MASTER'S THESIS

Interuniversity Master in Statistics and Operations Research UPC-UB

Title: Optimal management for innovative electric vehicle charging station with machine learning forecasting modules.

Author: Cardoner Valbuena, David

Advisor: Corchero García, Cristina; Nuñez del Toro, Cristina.

Department: Department of Statistics and Operations Research.

University: Universitat Politècnica de Catalunya

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

UNIVERSITAT DE BARCELONA

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Optimal management for innovative electric vehicle charging station with machine learning forecasting modules

David Cardoner

Advisor: Corchero García, Cristina; Nuñez del Toro, Cristina

Thanks to my parents for allowing me to cover all my curiosities. Thanks to my IREC colleagues for wasting their time helping me and my supervisors for allowing me to carry out this project

Abstract

In many situations the resources in the organizations are focused on the improvement of processes in order to reduce their costs. Seen in this way, the use of an energy management system can offer a reduction of costs in both the economic and technical aspects. From the technical point of view, we extend the useful life of the elements of the system, while from the economic point of view, we will try to minimize the costs of complying with the constraints of the problem. In order to develop the optimization model included in the energy management system, some inputs are necessary, such as, for example, the demand of the system or the price of energy for the next period of time considered. In this work, two families of machine-learning models are applied to forecast system demand, one based on decision trees and the other based on neural networks. The use of these models is currently being extended due to the improvement of the computing and processing capacity of the computers. In addition to this phenomenon, the growth of the available data volume allows to have a very broad knowledge of the behavior of a recurrent process, making these models able to learn the behavior of the series.
The objective sought in the present study is to optimize a system integrated by an electric vehicle charger, an energy storage system, a building and a cogeneration plant. For this, an optimization model focused on reducing system costs has been developed, deciding in each period of time what amount of energy to acquire, at what level to start the cogeneration plant and what to do with the potential energy. The use of a cogeneration plant in the system makes it necessary to take into account the generation of heat derived from the ignition of the machine when modeling the problem. Finally, in order to know the amount of energy, both electrical and thermal that will be required, a machine-learning model obtained from comparing different models and selecting the one that minimizes a given metric are used.

Keywords: energy management system, machine-learning, cogeneration plant, decision trees, neural networks, energy storage system

Resumen

En muchas situaciones, los recursos en las organizaciones se focalizan en la mejora de procesos para así reducir sus costes a futuro. Visto de este modo, la utilización de un sistema de gestión eléctrico puede ofrecer una reducción de costes tanto en la parte económica como en la técnica. Desde el punto de vista técnico, se incluyen las restricciones técnicas necesarias para alargar la vida útil de los elementos del sistema mientras que desde el punto de vista económico, se minimizaran los costes de cumplir las restricciones del sistema.
Para poder desarrollar el modelo de optimización a incluir en el gestor energético algunos inputs son necesarios, como por ejemplo, la demanda del sistema o el precio de la energía para el siguiente periodo de tiempo que se considerara de un día. Para obtener estos valores, se aplican dos famílias de modelos de machine-learning, uno basado en arboles de decisión y otro basado en redes neuronales. El uso de estos modelos se está extendiendo actualmente debido a la mejora de la capacidad de procesamiento y de cálculo de los ordenadores. Además de este fenómeno, el crecimiento del volumen de datos disponible permite tener un conocimiento muy amplio del comportamiento de un proceso recurrente, haciendo que estos modelos puedan aprender el comportamiento de la serie.
El objetivo buscado en el presente estudio es optimizar un sistema integrado por un cargador de vehiculo eléctrico, un sistema de almacenamiento eléctrico, un edificio y una planta de cogeneración. Para ello, se ha desarrollado un modelo de optimización focalizado en reducir los costes del sistema, decidiendo en cada período de tiempo que cantidad de energia adquirir, a que nivel encender la planta de cogeneración y que hacer con el potencial excedente. La utilización de una planta de cogeneración en el sistema hace que se sea necesrario tener en cuenta la generación de calor derivada del encendido de la planta al modelizar el problema. Finalmente, para poder conocer la cantidad de energía, tanto eléctrica como térmica que se requerirá, se emplea un modelo de machine- learning obtenido de comparar distintos modelos y seleccionar el que minimize una métrica determinada.

Keywords: sistema de gestión, machine-learning, maquina de cogeneración, arboles de decisión, redes neuronales

I

Contents

Chapter 1
Introduction

The increase in the use of electric vehicles (EV) planned for the next few years will require an important fast charging infrastructure to cover all users mobility needs [4],[16]. In urban environments in which the electrical distribution network may be close to the limit of its capacity, the installation of fast chargers, whose electric power is 50 kW per load point, could lead to costly investments to strengthen the network and increase its capacity. COFAST solves this problem by generating in-situ electricity, thus avoiding charging through the network.
This work is the extension of the COFAST model developed in IREC (Institut de Recerca en Energia de Catalunya).In this previous model, the optimization of the cogeneration plant was decided only based on the building demand, and the storage system and the charging point management were not integrated. The aim of this thesis is to design, build and evaluate an integral solution that consists of a fast-charging station for electric vehicles (EV) powered with a cogeneration plant that works with natural gas and supported by an electric storage system. The objective is reducing the grid-dependence of electric vehicle charging stations (EVSE) and increasing the overall energy efficiency.
The energy management system described is combining two interests, one focused in the technical requirements of the system like power limitations and other part focused on an economic model that minimizes the total users costs. The model takes into account the costs (gas natural costs and maintenance costs), the costs of the energy imported from the grid and the benefits obtained from the energy exported to the grid.
The model will be integrated in the energy management system of charging points for electric vehicles and a building with high termal demand, such as a sports center or a district heating network. The use of a cogeneration plant allows much better use of energy than the most common solutions (normally grid electricity and natural gas boilers) since the waste heat from electrical generation is used to satisfy the thermal demand of the building.
It has been shown that the use of electric vehicles and natural gas compared to other fuels has a very positive impact on air quality in cities. The use of these technologies contribute to the fulfillment of the objectives 2020 of energy and environment set by the European Commission with the horizon in 2020.
To obtain a realistic solution this model needs two external inputs that will be the power demand and water temperature. For this purpose, different machine-learning

techniques have been analyzed and evaluated.
Nowadays, the introduction of deep models has generated the possibility of training models with a high volume of data. These models should allow the learning of existing patterns in the data and giving accurate predictions. To contrast this idea a series of much simpler algorithms focused on making partitions of the subspace of the variables of the model were applied. The partitioning of the space is made by recursive binary partioning. Then, it will be interesting to test if tree methods will be able to measure the effect of previous states just separating the values by a rule generated by some feature.
The idea behind these methodologies come from [1] and [14], where both methodologies were applied to forecast time series problems. In [2] models like Convolutional Neural Networks (CNN) and Long Short Term Memory Networks (LSTM) were applied in predicting power demmand. In [15], both Recursive Partioning (RPART) and Conditional Inference Trees (CTREE) were applied to forecast time series using a Bootstrap aggregating (Bagging) approach to give stability to the model.
One of the objectives of this thesis is to find the best machine learning model not only in terms of accuracy but also in terms of computation times. It's important to take care of computation time because this model should be added to a production environment and provide a scalable solution.
In order to provide forecasts to define this inputs, a proces integrated by two modules, one that will forecast the next day for electrical demand and one to forecast the water temperature will be presented. To obtain the best model, a set of models will be performed over the chosen days and it's metrics compared ([24]),([25]).
In the following chapters, we give the details of the problems we have addressed and the formulations and solution techniques we have developed. In Chapter 1 a general introduction is described. This part also includes an overview of the system.Then, in Chapter 2, the theory behind the forecasting algorithms is detailed and the different models applied are defined formally, as well as the main features of the family of problems addressed in this thesis. In Chapter 3 COFAST is formally defined, detailing the diferent parameters, variables and constraints. In Chapter 4, results are presented. Finally, Chapter 5 includes the comparison of the structure of the solutions obtained for the studied policies. This comparison provides guidelines for the best scheduling policy to use for the recurrent service and best forecasting model to generate accurate descriptions of thermal temperature and power consumption.

1. System overview

The novelty of the COFAST solution (Fig. 1) for fast charging stations lies on two elements:

- CHP Plant: A Combined Heat and Power supply system. Generates energy using natural gas as input. In this process also generates residual heat used to heat water.
- EMS: Energy Management System. System of computer tools used by operators to monitor, control, and optimize the performance.

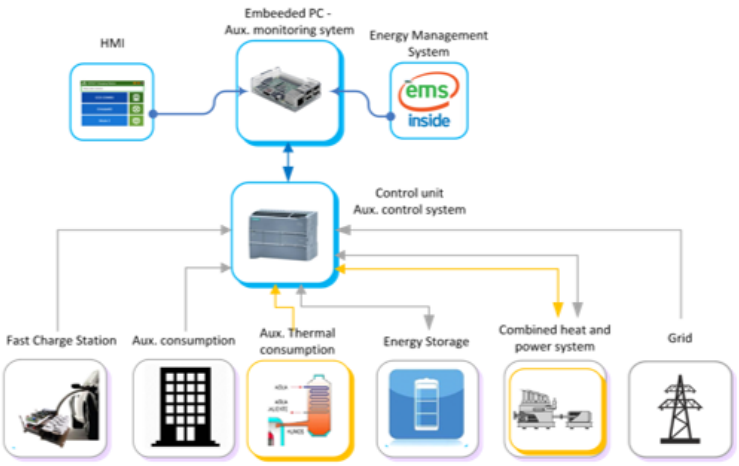

FIG. 1. Management system workflow.

The system described consist of an integrated fast charging station fueled by a small CHP system (142 kW electric power and 207 kW thermal power), which aims to provide fast charging services (54kW) to electric vehicles, satisfy electrical demands from an adjacent building, or to export to the grid. The system also includes a lithium ion battery with a capacity of 25 kWh.
Now, a definition of the parts that integrate the model will be performed to give a detailed analysis of the system.

- **System engine :** Transforms gas natural into electricity and residual heat produced in the gas natural combustion. This heating is directly transferred to the water heating circuit that is connected to a district heating and cooling grid. The CHP runs in parallel to the electric grid and will control the offered electric power output, and adjust it to the required power output sent from the EMS.
- **Charging Station:** High power converter, responsible to adapt three-phase electrical energy into the levels and modes required by the EV. The system will change the Maximum available power in the EVSE in order to guarantee the technical and contractual limits in the grid connection.

- **Storage system :** Composed by an ion lithium battery and battery charger/ inverter. In this part of the system, the management system will measure power set-points and check that the voltage and limits sent by the battery are not exceeded (system constraints).
- **Control system:** The brain of the management system. The software that controls the whole system. Includes the optimization model, sends/receives data from an industrial PLC, control the cooling/heat recovery circuit and some security sequences (for emergency stop).

2. Objectives

The objectives of the present work are as follows:

- To develop a machine-learning model for forecasting the next 24h electrical demand in quarter hour intervals.
- To develop a machine-learning model for forecasting the next 24h thermal demand in quarter hour intervals.
- To develop an optimization model that integrates the CHP operation, the EV charging processes, the building demand and the energy storage system management.
- To implement the forecast models in python and R.
- To implement the optimization model in python with Scip.
- To evaluate the integrated model with the input from the forecasting models.

These modules and optimization models should be implemented and tested in a software platform that will be installed in the project pilot, the implementation should be able to operate 24/7.

Chapter 2
Forecasting Methods

This section covers the definition of the different forecasting methodologies applied. Let's start first describing neural networks to focus then in tree-based methodologies.

1. Neural Networks

Artificial neural networks are computer programs that operate similarly to the neurons in the human brain (biological systems). These networks are a field of machine learning called deep learning.
To define a neural network let's imagine a black box model that takes inputs, let's call them features, and apply some transformations to achieve a result value (output). This machine learning approach has small units called neurons or units grouped into several layers that are connected to each other. Most neural networks are fully connected, which means each unit in its layer is connected tho other units in the closest layers. The connections between units are represented by the weight value, which can be either positive or negative. The amount of relation is measured with the value of these weights.
To sum up, it's important to notice all the elements that integrate a neural network.

- Input units: Units designed to recive the information (features) and attempt to learn about the process.
- Hidden layers: Layers that connects input units with output units applying transformations to change the diemensionality of the input space.
- Output units: Representation of inputs information over the applied transformations.

The real challenge here is finding the right weights (neuron value) in order to compute the correct results. To achieve that, backpropagation is applied. This methodology consists of propagating the error obtained when predicting the expected value to the previous layers of the network and, at each layer, modify a little bit the weights. This methodology uses some metric like, mean square error (MSE) to compute the difference between predicted and real value on a training set. With that error, the results are propagated throw the layers modifying a little

bit the weights of the network. After that, the network evaluates again the output and the same metric is applied. This procedure is what backpropagation does recalculate the weights taking into account the error obtained in previous iterations. One commonly used algorithm to find the set of weights that minimizes the error is gradient descent.
Gradient Descent is used while training a machine learning model. It is an optimization algorithm, based on a convex function, that modifies iteratively it's parameters to minimize a given function to its local minimum.

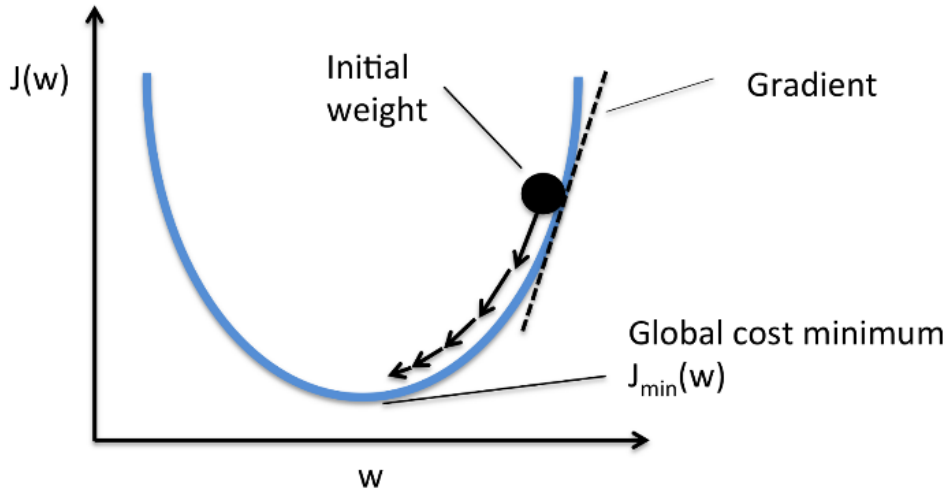

FIG. 1. Gradient descent algorithm. [19]

1.0.1. Recurrent Neural Networks (RNN). Thoughts have persistence but the neural networks about we talked before can't achieve this. If an event has to be classified to decide what kind of event is happening at the desired point, it is unclear how a neural network could be able to learn about previous events to inform later ones.
Recurrent neural networks,[7], address this problem observed in neural networks. They are networks with loops in them, allowing information to persist and this makes the idea of copy the network multiple times and passing the information to a successor each time.
The usage of this methodologies in predicting load demand is deeply used. A combination of a convolutional neural network with a recurrent neural network to forecast time series is described in [3]. These approaches outperform the ones developed using traditional methodologies such as autoregresive models.

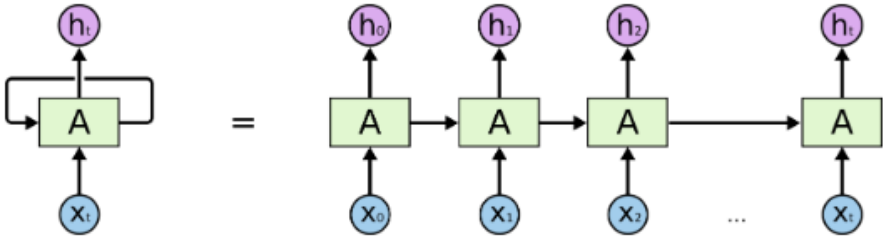

FIG. 2. Unrolled RNN. [17]

To understand better the difference, let's think about how neural networks work. A feed-forward network has no memory of the input that it received previously and only consider the current input without having any notion of time.
Recurrent neural networks are loops, and they can take into consideration what they learned from previous inputs.

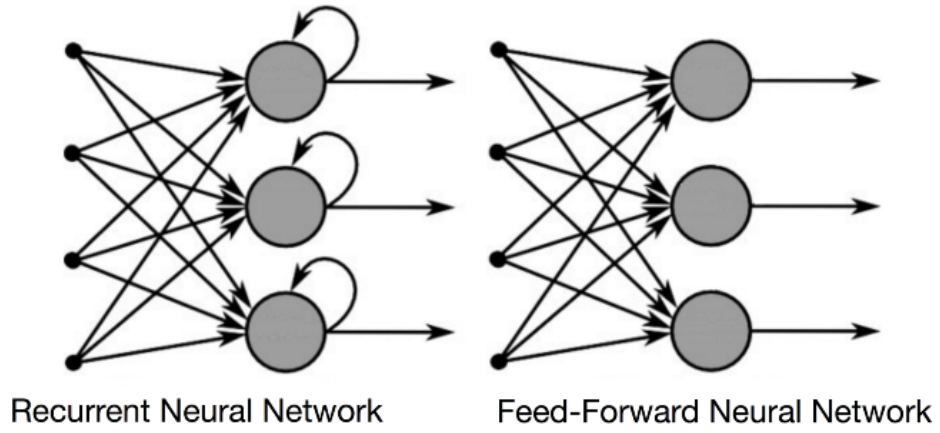

FIG. 3. RNN vs ANN. [5]

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, but when the gap between relevant information in the past and the moment to be used grows, they become unstable. This could make that the network is unable to learn to connect the information.
During the training of RNN, the information goes in loop resulting in very large updates to the model weights. This is due to the accumulation of gradient errors during an update. At an extreme, the values of weights can become so large becoming **NaN** values (exploding gradient). The same situation could occur but in the other direction. This situation is the vanishing gradient problem where values start to reduce until reach values of zero. The explosion occurs through exponential growth by repeatedly multiplying gradients through the network layers that have values larger than 1 or vanishing occurs if the values are less than 1.
In [9] problems of recurrent neural networks are treated, defining the problems of time dependence between relevant inputs and desired outputs and talking about

vanishing errors.
To solve those issues, Hochreiter and Schmidhuber proposed Long-Short Term Memory networks (LSTM). These networks are explicitly designed to avoid the long-term dependency problem. They also have this chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a special way to define which things to remember and which things to forget.
An RNN, LSTM and GRU developed to forecast the Turkish electricity market are described in [20]. The results obtained yield better results than existing ones based on traditional methodologies such as **ARIMA** or classical neural networks (**ANN**).

1.0.2. Long Short Term Memory. Long Short Term Memories (LSTM) are a kind of recurrent neural network comprised of a special architecture. The ideas behind this network are the problem of the vanishing/exploding gradient and the long term memory of networks. The architecture of the network consists of four layers that interact to decide the evolution of information.
Before start explaining the LSTM workflow, let's detail a little bit the input values shown below.

- x_t : input vector
- W_c, W_o : weight matrices
- C_t : Cell state vector
- f_t : Forget gate
- i_t : Input gate
- o_t : Output gate

Now we will define the activation functions performed below.

- σ_g : sigmoid function
- σ_c : hyperbolic tangent function

The network takes three inputs. X_t is the input of the current time step. h_{t-1} is the output from the previous LSTM unit and C_{t-1} is the *memory* of the previous unit, which is the most important input. As for outputs, h_t is the output of the current network. C_t is the memory of the current unit.
Forget layer, takes h_{t-1} and x_t and with that, output a number between 0 and 1 for each value in C_{t-1}. 1 will be keep this and 0 discard this. This idea is to remember or forget things from past state attending to the value obtained in the previous step.

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

The first value is called the forget value. If you shut it, no old memory will be kept. If it's fully opened, all old memory will pass through. $f_t * C_{t-1}$
The process of store information in the cell has two parts:

- The input gate layer which decides the values that will be updated
- The **tanh** layer that creates a vector of candidate values C_t

Combining this ones we will create an update to the state

$$i_t = \sigma_g(W_i * [h_{t-1}, x_t] + b_i) \; values\ to\ update \tag{2}$$

$$\tilde{C}_t = \sigma_c(W_c * [h_{t-1}, x_t] + b_c) \; candidate\ values \tag{3}$$

A tanh layer creates a vector of all the possible values from the new input
The second value is the new memory value. New memory will come and merge with the old memory. Exactly how much new memory should come in is controlled by the second valve.
Update the old cell state C_{t-1} into the new cell state C_t. Multiply the old state by f_t (forgeting the things choosed). Then add $i_t * \tilde{C}_t$ which are the candidate values scaled by the amount of updating that we decided for each state value.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4}$$

Now, let's talk about deciding which parts of the information to pass through the connections..
First a sigmoid layer, to choose the parts of the cell state that we are going to output. Second put the cell state through tanh (values between -1 and 1) and multiply by the output of the sigmoid gate to output only the choosed parts.

$$o_t = \sigma_g(W_o[h_{t-1}, x_t] + b_o) \; to\ choose\ just\ some\ parts \tag{5}$$

$$h_t = o_t * \sigma_c(C_t) \tag{6}$$

C_t will be the current state reescaled between -1 & 1

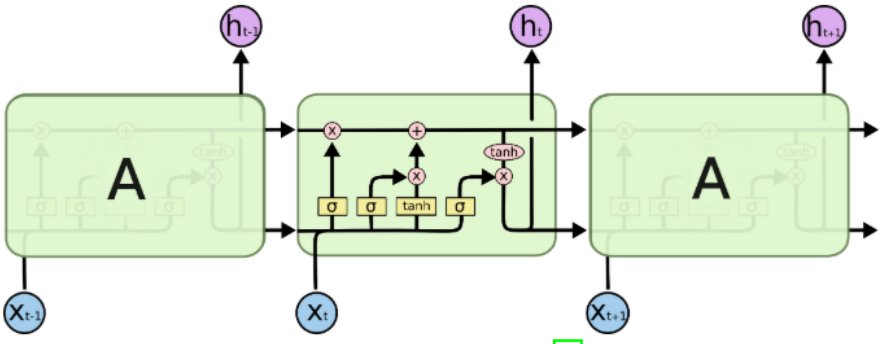

FIG. 4. LSTM Diagram.[17]

On the LSTM diagram, the top *pipe* is the memory *pipe*. The input is the old memory (a vector). The first cross that passes through is the forget value. It is actually an element-wise multiplication operation, so if you multiply the old memory C_{t-1} with a vector that is close to 0, that means you want to forget most of the old memory. You let the old memory goes through if your forget valve equals 1.
The tested LSTM structures will be treated in Model Development section so first, gradient and vanishing exploding problems will be explained.
LSTM was invented specifically to avoid the vanishing gradient problem. The

long dependencies between sequence data is called long-term dependencies because the distance between the relevant information and the point where it's needed to make a prediction is very wide. As the distance grows, RNN has a hard time of computation to learn this dependency because it encounters either vanishing or exploding gradient problem. These problems arise during the training of a deep network. Layers go through multiple matrix multiplications making that when arrive at earlier layers could have small values. If they are smaller than one then they tend to shrink exponentially to zero. While on the other hand if they have large values (greater than one) they get larger and eventually blowing up and crashing the model, this is the exploding gradient problem. Both things are problematic and that's why LSTM appeared, to solve that problem or try it at least.
Now, let's define mathematically the problem. Let's define the one-dimensional case.
Being a hidden state h_t at time step t. Defining it simple without biases we have:

$$h_t = \sigma(wh_{t-1}) \tag{7}$$

Taking the derivative of h_t,

$$\frac{\partial h_{t'}}{\partial h_t} = \prod_{k=1}^{t'-t} w\sigma'(wh_{t'-k}) = w^{t'-t} \prod_{k=1}^{t'-t} \sigma'(wh_{t'-k}) \tag{8}$$

The factored $w^{t'-t}$ is the crucial one. If the weight is not equal to 1, it will either decay or grow to zero exponentially fast.
In LSTMs, you have the cell state s_t. The derivative there is of the form

$$\frac{\partial s_{t'}}{\partial s_t} = \prod_{k=1}^{t'-t} \sigma(v_{t+k}) \tag{9}$$

Here v_t is the input to the forget gate. There is no exponentially fast decaying factor involved. Consequently, there is at least one path where the gradient does not vanish.
To deal with exploding or vanishing gradients some methods have proven to be effective, like using an L1 or L2 penalty on the recurrent weights or simply use LSTM models. **Tanh** activation is used in LSTM's because control these problems. Is a function whose second derivative can sustain for a long range before going to zero. The suitability of this function is its velocity of convergence which is found to converge faster in practice.

$$tanh(z) = \frac{e^z - e^- z}{e^z + e^- z} \tag{10}$$

These methodologies are effective but one of the most useful to solve this problem is gradient clipping; which places a predefined threshold on the gradients to prevent it from getting too large, and by doing this, it doesn't change the direction of the gradients it only changes its length. The proposed clipping is simple and computationally efficient, but it does however introduce an additional hyper-parameter, namely the threshold.

$$\|g\| > threshold \tag{11}$$

$$g = \frac{threshold * g}{\|g\|} \tag{12}$$

where **g** is the gradient and $\|g\|$ is the norm of the gradient.
Model structure implemented in this thesis was extracted from [1]. LSTM architectures were modified, adding some useful features related to the forecast variable/s. After LSTM architecture, the other approach evaluated in this thesis is presented. This method is the Convolutional LSTM, a methodology that combines convolutions with long short term memory layers.

1.0.3. Convolutional LSTM. First, let's define what a convolution is [17]. A convolution is a mathematical operation on two functions, let's call them **s** and **g**, to produce a third function that will express how the shape of one is modified by the other. To analyze the idea we will use an example of an object that tries to reach a position **c** by realizing two movements, a and b. After the first drop, the ball will move a units away from the starting point with probability $s(a)$ where s is the probability distribution function. Then we pick the ball from a and drop it again reaching a distance of b. The probability that reaches this distance will be $g(b)$. Then, fixing that $c = a + b$ we will have that the related probability will be $s(a) * g(b)$. Then to evaluate the likelihood function of the object reaching a distance of c we have to consider all the possibilities of reach c partitioning space in a and b. This solution is achieved by applying the sum of all the combinations of $s(a) * g(b)$, which can be defined as,

$$\sum_{a+b=c} s(a) * g(b) \tag{13}$$

In particular, the convolution of s and g evaluated at c is defined as:

$$(s * g)(c) = \sum_{a+b=c} s(a) * g(b) \tag{14}$$

and subsituting $b = c - a$, we will have the definition of a convolution, which is:

$$(s * g)(c) = \sum_{a+b=c} s(a) * g(c - a) \tag{15}$$

But, what are convolutional neural networks. In a CNN, like with neural networks, each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. This kind of networks, expect that data has a three-dimensional structure. The idea is that it can take an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. Convolutional networks use convolutions and pooling to capture spatially local patterns. The interesting part is that this convolutional approach could be applicable to a regression task due to spatial correlations within the data. During training, the weights in the kernel are optimized to detect relevant spatial patterns over a small region. After the convolution, we apply a nonlinear activation function.

After processing the grid with one or more convolution filters and flatten the output we pass both flattened outputs and the previous hidden states to an LSTM layer. Finally, the LSTM sends an output, which is then reshaped and used both to predict the next step and as an input at the next time step.
The structure of CNN-LSTM was developed respecting the structure found in [2]. CNN-LSTM model involves using Convolutional Neural Network (CNN) layers for feature extraction on input data combined with LSTMs to support sequence prediction.
The CNN does not directly support sequence input; instead, a 1 dimension CNN is capable of reading across sequence input. These can then be interpreted by an LSTM decoder. The approach evaluated in this document refers to hybrid models that use a CNN and LSTM. The CNN expects the input data to have the same 3-dimensional structure as the LSTM model.
The defined architecture comprises two convolutional layers followed by a max pooling layer, where:

- Fully-Connected layer: Every node in the first layer is connected to every node in the second layer. Normally this layer will be used as a final layer to output the final predicted value.
- Convolutional layer: A convolutional layer can be seen as a feature extractor. It takes the input and apply a special transformation on it. As a result, each of the transformed values get the most interesting part of the original input.

The results of the structures are then flattened into 1-dimensional array (fully-connected). The first convolutional layer acts across the input sequence and projects the results onto feature maps. The second performs the same operation on the feature maps created by the first layer, attempting to amplify any salient features.

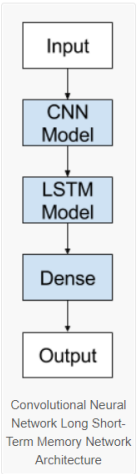

FIG. 5. Convolutional methodology [2]

The main difference in this CNN-LSTM approach is that has been developed using a GPU architecture to speed up the computation times. The main problem of this kind of networks is that has long computational time and some exploding gradient problems.

1.1. Tree-Based Models. Tree based learning algorithms are considered to be one of the best and mostly used supervised learning methods. Tree based methods empower stability and ease of interpretation. Unlike linear models, they map non-linear relationships and could be used for both classification or regression problems. Methods like decision trees, random forest or gradient boosting are tree-based methods.
The use of tree-based models to predict time series is descriibed in literature. Works such as [12] describes tree-methods such as Recursive Partitioning and Regression Trees (RPART), Conditional Inference Trees (CTREE) with Bootstrap Aggregating (BAGGING), and Random Forest (RF) models for short term load forecasting. These models are based on decision trees. A decision tree is a method in which the response variable is separated into branches based on a condition (variable). The end of the branch that doesn't split anymore is the decision (leaf) that determines the value of the response. In order to find a good solution (not overfiting) the idea is to trim it down until good and reproducible solutions are obtained. Other posible solution is to avoid growing large trees (early stopping).

1.1.1. RPART. This algorithm works by splitting the dataset recursively, which means that the subsets are further split until a predetermined termination criterion is reached. The method used to select the covariate to perform the splits in the response variable is an information measure (such as Gini index or Entropy). These methods are based on the degree of heterogeneity of the leaf nodes. For example, a leaf that contains only a single class has impurity zero.
The algorithm essentially minimizes the cost, which is a linear combination between the error (misclassified instances) $R(T)$ and the number of leaf nodes in the tree $|\hat{T}|$.

$$C_\alpha(T) = R(T) + \alpha|\hat{T}| \tag{16}$$

When $\alpha = 0$ the fully grown tree is returned. When α increases, a penalty is proportional to the number of leaf nodes. This tends to cause the minimum cost to occur for a tree that is a subtree of the original one. In practice, we vary α and pick the value that gives the sub tree that results in the smallest cross-validated prediction error.
To conclude, the idea behind this method is that the algorithm works by making the best possible choice (selected covariate) at each stage without considering if these choices remain optimal in future stages. The algorithm doesn't find a globally optimal tree.

1.1.2. CTREE. Conditional decision trees are created using statistical tests to select split points (covariates) on attributes rather than using a loss function [10]. The significance test or the multiple significance tests computed at each start of the algorithm, which consists of selecting the covariate, choose a split and recourse, are permutation tests. Permutation tests are a type of statistical significance tests in which the distribution of the statistic under the null hypothesis is obtained by calculating all possible values of the test statistic under rearrangements of the labels on the data points. If labels are exchangeable, under the null hypothesis, then the resulting tests will obtain exact significance levels.

The idea behind conditional trees focuses on two steps. First, recursive binary partitioning is applied using regression models that describe the conditional distribution of a response variable Y given the status of m covariates. Then we assume that the conditional distribution of the response Y given covariates X depends on the function of the covariates.

$$D(Y|X) = D(Y|X_1, ..., X_m) = D(Y|f(X_1, ..., X_m)) \tag{17}$$

We restrict to partition based regression relationships, for example r disjoint cells $B_1, ..., B_r$ partitioning the covariate space

$$X = \bigcup_{k=1}^{r} B_k \tag{18}$$

A model of the relationship is fitted based on a learning sample L_n, a random sample of n independent and identically distributed random variables observations (possibly with some X_{ji} omitted),

$$L_n = (Y_i, X_{1i}, ..., X_{mi});\ i = 1, ..., n \tag{19}$$

Then an algorithm for recursive binary partition for a given learning sample L_n can be formulated using weights $W = (w_1, ..., w_2)$ where each node of a tree is represented by a vector of case weights. This algorithm is then implemented in the following three steps:

- For weights W, the null hypothesis of independence between the covariates and the response is tested. If this hypothesis can't be rejected then the algorithm stops. Otherwise, the covariate X_{j*} with the strongest association to Y is chosen.
- Choose a set $A^* \in X_{j*}$ to split X_{j*} into two disjoint sets A^* and $X_{j*} \setminus A^*$. The case weights w_{left} and w_{right} determine the two subgroups with $w_{left,i} = w_i I(X_{j*i} \in A^*)$ and $w_{right,i} = w_i I(X_{j*i} \notin A^*)$ for all $i = 1, ..., n$ (I(.) denotes the indicator function).
- Recursively repeats steps 1 and 2 with modified case weights respectively.

These models were applied using an ensemble method called Bootstrapped Aggregation (Bagging). This method creates multiple models of the same type from different sub-samples of the same dataset. Finally, the predictions from each separate model are combined to provide a superior result. In this case the aggregation method were the median to obtain a value not influenced by extreme results.

1.1.3. Random Forest. This method builds an ensemble of decision trees, most of the time trained with the bagging method. The general idea behind that is the combination of learning models that increases the overall result. We can think of random forests as a bagging classifier but this method adds additional randomness to the model, while growing the trees. This method doesn't search for the most important feature in the set, it searches for the best feature among a random subset of features.
A Random forest is by definition a collection of decision trees with the principal difference being that while deep decision trees tends to suffer from overfitting, Random Forest was designed to prevent overfitting while adding randomness.

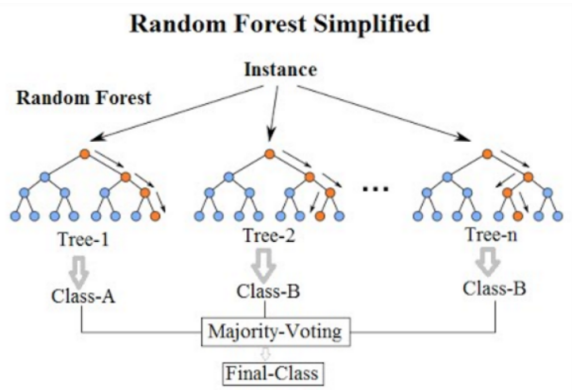

FIG. 6. Random Forest workflow [21]

1.1.4. Bagged Models. Bootstrap Aggregation is an ensemble technique that is used typically when a reduction of variance is desired.
Ensemble methods are models that combine several decisions trees classifiers to produce better predictive performance than a single decision tree classifier. The main principle behind the ensemble model is that a group of weak learners come together to form a strong learner, thus increasing the accuracy of the model. This increase in accuracy is because these methods reduce noise and bias.
In bagging, input data is sampled to generate multiple sets of input data. Then with these sets a tree-based method is applied for each tree and the combination of these weak learners is made applying some statistic. In this thesis, the median is implemented because it reduces noise by attacking the second quantile of the underlying distribution generated by trained trees. For each of those learners, the same baseline predictor (RPART & CTREE) is run to get a trained model for each set. Each tree is trained 100 times, taking samples of the 85% of the size of the training set.

1.2. Variables. Like it is said above, an important part of a time series model is the past but sometimes other variables are needed to achieve good predictions. Variables like weekday, month or the response variable laged by some time period are aplied but also other more sophisticated ideas were developed. Finally, a random forest model is used as a *feature selection* method. With this approach, after generate a big amount of variables, a random forest is applied to select the ones that explain more about the response variable.

1.2.1. Fourier terms. The Fourier Transform is a tool that breaks a waveform (a function or signal) into an alternate representation, characterized by sine and cosines. The Fourier Transform shows that any waveform can be re-written as the sum of sinusoidal functions. A function is periodic, with fundamental period T, if the following is true for all t:

$$f(t+T) = f(t) \tag{20}$$

This means that a function of time with period T will have the same value in T seconds. The fundamental period is the value of T (greater than zero) that is the smallest possible T for which Eq. 20 is always true.
A Fourier Series, with period T, is an infinite sum of sinusoidal functions (cosine and sine), each with a frequency that is an integer multiple of $1/T$ (the inverse of the fundamental period). This idea could be extended to non-periodic functions using the Fourier Transform. The Fourier Transform of a function $g(t)$ is defined by:

$$Fg(t) = G(f) = \int_{-Inf}^{Inf} g(t)e^{-2\pi ift}dt \tag{21}$$

1.2.2. Moving windows and signal split. The first approach performed is a moving window, a time series approach that consist in the generation of features applying statistics over the response variable. In this thesis, the mean and median were evaluated.
The second approach is a method for time series decomposition. Like it's said in [11], time series data can exhibit a variety of patterns, and it is often helpful to split a time series into several components, each representing an underlying pattern category.
When decomposing a time series into components time series is comprising three components: a trend-cycle component, a seasonal component, and a remainder component.
If an additive decomposition is assumed then,

$$y_t = S_t + T_t + R_t \tag{22}$$

where y_t is the data, S_t is the seasonal component, T_t is the trend-cycle component, and R_t is the remainder component, all evaluated at period t. Alternatively, a multiplicative decomposition would be written as

$$y_t = S_t \times T_t \times R_t \tag{23}$$

When deciding which of them to use, the additive decomposition is the most appropriate if the seasonal fluctuation does not vary with the level of the time series. When the variation in the seasonal pattern, or the variation around the trend-cycle, appears to be proportional to the level of the time series, then a multiplicative decomposition is more appropriate.

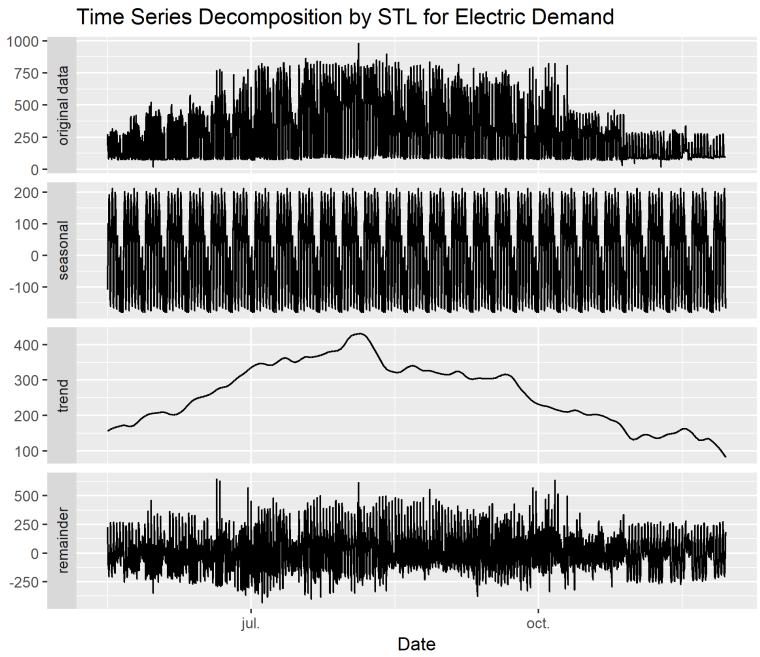

FIG. 7. STL Decomposition for electric demand

1.2.3. Feature selection. Feature selection is a process to reduce the number of variables, including only the most representative or important features. This process has three main benefits. First, increase the interpretability. Second, reduces the variance of the model and therefore the overfitting. Finally, it reduces the computational cost and time of training a model.

Random forests consist of hundred decision trees, each of them built over a random extraction of the observations from the dataset and a random extraction of the features. Not every tree sees all the features or all the observations, and this guarantees that the trees are uncorrelated and therefore less prone to over-fitting.

Every node in the decision trees is a condition on a single feature, designed to split the dataset into two so that similar response values end up in the same set. The idea of the importance of each feature is derived from how *pure* (optimal) each of the buckets is. For classification, it is typically either Gini impurity or information gain/entropy and for regression trees it is variance.

When training a tree, it is possible to compute how much each feature decreases the impurity. The more a feature decreases the impurity, the more important the feature is.

An interesting idea is that features that are selected at the top of the trees are in general more important than features that are selected at the end nodes of the trees. That's because at top splits the capacity of the model to split the response is much bigger than in nodes.

To avoid overfitting in all feature selection procedures, it is a good practice to select the features by examining only the training set.

The use of random forests as feature selection is deeply used in literature. In [13], random forests are used as a feature selection method to reduce high-dimensional

data, extracting the minimum subset of important variables. In [8] random forests are used as a tool for extracting important variables in terms of model interpretation and also to obtain a good predictive model in terms of accuracy.

1.3. Computation Times. Before talk about computational times and architectures evaluated, some keywords should be defined:

- CPU: The central processing unit is the hardware within a computer, which interprets the instructions of a computer program by performing the basic arithmetic, logic and input/output operations of the system.
- GPU: A graphics processing unit is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device.
- CUDA: CUDA is a parallel computing platform created by Nvidia. It allows using a CUDA-enabled graphics processing unit (GPU) for general purpose processing.
- cudnn: The NVIDIA CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks.

The core idea behind this is that GPU's are able to do a lot of parallel computations. A lot more than a CPU can do. While CPU's are few complex cores, GPU's are hundreds of simpler cores with a thousand of concurrent hardware threads.
We saw that the computationally intensive part of the neural network is made up of multiple matrix multiplications. We can simply do this by doing all the operations at the same time instead of doing it one after the other. This is in a nutshell why we use GPU instead of a CPU for training a neural network.

1.4. Evaluation Measures. To sum up, all the ideas explained, the best models will be tested by applying some metric evaluations. After testing the statistical methods proposed, it is necessary to evaluate the results in order to compare the different methods. With metrics, we will be able to check the quality of the models. MAPE and RMSE was tested.
Each machine learning model is trying to solve a problem with a different objective using a different dataset and hence, it is important to understand the context before choosing a metric [6]. MAPE is one of the most useful technique in time series forecasting that consist in measure the closeness of g(X) to Y_t. In this case $g(X)$ will be the forecasted value and Y_t the real value. RMSE basically measures root average squared error of our predictions. For each point, it calculates square difference between the predictions and the target and then, the values are averaged and the square root is applied. The higher this value, the worse the model is, like with MAPE.

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} |\frac{Yt - g(X)}{Yt}| \tag{24}$$

$$RMSE = \sqrt{E(Yt - g(X))} \tag{25}$$

MAPE could be sensible to the scale of data. When small values are evaluated exists the risk to achieve very high inconsistent values.
In time series, MAPE is more used in the literature so we will focus on it when diferent models will be described.

Chapter 3
Optimization Model

This section introduces the model definition of the COFAST solution and it's modelization.

1. Model Implementation

Parameters

I set of EV chargers
H set of time horizon (h)
Ts = Time step for CHP economic optimization (h)
T set of time intervals $(T) = H/T_s$
C_{GN} = Natural gas equivalent electrical price (€ /kWh)
C_M = CHP maintenance cost (€)
C_{on} = CHP start up cost (€)
C_V = cost of unserved EV demand (€ /kWh)
C_d = battery degradation cost (€ /kWh)
$Self$ = Self consumption of CHP (kWh)
$\overline{\lambda}$ = Maximum CHP operation level $\in [0, 1]$
$\underline{\lambda}$ = Minimum CHP operation level $\in [0, 1]$
$\overline{P_e}$ = Maximum CHP electrical power for $\overline{\lambda}$ (kW)
$\underline{P_e}$ = Minimum CHP electrical power for $\underline{\lambda}$ (kW)
$\overline{P_{th}}$ = Maximum CHP thermal power for $\overline{\lambda}$ (kW)
$\underline{P_{th}}$ = Minimum CHP thermal power for $\underline{\lambda}$ (kW)
$\overline{PC_i}$ = Maximum electrical power of EV charger $i \in I$ (kW)
$\overline{PBC}$ = Maximum battery charge power (kW)
$\underline{PBC}$ = Minimum battery charge power (kW)
$\overline{PBD}$ = Maximum battery discharge power (kW)
$\underline{PBD}$ = Minimum battery discharge power (kW)
$\overline{SOC_{ini}}$ = Initial state of charge (%)
$\overline{SOC}$ = Maximum battery state of charge (%)
$\underline{SOC}$ = Minimum battery state of charge (%)

$\overline{S}$ = Maximum number of CHP starts within 24 hours
CB = Battery capacity (kWh)
$\overline{PG}$ = Maximum interconnection electrical power (kW)

For $t \in T$,
C_e^t = Electricity tarif cost (€ /kWh) in period t
V_e^t = Electricity pool price (€ /kWh) in period t
V_{th}^t = Thermal power selling price (€ /kWh) in period t
DC_i^t = Electrical power demand from the EV charger $i \in I$ (kW)
DR_e^t = Electrical forecast power demand (kW) in period t
DR_{th}^t = Thermal forecast power demand (kW) in period t

Variables

CHP variables:
For $t \in T$,

$$f^t = \begin{cases} 1 & \text{if CHP machine operates in period t} \\ 0 & \text{otherwise} \end{cases}$$

$$z^t = \begin{cases} 1 & \text{if CHP start up in period t} \\ 0 & \text{otherwise} \end{cases}$$

$\lambda^t =$ CHP operation level in period t

$w^t =$ CHP Natural gas consumption in period t

$P_e^t =$ Total CHP generated electrical power in period t

$P_{th}^t =$ CHP generated thermal power in period t

$PR_e^t =$ CHP generated electrical power sent to demand in period t

$PV_e^t =$ CHP generated electrical power exported to the grid in period t

$PC_i^t =$ CHP Generated electrical power to EV station, in period t, for $i \in I$

$PB_{CHP}^t =$ CHP generated electrical power to charge battery in period t

Battery variables:
For $t \in T$,

$$x^t = \begin{cases} 1 & \text{if battery charges in period t} \\ 0 & \text{otherwise} \end{cases}$$

$PBD^t =$ Total discharged battery electrical power in period t

$PBV^t =$ Exported battery electrical power in period t

$PBC_i^t =$ Discharged battery electrical power to EV station in period t, for $i \in I$

$SOC^t =$ Battery state of charge at end in period t

$PN^t =$ Battery ramp at period t

Grid variables:
For $t \in T$,

$$v^t = \begin{cases} 1 & \text{if system imports from grid in t} \\ 0 & \text{otherwise} \end{cases}$$

$PGR^t =$ Imported interconnection electrical power to demand in period t

$PGB^t =$ Imported interconnection electrical power to battery in period t

$PGC_i^t =$ Imported interconnection electrical power to EV station in period t, for $i \in I$

The optimization model objective is to achieve the minimum costs over all time periods but with some considerations:

- The electrical demand must be fulfilled by the grid or the cogeneration plant.
- The EV demmand must be fullfilled by the cogeneration plant, the grid and the battery.
- The energy storage system could be charged by the grid or using the cogeneration plant.

First of all, let's describe the objective function used in the optimization problem.

$$
\begin{aligned}
\min \; & Ts \sum_{t \in T} C_{GN} \cdot P_e^t + C_M \cdot f^t + C_{on} \cdot z^t \\
& + Ts \sum_{t \in T} C_e^t \left(PGR^t + PGB^t + \sum_{i \in I} PGC_i^t \right) \\
& + Ts \sum_{t \in T} \left[C_V \cdot \left(DC_i^t - \sum_{i \in I} (PGC_i^t + PBC_i^t + PC_i^t) \right) \right] \\
& - Ts \sum_{t \in T} \left[V_e^t \cdot PV_e^t + V_{th}^t \cdot PR_{th}^t \right] \\
& + C_d \cdot \sum_{t \in T} PN^t
\end{aligned} \tag{26}
$$

The first term of the objective function, describes how to minimize the cost function taking into account the costs and benefits of energy purchased and sold from and to the grid respectively and also the mantainance costs of the cogeneration plant and the costs of purchasing the necesary natural gas.
The peculiarities of managing a real environment forces us to include a penalty term in the objective function in order to control some specific aspects as the total number of starting up processes, the battery charge-discharge cycles and the gap between demanded and fullfilled demand. Adding this term to the objective function will allow the model to control the amount of each variable in the processes.
The important part of the system is the capacity of integrate the system generation (building, grid and electric vehicle) and the battery system.
To facilitate the process of defining the constraints of the model, they will be divided according to the part of the system that they are restricting or limiting.

CHP Constraints.

$$z^t \geq f_t - f_{t-1} \qquad \forall t \in T \tag{27}$$

$$\sum_{t \in T} z^t \leq \overline{S} \tag{28}$$

$$z^t + z^{t+2} \leq 1 \qquad \forall t \in T \tag{29}$$

$$P_e^t \geq \left(\lambda^t - \underline{\lambda} \cdot f^t\right) \left(\frac{\overline{P_e} - \underline{P_e}}{\overline{\lambda} - \underline{\lambda}}\right) + \underline{Pe} \cdot f^t \qquad \forall T \in T \tag{30}$$

$$P_{th}^t \geq \left(\lambda^t - \underline{\lambda} \cdot f^t\right) \left(\frac{\overline{P_{th}} - \underline{P_{th}}}{\overline{\lambda} - \underline{\lambda}}\right) + \underline{P_{th}} \cdot f^t \qquad \forall t \in T \tag{31}$$

$$\underline{\lambda} \cdot f^t \leq \lambda^t \leq \overline{\lambda} \cdot f^t \qquad \forall t \in T \tag{32}$$

$$P_e^t = PB_{CHP}^t + PR_e^t + PV_e^t + Self \cdot f^t + \sum_{i \in I} PC_i^t \qquad \forall t \in T \tag{33}$$

$$P_e^t \leq \overline{Pe} \cdot f^t \qquad \forall t \in T \tag{34}$$

Eq. (27), (28), and (29) modelate the start-up of the CHP plant, where also the maximum number of start-up processes during a day is controlled. Eq. (30) and (31) define the lower bounds of power and thermal generation and the lower bound of natural gas consumption, considering a linear interpolation between the minimum and maximum technical limits. Eq. (32) sets the CHP operation level. Eq. (33) defines the power generation of the CHP plant. Finally, (34) is the upper bound for the generation of the CHP plant.

Interconnection Constraints.

$$PGB^t + \sum_{i \in I} PGC_i^t + PGR^t \leq \overline{PG} \cdot v^t \qquad \forall t \in T \tag{35}$$

$$PV_e^t + PBV^t \leq \overline{PG} \cdot (1 - v^t) \qquad \forall t \in T \tag{36}$$

Eq. (35) defines the maximum power that could be demanded by the system to the grid and Eq. (36) defines the maximum power that could be exported to the grid.

Storage system Constraints.

(37)
$$PBD^t = \sum_{i \in I} PBC^t + PBV^t \quad \forall t \in T$$

(38)
$$CB \cdot SOC^t = CB \cdot SOC^{t-1} + T_s \cdot (PGB^t + PB^t_{CHP} - PBD^t) \quad \forall t \in T: \ t > 0$$

(39)
$$CB \cdot SOC^t = CB \cdot SOC_{ini} + T_s \cdot (PGB^t + PB^t_{CHP} - PBD^t) \quad t = 0$$

(40)
$$SOC^t \geq SOC_{ini} \quad t = \|T - 1\|$$

(41)
$$\underline{SOC} \leq SOC^t \leq \overline{SOC} \quad \forall t \in T$$

(42)
$$\underline{PBC} \cdot x^t \leq PB^t_{CHP} + PGB^t \leq \overline{PBC} \cdot x^t \quad \forall t \in T$$

(43)
$$\underline{PBD} \cdot (1 - x^t) \leq PBD^t \leq \overline{PBD} \cdot (1 - x^t) \quad \forall t \in T$$

(44)
$$PBD^t - PGB^t - PB^t_{CHP} + PGB^{t-1} + PB^{t-1}_{CHP} \leq PN^t \quad \forall t \in T$$

(45)
$$PBD_{t-1} - PBD_t + PGB^t + PB^t_{CHP} \leq PN^t \quad \forall t \in T$$

Eq. (37) modelate the discharge of the battery. Eq. (38) defines the balance of the state of charge of the battery. In (41) the lower and upper bounds of the state of charge are fixed. Eq. (42) and (43) add the charge and discharge power bounds for the storage system. Finally, Eq.(44) and (45) the ramps for the battery are fixed. These values controls the charging and discharging process, increasing the life expectancy of the battery.

EV Constraints.

(46) $$PBC^t_i + PC^t_i + PGC^t_i \leq \overline{PC_i} \qquad \forall i \in I \quad \forall t \in T$$

Eq. (46) defines the upper bound for the electric vehicle (EV) charger.

1.1. Demand Constraints.

(47) $$PBC^t_i + PC^t_i + PGC^t_i \leq DC^t_i \qquad \forall i \in I \quad \forall t \in T$$

(48) $$P^t_e = DR^t_e \qquad \forall t \in T$$

(49) $$P^t_{th} = DR^t_{th} \qquad \forall t \in T$$

Eq. (47) sets how the electric vehicle (EV) demand could be fulfilled and Eq. (48) and (49) defines the power demand of the building settlement.

To conclude, Eq. (50), (51) and (52) set the non-negativity and the binary constraints.

$$P_e^t,\ PGR^t,\ PGR_{th}^t,\ w^t,\ PR_e^t,\ PV_e^t \geq 0 \qquad \forall t \in T \tag{50}$$

$$f^t,\ z^t,\ x^t \in \{0,1\} \qquad \forall t \in T \tag{51}$$

$$PC_{ei}^t,\ PBC_i^t,\ PGC_i^t \geq 0 \qquad \forall t \in T \quad \forall i \in I \tag{52}$$

Chapter 4
Results

1. Case Study

The building analyzed will be TubVerd, a distribution network of heat and cold. Tub Verd is an urban hot and cold network that transports hot water through pipes buried underground. The initiative takes advantage of the energy coming from the Waste water Treatment Plant of Mataró and from the Residus Sòlids Urbans del Maresme (CTRSUM). The system heats water for showers, for swimming pools as well as air to climatize all zones or generate cold for air conditioning.
One of the most important parts of this work is the data evaluated. Power demand and water temperature has been analyzed from 15 of may to 30 of November of 2018. Other variables such as temperature, humidity and irradiation has been obtained from sensors installed in Barcelona. This variables will be also inputs of the machine-learning module to increase his accuracy. In some cases, it is useful to include this variable because the irradiation can reduce or increase the power consumption depending if it is used the heating or the cooling system.
A pre-processing step was applied over this data when preparing it to be employed by machine learning model. This process tried to detect outliers but only extreme values that are measurement errors. A value of the serie is an extreme measurement if is over or under some threeshold (53 - 54).

$$th_{up} = P_{95} * Iqr * k \tag{53}$$

$$th_{dwn} = P_5 * Iqr * k \tag{54}$$

Where P_{95} is the 95th percentile, P_5 is the 5th percentile, k is a usually used value for scalind the data and Iqr is the interquartile range, the difference between 75th and 25th percentiles (55).

$$IQR = Q_3 - Q_1 \tag{55}$$

In Fig. 1 and Fig. 2 both series are illustrated.

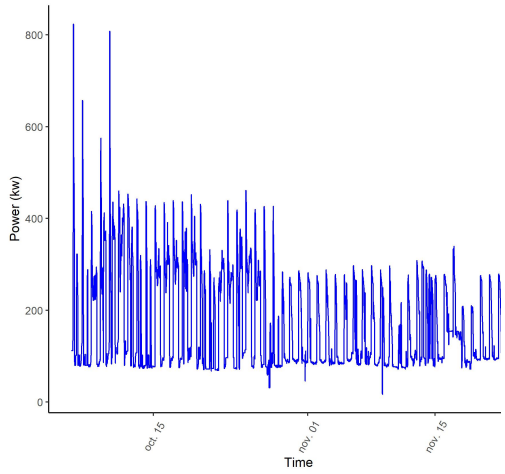

FIG. 1. Power demand serie

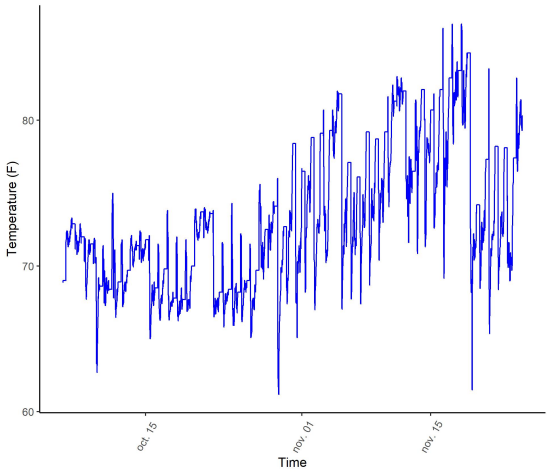

FIG. 2. Water temperature serie

2. Forecasting Results

2.1. Variable Selection. To obtain the most useful features a random forest is trained over the whole dataset and then the variables that best fit the model are selected to be used in the model implementation.

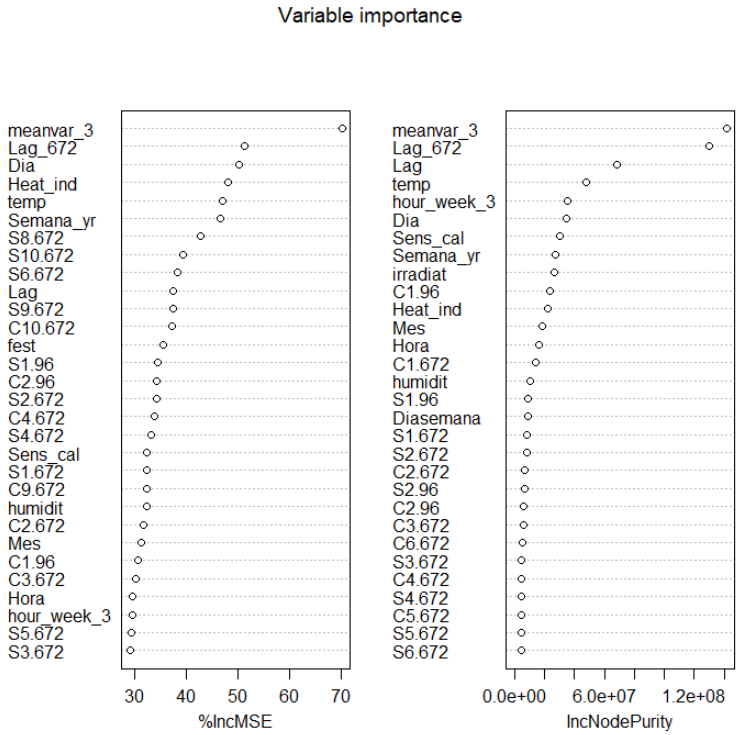

FIG. 3. Feature selection for power demmand

Analyzing this plot there are two metrics evaluated there, firs one is the percentage of increment of mean square error (%IncMSE) that measure the increment of error observed when permuting some levels or values of a variable, for example, if we permutate some values of moving average ($meanvar_3$) variable, the increment of error will be the 70% percent over the actual value. To obtain that value, the metric evaluated over the permutated values of each variable were MSE_{OOB}. The formula is:

$$IncMSE_j = \frac{(MSE_j - MSE_{OOB})}{MSE_{OOB}} * 100 \tag{56}$$

Chosen variables were the ones showed above. Variables such as Hour, Day of the year, Month and Temperature were chosen but also other variables including Fourier terms and lagged variables were chosen. Let's define some of them to give an idea of the most interesting features when forecasting power consumption.

- S_i, C_i, $i = 1, ..., 10$: Fourier Transform (Sinus and Cosinus).
- Si_{96}, Ci_{96}, $i = 1, ..., 10$: Fourier Transform with diary seasonality (Cosinus).
- Lag : Shifted Load serie
- Lag_{96} : Shifted Load serie with diary seasonality. Laged response for 24 hours or in this case 96 quarters of hour
- $mean_{var}$: Moving average with a 3 period wiindow.

2.2. Results for Bagged Trees. The idea of use tree-methods to predict electric and thermal consumption is obtained from [14] where time series forecasting using feature engineering methods like Fourier terms and the usage of series without trend are applied. These ideas combined with some extra feature engineering gives a powerful tool for forecasting time series in which daily seasonal patterns and trend behavior are observed.
The problem when applying this methodology is that,it is also necessary to predict the trend, using some linear method like an ARIMA. The reason is that tree methods, based on rules are not able to predict the trend. To do that, the aforementioned *STL* method was used. To predict the trend an ARIMA(0,2,0) is performed.
The bagged trees were generated using RPART and CTREE models. These techniques are evaluated over different sets of days. The results showed now faces the 28 and 29 of November of 2018 as weekly forecasts and 24 of November as weekend forecast. The best results were obtained by applying bagged models. The implemented models were a bagged version of CTREE and a bagged version of RPART. The use of this method improves the results obtained by simple tree models.

2.2.1. Power consumption forecasting. For weekly data, obtained results achieve a MAPE which is below the 10%. These results are good enough taking into account the shape of the series and how it changes during the year. Now, three metrics will be presented to evaluate both models. Fig. 4 - Fig. 7 show the prediccted versus the observed series for both days and both methods. This thesis focuses on MAPE as the relevant and most informative metric to evaluate time series forecasts.

Metric	*CTREE*	*RPART*
MAPE	4.51	6.25
RMSE	13.1	16.45
MAE	6.36	8.46

TABLE 1. Metric Tree weekday data (28-11-2018)

Metric	*CTREE*	*RPART*
MAPE	6.11	5.81
RMSE	11.5	12.18
MAE	7.03	7.02

TABLE 2. Metric Tree weekday data (29-11-2018)

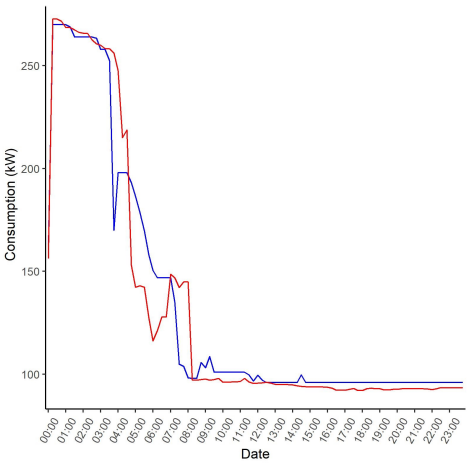

FIG. 4. Electric demmand 28-11-2018 using RPART

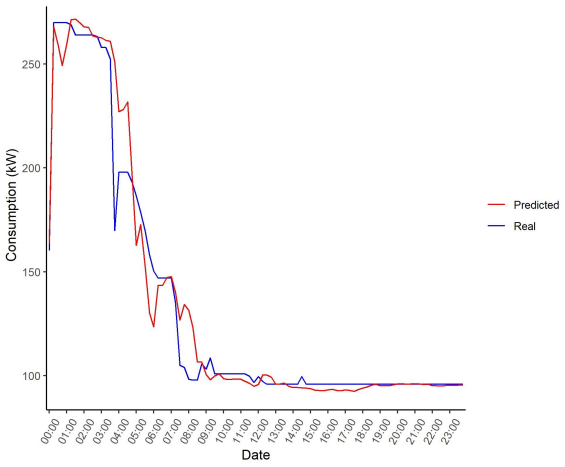

FIG. 5. Electricl demmand 28-11-2018 using CTREE

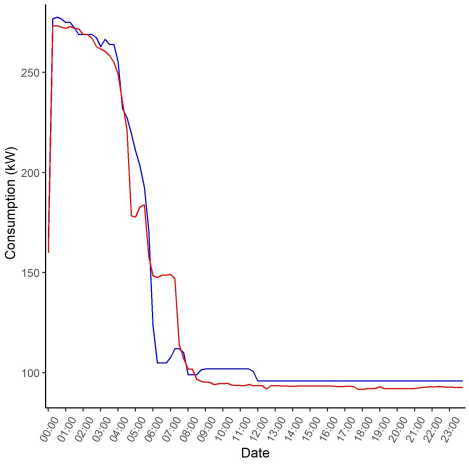

FIG. 6. Electricl demmand 29-11-2018 using RPART

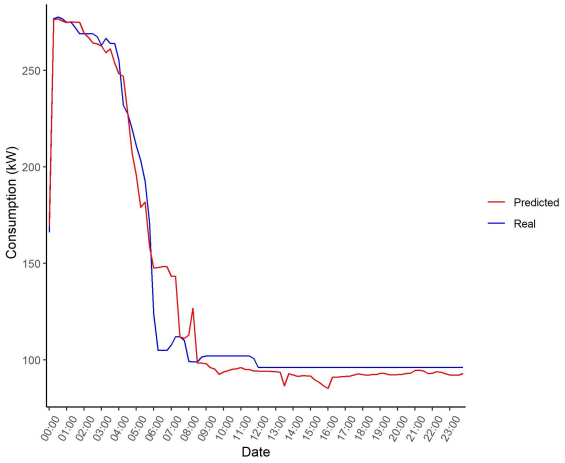

FIG. 7. Electric demmand 29-11-2018 using CTREE

The visualization shows that predicted values are following perfectly the evolution of the series.
For the weekends, the accuracy obtained with these models decreases drastically. The problem is focused on the randomness of this data. The model has to describe patterns that change from one weekend to another. This problem is also faced when comparing Saturdays and Sundays because while both days have a reduction in consumption and a charge of shape, given that one usually shows a consumption with three peaks and the other shows a consumption with only two peaks (-).

Metric	*CTREE*	*RPART*
MAPE	9.27	9.34
RMSE	17.4	20.1
MAE	10.2	10.7

TABLE 3. Metric Tree based weekend models (25-11-2018)

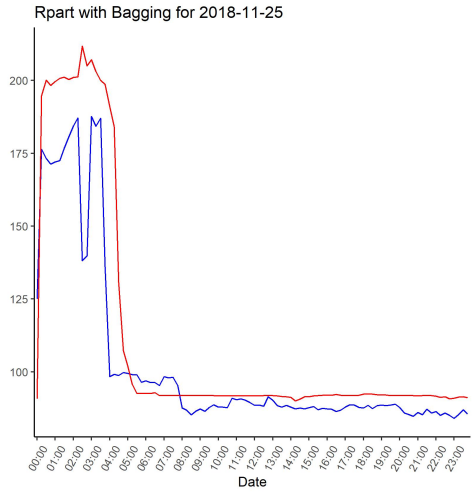

FIG. 8. Power demmand 25-11-2018 RPART

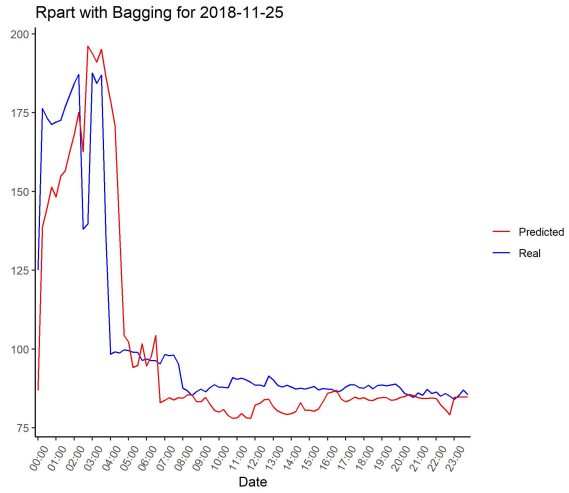

FIG. 9. Power demmand 25-11-2018 CTREE

2.2.2. Thermal demand forecasting. When evaluating the quality of predictions made for water temperature is easy to observe that the metric is very accurate. In this case, the variance is smaller than with power demand, so when analyzing the series it's important to study the value of the metric but also the visualization.

Metric	*CTREE*	*RPART*
MAPE	2.65	2.04
RMSE	2.62	1.73
MAE	1.88	1.44

TABLE 4. Metric Tree weekday data (28-11-2018) Temperature

Metric	*CTREE*	*RPART*
MAPE	1.62	2.1
RMSE	1.68	1.81
MAE	1.15	1.51

TABLE 5. Metric Tree weekday data (29-11-2018) Temperature

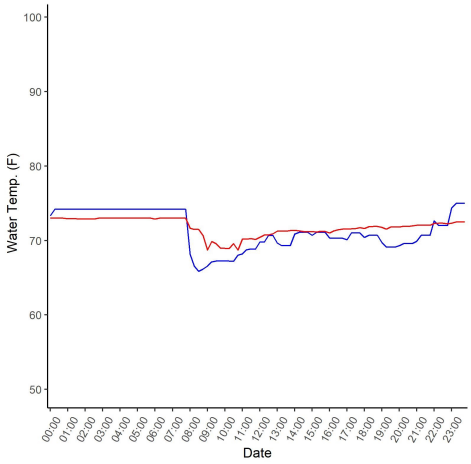

FIG. 10. Thermal demmand 28-11-2018 using RPART

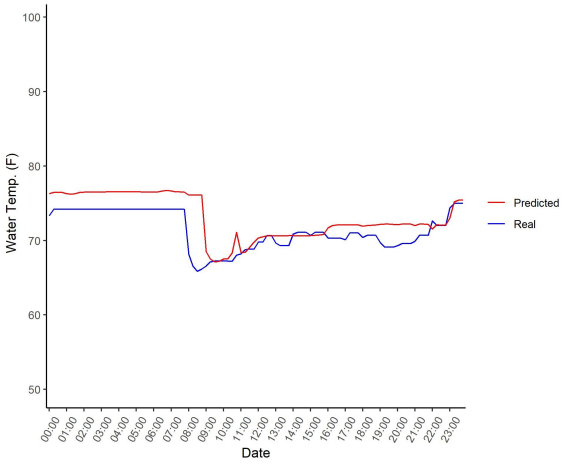

FIG. 11. Thermal demmand 28-11-2018 using CTREE

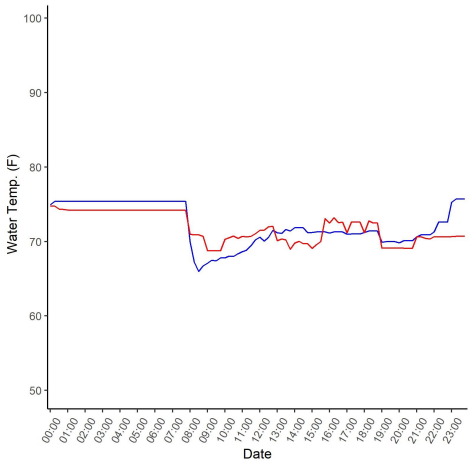

FIG. 12. Thermal demmand (29-11-2018) using RPART

FIG. 13. Thermal demmand (29-11-2018) using CTREE

While model metrics looks acceptable, the visualization shows that the model doesn't take at all the evolution of thermal consumption. This effect is due to the randomness existing inside the day. Each day have more or less a similar evolution (in terms of intercept), but with some specific patterns that change between days (Fig. 14 - 15). While the global pattern is captured by the model, the randomness not. Analyzing data an increasing tendency was detected in water temperature, then to take care of this underlying behavior, the series was divided in trend and seasonality. The tree algorithm only takes care of seasonality and the ARIMA takes care of the trend.

Metric	*CTREE*	*RPART*
MAPE	5.02	3.44
RMSE	4.47	3.21
MAE	4.18	2.88

TABLE 6. Metric Tree weekend models (25-11-2018) Temperature

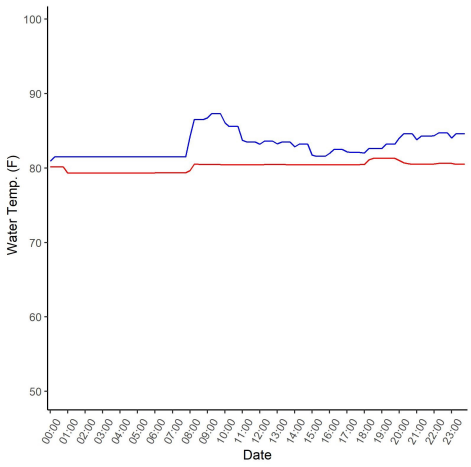

FIG. 14. Temperature, F 25-11-2018 using RPART

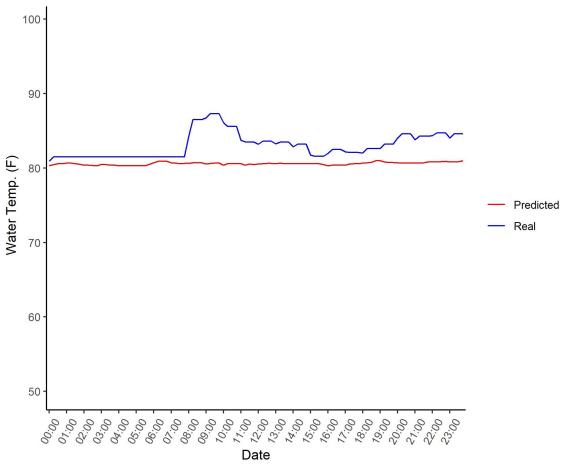

FIG. 15. Temperature, F 25-11-2018 using CTREE

Time Power D	*Time Thermal D*	*Model*
432s	400s	RPART
3600s	2940s	CTREE

TABLE 7. Computation Times Tree-based models

Analyzing these times of computation, the best approach will be the Bagged RPART methodology. Metric values are similar for thermal and electric demand comparison of models, but the necessary time to obtain the solution is ten times bigger when using CTREE methodologies. So, comparing tree-based models, the best approximation will be using an RPART Bagged model with 100 different trees and a sample size of 90% of the training data.

2.3. Results for LSTM's architectures. After describing the results obtained with tree-based methods lets analyze the results obtained using long short term memory structures. An interesting point of these methodologies are the capability of learning long term dependencies. The problem is that a big amount of data is required and this is sometimes difficult to achieve.
To start, let's define the different LSTM structures evaluated in this thesis.
Two kinds of LSTM models are tested, one approaches the problem generating matrices of n*1*m and then using an LSTM architecture to predict the next day comprised of 96 time periods. The second model generates a 3-dimensional matrix (days, time steps, variables) to apply then an LSTM architecture to forecast the

next 96 time periods.
In the training process, when using CPU's to train the models, the computational time has two behaviors. In models that frames the problem using a matrix of dimensions $n * 1 * m$ where n are the rows of the training set and m are the used features, the extra dimension is the time interval. When in models that frames the problem as a matrix of $m * 96 * n$ the time interval is set to 96 increasing model complexity and GPU requirements.
To test those methods, some combinations are evaluated for some LSTM parameters. Tested parameters were epochs, drops, and layers. In the training process, some days are also tested to obtain the best configuration.

- epochs: One forward pass and one backward pass of all the training examples
- drops: Percentage of data dropped when training the network. If more than one layer is evaluated, then drops could be applied in the first step of the training process, when features are set or when information is passed between layers.
- layers: Hidden states of the model.

A limitation of the problem is that when evaluating these models using GPU's, the required amount of RAM grows with the specifications. These specifications are the ones cited above as the number of epochs or the size of the batch. On the other hand, when a CPU architecture is used, the RAM limitations are not a problem but then other problem arises. This problem is related to the computation time needed to train the model.
When models are set to train an interesting effect is observed.
In models approached using just a time interval, the computational times are similar using a CPU or a GPU architecture.

$LSTM(n * 1 * m)$	CPU	GPU
Case 1	1000s	970s
Case 2	1400s	1150s
Case 3	1650s	1270s

TABLE 8. Computation Times LSTM (n*1*m)

The cases shown in the tables refer to the model evaluated under the same configuration of drops and epochs but using 1,2 and 3 layers respectively. By increasing the number of layers the computing time also increases. In models that frame the problem using a 3D matrix the computational time is much bigger in CPU's than in GPU's.

$LSTM(n * 96 * m)$	CPU	GPU
Case 1	2900s	1800s
Case 2	3300s	2600s
Case 3	4000s	3000s

TABLE 9. Computation Times LSTM (n*96*m)

Like with the model shown above, the cases shown in the tables were evaluated under the same configuration of drops and epochs but using 1,2 and 3 layers respectively. By increasing the number of layers the computing time also increases. At the end the idea developed was when the GPU specifications could be fulfilled, the model is trained using a graphical processor via CUDA architecture, which gives a fast implementation of the solution. When specifications couldn't be fulfilled the CPU architecture is used to train the model.

2.3.1. Power consumption forecasting. Like with tree-models, neural network architectures, obtained similar results, achieving a MAPE below 10%. Fig. 16 - 19 show the predicted versus the observed series for both days and both architectures.

$Metric$	$LSTM(n*1*m)$	$LSTM(n*96*m)$
MAPE	5.7	5
RMSE	14	12.8
MAE	4.7	4.3

TABLE 10. Metric LSTM weekdays (28-11-2018)

$Metric$	$LSTM(n*1*m)$	$LSTM(n*96*m)$
MAPE	7.4	6
RMSE	16.1	10
MAE	6.9	5.4

TABLE 11. Metric LSTM weekdays (29-11-2018)

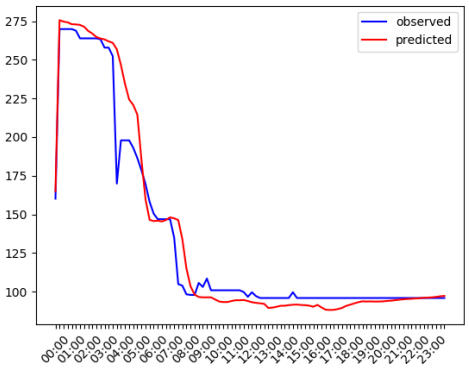

FIG. 16. Electric demmand 28-11-2018 (n*1*m) architecture

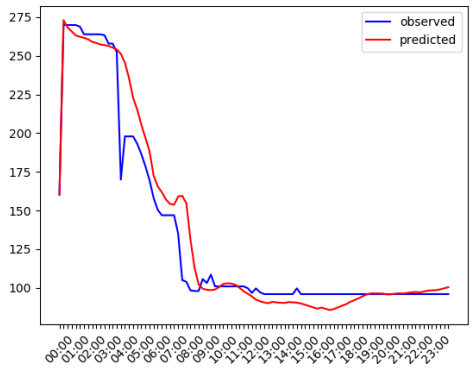

FIG. 17. Electricl demmand 28-11-2018 (n*96*m) architecture

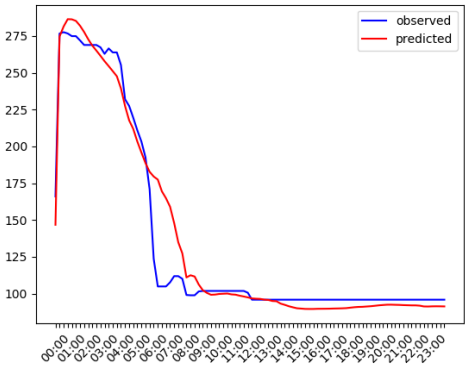

FIG. 18. Electricl demmand 29-11-2018 (n*1*m) architecture

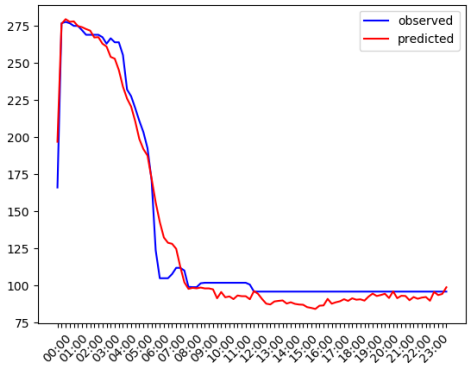

FIG. 19. Electric demmand 29-11-2018 (n*96*m) architecture

2.3.2. Thermal demand forecasting. With the use of neural networks, the value obtained for thermal demand was similar to that obtained with tree-based models. Both models are able to accurately predict the behavior of this series. Because the results obtained with both architectures are almost identical, only the one that has obtained a lower metric is shown (Fig. 20 - 21).

$Metric$	$LSTM(n*1*m)$	$LSTM(n*96*m)$
MAPE	2.1	2.09
RMSE	2.22	1.63
MAE	1.68	1.34

TABLE 12. Metric LSTM weekday (28-11-2018) Temperature

$Metric$	$LSTM(n*1*m)$	$LSTM(n*96*m)$
MAPE	1.52	1.57
RMSE	1.48	1.51
MAE	1.19	1.41

TABLE 13. Metric LSTM weekday (29-11-2018) Temperature

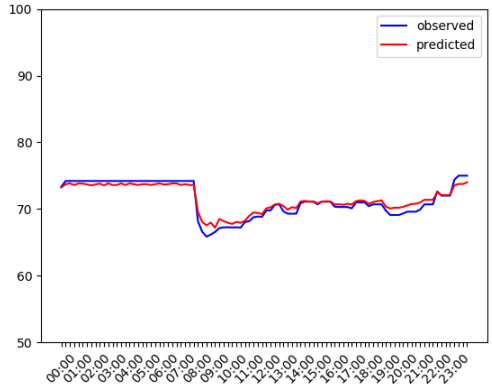

FIG. 20. Temperature (28-11-2018) (n*96*m) architecture

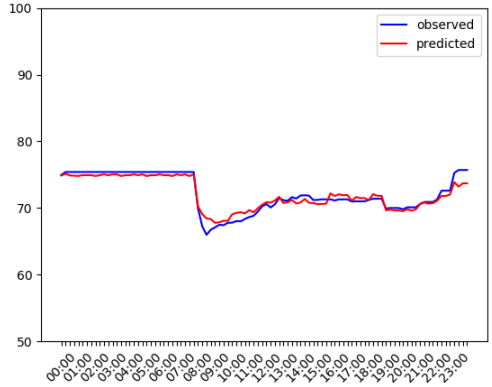

FIG. 21. Temperature (29-11-2018) (n*96*m) architecture

Table 14 includes computation times of methods based on a simple architecture integrated by timesteps and features, a more sophisticated architecture integrated by days, timesteps and features and the last approach that mixes convolutional layers and LSTM layers.

Type	$LSTM(n*1*m)$	$LSTM(n*96*m)$	CNN
Electric	240s	600s	630s
Thermal	230s	660s	695s

TABLE 14. Computation times RNN models

2.3.3. Optimized LSTM Model. After training the different architectures using some sets of parameters for epochs, batch size and percentage of drops, the best implementation was selected evaluating the **MAPE** metric. The resulted implementation includes a batch size of 32, 500 epochs and 30 % of drop applied in the first stage of model definition. With that values, the model is trained again

in a special way. The idea behind that is to train the model until first prediction need to be made, store it, predict the values for that day and then continue training the model until other prediction is needed. The interesting point of this approach is the capability of the software to store, predict and continue training the model. In models like that, which take a long time to obtain accurate predictions and need a big amount of data to be useful, the possibility of continue training the model from a previous state makes them attractive to set them in production areas.

Type	*Metric*	$LSTM(n*96*m)$
Electric	MAPE	5
Electric	RMSE	12.8
Thermal	MAPE	2.09
Thermal	RMSE	1.63

TABLE 15. Metric Optimized Architecture (28-11-2018)

Type	*Metric*	$LSTM(n*96*m)$
Electric	MAPE	30.5
Electric	RMSE	100.4
Thermal	MAPE	1.8
Thermal	RMSE	2.5

TABLE 16. Metric Optimized Architecture (27-07-2018)

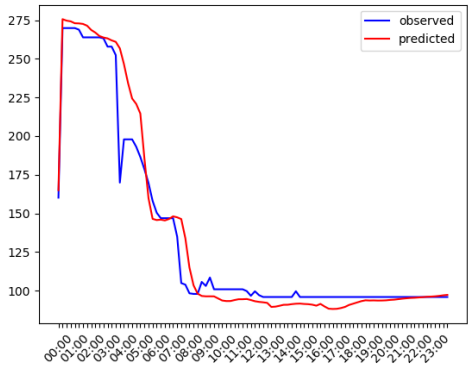

FIG. 22. Power demand 28-11-2018) Optimized Architecture

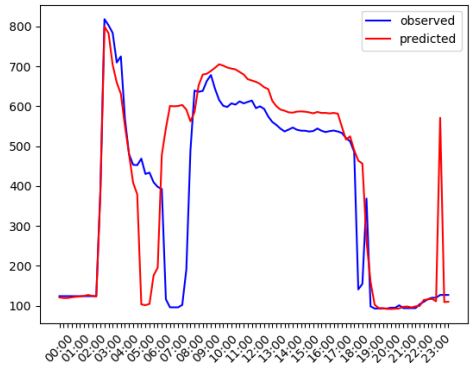

FIG. 23. Power demand (27-07-2018) Optimized Architecture

The shape of the distribution is very different between the two days evaluated in this model, both in form and magnitude. It is interesting to comment on how the model is able to capture the change that occurs in the series.

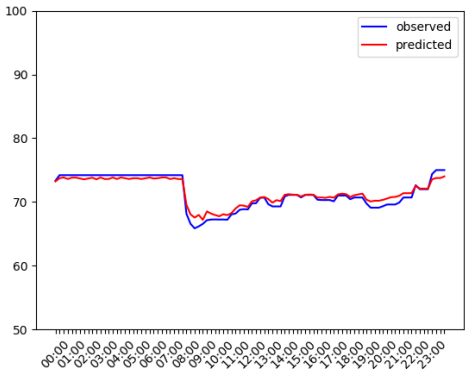

FIG. 24. Water Temperature (28-11-2018) Optimized Architecture

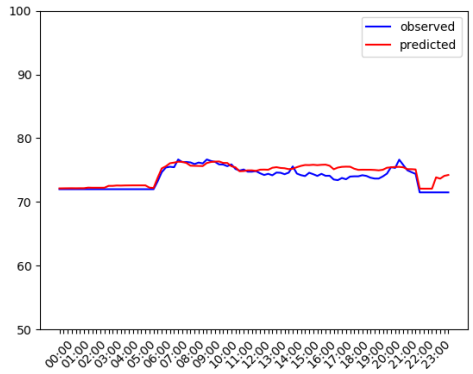

FIG. 25. Water Temperature (27-07-2018) Optimized Architecture

3. Model Results

The model was developed using a python interface to apply the SCIP optimization Suite. The model was implemented in a docker container integrated by the solver module and some manipulation and visualization modules to obtain a closed development system.
This section is divided in two parts, one that evaluates some test to detect if the plant works as expected and other that evaluate the model over a chosen day.
To evaluate the good procedure of the model, as the interest is to evaluate the integration of the EV and the energy storage system, a test where the electrical vehicle performance was evaluated fixing the value of the starting SOC to 1 and other parameters to appropiate values.

3.1. EV Test. The SOC of the battery is set to 1 or in other words at 100% of its capacity. A zero electrical demand is fixed for all periods. In this test, two comparisons are also evaluated, one setting a value of the energy cost smaller than the exportation costs and another fixing an equal value for the cost of purchase and export the energy. In addition, in this second case, the cost of the natural gas will be set higher than the cost of purchasing and selling the energy so that the model has no other incentive than trying to charge the electric vehicle with the battery.
When a very high sales value and a maximum initial SOC is set, everything that is generated from the CHP is exported, using the battery to cover the EV (the maximum charging/discharging power of the battery is 10 kW). By generating energy through the CHP, the model can also cover the thermal demand.
Then, the electricity price was fixed at a constant value for selling and purchasing actions and the cost of the natural gas was increased. The objective was to evaluate if the battery worked correctly charging the EV.

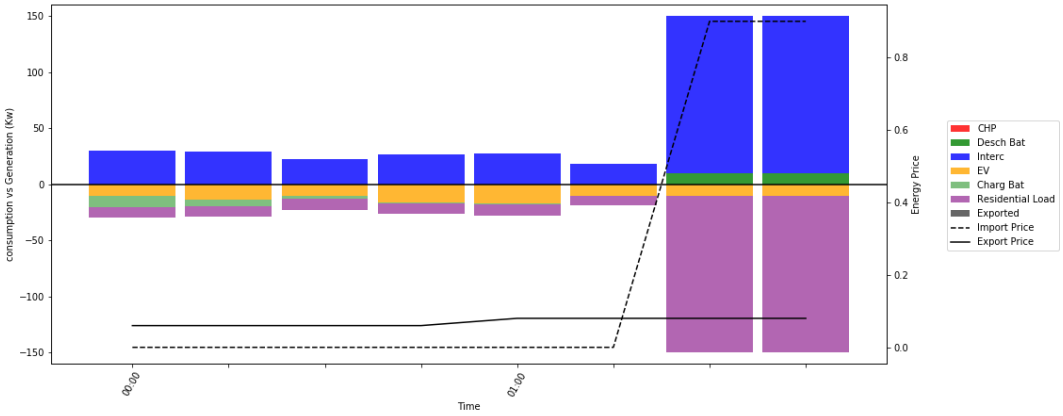

FIG. 26. EMS plot EV

3.2. Model Results. Finally, the model was evaluated over a day, integrated by 96 time periods. The chosen day was the 28th of November 2018. The input variables used were the forecast electrical demand, the forecast thermal demand, the energy prices for the day, extracted from [18] and the costs of the energy purchased from the grid. Other inputs were the costs for natural gas and the plant specifications, such as the maximum plant operation level and the maximum electrical power.

As it was explained before, the system could work using the energy provided by the cogeneration plant or taking it from the grid and if exists an excess of energy, the system would export the remaining. The model also allows the system to use the heating power of the cogeneration plant to heat water.

The main objective of the optimization problem is minimize the costs of supply the necessary energy to the whole system while trying to minimize some specifications,such as the number of ignitions of the plant by day.

Now, the main results of the analysis will be shown and also the value achieved by the objective function. The value achieved by the objective function was **174.37** €, which will be the costs of fulfilling the necessities of the system.

To understand better what the model is doing, Fig. 27 and Fig. 28 shows the evolution of the different elements of the system, such as the cogeneration plant (CHP), the building, the battery, the electric vehicle, and the grid.

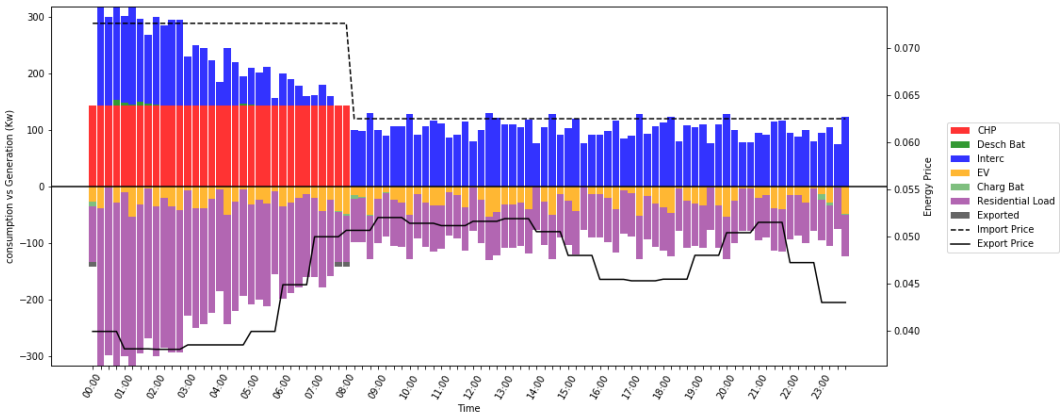

FIG. 27. EMS plot model

Fig. 27 describes on the positive values the energy generation, and on the negative values the demand. The dashed line correspond to the import prices, and the continous line to the export prices. It can be seen that when the demand is higher, during the first periods of the day, where also the prices are higher, the CHP starts-up to cover partially the demand. During some periods, where the electrical demand goes beyond the CHP generation, the excedent is sold. The EV demand is covered during all the studied intervals. At some specific intervals, green bars identify the battery discharge. In order to illustrate the battery process, since the impact on the global consumption is small due to the system dimension, Fig. 28 shows the battery charge and discharge processes.

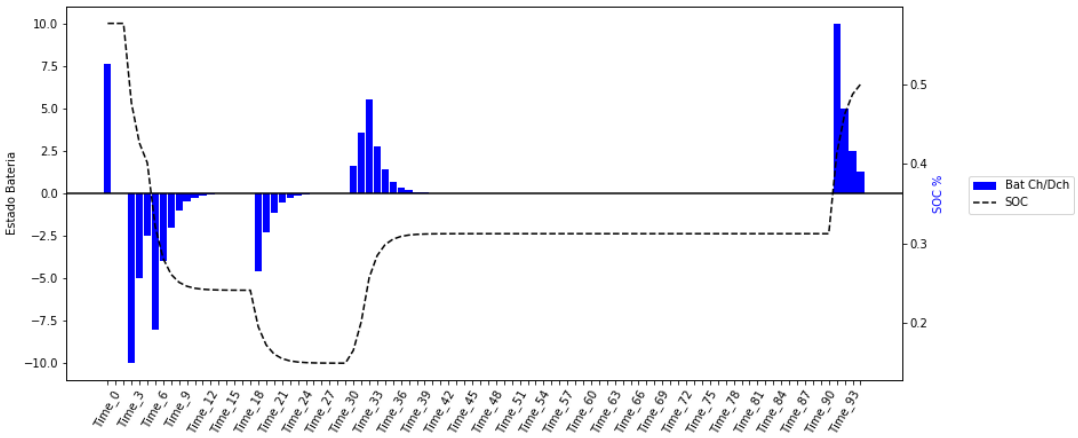

FIG. 28. EMS battery model

Chapter 5
Conclusions & Further Research

This work presents an energy management system componed by two forecasting modules and an optimization model to enhance the operation of a novel fast charging point system that incorporates a CHP plant. The incorporation of the economic aspects in the technical operation of the plant, allows optimizing the system in terms of economic benefits by optimizing the CHP operation, the imported and exported energy and the battery charging/discharging strategies. Simulation results suggest that additional benefits can also be achieved by connecting a CHP station to the electricity network on the location site. Moreover, results based on real operation data, provide evidence of the reduction of electricity costs for end consumers by including the optimal model into the system management. To conclude it's important to detail that in general, the energy management system, allows to a better response on higher peak load demand and fast response for the EV charging process by minimizing its cost.

The work has achieved the initial objectives by developing and implementing two forecast modules and an optimization model, and by integrating them into a real environment on testing phase.

From the results of the evaluated model, the principal ideas were:

- Energy is exported when the electric demand is lower than total cogeneration energy generation and the selling price is higher than the electricity cost. It's easy to see that when the system sells energy generated with the cogeneration plant, the selling price is higher but also the demand is lower than the generation.
- The system tries to charge the electric vehicle when there is an excess of energy and the gap between prices is not favorable (allowing the system to export the excess of energy). This could be an effect of the unserved EV demand cost, because depending on the value of this parameter and the cost of imported energy from the grid, one of them has a higher effect on the objective function.

From the forecasting methodologies, two different approaches were evaluated in this thesis, one using multiple binary partioning trees and the other applying LSTM architectures with stacked layers. Both tested models gave good predictions in terms of accuracy but, taking into account the small amount of data employed, tree-based models were the best ones, in terms of computation time and performance. RPART obtained a MAPE below 10% and only took seven minutes to be trained.
An interesting approach will be evaluated the performance of both models with more data available, because LSTMs are designed to be used with large volumes of data because of its capability of learn patterns, while the tree methods are only based on partioning the sample space by multiple variables sequentially, being his ability to learn from the data limited.
The main problem of the models applied in this thesis are the weekends. It would be necessary to evaluate a model fed with only weekends but more data will be needed to identify the underlying patterns.
Other problems were the **machine requirements**. When predicting LSTM's memory problems were faced. In this terms tree models also outperform the recurrent networks. When training neural networks in GPU system a good schedule of memory is needed to avoid lacks of resources.
It's also interesting to share that while deep neural networks look like a cool idea, other simpler approaches like decision trees with some ensembling method are sometimes more effective, fast and simpler.
Another interesting idea to analyze further is the feature selection methodology applied and his limitations. Random Forests and decision trees, in general, based on impurity reduction are biased towards preferring variables with more categories (high cardinality). Trees are biased to these type of variables. That's because is easier to find a value in which the model could be splitted when the number of *levels* are bigger.
Correlated features will show similar and lowered importance, compared to what their importance would be if the tree was built without correlated counterparts. Any of these correlated features can be used as the predictor, with no concrete preference of one over the others. This is not an issue if feature selection is applied to reduce overfitting, since it makes sense to remove features that are mostly duplicated. But when interpreting the data, it can lead to the incorrect conclusion that one variable is a strong predictor while the others not (related to the response variable).
The next steps are to apply the model in a production environment, providing the optimization model with the necessary inputs of electrical and thermal demand. Another interesting idea would be to apply a training methodology for recurrent neural networks, based on a process of training, predicting and saving the current state to be able to continue training the model. With this, you could have pre-trained models that would reduce the training time of the algorithms. A first approximation of this methodology has been implemented to obtain the best model based on neural networks. This approach allows predicting multiple days saving in each stage the state of the model to then continue to train and get the following prediction.

References

[1] Jason Brownlee. *Machine Learning Mastery*. https://machinelearningmastery.com/. URL: https://machinelearningmastery.com/.

[2] Jason Brownlee. *Machine Learning Mastery with Weka*. https://machinelearningmastery.com. 2019. URL: https://machinelearningmastery.com/how-to-develop-lstm-models-for-multi-step-time-series-forecasting-of-household-power-consumption/ (visited on 03/23/2019).

[3] Razvan-Gabriel Cirstea et al. "Correlated Time Series Forecasting using Deep Neural Networks: A Summary of Results". In: (2018). arXiv: 1808.09794. URL: http://arxiv.org/abs/1808.09794.

[4] M. Cruz-Zambrano et al. "Optimal location of fast charging stations in Barcelona: A flow-capturing approach". In: *International Conference on the European Energy Market, EEM* (2013), pp. 1–6. ISSN: 21654077. DOI: 10.1109/EEM.2013.6607414.

[5] Niklas Donges. *No Title*. https://towardsdatascience.com. 2018. URL: https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5 (visited on 04/23/2019).

[6] Georgios Drakos. *No Title*. https://towardsdatascience.com. 2019. URL: https://towardsdatascience.com/how-to-select-the-right-evaluation-metric-for-machine-learning-models-part-1-regrression-metrics-3606e25beae0 (visited on 04/24/2019).

[7] Rumelhart D E., Hinton G E, and and Williams R. J. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536.

[8] Robin Genuer, Jean-michel Poggi, and Christine Tuleau-malot. "Variable selection using Random Forests To cite this version :" in: *Pattern Recognition Letters* 31.14 (2012), pp. 2225–2236.

[9] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 06.02 (2003), pp. 107–116. ISSN: 0218-4885. DOI: 10.1142/s0218488598000094.

[10] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. "ctree: Conditional Inference Trees". In: *The Comprehensive R Archive Network* Quinlan 1993 (2015), pp. 1–34. URL: https://rdrr.io/rforge/partykit/f/inst/doc/ctree.pdf.

[11] Rob Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. 2nd Editio. URL: https://otexts.com/fpp2/.

[12] Aastha Kapoor and Ankush Sharma. "A Comparison of Short-Term Load Forecasting Techniques". In: *International Conference on Innovative Smart Grid Technologies, ISGT Asia 2018* (2018), pp. 1189–1194. DOI: 10.1109/ISGT-Asia.2018.8467788.

[13] Hideko Kawakubo and Hiroaki Yoshida. "Rapid Feature Selection Based on Random Forests for High-Dimensional Data". In: *Information Processing Society of Japan (IPSJ)* 3 (2012), pp. 1–7. URL: https://pdfs.semanticscholar.org/c38a/45ab41f81771ad5390a134857694dfe5e527.pdf{\%}0Ahttp://ci.nii.ac.jp/naid/110009421219/.

[14] Peter Laurinec. *No Title*. https://petolau.github.io/. URL: https://petolau.github.io/.

[15] Peter Laurinec. *No Title*. https://petolau.github.io/. URL: https://petolau.github.io/Ensemble-of-trees-for-forecasting-time-series/ (visited on 12/10/2018).

[16] Zifa Liu et al. "Energy storage capacity optimization for autonomy microgrid considering CHP and EV scheduling". In: *Applied Energy* 210 (2018), pp. 1113–1125. ISSN: 03062619. DOI: 10.1016/j.apenergy.2017.07.002. URL: https://doi.org/10.1016/j.apenergy.2017.07.002.

[17] Christopher Olah. *Convolutional Neural Networks Posts*. http://colah.github.io. URL: http://colah.github.io/posts/tags/convolutional{_}neural{_}networks.html.

[18] Omie. *omie*. http://www.omie.es/inicio. URL: http://www.omie.es/inicio.

[19] Suryansh S. *No Title*. https://hackernoon.com. 2018. URL: https://hackernoon.com/gradient-descent-aynk-7cbe95a778da (visited on 04/25/2019).

[20] Alper Tokgoz and Gozde Unal. "A RNN based time series approach for forecasting turkish electricity load". In: *26th IEEE Signal Processing and Communications Applications Conference, SIU 2018* (2018), pp. 1–4. DOI: 10.1109/SIU.2018.8404313.

[21] Koehrsen Will. *No Title*. https://medium.com. 2017. URL: https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d (visited on 04/24/2019).

Acronyms

ANN:	Artificial Neural Network
ARIMA:	Auto Regressive Integrated Moving Average
CHP:	Combined Heat and Power
CNN:	Convolutional Neural Network
CPU:	Central Processing Unit
CTREE:	Conditional Inference Trees
EV:	Electric vehicle
EVSE:	Electric vehicle supply equipment
EMS:	Energy Management System
GPU:	Graphics Processing Unit
LSTM:	Long Short Term Memory
MSE:	Mean Square Error
MAE:	Mean Absolute Error
MAPE:	Mean Absolute Percentage Error
P_5:	5th percentile
P_{95}:	95th percentile
PLC:	Programmable Logic Controller
RNN:	Recurrent Neural Network
RPART :	Recursive Partioning