Neurocomputing 423 (2021) 747–755

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Ensemble learning for electricity consumption forecasting in office buildings

Tiago Pinto [a,*], Isabel Praça [a], Zita Vale [b], Jose Silva [b]

[a] GECAD Research Group, Instituto Superior de Engenharia do Porto, R. Dr. António Bernardino de Almeida, 431, P-4249-015 Porto, Portugal
[b] Instituto Superior de Engenharia do Porto, Instituto Politécnico do Porto, R. Dr. António Bernardino de Almeida, 431, P-4249-015 Porto, Portugal

ARTICLE INFO

Article history:
Received 6 November 2019
Revised 28 January 2020
Accepted 6 February 2020
Available online 8 May 2020

Keywords:
Energy consumption
Ensemble learning
Machine learning
Load forecasting

ABSTRACT

This paper presents three ensemble learning models for short term load forecasting. Machine learning has evolved quickly in recent years, leading to novel and advanced models that are improving the forecasting results in multiple fields. However, in highly dynamic fields such as power and energy systems, dealing with the fast acquisition of large amounts of data from multiple data sources and taking advantage from the correlation between the multiple available variables is a challenging task, for which current models are not prepared. Ensemble learning is bringing promising results in this sense, as, by combining the results and use of multiple learners, is able to find new ways for current learning models to be used and optimized. In this paper three ensemble learning models are developed and the respective results compared: gradient boosted regression trees, random forests and an adaptation of Adaboost. Results for electricity consumption forecasting in hour-ahead are presented using a case-study based on real data from an office building. Results show that the adapted Adaboost model outperforms the reference models for hour-ahead load forecasting.

1. Introduction

Adequate models for forecasting energy consumption is crucial for all players in the energy sector. Reliable forecasts are important to enable the system to maintain the balance between consumption and generation at all times; they are essential for market players to be able to participate in market negotiations using proper demand estimates; and they are an important part of the development of suitable demand response programs [1]. In this way, electric power load forecasting is a core problem in power and energy systems. Electric power load forecasting can be categorized according to the planning horizon duration: 1 day for short-term load forecasting. 1 day to 1 year for medium-term load forecasting, and 1–10 years for long-term load forecasting [2]. Recently, many works have emerged to achieve accurate forecasting models. The two most common paths are: the conventional statistical approach such as multiple linear regression, which e.g. in [3] has reached 3.99% Mean Absolute Percentage Error (MAPE) on hourly consumption forecast. Statistical models yet frequently present low precision levels, mostly due to their simplicity and consequent unsuitability to fully model the dynamics of energy consumption. Artificial intelligence techniques are emerging with promising results, due to the ability to identify and make use of non-linear relationships between variables. Artificial neural networks [4], Support Vector Machines (SVM) [5], Random Forest [6] and Stochastic Gradient Boosting [7] are popular artificial intelligence models for short-term load forecasting.

Despite the recent advances in the field of short-term load forecasting using artificial intelligence models, there are still relevant gaps that need to be overcome. In specific, the ability of forecasting models to deal with the large number of different types of data, identifying and taking advantage on their correlation, and providing high quality results in fast response times, are issues that have not yet been solved. The research, development and application of traditional forecasting algorithms by their own is proving to be insufficient to overcome the above mentioned problems; thus, the combination and learning on how to use and combine different learning models is a promising path towards reaching suitable solutions. Ensemble learning is a technique that combines a set

Abbreviations: AR2, Adaboost; GBR, Gradient Boosted Regression Trees; HyFIS, Hybrid Neural Fuzzy Inference System; MAPE, Mean Absolute Percentage Error; RF, Random Forests; SVM, Support Vector Machines; WM, Wang and Mendel's Fuzzy Rule Learning Method.

* Correspondence author.

E-mail addresses: tcp@isep.ipp.pt (T. Pinto), icp@isep.ipp.pt (I. Praça), zav@isep.ipp.pt (Z. Vale).

https://doi.org/10.1016/j.neucom.2020.02.124
0925-2312/© 2020 Elsevier B.V. All rights reserved.

of independent learners together to improve the predictive power of the model [8]. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced [9]. Some reference methods in in this domain are Random Forests (RF) and Gradient Boosted Regression Trees (GBR) and Adaboost (AR2). In [10] RF achieved a 1.97% MAPE on one-day ahead electricity load forecasting for the ISO-NE CA area. RF also proved efficient in [11] achieving a MAPE of 2.75%. Throughout a vast number of papers random forests proves to be a good strategy for short-term load forecasting. GBR was used in [10] achieving an MAPE of 1.32% on one-day ahead electricity load forecasting for the ISO-NE CA area, outperforming RF in that same study. In [12] GBR it is used to predict vehicle time travel achieving a MAPE of 11%. GBR consistently keeps up with other state of the art models, even having better results in some studies. In [13] was concluded that all AR2 variants outperformed their single (base) algorithms including multilayer perceptron, general linear model, support vector regression, and decision tree for regression.

This work develops and assesses the results of three ensemble learning methodologies applied to the problem of short-term load forecasting. The considered models are RF, GBR and AR2, which are adapted in order to suit the particularities of the problem of electricity demand forecasting of office buildings. A software application is presented, which supports the processes of defining the parameters and adapting the methodologies to the envisaged problem. Results are compared to those achieved in previous studies, using different techniques, namely three fuzzy based systems: an Hybrid Neural Fuzzy Inference System (HyFIS) [14], the Wang and Mendel's Fuzzy Rule Learning Method (WM) [15]; and SVM [16]. The case study considers a data set gathered from real consumption of a campus building of ISEP/GECAD. Besides the application of the proposed models, the case study assesses the influence of diverse parameters with potential influence on energy consumption forecasting, such as temperature, humidity and luminosity; and also analyses the data range that should be used – if it is more beneficial for the results to use only recent data or how far back in the past should the forecasting models go.

After this introduction, Section 2 presents the Material and Methods of this work, including the formulation of the methods, explanation of the proposed approach and presentation of the computational tool. Section 3 presents the case study based on real data from an office building, and compares the achieved results to those obtained by other reference methods. Finally, Section 4 presents the most relevant conclusions of the work.

2. Material and methods

Three ensemble-learning models for energy demand forecasting are presented. The load demand from an office building of the campus of GECAD/ISEP in Porto, Portugal is considered. The forecasting models have been implemented in Python according to the models presented as follows.

Ensemble models blend the forecasts of multiple predictors as means to increase generalization and robustness. Ensemble models are usually categorized as: (i) averaging methods, which consider multiple independent predictors to average their forecasts; and (ii) boosting methods, which combine multiple low performing methodologies to reach a powerful ensemble. Examples of (i) are the bagging methods and RF while (ii) considers models such as Adaboost and GBR.

In order to conduct the short-term load forecasting process, weather and load history are taken as inputs to create a load-forecasting model. These models are the algorithms described in the following sub-sections. After the model is trained, it is then extrapolated with weather forecast data to generate a final forecast. The modeling process in Fig. 1 tends to capture the systematic variation, which, as an input to the extrapolating process, is crucial to the forecast accuracy.

2.1. Ensemble forecasting methods

2.1.1. Random Forests (RF)

The Random Forest (RF), first introduced by Bell lab researcher Tin Kam Ho [10] in 1995 (Random Decision Forests) is an ensemble learning method for classification and regression. RF algorithm uses the bagging technique for building an ensemble of decision trees. RF algorithm uses the bagging technique for building an ensemble of decision trees. It develops many decision trees based on random selection of data and random selection of variables. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, resulting an overall better model [17]. Pseudocode for this algorithm is illustrated Fig. 2.

The algorithm works as follows: for each tree in the forest, we select a bootstrap sample from S where *Si* denotes the *i*th bootstrap. We then learn a decision-tree. The algorithm at each node of tree selects a subset of features $f << F$ where$F$ is the set of features. The node then splits on the best feature in f.

2.1.2. Gradient Boosted Regression Trees (GBR)

GBR [19] considers low performance methods, usually decision trees, to develop a prediction model based on their ensemble. GBR puts the model together sequentially and reaches generalization through the optimization of a differentiable loss function. Fig. 3 illustrates the pseudocode of the original Gradient Boosting algorithm.

GBR considers additive models of the form (1):

$$F(x) = \sum_{m=1}^{M} \gamma mhm(x) \tag{1}$$

where *hm(x)* are the principle functions, which are called weak learners in the context of boosting and γm the step length that is chosen using line search (2):

$$\gamma m = argmin_y \sum_{i=1}^{n} L\left(yi, F_{m-1}(xi) - y \frac{\theta L(yi, Fm - 1(xi))}{\theta Fm - 1(xi)} \right) \tag{2}$$

Similarly to other boosting algorithms GBR builds the additive model in a forward stage wise approach (3):

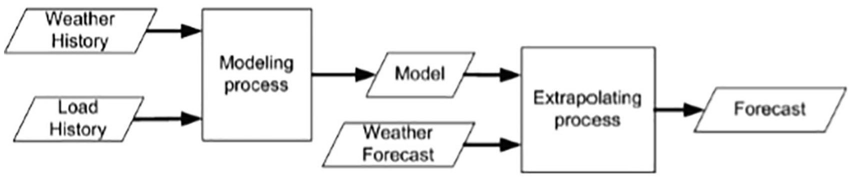


Fig. 1. Overview of the short-term load forecasting process.

```
Precondition: A training set $S := (x_1, y_1), \dots, (x_n, y_n)$, features $F$, and number
    of trees in forest $B$.
function RandomForest($S$, $F$)
    $H \leftarrow \emptyset$
    for $i \in 1, \dots, B$ do
        $S^{(i)} \leftarrow$ A bootstrap sample from $S$
        $h_i \leftarrow$ RandomizedTreeLearn($S^{(i)}$, $F$)
        $H \leftarrow H \cup \{h_i\}$
    end for
    return $H$
end function
function RandomizedTreeLearn($S$, $F$)
    At each node:
        $f \leftarrow$ very small subset of $F$
        Split on best feature in $f$
    return The learned tree
end function
```

Fig. 2. Random Forests Pseudocode [18].

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

Fig. 3. Gradient Boosting Original Algorithm Pseudocode [20].

$$Fm(x) = Fm - 1(x) + \gamma mhm(x) \tag{3}$$

At each stage the decision tree *hm(x)* is chosen to minimize the loss function *L* given the current model *Fm − 1* and its fit *Fm − 1 (xi)*, as in (4).

$$Fm(x) = Fm - 1(x) + argmin_h \sum_{i=1}^{n} L(yi, Fm - 1(xi) - h(x)) \tag{4}$$

The basic principle for solving this minimization problem is to use the steepest descent, which is the negative gradient of the loss function evaluated at the current model *Fm − 1*. This can be estimated as in (5):

$$Fm(x) = Fm - 1(x) + \gamma m \sum_{i=1}^{n} \nabla_F L(yi, Fm - 1(xi)) \tag{5}$$

The GBR model developed in this work is implemented using Python's scikit-learn package. 1400 boosting stages are considered. A large number of stages is considered because, since gradient boosting is traditionally rather robust to over-fitting, a high number of stages usually performs well. Scikit-learn supports many different loss function. We experienced all of them and found that the least absolute deviations function works better for our model.

The maximum depth of independent regression predictors is also tuned. This depth limits the number of nodes of the tree. The maximum depth that has reached the best outcomes considers a size equal to 10. Additionally, the minimum number of samples required to split an internal node is set to 2 and the learning rate is defined as 0.2. The learning rate (v) defines the scale of the gradient decent procedure.

$$Fm(x) = Fm - 1(x) + v\gamma mhm(x) \tag{6}$$

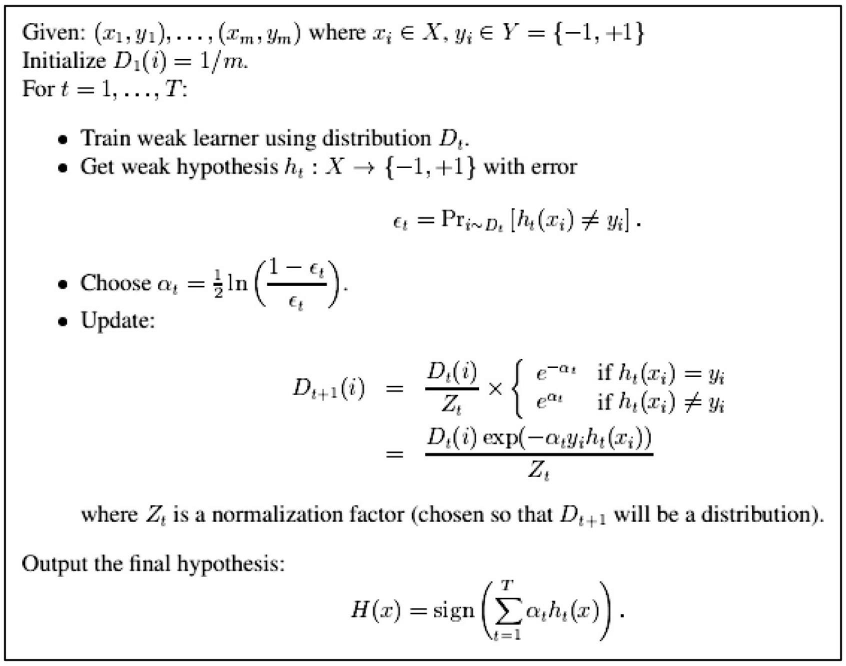
This regularization approach has been introduced in [13] and scales the contribution of each weak learner by v.

2.1.3. AdaBoost

Adaboost is a popular ML algorithm invented by Yoav Freund and Robert Schapire [17] in 1995, originally to solve classification problems. The core principle of AdaBoost is to fit a sequence of weak learners on repeatedly modified versions of the data. The data modification at each iteration consists of applying a weight to each classifier. In the first step of the algorithm, all weights are equally distributed. At each iteration, the weights are updated, where the weight is increased on those classifiers who misclassified the data and decreased on those who correctly classified the data. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. Pseudocode for the original Adaboost algorithm is given in Fig. 4. The algorithm takes as input a training set $(x1, y1) \cdots (xm, ym)$ where each xibelongs o some domain or instance space, and each label yi is in some label set Y. Adaboost calls a given weak or base learning algorithms repeatedly in a series of round t = 1, . . ., T. One of the main ideas of the algorithm is to maintain a distribution or set of weights over the training set. The weight of this distribution on training example I on round t is denoted $Dt(i)$. Initially, all the weight are the same equally, but on each round the weights of misclassified examples are increased so that the weak learner is forced to focus on the hard examples in the training set.

The weak learner's job is to find a weak hypothesis $ht : X- > \{-1, +1\}$ appropriate for the distribution Dt. The goodness of a weak hypothesis is measured by its error(*et*). The final hypothesis H is a weighted majority vote of the T weak hypotheses where αi is the weight assigned to ht. Fig. 4 presents the pseudocode of the original Adaboost algorithm.

Another used ensemble estimator is AdaBoost.R2 [21] which is a modified regression version of the famous AdaBoost ensemble

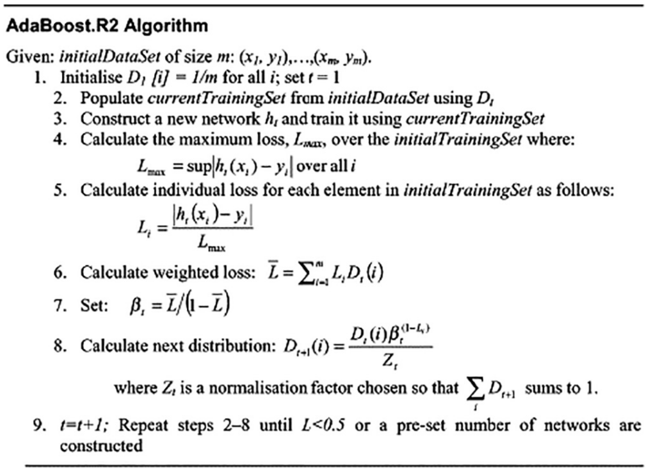


```
Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:
    • Train weak learner using distribution $D_t$.
    • Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error
          $\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$.
    • Choose $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
    • Update:
          $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
      where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution).
Output the final hypothesis:
    $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.
```

Fig. 4. Original Adaboost Algorithm [18].

estimator [17]. It sequentially fits estimators and each subsequent estimator concentrates on the samples that were predicted with higher loss. Fig. 5 illustrates the pseudocode of the Adaboost.R2 using artificial neural networks as base estimator.

The key architectural features of the ADABoost.R2 algorithm are an initial dataset and a sampling distribution. Each of the training data elements has a value in the sampling distribution and this value represents the probability of that element being included in the next training set. Initially, each value in the distribution is set to the same value giving each element in the initial dataset an equal chance of being included in the first training set. An initial training set is populated from this distribution and the first artificial neural network is trained. Each member of the initial dataset is then presented in turn to the network and the prediction errors are recorded. Using these prediction errors, the distributions for each element of the initial training set are adjusted using the formula presented in the algorithm. Once, the distributions have been adjusted, a new training set can be populated. This set is then used to train the next network, and the steps described above are repeated. New networks are iteratively trained until either the average prediction error for a network goes above 0.5 or a maximum number of networks have been trained.



```
AdaBoost.R2 Algorithm
Given: initialDataSet of size m: $(x_1, y_1), \ldots, (x_m, y_m)$.
1. Initialise $D_1[i] = 1/m$ for all $i$; set $t = 1$
    2. Populate currentTrainingSet from initialDataSet using $D_t$
    3. Construct a new network $h_t$ and train it using currentTrainingSet
    4. Calculate the maximum loss, $L_{max}$, over the initialTrainingSet where:
           $L_{max} = \sup|h_t(x_i) - y_i|$ over all $i$
    5. Calculate individual loss for each element in initialTrainingSet as follows:
           $L_i = \frac{|h_t(x_i) - y_i|}{L_{max}}$
    6. Calculate weighted loss: $\bar{L} = \sum_{i=1}^{m} L_i D_t(i)$
    7. Set: $\beta_t = \bar{L}/(1-\bar{L})$
    8. Calculate next distribution: $D_{t+1}(i) = \frac{D_t(i)\beta_t^{(1-L_i)}}{Z_t}$
           where $Z_t$ is a normalisation factor chosen so that $\sum_i D_{t+1}$ sums to 1.
9. $t=t+1$; Repeat steps 2–8 until $\bar{L}<0.5$ or a pre-set number of networks are constructed
```

Fig. 5. AR2 Algorithm [23].

The used algorithm, implemented in [22] slightly differs from [21] as it allows to use the weights directly in the fitted estimator and not only for weighted sampling of features, as follows:

1. start algorithm t = 0
2. To each training sample assign initial weight (7)

$$w_i^t := 1, i = 1, 2, \ldots, m \tag{7}$$

3. fit estimator t to the weighted training set with weights w_i^t
4. compute prediction $\hat{y}_i^t$ using the estimator t for each sample i
5. compute loss li for each training sample (8)

$$l_i^t = loss\left(|\hat{y}_i^t - yi|\right) \tag{8}$$

6. calculate average loss $\bar{l}^t$
7. calculate confidence β^t for the estimator (low β^t means high confidence in estimator t)
8. update weights of training samples (9)

$$w_i^{t+1} = w_i^t \cdot (\beta^t)^{(1-l_i^t)}, i = 1, 2, \ldots, m \tag{9}$$

9. $t = t + 1$ continue to step 3 while the average loss $\bar{l}^t < 0.5$

To forecast our target values, we built an AR2 model with 1400 boosting stages using Python's scikit-learn package and the learning rate has been set to 0.01. We experienced that square loss function works better for our model [21] (10):

$$l_i^t = \frac{|\hat{y}_i^t - yi|^2}{D^2} \tag{10}$$

where D is defined as (11):

$$D = sup\left\{\left|\hat{y}_i^t - yi\right|, i \in \{1, 2, \cdots, m\}\right\} \tag{11}$$

2.2. Short-term load forecasting computational tool

This section includes the details of the architecture of the developed application. The application is divided into two different components. These components are:

1. Forecasting app: This part includes the implementation of the forecasting algorithms and it is based on Python.
2. Main app: This part has the controllers of application, the connection with the other components and the graphical user interface.

The main app is the key component of the application, which connects the other component. All the communications between the user and the application are a part of the main app. Once the user selects the required data, the forecasting application prepares that data to run the forecasting methods. The forecasting app run the forecasting methods by the received data from the main app and returns the results of the forecasting. The main app shows the results to the user.

The forecasting menu is where the inputs are defined or introduced. The user should select the excel file, the sheets and the features he wants in the training data. Fig. 6 represents the forecasting menu of the application.

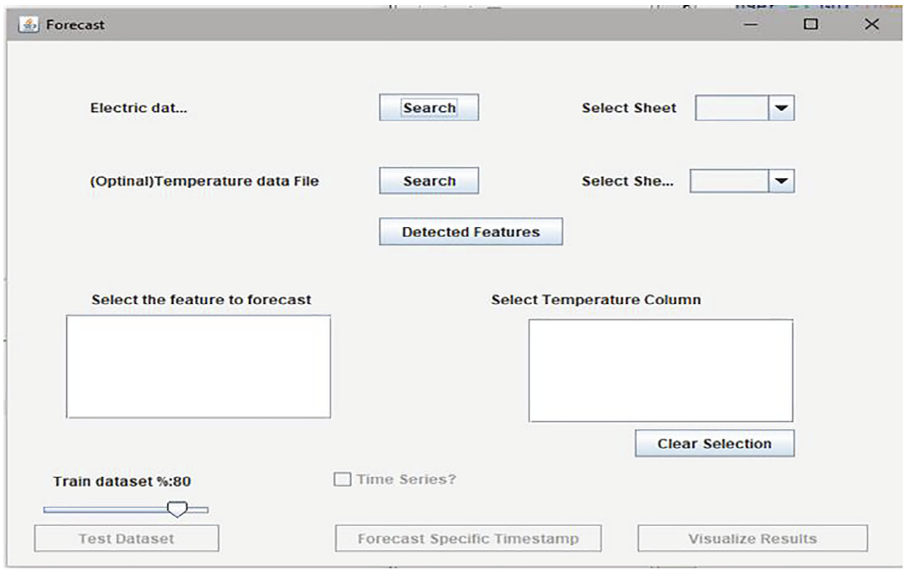


Fig. 6. Forecasting menu.

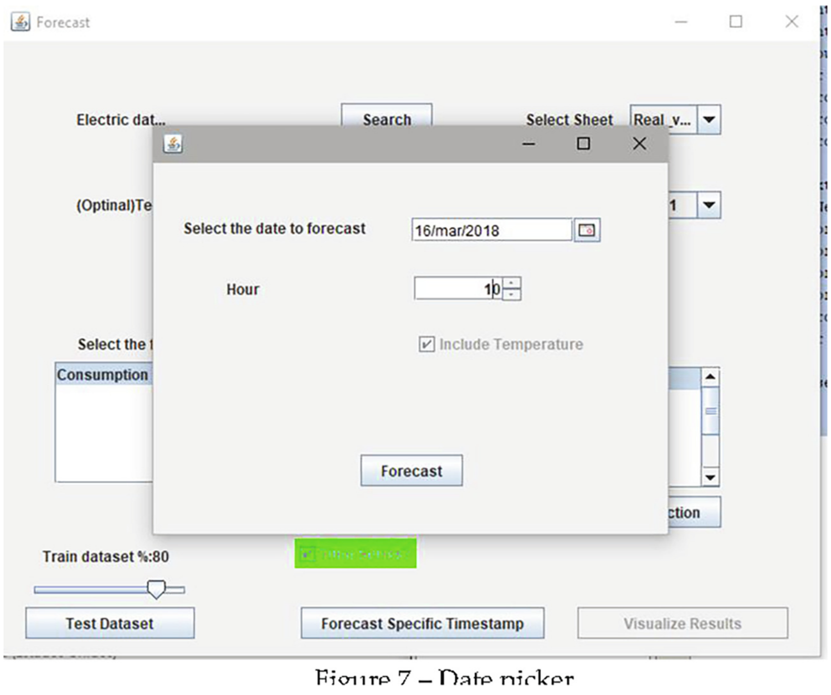
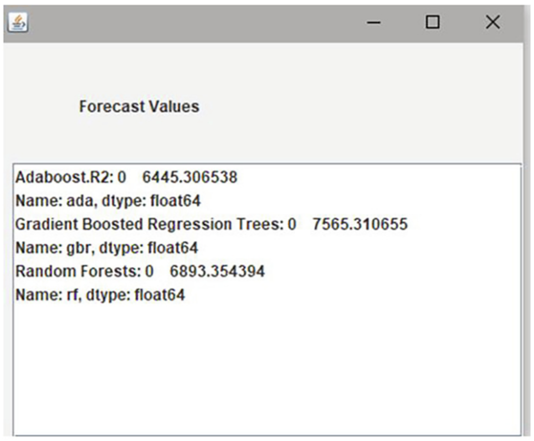


Fig. 7. Date picker.



```
Adaboost.R2: 0    6445.306538
Name: ada, dtype: float64
Gradient Boosted Regression Trees: 0    7565.310655
Name: gbr, dtype: float64
Random Forests: 0    6893.354394
Name: rf, dtype: float64
```

Fig. 8. Forecasted values menu.

If the input data is correct, an option to forecast a specific timestamp will be available, where the user is asked to insert the desired date. Fig. 7 represents the date chooser menu.

After the date is chosen, the user can now forecast the value of the electricity consumption. A new window pops-up showing the results of the forecasting. Fig. 8 shows the forecasted values menu.

3. Case study

3.1. Dataset description

Real monitored data from an office building located in the campus of ISEP/GECAD is used. This building is occupied in daily basis by about 30 people. These data concern not only the energy consumption but also other features such as the temperature, luminosity and humidity. Samples of these data sets are publicly available at [24]. This data is also available internally through a SQL-server based database that maintains and manages the information gathered by 5 energy analyzers. Each of these analyzers monitors and the electricity consumption from different devices, electrical sockets, lighting and HVAC, in a 10 s time interval. A java-based application has been developed to collect the data from the SQL server and calculate the average of the total electricity consumption of the building N – ISEP/GECAD for each hour. This application also builds the inputs necessary for feeding the forecasting models, molding the data according to the input format that is required.

The considered forecasting models have been implemented considering the multiple training strategies presented in Table 2. These strategies combine the features extracted from the 10 days prior to the target hour. A total of 15 features has been generated, namely the electrical consumption of the 3 previous hours, the hour of the day, the month of the year, the day of the Month, the year, the day of the week, the environmental temperature of the hour (°C), the environmental temperature from the previous 3 h, the consumption at the same hour from the 2 previous weeks, the environmental and the humidity of the hour. Table 1 presents the features used for training and testing of the building N consumption dataset. These features have been identified and selected based on the availability of data and a-priori knowledge of the problem. It is, therefore intended to assess and compare the performance of the forecasting models when using different combinations of these features, as incorporated by the training strategies presented in Table 2. In turn, these strategies are no more than different combinations of data sources used in the training process for the forecasting of the electricity consumption.

Table 1
Generated Features.

Feature Description	Nomenclature
Consumption of the 3 previous hours	$Z_{t-1} to Z_{t-3}$
Hour of the day	H_t
Month of the year	M_t
Day of the Month	D_t
Year	Y_t
Day of the Week	DoW_t
Environmental temperature of the hour (°C)	T_t
Environmental temperature from the previous 3 h	$T_{t-1} to T_{t-3}$
Consumption at the same hour from the 2 previous weeks	$Z_{t-168} and Z_{t-336}$
Environmental Humidity of the hour	Hu_t

Table 2
Training Strategies.

Strategy #	Z ~ * (Consumption over ...)
1	$Z \sim D_t * M_t * H_t * Y_t * DoW_t$
2	$Z \sim D_t * M_t * H_t * Y_t * DoW_t * T_t$
3	$Z \sim D_t * M_t * H_t * Y_t * DoW_t * T_t * Z_{t-1} to Z_{t-3}$
4	$Z \sim D_t * M_t * H_t * Y_t * DoW_t * T_t * Z_{t-1} to Z_{t-3} * T_{t-1} to T_{t-3}$
5	$Z \sim D_t * M_t * H_t * Y_t * DoW_t * T_t * Z_{t-168} and Z_{t-336}$
6	$Z \sim D_t * M_t * H_t * Y_t * DoW_t * T_t * Z_{t-1} to Z_{t-3} * T_{t-1} to T_{t-3} * Z_{t-168} and Z_{t-336}$
7	$Z \sim D_t * M_t * H_t * Y_t * DoW_t * T_t * Hu_t$

Fig. 9 shows the domain model with the relations of all the main concepts of this implementation. This model identifies the various entities of the project and presents the relations between them.

This domain model presents all the concepts and the relations of the application. Every time a value is forecasted, a forecast process should be created. Which includes all the needed information to forecast the final value. This object has a date which represents the date and the hour of the final result. The Input is characterized by three tables, namely: train input, train output and test input. These data tables include the needed historical data to train the methods.

3.2. Results

The proposed models are assessed considering the forecasting of the electricity consumption of the target office building from 00:00 until 23:00 of the date 05/04/2018. The entire data set is used for training according ot the different training strategies. The accuracy of the forecasts is measured using MAPE as comparison measure for forecasting errors.

The combination of the generated features in Table 1 resulted in the creation of seven training strategies, which are presented in Table 2.

The strategies presented in Table 2 have been designed to test the performance of different combinations of variables. The MAPE error results for each ensemble method and each strategy are presented in Fig. 10.

When comparing the use of features, it can be seen from Fig. 10 that training strategy 3 presents the best performance in all 3 considered methods. It can also be concluded that the use of environmental temperature in the training process enables improving the energy consumption forecasting process, as this is evident in all the scenarios. However, the lagged features that concern the environmental temperature produce a higher error. Additionally, the use of lagged features of past consumption enables reaching better results. This is also noticeable for environmental humidity.

In order to compare the achieved results with other benchmark reference models, many reference forecasting models could be considered such as e.g. Ridge, LASSO or Linear Regression

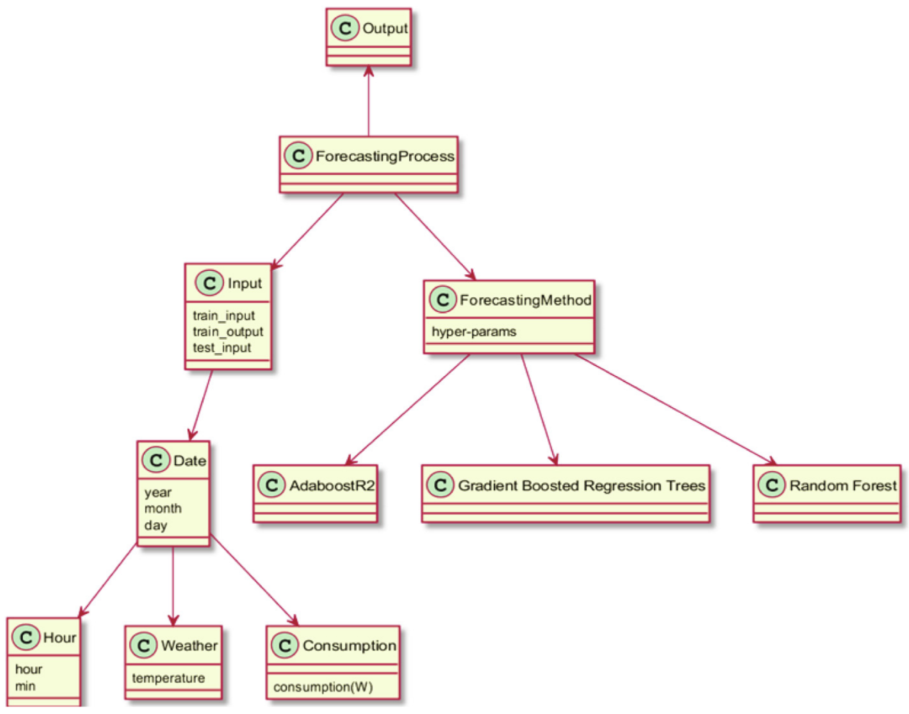
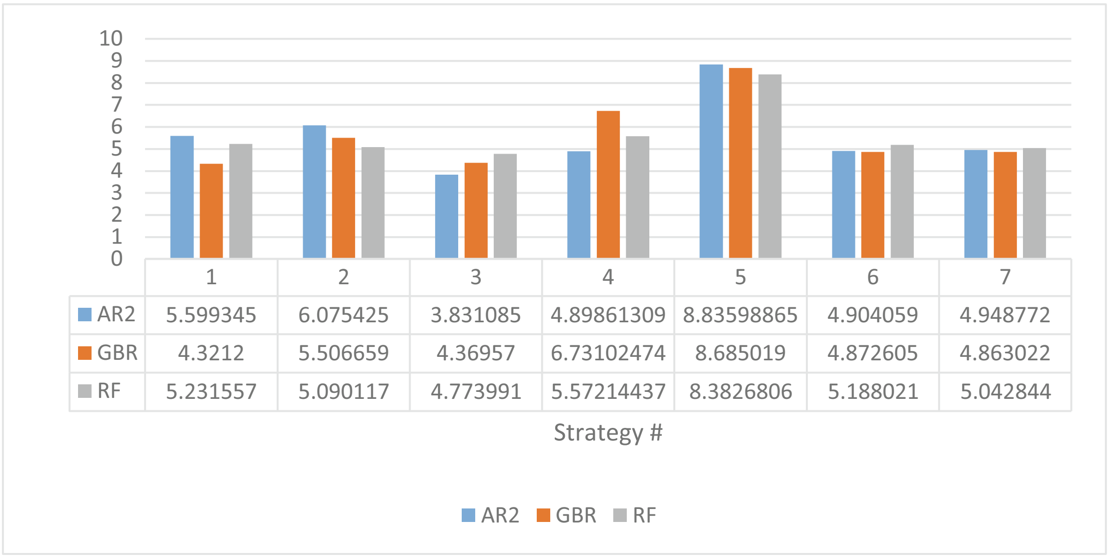


Fig. 9. Domain Model.



	1	2	3	4	5	6	7
AR2	5.599345	6.075425	3.831085	4.89861309	8.83598865	4.904059	4.948772
GBR	4.3212	5.506659	4.36957	6.73102474	8.685019	4.872605	4.863022
RF	5.231557	5.090117	4.773991	5.57214437	8.3826806	5.188021	5.042844

Fig. 10. Average forecasting errors of the AR2, GBR and RF.

[25,26]; however, for consistency reasons we are comparing the achieved results to those achieved by [27]. The work from [27] addresses the electricity consumption forecast considering the same building as this study (using the same data set), thus enabling reaching suitable comparison of results performance. The work from [27] considers several models, including fuzzy rule-based methods namely HyFIS and WM, and also SVM. In order to compare the results of the methods presented in this study to those presented in [27] the models are trained using the third training strategy, as it is the one reaching the best forecasting results, as shown by Fig. 10, and the 24 h of 10/04/2018 are forecasted. The comparison between the methods is shown in Fig. 11 and Table 3.

Fig. 11 and Table 3 show that AR2 is able to reach lower forecasting errors than the reference models, using the considered data for the specified target forecasting periods.

Table 3
Average forecasting methods errors.

	AR2	RF	GBR	SVM	HyFIS	WM
MAPE %	5,34	6,11	6,07	5,82	7,88	7,92

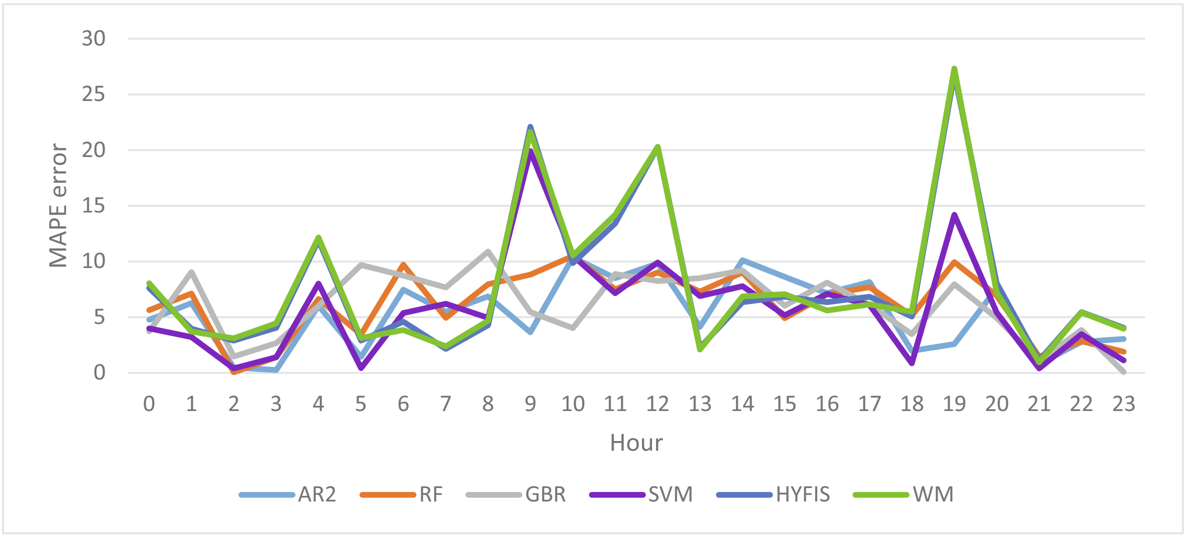


Fig. 11. Comparison between the results of AR2, RF and GBR and the results in [27].

4. Conclusions

Power and energy systems are a complex and dynamic domain, requiring fast responses and dealing with multiple sources of data. Decision-making in this scope, is, therefore, a challenging task.

This paper presents and compares several ensemble learning methodologies to perform hour-ahead forecast of the electricity consumption of an office building. The historical data from previous days and hours is used by the learning models to reach a prediction of the electricity consumption in the upcoming hours.

A case study based on real data form an office building, namely build N from GECAD/ISEP campus, has been presented. Results show that the performance of the adapted Adaboost model is superior to the benchmark forecasting models that have been considered, such as support vector regression and several fuzzy rule-based methods. In fact, the three considered ensemble learning models forecast more accurate values and with lower standard deviation. The exception is the support vector regression, which provides lower forecasting errors than RF and GBR. Considering environmental information such as temperature and humidity has shown to be advantageous. Considering lagged features concerning past hours has also shown to provide improved results.

These conclusions are especially relevant for works related to load forecasting, as the proposed models show to be adequate for dealing with the particular characteristics of this problem. Moreover, the achieved results enable identifying suitable forecasting models for the load forecasting problem, as well as assessing the advantage of using several alternative and complementary features in the learning process, referring to multiple time horizons.

Adding additional exogenous information in the forecasting models is suggested as future work. Some examples are the direct solar irradiation and thermal sensation. Such variables may prove to be helpful in improving the forecast accuracy. Additionally, further models, such as Ridge, LASSO or Linear Regression are to be experimented in future work.

CRediT authorship contribution statement

Tiago Pinto: Conceptualization, Methodology, Writing - original draft, Writing - review & editing. **Isabel Praça:** Investigation, Supervision, Validation, Writing - review & editing. **Zita Vale:** Investigation, Supervision, Validation, Writing - review & editing. **Jose Silva:** Software, Data curation, Writing - original draft.

Acknowledgement of funding sources

This work has been developed under the SPET project - PTDC/EEI-EEE/29165/2017 and has received funding from UID/EEA/00760/2019, funded by FEDER Funds through COMPETE and by National Funds through FCT.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

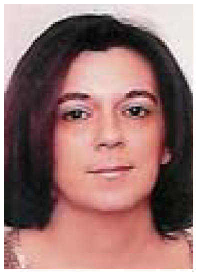
References

[1] X. Zhang, J. Wang, K. Zhang, Short-term electric load forecasting based on singular spectrum analysis and support vector machine optimized by Cuckoo search algorithm, Electr. Power Syst. Res. 146 (2017) 270–285.
[2] M.Q. Raza, A. Khosravi, A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings, Renew. Sustain. Energy Rev. 50 (2015) 1352–1372.
[3] A.Y. Saber, A.K.M.R. Alam, Short term load forecasting using multiple linear regression for big data, in: Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI); 2017; pp. 1–6.
[4] T. Pinto, T.M. Sousa, Z. Vale, Dynamic artificial neural network for electricity market prices forecast, Intell. Eng. Syst. (INES), 2012 IEEE 16th Int. Conf. 2012, 311–316.
[5] T. Pinto, T.M. Sousa, I. Praça, Z. Vale, H. Morais, Support Vector Machines for decision support in electricity markets' strategic bidding, Neurocomputing 172 (2016) 438–445.
[6] T. Ahmad, H. Chen, Nonlinear autoregressive and random forest approaches to forecasting electricity load for utility energy management systems, Sustain. Cities Soc. 45 (2019) 460–473.
[7] S. Touzani, J. Granderson, S. Fernandes, Gradient boosting machine for modeling the energy consumption of commercial buildings, Energy Build. 158 (2018) 1533–1543.
[8] S. Mao, J.-W. Chen, L. Jiao, S. Gou, R. Wang, Maximizing diversity by transformed ensemble learning, Appl. Soft Comput. 82 (2019) 105580.
[9] H. Liu, L. Zhang, Advancing Ensemble Learning Performance through data transformation and classifiers fusion in granular computing context, Expert Syst. Appl. 131 (2019) 20–29.
[10] T.K. Ho, Random Decision Forests. In Proceedings of the Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) – Volume 1; IEEE Computer Society: Washington, DC, USA, 1995; p. 278.

[11] G. Louppe, P. Geurts, Ensembles on random patches, in: Proceedings of the ECML/PKDD, 2012.
[12] X. Wu, J. He, T. Yip, J. Lu, N. Lu, A two-stage random forest method for short-term load forecasting, in: Proceedings of the 2016 IEEE Power and Energy Society General Meeting (PESGM), 2016, pp. 1–5.
[13] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Stat. 29 (2001) 1189–1232.
[14] G.J. Osório, J.C.O. Matias, J.P.S. Catalão, Short-term wind power forecasting using adaptive neuro-fuzzy inference system combined with evolutionary particle swarm optimization, wavelet transform and mutual information, Renew. Energy 75 (2015) 301–307.
[15] J. Gou, F. Hou, W. Chen, C. Wang, W. Luo, Improving Wang-Mendel method performance in fuzzy rules generation using the fuzzy C-means clustering algorithm, Neurocomputing 151 (2015) 1293–1304.
[16] P. Du, J. Wang, W. Yang, T. Niu, Multi-step ahead forecasting in electrical power system using a hybrid forecasting system, Renew. Energy 122 (2018) 533–550.
[17] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1997) 119–139.
[18] Y. Freund, R.E. Schapire, A Short Introduction to Boosting, in: Proceedings of the In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence; Morgan Kaufmann, 1999, pp. 1401–1406.
[19] J.H. Friedman, Stochastic gradient boosting, Comput. Stat. Data Anal. 38 (2002) 367–378.
[20] A. Natekin, A. Knoll, Gradient boosting machines, a tutorial, Front. Neurorobot. 7 (2013) 21.
[21] H. Drucker, Improving regressors using boosting techniques, in: Proceedings of the Proceedings of the Fourteenth International Conference on Machine Learning; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1997, pp. 107–115.
[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine Learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[23] B. Mac Namee, P. Cunningham, S. Byrne, O.I. Corrigan, The problem of bias in training data in regression problems in medical decision support, Artif. Intell. Med. 24 (2002) 51–70.
[24] Working Group on Intelligent Data Mining and Analysis (IDMA), I. Open Data Sets Available online: https://site.ieee.org/pes-iss/data-sets/ (accessed on Jan 28, 2020).
[25] S. Humeau, T.K. Wijaya, M. Vasirani, K. Aberer, Electricity load forecasting for residential customers: Exploiting aggregation and correlation between households, in: Proceedings of the 2013 Sustainable Internet and ICT for Sustainability (SustainIT), 2013, pp. 1–6.
[26] A. González-Briones, G. Hernández, J.M. Corchado, S. Omatu, M.S. Mohamad, Machine learning models for electricity consumption forecasting: a review, in: Proceedings of the 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), 2019, pp. 1–6.
[27] A. Jozi, T. Pinto, I. Praça, Z. Vale, Day-ahead forecasting approach for energy consumption of an office building using support vector machines, in: Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), 2018, pp. 1620–1625.

Tiago Pinto, Polytechnic of Porto, Portugal: Dr. Tiago Pinto received the BSc degree in 2008 and the MSc in Knowledge-based and Decision Support Technologies in 2011, both from the Polytechnic Institute of Porto (ISEP/IPP), Portugal, and the PhD in 2016 from the University of Trás-os-Montes e Alto Douro. He is also a Researcher at University of Salamanca, with the BISITE research group. His research interests include multi-agent simulation, machine learning, automated negotiation, decision support systems, smart grids and electricity markets.

Isabel Praça, Polytechnic of Porto, Portugal: Isabel Praça is a Professor and a Researcher at GECAD research group of the School of Engineering, at Polytechnic Institute of Porto. Her research areas include Multi-Agent Systems, Simulation, Decision Support Systems and Electricity Markets. She has a PhD in Electrical Engineering from the University of Trás-os-Montes e Alto Douro.



Zita Vale, Polytechnic of Porto, Portugal: Dr. Zita Vale (S'86–M'93–SM'10) is the director of the Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development (GECAD) and a professor at the Polytechnic Institute of Porto. Her main research interests concern Artificial Intelligence applications to power system operation and control, electricity markets and distributed generation. She received her diploma in Electrical Engineering in 1986 and her PhD in 1993, both from University of Porto.

José Silva, Polytechnic of Porto, Portugal: José Silva is a graduate student in the Knowledge Engineering and Decision Support Research Group (GECAD) of the School of Engineering, at Polytechnic Institute of Porto. His research interests include multi-agent simulation, electricity markets and learning techniques. He has a bachelor in Informatics from Polytechnic institute of Porto.