                                "2015-08-11",
                                "2015-08-12",
                                "2015-08-13",
                                "2015-08-17",
                                "2015-08-18",
                                "2015-08-19",
                                "2015-08-20",
                                "2015-08-25",
                                "2015-08-26",
                                "2015-08-27",
                                "2015-08-31")
        semiholidays2015fd <- as.POSIXct(semiholidays2015fd)

        semiholidays2016fd <- c("2016-07-18",
                                "2016-07-19",
                                "2016-07-20",
                                "2016-07-21",
                                "2016-07-25",
                                "2016-07-26",
                                "2016-07-27",
                                "2016-07-28",
                                "2016-08-01",
                                "2016-08-02",
                                "2016-08-03",
                                "2016-08-04",
                                "2016-08-08",
                                "2016-08-09",
                                "2016-08-10",
                                "2016-08-11",
                                "2016-08-16",
                                "2016-08-17",
                                "2016-08-18",
                                "2016-08-22",
                                "2016-08-23",
                                "2016-08-24",
                                "2016-08-25",
                                "2016-08-29",
                                "2016-08-30",
                                "2016-08-31")
        semiholidays2016fd <- as.POSIXct(semiholidays2016fd)
        semiholidaysTOTAL <- c(semiholidays2015fd,semiholidays2016fd)
  #PERIODO ESTIVAL (THE WORKING HOURS AND THE DURATION ARE DIFFERENT)

        #We ignore this days and treat them as fully working days. Maybe we should
  study that the hourly schedule will be dfferent, expecting the
        #consumption to increase earlier these days.
```

Results script code

```
    setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")

    source("tesisfunctions.R")

    ErrorsTableBIS <- as.data.frame(cleanMYdatasetBIS(ComparisonforErrors))

    ErrorsTableBIS <- format(ErrorsTableBIS, scientific=FALSE)

    ErrorsTableBIS[1, 1:5] <- ">1e+20"

    ErrorsTableBIS

    setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/Report/")

    write.csv(ErrorsTableBIS, file = "TableResults.csv")

    ComparisonGraph <- Comparison

    str(ComparisonGraph)

    ComparisonGraph$DayCategory <- as.factor(ComparisonGraph$DayCategory)

    print(levels(ComparisonGraph$DayCategory))

    ComparisonGraph$DayCategory                                           <-
factor(ComparisonGraph$DayCategory,levels(ComparisonGraph$DayCategory)[c(3,
2,1)])

    ComparisonGraph$DayType <- as.factor(ComparisonGraph$DayType)

    print(levels(ComparisonGraph$DayType))

    ComparisonGraph$DayType                                               <-
factor(ComparisonGraph$DayType,levels(ComparisonGraph$DayType)[c(2,6,7,5,1,
3,4)])

    str(ComparisonGraph$DayCategory)

    ComparisonGraph$MonthNum <- month(Comparison$Date)

    ComparisonGraph$MonthNum <- as.factor(ComparisonGraph$MonthNum)

    print(levels(ComparisonGraph$MonthNum))
    ComparisonGraph$MonthNum <- factor(ComparisonGraph$MonthNum,
                                   labels = c("Jan", "Feb", "Mar", "Apr", "May",
"Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"))

    ComparisonGraph$Seasson <- 0

    for(i in 1:nrow(ComparisonGraph)){

      if( ComparisonGraph$Date[i] <= "2016-06-20 23:00"){

        ComparisonGraph$Seasson[i] <- 1

      }  else  if(  ComparisonGraph$Date[i]  >  "2016-06-20  23:00"  &
ComparisonGraph$Date[i] <= "2016-09-22 00:00"){

        ComparisonGraph$Seasson[i] <- 2

      }  else  if(  ComparisonGraph$Date[i]  >  "2016-09-22  00:00"  &
ComparisonGraph$Date[i] <= "2016-12-21 00:00"){

        ComparisonGraph$Seasson[i] <- 3

      }  else  if(  ComparisonGraph$Date[i]  >  "2016-12-21  00:00"  &
ComparisonGraph$Date[i] <= "2017-03-20 15:00"){

        ComparisonGraph$Seasson[i] <- 4

      }  else  if(  ComparisonGraph$Date[i]  >  "2017-03-20  15:00"  &
ComparisonGraph$Date[i] <= "2017-06-21 15:00"){

          ComparisonGraph$Seasson[i] <- 1

    }

    }

    ComparisonGraph$Seasson <- as.factor(ComparisonGraph$Seasson)

    print(levels(ComparisonGraph$Seasson))

    ComparisonGraph$Seasson <- factor(ComparisonGraph$Seasson,
                                          labels   =   c("Spring",   "Summer",
"Autumn", "Winter"))

    ComparisonGraph$Hour <- as.factor(ComparisonGraph$Hour)
    print(levels(ComparisonGraph$Hour))

    names(ComparisonGraph)

    ComparisonGraph <- ComparisonGraph[,c(1,3,4,5,15,16,7,8)]

    ComparisonGraph$Dif        <-        ComparisonGraph$Consumption        -
ComparisonGraph$Forecast

    ComparisonGraph$BPE                                                    <-
round(ComparisonGraph$Dif/ComparisonGraph$Consumption*100, digits = 2)

    ComparisonGraph$APE                                                    <-
round(abs(ComparisonGraph$Dif)/abs(ComparisonGraph$Consumption)*100, digits
= 2)

    str(ComparisonGraph)

    View(ComparisonGraph)

    errorMAPEtry                                                           <-
round( sum(abs(ComparisonGraph$Dif)/ComparisonGraph$Consumption*100,  na.rm
= TRUE)/(length(which(!is.na(ComparisonGraph$Dif)))) , digits = 2)

    errorMAPEtry

    errorMAPEtry2 <- round( mean(ComparisonGraph$APE, na.rm=TRUE) , digits
= 2)

    errorMAPEtry2

    # complete.cases(Comparison)

    # buildgraphs <- ComparisonGraph[-which(is.na(Compariso))]

#BUILDING GRPAHICS

    fill <- "#4271AE"

    line <- "#1F3552"

    # meanAPEhour     <- aggregate(APE ~  hour, ComparisonGraph, mean)

    meanAPEweekday   <- aggregate(APE ~  DayType, ComparisonGraph, mean)
    meanAPEcategory  <- aggregate(APE ~  DayCategory, ComparisonGraph, mean)

    meanAPEseasson   <- aggregate(APE ~  Seasson, ComparisonGraph, mean)

    meanAPEhour      <- aggregate(APE ~  Hour, ComparisonGraph, mean)

       # meanBPEhour     <- aggregate(BPE ~  DayType, ComparisonGraph, mean)

    meanBPEweekday   <- aggregate(BPE ~  DayType, ComparisonGraph, mean)

    meanBPEcategory  <- aggregate(BPE ~  DayCategory, ComparisonGraph, mean)

    meanBPEseasson   <- aggregate(BPE ~  Seasson, ComparisonGraph, mean)

    meanBPEhour      <- aggregate(BPE ~  Hour, ComparisonGraph, mean)

  # BY WEEKDAYS

        APEweekday <- ggplot(ComparisonGraph, aes(x = DayType, y = APE)) +

        geom_boxplot(fill = fill, colour = line, alpha = 0.7,

                     outlier.colour = "#1F3552", outlier.shape = 20) +

        scale_y_continuous(name = "Percentage",

                           breaks = seq(0, 175, 25),

                           limits=c(0, 175)) +

        #scale_x_discrete(name = "Day of the Week") +

        ggtitle("APE by day of the week") +

        stat_summary(fun.y=mean, colour="darkred", geom="point",

                     shape=18, size=3,show_guide = FALSE) +

        scale_colour_manual(name = "Consumption") +

        geom_text(data = meanAPEweekday, aes(label = round( APE, digits =
2)), color = "darkred", vjust = -0.5)+

        theme(axis.text.y = element_text(colour="black", size = 12),

              axis.text.x = element_text(size = 12, face = "bold"),

              axis.title.y = element_text(size=14,face="bold"),
              axis.title.x=element_blank(),

              plot.title = element_text(hjust = 0.5, size = 18, face="bold"))

     APEweekday

     setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/BOXPLOT/")

     png("APEweekday.png", width = 709, height = 800)

     APEweekday

     print(APEweekday)

     dev.off()

     BPEweekday <- ggplot(ComparisonGraph, aes(x = DayType, y = BPE)) +

        geom_boxplot(fill = fill, colour = line, alpha = 0.7,

                     outlier.colour = "#1F3552", outlier.shape = 20) +

        scale_y_continuous(name = "Percentage",

                           breaks = seq(-200, 150, 25),

                           limits=c(-200, 150)) +

        #scale_x_discrete(name = "Day of the Week") +

        ggtitle("BPE by day of the week") +

        stat_summary(fun.y=mean, colour="darkred", geom="point",

                     shape=18, size=3,show_guide = FALSE) +

        geom_text(data = meanBPEweekday, aes(label = round( BPE, digits =
2)), color = "darkred", vjust = 1.2)+

        theme(axis.text.y = element_text(colour="black", size = 12),

              axis.text.x = element_text(size = 12, face = "bold"),

              axis.title.y = element_text(size=14,face="bold"),

              axis.title.x=element_blank(),
              plot.title = element_text(hjust = 0.5, size = 18, face="bold"))

     BPEweekday

     setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/BOXPLOT/")

     png("BPEweekday.png", width = 709, height = 800)

     BPEweekday

     print(BPEweekday)

     dev.off()

  # BY DAY CATEGORY

          APEcategory <- ggplot(ComparisonGraph, aes(x = DayCategory, y =
APE)) +

       geom_boxplot(fill = fill, colour = line, alpha = 0.7,

                    outlier.colour = "#1F3552", outlier.shape = 20) +

       scale_y_continuous(name = "Percentage",

                          breaks = seq(0, 175, 25),

                          limits=c(0, 175)) +

       #scale_x_discrete(name = "Day of the Week") +

       ggtitle("APE by Day Category") +

       stat_summary(fun.y=mean, colour="darkred", geom="point",

                    shape=18, size=3,show_guide = FALSE) +

       scale_colour_manual(name = "Consumption") +

       geom_text(data = meanAPEcategory, aes(label = round( APE, digits =
2)), color = "darkred", vjust = -0.5)+

       theme(axis.text.y = element_text(colour="black", size = 12),

             axis.text.x = element_text(size = 12, face = "bold"),

             axis.title.y = element_text(size=14,face="bold"),

             axis.title.x=element_blank(),
             plot.title = element_text(hjust = 0.5, size = 18, face="bold"))

     APEcategory

     setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/BOXPLOT/")

     png("APEcategory.png", width = 709, height = 800)

     APEcategory

     print(APEcategory)

     dev.off()

     BPEcategory <- ggplot(ComparisonGraph, aes(x = DayCategory, y = BPE))
+

       geom_boxplot(fill = fill, colour = line, alpha = 0.7,

                    outlier.colour = "#1F3552", outlier.shape = 20) +

       scale_y_continuous(name = "Percentage",

                          breaks = seq(-200, 150, 25),

                          limits=c(-200, 150)) +

       #scale_x_discrete(name = "Day of the Week") +

       ggtitle("BPE by Day Category") +

       stat_summary(fun.y=mean, colour="darkred", geom="point",

                    shape=18, size=3,show_guide = FALSE) +

       geom_text(data = meanBPEcategory, aes(label = round( BPE, digits =
2)), color = "darkred", vjust = 1.2)+

       theme(axis.text.y = element_text(colour="black", size = 12),

             axis.text.x = element_text(size = 12, face = "bold"),

             axis.title.y = element_text(size=14,face="bold"),

             axis.title.x=element_blank(),
             plot.title = element_text(hjust = 0.5, size = 18, face="bold"))

     BPEcategory

  setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/BOXPLOT/")

     png("BPEcategory.png", width = 709, height = 800)

     BPEcategory

     print(BPEcategory)

     dev.off()

# BY SEASSON

     APEseasson <- ggplot(ComparisonGraph, aes(x = Seasson, y = APE)) +

       geom_boxplot(fill = fill, colour = line, alpha = 0.7,

                    outlier.colour = "#1F3552", outlier.shape = 20) +

       scale_y_continuous(name = "Percentage",

                          breaks = seq(0, 175, 25),

                          limits=c(0, 175)) +

       #scale_x_discrete(name = "Day of the Week") +

       ggtitle("APE by Season") +

       stat_summary(fun.y=mean, colour="darkred", geom="point",

                    shape=18, size=3,show_guide = FALSE) +

       scale_colour_manual(name = "Consumption") +

       geom_text(data = meanAPEseasson, aes(label = round( APE, digits =
2)), color = "darkred", vjust = -0.5)+

       theme(axis.text.y = element_text(colour="black", size = 12),

             axis.text.x = element_text(size = 12, face = "bold"),

             axis.title.y = element_text(size=14,face="bold"),

             axis.title.x=element_blank(),
             plot.title = element_text(hjust = 0.5, size = 18, face="bold"))

     APEseasson

     setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/BOXPLOT/")

     png("APEseasson.png", width = 709, height = 800)

     APEseasson

     print(APEseasson)

     dev.off()

     BPEseasson <- ggplot(ComparisonGraph, aes(x = Seasson, y = BPE)) +

       geom_boxplot(fill = fill, colour = line, alpha = 0.7,

                    outlier.colour = "#1F3552", outlier.shape = 20) +

       scale_y_continuous(name = "Percentage",

                          breaks = seq(-200, 150, 25),

                          limits=c(-200, 150)) +

       #scale_x_discrete(name = "Day of the Week") +

       ggtitle("BPE by Season") +

       stat_summary(fun.y=mean, colour="darkred", geom="point",

                    shape=18, size=3,show_guide = FALSE) +

       geom_text(data = meanBPEseasson, aes(label = round( BPE, digits =
2)), color = "darkred", vjust = 1.2)+

       theme(axis.text.y = element_text(colour="black", size = 12),

             axis.text.x = element_text(size = 12, face = "bold"),

             axis.title.y = element_text(size=14,face="bold"),

             axis.title.x=element_blank(),

             plot.title = element_text(hjust = 0.5, size = 18, face="bold"))
```

```
     BPEseasson

     setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/BOXPLOT/")

     png("BPEseasson.png", width = 709, height = 800)

     BPEseasson

     print(BPEseasson)

     dev.off()

# BY HOUR

     APEhour <- ggplot(ComparisonGraph, aes(x = Hour, y = APE)) +

       geom_boxplot(fill = fill, colour = line, alpha = 0.7,

                    outlier.colour = "#1F3552", outlier.shape = 20) +

       scale_y_continuous(name = "Percentage",

                          breaks = seq(0, 175, 25),

                          limits=c(0, 175)) +

       #scale_x_discrete(name = "Day of the Week") +

       ggtitle("APE by hour") +

       stat_summary(fun.y=mean, colour="darkred", geom="point",

                    shape=18, size=3,show_guide = FALSE) +

       scale_colour_manual(name = "Consumption") +

       geom_text(data = meanAPEhour, aes(label = round( APE, digits = 1)),
color = "darkred", vjust = -0.5)+

       theme(axis.text.y = element_text(colour="black", size = 12),

             axis.text.x = element_text(size = 12, face = "bold"),

             axis.title.y = element_text(size=14,face="bold"),

             axis.title.x=element_blank(),

             plot.title = element_text(hjust = 0.5, size = 18, face="bold"))
```

```
    APEhour

    setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/BOXPLOT/")

    png("APEhour.png", width = 709, height = 800)

    APEhour

    print(APEhour)

    dev.off()

   BPEhour <- ggplot(ComparisonGraph, aes(x = Hour, y = BPE)) +

      geom_boxplot(fill = fill, colour = line, alpha = 0.7,

                   outlier.colour = "#1F3552", outlier.shape = 20) +

      scale_y_continuous(name = "Percentage",

                         breaks = seq(-200, 150, 25),

                         limits=c(-200, 150)) +

      #scale_x_discrete(name = "Day of the Week") +

      ggtitle("BPE by hour") +

      stat_summary(fun.y=mean, colour="darkred", geom="point",

                   shape=18, size=3,show_guide = FALSE) +

      geom_text(data = meanBPEhour, aes(label = round( BPE, digits = 1)),
color = "darkred", vjust = 1.2)+

      theme(axis.text.y = element_text(colour="black", size = 12),

            axis.text.x = element_text(size = 12, face = "bold"),

            axis.title.y = element_text(size=14,face="bold"),

            axis.title.x=element_blank(),

            plot.title = element_text(hjust = 0.5, size = 18, face="bold"))

    BPEhour
```

```
setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/BOXPLOT/")
 png("BPEhour.png", width = 709, height = 800)
 BPEhour
 print(BPEhour)
 dev.off()
distributionBPE <- ggplot(ComparisonGraph, aes(BPE)) +
  geom_histogram(binwidth = 1, color = line, fill = fill)+
  ggtitle("BPE Distribution") +
  scale_x_continuous(name = "Percentage") +
  scale_y_continuous(name = "Number of instances")+
  theme(axis.text.y = element_text(colour="black", size = 12),
        axis.text.x = element_text(size = 12, face = "bold"),
        axis.title.y = element_text(size=14,face="bold"),
        axis.title.x= element_text(size=14),
        plot.title = element_text(hjust = 0.5, size = 18, face="bold"))
distributionBPE
setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/BOXPLOT/")
png("BPEdistribution.png", width = 709, height = 432)
distributionBPE
print(distributionBPE)
dev.off()
  distriberror
  distributionMAPE <- ggplot(ComparisonGraph, aes(x = Date)) +
  geom_point(aes(y = APE), color = line, fill = fill) +
  scale_y_continuous(name = "Percentage",
                            breaks = seq(0, 300, 25),

                            limits=c(0, 300)) +

  #scale_x_discrete(name = "Day of the Week") +

  ggtitle("APE Evolution") +

  theme(axis.text.y = element_text(colour="black", size = 12),

        axis.text.x = element_text(size = 12, face = "bold"),

        axis.title.y = element_text(size=14,face="bold"),

        axis.title.x=element_blank(),

        plot.title = element_text(hjust = 0.5, size = 18, face="bold"))

distributionMAPE

    setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/BOXPLOT/")

png("MAPEevol.png", width = 709, height = 432)

distributionMAPE

print(distributionMAPE)

dev.off()
```

ANNEX V: PART II R Code Explanation

Script	Task	Lines
Main.R	It is the main and acts as the *scheleton* of the program. From it all the other scripts are sourced. Same as Part I.	502
totalconsumption.R	Set the consumption data ready to be passed to the ML models. Variables in original format (Process done in each particular ML script). The final features are: Date / Day / Month / MonthGen / DayType / DayCategory / Temperature / Consumption / DatePriorWeek/WeekBefore / DatePriorDate / DayBefore	190
Workingcalendar.R	Sets the official working calendar for years 2015 to 2017 for the COMPANY operating in the building. Same as Part I.	32
WeatherBCN.R	Deals with all the weather variables involved in the project. Not used for Part I. Same as Part I.	114
ann.R	Artificial Neural Network Model and Test	155
Knn.R	K Nearest Neighbor Model and Test	128
Rf.R	Random Forest model and test.	323

All the codes are available at the GitHUB student profile. For access or help please contact through apicatoste@gmail.com.

Date Pre-Process (totalconsumption.R)

```
# The only output of this script is a dataframe called "totalconsumption" that will be
used in other parts of the programms as

# the base data to apply different precooked algorithms.

  setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")

  source("main.R")

  myCorrectedConsumption <- list()
  #For some strangre reasson the day 30th June 2016 is skept in the CSV original file so
we add it manually as we had already

#calculated the consumption on this days in the script "main". We sourced the script at
the beggining of this script so we

#are good to go.

###### Preprocessing the whole period data set ######

  setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/Datos/Consumo/CSV/Total/")

  totalconsumption <- read.csv("wholeperiod.csv", sep =";", header = FALSE)[-c(1:15),
1:3]

  names(totalconsumption) = c("Day", "Hour", "Consumption")

  totalconsumption$Day  <- as.POSIXct(as.character(totalconsumption$Day), format =
"%d/%m/%Y")

  totalconsumption$Date <- paste(totalconsumption$Day, totalconsumption$Hour)

  totalconsumption$Date <- as.POSIXct(as.character(totalconsumption$Date), format =
"%Y-%m-%d %H:%M")

  totalconsumption <- totalconsumption[,c(1,2,4,3)]

  totalconsumption$Hour <- factor(totalconsumption$Hour, levels =
levels(totalconsumption$Hour)[2:25])

  totalconsumption$Consumption <- as.numeric(gsub(",",
"." ,totalconsumption$Consumption))

# We need to check that we have all the values for the period considered in the data
frame.

# That is:

  # 912 days

  # 21888 hours (same number of rows that the df should have)

        length(unique(totalconsumption$Day))
        nrow(unique(totalconsumption))

        #The 30th of June from 00:00 to 23:00 is just missing values.Substitute NAs in
the corresponding dataframe that belongs to the list of dataframes we created (If we
modified)

           detach("package:plyr")

           fixjune <-  totalconsumption %>%

             filter((Date >= "2016-06-27 00:00" & Date <="2016-06-29 23:00")) %>%

             group_by(Hour) %>%

             summarise(Consumption = round(mean(Consumption), digits = 0))

           fixjune <- as.data.frame(fixjune)

           fixjune$Day <- as.POSIXct(rep("2016-06-30",24), format = "%Y-%m-%d")

           fixjune$Date <- as.POSIXct(paste(fixjune$Day, fixjune$Hour, sep = " "),
format = "%Y-%m-%d %H:%M")

           fixjune <- fixjune[,c(3,1,4,2)]

           fixfebruary <- totalconsumption %>%

             filter(Date >= "2017-02-13 11:00" & Date < "2017-02-13 15:00" | Date >=
"2017-02-15 11:00" & Date < "2017-02-15 15:00") %>%

             group_by(Hour) %>%

             summarise(Consumption = round(mean(Consumption), digits = 0))



           fixfebruary <- as.data.frame(fixfebruary)

           fixfebruary$Day <- as.POSIXct(rep("2017-02-14",4), format = "%Y-%m-%d")

           fixfebruary$Date <- as.POSIXct(paste(fixfebruary$Day, fixfebruary$Hour, sep =
" "), format = "%Y-%m-%d %H:%M")

           fixfebruary <- fixfebruary[,c(3,1,4,2)]

           totalconsumption <- rbind(totalconsumption, fixjune, fixfebruary)
```

```
         totalconsumption <- arrange(totalconsumption, totalconsumption$Date)

      #After fixing the known missed values we check againt if there is still missed
hours in the df.

  # A SECOND PREPROCESS TO SET THE DATAFRAME IN TO THE CORRECT FORMAT

      library(plyr)

      totalconsumption$DayType <- weekdays(totalconsumption$Day)

      totalconsumption$DayType <- mapvalues(totalconsumption$DayType,

                                            from = c("lunes", "martes", "miércoles",
"jueves", "viernes", "sábado", "domingo"),

                                            to = c("Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday" , "Sunday"))

      totalconsumption$DayType <- as.factor(totalconsumption$DayType)

      totalconsumption$DayType <- factor(totalconsumption$DayType, levels = c("Monday",
"Tuesday", "Wednesday", "Thursday", "Friday", "Saturday" , "Sunday"))

      totalconsumption$MonthGen <- month(totalconsumption$Date)

      totalconsumption$MonthGen <- mapvalues(totalconsumption$MonthGen,

                                                                     from = c("1",
"2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12"),

                                                                     to = c("January",
"February", "March", "April", "May", "June" , "July", "August", "September", "October",
"November", "December"))

      totalconsumption$MonthGen <- as.factor(totalconsumption$MonthGen)

      totalconsumption$MonthGen <- factor(totalconsumption$MonthGen, levels =
c("January", "February", "March", "April", "May", "June" , "July", "August",
"September", "October", "November", "December"))

      totalconsumption$Month <- paste(totalconsumption$MonthGen,
year(totalconsumption$Date), sep = "")
```

```
      totalconsumption$DayCategory <-   totalconsumption$DayType

      totalconsumption$DayCategory <- mapvalues(totalconsumption$DayCategory,

                                          from = c("Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday" , "Sunday"),

                                          to = c("WORKINGDAY", "WORKINGDAY",
"WORKINGDAY", "WORKINGDAY", "SEMIWORKINGDAY", "OFFDAY" , "OFFDAY"))

      totalconsumption <- totalconsumption[, c(3,1,2,7,6,5,8,4)]

    #Setting more specifically the holidays. Set the WORKING CALENDAR for the company
and check what is happening these days

        setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")

        source("workingcalendar.R")

        for (i in 1:nrow(totalconsumption)) {

          if (totalconsumption$Day[i] %in% holidaysTOTAL) {

            totalconsumption$DayCategory[i] <- "OFFDAY"

          }

        }

        for (i in 1:nrow(totalconsumption)) {

          if (totalconsumption$Day[i] %in% semiholidaysTOTAL) {

            totalconsumption$DayCategory[i] <- "SEMIWORKINGDAY"

          }

        }

        for (i in 1:nrow(totalconsumption)) {

          if (totalconsumption$Day[i] %in% priorholidaysTOTAL) {

            totalconsumption$DayCategory[i] <- "SEMIWORKINGDAY"

          }
        }

    # We add another important predictor to work with. The outdoor temperature at each
point measure.

        setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")

        source("weathertotal.R")

        for (i in 1:nrow(totalconsumption)) {

            if(totalconsumption$Date[i] %in% temperatures$Superdate) {

              totalconsumption$Temperature[i] <-
temperatures$Temp[which(temperatures$Superdate == totalconsumption$Date[i])]

            } else {

              totalconsumption$Temperature[i] <- NA

            }

          }

        totalconsumption <- totalconsumption[,c(1,2,3,4,5,6,7,9,8)]

            #Fill In NAs by interpolation

            totalconsumption$Temperature <-
round(zoo::na.fill(totalconsumption$Temperature, c("extend")), digits = 0)


   # Create variable to check load in previous period

          totalconsumption$DatePriorDay <- totalconsumption$Date - 60*60*24

          totalconsumption$DayBefore <- rep(0)

          vectordaybefore <- c()

            for (i in 1:nrow(totalconsumption)){

              if(i < 24+1) { vectordaybefore[i] <- NA} else {
```

```
              vectordaybefore[i] <- totalconsumption$Consumption[i-(24)]

            }

          }

        totalconsumption$DayBefore <- vectordaybefore

        totalconsumption$DatePriorWeek <- totalconsumption$Date - 60*60*24*7

        totalconsumption$WeekBefore <- rep(0)

        vectorweekbefore <- c()

         for (i in 1:nrow(totalconsumption)){

         if(i < 24*7+1) { vectorweekbefore[i] <- NA} else {

             vectorweekbefore[i] <- totalconsumption$Consumption[i-(24*7)]

          }

        }

       totalconsumption$WeekBefore <- vectorweekbefore

  #Once the data set has been filled it is split so the monthly separated consunption
can be graphed

      monthlyConsumption <- split.data.frame(totalconsumption, totalconsumption$Month)

      monthlyConsumption[[31]] <- as.data.frame(totalconsumption)

      names(monthlyConsumption)[[31]] <- "totalconsumption"

         #In order to save all the graphs to my working directoy this for structure is
used . This bucle creates a list storing all the graphs in variables

          setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")

          source("tesisfunctions.R")

          for ( i in 1:length(monthlyConsumption) ) {

            setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/ConsumoMensual/")

            mesname <- names(monthlyConsumption)[i]
            png(paste(mesname, ".png", sep=""), width = 1000, height = 500)

            mygraph <- graphmyconsumption(i, monthlyConsumption)

            print(mygraph)

            dev.off()

            rm(mesname,i, mymonth, mygraph)

          }

        #In order to plot the whole period more visible we set the conditions for the
whole dataframe:

          setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/graficas/ConsumoMensual/")

          mesname <- names(monthlyConsumption)[31]

          png(paste(mesname, ".png", sep=""), width = 3000, height = 500)

          mygraph <- graphmysinglelonggraph(31, monthlyConsumption)

          print(mygraph)

          dev.off()

          rm(mesname,mygraph
```

ANN Code Sample

```
library(nnet)
library(devtools)
library(caret)
setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")
source("totalconsumption.R")
# Data Set Preparation
      anndataframe <- totalconsumption[,c(1:4,6:9,11,13)]
      anndataframe <- anndataframe[-which(is.na(anndataframe$WeekBefore)),]
      anndataframe$Month <- as.factor(month(anndataframe$Date))
      anndataframe <- anndataframe[,c(1:7,9,10,8)]
      dumsHour.ann <- dummy(anndataframe$Hour, sep = "_")
```

```
       dumsMonth.ann <- dummy(anndataframe$Month, sep = "_")
       dumsDayTipe.ann <- dummy(anndataframe$DayType, sep = "_")
       dumsDayCategory.ann <- dummy(anndataframe$DayCategory, sep = "_")
       anndataframe.dum <- anndataframe
       anndataframe.dum <- cbind(anndataframe.dum, dumsHour.ann, dumsMonth.ann,
dumsDayTipe.ann, dumsDayCategory.ann)
       anndataframe.dum <- anndataframe.dum[,c(1:9,11:56,10)]
# ANN MODEL 1
       set.seed(123)
       training.ann.1.idx <- createDataPartition(anndataframe.dum$Consumption, p = 0.7,
list = FALSE)
       rm(.Random.seed, envir=globalenv())
       training.ann.1 <- anndataframe.dum[training.ann.1.idx,]
       testing.ann.1  <- anndataframe.dum[-training.ann.1.idx,]
       model.ann.1 <- nnet(Consumption/152 ~ ., data = training.ann.1[, c(7:55,56)],
size = 6, decay = 0.1, maxit = 1000, linout = TRUE)
       predictions.ann.1 <- predict(model.ann.1,testing.ann.1)
       predictions.ann.1 <- predictions.ann.1*152
       checking.ann.1 <- checking.ann.1[,c(1:9,56,57)]
       names(checking.ann.1)[11] <- "Forecast"
       checking.ann.1 <- resultsevaluationANN(checking.ann.1)
       ErrorsTable.ann.1 <-  myerrorcalculationsBIS(checking.ann.1,"ann.1")
       ErrorsTable.ann.1
       Graphs.ann.1 <- graphmyresults(checking.ann.1, "ANN.1")
```

kNN Code sample

```
##We will build a knn model to predict CONSUMPTION #based on all other predictors
except Date.
library(caret)
library(scales)
library(FNN)
setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")
source("totalconsumption.R")
knndataframe <- totalconsumption[,c(1:4,6:9,11,13)]
knndataframe <- knndataframe[-which(is.na(knndataframe$WeekBefore)),]
knndataframe$Month <- as.factor(month(knndataframe$Date))
knndataframe <- knndataframe[,c(1:7,9,10,8)]
dumsHour.knn <- dummy(knndataframe$Hour, sep = "_")
dumsMonth.knn <- dummy(knndataframe$Month, sep = "_")
dumsDayTipe.knn <- dummy(knndataframe$DayType, sep = "_")
dumsDayCategory.knn <- dummy(knndataframe$DayCategory, sep = "_")
knndataframe.dum <- knndataframe
knndataframe.dum <- cbind(knndataframe.dum, dumsHour.knn, dumsMonth.knn,
dumsDayTipe.knn, dumsDayCategory.knn)
knndataframe.dum <- knndataframe.dum[,c(1:9,11:56,10)]
knndataframe.dum$Temperature <- rescale(knndataframe.dum$Temperature)
knndataframe.dum$DayBefore <- rescale(knndataframe.dum$DayBefore)
knndataframe.dum$WeekBefore <- rescale(knndataframe.dum$WeekBefore)
#Up to here we have prepared the dataset to Apply k NN model. We have deleted all the
NAs so it work. It is pending that we can add all the temperatures creating
#a model with the temperatures of the local stations and local data. So far we just
erase NAs contiaing rows.

#Model 1. Following instructions of cookbook. This is training with a RANDOM SAMPLE.
Not arranged by TIME. IMplication?
        rm(.Random.seed, envir=globalenv())
        set.seed(123)
        t.knn.1.idx <- createDataPartition(knndataframe.dum$Consumption, p = 0.7,
list = FALSE)
        trg.kNN.1     <- knndataframe.dum[t.knn.1.idx,]
        rest.kNN.1    <- knndataframe.dum[-t.knn.1.idx,]

        rm(.Random.seed, envir=globalenv())
        set.seed(123)
        v.knn.1.idx <- createDataPartition(rest.kNN.1$Consumption, p = 0.5, list =
FALSE)
        val.kNN.1 <- rest.kNN.1[v.knn.1.idx,]
        test.kNN.1 <- rest.kNN.1[-v.knn.1.idx,]

        model.kNN.1   <- knn.reg(trg.kNN.1[, c(7:55)], rest.kNN.1[, c(7:55)],
trg.kNN.1[,56], 2, algorithm = "brute")

        checking.kNN.1 <- resultsevaluation(rest.kNN.1, model.kNN.1)
        checking.kNN.1 <- checking.kNN.1[,c(1:9,56:60)]
        ErrorsTable.kNN.1 <-  myerrorcalculationsBIS(checking.kNN.1,"kNN.1")
        ErrorsTable.kNN.1

        Graphs.kNN.1 <- graphmyresults(checking.kNN.1, "kNN.1")
# DATES

        Date1 <- "2015-01-01 00:00"
        Date2 <- "2016-05-01 00:00"
        Date3 <- "2017-06-01 00:00"
        Date1comparison <- "2017-01-01 00:00"
        Date2comparison <- "2017-06-01 00:00"
        Datevalidation <- "2017-07-01 00:00"

# kNN MODEL 2

       trg.kNN.2     <- knndataframe.dum %>% filter(Date >= Date1 &  Date <  Date2)
       rest.kNN.2    <- knndataframe.dum %>% filter(Date >= Date2 &  Date <
Date2comparison)

       model.kNN.2   <- knn.reg(trg.kNN.2[, c(7:55)], rest.kNN.2[, c(7:55)],
trg.kNN.2[,56], 2, algorithm = "brute")

       checking.kNN.2 <- resultsevaluation(rest.kNN.2, model.kNN.2)
       checking.kNN.2 <- checking.kNN.2[,c(1:9,56:60)]

       ErrorsTable.kNN.2 <-  myerrorcalculationsBIS(checking.kNN.2, "kNN.2")
       ErrorsTable.kNN.2

       Graphs.kNN.2 <- graphmyresults(checking.kNN.2, "kNN.2")
```

Random Forest Code Sample

```
      library(randomForest)
      library(caret)

 setwd("C:/Users/Alvaro/Dropbox/TESIS/R tesis/")
 source("totalconsumption.R")

      rfdataframe <- totalconsumption[,c(1:4,6:9,11,13)]
      rfdataframe <- rfdataframe[-which(is.na(rfdataframe$WeekBefore)),]
      rfdataframe$Month <- as.factor(month(rfdataframe$Date))

      rfdataframe <- rfdataframe[,c(1:7,9,10,8)]

      rfdataframe <- totalconsumption[,c(1:4,6:9,11,13)]

      rfdataframe <- rfdataframe[-which(is.na(rfdataframe$WeekBefore)),]
      rfdataframe$Month <- as.factor(month(rfdataframe$Date))

      rfdataframe <- rfdataframe[,c(1:7,9,10,8)]

      dumsHour.rf <- dummy(rfdataframe$Hour, sep = "_")
      dumsMonth.rf <- dummy(rfdataframe$Month, sep = "_")
      dumsDayTipe.rf <- dummy(rfdataframe$DayType, sep = "_")
      dumsDayCategory.rf <- dummy(rfdataframe$DayCategory, sep = "_")

      rfdataframe.dum <- rfdataframe
      rfdataframe.dum <- cbind(rfdataframe.dum, dumsHour.rf, dumsMonth.rf,
dumsDayTipe.rf, dumsDayCategory.rf)

      rfdataframe.dum <- rfdataframe.dum[,c(1:9,11:56,10)]


# MODEL RF 1

      rm(.Random.seed, envir=globalenv())
      set.seed(123)
      t.rf.1.idx <- createDataPartition(rfdataframe.dum$Consumption, p = 0.7, list =
FALSE)

      trg.rf.1    <- rfdataframe.dum[t.rf.1.idx,]
      rest.rf.1   <- rfdataframe.dum[-t.rf.1.idx,]
tic()
      model.rf.1 <- randomForest(x = trg.rf.1[,7:55],
                                 y = trg.rf.1[,56],
                                 ntree=500,
                                 xtest = rest.rf.1[,7:55],
                                 ytest = rest.rf.1[,56],
                                 importance=TRUE,
                                           keep.forest=TRUE)
toc()
      checking.rf.1 <- cbind(rest.rf.1, model.rf.1$test[1])
      checking.rf.1 <- checking.rf.1[,c(1:9,56,57)]
      names(checking.rf.1)[11] <- "Forecast"
      checking.rf.1 <- resultsevaluationANN(checking.rf.1)


      ErrorsTable.rf.1 <-  myerrorcalculationsBIS(checking.rf.1,"mRF.1")
      ErrorsTable.rf.1

      Graphs.rf.1 <- graphmyresults(checking.rf.1, "mRF.1")

     plot(model.rf.1, log = "y")
     plot(model.rf.1 , pch=24, cex.lab=1.5, cex.axis = 1.5)
```